TRW

47467-H008-UX-00

HOST COMPUTER SOFTWARE SPECIFICATIONS FOR A ZERO-G PAYLOAD MANHANDLING SIMULATOR

10 NOVEMBER 1986

PREPARED FOR

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

CONTRACT NAS9-17554

(NASA-CR-180191) HOST COMPUTER SOFTWARE SPECIFICATIONS FOR A ZERO-g PAYLOAD MANHANDLING SIMULATOR (TRW, Inc., Houston, Tex.) 239 p CSCL 14B

N87-18586

Unclas
G3/14 43556

SYSTEM DEVELOPMENT DIVISION
TRW
DEFENSE SYSTEMS GROUP
HOUSTON, TEXAS

47467-H008-UX-00

HOST COMPUTER SOFTWARE SPECIFICATIONS

FOR A

ZERO-G PAYLOAD MANHANDLING SIMULATOR

10 November 1986

Prepared for

National Aeronautics and Space Administration

Lyndon B. Johnson Space Center

Contract NAS9-17554

Prepared by S. W. Wilson

S. W. Wilson
Staff Engineer
Systems Engineering and
Analysis Department

Approved by D K Phillips

D. K. Phillips
Manager
Systems Engineering and
Analysis Department

System Development Division

TRW
Defense Systems Group
Houston, Texas

FOREWORD

The HP PASCAL source code contained in Section 1 was developed for the Mission Planning and Analysis Division (MPAD) of NASA/JSC, and takes the place of detailed flow charts defining the host computer software specifications for MANHANDLE, a digital/graphical simulator that can be used to analyze the dynamics of onorbit (zero-g) payload manhandling operations. Input and output data for representative test cases are contained in Section 2.

MANHANDLE makes use of the utility software modules UTILMATH, UTILSPIF, and UTILVEMQ, whose specifications are defined in Reference 1. MANHANDLE can be operated in a standalone mode on the host computer to produce digital output only, or it can be interfaced with the 'bldmilmu' and 'runmilmu' programs in the MPAD IMI graphics system to produce runtime CRT images of simulated views from selectable "camera" locations. In addition, hardcopy plots of selected data outputs may be produced in the postprocessing mode if an HP-9872 plotter is connected to the host computer.

Although another language may be desirable for the ultimate implementation of the software, HP PASCAL (References 2 and 3) was chosen as the medium for developing, testing, and documenting the software specifications because, of all the options currently available on MPAD's mainline (HP-9000) computers, it was judged to be best suited to that purpose in terms of understandability, reliability, and maintainability. Most of the source code representing the specifications for the MANHANDLE host computer program software is common to both the Pascal 3.0 and the HP-UX 5.0 (Unix) operating systems, and to all models of the HP-9000 family of computers. The code that is dependent on the computer model and operating system is isolated in a small driver program (Section 1.1) and a system/PASCAL interface module (Section 1.5).

The predecessor of MANHANDLE was the PLGRAB simulator which was implemented on an HP-9000 Series 200 host computer by TRW for MPAD in support of the Westar/Palapa retrieval mission (STS 51-A). PLGRAB, in conjunction with a full-scale physical model (in both the geometrical and the inertial sense) of the Westar/Palapa satellites, was used to verify the feasibilty of a backup procedure that involved one of the EVA crewmen capturing the payloads manually while mounted on the Remote Manipulator System (RMS) by means of the Manipulator Foot Restraint (MFR). As it turned out, although manual capture did not become necessary, passive manhandling was required during STS 51-A when a specially-built RMS fixture failed to fit the satellites.

The Westar and Palapa satellites had comparatively small masses and moments of inertia, and the magnitudes of the forces and torques needed to capture and maneuver them were so small that RMS flexure and motion of the Orbiter due to reaction were considered to be negligible. Therefore only 6 degrees of freedom were accounted for in PLGRAB (just one more than the physical model, which was afforded 5 degrees of freedom by a gimbal suspension system on an air-bearing platform). By way of contrast, the Syncom repair mission (STS 51-I) not only involved manhandling as the primary mode for capturing and redeploying the payload, but the forces and torques that would have to be applied to the much heavier satellite were expected to be greater by an order of magnitude. Accordingly, the PLGRAB software was modified to provide additional modes of simulated manual control and to account for 9 more degrees of freedom: 6 for the Orbiter, plus 3 degrees of translational freedom for the EVA crewman due to RMS flexure. The RMS flexure model was a very simple one that utilized 3x3 matrices to represent the spring and damping forces in the RMS, and the effective inertia of the RMS/crewman system. These matrices were derived empirically from data (Reference 4) produced by the MPAD Payload Deployment and Retrieval System Simulator (PDRSS). Some of the data produced by the reconfigured simulator, designated MANHANDLE after its modification, are summarized in Reference 5.

Both PLGRAB and the original version of MANHANDLE were implemented hurriedly to meet flight schedules, making use of pre-existing code from the MPAD/TRW Man-in-the-Loop MMU (MILMU) simulator. As a result, the software left much to be desired in terms of orderliness and understandability, and much of it was specifically suited only to HP-9000 Series 200 computers using the Pascal 2.0 or 3.0 operating system. The code in this document has been extensively revised and rearranged to generalize it and make it easier to understand and maintain, but in functional terms it remains essentially the same program that was used to support the planning for STS 51-I. By its very nature, the manhandling of any particular payload is highly dependent on its specific configuration and its anticipated or required state of motion. Therefore it is very likely that future utilization of the MANHANDLE simulator will require some modification of the current software, particularly with regard to the control laws defined in the flight control module (Section 1.7). With this in mind, an effort has been made to make the current specifications, so far as it is possible and practicable at the present time, readily adaptable to future Orbiter and Space Station operations.

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

REFERENCES

1. S. W. Wilson, "Orbital Flight Simulation Utility Software Unit Specifications (Revision 1)," TRW Report No. 47467-H005-UX-01, 30 September 1986.

2. Hewlett-Packard Company, "HP Pascal Language Reference for the HP 9000 Series 200 Computers," Manual Part No. 98615-90050, February 1984.

3. Hewlett-Packard Company, "Pascal/9000 Language Reference Manual for the HP 9000 Series 500 Computers," Manual Part No. 97082-90001, May 1985.

4. E. J. Wolfer and R. T. Theobald, "SYNCOM Repair Mission (STS 51-I)," MDTSCO Transmittal Memo No. 1.2-TM-DF85055, 24 June 1985.

5. S. W. Wilson, "SYNCOM F3 Dynamics," TRW Letter No. 85:W482.1-105, 15 August 1985.

1. HOST COMPUTER PROGRAM SPECIFICATIONS

1.1. HP-9000 PASCAL Driver Program

1.1.1. Model 216 / Pascal 3.0 Driver

```
{ begin File 'MANHMAIN.TEXT' }

{ Payload Manhandling Simulator for HP-9000 Model 216 with Pascal 3.0 Op Sys }

$ search 'UTILUNIT' $
$ Sysprog On        $
$ Switch_strpos     $
$ Heap_dispose On   $
$ Ref 50            $

program MANHANDLE ( input, output ) ;

                    { Subject :  Simulation of Zero-G Payload Manhandling }

                    { NASA/JSC/MPAD/TRW :  Sam Wilson }

{

Using the Runge-Kutta 2nd order numerical method, MANHANDLE integrates the
differential equations of motion for a system of three bodies (each modeled
as if it were perfectly rigid in itself) --- consisting of the STS Orbiter, a
Payload, and an EVA Crewman attached to the Orbiter via the RMS/MFR or some
other flexible linkage --- having 15 degrees of freedom (three degrees in
translation and three degrees in rotation for all except the Crewman, who is
assumed to have three degrees of translational freedom only, which arises
from flexure of his linkage with the Orbiter).  Aside from damping and spring
forces arising in the flexible linkage, the only forces and torques accounted
for are those applied by the Crewman at and about a fixed point (handle) on
the Payload.

The numerical integration process may run slower or faster than real time,
depending on the stepsize specified by the user, which must be consistent
with the natural frequencies of the system if accurate results are to be
obtained.  In the COMPLETE, repeat COMPLETE, absence of spring and damping
forces in the flexible linkage between the Crewman and the Orbiter, good
results may be obtained with a stepsize as great as 1/30 of the Payload's
period of rotation.  In the presence of flexure forces, the integration
process usually will diverge if the stepsize is much greater than one tenth
of the natural period of the flexible linkage, which is typically about two
seconds in the case of the RMS/MFR system (but this may vary with the RMS
configuration, i.e., the joint angles).

During numerical integration, the state of the system is logged for future
use at the midpoint and the end of each integration step.  When numerical
integration is terminated by the user, a postprocessor gives an option to
"play back" the simulation at any desired sim/real time ratio.  Other
postprocessing options include the generation of hardcopy (ink) plots of the
histories of RMS flexure and control force/torque applied to the Payload by
the Crewman, and an attitude trajectory for the Payload spinaxis.

}
```

$ page $
{

Aside from the hardcopy plots mentioned in the preceding paragraph, MANHANDLE output includes the following:

(1) Periodically updated images of the three-body system --- generated by the MPAD IMI graphics system --- as seen from various "cameras" that may be selected by the user during runtime (i.e., while numerical integration is in progress, or during a playback).

(2) A periodically updated digital summary of numerical integration or playback progress, displayed on the CRT terminal screen of the host computer (i.e., not the IMI screen).

(3) A text file of digital output stored in a disk file of the host computer system, which contains a complete description of the simulated three-body system state at user-selected breakpoints in the numerical integration process, along with optional dumps of abbbreviated listings of data logged between breakpoints. At the user's option, this file may be dumped to a hardcopy printer after MANHANDLE execution is terminated.

(4) CRT images of the data described in item (3), generated on the terminal screen of the host computer concurrently with their storage on disk.

Use of the IMI graphics system and the hardcopy plotter is optional; the MANHANDLE simulator can be operated in a standalone mode to generate digital data (only), if so desired. Generation of hardcopy plots requires the accessibility of an HP-9872 plotter at bus address #5 on an HP-IB (IEEE-488) data bus having #7 as its select code. Use of the IMI graphics system requires its accessibility at address #17 on the same HP-IB bus. IMI usage is also dependent on the availability, in the IMI system, of the programs 'bldmilmu' and 'runmilmu', which were developed by LINCOM to support the MPAD/TRW MILMU simulator. The data files 'orbiter' and 'man0' (which describe the geometry of the Orbiter and the Crewman) must also be present in the IMI system, along with a data file describing the appropriate payload geometry.

}

```
$ page $
{

The bulk of the MANHANDLE source code is contained in the files referenced by
the series of "include" directives to the PASCAL compiler that follow this
section of comments.  The first two files in this series contain no code at
all, but instead consist of certain detailed technical data in the form of
further comments.  The first of these, 'Manhindx.I', contains an alpahbetical
index of all the global identifiers (names of constants, variables, data
types, functions, and procedures) used in the MANHANDLE program, describing
the nature of the identified item and the name of the module in which it is
declared.  Local identifiers (those declared in the "implement" section of a
module) are not included in this list.  The second file, 'Manhsymb.I',
contains a description of the system of notation used in the assignment of
names to the constants and variables used in MANHANDLE.  A careful reading of
the contents of this file at the outset will make it much easier to
understand the PASCAL source code in the remainder of the "include" files.

One of the source code files, 'Manhfcon.I', deserves special attention.  It
contains the source code for the MANHFCON module, which defines the control
laws for the computation of the forces and torques applied by the Crewman to
the "handle" of the payload, according to the mode of control selected by the
user.  The validity of any conclusions drawn from MANHANDLE simulation
results --- about onorbit manhandling feasibility for any particular payload
--- depend ultimately on how well the various modes and their associated
control laws conform to the realities of the anticipated operation.  Not the
least of these realities are the ideas of the particular crewman who is to
perform the operation, about what can be observed or sensed (e.g., from
tactile stimuli) about the motion state of the payload, and what constitutes
a natural and appropriate response on his part to various perceived
conditions. The modes and control laws currently contained in the MANHFCON
module were developed in support of the Westar/Palapa retrieval and the
Syncom repair missions, on the basis of preflight consultations with the
individual astronauts who were involved.  Their postflight comments were
favorable with regard to the validity and usefulness of the MANHANDLE
simulation results in predicting the controllability of those payloads, but
that does not mean necessarily that the existing logic in the MANHFCON module
should be applied, unmodified, to future manhandling operations.

}

$ include 'Manhindx.I.' $
$ include 'Manhsymb.I.' $
$ include 'Manhmisc.I.' $
$ include 'MANHSPIF.I.' $
$ include 'Manhbods.I.' $
$ include 'Manhfcon.I.' $
$ include 'Manhdump.I.' $
$ include 'Manhslog.I.' $
$ include 'Manhkams.I.' $
$ include 'Manhdisp.I.' $
$ include 'Manhscon.I.' $
$ include 'Manhprnt.I.' $
$ include 'Manhinit.I.' $
$ include 'Manhedit.I.' $
$ include 'Manhpost.I.' $
```

```
$ page $

import

     Sysglobals     ,  { Pascal 3.0 system module }
     Iodeclarations ,  { Pascal 3.0 system module }
     UTILMATH       ,
     UTILSPIF       ,
     MANHMISC       ,
     MANHSPIF       ,
     MANHDUMP       ,
     MANHSLOG       ,
     MANHKAMS       ,
     MANHDISP       ,
     MANHSCON       ,
     MANHPRNT       ,
     MANHINIT       ,
     MANHEDIT       ,
     MANHPOST       ;

var

     POSTPROC : boolean ;

begin  { program MANHANDLE }
INITIALIZE_IO                                                                        ;
Unitable^[31]        := Unitable^[6]                                                 ;
Unitable^[31].Uvid  := 'HP9872A'                        { hardcopy plotter     } ;
Unitable^[31].Devid :=   120000                         { (millisec) timeout   } ;
Unitable^[31].Sc    :=        7                         { HPIB select code     } ;
Unitable^[31].Ba    :=        5                         { HPIB bus address     } ;
try                                                 { set outer trap for error }
     rewrite ( LP , 'MANHPRNT.R' )                                                   ;
     USING_PLOTTER := USER_DECIDES_TO( 'Use HP-9872 plotter     ' )                  ;
     if USING_PLOTTER then rewrite ( PLT, 'HP9872A:' )                               ;
     USING_IMI     := USER_DECIDES_TO( 'Use IMI graphics system' )                   ;
     if USING_IMI then
          begin
          SET_UP_HPIB_INTERFACE_TO_IMI                                               ;
          PLIDENT := RJWORD_INPUT('Name of IMI geometry file for Payload',
                                  'sncm',8,8)                                        ;
          end ;
     new( SLOG )                                                                     ;
     DIGSHOWTINC := ONE                                                              ;
     SIMPERREAL  := ONE                                                              ;
     STROKE := 1                                                                     ;
     PLREV  := 0                                                                     ;
     REMOFFSET := 300                                                                ;
     REMHORFOV :=   9                                                                ;
     current_kamera := AHEAD                                                         ;
     DESIRED_KAMERA := current_kamera                                                ;
     OLDNAME := ''                                                                   ;
     NEWNAME := ''                                                                   ;
```

```
$ page $

     DESTINC := 30.0L0                                                          ;
     NOMSTEP :=  0.1L0                                                          ;
     repeat
          EDIT_INPUT_DATA_FILES                                                 ;
          INITIALIZE_SYSTEM_STATE                                               ;
          repeat
               DESTINC := FIXED_INPUT( 'Sim time increment   (sec)',
                                        DESTINC,9,3 )                           ;
               if DESTINC > ZERO then
                    begin
                    repeat
                         NOMSTEP := FIXED_INPUT( 'Integration stepsize (sec)',
                                                  NOMSTEP,9,3 )                 ;
                         if NOMSTEP <= ZERO then SOUND_ALARM                    ;
                         until NOMSTEP > ZERO ;
                    try                          { set inner trap for error }
                         PROPAGATE_SYSTEM_STATE                                 ;
                         DOCUMENT_THE_STATE_OF_THE_SYSTEM                       ;
                         if (curslogrec+2) > MAXSLOGRECS then
                             begin
                             SOUND_ALARM                                        ;
                             SHOWLN  (      '*** Simulation log saturated' )    ;
                             writeln ( LP, '*** Simulation log saturated' )     ;
                             DESTINC := ZERO                                    ;
                             end ;

                         recover   { come here only if inner trap is sprung }
                         begin
                         DESTINC := ZERO                                        ;
                         curslogrec := curslogrec - 1                           ;
                         WRITE_ERROR_MESSAGE                                    ;
                         if USER_DECIDES_TO( 'Dump debug data' ) then
                              begin
                              DUMP_EVERYTHING                                   ;
                              SHOWLN ( 'Debug data dumped to print file' )      ;
                              end ;
                         end ;

                    end ;
               until DESTINC <= ZERO ;
          writeln ( LP, '*** END OF SIMULATION ***' )                           ;
          writeln ( LP, SLOG^[0].SIMDESCRIP )                                   ;
          writeln ( LP, SLOG^[0].PROGSESSID )                                   ;
          START_NEW_PAGE                                                        ;
          if curslogrec > 1 then
               begin
               POSTPROC := USER_DECIDES_TO( 'Execute postprocessor' )           ;
               if not POSTPROC then   { make sure it is not just a mistake }
                    POSTPROC := USER_DECIDES_NOT_TO( 'Abandon logged data' )    ;
               if POSTPROC then
                    repeat
                         EXECUTE_POSTPROCESSOR                                  ;
                         until USER_DECIDES_NOT_TO( 'Re-run postprocessor' )    ;
               end ;
          until USER_DECIDES_NOT_TO( 'Restart simulation' ) ;
```

```
$ page $

     recover                              { come here only if outer trap is sprung }
     WRITE_ERROR_MESSAGE                                                              ;

close  ( LP, 'SAVE' )                                                                 ;
SHOWLN ( 'Printer output has been saved in text file ''MANHPRNT.R''' )                ;
CLEAN_UP_IO                                                                           ;
end .  { program MANHANDLE & File 'MANHMAIN.TEXT' }
```

1.1.2. Series 500 / HP-UX 5.0 Driver

```
{ begin File 'manhmain.p' }

{ Payload Manhandling Simulator for HP-9000 Series 500 with HP-UX 5.0 Op Sys }

$ search 'utilunit.o'       $
$ standard_level 'hp_modcal' $

program MANHANDLE ( input, output ) ;

                    { Subject :  Simulation of Zero-G Payload Manhandling }

                    { NASA/JSC/MPAD/TRW :  Sam Wilson }

{

Using the Runge-Kutta 2nd order numerical method, MANHANDLE integrates the
differential equations of motion for a system of three bodies (each modeled
as if it were perfectly rigid in itself) --- consisting of the STS Orbiter, a
Payload, and an EVA Crewman attached to the Orbiter via the RMS/MFR or some
other flexible linkage --- having 15 degrees of freedom (three degrees in
translation and three degrees in rotation for all except the Crewman, who is
assumed to have three degrees of translational freedom only, which arises
from flexure of his linkage with the Orbiter).  Aside from damping and spring
forces arising in the flexible linkage, the only forces and torques accounted
for are those applied by the Crewman at and about a fixed point (handle) on
the Payload.

The numerical integration process may run slower or faster than real time,
depending on the stepsize specified by the user, which must be consistent
with the natural frequencies of the system if accurate results are to be
obtained.  In the COMPLETE, repeat COMPLETE, absence of spring and damping
forces in the flexible linkage between the Crewman and the Orbiter, good
results may be obtained with a stepsize as great as 1/30 of the Payload's
period of rotation.  In the presence of flexure forces, the integration
process usually will diverge if the stepsize is much greater than one tenth
of the natural period of the flexible linkage, which is typically about two
seconds in the case of the RMS/MFR system (but this may vary with the RMS
configuration, i.e., the joint angles).

During numerical integration, the state of the system is logged for future
use at the midpoint and the end of each integration step.  When numerical
integration is terminated by the user, a postprocessor gives an option to
"play back" the simulation at any desired sim/real time ratio.  Other
postprocessing options include the generation of hardcopy (ink) plots of the
histories of RMS flexure and control force/torque applied to the Payload by
the Crewman, and an attitude trajectory for the Payload spinaxis.

}
```

$ page $
{

Aside from the hardcopy plots mentioned in the preceding paragraph, MANHANDLE output includes the following:

(1) Periodically updated images of the three-body system --- generated by the MPAD IMI graphics system --- as seen from various "cameras" that may be selected by the user during runtime (i.e., while numerical integration is in progress, or during a playback).

(2) A periodically updated digital summary of numerical integration or playback progress, displayed on the CRT terminal screen of the host computer (i.e., not the IMI screen).

(3) A text file of digital output stored in a disk file of the host computer system, which contains a complete description of the simulated three-body system state at user-selected breakpoints in the numerical integration process, along with optional dumps of abbbreviated listings of data logged between breakpoints. At the user's option, this file may be dumped to a hardcopy printer after MANHANDLE execution is terminated.

(4) CRT images of the data described in item (3), generated on the terminal screen of the host computer concurrently with their storage on disk.

Use of the IMI graphics system and the hardcopy plotter is optional; the MANHANDLE simulator can be operated in a standalone mode to generate digital data (only), if so desired. Generation of hardcopy plots requires the accessibility of an HP-9872 plotter via the pathname '/dev/plt9872'. Use of the IMI graphics system requires its accessibility on an HP-IB (IEEE-488) data bus via the pathname '/dev/imi_hpib'. IMI usage is also dependent on the availability, in the IMI system, of the programs 'bldmilmu' and 'runmilmu', which were developed by LINCOM to support the MPAD/TRW MILMU simulator. The data files 'orbiter' and 'man0' (which describe the geometry of the Orbiter and the Crewman) must also be present in the IMI system, along with a data file describing the appropriate payload geometry.

}

```
$ page $
{

The bulk of the MANHANDLE source code is contained in the files referenced by
the series of "include" directives to the PASCAL compiler that follow this
section of comments.  The first two files in this series contain no code at
all, but instead consist of certain detailed technical data in the form of
further comments.  The first of these, 'Manhindx.I', contains an alpahbetical
index of all the global identifiers (names of constants, variables, data
types, functions, and procedures) used in the MANHANDLE program, describing
the nature of the identified item and the name of the module in which it is
declared.  Local identifiers (those declared in the "implement" section of a
module) are not included in this list.  The second file, 'Manhsymb.I',
contains a description of the system of notation used in the assignment of
names to the constants and variables used in MANHANDLE.  A careful reading of
the contents of this file at the outset will make it much easier to
understand the PASCAL source code in the remainder of the "include" files.

One of the source code files, 'Manhfcon.I', deserves special attention.  It
contains the source code for the MANHFCON module, which defines the control
laws for the computation of the forces and torques applied by the Crewman to
the "handle" of the payload, according to the mode of control selected by the
user.  The validity of any conclusions drawn from MANHANDLE simulation
results --- about onorbit manhandling feasibility for any particular payload
--- depend ultimately on how well the various modes and their associated
control laws conform to the realities of the anticipated operation.  Not the
least of these realities are the ideas of the particular crewman who is to
perform the operation, about what can be observed or sensed (e.g., from
tactile stimuli) about the motion state of the payload, and what constitutes
a natural and appropriate response on his part to various perceived
conditions. The modes and control laws currently contained in the MANHFCON
module were developed in support of the Westar/Palapa retrieval and the
Syncom repair missions, on the basis of preflight consultations with the
individual astronauts who were involved.  Their postflight comments were
favorable with regard to the validity and usefulness of the MANHANDLE
simulation results in predicting the controllability of those payloads, but
that does not mean necessarily that the existing logic in the MANHFCON module
should be applied, unmodified, to future manhandling operations.

}

$ include 'Manhindx.I' $
$ include 'Manhsymb.I' $
$ include 'Manhmisc.I' $
$ include 'manhspif.I' $
$ include 'Manhbods.I' $
$ include 'Manhfcon.I' $
$ include 'Manhdump.I' $
$ include 'Manhslog.I' $
$ include 'Manhkams.I' $
$ include 'Manhdisp.I' $
$ include 'Manhscon.I' $
$ include 'Manhprnt.I' $
$ include 'Manhinit.I' $
$ include 'Manhedit.I' $
$ include 'Manhpost.I' $
```

```
$ page $

import

     UTILMATH ,
     UTILSPIF ,
     MANHMISC ,
     MANHSPIF ,
     MANHDUMP ,
     MANHSLOG ,
     MANHKAMS ,
     MANHDISP ,
     MANHSCON ,
     MANHPRNT ,
     MANHINIT ,
     MANHEDIT ,
     MANHPOST ;

var

     POSTPROC : boolean ;


begin  { program MANHANDLE }
INITIALIZE_IO                                                                  ;
try                                                  { set outer trap for error }
     rewrite ( LP , 'manhprnt.R' )                                             ;
     USING_PLOTTER := USER_DECIDES_TO( 'Use HP-9872 plotter     ' )            ;
     if USING_PLOTTER then rewrite ( PLT, '/dev/plt9872' )                     ;
     USING_IMI     := USER_DECIDES_TO( 'Use IMI graphics system' )             ;
     if USING_IMI then
          begin
          SET_UP_HPIB_INTERFACE_TO_IMI                                         ;
          PLIDENT := RJWORD_INPUT('Name of IMI geometry file for Payload',
                                  'sncm',8,8)                                  ;
          end ;
     new( SLOG )                                                               ;
     DIGSHOWTINC := ONE                                                        ;
     SIMPERREAL  := ONE                                                        ;
     STROKE := 1                                                               ;
     PLREV  := 0                                                               ;
     REMOFFSET := 300                                                          ;
     REMHORFOV :=   9                                                          ;
     current_kamera := AHEAD                                                   ;
     DESIRED_KAMERA := current_kamera                                          ;
     OLDNAME := ''                                                             ;
     NEWNAME := ''                                                             ;
```

```
$ page $

     DESTINC := 30.0L0                                                       ;
     NOMSTEP :=  0.1L0                                                       ;
     repeat
          EDIT_INPUT_DATA_FILES                                              ;
          INITIALIZE_SYSTEM_STATE                                            ;
          repeat
               DESTINC := FIXED_INPUT( 'Sim time increment   (sec)',
                                        DESTINC,9,3 )                        ;
               if DESTINC > ZERO then
                    begin
                    repeat
                         NOMSTEP := FIXED_INPUT( 'Integration stepsize (sec)',
                                                  NOMSTEP,9,3 )              ;
                         if NOMSTEP <= ZERO then SOUND_ALARM                 ;
                         until NOMSTEP > ZERO ;
                    try                           { set inner trap for error }
                         PROPAGATE_SYSTEM_STATE                              ;
                         DOCUMENT_THE_STATE_OF_THE_SYSTEM                    ;
                         if (curslogrec+2) > MAXSLOGRECS then
                              begin
                              SOUND_ALARM                                    ;
                              SHOWLN  (      '*** Simulation log saturated' )  ;
                              writeln ( LP, '*** Simulation log saturated' )   ;
                              DESTINC := ZERO                                ;
                              end ;

                         recover   { come here only if inner trap is sprung }
                         begin
                         DESTINC := ZERO                                     ;
                         curslogrec := curslogrec - 1                        ;
                         WRITE_ERROR_MESSAGE                                 ;
                         if USER_DECIDES_TO( 'Dump debug data' ) then
                              begin
                              DUMP_EVERYTHING                                ;
                              SHOWLN ( 'Debug data dumped to print file' )   ;
                              end ;
                         end ;

                    end ;
               until DESTINC <= ZERO ;
          writeln ( LP, '*** END OF SIMULATION ***' )                        ;
          writeln ( LP, SLOG^[0].SIMDESCRIP )                                ;
          writeln ( LP, SLOG^[0].PROGSESSID )                                ;
          START_NEW_PAGE                                                     ;
          if curslogrec > 1 then
               begin
               POSTPROC := USER_DECIDES_TO( 'Execute postprocessor' )        ;
               if not POSTPROC then    { make sure it is not just a mistake }
                    POSTPROC := USER_DECIDES_NOT_TO( 'Abandon logged data' ) ;
               if POSTPROC then
                    repeat
                         EXECUTE_POSTPROCESSOR                               ;
                         until USER_DECIDES_NOT_TO( 'Re-run postprocessor' ) ;
               end ;
          until USER_DECIDES_NOT_TO( 'Restart simulation' ) ;
```

```
$ page $

     recover                        { come here only if outer trap is sprung }
     WRITE_ERROR_MESSAGE                                                          ;

close ( LP, 'SAVE' )                                                              ;
SHOWLN  ( 'Printer output has been saved in text file ''manhprnt.R''' )           ;
CLEAN_UP_IO                                                                       ;
end .  { program MANHANDLE & File 'manhmain.p' }
```

1.2. Index of Global Identifiers

```
$ page $  { begin File 'Manhindx.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

{ Comment File }   { Subject :  Index of Global Identifiers }

                     { NASA/JSC/MPAD/TRW :  Sam Wilson }


{
                                              Declaration    Parent      Source
Identifier                                     Category    Identifier    Module
---------------------------------------------  ---------   ----------   --------

ADD_STRING_TO_IMI_BUFFER                     : procedure               : MANHSPIF
AERR                                         : var                     : MANHFCON
AERRMAX                                      : rec field : CNTRLCON    : MANHFCON
AERRMIN                                      : rec field : CNTRLCON    : MANHFCON
AERRTOL                                      : rec field : CNTRLCON    : MANHFCON
AHEAD                                        : type enum : KAMERA      : MANHKAMS
ALIGN                                        : type enum : CNTRLMODE   : MANHFCON
ANGDEG                                       : function                : UTILMATH
ANGMO_P_I                                    : rec field : SLOGREC     : MANHSLOG
ARR                                          : rec field : DERIVREC    : MANHBODS
ARR                                          : rec field : STATEREC    : MANHBODS
ARR                                          : rec field : IDATREC     : MANHINIT
ATAN2                                        : function                : UTILMATH
AVEL                                         : var                     : MANHFCON
AVELTOL                                      : rec field : CNTRLCON    : MANHFCON

BEHIND                                       : type enum : KAMERA      : MANHKAMS

C                                            : rec field : CHINPUTREC  : UTILSPIF
CAPTURE                                      : type enum : CNTRLMODE   : MANHFCON
CFORC_PH_PB                                  : rec field : SLOGREC     : MANHSLOG
CFORC_H_B                                    : rec field : INFOREC     : MANHBODS
CHAR_INPUT                                   : function                : UTILSPIF
CHECHOMODE                                   : type                    : UTILSPIF
CHINPUTREC                                   : type                    : UTILSPIF
CHWAITMODE                                   : type                    : UTILSPIF
CLEAN_UP_IO                                  : procedure               : UTILSPIF
CLEAR_SCREEN                                 : procedure               : UTILSPIF
CLOCKTICK                                    : function                : UTILSPIF
CNTRLCON                                     : var                     : MANHFCON
CNTRLMODE                                    : type                    : MANHFCON
CNTRLPAC                                     : const                   : MANHFCON
CNTRLPACARR                                  : type                    : MANHFCON
COMPUTE_FLIGHT_CONTROL_FORCE_AND_TORQUE      : procedure               : MANHFCON
crewman                                      : type enum : OBJECT      : MANHBODS
CRSP                                         : function                : UTILVEMQ
CTORQ_PH_PB                                  : rec field : SLOGREC     : MANHSLOG
CTORQ_H_B                                    : rec field : INFOREC     : MANHBODS
curmode                                      : rec field : SLOGREC     : MANHSLOG
current_kamera                               : var                     : MANHKAMS
current_mode                                 : var                     : MANHFCON
curslogrec                                   : var                     : MANHSLOG
CURTINC                                      : var                     : MANHSCON}
```

```
$ page $
{
                                           Declaration   Parent      Source
Identifier                                 Category      Identifier  Module
----------------------------------------   ---------     ----------  --------

DATESTRING                                 : function                : UTILSPIF
DEGPERRAD                                  : const                   : UTILMATH
DERIV                                      : var                     : MANHBODS
DERIVREC                                   : type                    : MANHBODS
DESIRED_KAMERA                             : var                     : MANHKAMS
desired_mode                               : var                     : MANHFCON
desmode                                    : rec field : SLOGREC     : MANHSLOG
DESPIN                                     : type enum : CNTRLMODE   : MANHFCON
DESTINC                                    : var                     : MANHSCON
DIAG3X3                                    : type                    : UTILVEMQ
DOCUMENT_THE_STATE_OF_THE_SYSTEM           : procedure               : MANHPRNT
DOTP                                       : function                : UTILVEMQ
DSPFORC                                    : rec field : CNTRLCON    : MANHFCON
DUMP_DERIV_RECORDS                         : procedure               : MANHDUMP
DUMP_EVERYTHING                            : procedure               : MANHDUMP
DUMP_INFO_RECORDS                          : procedure               : MANHDUMP
DUMP_STATE_RECORDS                         : procedure               : MANHDUMP
DUMPEUL                                    : procedure               : MANHDUMP
DUMPINT                                    : procedure               : MANHDUMP
DUMPMAT                                    : procedure               : MANHDUMP
DUMPQUAT                                   : procedure               : MANHDUMP
DUMPREAL                                   : procedure               : MANHDUMP
DUMPVEC                                    : procedure               : MANHDUMP

EDIT_INPUT_DATA_FILES                      : procedure               : MANHEDIT
EULARR                                     : type                    : UTILVEMQ
EULDEG                                     : function                : UTILVEMQ
EULPYR                                     : type                    : UTILVEMQ
EULRAD                                     : function                : UTILVEMQ

FETCHLN                                    : procedure               : UTILSPIF
FILESIZE                                   : const                   : MANHINIT
FILL12                                     : const                   : MANHMISC
FIXED_INPUT                                : function                : UTILSPIF
FLEX_RO_R                                  : rec field : SLOGREC     : MANHSLOG
FLEXFORCMAG                                : rec field : SLOGREC     : MANHSLOG
FORC_CM_I                                  : rec field : INFOREC     : MANHBODS
FORC_CRO_R                                 : var                     : MANHBODS
FORCLIM                                    : rec field : CNTRLCON    : MANHFCON
FREE                                       : type enum : CNTRLMODE   : MANHFCON
FTPERIN                                    : const                   : MANHMISC

GOTWHAT                                    : type                    : UTILSPIF

HOLD                                       : type enum : CNTRLMODE   : MANHFCON
HORFOV                                     : rec field : KAMARR      : MANHKAMS}
```

$ page $
{

Identifier	Declaration Category	Parent Identifier	Source Module
IDATREC	: type		: MANHINIT
IDN3X3	: const		: UTILVEMQ
IMATQ	: function		: UTILVEMQ
IMIN	: function		: UTILMATH
INFO	: var		: MANHBODS
INFOREC	: type		: MANHBODS
INITIALIZE_IMI	: procedure		: MANHDISP
INITIALIZE_IO	: procedure		: UTILSPIF
INITIALIZE_SYSTEM_STATE	: procedure		: MANHINIT
INTEGER_INPUT	: function		: UTILSPIF
IPOS_CM_I	: rec field	: STATEREC	: MANHBODS
IPOS_OCM_I	: rec field	: SLOGREC	: MANHSLOG
IPOS_PCM_I	: rec field	: SLOGREC	: MANHSLOG
IPOS_RO_I	: rec field	: SLOGREC	: MANHSLOG
IPOSDOT_CM_I	: rec field	: DERIVREC	: MANHBODS
IRATE_I_B	: rec field	: STATEREC	: MANHBODS
IRATEDOT_I_B	: rec field	: DERIVREC	: MANHBODS
IROT	: function		: UTILVEMQ
IVEL_CM_I	: rec field	: STATEREC	: MANHBODS
IVELDOT_CM_I	: rec field	: DERIVREC	: MANHBODS
KAMARR	: type		: MANHKAMS
KAMERA	: type		: MANHKAMS
KAMINFO	: var		: MANHKAMS
LEFT	: type enum	: KAMERA	: MANHKAMS
LERR	: var		: MANHFCON
LERRLIM	: rec field	: CNTRLCON	: MANHFCON
LERRTOL	: rec field	: CNTRLCON	: MANHFCON
LINESTR	: type		: UTILSPIF
LOC_H_PB	: var		: MANHBODS
LOC_KO_I	: rec field	: KAMARR	: MANHKAMS
LOC_KO_OB	: rec field	: KAMARR	: MANHKAMS
LOC_N_OB	: rec field	: SLOGREC	: MANHSLOG
LOC_OCM_OS	: rec field	: SLOGREC	: MANHSLOG
LOC_PCM_PS	: rec field	: SLOGREC	: MANHSLOG
LOITER	: procedure		: UTILSPIF
LP	: var		: UTILSPIF
LVEL	: var		: MANHFCON
LVELTOL	: rec field	: CNTRLCON	: MANHFCON
MASS	: rec field	: INFOREC	: MANHBODS
MAT3X3	: type		: UTILVEMQ
MAXSLOGRECS	: const		: MANHSLOG
MINV	: function		: UTILVEMQ
MOVE_UP	: procedure		: UTILSPIF
MXM	: function		: UTILVEMQ}

$ page $
{

Identifier	Declaration Category	Parent Identifier	Source Module
NAMEPAC	: type		: UTILSPIF
NAMESTR	: type		: UTILSPIF
NEWNAME	: var		: MANHEDIT
NOCHECHO	: type enum	: CHECHOMODE	: UTILSPIF
NOCHWAIT	: type enum	: CHWAITMODE	: UTILSPIF
NOMAVEL	: rec field	: CNTRLCON	: MANHFCON
NOMLVEL	: rec field	: CNTRLCON	: MANHFCON
NOMSTEP	: var		: MANHSCON
NOTHING	: type enum	: GOTWHAT	: UTILSPIF
NUMOBJECTS	: const		: MANHBODS
NUMSLOGRECS	: rec field	: SLOGREC	: MANHSLOG
OBJECT	: type		: MANHBODS
OBPOS_H_OB	: var		: MANHBODS
OBPOS_N_R	: var		: MANHBODS
OBPOS_RO_R	: rec field	: STATEREC	: MANHBODS
OBPOSDOT_RO_R	: rec field	: DERIVREC	: MANHBODS
OBVEL_H_OB	: var		: MANHBODS
OBVEL_RO_R	: rec field	: STATEREC	: MANHBODS
OBVELDOT_RO_R	: rec field	: DERIVREC	: MANHBODS
OLDNAME	: var		: MANHEDIT
OLDSTEP	: var		: MANHSCON
ONE	: const		: UTILMATH
orbiter	: type enum	: OBJECT	: MANHBODS
OVER	: type enum	: KAMERA	: MANHKAMS
PAW	: type enum	: KAMERA	: MANHKAMS
payload	: type enum	: OBJECT	: MANHBODS
PERCENT	: const		: MANHMISC
PITCH	: type enum	: CNTRLMODE	: MANHFCON
PLAY_BACK_VISUAL_DATA	: procedure		: MANHDISP
PLIDENT	: var		: MANHDISP
PLOT_CONTROL_FORCE_AND_TORQUE_HISTORY	: procedure		: MANHPOST
PLOT_PAYLOAD_SPINAXIS_ATTITUDE_TRAJECTORY	: procedure		: MANHPOST
PLOT_RMS_FLEXURE_HISTORY	: procedure		: MANHPOST
PLREV	: var		: MANHPOST
PLT	: var		: MANHPOST
PROGID	: const		: MANHMISC
PROGSESSID	: rec field	: SLOGREC	: MANHSLOG
PROMPTSTR	: type		: UTILSPIF
PROPAGATE_SYSTEM_STATE	: procedure		: MANHSCON
PVID	: rec field	: IDATREC	: MANHINIT
PYR_I_OB	: rec field	: SLOGREC	: MANHSLOG
PYR_I_PB	: rec field	: SLOGREC	: MANHSLOG
PYR_I_R	: rec field	: SLOGREC	: MANHSLOG
PYRQ	: function		: UTILVEMQ}

$ page $
{

Identifier	Declaration Category	Parent Identifier	Source Module
Q	: rec field	: CHINPUTREC	: UTILSPIF
QCXQ	: function		: UTILVEMQ
QMAT	: function		: UTILVEMQ
QPYR	: function		: UTILVEMQ
QUAT_I_B	: rec field	: STATEREC	: MANHBODS
QUAT_I_K	: rec field	: KAMARR	: MANHKAMS
QUAT_OB_K	: rec field	: KAMARR	: MANHKAMS
QUAT_OB_PB	: var		: MANHBODS
QUAT_OB_PD	: var		: MANHBODS
QUAT_OB_R	: var		: MANHBODS
QUAT_R_PB	: var		: MANHBODS
QUATDOT_I_B	: rec field	: DERIVREC	: MANHBODS
QUATERNION	: type		: UTILVEMQ
QXQ	: function		: UTILVEMQ
RADPERDEG	: const		: UTILMATH
RALAC_CM_B	: rec field	: INFOREC	: MANHBODS
REALARR3	: type		: MANHFCON
REMHORFOV	: var		: MANHKAMS
REMOFFSET	: var		: MANHKAMS
RESET_IMI_BUFFER	: procedure		: MANHSPIF
RESTORE_CURSOR	: procedure		: UTILSPIF
RIGHT	: type enum	: KAMERA	: MANHKAMS
RJWORD_INPUT	: function		: UTILSPIF
RMIN	: function		: UTILMATH
RNERT_CM_B	: rec field	: INFOREC	: MANHBODS
RNERT_PCM_PB	: var		: MANHBODS
RNERT_PH_PB	: var		: MANHBODS
ROLL	: type enum	: CNTRLMODE	: MANHFCON
ROT	: function		: UTILVEMQ
RSIGN	: function		: UTILMATH}

$ page $
{

Identifier	Declaration Category	Parent Identifier	Source Module
S	: rec field	: QUATERNION	: UTILVEMQ
SAVE_SIMULATION_LOG	: procedure		: MANHSLOG
SAW	: type enum	: KAMERA	: MANHKAMS
SET_UP_HPIB_INTERFACE_TO_IMI	: procedure		: MANHSPIF
SET_UP_KAMERA_DATA	: procedure		: MANHKAMS
SHOW_PLOTTER_INITIALIZATION_MESSAGE	: procedure		: MANHPOST
SHOWLN	: procedure		: UTILSPIF
SIMDESCRIP	: rec field	: SLOGREC	: MANHSLOG
SIMPERREAL	: var		: MANHDISP
SIMTIME	: var		: MANHMISC
SLOG	: var		: MANHSLOG
SLOGARR	: type		: MANHSLOG
SLOGREC	: type		: MANHSLOG
SOMETHING	: type enum	: GOTWHAT	: UTILSPIF
SOUND_ALARM	: procedure		: UTILSPIF
SOUND_ALERT	: procedure		: UTILSPIF
SPINUP	: type enum	: CNTRLMODE	: MANHFCON
SPUFORC	: rec field	: CNTRLCON	: MANHFCON
START_NEW_PAGE	: procedure		: UTILSPIF
STATE	: var		: MANHBODS
STATEREC	: type		: MANHBODS
STATESIZE	: const		: MANHBODS
STDGRAVACC	: const		: MANHMISC
STORE_SIMULATION_LOG_RECORD	: procedure		: MANHSLOG
STROKE	: var		: MANHPOST
STRUC2BODY	: const		: MANHMISC
SUPPRESS_CURSOR	: procedure		: UTILSPIF
SXV	: function		: UTILVEMQ
TALAC_CRO_R	: rec field	: INFOREC	: MANHBODS
TDAMP_CRO_R	: rec field	: INFOREC	: MANHBODS
TICKSPERSEC	: const		: UTILSPIF
TIME	: rec field	: SLOGREC	: MANHSLOG
TIMECON	: rec field	: CNTRLCON	: MANHFCON
TNERT_PH_PB	: var		: MANHBODS
TORQ_CM_B	: rec field	: INFOREC	: MANHBODS
TORQLIM	: rec field	: CNTRLCON	: MANHFCON
TRANSFER_BUFFER_CONTENTS_TO_IMI	: procedure		: MANHSPIF
TSPRG_CRO_R	: rec field	: INFOREC	: MANHBODS
TWO	: const		: UTILMATH}

```
$ page $
{
                                         Declaration    Parent      Source
Identifier                                Category    Identifier    Module
-----------------------------------------  ---------  ----------   --------

UNDER                                    : type enum : KAMERA     : MANHKAMS
UNIQUAT                                  : function               : UTILVEMQ
UPDATE_DISPLAYS                          : procedure              : MANHDISP
UPDATE_IMI_DATA                          : procedure              : MANHDISP
USER_DECIDES                             : function               : MANHMISC
USER_DECIDES_NOT_TO                      : function               : MANHMISC
USER_DECIDES_TO                          : function               : UTILSPIF
USING_IMI                                : var                    : MANHDISP
USING_PLOTTER                            : var                    : MANHPOST

V                                        : rec field : QUATERNION : UTILVEMQ
VBASE_I_B                                : rec field : INFOREC    : MANHBODS
VBASE_OB_PB                              : var                    : MANHBODS
VBASE_OB_R                               : var                    : MANHBODS
VBASE_R_PB                               : var                    : MANHBODS
VECTOR                                   : type                   : UTILVEMQ
VDIF                                     : function               : UTILVEMQ
VMAG                                     : function               : UTILVEMQ
VSUM                                     : function               : UTILVEMQ
VXD                                      : function               : UTILVEMQ
VXM                                      : function               : UTILVEMQ
VXMT                                     : function               : UTILVEMQ

WOBBLE_CLOK_P                            : rec field : SLOGREC    : MANHSLOG
WOBBLE_CONE_P                            : rec field : SLOGREC    : MANHSLOG
WORD_INPUT                               : function               : UTILSPIF
WORDSTR                                  : type                   : UTILSPIF
WRITE_ERROR_MESSAGE                      : procedure              : MANHSPIF

YAW                                      : type enum : CNTRLMODE  : MANHFCON
YUNVEC                                   : const                  : UTILVEMQ

ZERO                                     : const                  : UTILMATH
ZERVEC                                   : const                  : UTILVEMQ
ZUNVEC                                   : const                  : UTILVEMQ}


{ end File 'Manhindx.I' }
```

1.3. <u>Symbology for Variables and Constants</u>

$ page $ { begin File 'Manhsymb.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

{ Comment File }   { Subject : Symbology for Variables & Constants }

{ NASA/JSC/MPAD/TRW : Sam Wilson }

{

ELEMENTS OF VARIABLE NAMES

Body Identification

% Token representing any one of the following body identification symbols.

C Crewman (Payload handler).

P Payload (rigid body to be handled by Crewman).

O STS Orbiter.

Coordinate System Identification

$ Token representing any one of the following coordinate-system identification symbols.

%B Body-fixed reference frame generally used for flight control. Origin is at body's center of mass (CM). X axis points forward; Y axis points to the right. The body identification symbol represented by the % token is used only when it is needed to avoid ambiguity.

%D Desired-attitude reference system. Axes define desired orientation of %B frame. Origin undefined. The body identification symbol represented by % token is used only when it is needed to avoid ambiguity.

%G Body-fixed reference frame used by the IMI graphics system to define the geometry of a particular body. Axes aligned with %B axes. For Payload, origin coincides with origin of structural (S) coordinates. For Orbiter and Crewman, origin is fixed arbitrarily. The body identification symbol represented by % token is used only when it is needed to avoid ambiguity.

I Inertial reference frame. Origin is at the initial nominal position (N) of Payload handling point (H), which is defined (in terms of Orbiter structural coordinates) by user input. Alignment of axes is fixed in inertial space, but otherwise undefined.

K Camera-fixed coordinate system. Origin at optical center of camera. X axis coincides with optical axis and points in the direction of sight; Y axis points to the right. }

$ page $
{

%P Body-fixed frame of reference used for the analysis of rotational dynamics. Origin at CM; axes aligned with principal axes of inertia. Body identification symbol represented by % token is used only when it is needed to avoid ambiguity.

R Crewman's reach-envelope coordinate system. Origin at center of reach envelope. X axis points forward; Y axis points to crewman's right.

%S Structural reference frame for a particular body. Location of the origin is arbitrary. X axis points aft; Y axis points to the right. Body identification symbol represented by % token is used only when it is needed to avoid ambiguity. These frames are unique in that locations are measured in inches, as opposed to feet in all other frames.

Force/Torque Source Ientification

Token representing any one of the following force / torque source identification symbols.

A Aerodynamics (ignored in this simulation).

C Crewman (Payload handler).

G Gravity (ignored in this simulation).

R Reaction control system jets (not modeled in this simulation).

Point Identification

@ Token representing any one of the following point identification symbols.

%CM Center of mass (origin of %B and %P coordinate systems). The body identification symbol represented by % token is used only when it is needed to avoid ambiguity.

%GO Origin of %G coordinate system. The body identification symbol represented by % token is used only when it is needed to avoid ambiguity.

H Position of Payload handling point (fixed in Payload body frame).

KO Origin of camera coordinate system K (i.e., optical center).

N Nominal position of Payload handling point relative to the structure of the Orbiter.

RO Origin of R axes (center of Crewman's reach envelope). }

$ page $
{

VARIABLE NAMES

Vectors (1x3 Matrices)

ANGMO_%_$ Angular momentum (slug*ft*ft/sec = ft*lb*sec, in the inertial frame I, about the CM) of the body whose identification symbol is represented by the % token. The $ token represents the identification symbol of the coordinate system upon whose axes the vector components are resolved. For example, ANGMO_P_OB represents the angular momentum of the Payload, expressed in terms of Orbiter body-axis components.

FLEX_@_$ Linear displacement (ft unless indicated otherwise) of a point nominally fixed in the Orbiter's body-fixed frame, due to flexure of the Remote Manipulator System (RMS) arm. The @ token represents a symbol which identifies the point whose displacement is being defined. The symbol represented by the $ token identifies the coordinate system upon whose axes the displacement is resolved.

#FORC_%@_$ Force vector (lb). The symbol represented by the # token, if present, identifies the source of the force. The symbol represented by the % token identifies the body on which the force is exerted, and is included only when it is needed to avoid ambiguity. The @ token represents a symbol identifying the point at which the force is applied (or can be thought of as being applied). The symbol represented by $ identifies the resolvent coordinate system (i.e., that upon whose axes the vector components are resolved). For example, CFORC_PH_I represents the force exerted by the Crewman on the Payload at the handling point, expressed in components parallel to the axes of the inertial frame I.

LOC_@_$ Location (i.e., fixed position) vector (inches if $ = %S, feet otherwise). The @ token represents a symbol identifying the point whose location is being defined. The $ token represents a symbol identifying the coordinate system from whose origin the location is measured and upon whose axes the vector components are resolved. For example, LOC_KO_OS defines the location of a camera fixed in the Orbiter's structural coordinate system. }

$ page $
{

$POS_@_$ — Position vector (ft). The symbol represented by the @ token identifies the point whose position is being defined. The symbol represented by the first $ token identifies the coordinate system from whose origin the position is measured, while the second $ token represents the identification symbol of the resolvent coordinate system. For example, RPOS_H_OB represents the position of the Payload handling point relative to the center of the Crewman's reach envelope, expressed in terms of vector components parallel to the Orbiter body (OB) axes.

$RATE_$_$ — Rate (i.e., angular velocity) vector (rad/sec internally, deg/sec externally). The second $ token represents the identification symbol of the frame whose angular velocity is being defined, the first $ token represents the symbol of the reference frame with respect to which the angular velocity is measured, and the third represents the symbol of the frame on whose axes the vector components are resolved. For example, OBRATE_PB_I is a vector defining the angular velocity of the Payload body frame (PB) relative to the Orbiter body frame (OB), with components resolved on the I axes. The symbol IRATE_B_B represents the usual "body rate vector" (with respect to inertial space) in a context where explicit identification of the body associated with the B frame is unnecessary.

#TORQ_%@_$ — Torque vector (ft*lb). The symbol represented by the # token, if present, identifies the source of the torque. The symbol represented by the % token identifies the body on which the torque is exerted, and is included only when it is needed to avoid ambiguity. The @ token represents a symbol identifying the point at which the torque is measured. The symbol represented by $ identifies the resolvent coordinate system (i.e., that upon whose axes the vector components are resolved). For example, CTORQ_PH_PB represents the torque exerted on the Payload at the handling point by the Crewman, expressed in terms of Payload body components.

$VEL_@_$ — Linear velocity vector (ft/sec). The symbol represented by the @ token identifies the point whose velocity is being defined. The symbol represented by the first $ token identifies the reference frame in which the velocity is measured. The second $ token represents a symbol identifying the coordinate system upon whose axes the vector components are resolved. For instance, RVEL_H_PB represents the velocity of the payload handling point with respect to the Crewman's reach axes, with components resolved on the Payload body axes. }

$ page $
{

3x3 Matrices

RALAC_%@_$ Rotational alacrity matrix (inverse of RNERT_%@_$).

RNERT_%@_$ Rotational inertia tensor (slug*ft*ft = ft*lb*sec*sec). The symbol represented by the % token identifies the rigid body whose rotational inertia is being defined. The symbol represented by the @ token identifies the point about which the inertia is being defined. The symbol represented by the $ token identifies the coordinate system axes upon which the tensor components are resolved. For example, RNERT_PH_PB represents the rotational inertia tensor of the Payload about its handling point, with components resolved on the Payload body axes.

TALAC_%@_$ Translational alacrity matrix (inverse of TNERT_%@_$).

TDAMP_%@_$ Translational damping constant matrix (lb*sec/ft) for the point whose identification symbol is represented by the @ token. When premultiplied by the velocity vector of the point in question, yields the damping force exerted on the body whose identification symbol is represented by the % token. Force and velocity vector components are resolved upon the axes of the coordinate system whose identification symbol is represented by the $ token.

TNERT_%@_$ Translational inertia matrix (slug = lb*sec*sec/ft) for the point whose identification symbol is represented by the @ token and the body whose identification symbol is represented by the % token. When premultiplied by a vector defining a desired change in the linear acceleration of the point in question, yields the force that must be exerted on the body ... at that point. If the force application point does not coincide with the body CM, the linear acceleration includes that induced by angular acceleration about the CM (due to the moment of the force). Force and acceleration vector components are resolved on the axes of the coordinate system whose identification symbol is represented by the $ token.

TSPRG_%@_$ Translational spring constant matrix (lb/ft) for the point whose identification symbol is represented by the @ token. When premultiplied by the displacement vector of the point in question (relative to the position where the spring force is zero), yields the spring force exerted on the body whose identification symbol is represented by the % token. Force and displacement vector components are resolved upon the axes of the coordinate system whose identification symbol is represented by the $ token. }

$ page $
{

VBASE_$_$ Vector base matrix, i.e., a matrix whose rows are unit vectors aligned with the axes of the coordinate system whose symbol is represented by the first $ token. The second $ token represents a symbol identifying the coordinate system upon whose axes the unit vector components are resolved. Such a matrix is used to transform vector components from one resolvent frame to another. For example, using the VXM function that computes the product of a vector with a 3x3 matrix, the PASCAL statement

```
CFORC_PH_I := VXM( CFORC_PH_PB, VBASE_PB_I )
```

describes a transformation --- from the Payload body frame PB to the inertial frame I --- of the components of force exerted on the Payload by the Crewman.

Euler Angles and Quaternions

PYR_$_$ An array of three angle values (rad internally, deg externally) for a pitch/yaw/roll Euler rotation sequence that would rotate the axes of one coordinate system (whose identification symbol is represented by the first $ token) into alignment with the axes of another system (whose identification symbol is represented by the second $ token). For instance, PYR_OB_PB is an array of Euler angles that defines the attitude of the Payload body axes with respect to the Orbiter body axes.

QUAT_$_$ A unit quaternion which defines an eigenaxis rotation that would rotate the axes of one coordinate system (whose identification symbol is represented by the first $ token) into alignment with the axes of another system (whose identification symbol is represented by the second $ token). For example, in a context where explicit identification of the body is unnecessary, QUAT_I_B represents a quaternion defining the orientation of the body axes B with respect to the inertial frame I. }

{ end File 'Manhsymb.I' }

1.4. Miscellany

```
$ page $  { begin File 'Manhmisc.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHMISC ;  { Subject :  Miscellany }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILSPIF ,
     UTILVEMQ ;

export                              { begin externally visible declarations }

     const

          PROGID = 'MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ ' ;
          FILL12 = '            ' { 12 spaces } ;

          FTPERIN    =   0.0833333333333333L0 ;                  { 1/12       }
          PERCENT    =   0.01L0               ;                  { 1/100      }
          STDGRAVACC =  32.174L0              ;                  { ft/sec/sec }

          STRUC2BODY = DIAG3X3 [             { diagonal matrix for converting }
               -0.0833333333333333L0 ,       { position vector components     }
               +0.0833333333333333L0 ,       { from structural (S) reference  }
               -0.0833333333333333L0 ] ;     { to body (B) reference          }

     var

          SIMTIME : longreal ;  { Simulated Time (since start of simulation ) }

     function  USER_DECIDES_NOT_TO( DO_THIS : PROMPTSTR ) : boolean ;
     function  USER_DECIDES(   THIS_IS_TRUE : PROMPTSTR ) : boolean ;

implement                       { begin externally invisible part of module }


function  USER_DECIDES_NOT_TO( DO_THIS : PROMPTSTR ) : boolean ;

     begin
     USER_DECIDES_NOT_TO := not USER_DECIDES_TO( DO_THIS )                  ;
     end ;


function  USER_DECIDES( THIS_IS_TRUE : PROMPTSTR ) : boolean  ;

     begin
     USER_DECIDES := USER_DECIDES_TO{say}( THIS_IS_TRUE )                  ;
     end ;


end ;  { module MANHMISC & File 'Manhmisc.I' }
```

1.5. PASCAL Interface with HP-9000 Operating System

1.5.1. Model 216 / Pascal 3.0 Interface Module

```
$ page $  { begin File 'MANHSPIF.I' }

{ Payload Manhandling Simulator for HP-9000 Model 216 with Pascal 3.0 Op Sys }

module MANHSPIF ;  { Subject :  System/PASCAL Interface }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     Iodeclarations , { Pascal 3.0 system module }
     General_1      , { Pascal 3.0 system module }
     General_3      , { Pascal 3.0 system module }
     General_4      , { Pascal 3.0 system module }
     UTILSPIF       ;

export                              { begin externally visible declarations }

     procedure SET_UP_HPIB_INTERFACE_TO_IMI                 ;
     procedure RESET_IMI_BUFFER                             ;
     procedure ADD_STRING_TO_IMI_BUFFER ( STR : LINESTR )   ;
     procedure TRANSFER_BUFFER_CONTENTS_TO_IMI              ;
     procedure WRITE_ERROR_MESSAGE                          ;

implement                          { begin externally invisible part of module }

     const

          IMI_DEVICE_CODE = 717 ;

     var

          IMI_BUFFER : Buf_info_type ;
```

```
$ page $


procedure SET_UP_HPIB_INTERFACE_TO_IMI ;

     begin
     Iobuffer ( IMI_BUFFER, 255 )                                        ;
     end ;


procedure RESET_IMI_BUFFER ;

     begin
     repeat
          { nothing }
          until not Buffer_busy ( IMI_BUFFER )                           ;
     Buffer_reset ( IMI_BUFFER )                                         ;
     end ;


procedure ADD_STRING_TO_IMI_BUFFER ( STR : LINESTR ) ;

     begin
     Writebuffer_string ( IMI_BUFFER, STR )                              ;
     end ;


procedure TRANSFER_BUFFER_CONTENTS_TO_IMI ;

     begin
     Transfer ( IMI_DEVICE_CODE, Overlap_intr, From_memory,
                IMI_BUFFER, Buffer_data( IMI_BUFFER ) )                  ;
     end ;
```

```
$ page $

procedure WRITE_ERROR_MESSAGE ;

     var

          K   : integer ;
          MSG : LINESTR ;

     begin
     Ioreset ( IMI_DEVICE_CODE div 100 )                                          ;
     MSG := ''                                                                    ;
     case escapecode of

         -26 :
          strwrite ( MSG,1,K,Ioerror_message( Ioe_result ) )                      ;

         -20 :
          strwrite ( MSG,1,K,'Stopped by user' )                                  ;

         -10 :
          strwrite ( MSG,1,K,'I/O system error; Ioresult ',Ioresult:1 )           ;

          otherwise
          strwrite ( MSG,1,K,'Escapecode ',escapecode:1 )                         ;

          end ; { case escapecode }

     SHOWLN ( ''  ) ;
     SHOWLN ( MSG ) ;
     SOUND_ALARM    ;
     for K := 1 to 2 do writeln ( LP ) ;
     writeln ( LP, MSG ) ;
     end ;

end ;  { module MANHSPIF & File 'MANHSPIF.I' }
```

1.5.2. Series 500 / HP-UX 5.0 Interface Module

```
$ page $  { begin File 'manhspif.I' }

{ Payload Manhandling Simulator for HP-9000 Series 500 with HP-UX 5.0 Op Sys }

module MANHSPIF ;  { Subject :  System/PASCAL Interface }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILSPIF ;

export                                { begin externally visible declarations }

     procedure SET_UP_HPIB_INTERFACE_TO_IMI                ;
     procedure RESET_IMI_BUFFER                            ;
     procedure ADD_STRING_TO_IMI_BUFFER ( STR : LINESTR ) ;
     procedure TRANSFER_BUFFER_CONTENTS_TO_IMI             ;
     procedure WRITE_ERROR_MESSAGE                         ;

implement                         { begin externally invisible part of module }

     const

          IMI_PATHSTR = '/dev/imi_hpib' ;
          IMI_IOLEN   =       1         ;
          O_WRONLY    =       1         ;
          NULLCHAR    = chr( 0 )        ;
          HPIBWRITE   = chr( 2 )        ;

     type

          PATHPAC  =  packed array [ 1..20  ] of char ;
          IOBUFPAC =  packed array [ 1..255 ] of char ;
          IOBUFPTR = ^IOBUFPAC ;

          IODETAIL =
               record
               MODE       : char     ;
               TERMINATOR : char     ;
               count      : integer  ;
               BUF        : IOBUFPTR ;
               end ; { record }

     var

          IMI_EID           : integer  ;
          IMI_IOVEC         : IODETAIL ;
          IMI_PATHPAC       : PATHPAC  ;
          IMI_TRANSFER_FLAG : integer  ;

function  errno   $ alias 'Pas.Errno' $   : integer ; external ;

function  hpib_io(      EID   : integer  ;
                    var IOVEC : IODETAIL ;
                        IOLEN : integer   ) : integer ; external ;

function  open(     var PATH  : PATHPAC  ;
                        OFLAG : integer   ) : integer ; external ;
```

```
procedure SET_UP_HPIB_INTERFACE_TO_IMI ;

     var

          k : integer ;

     begin
     for k := 1 to strlen( IMI_PATHSTR ) do
          IMI_PATHPAC[k] := IMI_PATHSTR[k]                              ;
     IMI_PATHPAC [ 1 + strlen( IMI_PATHSTR ) ] := NULLCHAR             ;
     IMI_EID := open( IMI_PATHPAC, O_WRONLY )                          ;
     if IMI_EID = -1 then escape ( 101 )                               ;
     with IMI_IOVEC do
          begin
          MODE := HPIBWRITE                                            ;
          TERMINATOR := NULLCHAR                                       ;
          new ( BUF )                                                  ;
          end ;
     IMI_TRANSFER_FLAG := 0                                            ;
     end ;


procedure RESET_IMI_BUFFER ;

     begin
     if IMI_TRANSFER_FLAG <> 0 then escape ( 102 )                     ;
     IMI_IOVEC.count := 0                                              ;
     end ;


procedure ADD_STRING_TO_IMI_BUFFER ( STR : LINESTR ) ;

     var

          k : integer ;

     begin
     with IMI_IOVEC do
          for k := 1 to strlen( STR ) do
               begin
               count := count + 1                                      ;
               BUF^[count] := STR[k]                                   ;
               end ;
     end ;


procedure TRANSFER_BUFFER_CONTENTS_TO_IMI ;

     begin
     IMI_TRANSFER_FLAG := hpib_io( IMI_EID, IMI_IOVEC, IMI_IOLEN )     ;
     end ;
```

```
$ page $


procedure WRITE_ERROR_MESSAGE ;

     var

          K   : integer ;
          MSG : LINESTR ;

     begin
     MSG := ''                                                              ;
     case escapecode of

         -1301 :
          strwrite ( MSG,1,K,'HP-UX errno ',errno:1 )                       ;

         -1300..-1201 :
          strwrite ( MSG,1,K,'Pascal I/O error ',-(escapecode+1200):1 )     ;

         -20 :
          strwrite ( MSG,1,K,'Stopped by user' )                            ;

         -19..-1 :
          strwrite ( MSG,1,K,'HP-UX signal ',-escapecode:1,' received' )    ;

          otherwise
          strwrite ( MSG,1,K,'Escapecode ',escapecode:1 )                   ;

          end ; { case escapecode }

     SHOWLN ( ''  ) ;
     SHOWLN ( MSG ) ;
     SOUND_ALARM    ;
     for K := 1 to 2 do writeln ( LP ) ;
     writeln ( LP, MSG ) ;
     end ;


end ;  { module MANHSPIF & File 'manhspif.I' }
```

1.6. Body (Object) Data

```
$ page $  { begin File 'Manhbods.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHBODS ;  { Subject :  Body (Object) Data }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILSPIF ,
     UTILVEMQ ;

export                                  { begin externally visible declarations }

     const

          NUMOBJECTS =  3 ;  { no. of objects that may appear in camera image }

     type

          OBJECT = (
               orbiter ,     { objects that may appear in camera image;        }
               payload ,     { graphics system requires a file of coordinates }
               crewman ) ;   { to define the geometry of each one              }

          INFOREC =           { record containing information about an object }
               record
               case integer of

                    1 : ( RNERT_CM_B : MAT3X3   ;         { Orbiter & Payload }
                          RALAC_CM_B : MAT3X3   ;
                          TORQ_CM_B  : VECTOR   ;
                          VBASE_I_B  : MAT3X3   ;
                          FORC_CM_I  : VECTOR   ;
                          MASS       : longreal ;      { slug = lb*sec*sec/ft }
                          CFORC_H_B  : VECTOR   ;
                          CTORQ_H_B  : VECTOR   ) ;

                    2 : ( TALAC_CRO_R : MAT3X3 ;                   { Crewman }
                          TDAMP_CRO_R : MAT3X3 ;
                          TSPRG_CRO_R : MAT3X3 ) ;

               end ; { case & record }
```

```
$ page $

     const

          STATESIZE  = 13 ;  { number of scalar variables required to define  }
                             { the motion state of one object                }

     type

          STATEREC =            { record defining an object's state of motion }
               record
               case integer of

                    1 : ( ARR        : array [1..STATESIZE] of longreal ) ;

                    2 : ( IPOS_CM_I  : VECTOR     ;       { Orbiter and      }
                          IVEL_CM_I  : VECTOR     ;       { Payload,         }
                          QUAT_I_B   : QUATERNION ;       { relative to      }
                          IRATE_B_B  : VECTOR     ) ;     { inertial frame I }

                    3 : ( OBPOS_RO_R : VECTOR     ;       { Crewman, rela-   }
                          OBVEL_RO_R : VECTOR     ) ;     { tive to Orbiter  }
                                                          { body axes (OB)   }
               end ; { case & record }

          DERIVREC =       { record of state variable derivatives w.r.t. time }
               record
               case integer of

                    1 : ( ARR           : array [1..STATESIZE] of longreal ) ;

                    2 : ( IPOSDOT_CM_I  : VECTOR     ;    { Orbiter and      }
                          IVELDOT_CM_I  : VECTOR     ;    { Payload,         }
                          QUATDOT_I_B   : QUATERNION ;    { relative to      }
                          IRATEDOT_B_B  : VECTOR     ) ;  { inertial frame I }

                    3 : ( OBPOSDOT_RO_R : VECTOR     ;    { Crewman, rela-   }
                          OBVELDOT_RO_R : VECTOR     ) ;  { tive to Orbiter  }
                                                          { body axes (OB)   }
               end ; { case & record }

               { * NOTE:  For the purpose of integrating the differential      }
               {          equations of motion for the Crewman (only), the      }
               {          Orbiter's body-fixed coordinate system OB is         }
               {          treated as if it were an inertial frame.  This is    }
               {          justified on the basis that the acceleration and     }
               {          angular velocity of the Orbiter will of necessity    }
               {          be very small while the Crewman is manhandling the   }
               {          payload, that the matrix constants (TALAC,TDAMP,     }
               {          TSPRG) used in the Crewman's equations of motion     }
               {          are only approximate, and that divergence of the     }
               {          Orbiter-relative position of the Crewman is pre-     }
               {          cluded by his physical connection to the Orbiter     }
               {          structure through the Remote Manipulator System      }
               {          (RMS).                                               }
```

```
$ page $

     var

          INFO  : array [ OBJECT ] of INFOREC  ;
          STATE : array [ OBJECT ] of STATEREC ;
          DERIV : array [ OBJECT ] of DERIVREC ;

          FORC_CRO_R : VECTOR ;
          LOC_H_PB   : VECTOR ;
          OBPOS_H_OB : VECTOR ;
          OBVEL_H_OB : VECTOR ;
          OBPOS_N_R  : VECTOR ;

          RNERT_PCM_PB : MAT3X3 ;
          RNERT_PH_PB  : MAT3X3 ;
          TNERT_PH_PB  : MAT3X3 ;
          VBASE_R_PB   : MAT3X3 ;
          VBASE_OB_R   : MAT3X3 ;
          VBASE_OB_PB  : MAT3X3 ;

          QUAT_R_PB  : QUATERNION ;
          QUAT_OB_R  : QUATERNION ;
          QUAT_OB_PB : QUATERNION ;
          QUAT_OB_PD : QUATERNION ;

implement                          { begin externally invisible part of module }

end ;  { module MANHBODS & File 'Manhbods.I' }
```

1.7. Flight Control

```
$ page $  { begin File 'Manhfcon.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHFCON ;  { Subject :  Flight Control }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILMATH ,
     UTILSPIF ,
     UTILVEMQ ,
     MANHBODS ;

export                               { begin externally visible declarations }

     type

          REALARR3 = array [1..3] of longreal ;

          CNTRLMODE = (                      { Payload (PL) flight control option }
               FREE     ,    {     no control ( "hands off" )                     }
               ALIGN    ,    { *   align PL angvel vector with PBx (spin) axis    }
               DESPIN   ,    {     oppose rotation of PL about PBx (spin) axis    }
               CAPTURE  ,    {     nullify PL ang and lin vel (hold afterward)    }
               HOLD     ,    {     maintain captured state (capture 1st if nec)   }
               ROLL     ,    {     nullify PL roll error (capture first if nec)   }
               YAW      ,    {     nullify PL yaw error (after roll correction)   }
               PITCH    ,    {     nullify PL pch error (after roll & yaw corr)   }
               SPINUP   ) ;  {     accelerate PL rotatn about PBx (spin) axis     }

                             { * NOTE: The ALIGN mode is included for analysis    }
                             {         only; it is not considered feasible of     }
                             {         implementation by Crewman.                 }

          CNTRLPACARR = array [ CNTRLMODE ] of NAMEPAC ;

     const

          CNTRLPAC = CNTRLPACARR [
               NAMEPAC [ ' FREE    ' ] ,
               NAMEPAC [ ' ALIGN   ' ]  ,
               NAMEPAC [ ' DESPIN  ' ]  ,
               NAMEPAC [ ' CAPTURE' ]  ,
               NAMEPAC [ ' HOLD    ' ]  ,
               NAMEPAC [ ' ROLL    ' ]  ,
               NAMEPAC [ ' YAW     ' ]  ,
               NAMEPAC [ ' PITCH   ' ]  ,
               NAMEPAC [ ' SPINUP  ' ]  ] ;
```

```
$ page $

     var

          current_mode : CNTRLMODE ;
          desired_mode : CNTRLMODE ;

          AERR : EULPYR ;                      { angular error    = PYR_PD_PB    }
          AVEL : VECTOR ;                      { angular velocity = PDRATE_PB_PB }
          LERR : VECTOR ;                      { linear error     = RPOS_H_R     }
          LVEL : VECTOR ;                      { linear velocity  = RVEL_H_R     }

          CNTRLCON :        { record containing manhandling control constants }
               record
               AERRMAX : REALARR3 ; { max PB pch/yaw/rol w.r.t. PD axes ( * ) }
               AERRMIN : REALARR3 ; { min PB pch/yaw/rol w.r.t. PD axes ( * ) }
               AERRTOL : REALARR3 ; { pch/yaw/rol tolerances                  }
               AVELTOL : REALARR3 ; { angular velocity component tolerances   }
               DSPFORC : VECTOR   ; { despin force vector applied at H ( ** ) }
               FORCLIM : REALARR3 ; { force component limiting magnitudes     }
               LERRLIM : REALARR3 ; { pos error compnent lim magnitudes ( * ) }
               LERRTOL : REALARR3 ; { position error component tolerances     }
               LVELTOL : longreal ; { linear vel component tol   ( any axis ) }
               NOMAVEL : REALARR3 ; { nom rot rates for correcting ang error  }
               NOMLVEL : longreal ; { nom vel mag for correcting linear error }
               SPUFORC : VECTOR   ; { spinup force vector applied at H ( ** ) }
               TIMECON : longreal ; { time allowed to achieve corrective vel  }
               TORQLIM : REALARR3 ; { torque component limiting magnitudes    }
               end ; {record}

                              {  * NOTE: H out of reach if limits exceeded  }
                              { ** NOTE: FORCLIM does not apply             }


     procedure COMPUTE_FLIGHT_CONTROL_FORCE_AND_TORQUE ;

implement                       { begin externally invisible part of module }
```

```
$ page $

     var

          DESRACC : VECTOR   ;  { desired rotational    acceleration }
          DESTACC : VECTOR   ;  { desired translational acceleration }
          GAIN    : longreal ;
          VERR    : longreal ;

     procedure COMPUTE_CONTROL_STATE                                   ;forward;
     procedure NULLIFY_FORCE_AND_TORQUE                                ;forward;
     procedure COMPUTE_ALIGNMENT_FORCE_AND_TORQUE                      ;forward;
     procedure COMPUTE_DESPIN_FORCE_AND_TORQUE                         ;forward;
     procedure COMPUTE_CAPTURE_FORCE_AND_TORQUE                        ;forward;
     procedure COMPUTE_CAPTURE_FORCE                                   ;forward;
     procedure COMPUTE_CAPTURE_DESRACC                                 ;forward;
     procedure COMPUTE_CONTROL_TORQUE                                  ;forward;
     procedure COMPUTE_ROLL_FORCE_AND_TORQUE                           ;forward;
     procedure CHECK_AGAIN_EXCEPTING_ROTATIONAL_AXIS ( E : integer )   ;forward;
     procedure COMPUTE_YAW_FORCE_AND_TORQUE                            ;forward;
     procedure COMPUTE_PITCH_FORCE_AND_TORQUE                          ;forward;
     procedure COMPUTE_SPINUP_FORCE_AND_TORQUE                         ;forward;
     function  CREWMAN_LATERAL_CORRECTIVE_FORCE : VECTOR               ;forward;
```

```
$ page $

procedure COMPUTE_FLIGHT_CONTROL_FORCE_AND_TORQUE ;

     var

          i            : integer ;
          OUT_OF_REACH : boolean ;

     begin
     COMPUTE_CONTROL_STATE                                                         ;
     OUT_OF_REACH := false                                                         ;
     with CNTRLCON do
          for i := 1 to 3 do
               begin
               if abs( LERR[i] ) > LERRLIM[i] then
                    OUT_OF_REACH := true                                           ;
               if AERR[i] > AERRMAX[i] then
                    OUT_OF_REACH := true                                           ;
               if AERR[i] < AERRMIN[i] then
                    OUT_OF_REACH := true                                           ;
               end ;
     if OUT_OF_REACH or ( desired_mode = FREE )

          then
          NULLIFY_FORCE_AND_TORQUE

          else
          case desired_mode of
               ALIGN   : COMPUTE_ALIGNMENT_FORCE_AND_TORQUE                        ;
               DESPIN  : COMPUTE_DESPIN_FORCE_AND_TORQUE                           ;
               CAPTURE,
               HOLD    : COMPUTE_CAPTURE_FORCE_AND_TORQUE                          ;
               ROLL    : COMPUTE_ROLL_FORCE_AND_TORQUE                             ;
               YAW     : COMPUTE_YAW_FORCE_AND_TORQUE                              ;
               PITCH   : COMPUTE_PITCH_FORCE_AND_TORQUE                            ;
               SPINUP  : COMPUTE_SPINUP_FORCE_AND_TORQUE                           ;
               end ; { case desired_mode }

     end ;
```

```
$ page $

procedure COMPUTE_CONTROL_STATE ;

     var

          IPOS_H_I : VECTOR ;
          IVEL_H_I : VECTOR ;

     begin
     with INFO[payload], STATE[payload] do
          begin
          VBASE_I_B := QMAT( QUAT_I_B )                                     ;
          IPOS_H_I  := VSUM( IPOS_CM_I, VXMT( LOC_H_PB,
                                              VBASE_I_B ) )                 ;
          IVEL_H_I  := VSUM( IVEL_CM_I, VXMT( CRSP( IRATE_B_B, LOC_H_PB ),
                                              VBASE_I_B ) )                 ;
          end ;
     with INFO[orbiter], STATE[orbiter], CNTRLCON do
          begin
          VBASE_I_B   := QMAT( QUAT_I_B )                                   ;
          QUAT_R_PB   := QCXQ( QUAT_OB_R, QUAT_OB_PB )                      ;
          VBASE_R_PB  := QMAT( QUAT_R_PB )                                  ;
          OBPOS_H_OB :=        VXM(  VDIF( IPOS_H_I, IPOS_CM_I ), VBASE_I_B )  ;
          OBVEL_H_OB := VDIF( VXM(  VDIF( IVEL_H_I, IVEL_CM_I ), VBASE_I_B ),
                              CRSP( IRATE_B_B, OBPOS_H_OB ) )               ;
          AERR        := QPYR( QCXQ( QUAT_OB_PD, QUAT_OB_PB ) )             ;
          AVEL        := VDIF( STATE[payload].IRATE_B_B,
                              VXM( IRATE_B_B, VBASE_OB_PB ) )               ;
          end ;
     with STATE[crewman], CNTRLCON do
          begin
          LERR := VDIF( VXM( OBPOS_H_OB, VBASE_OB_R ), OBPOS_RO_R )          ;
          LVEL := VDIF( VXM( OBVEL_H_OB, VBASE_OB_R ), OBVEL_RO_R )          ;
          end ;
     end ;
```

```
$ page $

procedure NULLIFY_FORCE_AND_TORQUE ;

     begin
     current_mode := FREE                                                   ;
     with INFO[payload] do
          begin
          CFORC_H_B  := ZERVEC                                              ;
          CTORQ_H_B  := ZERVEC                                              ;
          end ;
     with INFO[orbiter] do
          begin
          CFORC_H_B  := ZERVEC                                              ;
          CTORQ_H_B  := ZERVEC                                              ;
          end ;
     end ;


procedure COMPUTE_ALIGNMENT_FORCE_AND_TORQUE ;

     var

          i       : integer ;
          TORQ    : VECTOR  ;

     begin
     current_mode := ALIGN                                                  ;
     with CNTRLCON do
          begin
          DESRACC[1] := ZERO                                                ;
          for i := 2 to 3 do
               begin
               VERR := AVEL[i]                                              ;
               if AVELTOL[i] > ZERO
                    then GAIN := RMIN( ONE, abs(VERR/AVELTOL[i]) )
                    else GAIN := ZERO                                       ;
               DESRACC[i] := -( GAIN*VERR/TIMECON )                         ;
               end ;
          TORQ := VXM( DESRACC, RNERT_PH_PB )                               ;
          for i := 1 to 3 do
               if abs(TORQ[i]) > TORQLIM[i] then
                    TORQ[i] := RSIGN( TORQ[i] ) * TORQLIM[i]                ;
          end ;
     with INFO[payload] do
          begin
          CFORC_H_B := ZERVEC                                               ;
          CTORQ_H_B := TORQ                                                 ;
          end ;
     with INFO[orbiter] do
          begin
          CFORC_H_B := ZERVEC                                               ;
          CTORQ_H_B := VXMT( VDIF( ZERVEC, TORQ ), VBASE_OB_PB )            ;
          end ;
     end ;
```

```
$ page $

procedure COMPUTE_DESPIN_FORCE_AND_TORQUE ;

     var

          FORC : VECTOR ;

     begin
     current_mode := DESPIN                                                  ;
     FORC := VSUM( CNTRLCON.DSPFORC, CREWMAN_LATERAL_CORRECTIVE_FORCE )      ;
     with INFO[payload] do
          begin
          CFORC_H_B := FORC                                                  ;
          CTORQ_H_B := ZERVEC                                                ;
          end ;
     with INFO[orbiter] do
          begin
          CFORC_H_B := VXMT( VDIF( ZERVEC, FORC ), VBASE_OB_PB )             ;
          CTORQ_H_B := ZERVEC                                                ;
          end ;
     end ;


function CREWMAN_LATERAL_CORRECTIVE_FORCE : VECTOR ;

     var

          FORC : VECTOR ;

     begin
     with  CNTRLCON do
          if NOMLVEL > ZERO

               then
               begin
               DESTACC := ZERVEC                                             ;
               if abs( LERR[2] ) > LERRTOL[2]
                    then VERR := LVEL[2] + RSIGN( LERR[2] ) * NOMLVEL
                    else VERR := LVEL[2]                                     ;
               if LVELTOL > ZERO
                    then GAIN := RMIN( ONE, abs(VERR/LVELTOL) )
                    else GAIN := ZERO                                        ;
               DESTACC[2] := -( GAIN*VERR/TIMECON )                          ;
               FORC := VXM( VXM( DESTACC, VBASE_R_PB ), TNERT_PH_PB )        ;
               end

               else
               FORC := ZERVEC                                                ;

     CREWMAN_LATERAL_CORRECTIVE_FORCE := FORC                                ;
     end ;
```

```
$ page $

procedure COMPUTE_CAPTURE_FORCE_AND_TORQUE ;

     begin
     COMPUTE_CAPTURE_FORCE                                                      ;
     COMPUTE_CAPTURE_DESRACC                                                    ;
     COMPUTE_CONTROL_TORQUE                                                     ;
     end ;


procedure COMPUTE_CAPTURE_FORCE ;

     var

          i    : integer ;
          FORC : VECTOR  ;

     begin
     current_mode := HOLD                                                       ;
     with CNTRLCON do
          begin
          for i := 1 to 3 do
               begin
               if abs( LERR[i] ) > LERRTOL[i]

                    then
                    begin
                    current_mode := CAPTURE                                     ;
                    GAIN := ( abs(LERR[i]) - LERRTOL[i] ) /
                            (  LERRLIM[i]  - LERRTOL[i] )                       ;
                    VERR := LVEL[i] + RSIGN( LERR[i] ) * GAIN * NOMLVEL         ;
                    end

                    else
                    VERR := LVEL[i]                                             ;

               if LVELTOL > ZERO
                    then GAIN := RMIN( ONE, abs(VERR/LVELTOL) )
                    else GAIN := ZERO                                           ;
               if GAIN = ONE then
                    current_mode := CAPTURE                                     ;
               DESTACC[i] := -( GAIN*VERR/TIMECON )                             ;
               end ;
          FORC    := VXM( VXM( DESTACC, VBASE_R_PB ), TNERT_PH_PB )             ;
          for i := 1 to 3 do
               if abs( FORC[i] ) > FORCLIM[i] then
                    FORC[i] := RSIGN( FORC[i] ) * FORCLIM[i]                    ;
          end ;
     INFO[payload].CFORC_H_B := FORC                                            ;
     INFO[orbiter].CFORC_H_B := VXMT( VDIF( ZERVEC, FORC ), VBASE_OB_PB )       ;
     end ;
```

```
$ page $

procedure COMPUTE_CAPTURE_DESRACC ;

     var

          i : integer ;

     begin
     with CNTRLCON do
          for i := 1 to 3 do
               begin
               VERR := AVEL[i]                                                  ;
               if AVELTOL[i] > ZERO
                    then GAIN := RMIN( ONE, abs(VERR/AVELTOL[i]) )
                    else GAIN := ZERO                                           ;
               if GAIN = ONE then
                    current_mode := CAPTURE                                     ;
               DESRACC[i] := -( GAIN*VERR/TIMECON )                             ;
               end ;
     end ;


procedure COMPUTE_CONTROL_TORQUE ;

     var

          i    : integer ;
          TORQ : VECTOR  ;

     begin
     with CNTRLCON do
          begin
          TORQ := VDIF( VXM(  DESRACC , RNERT_PH_PB ),
                        CRSP( LOC_H_PB, INFO[payload].CFORC_H_B ) )             ;
          for i := 1 to 3 do
               if abs( TORQ[i] ) > TORQLIM[i] then
                    TORQ[i] := RSIGN( TORQ[i] ) * TORQLIM[i]                    ;
          end ;
     INFO[payload].CTORQ_H_B := TORQ                                            ;
     INFO[orbiter].CTORQ_H_B := VXMT( VDIF( ZERVEC, TORQ ), VBASE_OB_PB )       ;
     end ;
```

```
$ page $

procedure COMPUTE_ROLL_FORCE_AND_TORQUE ;

     begin
     COMPUTE_CAPTURE_FORCE                                                   ;
     COMPUTE_CAPTURE_DESRACC                                                 ;
     if current_mode = CAPTURE then
          CHECK_AGAIN_EXCEPTING_ROTATIONAL_AXIS ( 1 )                        ;
     if current_mode <> CAPTURE then
          with CNTRLCON do
               if abs( AERR[3] ) > AERRTOL[3] then
                    begin
                    current_mode := ROLL                                     ;
                    VERR := AVEL[1] + RSIGN( AERR[3] ) * NOMAVEL[1]          ;
                    if AVELTOL[1] > ZERO
                         then GAIN := RMIN( ONE, abs(VERR/AVELTOL[1]) )
                         else GAIN := ZERO                                   ;
                    DESRACC[1] := -( GAIN*VERR/TIMECON )                     ;
                    end ;
     COMPUTE_CONTROL_TORQUE                                                  ;
     end ;

procedure CHECK_AGAIN_EXCEPTING_ROTATIONAL_AXIS ( E : integer ) ;

     var

          i : integer ;

     begin
     case E of
          1 : current_mode := ROLL                                           ;
          2 : current_mode := PITCH                                          ;
          3 : current_mode := YAW                                            ;
          end ; { case E }
     with CNTRLCON do
          for i := 1 to 3 do
               begin
               if abs( LERR[i] ) > ( 1.5 * LERRTOL[i] ) then
                    current_mode := CAPTURE                                  ;
               if i <> E then
                    if abs( AVEL[i] ) > AVELTOL[i] then
                         current_mode := CAPTURE                             ;
               end ;
     end ;
```

```
$ page $

procedure COMPUTE_YAW_FORCE_AND_TORQUE ;

     begin
     with CNTRLCON do
          if abs( AERR[3] ) > AERRTOL[3]

               then
               COMPUTE_ROLL_FORCE_AND_TORQUE

               else
               begin
               COMPUTE_CAPTURE_FORCE                                          ;
               COMPUTE_CAPTURE_DESRACC                                        ;
               if current_mode = CAPTURE then
                    CHECK_AGAIN_EXCEPTING_ROTATIONAL_AXIS ( 3 )               ;
               if current_mode <> CAPTURE then
                    if abs( AERR[2] ) > AERRTOL[2] then
                         begin
                         current_mode := YAW                                  ;
                         VERR := AVEL[3] + RSIGN( AERR[2] ) * NOMAVEL[3]      ;
                         if AVELTOL[3] > ZERO
                              then GAIN := RMIN( ONE, abs(VERR/AVELTOL[3]) )
                              else GAIN := ZERO                               ;
                         DESRACC[3] := -( GAIN*VERR/TIMECON )                 ;
                         end ;
               end ;
     COMPUTE_CONTROL_TORQUE                                                   ;
     end ;
```

```
$ page $

procedure COMPUTE_PITCH_FORCE_AND_TORQUE ;

     begin
     with CNTRLCON do
          if abs( AERR[3] ) > AERRTOL[3]

               then
               COMPUTE_ROLL_FORCE_AND_TORQUE

               else
               if abs( AERR[2] ) > AERRTOL[2]

                    then
                    COMPUTE_YAW_FORCE_AND_TORQUE

                    else
                    begin
                    COMPUTE_CAPTURE_FORCE                                    ;
                    COMPUTE_CAPTURE_DESRACC                                  ;
                    if current_mode = CAPTURE then
                         CHECK_AGAIN_EXCEPTING_ROTATIONAL_AXIS ( 2 )         ;
                    if current_mode <> CAPTURE then
                         if abs( AERR[1] ) > AERRTOL[1] then
                              begin
                              current_mode := PITCH                          ;
                              VERR := AVEL[2] + RSIGN( AERR[1] ) * NOMAVEL[2] ;
                              if AVELTOL[2] > ZERO
                                   then GAIN := RMIN(ONE,abs(VERR/AVELTOL[2]))
                                   else GAIN := ZERO                         ;
                              DESRACC[2] := -( GAIN*VERR/TIMECON )           ;
                              end ;
                    end ;
     COMPUTE_CONTROL_TORQUE                                                  ;
     end ;


procedure COMPUTE_SPINUP_FORCE_AND_TORQUE ;

     var

          FORC : VECTOR ;

     begin
     current_mode := SPINUP                                                  ;
     FORC := VSUM( CNTRLCON.SPUFORC, CREWMAN_LATERAL_CORRECTIVE_FORCE )      ;
     with INFO[payload] do
          begin
          CFORC_H_B := FORC                                                  ;
          CTORQ_H_B := ZERVEC                                                ;
          end ;
     with INFO[orbiter] do
          begin
          CFORC_H_B := VXMT( VDIF( ZERVEC, FORC), VBASE_OB_PB )              ;
          CTORQ_H_B := ZERVEC                                                ;
          end ;
     end ;

end ;  { module MANHFCON & File 'Manhfcon.I' }
```

1.8. Data Dump Routines

```
$ page $  { begin File 'Manhdump.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHDUMP ;  { Subject :  Data Dump Routines }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILSPIF ,
     UTILVEMQ ,
     MANHMISC ,
     MANHBODS ,
     MANHFCON ;

export                                  { begin externally visible declarations }

     procedure DUMP_EVERYTHING                                    ;
     procedure DUMP_INFO_RECORDS                                  ;
     procedure DUMP_STATE_RECORDS                                 ;
     procedure DUMP_DERIV_RECORDS                                 ;
     procedure DUMPEUL  ( NAME : WORDSTR ; E : EULARR     ) ;
     procedure DUMPINT  ( NAME : WORDSTR ; I : integer    ) ;
     procedure DUMPMAT  ( NAME : WORDSTR ; M : MAT3X3     ) ;
     procedure DUMPQUAT ( NAME : WORDSTR ; Q : QUATERNION ) ;
     procedure DUMPREAL ( NAME : WORDSTR ; R : longreal   ) ;
     procedure DUMPVEC  ( NAME : WORDSTR ; V : VECTOR     ) ;


implement                           { begin externally invisible part of module }

     const
                 {         1         2}
                 {12345678901234567890}
          NULL = '                    ' ;
```

```
$ page $

procedure DUMP_EVERYTHING ;

     begin
     START_NEW_PAGE ;
     DUMPREAL ( 'SIMTIME                ',SIMTIME       )                          ;
     writeln ( LP, 'current_mode         ',CNTRLPAC[current_mode] )                ;
     writeln ( LP ) ;
     writeln ( LP, 'desired_mode         ',CNTRLPAC[desired_mode] )                ;
     writeln ( LP ) ;
     DUMPEUL  ( 'AERR                   ',AERR          )                          ;
     DUMPVEC  ( 'AVEL                   ',AVEL          )                          ;
     DUMPVEC  ( 'LERR                   ',LERR          )                          ;
     DUMPVEC  ( 'LVEL                   ',LVEL          )                          ;
     DUMPVEC  ( 'FORC_CRO_R             ',FORC_CRO_R    )                          ;
     DUMPVEC  ( 'LOC_H_PB               ',LOC_H_PB      )                          ;
     DUMPVEC  ( 'OBPOS_H_OB             ',OBPOS_H_OB    )                          ;
     DUMPVEC  ( 'OBVEL_H_OB             ',OBVEL_H_OB    )                          ;
     DUMPVEC  ( 'OBPOS_N_R              ',OBPOS_N_R     )                          ;
     DUMPMAT  ( 'RNERT_PCM_PB           ',RNERT_PCM_PB  )                          ;
     DUMPMAT  ( 'RNERT_PH_PB            ',RNERT_PH_PB   )                          ;
     DUMPMAT  ( 'TNERT_PH_PB            ',TNERT_PH_PB   )                          ;
     DUMPMAT  ( 'VBASE_R_PB             ',VBASE_R_PB    )                          ;
     DUMPMAT  ( 'VBASE_OB_R             ',VBASE_OB_R    )                          ;
     DUMPMAT  ( 'VBASE_OB_PB            ',VBASE_OB_PB   )                          ;
     DUMPQUAT ( 'QUAT_R_PB              ',QUAT_R_PB     )                          ;
     DUMPQUAT ( 'QUAT_OB_R              ',QUAT_OB_R     )                          ;
     DUMPQUAT ( 'QUAT_OB_PB             ',QUAT_OB_PB    )                          ;
     DUMPQUAT ( 'QUAT_OB_PD             ',QUAT_OB_PD    )                          ;
     START_NEW_PAGE                                                                ;
     DUMP_INFO_RECORDS                                                             ;
     DUMP_STATE_RECORDS                                                            ;
     DUMP_DERIV_RECORDS                                                            ;
     end ;
```

```
$ page $

procedure DUMP_INFO_RECORDS ;

     var

          body : OBJECT ;

     begin
     for body := orbiter to payload do
          with INFO[body] do
               begin
               write   ( LP, 'INFO[':25 )                                        ;
               if body = orbiter
                    then write ( LP, 'orbiter' )
                    else write ( LP, 'payload' )                                 ;
               writeln ( LP, ']' )                                               ;
               writeln ( LP )                                                    ;
                          {         1         2}
                          {12345678901234567890}
               DUMPMAT  ( 'RNERT_CM_B          ',RNERT_CM_B )                    ;
               DUMPMAT  ( 'RALAC_CM_B          ',RALAC_CM_B )                    ;
               DUMPVEC  ( 'TORQ_CM_B           ',TORQ_CM_B  )                    ;
               DUMPMAT  ( 'VBASE_I_B           ',VBASE_I_B  )                    ;
               DUMPVEC  ( 'FORC_CM_I           ',FORC_CM_I  )                    ;
               DUMPREAL ( 'MASS                ',MASS       )                    ;
               DUMPVEC  ( 'CFORC_H_B           ',CFORC_H_B  )                    ;
               DUMPVEC  ( 'CTORQ_H_B           ',CTORQ_H_B  )                    ;
               writeln ( LP ) ;
               end ;
     with INFO[crewman] do
          begin
          writeln ( LP, 'INFO[crewman]':33 )                                     ;
          writeln ( LP )                                                         ;
                     {         1         2}
                     {12345678901234567890}
          DUMPMAT  ( 'TALAC_CRO_R         ',TALAC_CRO_R )                        ;
          DUMPMAT  ( 'TDAMP_CRO_R         ',TDAMP_CRO_R )                        ;
          DUMPMAT  ( 'TSPRG_CRO_R         ',TSPRG_CRO_R )                        ;
          writeln ( LP )                                                         ;
          end ;
     end ;
```

```
$ page $

procedure DUMP_STATE_RECORDS ;

     var

          body : OBJECT ;

     begin
     for body := orbiter to payload do
          with STATE[body] do
               begin
               write   ( LP, 'STATE[':25 )                                  ;
               if body = orbiter
                    then write ( LP, 'orbiter' )
                    else write ( LP, 'payload' )                            ;
               writeln ( LP, ']' )                                          ;
               writeln ( LP )                                               ;
                          {         1         2}
                          {12345678901234567890}
               DUMPVEC  ( 'IPOS_CM_I           ',IPOS_CM_I  )               ;
               DUMPVEC  ( 'IVEL_CM_I           ',IVEL_CM_I  )               ;
               DUMPQUAT ( 'QUAT_I_B            ',QUAT_I_B   )               ;
               DUMPVEC  ( 'IRATE_B_B           ',IRATE_B_B  )               ;
               writeln ( LP )                                               ;
               end ;
     with STATE[crewman] do
          begin
          writeln ( LP, 'STATE[crewman]':33 )                               ;
          writeln ( LP )                                                    ;
                     {         1         2}
                     {12345678901234567890}
          DUMPVEC  ( 'OBPOS_RO_R          ',OBPOS_RO_R )                    ;
          DUMPVEC  ( 'OBVEL_RO_R          ',OBVEL_RO_R )                    ;
          writeln ( LP )                                                    ;
          end ;
     START_NEW_PAGE                                                         ;
     end ;
```

```
$ page $

procedure DUMP_DERIV_RECORDS ;

     var

          body : OBJECT ;

     begin
     for body := orbiter to payload do
          with DERIV[body] do
               begin
               write   ( LP, 'DERIV[':25 )                                       ;
               if body = orbiter
                    then write ( LP, 'orbiter' )
                    else write ( LP, 'payload' )                                 ;
               writeln ( LP, ']' )                                               ;
               writeln ( LP )                                                    ;
                         {         1         2}
                         {12345678901234567890}
               DUMPVEC  ( 'IPOSDOT_CM_I        ',IPOSDOT_CM_I   )                ;
               DUMPVEC  ( 'IVELDOT_CM_I        ',IVELDOT_CM_I   )                ;
               DUMPQUAT ( 'QUATDOT_I_B         ',QUATDOT_I_B    )                ;
               DUMPVEC  ( 'IRATEDOT_B_B        ',IRATEDOT_B_B   )                ;
               writeln ( LP )                                                    ;
               end ;
     with DERIV[crewman] do
          begin
          writeln ( LP, 'DERIV[crewman]':33 )                                    ;
          writeln ( LP )                                                         ;
                    {         1         2}
                    {12345678901234567890}
          DUMPVEC  ( 'OBPOSDOT_RO_R       ',OBPOSDOT_RO_R )                      ;
          DUMPVEC  ( 'OBVELDOT_RO_R       ',OBVELDOT_RO_R )                      ;
          writeln ( LP )                                                         ;
          end ;
     START_NEW_PAGE                                                              ;
     end ;
```

```
$ page $

procedure DUMPEUL ( NAME : WORDSTR ; E : EULARR ) ;

     begin
     writeln ( LP, NAME,E[1]:18,'   ', E[2]:18,'   ', E[3]:18 )             ;
     writeln ( LP )                                                        ;
     end ;

procedure DUMPINT ( NAME : WORDSTR ; I : integer ) ;

     begin
     writeln ( LP, NAME,I:18 )                                             ;
     writeln ( LP )                                                        ;
     end ;

procedure DUMPMAT ( NAME : WORDSTR ; M : MAT3X3 ) ;

     begin
     writeln ( LP, NAME,M[1,1]:18,'   ', M[1,2]:18,'   ', M[1,3]:18 )      ;
     writeln ( LP, NULL,M[2,1]:18,'   ', M[2,2]:18,'   ', M[2,3]:18 )      ;
     writeln ( LP, NULL,M[3,1]:18,'   ', M[3,2]:18,'   ', M[3,3]:18 )      ;
     writeln ( LP )                                                        ;
     end ;

procedure DUMPQUAT ( NAME : WORDSTR ; Q : QUATERNION ) ;

     begin
     writeln ( LP, NAME,Q.H:18,'   ' )                                     ;
     writeln ( LP, NULL,Q.I:18,'   ', Q.J:18,'   ', Q.K:18 )               ;
     writeln ( LP )                                                        ;
     end ;

procedure DUMPREAL ( NAME : WORDSTR ; R : longreal ) ;

     begin
     writeln ( LP, NAME,R:18 )                                             ;
     writeln ( LP )                                                        ;
     end ;

procedure DUMPVEC ( NAME : WORDSTR ; V : VECTOR ) ;

     begin
     writeln ( LP, NAME,V[1]:18,'   ', V[2]:18,'   ', V[3]:18 )            ;
     writeln ( LP )                                                        ;
     end ;

end ;  { module MANHDUMP & File 'Manhdump.I' }
```

1.9. Simulation Log

```
$ page $  { begin File 'Manhslog.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHSLOG ;  { Subject :  Simulation Log }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILMATH ,
     UTILSPIF ,
     UTILVEMQ ,
     MANHMISC ,
     MANHBODS ,
     MANHFCON ;

export                                { begin externally visible declarations }

     const

          MAXSLOGRECS = 1501 ;

     type

          SLOGREC =
               record
               case integer of

                    1: ( SIMDESCRIP    : LINESTR   ;
                         PROGSESSID    : LINESTR   ;
                         LOC_N_OB      : VECTOR    ;
                         LOC_OCM_OS    : VECTOR    ;
                         LOC_PCM_PS    : VECTOR    ;
                         NUMSLOGRECS   : integer   ) ;

                    2: ( TIME          : longreal  ;
                         desmode       : CNTRLMODE ;
                         curmode       : CNTRLMODE ;
                         CFORC_PH_PB   : VECTOR    ;
                         CTORQ_PH_PB   : VECTOR    ;
                         FLEXFORCMAG   : longreal  ;
                         FLEX_RO_R     : VECTOR    ;
                         ANGMO_P_I     : VECTOR    ;
                         WOBBLE_CONE_P : longreal  ; { orientation of PL ang  }
                         WOBBLE_CLOK_P : longreal  ; { momentum wrt PB axes   }
                         IPOS_OCM_I    : VECTOR    ;
                         IPOS_PCM_I    : VECTOR    ;
                         IPOS_RO_I     : VECTOR    ;
                         PYR_I_OB      : EULPYR    ;
                         PYR_I_PB      : EULPYR    ;
                         PYR_I_R       : EULPYR    ) ;

               end ; { case & record }

          SLOGARR = array [ 0..MAXSLOGRECS ] of SLOGREC ;
```

```
$ page $

     var

          curslogrec  : integer  ;     { current simulation log index          }
          PRTSLOGREC  : integer  ;     { 1st index for printing flex/fcon log }
          SLOG        : ^SLOGARR ;     { pointer to simulation log             }

     procedure STORE_SIMULATION_LOG_RECORD ;
     procedure SAVE_SIMULATION_LOG         ;

implement                      { begin externally invisible part of module }


procedure STORE_SIMULATION_LOG_RECORD ;

     var

          ANGMO_P_PB : VECTOR     ;
          QUAT_I_R   : QUATERNION ;
          NORMANGMO  : longreal   ;

     begin
     curslogrec := curslogrec + 1                                              ;
     with SLOG^[curslogrec], STATE[orbiter] do
          begin
          QUAT_I_R     := QXQ(  QUAT_I_B, QUAT_OB_R )                          ;
          QUAT_OB_PB   := QCXQ( QUAT_I_B, STATE[payload].QUAT_I_B )            ;
          VBASE_OB_PB  := QMAT( QUAT_OB_PB )                                   ;
          IPOS_OCM_I   := IPOS_CM_I                                            ;
          PYR_I_OB     := QPYR( QUAT_I_B )                                     ;
          PYR_I_R      := QPYR( QUAT_I_R )                                     ;
          end ;
     with SLOG^[curslogrec], STATE[payload], INFO[payload]  do
          begin
          ANGMO_P_PB     := VXM( IRATE_B_B, RNERT_CM_B )                       ;
          TIME           := SIMTIME                                            ;
          desmode        := desired_mode                                       ;
          curmode        := current_mode                                       ;
          CFORC_PH_PB    := CFORC_H_B                                          ;
          CTORQ_PH_PB    := CTORQ_H_B                                          ;
          ANGMO_P_I      := VXMT(  ANGMO_P_PB, VBASE_I_B )                     ;
          NORMANGMO      := sqrt(  sqr(ANGMO_P_PB[2]) + sqr(ANGMO_P_PB[3]) )   ;
          WOBBLE_CONE_P  := ATAN2( NORMANGMO      ,  ANGMO_P_PB[1] )           ;
          WOBBLE_CLOK_P  := ATAN2( ANGMO_P_PB[2] , -ANGMO_P_PB[3] )            ;
          IPOS_PCM_I     := IPOS_CM_I                                          ;
          PYR_I_PB       := QPYR( QUAT_I_B )                                   ;
          end ;
     with SLOG^[curslogrec], STATE[crewman] do
          begin
          FLEXFORCMAG := VMAG( FORC_CRO_R )                                    ;
          FLEX_RO_R   := VDIF( OBPOS_RO_R, OBPOS_N_R )                         ;
          IPOS_RO_I   := VSUM( IPOS_OCM_I, IROT( OBPOS_RO_R, QUAT_I_R ) )      ;
          end ;
     SLOG^[0].NUMSLOGRECS := curslogrec                                        ;
     end ;
```

```
$ page $

procedure SAVE_SIMULATION_LOG ;

     var

          FILENAME : WORDSTR         ;
          i        : integer         ;
          OKAY     : boolean         ;
          SLOGFILE : file of SLOGREC ;

     begin
     repeat
          repeat
               FILENAME := WORD_INPUT( 'Name for simulation log file', '' )   ;
               if FILENAME = ''
                    then OKAY := USER_DECIDES_TO( 'Abandon simulation log' )
                    else OKAY := true                                           ;
               until OKAY ;
          if FILENAME <> '' then
               begin
               try                             { set trap for possible error }
                    rewrite ( SLOGFILE, FILENAME )                              ;

                    recover                 { come here after error, if any }
                    begin
                    OKAY := false                                               ;
                    SHOWLN ( 'Unable to open simulation log file' )             ;
                    SOUND_ALARM                                                 ;
                    end ;

               if OKAY then
                    try                        { set trap for possible error }
                         for i := 0 to curslogrec do
                              write ( SLOGFILE, SLOG^[i] )                      ;
                         close ( SLOGFILE, 'SAVE' )                             ;
                         writeln (LP,'Simulation log saved under the name "',
                                        FILENAME,'"')                           ;

                         recover            { come here after error, if any }
                         begin
                         OKAY := false                                          ;
                         SHOWLN ('Not enough disk space for simulation log')    ;
                         SOUND_ALARM                                            ;
                         end ;

               end ;

          until OKAY ;
     end ;

end ;  { module MANHSLOG & File 'Manhslog.I' }
```

1.10. IMI Simulated Cameras

```
$ page $  { begin File 'Manhkams.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHKAMS ;  { Subject :  IMI Simulated Cameras }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILMATH ,
     UTILSPIF ,
     UTILVEMQ ,
     MANHMISC ,
     MANHSLOG ;

export                                   { begin externally visible declarations }

     type

          KAMERA = (  { Camera (identified by location).  PAW & SAW are fixed }
                      { to Orbiter structure; all others are fixed in I frame.}
               AHEAD  ,                 { ahead of Payload                    }
               BEHIND ,                 { behind Payload                      }
               LEFT   ,                 { on left (port) side of Payload      }
               RIGHT  ,                 { on right (starboard) side of Payload }
               OVER   ,                 { over (above) Payload                }
               UNDER  ,                 { under (beneath) Payload             }
               PAW    ,                 { port aft window                     }
               SAW    ) ;               { starboard aft window                }

          KAMARR = array [ KAMERA ] of
               record
               HORFOV : longreal ;
               case integer of

                    1 : ( LOC_KO_I   : VECTOR     ;
                          QUAT_I_K   : QUATERNION ) ;

                    2 : ( LOC_KO_OB  : VECTOR     ;
                          QUAT_OB_K  : QUATERNION ) ;

               end ; { case & record }

     var

          current_kamera : KAMERA   ;
          DESIRED_KAMERA : KAMERA   ;
          KAMINFO        : KAMARR   ;
          REMOFFSET      : integer  ; { remote camera dist (ft) from I origin }
          REMHORFOV      : integer  ; { remote cam horizntl fld of view (deg) }

     procedure SET_UP_KAMERA_DATA ;

implement                          { begin externally invisible part of module }
```

```
$ page $

     type

          KAMLOCARR = array [ KAMERA ] of VECTOR ;
          KAMATTARR = array [ KAMERA ] of EULPYR ;

     const

          KAMLOC = KAMLOCARR [
               VECTOR [    1.0L0,    0.0L0,    0.0L0 ] { AHEAD  (_,I)  } ,
               VECTOR [   -1.0L0,    0.0L0,    0.0L0 ] { BEHIND (_,I)  } ,
               VECTOR [    0.0L0,   -1.0L0,    0.0L0 ] { LEFT   (_,I)  } ,
               VECTOR [    0.0L0,    1.0L0,    0.0L0 ] { RIGHT  (_,I)  } ,
               VECTOR [    0.0L0,    0.0L0,   -1.0L0 ] { OVER   (_,I)  } ,
               VECTOR [    0.0L0,    0.0L0,    1.0L0 ] { UNDER  (_,I)  } ,
               VECTOR [  572.0L0,  -15.0L0,  480.0L0 ] { PAW * (in,OS) } ,
               VECTOR [  572.0L0,   15.0L0,  480.0L0 ] { SAW * (in,OS) } ] ;

                    { * NOTE:  4" forward of fwd bulkhead of cargo bay.  }
                    {          IMI roundoff error produces jittery       }
                    {          images of window edges if locations are   }
                    {          set much farther forward than this.       }

          KAMATT = KAMATTARR [
               EULPYR [    0.0L0,  180.0L0,    0.0L0 ] { AHEAD  (deg,I ) } ,
               EULPYR [    0.0L0,    0.0L0,    0.0L0 ] { BEHIND (deg,I ) } ,
               EULPYR [    0.0L0,   90.0L0,    0.0L0 ] { LEFT   (deg,I ) } ,
               EULPYR [    0.0L0,  -90.0L0,    0.0L0 ] { RIGHT  (deg,I ) } ,
               EULPYR [  -90.0L0,    0.0L0,  180.0L0 ] { OVER   (deg,I ) } ,
               EULPYR [   90.0L0,    0.0L0,  180.0L0 ] { UNDER  (deg,I ) } ,
               EULPYR [    0.0L0,  180.0L0,    0.0L0 ] { PAW    (deg,OB) } ,
               EULPYR [    0.0L0,  180.0L0,    0.0L0 ] { SAW    (deg,OB) } ] ;
```

```
$ page $

procedure SET_UP_KAMERA_DATA ;

     var

          DIST        : longreal ;
          kam         : KAMERA   ;
          KPOS_N_OB   : VECTOR   ;
          TEMVEC      : VECTOR   ;
          TEST        : longreal ;
          VBASE_K_OB  : MAT3X3   ;

     begin
     REMOFFSET := INTEGER_INPUT( 'Camera offset   (ft)', REMOFFSET, 6 )        ;
     REMHORFOV := INTEGER_INPUT( 'Horizontal FOV (deg)', REMHORFOV, 6 )        ;
     for kam := AHEAD to UNDER do
          with KAMINFO[kam] do
               begin
               HORFOV   := RADPERDEG  *  REMHORFOV                             ;
               DIST     := REMOFFSET                                           ;
               LOC_KO_I := SXV(  DIST,   KAMLOC[kam]    )                      ;
               QUAT_I_K := PYRQ( EULRAD( KAMATT[kam] ) )                       ;
               end ;
     for kam := PAW to SAW do
          with KAMINFO[kam], SLOG^[0] do
               begin
               HORFOV       := RADPERDEG  *  83                                ;
               LOC_KO_OB    := VXD(  VDIF( KAMLOC[kam], LOC_OCM_OS ),
                                     STRUCZBODY                      )         ;
               KPOS_N_OB       := VDIF( LOC_N_OB , LOC_KO_OB     )             ;
               DIST            := VMAG( KPOS_N_OB )                            ;
               VBASE_K_OB[1]   := SXV(  ONE/DIST , KPOS_N_OB     )             ;
               TEMVEC          := CRSP( ZUNVEC    , VBASE_K_OB[1] )            ;
               TEST            := VMAG( TEMVEC )                               ;
               if TEST > ZERO
                    then VBASE_K_OB[2] := SXV(   ONE/TEST, TEMVEC )
                    else VBASE_K_OB[2] := VDIF( ZERVEC    , YUNVEC )           ;
               VBASE_K_OB[3] := CRSP(  VBASE_K_OB[1], VBASE_K_OB[2] )          ;
               QUAT_OB_K       := IMATQ( VBASE_K_OB )                          ;
               end ;
     end ;


end ;  { module MANHKAMS & File 'Manhkams.I' }
```

1.11. Simulation Runtime Displays

```
$ page $  { begin File 'Manhdisp.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHDISP ;  { Subject :  Simulation Runtime Displays }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILMATH ,
     UTILSPIF ,
     UTILVEMQ ,
     MANHMISC ,
     MANHSPIF ,
     MANHBODS ,
     MANHFCON ,
     MANHSLOG ,
     MANHKAMS ;

export                              { begin externally visible declarations }

     var

          DIGSHOWTINC : longreal ;
          PLIDENT     : NAMESTR  ;
          SIMPERREAL  : longreal ;
          USING_IMI   : boolean  ;

     procedure INITIALIZE_IMI        ;
     procedure UPDATE_DISPLAYS       ;
     procedure UPDATE_IMI_DATA       ;
     procedure PLAY_BACK_VISUAL_DATA ;

implement                           { begin externally invisible part of module }

     const

          LOC_CGO_R  = VECTOR [   -3.0L0,  0.0L0,   0.0L0 ] ;            { ft }
          LOC_OGO_OS = VECTOR [ 1000.0L0,  0.0L0, 400.0L0 ] ;            { in }

          UNISCALE   = 1L+6 ;   { scale factor; unit vectors & angles         }
          POSSCALE   = 1L+3 ;   { scale factor; positions other than of earth }
          GEOSCALE   = 1L-3 ;   { scale factor; position of earth             }

                                { NOTE:  Use of the above scale factors is    }
                                {        made necessary by an IMI deficiency  }
                                {        that prevents the transmission of    }
                                {        real (or longreal) numbers to the    }
                                {        IMI graphics system via the IEEE-488 }
                                {        interface.  To get around this prob- }
                                {        lem, such data are transmitted as    }
                                {        32-bit binary integers representing  }
                                {        the indicated multiples of the real  }
                                {        values.                              }

     type

          VALTYPE   = (   { designation of data type being transmitted to IMI }
               ascii ,                      { char    (1 byte  per value) }
```

```
          intgr ) ;                          { integer (4 bytes per value) }

     VALINTARR = array [ VALTYPE ] of integer ;

  var

     BYTESPERVAL : VALINTARR  ;
     IPOS_X_I    : VECTOR     ;
     LOC_CM_G    : array [ OBJECT ] of VECTOR  ;
     NAME        : array [ OBJECT ] of NAMESTR ;
     NUMBYTES    : integer    ;
     NUMVALS     : integer    ;
     OUTSTRING   : string[40] ;
     PYR_I_X     : EULPYR     ;
     valkind     : VALTYPE    ;
     VALTYPEKODE : VALINTARR  ;

     OUT : record
           case integer of
           1 : ( INT  : array [0..9] of integer      ) ;
           2 : ( BYTE : packed array [1..40] of char )
           end ; { case & record }

{ procedure INITIALIZE_IMI                                          ; EXPORT ; }
  function  STREIGHT(     NAME  : NAMESTR   )            : NAMESTR ;forward ;
  procedure SHOW_IMI_INITIALIZATION_MESSAGE                        ;forward ;
  procedure SHOW_1ST_SET_OF_IMI_INITIALIZATION_INSTRUCTIONS        ;forward ;
  procedure SHOW_2ND_SET_OF_IMI_INITIALIZATION_INSTRUCTIONS        ;forward ;
  procedure PUT_COORDINATE_TRANSFORMATION_MATRIX_IN_BUFFER         ;forward ;
  function  TOTNUMBYTES(  vkind : VALTYPE   ;
                          NVALS : integer   )            : integer ;forward ;
  function  FIRST4BYTES(  vkind : VALTYPE   ;
                          NVALS : integer   )            : integer ;forward ;
  procedure PUT_SUNPOS_AND_NUMOBJECTS_IN_BUFFER                    ;forward ;
  procedure PUT_OBJECT_NAMES_IN_BUFFER                             ;forward ;
  procedure PUT_CM_OFFSETS_IN_BUFFER                               ;forward ;
{ procedure UPDATE_DISPLAYS                                         ; EXPORT ; }
{ procedure UPDATE_IMI_DATA                                         ; EXPORT ; }
  procedure PUT_KAMERA_UPDATE_IN_BUFFER                            ;forward ;
  procedure PUT_EARTH_UPDATE_IN_BUFFER                             ;forward ;
  procedure PUT_ORBITER_UPDATE_IN_BUFFER                           ;forward ;
  procedure PUT_CREWMAN_UPDATE_IN_BUFFER                           ;forward ;
  procedure PUT_PAYLOAD_UPDATE_IN_BUFFER                           ;forward ;
  procedure PUT_UPDATE_IN_BUFFER ( LENSCALE : longreal ;
                                   KAMDATA  : boolean  )           ;forward ;
{ procedure PLAY_BACK_VISIUAL_DATA                                  ; EXPORT ; }
  procedure SHOW_PLAYBACK_CONTROL_KEY_MAP                          ;forward ;
  function  USER_WANTS_TO_CONTINUE                       : boolean ;forward ;


procedure INITIALIZE_IMI ;

  var

     BYPASS : boolean ;
     OKAY   : boolean ;

  begin
```

```
     BYTESPERVAL[ascii] := 1                                                   ;
     BYTESPERVAL[intgr] := 4                                                   ;
     VALTYPEKODE[ascii] := 2                                                   ;
     VALTYPEKODE[intgr] := 4                                                   ;
     SHOW_IMI_INITIALIZATION_MESSAGE                                           ;
     BYPASS := USER_DECIDES_TO( 'Bypass IMI initialization' )                  ;
     if not BYPASS then
          begin
          SHOW_1ST_SET_OF_IMI_INITIALIZATION_INSTRUCTIONS                      ;
          repeat
               OKAY := USER_DECIDES(
                              'First part of IMI initialization complete' )    ;
               until OKAY ;
          with SLOG^[0] do
               begin
               NAME[orbiter]      := STREIGHT( 'orbiter' )                     ;
               NAME[payload]      := STREIGHT(  PLIDENT  )                     ;
               NAME[crewman]      := STREIGHT( 'man0'    )                     ;
               LOC_CM_G[orbiter] := VXD( VDIF( LOC_OCM_OS, LOC_OGO_OS ),
                                         STRUC2BODY )                          ;
               LOC_CM_G[payload] := VXD( LOC_PCM_PS, STRUC2BODY )              ;
               LOC_CM_G[crewman] := VDIF( ZERVEC, LOC_CGO_R )     { "CM"=RO } ;
               end ;
          RESET_IMI_BUFFER                                                     ;
          PUT_COORDINATE_TRANSFORMATION_MATRIX_IN_BUFFER                       ;
          PUT_SUNPOS_AND_NUMOBJECTS_IN_BUFFER                                  ;
          PUT_OBJECT_NAMES_IN_BUFFER                                           ;
          PUT_CM_OFFSETS_IN_BUFFER                                             ;
          TRANSFER_BUFFER_CONTENTS_TO_IMI                                      ;
          SHOW_2ND_SET_OF_IMI_INITIALIZATION_INSTRUCTIONS                      ;
          repeat
               OKAY := USER_DECIDES(
                              'Second part of IMI initialization complete' )   ;
               until OKAY ;
          end ;

     end ;


function STREIGHT ( NAME : NAMESTR ) : NAMESTR ;

     begin
     while strlen( NAME ) < 8 do NAME := NAME + ' '                            ;
     setstrlen( NAME, 8 )                                                      ;
     STREIGHT := NAME                                                          ;
     end ;
```

```
$ page $

procedure SHOW_IMI_INITIALIZATION_MESSAGE ;

     type

          L      = string[55] ;
          MSGARR = array [ 1..4 ] of L ;

     const

          MSGLINE = MSGARR [
                 {            1          2          3          4          5
                 {1234567890123456789012345678901234567890123456789012345}
          { 1} L['IMI initialization is required if it has not been done '],
          { 2} L['already, or if the Payload name or the Orbiter center  '],
          { 3} L['of mass location has been changed since the previous   '],
          { 4} L['initialization.                                         ']] ;

     var

          i : integer ;

     begin
     CLEAR_SCREEN                                                         ;
     SHOWLN ( '' )                                                        ;
     for i := 1 to 4 do SHOWLN ( FILL12+MSGLINE[i] )                      ;
     SHOWLN ( '' )                                                        ;
     end ;
```

```
$ page $

procedure SHOW_1ST_SET_OF_IMI_INITIALIZATION_INSTRUCTIONS ;

     type

          L      = string[55] ;
          MSGARR = array [ 1..19 ] of L ;

     const

          MSGLINE = MSGARR [
                 {          1         2         3         4         5
                 {1234567890123456789012345678901234567890123456789012345}
          { 1} L['The first part of the IMI initialization process con-  '],
          { 2} L['sists of making the following entries at the IMI repeat'],
          { 3} L['IMI terminal (NOT this one).                           '],
          { 4} L['                                                       '],
          { 5} L['1.  If you have not already done it, log in to the IMI '],
          { 6} L['    operating system.  Otherwise, press the <BREAK> key'],
          { 7} L['    to terminate any IMI program currently running.    '],
          { 8} L['                                                       '],
          { 9} L['2.  Enter the following two command lines at the IMI   '],
          {10} L['    terminal:                                          '],
          {11} L['                                                       '],
          {12} L['         > cd /b/milmu                                 '],
          {13} L['         > bldmilmu.cmd                                '],
          {14} L['                                                       '],
          {15} L['Be careful about spacing in these commands; it is im-  '],
          {16} L['portant.  You do not have to type the ">" symbols; they'],
          {17} L['are prompting characters supplied by the IMI operating '],
          {18} L['system when it is read to accept an input.  All command'],
          {19} L['lines must be terminated by pressing the <ENTER> key.  ']] ;

     var

          i : integer ;

     begin
     CLEAR_SCREEN                                                          ;
     SHOWLN ( '' )                                                         ;
     for i := 1 to 19 do SHOWLN ( FILL12+MSGLINE[i] )                      ;
     SHOWLN ( '' )                                                         ;
     end ;
```

```
$ page $

procedure SHOW_2ND_SET_OF_IMI_INITIALIZATION_INSTRUCTIONS ;

     type

          L      = string[55] ;
          MSGARR = array [ 1..21 ] of L ;

     const

          MSGLINE = MSGARR [
                {          1         2         3         4         5
                {1234567890123456789012345678901234567890123456789012345}
          { 1} L['MANHANDLE has transmitted its initialization data to   '],
          { 2} L['the IMI, which is now executing the "bldmilmu" program.'],
          { 3} L['When "bldmilmu" terminates execution, the IMI operating'],
          { 4} L['system will display ">" on the IMI terminal to signal  '],
          { 5} L['that it is ready for another command.                  '],
          { 6} L['                                                       '],
          { 7} L['When you see the ">" character, it is time to start the'],
          { 8} L['second part of the initialization process.  This is    '],
          { 9} L['done by entering the command line                      '],
          {10} L['                                                       '],
          {11} L['         > runmilmu.cmd                                '],
          {12} L['                                                       '],
          {13} L['at the IMI terminal.  This causes execution of the     '],
          {14} L['"runmilmu" program which, first of all, fills the IMI  '],
          {15} L['terminal screen with a great many long lines of data   '],
          {16} L['whose meaning is of no concern here.                   '],
          {17} L['                                                       '],
          {18} L['The second part of the IMI initialization process is   '],
          {19} L['complete when "runmilmu" stops sending data to the IMI '],
          {20} L['terminal screen (i.e., when the cursor on the IMI ter- '],
          {21} L['minal screen becomes stationary).                      ']] ;

     var

          i : integer ;

     begin
     CLEAR_SCREEN                                                          ;
     SHOWLN ( '' )                                                         ;
     for i := 1 to 21 do SHOWLN ( FILL12+MSGLINE[i] )                      ;
     SHOWLN ( '' )                                                         ;
     end ;
```

```
$ page $

procedure PUT_COORDINATE_TRANSFORMATION_MATRIX_IN_BUFFER ;

     var

          i : integer ;
          j : integer ;
          k : integer ;

     begin
     valkind     := INTGR                                                    ;
     NUMVALS     :=     9                                                    ;
     NUMBYTES    := TOTNUMBYTES( valkind, NUMVALS )                          ;
     OUT.INT[0]  := FIRST4BYTES( valkind, NUMVALS )                          ;
     for i := 1 to 3 do
          for j := 1 to 3 do
               OUT.INT[3*(i-1)+j] := round( UNISCALE*IDN3X3[i,j] )           ;

                      { IDN3X3 = 3x3 identity matrix (see UTILVEMQ module). }
                      { Normally, the Mean_of_1950.0 to Mean_of_date        }
                      { coordinate transformation matrix would be sent      }
                      { to assure correct mapping of stars into camera      }
                      { images, but star positions are irrelevant in this   }
                      { simulation.                                         }

     setstrlen( OUTSTRING, NUMBYTES )                                        ;
     for k := 1 to NUMBYTES do OUTSTRING[k] := OUT.BYTE[k]                   ;
     ADD_STRING_TO_IMI_BUFFER ( OUTSTRING )                                  ;
     end ;


function  TOTNUMBYTES( vkind : VALTYPE ; NVALS : integer ) : integer ;

     begin
     TOTNUMBYTES := 4 + NVALS * BYTESPERVAL[vkind]                           ;
     end ;


function  FIRST4BYTES( vkind : VALTYPE ; NVALS : integer ) : integer ;

     var

          FIRST2BYTES : 0..65535 ; { number of bytes following these two }

     begin
     FIRST2BYTES := TOTNUMBYTES( vkind, NVALS ) - 2                          ;
     FIRST4BYTES := 65536 * FIRST2BYTES        +           { bytes 1 & 2 }
                      256 * VALTYPEKODE[vkind] +           { byte  3     }
                             NVALS                         { byte  4     } ;
     end ;
```

```
$ page $

procedure PUT_SUNPOS_AND_NUMOBJECTS_IN_BUFFER ;

     var

          i      : integer ;
          k      : integer ;
          SUNPOS : VECTOR  ;

     begin
     SUNPOS[1]   := -sqrt( TWO ) / TWO      { form unit vector defining dummy } ;
     SUNPOS[2]   :=  ZERO                   { direction to sun that will keep } ;
     SUNPOS[3]   :=  SUNPOS[1]              { it (the sun) out of sight       } ;
     valkind     :=  INTGR                                                       ;
     NUMVALS     :=     4                                                        ;
     NUMBYTES    := TOTNUMBYTES( valkind, NUMVALS )                              ;
     OUT.INT[0]  := FIRST4BYTES( valkind, NUMVALS )                              ;
     for i := 1 to 3 do
          OUT.INT[i] := round( UNISCALE*SUNPOS[i] )                              ;
     OUT.INT[4] := NUMOBJECTS                                                    ;
     setstrlen( OUTSTRING, NUMBYTES )                                            ;
     for k := 1 to NUMBYTES do OUTSTRING[k] := OUT.BYTE[k]                       ;
     ADD_STRING_TO_IMI_BUFFER ( OUTSTRING )                                      ;
     end ;


procedure PUT_OBJECT_NAMES_IN_BUFFER ;

     var

          k   : integer ;
          obj : OBJECT  ;

     begin
     valkind     := ASCII                                                        ;
     NUMVALS     :=     8                                                        ;
     NUMBYTES    := TOTNUMBYTES( valkind, NUMVALS )                              ;
     OUT.INT[0]  := FIRST4BYTES( valkind, NUMVALS )                              ;
     setstrlen( OUTSTRING, NUMBYTES )                                            ;
     for k := 1 to 4 do OUTSTRING[k] := OUT.BYTE[k]                              ;
     for obj := orbiter to crewman do
          begin
          for k := 1 to 8 do OUTSTRING[k+4] := NAME[obj,k]                       ;
          ADD_STRING_TO_IMI_BUFFER ( OUTSTRING )                                 ;
          end ;
     end ;
```

```
$ page $

procedure PUT_CM_OFFSETS_IN_BUFFER ;

     var

          i   : integer ;
          k   : integer ;
          obj : OBJECT  ;

     begin
     valkind    := INTGR                                                        ;
     NUMVALS    :=      3                                                       ;
     NUMBYTES   := TOTNUMBYTES( valkind, NUMVALS )                              ;
     OUT.INT[0] := FIRST4BYTES( valkind, NUMVALS )                              ;
     setstrlen( OUTSTRING, NUMBYTES )                                           ;
     for k := 1 to 4 do OUTSTRING[k] := OUT.BYTE[k]                             ;
     for obj := orbiter to crewman do
          begin
          for i := 1 to 3 do
               OUT.INT[i] := round( POSSCALE*LOC_CM_G[obj,i] )                  ;
          for k := 5 to NUMBYTES do OUTSTRING[k] := OUT.BYTE[k]                 ;
          ADD_STRING_TO_IMI_BUFFER ( OUTSTRING )                                ;
          end ;
     end ;
```

```
$ page $

procedure UPDATE_DISPLAYS ;

     var

          i   : integer ;
          K   : integer ;
          STR : LINESTR ;

     begin
     UPDATE_IMI_DATA                                                         ;
     for i := 1 to 3 do MOVE_UP                        { move cursor up 3 lines } ;
     with SLOG^[curslogrec] do
          begin
          setstrlen ( STR, 0 )                                               ;
          strwrite  ( STR,1,K,'Time = ',TIME:8:3,' sec    ' )                ;
          strwrite  ( STR,K,K,'Control mode:  ' )                            ;
          strwrite  ( STR,K,K,'Desired = ',CNTRLPAC[desmode],' ' )           ;
          strwrite  ( STR,K,K,'Current = ',CNTRLPAC[curmode],' ' )           ;
          SHOWLN    ( STR )                                                  ;
          SHOWLN    ( '' )                                                   ;
          setstrlen ( STR, 0 )                                               ;
          if curmode = FREE

               then
               strwrite ( STR,1,K,'PL Wobble Cone = ',
                                   DEGPERRAD*WOBBLE_CONE_P:6:2,' deg',
                                  '   Clok = ',
                                   DEGPERRAD*WOBBLE_CLOK_P:8:2,' deg',
                                  '   RMS Flex Force = ',
                                   FLEXFORCMAG:6:3,' lb'            )

               else
               begin
               strwrite ( STR,1,K,'Control:PB ' )                            ;
               for i := 1 to 3 do strwrite ( STR,K,K,CTORQ_PH_PB[i]:9:2 )    ;
               strwrite ( STR,K,K,' Torq  ' )                                ;
               for i := 1 to 3 do strwrite ( STR,K,K,CFORC_PH_PB[i]:9:3 )    ;
               strwrite ( STR,K,K,' Force' )                                 ;
               end ;

          SHOWLN ( STR )                                                     ;
          end ;
     end ;
```

```
$ page $

procedure UPDATE_IMI_DATA ;

      begin
      if USING_IMI then
           begin
           RESET_IMI_BUFFER                                                   ;
           PUT_KAMERA_UPDATE_IN_BUFFER                                        ;
           PUT_EARTH_UPDATE_IN_BUFFER                                         ;
           PUT_ORBITER_UPDATE_IN_BUFFER                                       ;
           PUT_PAYLOAD_UPDATE_IN_BUFFER                                       ;
           PUT_CREWMAN_UPDATE_IN_BUFFER                                       ;
           TRANSFER_BUFFER_CONTENTS_TO_IMI                                    ;
           end ;
      end ;


procedure PUT_KAMERA_UPDATE_IN_BUFFER ;

      var

           QUAT_I_OB : QUATERNION ;

      begin
      current_kamera := DESIRED_KAMERA                                        ;
      if ( current_kamera = PAW ) or ( current_kamera = SAW )

           then
           begin
           with KAMINFO[current_kamera], SLOG^[curslogrec] do
                begin
                QUAT_I_OB := PYRQ( PYR_I_OB )                                 ;
                IPOS_X_I  := VSUM( IPOS_OCM_I, IROT( LOC_KO_OB, QUAT_I_OB ) ) ;
                PYR_I_X   := QPYR( QXQ( QUAT_I_OB, QUAT_OB_K ) )              ;
                end ;
           end

           else
           begin
           with KAMINFO[current_kamera] do
                begin
                IPOS_X_I  := LOC_KO_I                                         ;
                PYR_I_X   := QPYR( QUAT_I_K )                                 ;
                end ;
           end ;

      PUT_UPDATE_IN_BUFFER ( POSSCALE, true )                                 ;
      end ;


procedure PUT_EARTH_UPDATE_IN_BUFFER ;

      begin
      IPOS_X_I := SXV( 1000000, IPOS_X_I      )    { put earth behind camera, } ;
      PUT_UPDATE_IN_BUFFER ( GEOSCALE, false )     { where it can't be seen   } ;
      end ;
```

```
$ page $

procedure PUT_ORBITER_UPDATE_IN_BUFFER ;

     begin
     with SLOG^[curslogrec] do
          begin
          IPOS_X_I := IPOS_OCM_I                                          ;
          PYR_I_X  := PYR_I_OB                                            ;
          end ;
     PUT_UPDATE_IN_BUFFER ( POSSCALE, false )                             ;
     end ;


procedure PUT_PAYLOAD_UPDATE_IN_BUFFER ;

     begin
     with SLOG^[curslogrec] do
          begin
          IPOS_X_I := IPOS_PCM_I                                          ;
          PYR_I_X  := PYR_I_PB                                            ;
          end ;
     PUT_UPDATE_IN_BUFFER ( POSSCALE, false )                             ;
     end ;


procedure PUT_CREWMAN_UPDATE_IN_BUFFER ;

     begin
     with SLOG^[curslogrec] do
          begin
          IPOS_X_I := IPOS_RO_I                                           ;
          PYR_I_X  := PYR_I_R                                             ;
          end ;
     PUT_UPDATE_IN_BUFFER ( POSSCALE, false )                             ;
     end ;
```

```
$ page $

procedure PUT_UPDATE_IN_BUFFER ( LENSCALE : longreal ; KAMDATA : boolean ) ;

     var

          i : integer ;
          k : integer ;

     begin
     valkind := INTGR                                                         ;
     if KAMDATA
          then NUMVALS  := 7
          else NUMVALS  := 6                                                  ;
     NUMBYTES   := TOTNUMBYTES( valkind, NUMVALS )                            ;
     OUT.INT[0] := FIRST4BYTES( valkind, NUMVALS )                            ;
     for i := 1 to 3 do
          OUT.INT[i] := round( LENSCALE*IPOS_X_I[i]  )                        ;
     for i := 4 to 6 do
          OUT.INT[i] := round( UNISCALE*PYR_I_X[i-3] )                        ;
     if KAMDATA then
          OUT.INT[7] := round( UNISCALE*KAMINFO[current_kameral].HORFOV )     ;
     setstrlen ( OUTSTRING, NUMBYTES )                                        ;
     for k := 1 to NUMBYTES do OUTSTRING[k] := OUT.BYTE[k]                    ;
     ADD_STRING_TO_IMI_BUFFER ( OUTSTRING )                                   ;
     end ;
```

```
$ page $

procedure PLAY_BACK_VISUAL_DATA ;

     var

          BASETICK       : integer    ;
          DIGSHOWTIME    : longreal   ;
          kam            : KAMERA     ;
          LASTREC        : integer    ;
          TICKSPERSIMSEC : longreal   ;
          TICK           : integer    ;
          TOCK           : integer    ;

     begin
     SIMPERREAL  := FIXED_INPUT( 'Sim/real time ratio: ', SIMPERREAL, 7, 4 )  ;
     DIGSHOWTINC := FIXED_INPUT( 'Digital display update interval (sec)',
                                 DIGSHOWTINC,7,3 )                            ;
     DIGSHOWTIME := DIGSHOWTINC                                               ;
     TICKSPERSIMSEC := TICKSPERSEC / SIMPERREAL                               ;
     SET_UP_KAMERA_DATA                                                       ;
     SUPPRESS_CURSOR                                                          ;
     SHOW_PLAYBACK_CONTROL_KEY_MAP                                            ;
     LASTREC     := SLOG^[0].NUMSLOGRECS                                      ;
     curslogrec := 1                                                          ;
     UPDATE_DISPLAYS                                                          ;
```

```
$ page $

     BASETICK := CLOCKTICK + 100                                          ;
     while USER_WANTS_TO_CONTINUE and (curslogrec<LASTREC) do
          begin
          curslogrec := curslogrec + 1                                    ;
          SIMTIME := SLOG^[curslogrec].TIME                               ;
          TOCK := round( TICKSPERSIMSEC * SIMTIME ) + BASETICK            ;
          repeat
               TICK := CLOCKTICK                                          ;
               until TICK >= TOCK                                         ;
          if curslogrec = LASTREC

               then
               UPDATE_DISPLAYS

               else
               if TICK < ( TOCK + 20 ) then

                    if DIGSHOWTINC = ZERO

                         then
                         UPDATE_IMI_DATA                       { only }

                         else
                         if SIMTIME < DIGSHOWTIME

                              then
                              UPDATE_IMI_DATA                  { only }

                              else
                              begin
                              UPDATE_DISPLAYS                             ;
                              DIGSHOWTIME := DIGSHOWTINC *
                                             (1+trunc(SIMTIME/DIGSHOWTINC))  ;
                              end ;

          end ;
     LOITER ( 2000 )                                                      ;
     RESTORE_CURSOR                                                       ;
     end ;
```

```
$ page $

procedure SHOW_PLAYBACK_CONTROL_KEY_MAP ;

    type

        L      = string[55] ;
        MSGARR = array [ 1..13 ] of L ;

    const

        MSGLINE = MSGARR [
                 {         1         2         3         4         5     }
                 {1234567890123456789012345678901234567890123456789012345}
        { 1} L['                    PLAYBACK CONTROLS                  '],
        { 2} L['                                                       '],
        { 3} L['             Desired                      Desired      '],
        { 4} L['       Key   Camera Loc             Key   Action       '],
        { 5} L['                                                       '],
        { 6} L['        a      ahead                 q   Stop playback '],
        { 7} L['        b      behind                                  '],
        { 8} L['        l      left                                    '],
        { 9} L['        r      right                                   '],
        {10} L['        o      over                                    '],
        {11} L['        u      under                                   '],
        {12} L['        p      port aft window                         '],
        {13} L['        s      stbd aft window                         ']] ;

    var

        i : integer ;

    begin
    CLEAR_SCREEN                                                       ;
    for i := 1 to  6 do SHOWLN ( '' )        { leave room for runtime data } ;
    for i := 1 to 13 do SHOWLN ( FILL12+MSGLINE[i] )                   ;
    SHOWLN ( '' )                                                      ;
    for i := 1 to 16 do MOVE_UP        { position cursor at top of screen } ;
    end ;
```

C-2

```
$ page $

function  USER_WANTS_TO_CONTINUE : boolean ;


     var

          CHINPUT : CHINPUTREC ;

     begin
     USER_WANTS_TO_CONTINUE := true                                          ;
     repeat
          CHINPUT := CHAR_INPUT( NOCHWAIT, NOCHECHO )                        ;
          if CHINPUT.Q = SOMETHING then
               case CHINPUT.C of
                    'a':  DESIRED_KAMERA := AHEAD                            ;
                    'b':  DESIRED_KAMERA := BEHIND                           ;
                    'l':  DESIRED_KAMERA := LEFT                             ;
                    'o':  DESIRED_KAMERA := OVER                             ;
                    'p':  DESIRED_KAMERA := PAW                              ;
                    'q':  USER_WANTS_TO_CONTINUE := false                    ;
                    'r':  DESIRED_KAMERA := RIGHT                            ;
                    's':  DESIRED_KAMERA := SAW                              ;
                    'u':  DESIRED_KAMERA := UNDER                            ;
                    otherwise SOUND_ALERT                                    ;
                    end ;  { case KBDCHAR }
          until CHINPUT.Q = NOTHING                                          ;
     end ;


end ;  { module MANHDISP & File 'Manhdisp.I' }
```

1.12. Simulation Control

```
$ page $  { begin File 'Manhscon.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHSCON ;  { Subject :  Simulation Control }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILMATH ,
     UTILSPIF ,
     UTILVEMQ ,
     MANHMISC ,
     MANHBODS ,
     MANHFCON ,
     MANHSLOG ,
     MANHKAMS ,
     MANHDISP ;

export                               { begin externally visible declarations }

     var

          CURTINC : longreal ;    { current value of time increment          }
          DESTINC : longreal ;    { desired time increment (supplied by user) }
          NOMSTEP : longreal ;    { nominal size of integration step         }
          OLDSTEP : longreal ;    { size of last step taken                  }

     procedure PROPAGATE_SYSTEM_STATE ;

implement                           { begin externally invisible part of module }

     const

          TINCTOL = 1.0L-6 ;           { tolerance on desired time increment }

     function  ANOTHER_STEP_IS_NEEDED : boolean ; forward ;
     procedure SHOW_RUNTIME_CONTROL_KEY_MAP     ; forward ;
     procedure ADVANCE_ONE_RK2_STEP             ; forward ;
     procedure COMPUTE_DERIVATIVES              ; forward ;
```

```
$ page $

procedure PROPAGATE_SYSTEM_STATE ;

     begin
     writeln ( LP, 'Time':8,'Stepsize':12  )                             ;
     writeln ( LP )                                                      ;
     OLDSTEP := ZERO                                                     ;
     PRTSLOGREC := curslogrec                                            ;
     CURTINC := ZERO                                                     ;
     SHOW_RUNTIME_CONTROL_KEY_MAP                                        ;
     SUPPRESS_CURSOR                                                     ;
     UPDATE_DISPLAYS                                                     ;
     while ANOTHER_STEP_IS_NEEDED do
          ADVANCE_ONE_RK2_STEP                                           ;
     LOITER ( 2000 )                                                     ;
     RESTORE_CURSOR                                                      ;
     end ;


function  ANOTHER_STEP_IS_NEEDED : boolean ;

     var

          CHINPUT : CHINPUTREC ;

     begin
     if ( (CURTINC+TINCTOL) >= DESTINC ) or ( (curslogrec+2) > MAXSLOGRECS )
          then ANOTHER_STEP_IS_NEEDED := false
          else ANOTHER_STEP_IS_NEEDED := true                            ;
     repeat
          CHINPUT := CHAR_INPUT( NOCHWAIT, NOCHECHO ) ;
          if CHINPUT.Q = SOMETHING then
               case CHINPUT.C of
                    '*':  NOMSTEP := NOMSTEP * TWO                       ;
                    '/':  NOMSTEP := NOMSTEP / TWO                       ;
                    'A':  desired_mode := ALIGN                          ;
                    'C':  desired_mode := CAPTURE                        ;
                    'D':  desired_mode := DESPIN                         ;
                    'F':  desired_mode := FREE                           ;
                    'H':  desired_mode := HOLD                           ;
                    'P':  desired_mode := PITCH                          ;
                    'R':  desired_mode := ROLL                           ;
                    'S':  desired_mode := SPINUP                         ;
                    'Y':  desired_mode := YAW                            ;
                    'a':  DESIRED_KAMERA := AHEAD                        ;
                    'b':  DESIRED_KAMERA := BEHIND                       ;
                    'l':  DESIRED_KAMERA := LEFT                         ;
                    'o':  DESIRED_KAMERA := OVER                         ;
                    'p':  DESIRED_KAMERA := PAW                          ;
                    'q':  ANOTHER_STEP_IS_NEEDED := false                ;
                    'r':  DESIRED_KAMERA := RIGHT                        ;
                    's':  DESIRED_KAMERA := SAW                          ;
                    'u':  DESIRED_KAMERA := UNDER                        ;
                    otherwise SOUND_ALERT                                ;
                    end ;  { case KBDCHAR }
          until CHINPUT.Q = NOTHING                                      ;
     end ;
```

```
$ page $

procedure SHOW_RUNTIME_CONTROL_KEY_MAP ;

     type

          L      = string[55] ;
          MSGARR = array [ 1..14 ] of L ;

     const

          MSGLINE = MSGARR [
                 {         1         2         3         4         5
                 {1234567890123456789012345678901234567890123456789012345}
          { 1} L['                    RUNTIME CONTROLS                   '],
          { 2} L['                                                       '],
          { 3} L['      Desired                 Desired             Desired '],
          { 4} L['Key  Cntrl Mode     Key   Camera Loc      Key   Action    '],
          { 5} L['                                                       '],
          { 6} L[' F    FREE           a     ahead           q   Stop sim '],
          { 7} L[' A    ALIGN          b     behind          *   Dble  step'],
          { 8} L[' D    DESPIN         l     left            /   Halve step'],
          { 9} L[' C    CAPTURE        r     right                     '],
          {10} L[' H    HOLD           o     over                      '],
          {11} L[' R    ROLL           u     under                     '],
          {12} L[' Y    YAW            p     port aft window           '],
          {13} L[' P    PITCH          s     stbd aft window           '],
          {14} L[' S    SPINUP                                         ']] ;

     var

          i : integer ;

     begin
     CLEAR_SCREEN                                                            ;
     for i := 1 to  6 do SHOWLN ( '' )        { leave room for runtime data } ;
     for i := 1 to 14 do SHOWLN ( FILL12+MSGLINE[i] )                          ;
     SHOWLN ( '' )                                                           ;
     for i := 1 to 17 do MOVE_UP        { position cursor at top of screen } ;
     end ;
```

```
$ page $

procedure ADVANCE_ONE_RK2_STEP ;

     var

          ANCHOR  : array [OBJECT] of STATEREC ;
          body    : OBJECT   ;
          HAFSTEP : longreal ;
          i       : integer  ;
          NUMVARS : integer  ;
          PASSNUM : integer  ;
          STEP    : longreal ;

     begin
     if ( CURTINC + 1.1L0 * NOMSTEP ) < DESTINC
          then STEP := NOMSTEP
          else STEP := DESTINC - CURTINC                                ;
     if abs(STEP-OLDSTEP) > 0.0005L0 then
          writeln ( LP, SIMTIME:8:3,STEP:12:3 )                         ;
     OLDSTEP := STEP                                                    ;
     HAFSTEP := STEP / TWO                                              ;
     for PASSNUM := 1 to 2 do
          begin
          COMPUTE_DERIVATIVES                                           ;
          for body := orbiter to crewman do
               begin
               if body = crewman
                    then NUMVARS := 6
                    else NUMVARS := STATESIZE                           ;
               if PASSNUM = 1

                    then
                    begin
                    ANCHOR[body] := STATE[body]                         ;
                    for i := 1 to NUMVARS do
                         STATE[body].ARR[i] := ANCHOR[body].ARR[i] +
                                        HAFSTEP * DERIV[body].ARR[i]    ;
                    end

                    else
                    for i := 1 to NUMVARS do
                         STATE[body].ARR[i] := ANCHOR[body].ARR[i] +
                                           STEP * DERIV[body].ARR[i]    ;

               if body <> crewman then
                    with STATE[body] do
                         QUAT_I_B := UNIQUAT( QUAT_I_B )                ;
               end ;

          CURTINC := CURTINC + HAFSTEP                                  ;
          SIMTIME := SIMTIME + HAFSTEP                                  ;
          STORE_SIMULATION_LOG_RECORD                                   ;
          UPDATE_DISPLAYS                                               ;
          end ;
     end ;
```

```
$ page $

procedure COMPUTE_DERIVATIVES ;

     var

          body         : OBJECT     ;
          EFFTORQ      : VECTOR     ;   { ft*lb, "effective" torque  about CM }
          ANGMO_body_B : VECTOR     ;
          MOMENT       : VECTOR     ;   { ft*lb, moment of CFORC_H_B about CM }
          Q            : QUATERNION ;

     begin
     COMPUTE_FLIGHT_CONTROL_FORCE_AND_TORQUE                                  ;
     for body := orbiter to payload do
          with DERIV[body], STATE[body], INFO[body] do
               begin
               FORC_CM_I    := VXMT( CFORC_H_B, VBASE_I_B )                   ;
               if body = orbiter
                    then MOMENT := CRSP( OBPOS_H_OB, CFORC_H_B )
                    else MOMENT := CRSP(    LOC_H_PB, CFORC_H_B )             ;
               TORQ_CM_B    := VSUM( CTORQ_H_B, MOMENT )                      ;
               IPOSDOT_CM_I := IVEL_CM_I                                      ;
               IVELDOT_CM_I := SXV( ONE/MASS, FORC_CM_I )                     ;
               Q.S          := 0                                              ;
               Q.V          := SXV( 0.5, IRATE_B_B )                          ;
               QUATDOT_I_B  := QXQ( QUAT_I_B, Q )                             ;
               ANGMO_body_B := VXM( IRATE_B_B, RNERT_CM_B )                   ;
               EFFTORQ      := VDIF( TORQ_CM_B,
                                     CRSP( IRATE_B_B,ANGMO_body_B  ) )        ;
               IRATEDOT_B_B := VXM( EFFTORQ, RALAC_CM_B )                     ;
               end ;
     with DERIV[crewman], STATE[crewman], INFO[crewman], SLOG^[curslogrec] do
          begin
          FORC_CRO_R    := VDIF( VXM(  INFO[orbiter].CFORC_H_B, VBASE_OB_R ),
                                 VSUM( VXM( FLEX_RO_R , TSPRG_CRO_R ),
                                       VXM( OBVEL_RO_R, TDAMP_CRO_R ) ) )     ;
          OBPOSDOT_RO_R := OBVEL_RO_R                                         ;
          OBVELDOT_RO_R := VXM( FORC_CRO_R, TALAC_CRO_R )                     ;
          end ;
     end ;

end ;  { module MANHSCON & File 'Manhscon.I' }
```

1.13. Printed Output Data

```
$ page $  { begin File 'Manhprnt.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHPRNT ;  { Subject :  Printed Output Data }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILMATH ,
     UTILSPIF ,
     UTILVEMQ ,
     MANHMISC ,
     MANHBODS ,
     MANHFCON ,
     MANHSLOG ;

export                                    { begin externally visible declarations }

     procedure DOCUMENT_THE_STATE_OF_THE_SYSTEM ;

implement                             { begin externally invisible part of module }

     var

          ANGMO_P_I     : VECTOR     ;
          ANGMO_P_PB    : VECTOR     ;
          IPOS_OCM_I    : VECTOR     ;
          IPOS_PCM_I    : VECTOR     ;
          IRATE_OB_OB   : VECTOR     ;
          IRATE_PB_PB   : VECTOR     ;
          IVEL_OCM_I    : VECTOR     ;
          IVEL_PCM_I    : VECTOR     ;
          K             : integer    ;
          OBPOS_PCM_OB  : VECTOR     ;
          OBRATE_PB_PB  : VECTOR     ;
          OBVEL_PCM_OB  : VECTOR     ;
          QUAT_I_OB     : QUATERNION ;
          QUAT_I_PB     : QUATERNION ;
          STR           : LINESTR    ;
          VBASE_I_OB    : MAT3X3     ;
          VBASE_I_PB    : MAT3X3     ;

     procedure PRINT_RMS_FLEXURE_AND_PAYLOAD_FLIGHT_CONTROL_LOG        ;forward;
     procedure PRINT_TIME_AND_CONTROL_MODE_AND_PAYLOAD_WOBBLE          ;forward;
     procedure PRINT_PAYLOAD_CONTROL_FORCE_AND_TORQUE                  ;forward;
     procedure PRINT_PAYLOAD_CONTROL_STATE                             ;forward;
     procedure PRINT_PAYLOAD_STATE_WRT_ORBITER_BODY_AXES               ;forward;
     procedure PRINT_PAYLOAD_STATE_WRT_INERTIAL_AXES                   ;forward;
     procedure PRINT_ORBITER_STATE_WRT_INERTIAL_AXES                   ;forward;
```

```
$ page $

procedure DOCUMENT_THE_STATE_OF_THE_SYSTEM ;

     begin
     COMPUTE_FLIGHT_CONTROL_FORCE_AND_TORQUE                                     ;
     VBASE_I_OB   :=  INFO[orbiter].VBASE_I_B                                    ;
     QUAT_I_OB    := STATE[orbiter].QUAT_I_B                                     ;
     IPOS_OCM_I   := STATE[orbiter].IPOS_CM_I                                    ;
     IVEL_OCM_I   := STATE[orbiter].IVEL_CM_I                                    ;
     IRATE_OB_OB  := STATE[orbiter].IRATE_B_B                                    ;
     VBASE_I_PB   :=  INFO[payload].VBASE_I_B                                    ;
     QUAT_I_PB    := STATE[payload].QUAT_I_B                                     ;
     IPOS_PCM_I   := STATE[payload].IPOS_CM_I                                    ;
     IVEL_PCM_I   := STATE[payload].IVEL_CM_I                                    ;
     IRATE_PB_PB  := STATE[payload].IRATE_B_B                                    ;
     ANGMO_P_PB   := VXM(  IRATE_PB_PB, RNERT_PCM_PB )                           ;
     ANGMO_P_I    := VXMT( ANGMO_P_PB , VBASE_I_PB   )                           ;
     OBPOS_PCM_OB :=       VXM( VDIF( IPOS_PCM_I, IPOS_OCM_I ), VBASE_I_OB )     ;
     OBVEL_PCM_OB := VDIF( VXM( VDIF( IVEL_PCM_I, IVEL_OCM_I ), VBASE_I_OB ),
                           CRSP( IRATE_OB_OB, OBPOS_PCM_OB )                   ) ;
     OBRATE_PB_PB := VDIF( IRATE_PB_PB , VXM( IRATE_OB_OB, VBASE_OB_PB ) )       ;

     if curslogrec > PRTSLOGREC then
          if USER_DECIDES_TO( 'Print RMS flexure & PL flt control log' ) then
               PRINT_RMS_FLEXURE_AND_PAYLOAD_FLIGHT_CONTROL_LOG                  ;

     CLEAR_SCREEN                                                                ;
     PRINT_TIME_AND_CONTROL_MODE_AND_PAYLOAD_WOBBLE                              ;
     PRINT_PAYLOAD_CONTROL_FORCE_AND_TORQUE                                      ;
     PRINT_PAYLOAD_CONTROL_STATE                                                 ;
     PRINT_PAYLOAD_STATE_WRT_ORBITER_BODY_AXES                                   ;
     PRINT_PAYLOAD_STATE_WRT_INERTIAL_AXES                                       ;
     PRINT_ORBITER_STATE_WRT_INERTIAL_AXES                                       ;
     end;
```

```
$ page $

procedure PRINT_RMS_FLEXURE_AND_PAYLOAD_FLIGHT_CONTROL_LOG ;

     var

          ENDSLOGREC : integer ;
          HEADERA    : LINESTR ;
          HEADERB    : LINESTR ;
          i          : integer ;
          n          : integer ;

     begin
     setstrlen ( HEADERA, 0 )                                                       ;
     setstrlen ( HEADERB, 0 )                                                       ;
     strwrite ( HEADERA,1,K,     'Time':7,    'zFlex':7 )                           ;
     strwrite ( HEADERA,K,K, 'DesMode':9, 'CurMode':8 )                             ;
     strwrite ( HEADERA,K,K,    'T_PBx':8,    'T_PBy':8,    'T_PBz':8 )             ;
     strwrite ( HEADERA,K,K,    'F_PBx':8,    'F_PBy':8,    'F_PBz':8 )             ;
     strwrite ( HEADERB,1,K,    '(sec)':7,     '(in)':7 )                           ;
     strwrite ( HEADERB,K,K,        ' ':9,        ' ':8 )                           ;
     strwrite ( HEADERB,K,K, '(ft*lb)':8, '(ft*lb)':8, '(ft*lb)':8 )                ;
     strwrite ( HEADERB,K,K,     '(lb)':8,     '(lb)':8,     '(lb)':8 )             ;

     START_NEW_PAGE                                                                 ;
     writeln ( LP, HEADERA )                                                        ;
     writeln ( LP, HEADERB )                                                        ;
     writeln ( LP )                                                                 ;
     n := PRTSLOGREC                                                                ;
     ENDSLOGREC := IMIN( curslogrec, n+18 )                                         ;
```

```
$ page $

    while n <= curslogrec do
          begin
          CLEAR_SCREEN                                                              ;
          SHOWLN ( HEADERA )                                                        ;
          SHOWLN ( HEADERB )                                                        ;
          SHOWLN ( '' )                                                             ;
          while n <= ENDSLOGREC do
                begin
                setstrlen ( STR, 0 )                                                ;
                with SLOG^[n] do
                      begin
                      strwrite ( STR,1,K,TIME:7:3,(12*FLEX_RO_R[3]):7:2,' '   )     ;
                      strwrite ( STR,K,K,CNTRLPAC[desmode],CNTRLPAC[curmode] )      ;
                      if curmode = FREE

                            then
                            strwrite ( STR,K,K,'   Wobble Cone ='           ,
                                                (DEGPERRAD*WOBBLE_CONE_P):8:2,
                                                ',  Clok ='                 ,
                                                (DEGPERRAD*WOBBLE_CLOK_P):8:2,
                                                ' (deg)'                    )

                            else
                            begin
                            for i := 1 to 3 do
                                  strwrite ( STR,K,K,CTORQ_PH_PB[i]:8:2 )           ;
                            for i := 1 to 3 do
                                  strwrite ( STR,K,K,CFORC_PH_PB[i]:8:2 )           ;
                            end ;

                      end ;
                SHOWLN  (      STR )                                                ;
                writeln ( LP, STR )                                                 ;
                n := n + 1                                                          ;
                end ;
          repeat
                { nothing }
                until USER_DECIDES_TO( 'Proceed' ) ;
          ENDSLOGREC := IMIN( curslogrec, n+17 )                                    ;
          end ;
    end ;
```

```
$ page $

procedure PRINT_TIME_AND_CONTROL_MODE_AND_PAYLOAD_WOBBLE ;

     var

          NOMSPINRATEdeg_P : longreal ;
          NORMANGMO        : longreal ;
          WOBBLE_CLOKdeg_P : longreal ;
          WOBBLE_CONEdeg_P : longreal ;

     begin
     NORMANGMO        := sqrt( sqr( ANGMO_P_PB[2] ) + sqr( ANGMO_P_PB[3] ) )  ;
     WOBBLE_CONEdeg_P := DEGPERRAD * ATAN2( NORMANGMO    ,  ANGMO_P_PB[1] )   ;
     WOBBLE_CLOKdeg_P := DEGPERRAD * ATAN2( ANGMO_P_PB[2], -ANGMO_P_PB[3] )   ;
     NOMSPINRATEdeg_P := DEGPERRAD * ( ANGMO_P_PB[1] / RNERT_PCM_PB[1,1] )    ;

     SHOWLN  ( '' )                                                            ;
     if curslogrec > 1 then START_NEW_PAGE                                     ;
     writeln ( LP )                                                            ;

     setstrlen ( STR, 0 )                                                      ;
     strwrite  ( STR,1,K,'Time = ',SIMTIME:8:3 )                               ;
     strwrite  ( STR,K,K,'DesMode = ':20,CNTRLPAC[desired_mode] )              ;
     strwrite  ( STR,K,K,'CurMode = ':17,CNTRLPAC[current_mode] )              ;
     SHOWLN  (     STR )                                                       ;
     writeln ( LP, STR )                                                       ;

     setstrlen ( STR, 0 )                                                      ;
     strwrite  ( STR,1,K,'PL Nomspin = '    ,NOMSPINRATEdeg_P:6:2 )            ;
     strwrite  ( STR,K,K,'Wobble Cone = ':20,WOBBLE_CONEdeg_P:6:2 )            ;
     strwrite  ( STR,K,K,'Wobble Clok = ':19,WOBBLE_CLOKdeg_P:7:2 )            ;
     SHOWLN  (     STR )                                                       ;
     writeln ( LP, STR )                                                       ;

     SHOWLN  ( '' )                                                            ;
     writeln ( LP )                                                            ;
     end ;
```

```
$ page $

procedure PRINT_PAYLOAD_CONTROL_FORCE_AND_TORQUE ;

     var

          CFORC_PH_PB : VECTOR  ;
          CFORC_PH_R  : VECTOR  ;
          CTORQ_PH_PB : VECTOR  ;
          CTORQ_PH_R  : VECTOR  ;
          i           : integer ;

     begin
     CTORQ_PH_PB := INFO[payload].CTORQ_H_B                                          ;
     CFORC_PH_PB := INFO[payload].CFORC_H_B                                          ;
     CFORC_PH_R  := VXMT( CFORC_PH_PB, VBASE_R_PB )                                  ;
     CTORQ_PH_R  := VXMT( CTORQ_PH_PB, VBASE_R_PB )                                  ;

     setstrlen ( STR, 0 )                                                            ;
     strwrite  ( STR,1,K,'Cntrl@H: R  ' )                                            ;
     for i := 1 to 3 do strwrite ( STR,K,K,CTORQ_PH_R[i]:9:2 )                       ;
     strwrite ( STR,K,K,' Torq  ' )                                                  ;
     for i := 1 to 3 do strwrite ( STR,K,K,CFORC_PH_R[i]:9:3 )                       ;
     strwrite ( STR,K,K,' Forc' )                                                    ;
     SHOWLN  (     STR )                                                             ;
     writeln ( LP, STR )                                                             ;

     setstrlen ( STR, 0 )                                                            ;
     strwrite  ( STR,1,K,'Cntrl@H:PB  ' )                                            ;
     for i := 1 to 3 do strwrite ( STR,K,K,CTORQ_PH_PB[i]:9:2 )                      ;
     strwrite ( STR,K,K,' Torq  ' )                                                  ;
     for i := 1 to 3 do strwrite ( STR,K,K,CFORC_PH_PB[i]:9:3 )                      ;
     strwrite ( STR,K,K,' Forc' )                                                    ;
     SHOWLN  (     STR )                                                             ;
     writeln ( LP, STR )                                                             ;

     SHOWLN  ( '' )                                                                  ;
     writeln ( LP )                                                                  ;
     end ;
```

```
$ page $

procedure PRINT_PAYLOAD_CONTROL_STATE ;

     var

          i                  : integer ;
          PDRATEdeg_PB_PB    : VECTOR  ;
          PYRdeg_PD_PB       : EULPYR  ;
          RPOS_H_R           : VECTOR  ;
          RVEL_H_R           : VECTOR  ;

     begin
     PYRdeg_PD_PB      := EULDEG( AERR )                                              ;
     PDRATEdeg_PB_PB   := SXV( DEGPERRAD, AVEL )                                      ;
     RPOS_H_R          := LERR                                                        ;
     RVEL_H_R          := LVEL                                                        ;

     setstrlen ( STR, 0 )                                                             ;
     strwrite  ( STR,1,K,'PB Axes:PD  ' )                                             ;
     for i := 1 to 3 do strwrite ( STR,K,K,PYRdeg_PD_PB[i]:9:2 )                      ;
     strwrite ( STR,K,K,' PYR    ' )                                                  ;
     for i := 1 to 3 do strwrite ( STR,K,K,PDRATEdeg_PB_PB[i]:9:3 )                   ;
     strwrite ( STR,K,K,' Rate' )                                                     ;
     SHOWLN  (      STR )                                                             ;
     writeln ( LP, STR )                                                              ;

     setstrlen ( STR, 0 )                                                             ;
     strwrite  ( STR,1,K,'PL Hndl: R  ' )                                             ;
     for i := 1 to 3 do strwrite ( STR,K,K,RPOS_H_R[i]:9:2 )                          ;
     strwrite ( STR,K,K,' Pos    ' )                                                  ;
     for i := 1 to 3 do strwrite ( STR,K,K,RVEL_H_R[i]:9:3 )                          ;
     strwrite ( STR,K,K,' Vel' )                                                      ;
     SHOWLN  (      STR )                                                             ;
     writeln ( LP, STR )                                                              ;

     SHOWLN  ( '' )                                                                   ;
     writeln ( LP )                                                                   ;
     end ;
```

```
$ page $

procedure PRINT_PAYLOAD_STATE_WRT_ORBITER_BODY_AXES ;

     var

          i                  : integer  ;
          OBRATEdeg_PB_PB    : VECTOR   ;
          PYRdeg_OB_PB       : EULPYR   ;

     begin
     PYRdeg_OB_PB      := EULDEG( QPYR( QUAT_OB_PB ) )                          ;
     OBRATEdeg_PB_PB   := SXV( DEGPERRAD, OBRATE_PB_PB )                         ;

     setstrlen ( STR, 0 )                                                        ;
     strwrite  ( STR,1,K,'PB Axes:OB  ' )                                        ;
     for i := 1 to 3 do strwrite ( STR,K,K,PYRdeg_OB_PB[i]:9:2 )                 ;
     strwrite ( STR,K,K,' PYR    ' )                                             ;
     for i := 1 to 3 do strwrite ( STR,K,K,OBRATEdeg_PB_PB[i]:9:3 )              ;
     strwrite ( STR,K,K,' Rate' )                                                ;
     SHOWLN  (      STR )                                                        ;
     writeln ( LP, STR )                                                         ;

     setstrlen ( STR, 0 )                                                        ;
     strwrite  ( STR,1,K,'PL CM   :OB   ' )                                      ;
     for i := 1 to 3 do strwrite ( STR,K,K,OBPOS_PCM_OB[i]:9:2 )                 ;
     strwrite ( STR,K,K,' Pos     ' )                                            ;
     for i := 1 to 3 do strwrite ( STR,K,K,OBVEL_PCM_OB[i]:9:3 )                 ;
     strwrite ( STR,K,K,' Vel' )                                                 ;
     SHOWLN  (      STR )                                                        ;
     writeln ( LP, STR )                                                         ;

     SHOWLN  ( '' )                                                              ;
     writeln ( LP )                                                              ;
     end ;


procedure PRINT_PAYLOAD_STATE_WRT_INERTIAL_AXES ;

     var

          i                 : integer  ;
          ANGMO_P_MAG       : longreal ;
          ANGMO_P_IPCHdeg   : longreal ;
          ANGMO_P_IYAWdeg   : longreal ;
          ANGMO_P_IZX       : longreal ;
          IRATEdeg_PB_PB    : VECTOR   ;
          PYRdeg_I_PB       : EULPYR   ;
          ROTKENERGY_P      : longreal ;
```

```
$ page $

    begin
    ROTKENERGY_P     := DOTP( IRATE_PB_PB, ANGMO_P_PB ) / TWO                    ;
    ANGMO_P_IZX      := sqrt( sqr( ANGMO_P_I[3] ) + sqr( ANGMO_P_I[1] ) )        ;
    ANGMO_P_IYAWdeg := DEGPERRAD * ATAN2(  ANGMO_P_I[2], ANGMO_P_IZX  )          ;
    ANGMO_P_IPCHdeg := DEGPERRAD * ATAN2( -ANGMO_P_I[3], ANGMO_P_I[1] )          ;
    ANGMO_P_MAG      := VMAG( ANGMO_P_I )                                        ;
    IRATEdeg_PB_PB  := SXV( DEGPERRAD, IRATE_PB_PB )                             ;
    PYRdeg_I_PB      := EULDEG( QPYR( QUAT_I_PB ) )                              ;

    setstrlen ( STR, 0 )                                                         ;
    strwrite  ( STR,1,K,'PL Rot K E  ' )                                         ;
    strwrite  ( STR,K,K,ROTKENERGY_P:9:2 )                                       ;
    SHOWLN  (     STR )                                                          ;
    writeln ( LP, STR )                                                          ;

    setstrlen ( STR, 0 )                                                         ;
    strwrite  ( STR,1,K,'PLAngmo: I  ' )                                         ;
    strwrite  ( STR,K,K,ANGMO_P_IPCHdeg:9:2 )                                    ;
    strwrite  ( STR,K,K,ANGMO_P_IYAWdeg:9:2 )                                    ;
    strwrite  ( STR,K,K,ANGMO_P_MAG:9:2 )                                        ;
    strwrite ( STR,K,K,' PYMag ' )                                               ;
    for i := 1 to 3 do strwrite ( STR,K,K,ANGMO_P_I[i]:9:3 )                     ;
    strwrite ( STR,K,K,' Hxyz' )                                                 ;
    SHOWLN  (     STR )                                                          ;
    writeln ( LP, STR )                                                          ;

    setstrlen ( STR, 0 )                                                         ;
    strwrite  ( STR,1,K,'PB Axes: I  ' )                                         ;
    for i := 1 to 3 do strwrite ( STR,K,K,PYRdeg_I_PB[i]:9:2 )                   ;
    strwrite ( STR,K,K,' PYR    ' )                                              ;
    for i := 1 to 3 do strwrite ( STR,K,K,IRATEdeg_PB_PB[i]:9:3 )                ;
    strwrite ( STR,K,K,' Rate' )                                                 ;
    SHOWLN  (     STR )                                                          ;
    writeln ( LP, STR )                                                          ;

    setstrlen ( STR, 0 )                                                         ;
    strwrite  ( STR,1,K,'PL CM   : I  ' )                                        ;
    for i := 1 to 3 do strwrite ( STR,K,K,IPOS_PCM_I[i]:9:2 )                    ;
    strwrite ( STR,K,K,' Pos    ' )                                              ;
    for i := 1 to 3 do strwrite ( STR,K,K,IVEL_PCM_I[i]:9:3 )                    ;
    strwrite ( STR,K,K,' Vel' )                                                  ;
    SHOWLN  (     STR )                                                          ;
    writeln ( LP, STR )                                                          ;

    SHOWLN  ( '' )                                                               ;
    writeln ( LP )                                                               ;
    end ;
```

```
$ page $

procedure PRINT_ORBITER_STATE_WRT_INERTIAL_AXES ;

     var

          i                : integer ;
          IRATEdeg_OB_OB : VECTOR  ;
          PYRdeg_I_OB      : EULPYR  ;

     begin
     IRATEdeg_OB_OB    := SXV( DEGPERRAD, IRATE_OB_OB )                              ;
     PYRdeg_I_OB       := EULDEG( QPYR( QUAT_I_OB ) )                                ;

     setstrlen ( STR, 0 )                                                           ;
     strwrite  ( STR,1,K,'OB Axes: I   ' )                                          ;
     for i := 1 to 3 do strwrite ( STR,K,K,PYRdeg_I_OB[i]:9:2 )                     ;
     strwrite ( STR,K,K,' PYR    ' )                                                ;
     for i := 1 to 3 do strwrite ( STR,K,K,IRATEdeg_OB_OB[i]:9:3 )                  ;
     strwrite ( STR,K,K,' Rate' )                                                   ;
     SHOWLN  (      STR )                                                           ;
     writeln ( LP, STR )                                                            ;

     setstrlen ( STR, 0 )                                                           ;
     strwrite  ( STR,1,K,'Orb CM : I   ' )                                          ;
     for i := 1 to 3 do strwrite ( STR,K,K,IPOS_OCM_I[i]:9:2 )                      ;
     strwrite ( STR,K,K,' Pos    ' )                                                ;
     for i := 1 to 3 do strwrite ( STR,K,K,IVEL_OCM_I[i]:9:3 )                      ;
     strwrite ( STR,K,K,' Vel' )                                                    ;
     SHOWLN  (      STR )                                                           ;
     writeln ( LP, STR )                                                            ;

     SHOWLN  ( '' )                                                                 ;
     for i := 1 to 3 do writeln ( LP )                                              ;
     end ;


end ;  { module MANHPRNT & File 'Manhprnt.I' }
```

1.14. Initialization of System State

```
$ page $  { begin File 'Manhinit.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHINIT ;  { Subject :  Initialization of System State }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILMATH ,
     UTILSPIF ,
     UTILVEMQ ,
     MANHMISC ,
     MANHBODS ,
     MANHFCON ,
     MANHSLOG ,
     MANHKAMS ,
     MANHDISP ,
     MANHSCON ,
     MANHPRNT ;

export                          { begin externally visible declarations }

     const

          FILESIZE = 24 ;

     type

          IDATREC =                    { contents of generic input data file }
               record
               PVID : NAMEPAC                          ; { prog version ident }
               ARR  : array[ 1..FILESIZE ] of longreal ; { numeric data       }
               end ; { record }

     procedure INITIALIZE_SYSTEM_STATE ;

implement                     { begin externally invisible part of module }
```

```
$ page $

     type

          ARMSREC =      { record containing RMS translational alacrity matrix }
               record
               case integer of

                    0 : ( IDAT         : IDATREC ) ;

                    1 : ( PVID         : NAMEPAC ;
                          TALAC_CRO_R : MAT3X3   ) ;

               end ;  { case & record }


          DRMSREC =     { record containing RMS transln'l damping const matrix }
               record
               case integer of

                    0 : ( IDAT         : IDATREC ) ;

                    1 : ( PVID         : NAMEPAC ;
                          TDAMP_CRO_R : MAT3X3   ) ;

               end ;  { case & record }


          SRMSREC =      { record containing RMS transln'l spring const matrix }
               record
               case integer of

                    0 : ( IDAT         : IDATREC ) ;

                    1 : ( PVID         : NAMEPAC ;
                          TSPRG_CRO_R : MAT3X3   ) ;

               end ;  { case & record }
```

```
$ page $

          NERTREC =        { record containing Orbiter or Payload inertia data }
               record
               case integer of

                    0 : ( IDAT        : IDATREC   ) ;

                    1 : ( PVID        :  NAMEPAC ;
                          WEIGHT      : longreal ;     { lb; in std gravity fld  }
                          LOC_CM_S    : VECTOR   ;     { in                      }
                          MOMNERT_S   : REALARR3 ;     { slug*ft*ft; Ixx,Iyy,Izz }
                          PRDNERT_S   : REALARR3 ) ;   { slug*ft*ft; Pxy,Pxz,Pyz }

                                   {                      Ixx    -Pxy    -Pxz     }
                                   {                                              }
                                   {  RNERT_CM_S  =     -Pxy     Iyy    -Pyz     }
                                   {                                              }
                                   {                     -Pxz    -Pyz     Izz     }

               end ; { case & record }


          POSIREC =        { record containing flt control position parameters }
               record
               case integer of

                    0 : ( IDAT         : IDATREC   ) ;

                    1 : ( PVID         : NAMEPAC   ;
                          LOC_N_OS     : VECTOR    ;           { in           }
                          LOC_H_PS     : VECTOR    ;           { in           }
                          LERRLIMin    : REALARR3 ;            { in           }
                          LERRTOLin    : REALARR3 ;            { in           }
                          NOMLVELin    : longreal ;            { in/sec       }
                          LVELTOLpct   : longreal ) ;          { % of NOMLVEL }

               end ; { case & record }


          ATTIREC =       { record containing flt control attitude parameters   }
               record
               case integer of

                    0 : ( IDAT           : IDATREC   ) ;

                    1 : ( PVID           : NAMEPAC   ;
                          PYRdeg_OB_R    : EULPYR    ;         { deg          }
                          PYRdeg_R_PD    : EULPYR    ;         { deg          }
                          AERRMAXdeg     : REALARR3 ;          { deg          }
                          AERRMINdeg     : REALARR3 ;          { deg          }
                          AERRTOLdeg     : REALARR3 ;          { deg          }
                          NOMAVELdeg     : REALARR3 ;          { deg/sec      }
                          AVELTOLpct     : longreal ) ;        { % of NOMAVEL }

               end ; { case & record }
```

```
$ page $

          LOADREC =  { record containing flt control force/torque parameters }
               record
               case integer of

                    0 : ( IDAT    : IDATREC  ) ;

                    1 : ( PVID    : NAMEPAC  ;
                          SPUFORC : VECTOR   ;                      { lb     }
                          DSPFORC : VECTOR   ;                      { lb     }
                          FORCLIM : REALARR3 ;                      { lb     }
                          TORQLIM : REALARR3 ;                      { ft*lb  }
                          TIMECON : longreal ) ;                    { sec    }

               end ; { case & record }


          ICONREC = { record containing initial conditions for the simulation }
               record
               case integer of

                    0 : ( IDAT                : IDATREC  ) ;

                    1 : ( PVID                : NAMEPAC  ;
                          PYRdeg_I_OB         : EULPYR   ;          { deg     }
                          IRATEdeg_OB_OB      : VECTOR   ;          { deg/sec }
                          RPOS_H_R            : VECTOR   ;          { ft      }
                          IVEL_PCM_OB         : VECTOR   ;          { ft/sec  }
                          PYRdeg_PD_PB        : EULPYR   ;          { deg     }
                          NOMSPINRATEdeg_P    : longreal ;          { deg/sec }
                          WOBBLE_CONEdeg_P    : longreal ;          { deg     }
                          WOBBLE_CLOKdeg_P    : longreal ) ;        { deg     }

               end ; { case & record }
```

```
$ page $

     var

          ARMS       : ARMSREC          ;
          ARMSFILE   : file of ARMSREC  ;
          ATTI       : ATTIREC          ;
          ATTIFILE   : file of ATTIREC  ;
          DRMS       : DRMSREC          ;
          DRMSFILE   : file of DRMSREC  ;
          ICON       : ICONREC          ;
          ICONFILE   : file of ICONREC  ;
          LOAD       : LOADREC          ;
          LOADFILE   : file of LOADREC  ;
          NERO       : NERTREC          ;
          NERP       : NERTREC          ;
          NERTFILE   : file of NERTREC  ;
          POSI       : POSIREC          ;
          POSIFILE   : file of POSIREC  ;
          SRMS       : SRMSREC          ;
          SRMSFILE   : file of SRMSREC  ;

     procedure IDENTIFY_SIMULATION                              ; forward ;
     procedure GET_INPUT_DATA_FILES                             ; forward ;
     procedure SET_UP_ORBITER_CONSTANTS                         ; forward ;
     procedure SET_UP_PAYLOAD_CONSTANTS                         ; forward ;
     procedure SET_UP_CREWMAN_CONSTANTS                         ; forward ;
     procedure SET_UP_FLIGHT_CONTROL                            ; forward ;
     procedure SET_UP_INITIAL_STATE                             ; forward ;

     function  SXM(         S           : longreal ;
                            M           : MAT3X3   ) : MAT3X3 ; forward ;

     function  LOCAL_RNERT( RNERT_CM_A  : MAT3X3   ;
                            MASS        : longreal ;
                            LOC_P_A     : VECTOR   ) : MAT3X3 ; forward ;


procedure INITIALIZE_SYSTEM_STATE ;

     begin

     IDENTIFY_SIMULATION                                                  ;
     GET_INPUT_DATA_FILES                                                 ;
     SET_UP_ORBITER_CONSTANTS                                             ;
     SET_UP_PAYLOAD_CONSTANTS                                             ;
     SET_UP_CREWMAN_CONSTANTS                                             ;
     SET_UP_FLIGHT_CONTROL                                                ;
     SET_UP_INITIAL_STATE                                                 ;
     SET_UP_KAMERA_DATA                                                   ;
     if USING_IMI then INITIALIZE_IMI                                     ;
     COMPUTE_FLIGHT_CONTROL_FORCE_AND_TORQUE                              ;
     STORE_SIMULATION_LOG_RECORD                                          ;
     UPDATE_IMI_DATA                                                      ;
     DOCUMENT_THE_STATE_OF_THE_SYSTEM                                     ;
     end;
```

```
$ page $

procedure IDENTIFY_SIMULATION ;

     begin
     with SLOG^[0] do
          begin
          PROGSESSID := PROGID + DATESTRING                                    ;
          SHOWLN ( 'Enter one-line description of simulation' )              ;
          SOUND_ALERT                                                          ;
          FETCHLN ( SIMDESCRIP )                                               ;
          writeln ( LP )                                                       ;
          writeln ( LP, PROGSESSID )                                           ;
          writeln ( LP, SIMDESCRIP )                                           ;
          writeln ( LP )                                                       ;
          end ;
     end ;


procedure GET_INPUT_DATA_FILES ;

     begin

     reset ( ARMSFILE, '_arms' )                                               ;
     read  ( ARMSFILE,  ARMS   )                                               ;
     close ( ARMSFILE,  'SAVE' )                                               ;

     reset ( DRMSFILE, '_drms' )                                               ;
     read  ( DRMSFILE,  DRMS   )                                               ;
     close ( DRMSFILE,  'SAVE' )                                               ;

     reset ( SRMSFILE, '_srms' )                                               ;
     read  ( SRMSFILE,  SRMS   )                                               ;
     close ( SRMSFILE,  'SAVE' )                                               ;

     reset ( NERTFILE, '_nero' )                                               ;
     read  ( NERTFILE,  NERO   )                                               ;
     close ( NERTFILE,  'SAVE' )                                               ;

     reset ( NERTFILE, '_nerp' )                                               ;
     read  ( NERTFILE,  NERP   )                                               ;
     close ( NERTFILE,  'SAVE' )                                               ;

     reset ( POSIFILE, '_posi' )                                               ;
     read  ( POSIFILE,  POSI   )                                               ;
     close ( POSIFILE,  'SAVE' )                                               ;

     reset ( ATTIFILE, '_atti' )                                               ;
     read  ( ATTIFILE,  ATTI   )                                               ;
     close ( ATTIFILE,  'SAVE' )                                               ;

     reset ( LOADFILE, '_load' )                                               ;
     read  ( LOADFILE,  LOAD   )                                               ;
     close ( LOADFILE,  'SAVE' )                                               ;

     reset ( ICONFILE, '_icon' )                                               ;
     read  ( ICONFILE,  ICON   )                                               ;
     close ( ICONFILE,  'SAVE' )                                               ;
     end ;
```

```
$ page $

procedure SET_UP_ORBITER_CONSTANTS ;

     var

          i : integer ;

     begin
     with INFO[orbiter], SLOG^[0], NERO, POSI do
          begin
          MASS := WEIGHT / STDGRAVACC                                            ;
          for i := 1 to 3 do
               RNERT_CM_B[i,i] := MOMNERT_S[i]                                   ;
          RNERT_CM_B[1,2] :=  PRDNERT_S[1]                                       ;
          RNERT_CM_B[1,3] := -PRDNERT_S[2]                                       ;
          RNERT_CM_B[2,3] :=  PRDNERT_S[3]                                       ;
          RNERT_CM_B[3,2] :=  RNERT_CM_B[2,3]                                    ;
          RNERT_CM_B[3,1] :=  RNERT_CM_B[1,3]                                    ;
          RNERT_CM_B[2,1] :=  RNERT_CM_B[1,2]                                    ;
          RALAC_CM_B := MINV( RNERT_CM_B )                                       ;
          LOC_OCM_OS := LOC_CM_S                                                 ;
          LOC_N_OB   := VXD( VDIF( LOC_N_OS , LOC_CM_S ), STRUC2BODY )           ;
          end ;
     end ;

procedure SET_UP_PAYLOAD_CONSTANTS ;

     var

          i : integer ;

     begin
     with INFO[payload], SLOG^[0], NERP, POSI do
          begin
          MASS := WEIGHT / STDGRAVACC                                            ;
          for i := 1 to 3 do
               RNERT_CM_B[i,i] := MOMNERT_S[i]                                   ;
          RNERT_CM_B[1,2] :=  PRDNERT_S[1]                                       ;
          RNERT_CM_B[1,3] := -PRDNERT_S[2]                                       ;
          RNERT_CM_B[2,3] :=  PRDNERT_S[3]                                       ;
          RNERT_CM_B[3,2] :=  RNERT_CM_B[2,3]                                    ;
          RNERT_CM_B[3,1] :=  RNERT_CM_B[1,3]                                    ;
          RNERT_CM_B[2,1] :=  RNERT_CM_B[1,2]                                    ;
          RALAC_CM_B := MINV( RNERT_CM_B )                                       ;
          LOC_PCM_PS := LOC_CM_S                                                 ;
          LOC_H_PB   := VXD( VDIF( LOC_H_PS, LOC_CM_S ), STRUC2BODY )            ;
          end ;
     end ;


procedure SET_UP_CREWMAN_CONSTANTS ;

     begin
     INFO[crewman].TALAC_CRO_R := ARMS.TALAC_CRO_R                               ;
     INFO[crewman].TDAMP_CRO_R := DRMS.TDAMP_CRO_R                               ;
     INFO[crewman].TSPRG_CRO_R := SRMS.TSPRG_CRO_R                               ;
     end ;
```

```
$ page $

procedure SET_UP_FLIGHT_CONTROL ;

     var

          C          : char      ;
          i          : integer   ;
          MODEWORD : WORDSTR   ;
          PMASS      : longreal ;

     begin
     with CNTRLCON do
          begin
          NOMLVEL := POSI.NOMLVELin  * FTPERIN                                  ;
          LVELTOL := POSI.LVELTOLpct * PERCENT * NOMLVEL                        ;
          DSPFORC := LOAD.DSPFORC                                               ;
          SPUFORC := LOAD.SPUFORC                                               ;
          TIMECON := LOAD.TIMECON                                               ;
          for i := 1 to 3 do
               begin
               AERRMAX[i] := ATTI.AERRMAXdeg[i] * RADPERDEG                     ;
               AERRMIN[i] := ATTI.AERRMINdeg[i] * RADPERDEG                     ;
               AERRTOL[i] := ATTI.AERRTOLdeg[i] * RADPERDEG                     ;
               NOMAVEL[i] := ATTI.NOMAVELdeg[i] * RADPERDEG                     ;
               AVELTOL[i] := ATTI.AVELTOLpct    * PERCENT   * NOMAVEL[i]        ;
               LERRLIM[i] := POSI.LERRLIMin[i]  * FTPERIN                       ;
               LERRTOL[i] := POSI.LERRTOLin[i]  * FTPERIN                       ;
               FORCLIM[i] := LOAD.FORCLIM[i] ;
               TORQLIM[i] := LOAD.TORQLIM[i] ;
               end ;
          end ;
     PMASS        := INFO[payload].MASS                                         ;
     RNERT_PCM_PB := INFO[payload].RNERT_CM_B                                   ;
     RNERT_PH_PB  := LOCAL_RNERT( RNERT_PCM_PB, PMASS, LOC_H_PB )               ;
     TNERT_PH_PB  := SXM( PMASS, MXM( RNERT_PCM_PB, MINV( RNERT_PH_PB ) ) )     ;
     MODEWORD := WORD_INPUT(
               'Cntrl mode {Free,Align,Dspin,Capture,Hold,Rol,Yaw,Pch,Spnup}',
               'Free' )                                                         ;
     C:= MODEWORD[1]                                                            ;
     case C of
          'A':  desired_mode := ALIGN                                           ;
          'C':  desired_mode := CAPTURE                                         ;
          'D':  desired_mode := DESPIN                                          ;
          'F':  desired_mode := FREE                                            ;
          'H':  desired_mode := HOLD                                            ;
          'P':  desired_mode := PITCH                                           ;
          'R':  desired_mode := ROLL                                            ;
          'S':  desired_mode := SPINUP                                          ;
          'Y':  desired_mode := YAW                                             ;
          end ; { case C }
     current_mode := FREE                                                       ;
     end ;
```

```
$ page $

procedure SET_UP_INITIAL_STATE ;

     var

          ANGMO_P_PB : VECTOR     ;
          AXIANGMO   : longreal   ;
          CLOK       : longreal   ;
          CONE       : longreal   ;
          NORMANGMO  : longreal   ;
          QUAT_R_PD  : QUATERNION ;
          QUAT_PD_PB : QUATERNION ;

     begin
     QUAT_OB_R    := PYRQ( EULRAD( ATTI.PYRdeg_OB_R   ) )                          ;
     QUAT_R_PD    := PYRQ( EULRAD( ATTI.PYRdeg_R_PD   ) )                          ;
     QUAT_PD_PB   := PYRQ( EULRAD( ICON.PYRdeg_PD_PB ) )                           ;
     VBASE_OB_R   := QMAT( QUAT_OB_R )                                             ;
     OBPOS_N_R    := VXM(  SLOG^[0].LOC_N_OB, VBASE_OB_R )                         ;
     QUAT_OB_PD   := QXQ(  QUAT_OB_R         , QUAT_R_PD  )                        ;
     QUAT_OB_PB   := QXQ(  QUAT_OB_PD        , QUAT_PD_PB )                        ;
     VBASE_OB_PB := QMAT( QUAT_OB_PB )                                             ;
     with STATE[orbiter], SLOG^[0] do
          begin
          IPOS_CM_I := VDIF( ZERVEC, LOC_N_OB )                                    ;
          IVEL_CM_I := ZERVEC                                                      ;
          QUAT_I_B  := PYRQ( EULRAD( ICON.PYRdeg_I_OB ) )                          ;
          IRATE_B_B := SXV( RADPERDEG, ICON.IRATEdeg_OB_OB )                       ;
          end ;
     with STATE[payload], INFO[payload] do
          begin
          IPOS_CM_I := VDIF( VXMT( ICON.RPOS_H_R, VBASE_OB_R  ) ,
                             VXMT( LOC_H_PB      , VBASE_OB_PB ) )                 ;
          IVEL_CM_I := ICON.IVEL_PCM_OB                                            ;
          QUAT_I_B  := QXQ( STATE[orbiter].QUAT_I_B, QUAT_OB_PB )                  ;
          CONE      := ICON.WOBBLE_CONEdeg_P * RADPERDEG                           ;
          CLOK      := ICON.WOBBLE_CLOKdeg_P * RADPERDEG                           ;
          AXIANGMO  := ICON.NOMSPINRATEdeg_P * RADPERDEG * RNERT_CM_B[1,1]         ;
          NORMANGMO := AXIANGMO * sin( CONE ) / cos( CONE )                        ;
          ANGMO_P_PB[1] :=  AXIANGMO                                               ;
          ANGMO_P_PB[2] :=  NORMANGMO * sin( CLOK )                                ;
          ANGMO_P_PB[3] := -NORMANGMO * cos( CLOK )                                ;
          IRATE_B_B := VXM( ANGMO_P_PB, RALAC_CM_B )                               ;
          end ;
     with STATE[crewman], SLOG^[0] do
          begin
          OBPOS_RO_R := VXM( LOC_N_OB, VBASE_OB_R )                                ;
          OBVEL_RO_R := ZERVEC                                                     ;
          end ;
     SIMTIME := ZERO                                                               ;
     curslogrec := 0                                                               ;
     PRTSLOGREC := 1                                                               ;
     end ;
```

```
$ page $

function SXM( S: longreal ; M : MAT3X3 ) : MAT3X3 ;

        { The value of this function is the product of the scalar S with     }
        { the matrix M                                                        }

    var

        i : integer ;
        j : integer ;

    begin
    for i := 1 to 3 do
        for j := 1 to 3 do
            M[i,j] := S * M[i,j]                                               ;
    SXM := M                                                                   ;
    end ;

function LOCAL_RNERT( RNERT_CM_A : MAT3X3   ;
                      MASS       : longreal ;
                      LOC_P_A    : VECTOR    ) : MAT3X3 ;

        { The value of this function is the rotational inertia tensor about  }
        { the point whose location is defined by the vector LOC_P_A.          }

    var

        i          : integer  ;
        j          : integer  ;
        RNERT_P_A  : MAT3X3   ;
        RSQ        : longreal ;

    begin
    RSQ := DOTP( LOC_P_A, LOC_P_A )                                            ;
    for i := 1 to 3 do
        for j := 1 to 3 do
            RNERT_P_A[i,j] := -LOC_P_A[i] * LOC_P_A[j]                         ;
    for i := 1 to 3 do
        RNERT_P_A[i,i] := RNERT_P_A[i,i] + RSQ                                 ;
    for i := 1 to 3 do
        for j := 1 to 3 do
            RNERT_P_A[i,j] := RNERT_CM_A[i,j] + MASS * RNERT_P_A[i,j]          ;
    LOCAL_RNERT := RNERT_P_A                                                   ;
    end ;


end ;   { module MANHINIT & File 'Manhinit.I' }
```

1.15. Input Data Editing Routines

```
$ page $  { begin File 'Manhedit.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHEDIT ;  { Subject :  Input Data Editing Routines }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILSPIF ,
     MANHMISC ,
     MANHINIT ;

export                                  { begin externally visible declarations }

     var

          OLDNAME : string [5] ;        { "first name" of data source file }
          NEWNAME : string [5] ;        { "first name" of data save   file }

     procedure EDIT_INPUT_DATA_FILES ;

implement                          { begin externally invisible part of module }
```

```
$ page $

     type

          FILETYPE = (
               arms ,    { alacrity          matrix; RMS translational flexure }
               drms ,    { damping constant matrix; RMS translational flexure }
               srms ,    { spring  constant matrix; RMS translational flexure }
               nero ,    { inertia data for Orbiter                            }
               nerp ,    { inertia data for Payload                            }
               posi ,    { position parameters for Payload flight control      }
               atti ,    { attitude parameters for Payload flight control      }
               load ,    { load     parameters for Payload flight control      }
               icon ) ;  { initial conditions for simulation                   }

          NUMIVARR = array [ FILETYPE ] of integer ;

     const

          NUMIVAR = NUMIVARR [     { number of scalar input variables per file }
                9 ,    { arms }
                9 ,    { drms }
                9 ,    { srms }
               10 ,    { nero }
               10 ,    { nerp }
               14 ,    { posi }
               19 ,    { atti }
               13 ,    { load }
               18 ] ;  { icon }

     type

          LASTNAMARR = array [ FILETYPE ] of NAMESTR ;

     const

          LASTNAME = LASTNAMARR [   { file name suffix to identify file type }
               NAMESTR [ '_arms' ] ,
               NAMESTR [ '_drms' ] ,
               NAMESTR [ '_srms' ] ,
               NAMESTR [ '_nero' ] ,
               NAMESTR [ '_nerp' ] ,
               NAMESTR [ '_posi' ] ,
               NAMESTR [ '_atti' ] ,
               NAMESTR [ '_load' ] ,
               NAMESTR [ '_icon' ] ] ;
```

```
$ page $

     type

          HEADERARR = array [ FILETYPE ] of PROMPTSTR ;

     const

          HEADER =

     HEADERARR [
                         {          1         2         3         4         5   }
                         {12345678901234567890123456789012345678901234567890123}
          PROMPTSTR['Translational Alacrity Matrix for RMS Flexure        '],
          PROMPTSTR['Translational Damping Constant Matrix for RMS Flexure'],
          PROMPTSTR['Translational Spring Constant Matrix for RMS Flexure '],
          PROMPTSTR['Orbiter Inertia Data                                 '],
          PROMPTSTR['Payload Inertia Data                                 '],
          PROMPTSTR['Position Parameters for Payload Flight Control       '],
          PROMPTSTR['Attitude Parameters for Payload Flight Control       '],
          PROMPTSTR['Force & Torque Parameters for Payload Flight Control '],
          PROMPTSTR['Initial Conditions for the Simulation                ']] ;

     type

          PROMPTARR = array [ 1..FILESIZE ] of PROMPTSTR ;
          VIDENTARR = array [ arms..icon  ] of PROMPTARR ;

     const

          VIDENT = VIDENTARR [

PROMPTARR [ { arms }
                    {          1         2         3         4         5     }
                    {1234567890123456789012345678901234567890123456789012345}
     PROMPTSTR['Alacrity matrix, element [1,1]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [1,2]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [1,3]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [2,1]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [2,2]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [2,3]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [3,1]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [3,2]..........(ft/sec/sec/lb)'],
     PROMPTSTR['Alacrity matrix, element [3,3]..........(ft/sec/sec/lb)'],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       ']],
```

```
$ page $

PROMPTARR [ { drms }

     PROMPTSTR['Damping constant matrix, element [1,1]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [1,2]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [1,3]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [2,1]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [2,2]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [2,3]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [3,1]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [3,2]......(lb/ft/sec)'],
     PROMPTSTR['Damping constant matrix, element [3,3]......(lb/ft/sec)'],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        ']],

PROMPTARR [ { srms }

     PROMPTSTR['Spring constant matrix, element [1,1]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [1,2]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [1,3]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [2,1]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [2,2]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [2,3]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [3,1]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [3,2]...........(lb/ft)'],
     PROMPTSTR['Spring constant matrix, element [3,3]...........(lb/ft)'],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        '],
     PROMPTSTR['                                                        ']],
```

```
$ page $

PROMPTARR [ { nero }

     PROMPTSTR['Orbiter weight....................................(lb)'],
     PROMPTSTR['CM STA (structural x coordinate of mass center)....(in)'],
     PROMPTSTR['CM  BL (structural y coordinate of mass center)....(in)'],
     PROMPTSTR['CM  WL (structural z coordinate of mass center)....(in)'],
     PROMPTSTR['Ixx about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Iyy about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Izz about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Pxy about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Pxz about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Pyz about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      ']],

PROMPTARR [ { nerp }

     PROMPTSTR['Payload weight....................................(lb)'],
     PROMPTSTR['CM STA (structural x coordinate of mass center)....(in)'],
     PROMPTSTR['CM  BL (structural y coordinate of mass center)....(in)'],
     PROMPTSTR['CM  WL (structural z coordinate of mass center)....(in)'],
     PROMPTSTR['Ixx about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Iyy about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Izz about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Pxy about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Pxz about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['Pyz about CM, structural coordinates.......(slug*ft*ft)'],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      '],
     PROMPTSTR['                                                      ']],
```

```
$ page $

PROMPTARR [ { posi }

     PROMPTSTR['Nominal Orbiter STA of PL handling point...........(in)'],
     PROMPTSTR['Nominal Orbiter  BL of PL handling point...........(in)'],
     PROMPTSTR['Nominal Orbiter  WL of PL handling point...........(in)'],
     PROMPTSTR['Payload STA of PL handle...........................(in)'],
     PROMPTSTR['Payload  BL of PL handle...........................(in)'],
     PROMPTSTR['Payload  WL of PL handle...........................(in)'],
     PROMPTSTR['Crewman reach limit from RO in +/- Rx direction....(in)'],
     PROMPTSTR['Crewman reach limit from RO in +/- Ry direction....(in)'],
     PROMPTSTR['Crewman reach limit from RO in +/- Rz direction....(in)'],
     PROMPTSTR['Rx position tolerance for PL handle................(in)'],
     PROMPTSTR['Ry position tolerance for PL handle................(in)'],
     PROMPTSTR['Rz position tolerance for PL handle................(in)'],
     PROMPTSTR['Nominal vel for handle position correction.....(in/sec)'],
     PROMPTSTR['Tolerance for corrective velocity...................(%)'],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       ']],

PROMPTARR [ { atti }

     PROMPTSTR['Crewman pitch wrt Orbiter body axes...............(deg)'],
     PROMPTSTR['Crewman yaw   wrt Orbiter body axes...............(deg)'],
     PROMPTSTR['Crewman roll  wrt Orbiter body axes...............(deg)'],
     PROMPTSTR['Desired Payload pitch wrt Crewman body axes.......(deg)'],
     PROMPTSTR['Desired Payload yaw   wrt Crewman body axes.......(deg)'],
     PROMPTSTR['Desired Payload roll  wrt Crewman body axes.......(deg)'],
     PROMPTSTR['Positive PL pitch limit wrt desired attitude......(deg)'],
     PROMPTSTR['Positive PL yaw   limit wrt desired attitude......(deg)'],
     PROMPTSTR['Positive PL roll  limit wrt desired attitude......(deg)'],
     PROMPTSTR['Negative PL pitch limit wrt desired attitude......(deg)'],
     PROMPTSTR['Negative PL yaw   limit wrt desired attitude......(deg)'],
     PROMPTSTR['Negative PL roll  limit wrt desired attitude......(deg)'],
     PROMPTSTR['PL pitch tolerance wrt desired attitude...........(deg)'],
     PROMPTSTR['PL yaw   tolerance wrt desired attitude...........(deg)'],
     PROMPTSTR['PL roll  tolerance wrt desired attitude...........(deg)'],
     PROMPTSTR['Nominal maneuver rate about PL Bx axis........(deg/sec)'],
     PROMPTSTR['Nominal maneuver rate about PL By axis........(deg/sec)'],
     PROMPTSTR['Nominal maneuver rate about PL Bz axis........(deg/sec)'],
     PROMPTSTR['Maneuver rate tolerance.............................(%)'],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       ']],
```

```
$ page $

PROMPTARR [ { load }

     PROMPTSTR['Spinup axial      (PL Bx) force...................(lb)'],
     PROMPTSTR['Spinup normal     (PL By) force...................(lb)'],
     PROMPTSTR['Spinup tangential (PL Bz) force...................(lb)'],
     PROMPTSTR['Despin axial      (PL Bx) force...................(lb)'],
     PROMPTSTR['Despin normal     (PL By) force...................(lb)'],
     PROMPTSTR['Despin tangential (PL Bz) force...................(lb)'],
     PROMPTSTR['Axial      (PL Bx) force limit for capture.........(lb)'],
     PROMPTSTR['Normal     (PL By) force limit for capture.........(lb)'],
     PROMPTSTR['Tangential (PL Bz) force limit for capture.........(lb)'],
     PROMPTSTR['Roll  (PL Bx) torque limit for capture..........(ft*lb)'],
     PROMPTSTR['Pitch (PL By) torque limit for capture..........(ft*lb)'],
     PROMPTSTR['Yaw   (PL Bz) torque limit for capture..........(ft*lb)'],
     PROMPTSTR['Time constant for computing desired accelerations (sec)'],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       ']],

PROMPTARR [ { icon }

     PROMPTSTR['Orbiter pitch wrt I axes.........................(deg)'],
     PROMPTSTR['Orbiter yaw   wrt I axes.........................(deg)'],
     PROMPTSTR['Orbiter roll  wrt I axes.........................(deg)'],
     PROMPTSTR['Orbiter Bx component of ang vel wrt I axes....(deg/sec)'],
     PROMPTSTR['Orbiter By component of ang vel wrt I axes....(deg/sec)'],
     PROMPTSTR['Orbiter Bz component of ang vel wrt I axes....(deg/sec)'],
     PROMPTSTR['Rx component of PL handle position.................(ft)'],
     PROMPTSTR['Ry component of PL handle position.................(ft)'],
     PROMPTSTR['Rz component of PL handle position.................(ft)'],
     PROMPTSTR['Payload CM Xdot wrt Orbiter body axes..........(ft/sec)'],
     PROMPTSTR['Payload CM Ydot wrt Orbiter body axes..........(ft/sec)'],
     PROMPTSTR['Payload CM Zdot wrt Orbiter body axes..........(ft/sec)'],
     PROMPTSTR['Payload pitch wrt desired attitude................(deg)'],
     PROMPTSTR['Payload yaw   wrt desired attitude................(deg)'],
     PROMPTSTR['Payload roll  wrt desired attitude................(deg)'],
     PROMPTSTR['Payload nominal spin rate.....................(deg/sec)'],
     PROMPTSTR['Payload wobble cone angle.........................(deg)'],
     PROMPTSTR['Payload wobble clok angle.........................(deg)'],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       '],
     PROMPTSTR['                                                       ']]] ;
```

```
$ page $

     var

          CHANGED   : boolean     ;
          CURPVID   : NAMEPAC     ;            { current program version ident }
          DATA      : IDATREC     ;
          DATAFILE  : file of IDATREC ;
          FILENAME  : string [20] ;
          OKAY      : boolean     ;
          OLDVALUE  : longreal    ;
          PRINT     : boolean     ;
          PRINTNAME : string [5]  ;

     procedure GET_DATA_FILE  ( kind : FILETYPE ) ; forward ;
     procedure EDIT_DATA_FILE ( kind : FILETYPE ) ; forward ;
     procedure SAVE_DATA_FILE ( kind : FILETYPE ) ; forward ;


procedure EDIT_INPUT_DATA_FILES ;

     var

          i    : integer  ;
          kind : FILETYPE ;

     begin
     CURPVID := 'MANH    '                                                      ;
     for i := 5 to 8 do
          CURPVID[i] := PROGID[i+14]                                            ;
     repeat
          CLEAR_SCREEN                                                          ;
          for kind := arms to icon do
               if USER_DECIDES_TO( 'Edit '+HEADER[kind] ) then
                    begin
                    GET_DATA_FILE  ( kind )                                     ;
                    EDIT_DATA_FILE ( kind )                                     ;
                    SAVE_DATA_FILE ( kind )                                     ;
                    end ;
          until USER_DECIDES_NOT_TO( 'Re-run editor' )                          ;
     CLEAR_SCREEN                                                               ;
     end ;
```

```
$ page $

procedure GET_DATA_FILE  ( kind : FILETYPE ) ;

     begin
     CLEAR_SCREEN                                                              ;
     SHOWLN ( '' )                                                             ;
     PRINTNAME := OLDNAME                                                      ;
     repeat
          try
               PRINTNAME := WORD_INPUT( 'File name', PRINTNAME )               ;
               FILENAME  := PRINTNAME+LASTNAME[kind]                           ;
               reset ( DATAFILE,  FILENAME )                                   ;
               read  ( DATAFILE,  DATA     )                                   ;
               close ( DATAFILE, 'SAVE'    )                                   ;
               OKAY := true                                                    ;

               recover
               begin
               OKAY := false                                                   ;
               PRINTNAME := ''                                                 ;
               end ;

          until OKAY                                                           ;
     OLDNAME := PRINTNAME                                                      ;
     end ;


procedure EDIT_DATA_FILE ( kind : FILETYPE ) ;

     var

          i        : integer  ;
          OLDVALUE : longreal ;

     begin
     DATA.PVID := CURPVID                                                      ;
     CHANGED   := false                                                        ;
     repeat
          CLEAR_SCREEN                                                         ;
          SHOWLN ( '' )                                                        ;
          SHOWLN ( '"'+PRINTNAME+'" '+HEADER[kind] )                           ;
          SHOWLN ( '' )                                                        ;
          for i := 1 to NUMIVAR[kind] do
               begin
               OLDVALUE := DATA.ARR[i]                                         ;
               DATA.ARR[i]  := FIXED_INPUT( VIDENT[kind,i],DATA.ARR[i],17,5 )  ;
               if DATA.ARR[i] <> OLDVALUE then CHANGED := true                 ;
               end ;
          if CHANGED then PRINTNAME := ''                                      ;
          LOITER ( 1000 )                                                      ;
          SOUND_ALERT                                                          ;
          LOITER ( 500 )                                                       ;
          until USER_DECIDES_NOT_TO( 'Re_edit this file' )                     ;
     end ;
```

```
$ page $

procedure SAVE_DATA_FILE ( kind: FILETYPE ) ;

     var

          i        : integer   ;
          SAVENAME : string[5] ;

     begin
     if CHANGED or ( OLDNAME <> '' ) then
          begin
          FILENAME := LASTNAME[kind]                                         ;
          rewrite ( DATAFILE,  FILENAME )                                    ;
          write   ( DATAFILE,  DATA     )                                    ;
          close   ( DATAFILE, 'SAVE'    )                                    ;
          end ;
     SAVENAME := NEWNAME                                                     ;
     repeat
          OKAY := true                                                       ;
          try                                     { set outer trap for error }
               SAVENAME := WORD_INPUT( 'Name for this file',SAVENAME )       ;
               if SAVENAME <> '' then
                    begin
                    FILENAME := SAVENAME+LASTNAME[kind]                      ;
                    try                           { set inner trap for error }
                         reset ( DATAFILE, FILENAME ) ;
                         close ( DATAFILE, 'SAVE'   ) ;
                         OKAY := USER_DECIDES_TO( 'Write over existing file'+
                                                  ' named "'+SAVENAME+'"'   ) ;

                         recover      { come here when inner trap is sprung }
                         OKAY := true ;

                    if OKAY then
                         begin
                         rewrite ( DATAFILE,  FILENAME )                     ;
                         write   ( DATAFILE,  DATA     )                     ;
                         close   ( DATAFILE, 'SAVE'    )                     ;
                         SHOWLN  ('File saved under the name "'+SAVENAME+'"') ;
                         PRINTNAME := SAVENAME                               ;
                         end ;
                    end ;

               recover                { come here when outer trap is sprung }
               OKAY := false                                                 ;

          if not OKAY then SAVENAME := NEWNAME                               ;
          until OKAY ;
     NEWNAME := SAVENAME                                                     ;
     for i := 1 to 4 do writeln ( LP )                                       ;
     writeln ( LP, PROGID+DATESTRING )                                       ;
     for i := 1 to 4 do writeln ( LP )                                       ;
     if CHANGED or ( SAVENAME <> '' ) then write ( LP, 'New ' )              ;
     writeln ( LP, '"',PRINTNAME,'" ',HEADER[kind] )                         ;
     writeln ( LP )                                                          ;
     for i := 1 to NUMIVAR[kind] do
          writeln ( LP, VIDENT[kind,i],DATA.ARR[i]:17:5 )                    ;
     START_NEW_PAGE                                                          ;
     end ;
```

```
end ;  { module MANHEDIT & File 'Manhedit.I' }
```

1.16. Postprocessing of Simulation Log Data

```
$ page $  { begin File 'Manhpost.I' }

{ Payload Manhandling Simulator for HP-9000 Series 200/300/500 Computers }

module MANHPOST ;  { Subject :  Postprocessing of Simulation Log Data }

                   { NASA/JSC/MPAD/TRW :  Sam Wilson }

import

     UTILSPIF ,
     UTILMATH ,
     UTILVEMQ ,
     MANHMISC ,
     MANHFCON ,
     MANHSLOG ,
     MANHDISP ;

export                                  { begin externally visible declarations }

     var

          PLT            : text    ;
          PLREV          : integer ;
          STROKE         : integer ;
          USING_PLOTTER  : boolean ;

     procedure EXECUTE_POSTPROCESSOR ;

implement                               { begin externally invisible part of module }

     var

          PATTSAV       : EULPYR   ;
          ROLLSAV       : longreal ;
          STROKE_MOD_3  : integer  ;
          TIMESAV       : longreal ;

     procedure SHOW_PLOTTER_INITIALIZATION_MESSAGE                        ; forward ;
     procedure PLOT_CONTROL_FORCE_AND_TORQUE_HISTORY                      ; forward ;
     procedure DRAW_CONTROL_DATA_CURVES ( WINDOW : integer )              ; forward ;
     procedure DRAW_CONTROL_XAXIS                                         ; forward ;
     procedure DRAW_CONTROL_YAXIS                                         ; forward ;
     procedure PLOT_PAYLOAD_SPINAXIS_ATTITUDE_TRAJECTORY                  ; forward ;
     procedure GET_STROKE_AND_PLREV_NUMBER_FROM_USER                      ; forward ;
     procedure DRAW_ATTITUDE_XAXIS                                        ; forward ;
     procedure DRAW_ATTITUDE_YAXIS                                        ; forward ;
     procedure DRAW_ATTITUDE_TRAJECTORY                                   ; forward ;
     procedure PLOT_ANGULAR_MOMENTUM_DIRECTION ( ANGMO : VECTOR )         ; forward ;
     procedure PLOT_REV_IDENTIFICATION_SYMBOL ( XO, YO : integer )        ; forward ;
     procedure PLOT_RMS_FLEXURE_HISTORY                                   ; forward ;
     procedure DRAW_FLEXURE_DATA_CURVES                                   ; forward ;
     procedure DRAW_FLEXURE_XAXIS                                         ; forward ;
     procedure DRAW_FLEXURE_YAXIS                                         ; forward ;
```

```
$ page $

     procedure EXECUTE_POSTPROCESSOR ;

          begin
          while USER_DECIDES_TO( 'Play back visual data' ) do
               PLAY_BACK_VISUAL_DATA                                        ;
          if USING_PLOTTER then
               begin
               CLEAR_SCREEN                                                 ;
               if USER_DECIDES_TO('Plot RMS flexure history') then
                    begin
                    SHOW_PLOTTER_INITIALIZATION_MESSAGE                     ;
                    PLOT_RMS_FLEXURE_HISTORY                                ;
                    end ;
               CLEAR_SCREEN                                                 ;
               if USER_DECIDES_TO('Plot control force & torque history') then
                    begin
                    SHOW_PLOTTER_INITIALIZATION_MESSAGE                     ;
                    PLOT_CONTROL_FORCE_AND_TORQUE_HISTORY                   ;
                    end ;
               CLEAR_SCREEN                                                 ;
               if USER_DECIDES_TO('Plot PL spinaxis attitude trajectory') then
                    begin
                    SHOW_PLOTTER_INITIALIZATION_MESSAGE                     ;
                    PLOT_PAYLOAD_SPINAXIS_ATTITUDE_TRAJECTORY               ;
                    end ;
               end ;
          end ;
```

```
$ page $

procedure SHOW_PLOTTER_INITIALIZATION_MESSAGE ;

     type

          L      = string[55] ;
          MSGARR = array [ 1..14 ] of L ;

     const

          MSGLINE = MSGARR [
                 {         1         2         3         4         5
                 {1234567890123456789012345678901234567890123456789012345}
          { 1} L['Before proceeding, verify that an 8.5" x 11" sheet of  '],
          { 2} L['paper is present on the bed of the plotter and that the'],
          { 3} L['electrostatic "hold" is on.  The edges of the paper    '],
          { 4} L['should be against the lower and the left-hand stops on '],
          { 5} L['the edges of the plotter bed.  The long dimension of   '],
          { 6} L['the paper must be aligned with the short dimension of  '],
          { 7} L['the plotter bed.                                       '],
          { 8} L['                                                       '],
          { 9} L['Also verify that pens are installed in penholders #1,  '],
          {10} L['#2, and #3.  Recommended pen colors are as follows:    '],
          {11} L['                                                       '],
          {12} L['                Pen #1 : Black                         '],
          {13} L['                Pen #2 : Red                           '],
          {14} L['                Pen #3 : Green                         ']] ;

     var

          i : integer ;

     begin
     SHOWLN ( '' )                                                          ;
     for i := 1 to 14 do SHOWLN ( FILL12+MSGLINE[i] )                       ;
     SHOWLN ( '' )                                                          ;
     end ;
```

```
$ page $

procedure PLOT_CONTROL_FORCE_AND_TORQUE_HISTORY ;

     var

          BACKSPACES : longreal ;
          i          : integer  ;
          WINDOW     : integer  ;

     begin
     repeat
          { nothing }
          until USER_DECIDES( 'Ready to plot' )                                ;
     writeln ( PLT, 'DF'  )                        { reset plotter        } ;
     writeln ( PLT, 'IP500,1000,7800,10085' )      { set scaling points   } ;
     writeln ( PLT, 'SC-1000,15500,-3600,1000'  )  { select upper window  } ;
     writeln ( PLT, 'SP1' )                        { select pen #1 (black) } ;
     with SLOG^[0] do
          begin
          writeln ( PLT, 'PA8000,1200'           )                          ;
          BACKSPACES := -strlen( PROGSESSID ) / TWO                         ;
          writeln ( PLT, 'CP',BACKSPACES:5:1,',0' )                         ;
          writeln ( PLT, 'LB',PROGSESSID,#3       ) { identify prog/session } ;
          writeln ( PLT, 'PA8000,-3800'           )                         ;
          BACKSPACES := -strlen( SIMDESCRIP ) / TWO                         ;
          writeln ( PLT, 'CP',BACKSPACES:5:1,',0' )                         ;
          writeln ( PLT, 'LB',SIMDESCRIP,#3       ) { identify simulation   } ;
          end ;
     for i := 1 to 3 do
          begin                                   { identifying line codes }
          writeln ( PLT, 'PA3000,',-(960+120*i):6,';PU')     { position   } ;
          writeln ( PLT, 'CP1.0,-0.2' )                      { for label  } ;
          case i of                                { label line segment     }
               1: writeln ( PLT, 'LBX Component (Payload Body Axes)',#3 )   ;
               2: writeln ( PLT, 'LBY Component (Payload Body Axes)',#3 )   ;
               3: writeln ( PLT, 'LBZ Component (Payload Body Axes)',#3 )   ;
               end ; { case i }
          end ;
     for WINDOW := 1 to 2 do
          DRAW_CONTROL_DATA_CURVES ( WINDOW )                               ;
     writeln ( PLT, 'PU;SP0' )                        { put pen in holder } ;
     end ;
```

```
$ page $

procedure DRAW_CONTROL_DATA_CURVES ( WINDOW : integer ) ;

     var

          i : integer ;
          n : integer ;
          X : integer ;
          Y : integer ;

     begin
     writeln ( PLT, 'SP1' )                            { select pen #1 (black) } ;
     if WINDOW = 2 then
          writeln ( PLT, 'SC-1000,15500,-1000,3600')  { select lower window } ;
     DRAW_CONTROL_XAXIS                                                          ;
     writeln ( PLT, 'PA12000,0' )                                                ;
     writeln ( PLT, 'CP0.0,-2.2;LBTime (sec)',#3 )  { identify time axis    } ;
     DRAW_CONTROL_YAXIS                                                          ;
     writeln ( PLT, 'PA0,0' )                                                    ;
     writeln ( PLT, 'DI0,1' )                          { vertical label slant  } ;
     if WINDOW = 1
          then writeln ( PLT, 'CP-9.0,1.5;LBControl Force (lb)',#3 )
          else writeln ( PLT, 'CP-11.0,1.5;LBControl Torque (ft*lb)',#3 )       ;
     writeln ( PLT, 'DI1,0' )                          { horizntl label slant  } ;
     writeln ( PLT, 'SP2' )                            { select pen #2 (red)   } ;
     for i := 1 to 3 do
          begin
          case i of
               1: writeln ( PLT, 'LT1,0.3' )           { select dotted line    } ;
               2: writeln ( PLT, 'LT3,1.0' )           { select dashed line    } ;
               3: writeln ( PLT, 'LT'      )           { select solid  line    } ;
               end ; { case i }
          if WINDOW = 1 then
               begin
               writeln ( PLT, 'PA2000,',-(960+120*i):6,';PD')   { draw line } ;
               writeln ( PLT, 'PA3000,',-(960+120*i):6,';PU')   { segment   } ;
               end ;
          for n := 1 to SLOG^[0].NUMSLOGRECS do
               begin
               X := round( 100*SLOG^[n].TIME)                                    ;
               if X <= 15000 then
                    begin
                    if SLOG^[n].curmode = FREE
                         then Y := 0
                         else if WINDOW = 1
                              then Y := round( 100*SLOG^[n].CFORC_PH_PB[i])
                              else Y := round( 100*SLOG^[n].CTORQ_PH_PB[i])     ;
                    writeln ( PLT, 'PA',X:6,',',Y:6,';PD' )                      ;
                    end ;
               end ;
          writeln ( PLT, 'PU' )                                   { raise pen } ;
          end ;
     end ;
```

```
$ page $

procedure DRAW_CONTROL_XAXIS ;

     var

          X : integer ;

     begin
     for X := 0 to 150 do
          begin               { drawing 1-second & 5-second time tick marks }
          if ( X mod 5 ) = 0
               then writeln ( PLT, 'TL1.0,1.0' )     { draw long  tick mark  }
               else writeln ( PLT, 'TL0.5,0.5' )     { draw short tick mark  } ;
          writeln ( PLT, 'PA',(100*X):6,',0' )      { go to the spot        } ;
          writeln ( PLT, 'PD;XT' )                  { lower pen & draw tick } ;
          end ;
     writeln ( PLT, 'PU' )                          { raise pen             } ;
     for X := 1 to 15 do
          begin        { putting numeric label on every 10-second tick mark }
          writeln ( PLT, 'PA',(1000*X):6,',0' )     { go to tick mark       } ;
          if X < 10
               then writeln ( PLT, 'CP-1.0,-1.2' ) { back up 1.0 spaces    }
               else writeln ( PLT, 'CP-1.5,-1.2' ) { back up 1.5 spaces    } ;
          writeln ( PLT, 'LB',(10*X):1,#3 )         { print the label       } ;
          end ;
     end ;


procedure DRAW_CONTROL_YAXIS ;

     var

          Y : integer ;

     begin
     for Y := -10 to 10 do
          begin      { drawing 1-lb (or ft*lb) & 5-lb (or ft*lb) tick marks }
          if ( Y mod 5 ) = 0
               then writeln ( PLT, 'TL1.0,1.0' )     { draw long  tick mark  }
               else writeln ( PLT, 'TL0.5,0.5' )     { draw short tick mark  } ;
          writeln ( PLT, 'PA0,',(100*Y):6 )         { go to the spot        } ;
          writeln ( PLT, 'PD;YT' )                  { lower pen & draw tick } ;
          end ;
     writeln ( PLT,'PU' )                           { raise pen             } ;
     for Y := -1 to 1 do
          begin     { putting numeric labels on 0- & 10-lb (or ft*lb) ticks }
          writeln ( PLT, 'PA0,',(1000*Y):6 )            { go to tick mark   } ;
          if Y < 0
               then writeln ( PLT, 'CP-4.0,-0.2' )      { "-10"; back up 4  }
               else if Y < 1
                    then writeln ( PLT, 'CP-2.0,-0.2' ) {   "0"; back up 2  }
                    else writeln ( PLT, 'CP-3.0,-0.2' ) {  "10"; back up 3  } ;
          writeln ( PLT, 'LB',(10*Y):1,#3 )             { print the label   } ;
          end ;
     end ;
```

```
$ page $

procedure PLOT_PAYLOAD_SPINAXIS_ATTITUDE_TRAJECTORY ;

     var

          BACKSPACES : longreal ;
          i          : integer  ;

     begin
     repeat
          { nothing }
          until USER_DECIDES( 'Ready to plot' )                                   ;
     GET_STROKE_AND_PLREV_NUMBER_FROM_USER                                        ;
     writeln ( LP )                                                               ;
     writeln ( LP,' Plot   Start     Time     Payload Attitude wrt I Axes' )      ;
     writeln ( LP,'Symbol  Rev #     (sec)      Pitch      Yaw       Roll   ' )   ;
     writeln ( LP )                                                               ;
     writeln ( PLT, 'DF' )                           { reset plotter         }    ;
     writeln ( PLT, 'IP500,1000,7800,10085' )        { set scaling points    }    ;
     writeln ( PLT, 'SC-11000,11000,-13750,13750' ) { define window         }    ;
     writeln ( PLT, 'SP1' )                          { select pen #1 (black) }    ;
     with SLOG^[0] do
          begin
          writeln ( PLT, 'PA0,13500'               )                              ;
          BACKSPACES := -strlen( PROGSESSID ) / TWO                               ;
          writeln ( PLT, 'CP',BACKSPACES:5:1,',0' )                               ;
          writeln ( PLT, 'LB',PROGSESSID,#3        ) { identify prog/session }    ;
          writeln ( PLT, 'PA0,-14000'              )                              ;
          BACKSPACES := -strlen( SIMDESCRIP ) / TWO                               ;
          writeln ( PLT, 'CP',BACKSPACES:5:1,',0' )                               ;
          writeln ( PLT, 'LB',SIMDESCRIP,#3        ) { identify simulation   }    ;
          end ;
     DRAW_ATTITUDE_XAXIS                                                          ;
     writeln ( PLT, 'PA-9600,0' )                                                 ;
     writeln ( PLT, 'CP0.0,-2.1' )                                                ;
     writeln ( PLT, 'LBPL Yaw (deg, wrt I axes)',#3 )                             ;
     DRAW_ATTITUDE_YAXIS                                                          ;
     writeln ( PLT, 'DI0,1' )                         { vertical label slant }    ;
     writeln ( PLT, 'PA0,7500' )                                                  ;
     writeln ( PLT, 'CP-6.0,1.6' )                                                ;
     writeln ( PLT, 'LBPL Pitch (deg, wrt I axes)',#3 )                           ;
     writeln ( PLT, 'DI1,0' )                         { horizntl label slant }    ;
     DRAW_ATTITUDE_TRAJECTORY                                                     ;
     writeln ( PLT, 'PU;SP0' )                           { put pen in holder }    ;
     START_NEW_PAGE                                                               ;
     end ;
```

```
$ page $

procedure GET_STROKE_AND_PLREV_NUMBER_FROM_USER ;

     var

          VALUE        : integer   ;

     begin
     repeat
          VALUE := STROKE                                                          ;
          VALUE := INTEGER_INPUT( 'Spinup/despin stroke number',VALUE,2 )          ;
          if VALUE < 0 then
               begin
               SOUND_ALARM                                                         ;
               MOVE_UP                                                             ;
               end ;
          until VALUE >= 0 ;
     STROKE := VALUE                                                               ;
     repeat
          VALUE := PLREV                                                           ;
          VALUE := INTEGER_INPUT( 'Initial payload rev number ',VALUE,2  )         ;
          if VALUE < 0 then
               begin
               SOUND_ALARM                                                         ;
               MOVE_UP                                                             ;
               end ;
          until VALUE >= 0 ;
     PLREV := VALUE                                                                ;
     end ;
```

```
$ page $

procedure DRAW_ATTITUDE_XAXIS ;

     var

          D : longreal ;
          X : integer  ;

     begin
     for X := -20 to 20 do
          begin                    { drawing 1-deg & 5-deg yaw tick marks }
          if ( X mod 5 ) = 0
               then writeln ( PLT, 'TL1.0,1.0' )    { draw long tick mark    }
               else writeln ( PLT, 'TL0.5,0.5' )    { draw short tick mark   } ;
          writeln ( PLT, 'PA',(-500*X):6,',0' )     { go to the spot         } ;
          writeln ( PLT, 'PD;XT' )                  { lower pen & draw tick  } ;
          end ;
     writeln ( PLT, 'PU' )                          { raise pen              } ;
     for X := -4 to 4 do
          if X <> 0 then           { put numeric label on long tick mark    }
               begin
               writeln ( PLT, 'PA',(-2500*X):6,',0' )  { go to the tick     } ;
               if abs( X ) < 2
                    then D := -0.4
                    else D := -0.9                                          ;
               if X < 0 then D := D - 1                                     ;
               writeln ( PLT, 'CP',D:4:1,',-1.15' )    { go to 1st char pos } ;
               writeln ( PLT, 'LB',(5*X):1,#3 )        { print the label    } ;
               end ;
     end ;
```

```
$ page $

procedure DRAW_ATTITUDE_YAXIS ;

     var

          D : longreal ;
          Y : integer  ;

     begin
     for Y := -25 to 25 do
          begin                    { drawing 1-deg & 5-deg pitch tick marks }
          if ( Y mod 5 ) = 0
               then write ( PLT, 'TL1.0,1.0;' )     { draw long  tick mark  }
               else write ( PLT, 'TL0.5,0.5;' )     { draw short tick mark  } ;
          writeln ( PLT, 'PA0,',(500*Y):6 )         { go to the spot        } ;
          writeln ( PLT, 'PD;YT' )                  { lower pen & draw tick } ;
          end ;
     writeln ( PLT,'PU' )                           { raise pen             } ;
     for Y := -5 to 5 do
          if Y <> 0 then               { put numeric label on long tick mark }
               begin
               writeln ( PLT, 'PA0,',(2500*Y):6 )   { go to the tick        } ;
               if abs( Y ) < 2
                    then D := -2.2
                    else D := -3.2                                            ;
               if Y < 0 then D := D - 1                                       ;
               writeln ( PLT, 'CP',D:4:1,',-0.2' )  { go to 1st char pos    } ;
               writeln ( PLT, 'LB',(5*Y):1,#3 )     { print the label       } ;
               end ;
     end ;
```

```
$ page $

procedure DRAW_ATTITUDE_TRAJECTORY ;

     var

          n               : integer   ;
          OLDMODE         : CNTRLMODE ;
          ROLL_DECREASING : boolean   ;
          ROLLNOW         : longreal  ;
          X               : integer   ;
          Y               : integer   ;

     begin
     ROLL_DECREASING := false                                               ;
     ROLLSAV := ZERO                                                        ;
     OLDMODE := FREE                                                        ;
     PLOT_ANGULAR_MOMENTUM_DIRECTION ( SLOG^[1].ANGMO_P_I )                 ;
     for n := 1 to SLOG^[0].NUMSLOGRECS do
          with SLOG^[n] do
               begin
               if curmode = FREE

                    then
                    begin
                    if OLDMODE <> FREE then
                         begin
                         STROKE := STROKE + 1                               ;
                         PLOT_ANGULAR_MOMENTUM_DIRECTION ( ANGMO_P_I )      ;
                         end ;
                    ROLLNOW := abs( ANGDEG( PYR_I_PB[3] ) )                 ;
                    if ROLLNOW < ROLLSAV

                         then
                         ROLL_DECREASING := true

                         else
                         begin
                         if ROLL_DECREASING then
                              PLOT_REV_IDENTIFICATION_SYMBOL ( X, Y )       ;
                         ROLL_DECREASING := false                           ;
                         end ;

                    ROLLSAV := ROLLNOW                                      ;
                    TIMESAV := TIME                                         ;
                    PATTSAV := EULDEG( PYR_I_PB )                           ;
                    X := -round( 500 * PATTSAV[2] )                         ;
                    Y :=  round( 500 * PATTSAV[1] )                         ;
                    writeln ( PLT, 'PA',X:6,',',Y:6,';PD' )                 ;
                    end

                    else
                    if OLDMODE = FREE then
                         writeln ( PLT, 'PU' )                              ;

               OLDMODE := curmode                                           ;
               end ;
     end ;
```

```
$ page $

procedure PLOT_ANGULAR_MOMENTUM_DIRECTION ( ANGMO : VECTOR ) ;

     var

          ANGMOZX : longreal ;
          PCH     : longreal ;
          X       : integer  ;
          Y       : integer  ;
          YAW     : longreal ;

     begin
     STROKE_MOD_3 := STROKE mod 3                                                      ;
     case STROKE_MOD_3 of
          0: writeln (PLT,'SP3;LT')            { select green solid  line } ;
          1: writeln (PLT,'SP2;LT1,0.3')       { select red   dotted line } ;
          2: writeln (PLT,'SP1;LT3,1.0')       { select black dashed line } ;
          end ; { case STROKE_MOD_3 }
     ANGMOZX :=  sqrt( sqr( ANGMO[3] ) + sqr( ANGMO[1] ) )                             ;
     YAW :=  DEGPERRAD * ATAN2(  ANGMO[2], ANGMOZX  )                                  ;
     PCH :=  DEGPERRAD * ATAN2( -ANGMO[3], ANGMO[1] )                                  ;
     X   := -round( 500 * YAW )                                                        ;
     Y   :=  round( 500 * PCH )                                                        ;
     writeln ( PLT, 'PU;PA',X:6,',',Y:6          )       { go to the spot      } ;
     writeln ( PLT, 'PR-300,0;PD;PR600,0;PU'     )       { draw horizntl line } ;
     writeln ( PLT, 'PR-300,-300;PD;PR0,600;PU' )        { draw vertical line } ;
     end ;
```

```
$ page $

procedure PLOT_REV_IDENTIFICATION_SYMBOL ( XO, YO : integer ) ;

     const

          R = 550 ;  { radius of shell enclosing symbol }

     var

          C    : char     ;
          Ci   : longreal ;
          CINC : longreal ;
          CL   : longreal ;
          i    : integer  ;
          N    : integer  ;
          Si   : longreal ;
          SINC : longreal ;
          X    : integer  ;
          Y    : integer  ;

     begin
     PLREV := PLREV + 1                               { bump the rev counter    } ;
     if PLREV < 10
          then C := chr( 48+PLREV )                   { symbols = '1'..'9'      }
          else C := chr( 55+PLREV )                   { symbols = 'A'..'Z'      } ;
     writeln ( PLT, 'LT'    )                         { select solid line       } ;
     writeln ( PLT, 'SM',C,';PR0,0;SM' )              { plot the symbol         } ;
     writeln ( PLT, 'PU;PA',XO:6,',',(YO+R):6 )     { move to first corner of } ;
     writeln ( PLT, 'PD'                      )     { shell & lower pen       } ;
     Ci   := ONE                                    { initialize cosine       } ;
     Si   := ZERO                                   { initialize sine         } ;
     N    := STROKE + 3                             { number of shell corners } ;
     CINC := cos( TWOPI / N )                       { cos of angle increment  } ;
     SINC := sin( TWOPI / N )                       { sin of angle increment  } ;
     for i := 1 to N do
          begin
          CL := Ci                                  { save last cosine        } ;
          Ci := CL * CINC - Si * SINC               { update cosine           } ;
          Si := Si * CINC + CL * SINC               { update sine             } ;
          X  := round( XO + R * Si )                { compute coordinates of  } ;
          Y  := round( YO + R * Ci )                { next corner of shell    } ;
          writeln ( PLT, 'PA',X:6,',',Y:6 )         { draw line to next corner } ;
          end ;
     writeln ( PLT, 'PU;PA',XO:6,',',YO:6 )           { restore pen position    } ;
     writeln ( PLT, 'PD'                  )           { lower pen               } ;
     case STROKE_MOD_3 of                             { restore line style      }
          0: writeln ( PLT, 'LT' )                                     { solid  } ;
          1: writeln ( PLT, 'LT1,0.3' )                                { dotted } ;
          2: writeln ( PLT, 'LT3,1.0' )                                { dashed } ;
          end ; { case STROKE_MOD_3 }
     writeln ( LP, '  "',C:1,'"',PLREV:7 ,            { send detailed info to   }
                    TIMESAV:12:3,'  '    ,            { standard print file     }
                    PATTSAV[1]:9:2       ,
                    PATTSAV[2]:9:2       ,
                    PATTSAV[3]:9:2,' (deg)' )                                    ;
     end ;
```

```
$ page $

procedure PLOT_RMS_FLEXURE_HISTORY ;

     var

          BACKSPACES : longreal ;
          i          : integer  ;

     begin
     repeat
          { nothing }
          until USER_DECIDES( 'Ready to plot' )                                    ;
     writeln ( PLT, 'DF'  )                            { reset plotter          } ;
     writeln ( PLT, 'IP500,1000,7800,10085'  )         { set scaling points     } ;
     writeln ( PLT, 'SC-200,4800,-1200,1440' )         { define window          } ;
     writeln ( PLT, 'SP1' )                            { select pen #1 (black)  } ;
     with SLOG^[0] do
          begin
          writeln ( PLT, 'PA2250,1560'             )                              ;
          BACKSPACES := -strlen( PROGSESSID ) / TWO                               ;
          writeln ( PLT, 'CP',BACKSPACES:5:1,',0' )                               ;
          writeln ( PLT, 'LB',PROGSESSID,#3        ) { identify prog/session } ;
          writeln ( PLT, 'PA2250,-1240'            )                              ;
          BACKSPACES := -strlen( SIMDESCRIP ) / TWO                               ;
          writeln ( PLT, 'CP',BACKSPACES:5:1,',0' )                               ;
          writeln ( PLT, 'LB',SIMDESCRIP,#3        ) { identify simulation    } ;
          end ;
     for i := 1 to 3 do
          begin                                       { identifying line codes }
          writeln ( PLT, 'PA800,',(1000-72*i):6,';PU' )           { position    } ;
          writeln ( PLT, 'CP1.0,-0.2' )                          { for label   } ;
          case i of                                    { label line segment     }
               1: writeln ( PLT, 'LBX Component (Crewman Reach Axes)',#3 )        ;
               2: writeln ( PLT, 'LBY Component (Crewman Reach Axes)',#3 )        ;
               3: writeln ( PLT, 'LBZ Component (Crewman Reach Axes)',#3 )        ;
               end ; { case i }
          end ;
     DRAW_FLEXURE_DATA_CURVES                                                      ;
     writeln ( PLT, 'PU;SP0' )                             { put pen in holder } ;
     end ;
```

```
$ page $

procedure DRAW_FLEXURE_DATA_CURVES ;

     var

          i : integer ;
          n : integer ;
          X : integer ;
          Y : integer ;

     begin
     writeln ( PLT, 'SP1' )                      { select pen #1 (black) } ;
     DRAW_FLEXURE_XAXIS                                                    ;
     writeln ( PLT, 'PA3600,0' )                                           ;
     writeln ( PLT, 'CP0.0,-2.2;LBTime (sec)',#3 )  { identify time axis    } ;
     DRAW_FLEXURE_YAXIS                                                    ;
     writeln ( PLT, 'PA0,0' )                                              ;
     writeln ( PLT, 'DI0,1' )                    { vertical label slant  } ;
     writeln ( PLT, 'CP-8.0,1.5;LBRMS Flexure (in)',#3 )                   ;
     writeln ( PLT, 'DI1,0' )                    { horizntl label slant  } ;
     writeln ( PLT, 'SP2'   )                    { select pen #2 (red)   } ;
     for i := 1 to 3 do
          begin
          case i of
               1: writeln ( PLT, 'LT1,0.3' )     { select dotted line    } ;
               2: writeln ( PLT, 'LT3,1.0' )     { select dashed line    } ;
               3: writeln ( PLT, 'LT'  )         { select solid  line    } ;
               end ; { case i }
          writeln ( PLT, 'PA500,',(1000-72*i):6,';PD' )                    ;
          writeln ( PLT, 'PA800,',(1000-72*i):6,';PU' ) { draw line segment } ;
          for n := 1 to SLOG^[0].NUMSLOGRECS do
               begin
               X := round( 100*SLOG^[n].TIME)                 { seconds } ;
               if X <= 4500 then
                    begin
                    Y := round( 100*12*SLOG^[n].FLEX_RO_R[i] )   { inches } ;
                    writeln ( PLT, 'PA',X:6,',',Y:6,';PD' )                ;
                    end ;
               end ;
          writeln ( PLT, 'PU' )                                            ;
          end ;
     end ;
```

```
$ page $

procedure DRAW_FLEXURE_XAXIS ;

     var

          X : integer ;

     begin
     for X := 0 to 45 do
          begin                { drawing 1-second & 5-second time tick marks }
          if ( X mod 5 ) = 0
               then write ( PLT, 'TL1.0,1.0;' )     { draw long  tick mark   }
               else write ( PLT, 'TL0.5,0.5;' )     { draw short tick mark   } ;
          writeln ( PLT, 'PA',(100*X):6,',0'  )     { go to the spot         } ;
          writeln ( PLT, 'PD;XT'              )     { lower pen & draw tick  } ;
          end ;
     writeln ( PLT, 'PU' )                          { raise pen              } ;
     for X := 1 to 9 do
          begin        { putting numeric label on every 5-second tick mark }
          write   ( PLT, 'PA',(500*X):6,',0;' )     { go to tick mark        } ;
          if X < 2
               then write ( PLT, 'CP-0.5,-1.2;' )   { back up 0.5 spaces     }
               else write ( PLT, 'CP-1.0,-1.2;' )   { back up 1.0 spaces     } ;
          writeln ( PLT, 'LB',(5*X):1,#3 )          { print the label        } ;
          end ;
     end ;


procedure DRAW_FLEXURE_YAXIS ;

     var

          Y : integer ;

     begin                           { drawing 1-inch & 2-inch tick marks }
     for Y := -6 to 6 do
          begin
          if ( Y mod 2 ) = 0
               then write ( PLT, 'TL1.0,1.0;' )     { draw long  tick mark   }
               else write ( PLT, 'TL0.5,0.5;' )     { draw short tick mark   } ;
          writeln ( PLT, 'PA0,',(100*Y):6 )         { go to the spot         } ;
          writeln ( PLT, 'PD;YT'          )         { lower pen & draw tick  } ;
          end ;
     writeln ( PLT, 'PU' )                          { raise pen              } ;
     for Y := -3 to 3 do
          begin                { putting numeric labels on 2-inch tick marks }
          write  ( PLT, 'PA0,',(200*Y):6,';' )                               ;
          if Y < 0
               then writeln ( PLT, 'CP-3.0,-0.2' )  { back up 3 spaces       }
               else writeln ( PLT, 'CP-2.0,-0.2' )  { back up 2 spaces       } ;
          writeln ( PLT, 'LB',(2*Y):1,#3 )                                   ;
          end ;
     end ;


end ;  { module MANHPOST & File 'Manhpost.I' }
```

2. TEST CASE RESULTS FROM HP-9000 MODEL 540 HOST COMPUTER

2.1. Palapa Capture

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:32:10 Mon 03 Nov 1986

"rms1" Translational Alacrity Matrix for RMS Flexure

Alacrity matrix, element [1,1]..........(ft/sec/sec/lb)	0.05000
Alacrity matrix, element [1,2]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [1,3]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [2,1]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [2,2]..........(ft/sec/sec/lb)	0.05000
Alacrity matrix, element [2,3]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [3,1]..........(ft/sec/sec/lb)	0.01750
Alacrity matrix, element [3,2]..........(ft/sec/sec/lb)	0.01750
Alacrity matrix, element [3,3]..........(ft/sec/sec/lb)	0.05000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:32:21 Mon 03 Nov 1986

"rms1" Translational Damping Constant Matrix for RMS Flexure

Damping constant matrix, element [1,1]......(lb/ft/sec)	10.00000
Damping constant matrix, element [1,2]......(lb/ft/sec)	0.00000
Damping constant matrix, element [1,3]......(lb/ft/sec)	0.00000
Damping constant matrix, element [2,1]......(lb/ft/sec)	0.00000
Damping constant matrix, element [2,2]......(lb/ft/sec)	10.00000
Damping constant matrix, element [2,3]......(lb/ft/sec)	0.00000
Damping constant matrix, element [3,1]......(lb/ft/sec)	0.00000
Damping constant matrix, element [3,2]......(lb/ft/sec)	0.00000
Damping constant matrix, element [3,3]......(lb/ft/sec)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:32:36 Mon 03 Nov 1986

"rms1" Translational Spring Constant Matrix for RMS Flexure

Spring constant matrix, element [1,1]...........(lb/ft)	20.00000
Spring constant matrix, element [1,2]...........(lb/ft)	0.00000
Spring constant matrix, element [1,3]...........(lb/ft)	0.00000
Spring constant matrix, element [2,1]...........(lb/ft)	0.00000
Spring constant matrix, element [2,2]...........(lb/ft)	40.00000
Spring constant matrix, element [2,3]...........(lb/ft)	0.00000
Spring constant matrix, element [3,1]...........(lb/ft)	0.00000
Spring constant matrix, element [3,2]...........(lb/ft)	0.00000
Spring constant matrix, element [3,3]...........(lb/ft)	150.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:32:52 Mon 03 Nov 1986

"nom" Orbiter Inertia Data

Orbiter weight.....................................(lb)	208342.00000
CM STA (structural x coordinate of mass center)....(in)	1107.80000
CM BL (structural y coordinate of mass center)....(in)	0.00000
CM WL (structural z coordinate of mass center)....(in)	375.90000
Ixx about CM, structural coordinates.......(slug*ft*ft)	921642.70000
Iyy about CM, structural coordinates.......(slug*ft*ft)	6861141.90000
Izz about CM, structural coordinates.......(slug*ft*ft)	7148541.30000
Pxy about CM, structural coordinates.......(slug*ft*ft)	12353.70000
Pxz about CM, structural coordinates.......(slug*ft*ft)	228384.60000
Pyz about CM, structural coordinates.......(slug*ft*ft)	484.50000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:33:09 Mon 03 Nov 1986

"plapa" Payload Inertia Data

Payload weight.....................................(lb)	1262.00000
CM STA (structural x coordinate of mass center)....(in)	-38.20000
CM BL (structural y coordinate of mass center)....(in)	0.00000
CM WL (structural z coordinate of mass center)....(in)	0.00000
Ixx about CM, structural coordinates.......(slug*ft*ft)	290.90000
Iyy about CM, structural coordinates.......(slug*ft*ft)	280.75470
Izz about CM, structural coordinates.......(slug*ft*ft)	254.94530
Pxy about CM, structural coordinates.......(slug*ft*ft)	0.00000
Pxz about CM, structural coordinates.......(slug*ft*ft)	0.00000
Pyz about CM, structural coordinates.......(slug*ft*ft)	1.80312

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:33:25 Mon 03 Nov 1986

"plapa" Position Parameters for Payload Flight Control

Nominal Orbiter STA of PL handling point...........(in)	800.00000
Nominal Orbiter BL of PL handling point...........(in)	24.00000
Nominal Orbiter WL of PL handling point...........(in)	600.00000
Payload STA of PL handle..........................(in)	-132.90800
Payload BL of PL handle...........................(in)	0.00000
Payload WL of PL handle...........................(in)	11.50000
Crewman reach limit from RO in +/- Rx direction....(in)	24.00000
Crewman reach limit from RO in +/- Ry direction....(in)	24.00000
Crewman reach limit from RO in +/- Rz direction....(in)	24.00000
Rx position tolerance for PL handle...............(in)	6.00000
Ry position tolerance for PL handle...............(in)	6.00000
Rz position tolerance for PL handle...............(in)	6.00000
Nominal vel for handle position correction.....(in/sec)	1.20000
Tolerance for corrective velocity..................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:33:46 Mon 03 Nov 1986

"plapa" Attitude Parameters for Payload Flight Control

Crewman pitch wrt Orbiter body axes...............(deg)	0.00000
Crewman yaw wrt Orbiter body axes...............(deg)	90.00000
Crewman roll wrt Orbiter body axes...............(deg)	0.00000
Desired Payload pitch wrt Crewman body axes.......(deg)	-90.00000
Desired Payload yaw wrt Crewman body axes.......(deg)	0.00000
Desired Payload roll wrt Crewman body axes.......(deg)	0.00000
Positive PL pitch limit wrt desired attitude......(deg)	90.00000
Positive PL yaw limit wrt desired attitude......(deg)	90.00000
Positive PL roll limit wrt desired attitude......(deg)	360.00000
Negative PL pitch limit wrt desired attitude......(deg)	-90.00000
Negative PL yaw limit wrt desired attitude......(deg)	-90.00000
Negative PL roll limit wrt desired attitude......(deg)	-360.00000
PL pitch tolerance wrt desired attitude...........(deg)	5.00000
PL yaw tolerance wrt desired attitude...........(deg)	5.00000
PL roll tolerance wrt desired attitude...........(deg)	5.00000
Nominal maneuver rate about PL Bx axis........(deg/sec)	1.00000
Nominal maneuver rate about PL By axis........(deg/sec)	1.00000
Nominal maneuver rate about PL Bz axis........(deg/sec)	1.00000
Maneuver rate tolerance............................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:34:02 Mon 03 Nov 1986

"plapa" Force & Torque Parameters for Payload Flight Control

Spinup axial (PL Bx) force....................(lb)	0.00000
Spinup normal (PL By) force....................(lb)	0.00000
Spinup tangential (PL Bz) force....................(lb)	0.00000
Despin axial (PL Bx) force....................(lb)	0.00000
Despin normal (PL By) force....................(lb)	0.00000
Despin tangential (PL Bz) force....................(lb)	0.00000
Axial (PL Bx) force limit for capture.........(lb)	2.00000
Normal (PL By) force limit for capture.........(lb)	2.00000
Tangential (PL Bz) force limit for capture.........(lb)	2.00000
Roll (PL Bx) torque limit for capture..........(ft*lb)	2.00000
Pitch (PL By) torque limit for capture..........(ft*lb)	2.00000
Yaw (PL Bz) torque limit for capture..........(ft*lb)	2.00000
Time constant for computing desired accelerations (sec)	5.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:34:26 Mon 03 Nov 1986

"plapa" Initial Conditions for the Simulation

Orbiter pitch wrt I axes..........................(deg)	0.00000
Orbiter yaw wrt I axes..........................(deg)	0.00000
Orbiter roll wrt I axes..........................(deg)	0.00000
Orbiter Bx component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter By component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter Bz component of ang vel wrt I axes....(deg/sec)	0.00000
Rx component of PL handle position.................(ft)	0.00000
Ry component of PL handle position.................(ft)	0.00000
Rz component of PL handle position.................(ft)	0.00000
Payload CM Xdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Ydot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Zdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload pitch wrt desired attitude................(deg)	0.00000
Payload yaw wrt desired attitude................(deg)	0.00000
Payload roll wrt desired attitude................(deg)	0.00000
Payload nominal spin rate.....................(deg/sec)	6.00000
Payload wobble cone angle.........................(deg)	10.00000
Payload wobble clok angle.........................(deg)	0.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:34:28 Mon 03 Nov 1986
Palapa Capture

Time = 0.000   DesMode = CAPTURE   CurMode = CAPTURE
PL Nomspin = 6.00   Wobble Cone = 10.00   Wobble Clok = -7.4L-17

Cntrl@H: R	-2.00	0.00	-2.00 Torq	-0.000	0.053	-0.000	Forc
Cntrl@H:PB	-2.00	0.00	2.00 Torq	-0.000	0.053	0.000	Forc
PB Axes:PD	0.00	0.00	0.00 PYR	6.000	0.008	-1.207	Rate
PL Hndl: R	-0.00	0.00	0.00 Pos	0.001	-0.066	-0.000	Vel
PB Axes:OB	-90.00	0.00	90.00 PYR	6.000	0.008	-1.207	Rate
PL CM :OB	25.65	1.04	-26.57 Pos	0.000	0.000	0.000	Vel
PL Rot K E	1.65						
PLAngmo: I	-90.00	10.00	30.93 PYMag	0.000	5.371	30.463	Hxyz
PB Axes: I	-90.00	0.00	90.00 PYR	6.000	0.008	-1.207	Rate
PL CM : I	-0.00	-0.96	-7.89 Pos	0.000	0.000	0.000	Vel
OB Axes: I	0.00	0.00	0.00 PYR	0.000	0.000	0.000	Rate
Orb CM : I	-25.65	-2.00	18.68 Pos	0.000	0.000	0.000	Vel

Time	Stepsize
0.000	0.200

Time (sec)	zFlex (in)	DesMode	CurMode	T_PBx (ft*lb)	T_PBy (ft*lb)	T_PBz (ft*lb)	F_PBx (lb)	F_PBy (lb)	F_PBz (lb)
0.000	0.00	CAPTURE	CAPTURE	-2.00	0.00	2.00	-0.00	0.05	0.00
0.100	0.00	CAPTURE	CAPTURE	-2.00	0.00	2.00	-0.00	0.05	0.00
0.200	0.00	CAPTURE	CAPTURE	-2.00	-0.00	2.00	-0.00	0.05	0.00
0.300	0.00	CAPTURE	CAPTURE	-2.00	-0.00	2.00	-0.00	0.04	0.00
0.400	0.00	CAPTURE	CAPTURE	-2.00	-0.01	2.00	-0.00	0.04	0.00
0.500	0.00	CAPTURE	CAPTURE	-2.00	-0.01	2.00	-0.00	0.03	-0.00
0.600	0.00	CAPTURE	CAPTURE	-2.00	-0.01	2.00	-0.00	0.03	0.00
0.700	0.00	CAPTURE	CAPTURE	-2.00	-0.01	2.00	-0.00	0.02	0.00
0.800	0.00	CAPTURE	CAPTURE	-2.00	-0.01	1.95	-0.00	0.02	0.00
0.900	0.00	CAPTURE	CAPTURE	-2.00	-0.02	1.60	-0.00	0.01	0.00
1.000	0.00	CAPTURE	CAPTURE	-2.00	-0.02	1.32	-0.00	0.01	0.00
1.100	0.00	CAPTURE	CAPTURE	-2.00	-0.02	1.16	-0.00	0.00	0.00
1.200	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.97	-0.00	0.00	0.00
1.300	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.85	-0.00	0.00	0.00
1.400	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.72	-0.00	0.00	0.00
1.500	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.63	-0.00	0.00	0.00
1.600	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.54	-0.00	-0.00	0.00
1.700	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.49	-0.00	-0.00	0.00
1.800	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.43	-0.00	-0.00	0.00
1.900	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.39	-0.00	-0.00	0.00
2.000	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.35	-0.00	-0.00	0.00
2.100	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.33	-0.00	-0.00	0.00
2.200	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.30	-0.00	-0.00	0.00
2.300	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.29	0.00	-0.00	0.00
2.400	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.27	0.00	-0.00	0.00
2.500	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.26	0.00	-0.00	0.00
2.600	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.25	0.00	-0.00	0.00
2.700	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.24	0.00	-0.00	0.00
2.800	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.24	0.00	-0.00	0.00
2.900	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.23	0.00	-0.00	0.00
3.000	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.23	0.00	-0.00	0.00
3.100	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.22	0.00	-0.00	0.00
3.200	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.22	0.00	-0.00	0.00
3.300	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.22	0.00	-0.00	0.00
3.400	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.22	0.00	-0.00	0.00
3.500	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.22	0.00	-0.00	0.00
3.600	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.22	0.00	-0.00	0.00
3.700	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
3.800	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
3.900	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.000	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.100	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.200	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.300	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.400	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.500	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.600	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.700	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.800	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
4.900	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
5.000	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
5.100	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
5.200	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
5.300	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.21	0.00	-0.00	0.00
5.400	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
5.500	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
5.600	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00

5.700	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
5.800	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
5.900	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.000	0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.100	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.200	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.300	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.400	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.500	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.600	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.700	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.800	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
6.900	-0.00	CAPTURE	CAPTURE	-2.00	-0.02	0.20	0.00	-0.00	0.00
7.000	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.100	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.200	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.300	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.400	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.500	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.600	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.700	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.800	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
7.900	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.000	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.100	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.200	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.300	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.400	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.500	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.600	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.700	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.800	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
8.900	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
9.000	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.20	0.00	-0.00	0.00
9.100	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.200	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.300	-0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.400	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.500	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.600	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.700	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.800	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
9.900	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
10.000	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
10.100	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
10.200	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	0.00	-0.00	0.00
10.300	0.00	CAPTURE	CAPTURE	-2.00	-0.01	0.19	-0.00	-0.00	0.00
10.400	0.00	CAPTURE	CAPTURE	-1.98	-0.01	0.19	-0.00	-0.00	0.00
10.500	-0.00	CAPTURE	CAPTURE	-1.94	-0.01	0.19	-0.00	-0.00	0.00
10.600	-0.00	CAPTURE	CAPTURE	-1.90	-0.01	0.19	-0.00	-0.00	0.00
10.700	-0.00	CAPTURE	CAPTURE	-1.86	-0.01	0.19	-0.00	-0.00	0.00
10.800	-0.00	CAPTURE	CAPTURE	-1.82	-0.01	0.19	-0.00	-0.00	0.00
10.900	-0.00	CAPTURE	CAPTURE	-1.79	-0.01	0.18	-0.00	-0.00	0.00
11.000	-0.00	CAPTURE	CAPTURE	-1.75	-0.01	0.18	-0.00	-0.00	0.00
11.100	-0.00	CAPTURE	CAPTURE	-1.72	-0.01	0.18	-0.00	-0.00	0.00
11.200	-0.00	CAPTURE	CAPTURE	-1.68	-0.01	0.18	-0.00	-0.00	0.00
11.300	-0.00	CAPTURE	CAPTURE	-1.65	-0.01	0.17	-0.00	-0.00	0.00
11.400	-0.00	CAPTURE	CAPTURE	-1.62	-0.01	0.17	-0.00	-0.00	0.00
11.500	0.00	CAPTURE	CAPTURE	-1.59	-0.01	0.17	-0.00	-0.00	0.00
11.600	0.00	CAPTURE	CAPTURE	-1.55	-0.01	0.16	-0.00	-0.00	0.00

11.700	0.00	CAPTURE	CAPTURE	-1.52	-0.01	0.16	-0.00	-0.00	0.00
11.800	0.00	CAPTURE	CAPTURE	-1.49	-0.01	0.16	-0.00	-0.00	0.00
11.900	0.00	CAPTURE	CAPTURE	-1.46	-0.01	0.16	-0.00	-0.00	0.00
12.000	0.00	CAPTURE	CAPTURE	-1.43	-0.01	0.15	-0.00	-0.00	0.00
12.100	0.00	CAPTURE	CAPTURE	-1.41	-0.01	0.15	-0.00	-0.00	0.00
12.200	0.00	CAPTURE	CAPTURE	-1.38	-0.01	0.15	-0.00	-0.00	0.00
12.300	0.00	CAPTURE	CAPTURE	-1.35	-0.01	0.14	-0.00	-0.00	0.00
12.400	0.00	CAPTURE	CAPTURE	-1.32	-0.01	0.14	-0.00	-0.00	0.00
12.500	0.00	CAPTURE	CAPTURE	-1.30	-0.01	0.14	-0.00	-0.00	0.00
12.600	0.00	CAPTURE	CAPTURE	-1.27	-0.00	0.14	-0.00	-0.00	0.00
12.700	-0.00	CAPTURE	CAPTURE	-1.25	-0.00	0.13	-0.00	-0.00	0.00
12.800	-0.00	CAPTURE	CAPTURE	-1.22	-0.00	0.13	-0.00	-0.00	0.00
12.900	-0.00	CAPTURE	CAPTURE	-1.20	-0.00	0.13	-0.00	-0.00	0.00
13.000	-0.00	CAPTURE	CAPTURE	-1.17	-0.00	0.13	-0.00	-0.00	0.00
13.100	-0.00	CAPTURE	CAPTURE	-1.15	-0.00	0.12	-0.00	-0.00	0.00
13.200	-0.00	CAPTURE	CAPTURE	-1.13	-0.00	0.12	-0.00	-0.00	0.00
13.300	-0.00	CAPTURE	CAPTURE	-1.11	-0.00	0.12	-0.00	-0.00	0.00
13.400	-0.00	CAPTURE	CAPTURE	-1.08	-0.00	0.12	-0.00	-0.00	0.00
13.500	-0.00	CAPTURE	CAPTURE	-1.06	-0.00	0.11	0.00	-0.00	0.00
13.600	-0.00	CAPTURE	CAPTURE	-1.04	-0.00	0.11	0.00	-0.00	0.00
13.700	-0.00	CAPTURE	CAPTURE	-1.02	-0.00	0.11	0.00	-0.00	0.00
13.800	0.00	CAPTURE	CAPTURE	-1.00	-0.00	0.11	0.00	-0.00	0.00
13.900	0.00	CAPTURE	CAPTURE	-0.98	-0.00	0.10	0.00	-0.00	0.00
14.000	0.00	CAPTURE	CAPTURE	-0.96	-0.00	0.10	0.00	-0.00	0.00
14.100	0.00	CAPTURE	CAPTURE	-0.94	-0.00	0.10	0.00	-0.00	0.00
14.200	0.00	CAPTURE	CAPTURE	-0.92	-0.00	0.10	0.00	-0.00	0.00
14.300	0.00	CAPTURE	CAPTURE	-0.91	-0.00	0.10	0.00	-0.00	0.00
14.400	0.00	CAPTURE	CAPTURE	-0.89	-0.00	0.09	0.00	-0.00	0.00
14.500	0.00	CAPTURE	CAPTURE	-0.87	-0.00	0.09	0.00	-0.00	0.00
14.600	0.00	CAPTURE	CAPTURE	-0.85	-0.00	0.09	0.00	-0.00	0.00
14.700	0.00	CAPTURE	CAPTURE	-0.84	-0.00	0.09	0.00	-0.00	0.00
14.800	0.00	CAPTURE	CAPTURE	-0.82	-0.00	0.07	0.00	-0.00	0.00
14.900	-0.00	CAPTURE	CAPTURE	-0.81	-0.00	0.07	-0.00	-0.00	0.00
15.000	-0.00	CAPTURE	CAPTURE	-0.79	-0.00	0.06	-0.00	-0.00	0.00
15.100	-0.00	CAPTURE	CAPTURE	-0.77	-0.00	0.05	-0.00	-0.00	0.00
15.200	-0.00	CAPTURE	CAPTURE	-0.76	-0.00	0.05	-0.00	0.00	0.00
15.300	-0.00	CAPTURE	CAPTURE	-0.74	-0.00	0.05	-0.00	0.00	0.00
15.400	-0.00	CAPTURE	CAPTURE	-0.73	-0.00	0.04	-0.00	0.00	0.00
15.500	-0.00	CAPTURE	CAPTURE	-0.72	-0.00	0.04	-0.00	0.00	0.00
15.600	-0.00	CAPTURE	CAPTURE	-0.70	-0.00	0.04	-0.00	0.00	0.00
15.700	0.00	CAPTURE	CAPTURE	-0.69	-0.00	0.04	-0.00	0.00	0.00
15.800	0.00	CAPTURE	CAPTURE	-0.67	-0.00	0.04	-0.00	0.00	0.00
15.900	0.00	CAPTURE	CAPTURE	-0.66	-0.00	0.04	-0.00	0.00	0.00
16.000	0.00	CAPTURE	CAPTURE	-0.65	-0.00	0.04	-0.00	0.00	0.00
16.100	0.00	CAPTURE	CAPTURE	-0.63	-0.00	0.04	-0.00	0.00	0.00
16.200	0.00	CAPTURE	CAPTURE	-0.62	-0.00	0.04	-0.00	0.00	0.00
16.300	0.00	CAPTURE	CAPTURE	-0.61	-0.00	0.04	-0.00	0.00	0.00
16.400	0.00	CAPTURE	CAPTURE	-0.60	-0.00	0.03	-0.00	0.00	0.00
16.500	0.00	CAPTURE	CAPTURE	-0.59	-0.00	0.03	-0.00	0.00	0.00
16.600	0.00	CAPTURE	CAPTURE	-0.57	-0.00	0.03	-0.00	0.00	0.00
16.700	0.00	CAPTURE	CAPTURE	-0.56	-0.00	0.03	-0.00	0.00	0.00
16.800	0.00	CAPTURE	CAPTURE	-0.55	-0.00	0.03	-0.00	0.00	0.00
16.900	0.00	CAPTURE	CAPTURE	-0.54	-0.00	0.03	-0.00	0.00	0.00
17.000	0.00	CAPTURE	CAPTURE	-0.53	-0.00	0.03	-0.00	0.00	0.00
17.100	0.00	CAPTURE	CAPTURE	-0.52	-0.00	0.03	-0.00	0.00	0.00
17.200	0.00	CAPTURE	CAPTURE	-0.51	-0.00	0.03	-0.00	0.00	0.00
17.300	0.00	CAPTURE	CAPTURE	-0.50	-0.00	0.03	-0.00	0.00	0.00
17.400	0.00	CAPTURE	CAPTURE	-0.49	-0.00	0.03	-0.00	0.00	0.00
17.500	0.00	CAPTURE	CAPTURE	-0.48	-0.00	0.03	-0.00	0.00	0.00
17.600	0.00	CAPTURE	CAPTURE	-0.47	-0.00	0.03	-0.00	0.00	0.00

17.700	0.00	CAPTURE	CAPTURE	-0.46	-0.00	0.03	-0.00	0.00	0.00
17.800	0.00	CAPTURE	CAPTURE	-0.45	-0.00	0.03	-0.00	0.00	0.00
17.900	0.00	CAPTURE	CAPTURE	-0.44	-0.00	0.03	-0.00	0.00	0.00
18.000	0.00	CAPTURE	CAPTURE	-0.43	-0.00	0.03	-0.00	0.00	0.00
18.100	0.00	CAPTURE	CAPTURE	-0.42	-0.00	0.03	-0.00	0.00	0.00
18.200	0.00	CAPTURE	CAPTURE	-0.42	-0.00	0.03	-0.00	0.00	0.00
18.300	0.00	CAPTURE	CAPTURE	-0.41	-0.00	0.03	-0.00	0.00	0.00
18.400	0.00	CAPTURE	CAPTURE	-0.40	-0.00	0.03	-0.00	0.00	0.00
18.500	0.00	CAPTURE	CAPTURE	-0.39	-0.00	0.03	-0.00	0.00	0.00
18.600	0.00	CAPTURE	CAPTURE	-0.38	-0.00	0.03	-0.00	0.00	0.00
18.700	0.00	CAPTURE	CAPTURE	-0.38	-0.00	0.03	-0.00	0.00	0.00
18.800	0.00	CAPTURE	CAPTURE	-0.37	-0.00	0.02	-0.00	0.00	0.00
18.900	0.00	CAPTURE	CAPTURE	-0.36	-0.00	0.02	-0.00	0.00	0.00
19.000	0.00	CAPTURE	CAPTURE	-0.35	0.00	0.02	-0.00	0.00	0.00
19.100	0.00	CAPTURE	CAPTURE	-0.35	0.00	0.02	-0.00	0.00	0.00
19.200	0.00	CAPTURE	CAPTURE	-0.34	0.00	0.02	-0.00	0.00	0.00
19.300	0.00	CAPTURE	CAPTURE	-0.33	0.00	0.02	-0.00	0.00	0.00
19.400	0.00	CAPTURE	CAPTURE	-0.33	0.00	0.02	-0.00	0.00	0.00
19.500	0.00	CAPTURE	CAPTURE	-0.32	0.00	0.02	-0.00	0.00	0.00
19.600	0.00	CAPTURE	CAPTURE	-0.31	0.00	0.02	-0.00	0.00	0.00
19.700	0.00	CAPTURE	CAPTURE	-0.31	0.00	0.02	-0.00	0.00	0.00
19.800	0.00	CAPTURE	CAPTURE	-0.30	0.00	0.02	-0.00	0.00	0.00
19.900	0.00	CAPTURE	CAPTURE	-0.30	0.00	0.02	-0.00	0.00	0.00
20.000	0.00	CAPTURE	CAPTURE	-0.29	0.00	0.02	-0.00	0.00	0.00
20.100	0.00	CAPTURE	CAPTURE	-0.28	0.00	0.02	-0.00	0.00	0.00
20.200	0.00	CAPTURE	CAPTURE	-0.28	0.00	0.02	-0.00	0.00	0.00
20.300	0.00	CAPTURE	CAPTURE	-0.27	0.00	0.02	-0.00	0.00	0.00
20.400	0.00	CAPTURE	CAPTURE	-0.27	0.00	0.02	-0.00	0.00	0.00
20.500	0.00	CAPTURE	CAPTURE	-0.26	0.00	0.02	-0.00	0.00	0.00
20.600	0.00	CAPTURE	CAPTURE	-0.26	0.00	0.02	-0.00	0.00	0.00
20.700	0.00	CAPTURE	CAPTURE	-0.25	0.00	0.02	-0.00	0.00	0.00
20.800	0.00	CAPTURE	CAPTURE	-0.25	0.00	0.02	-0.00	0.00	0.00
20.900	0.00	CAPTURE	CAPTURE	-0.24	0.00	0.02	-0.00	0.00	0.00
21.000	0.00	CAPTURE	CAPTURE	-0.24	0.00	0.02	-0.00	0.00	0.00
21.100	0.00	CAPTURE	CAPTURE	-0.23	0.00	0.02	-0.00	0.00	0.00
21.200	0.00	CAPTURE	CAPTURE	-0.23	0.00	0.02	-0.00	0.00	0.00
21.300	0.00	CAPTURE	CAPTURE	-0.22	0.00	0.02	-0.00	0.00	0.00
21.400	0.00	CAPTURE	CAPTURE	-0.22	0.00	0.02	-0.00	0.00	0.00
21.500	0.00	CAPTURE	CAPTURE	-0.21	0.00	0.02	-0.00	0.00	0.00
21.600	0.00	CAPTURE	CAPTURE	-0.21	0.00	0.02	-0.00	0.00	0.00
21.700	0.00	CAPTURE	CAPTURE	-0.21	0.00	0.02	-0.00	0.00	0.00
21.800	0.00	CAPTURE	CAPTURE	-0.20	0.00	0.02	-0.00	0.00	0.00
21.900	0.00	CAPTURE	CAPTURE	-0.20	0.00	0.02	-0.00	0.00	0.00
22.000	0.00	CAPTURE	CAPTURE	-0.19	0.00	0.02	-0.00	0.00	0.00
22.100	0.00	CAPTURE	CAPTURE	-0.19	0.00	0.02	-0.00	0.00	0.00
22.200	0.00	CAPTURE	CAPTURE	-0.19	0.00	0.02	-0.00	0.00	0.00
22.300	0.00	CAPTURE	CAPTURE	-0.18	0.00	0.02	-0.00	0.00	0.00
22.400	0.00	CAPTURE	CAPTURE	-0.18	0.00	0.02	-0.00	0.00	0.00
22.500	0.00	CAPTURE	CAPTURE	-0.18	0.00	0.02	-0.00	0.00	0.00
22.600	0.00	CAPTURE	CAPTURE	-0.17	0.00	0.01	-0.00	0.00	0.00
22.700	0.00	CAPTURE	CAPTURE	-0.17	0.00	0.01	-0.00	0.00	0.00
22.800	0.00	CAPTURE	CAPTURE	-0.17	0.00	0.01	-0.00	0.00	0.00
22.900	0.00	CAPTURE	CAPTURE	-0.16	0.00	0.01	-0.00	0.00	0.00
23.000	0.00	CAPTURE	CAPTURE	-0.16	0.00	0.01	-0.00	0.00	0.00
23.100	0.00	CAPTURE	CAPTURE	-0.16	0.00	0.01	-0.00	0.00	0.00
23.200	0.00	CAPTURE	CAPTURE	-0.15	0.00	0.01	-0.00	0.00	0.00
23.300	0.00	CAPTURE	CAPTURE	-0.15	0.00	0.01	-0.00	0.00	0.00
23.400	0.00	CAPTURE	CAPTURE	-0.15	0.00	0.01	-0.00	0.00	0.00
23.500	0.00	CAPTURE	CAPTURE	-0.14	0.00	0.01	-0.00	0.00	0.00
23.600	0.00	CAPTURE	CAPTURE	-0.14	0.00	0.01	-0.00	0.00	0.00

23.700	0.00	CAPTURE	CAPTURE	-0.14	0.00	0.01	-0.00	0.00	0.00
23.800	0.00	CAPTURE	CAPTURE	-0.14	0.00	0.01	-0.00	0.00	0.00
23.900	0.00	CAPTURE	CAPTURE	-0.13	0.00	0.01	-0.00	0.00	0.00
24.000	0.00	CAPTURE	CAPTURE	-0.13	0.00	0.01	-0.00	0.00	0.00
24.100	0.00	CAPTURE	CAPTURE	-0.13	0.00	0.01	-0.00	0.00	0.00
24.200	0.00	CAPTURE	CAPTURE	-0.12	0.00	0.01	-0.00	0.00	0.00
24.300	0.00	CAPTURE	CAPTURE	-0.12	0.00	0.01	-0.00	0.00	0.00
24.400	0.00	CAPTURE	CAPTURE	-0.12	0.00	0.01	-0.00	0.00	0.00
24.500	0.00	CAPTURE	CAPTURE	-0.12	0.00	0.01	-0.00	0.00	0.00
24.600	0.00	CAPTURE	CAPTURE	-0.12	0.00	0.01	-0.00	0.00	0.00
24.700	0.00	CAPTURE	CAPTURE	-0.11	0.00	0.01	-0.00	0.00	0.00
24.800	0.00	CAPTURE	CAPTURE	-0.11	0.00	0.01	-0.00	0.00	0.00
24.900	0.00	CAPTURE	CAPTURE	-0.11	0.00	0.01	-0.00	0.00	0.00
25.000	0.00	CAPTURE	CAPTURE	-0.11	0.00	0.01	-0.00	0.00	0.00
25.100	0.00	CAPTURE	CAPTURE	-0.10	0.00	0.01	-0.00	0.00	0.00
25.200	0.00	CAPTURE	CAPTURE	-0.10	0.00	0.01	-0.00	0.00	0.00
25.300	0.00	CAPTURE	HOLD	-0.10	0.00	0.01	-0.00	0.00	0.00
25.400	0.00	CAPTURE	HOLD	-0.09	0.00	0.01	-0.00	0.00	0.00
25.500	0.00	CAPTURE	HOLD	-0.09	0.00	0.01	-0.00	0.00	0.00
25.600	0.00	CAPTURE	HOLD	-0.09	0.00	0.02	-0.00	0.00	0.00
25.700	0.00	CAPTURE	HOLD	-0.08	0.00	0.02	-0.00	0.00	0.00
25.800	0.00	CAPTURE	HOLD	-0.08	0.00	0.02	-0.00	0.00	0.00
25.900	0.00	CAPTURE	HOLD	-0.08	0.00	0.02	-0.00	0.00	0.00
26.000	0.00	CAPTURE	HOLD	-0.08	0.00	0.02	-0.00	0.00	0.00
26.100	0.00	CAPTURE	HOLD	-0.07	0.00	0.02	-0.00	0.00	0.00
26.200	0.00	CAPTURE	HOLD	-0.07	0.00	0.02	-0.00	0.00	0.00
26.300	0.00	CAPTURE	HOLD	-0.07	0.00	0.02	-0.00	0.00	0.00
26.400	0.00	CAPTURE	HOLD	-0.07	0.00	0.02	-0.00	0.00	0.00
26.500	0.00	CAPTURE	HOLD	-0.06	0.00	0.02	-0.00	0.00	0.00
26.600	0.00	CAPTURE	HOLD	-0.06	0.00	0.02	-0.00	0.00	0.00
26.700	0.00	CAPTURE	HOLD	-0.06	0.00	0.02	-0.00	0.00	0.00
26.800	0.00	CAPTURE	HOLD	-0.06	0.00	0.02	-0.00	0.00	0.00
26.900	0.00	CAPTURE	HOLD	-0.06	0.00	0.02	-0.00	0.00	0.00
27.000	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.100	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.200	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.300	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.400	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.500	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.600	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.700	0.00	CAPTURE	HOLD	-0.05	0.00	0.01	-0.00	0.00	0.00
27.800	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
27.900	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.000	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.100	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.200	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.300	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.400	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.500	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.600	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.700	0.00	CAPTURE	HOLD	-0.04	0.00	0.01	-0.00	0.00	0.00
28.800	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
28.900	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.000	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.100	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.200	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.300	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.400	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.500	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.600	0.00	CAPTURE	HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00

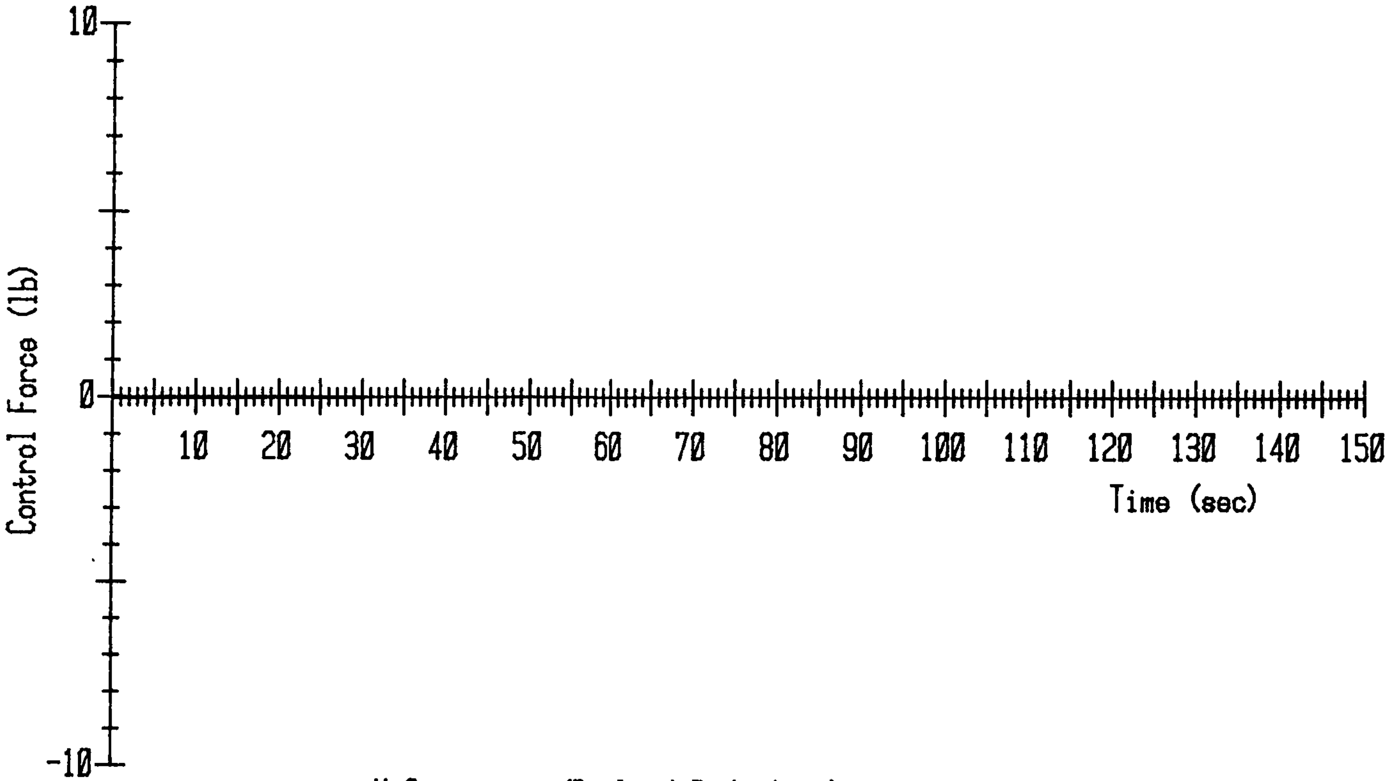
29.700	0.00	CAPTURE HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.800	0.00	CAPTURE HOLD	-0.03	0.00	0.01	-0.00	0.00	0.00
29.900	0.00	CAPTURE HOLD	-0.03	-0.00	0.01	-0.00	0.00	0.00
30.000	0.00	CAPTURE HOLD	-0.03	-0.00	0.01	-0.00	0.00	0.00

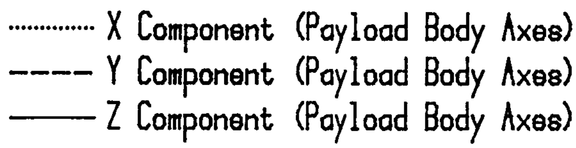
Time = 30.000   DesMode = CAPTURE   CurMode = HOLD
PL Nomspin = 0.05   Wobble Cone = 18.46   Wobble Clok = 6.91

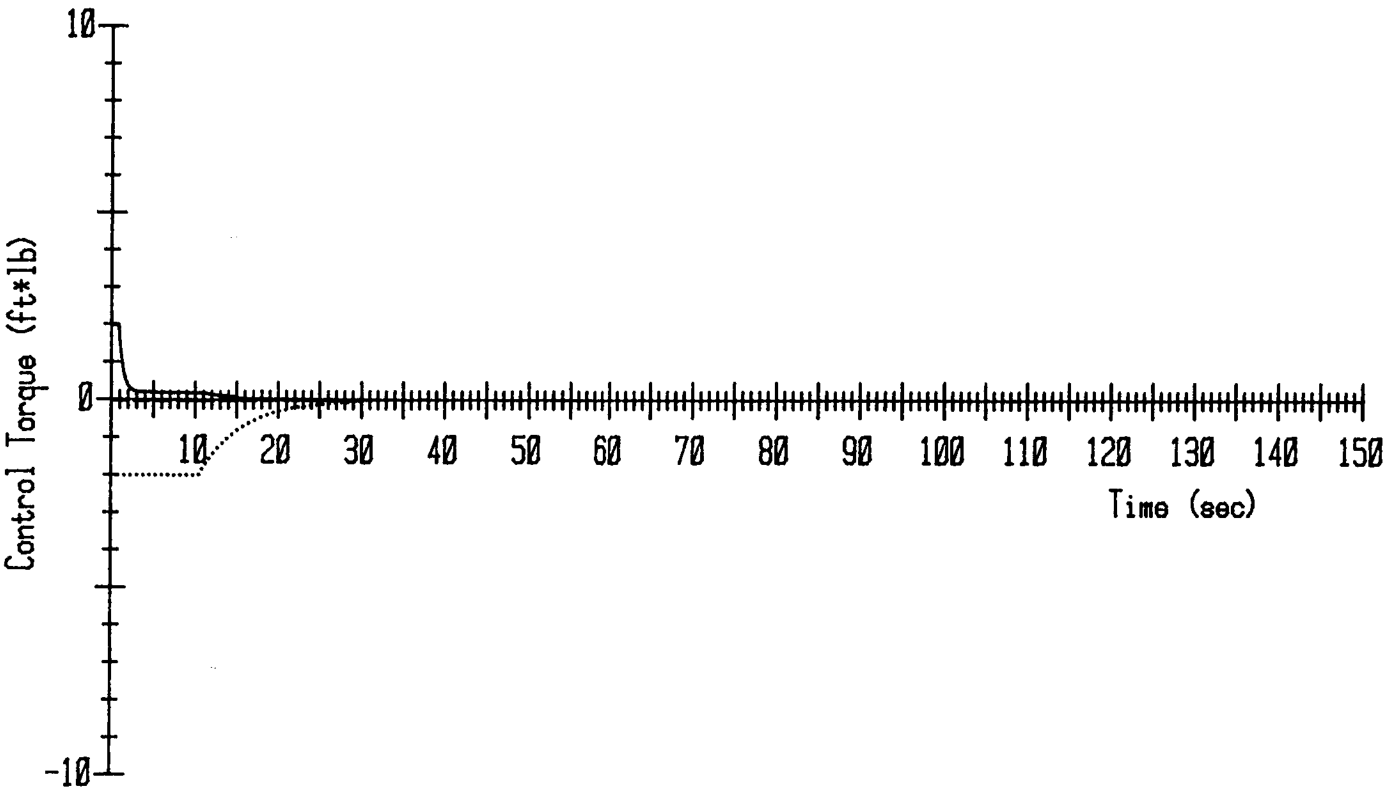
Cntrl@H: R	-0.01	-0.00	-0.03	Torq	-0.000	0.000	-0.000	Forc
Cntrl@H:PB	-0.03	-0.00	0.01	Torq	-0.000	0.000	0.000	Forc
PB Axes:PD	2.91	-5.52	50.70	PYR	0.051	0.002	-0.019	Rate
PL Hndl: R	0.05	-0.01	-0.01	Pos	0.001	-0.001	-0.000	Vel
PB Axes:OB	-84.47	2.90	140.42	PYR	0.051	0.002	-0.019	Rate
PL CM :OB	25.64	1.04	-26.57	Pos	-0.000	-0.000	-0.000	Vel
PL Rot K E	0.00							
PLAngmo: I	-100.32	12.63	0.28	PYMag	-0.048	0.060	0.264	Hxyz
PB Axes: I	-84.47	2.90	140.43	PYR	0.051	0.002	-0.019	Rate
PL CM : I	-0.01	-0.96	-7.89	Pos	-0.000	-0.000	-0.000	Vel
OB Axes: I	0.00	0.01	0.00	PYR	0.000	0.000	0.000	Rate
Orb CM : I	-25.65	-2.00	18.68	Pos	0.000	0.000	0.000	Vel

*** END OF SIMULATION ***
Palapa Capture
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:34:28 Mon 03 Nov 1986

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:34:28 Mon 03 Nov 1986







Palapa Capture

2.2. Syncom 10#/24" Despin Stroke Displaced Aft 6"

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:58:45 Mon 03 Nov 1986

"sncmc" Payload Inertia Data

Payload weight......................................(lb)	15267.00000
CM STA (structural x coordinate of mass center)....(in)	-17.70378
CM BL (structural y coordinate of mass center)....(in)	-0.03615
CM WL (structural z coordinate of mass center)....(in)	0.08976
Ixx about CM, structural coordinates.......(slug*ft*ft)	6386.85867
Iyy about CM, structural coordinates.......(slug*ft*ft)	4428.63556
Izz about CM, structural coordinates.......(slug*ft*ft)	5276.48977
Pxy about CM, structural coordinates.......(slug*ft*ft)	-40.46375
Pxz about CM, structural coordinates.......(slug*ft*ft)	-26.30184
Pyz about CM, structural coordinates.......(slug*ft*ft)	30.36689

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:59:07 Mon 03 Nov 1986

"dsda6" Position Parameters for Payload Flight Control

Nominal Orbiter STA of PL handling point...........(in)	800.00000
Nominal Orbiter BL of PL handling point...........(in)	24.00000
Nominal Orbiter WL of PL handling point...........(in)	600.00000
Payload STA of PL handle...........................(in)	-11.70378
Payload BL of PL handle...........................(in)	-86.12500
Payload WL of PL handle...........................(in)	0.00000
Crewman reach limit from RO in +/- Rx direction....(in)	12.00000
Crewman reach limit from RO in +/- Ry direction....(in)	12.00000
Crewman reach limit from RO in +/- Rz direction....(in)	12.00000
Rx position tolerance for PL handle...............(in)	6.00000
Ry position tolerance for PL handle...............(in)	6.00000
Rz position tolerance for PL handle...............(in)	6.00000
Nominal vel for handle position correction.....(in/sec)	0.00000
Tolerance for corrective velocity..................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:59:33 Mon 03 Nov 1986

"cap" Attitude Parameters for Payload Flight Control

Crewman pitch wrt Orbiter body axes...............(deg)	0.00000
Crewman yaw wrt Orbiter body axes...............(deg)	90.00000
Crewman roll wrt Orbiter body axes...............(deg)	0.00000
Desired Payload pitch wrt Crewman body axes.......(deg)	0.00000
Desired Payload yaw wrt Crewman body axes.......(deg)	90.00000
Desired Payload roll wrt Crewman body axes.......(deg)	180.00000
Positive PL pitch limit wrt desired attitude......(deg)	15.00000
Positive PL yaw limit wrt desired attitude......(deg)	15.00000
Positive PL roll limit wrt desired attitude......(deg)	20.00000
Negative PL pitch limit wrt desired attitude......(deg)	-15.00000
Negative PL yaw limit wrt desired attitude......(deg)	-15.00000
Negative PL roll limit wrt desired attitude......(deg)	-10.00000
PL pitch tolerance wrt desired attitude...........(deg)	2.00000
PL yaw tolerance wrt desired attitude...........(deg)	2.00000
PL roll tolerance wrt desired attitude...........(deg)	2.00000
Nominal maneuver rate about PL Bx axis........(deg/sec)	0.50000
Nominal maneuver rate about PL By axis........(deg/sec)	0.50000
Nominal maneuver rate about PL Bz axis........(deg/sec)	0.50000
Maneuver rate tolerance............................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 10:59:56 Mon 03 Nov 1986

"lim10" Force & Torque Parameters for Payload Flight Control

Spinup axial (PL Bx) force....................(lb)	0.00000
Spinup normal (PL By) force....................(lb)	0.00000
Spinup tangential (PL Bz) force....................(lb)	-10.00000
Despin axial (PL Bx) force....................(lb)	0.00000
Despin normal (PL By) force....................(lb)	0.00000
Despin tangential (PL Bz) force....................(lb)	10.00000
Axial (PL Bx) force limit for capture.........(lb)	10.00000
Normal (PL By) force limit for capture.........(lb)	10.00000
Tangential (PL Bz) force limit for capture.........(lb)	10.00000
Roll (PL Bx) torque limit for capture..........(ft*lb)	10.00000
Pitch (PL By) torque limit for capture..........(ft*lb)	10.00000
Yaw (PL Bz) torque limit for capture..........(ft*lb)	10.00000
Time constant for computing desired accelerations (sec)	5.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:00:20 Mon 03 Nov 1986

"fds" Initial Conditions for the Simulation

Orbiter pitch wrt I axes..........................(deg)	0.00000
Orbiter yaw wrt I axes..........................(deg)	0.00000
Orbiter roll wrt I axes..........................(deg)	0.00000
Orbiter Bx component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter By component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter Bz component of ang vel wrt I axes....(deg/sec)	0.00000
Rx component of PL handle position.................(ft)	1.83333
Ry component of PL handle position.................(ft)	0.00000
Rz component of PL handle position.................(ft)	-4.80000
Payload CM Xdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Ydot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Zdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload pitch wrt desired attitude................(deg)	0.00000
Payload yaw wrt desired attitude................(deg)	0.00000
Payload roll wrt desired attitude................(deg)	-42.00000
Payload nominal spin rate.....................(deg/sec)	6.00000
Payload wobble cone angle.........................(deg)	1.76118
Payload wobble clok angle.........................(deg)	-138.70817

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:00:23 Mon 03 Nov 1986
Syncom 10#/24" Despin Stroke Displaced Aft 6"

Time = 0.000 DesMode = DESPIN CurMode = FREE
PL Nomspin = 6.00 Wobble Cone = 1.76 Wobble Clok = -138.71

Cntrl@H: R	0.00	0.00	0.00 Torq	0.000	0.000	0.000	Forc
Cntrl@H:PB	0.00	0.00	0.00 Torq	0.000	0.000	0.000	Forc
PB Axes:PD	0.00	4.8L-30	-42.00 PYR	5.999	-0.122	0.139	Rate
PL Hndl: R	1.83	0.00	-4.80 Pos	-0.505	0.017	0.558	Vel
PB Axes:OB	180.00	0.00	-42.00 PYR	5.999	-0.122	0.139	Rate
PL CM :OB	25.15	9.16	-18.67 Pos	0.000	0.000	0.000	Vel
PL Rot K E	35.05						
PLAngmo: I	178.24	0.02	669.15 PYMag	-668.830	0.254	-20.564	Hxyz
PB Axes: I	180.00	0.00	-42.00 PYR	5.999	-0.122	0.139	Rate
PL CM : I	-0.50	7.16	0.01 Pos	0.000	0.000	0.000	Vel
OB Axes: I	0.00	0.00	0.00 PYR	0.000	0.000	0.000	Rate
Orb CM : I	-25.65	-2.00	18.68 Pos	0.000	0.000	0.000	Vel

Time	Stepsize
0.000	0.200

Time (sec)	zFlex (in)	DesMode	CurMode	T_PBx (ft*lb)	T_PBy (ft*lb)	T_PBz (ft*lb)	F_PBx (lb)	F_PBy (lb)	F_PBz (lb)
0.000	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.100	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.200	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.300	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.400	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.500	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.600	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.700	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.800	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
0.900	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.000	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.100	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.200	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.300	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.400	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.500	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.600	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.700	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.800	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
1.900	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.000	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.100	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.200	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.300	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.400	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.500	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.600	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.700	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.800	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
2.900	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.000	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.100	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.200	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.300	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.400	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.500	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.600	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.700	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.800	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
3.900	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.000	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.100	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.200	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.300	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.400	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.500	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.600	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.700	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.800	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
4.900	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
5.000	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
5.100	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
5.200	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
5.300	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
5.400	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
5.500	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)
5.600	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok =	-138.71	(deg)

5.700	0.00	DESPIN	FREE	Wobble Cone =		1.76,	Clok = -138.71 (deg)		
5.800	0.00	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
5.900	0.12	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.000	0.34	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.100	0.55	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.200	0.80	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.300	1.02	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.400	1.21	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.500	1.38	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.600	1.46	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.700	1.53	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.800	1.49	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
6.900	1.44	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.000	1.30	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.100	1.17	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.200	0.97	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.300	0.80	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.400	0.62	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.500	0.47	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.600	0.36	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.700	0.27	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.800	0.25	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
7.900	0.25	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.000	0.32	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.100	0.40	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.200	0.54	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.300	0.66	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.400	0.81	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.500	0.94	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.600	1.05	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.700	1.15	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.800	1.20	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
8.900	1.23	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
9.000	1.21	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
9.100	1.17	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
9.200	1.08	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
9.300	1.00	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
9.400	0.88	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
9.500	0.78	DESPIN	DESPIN	0.00	0.00	0.00	0.00	0.00	10.00
9.600	0.67	DESPIN	FREE	Wobble Cone =		2.48,	Clok = 162.32 (deg)		
9.700	0.46	DESPIN	FREE	Wobble Cone =		2.49,	Clok = 162.38 (deg)		
9.800	0.17	DESPIN	FREE	Wobble Cone =		2.49,	Clok = 162.44 (deg)		
9.900	-0.08	DESPIN	FREE	Wobble Cone =		2.50,	Clok = 162.51 (deg)		
10.000	-0.34	DESPIN	FREE	Wobble Cone =		2.50,	Clok = 162.57 (deg)		
10.100	-0.56	DESPIN	FREE	Wobble Cone =		2.51,	Clok = 162.64 (deg)		
10.200	-0.70	DESPIN	FREE	Wobble Cone =		2.51,	Clok = 162.70 (deg)		
10.300	-0.81	DESPIN	FREE	Wobble Cone =		2.52,	Clok = 162.76 (deg)		
10.400	-0.81	DESPIN	FREE	Wobble Cone =		2.52,	Clok = 162.83 (deg)		
10.500	-0.80	DESPIN	FREE	Wobble Cone =		2.52,	Clok = 162.89 (deg)		
10.600	-0.67	DESPIN	FREE	Wobble Cone =		2.53,	Clok = 162.95 (deg)		
10.700	-0.54	DESPIN	FREE	Wobble Cone =		2.53,	Clok = 163.02 (deg)		
10.800	-0.33	DESPIN	FREE	Wobble Cone =		2.54,	Clok = 163.08 (deg)		
10.900	-0.14	DESPIN	FREE	Wobble Cone =		2.54,	Clok = 163.14 (deg)		
11.000	0.08	DESPIN	FREE	Wobble Cone =		2.55,	Clok = 163.20 (deg)		
11.100	0.27	DESPIN	FREE	Wobble Cone =		2.55,	Clok = 163.27 (deg)		
11.200	0.43	DESPIN	FREE	Wobble Cone =		2.56,	Clok = 163.33 (deg)		
11.300	0.56	DESPIN	FREE	Wobble Cone =		2.56,	Clok = 163.39 (deg)		
11.400	0.62	DESPIN	FREE	Wobble Cone =		2.57,	Clok = 163.45 (deg)		
11.500	0.66	DESPIN	FREE	Wobble Cone =		2.57,	Clok = 163.52 (deg)		
11.600	0.60	DESPIN	FREE	Wobble Cone =		2.57,	Clok = 163.58 (deg)		

11.700	0.54	DESPIN	FREE	Wobble Cone =	2.58,	Clok =	163.64	(deg)
11.800	0.40	DESPIN	FREE	Wobble Cone =	2.58,	Clok =	163.70	(deg)
11.900	0.27	DESPIN	FREE	Wobble Cone =	2.59,	Clok =	163.76	(deg)
12.000	0.09	DESPIN	FREE	Wobble Cone =	2.59,	Clok =	163.83	(deg)
12.100	-0.06	DESPIN	FREE	Wobble Cone =	2.60,	Clok =	163.89	(deg)
12.200	-0.22	DESPIN	FREE	Wobble Cone =	2.60,	Clok =	163.95	(deg)
12.300	-0.35	DESPIN	FREE	Wobble Cone =	2.61,	Clok =	164.01	(deg)
12.400	-0.43	DESPIN	FREE	Wobble Cone =	2.61,	Clok =	164.07	(deg)
12.500	-0.50	DESPIN	FREE	Wobble Cone =	2.61,	Clok =	164.13	(deg)
12.600	-0.49	DESPIN	FREE	Wobble Cone =	2.62,	Clok =	164.20	(deg)
12.700	-0.48	DESPIN	FREE	Wobble Cone =	2.62,	Clok =	164.26	(deg)
12.800	-0.40	DESPIN	FREE	Wobble Cone =	2.63,	Clok =	164.32	(deg)
12.900	-0.32	DESPIN	FREE	Wobble Cone =	2.63,	Clok =	164.38	(deg)
13.000	-0.19	DESPIN	FREE	Wobble Cone =	2.64,	Clok =	164.44	(deg)
13.100	-0.07	DESPIN	FREE	Wobble Cone =	2.64,	Clok =	164.50	(deg)
13.200	0.06	DESPIN	FREE	Wobble Cone =	2.65,	Clok =	164.56	(deg)
13.300	0.17	DESPIN	FREE	Wobble Cone =	2.65,	Clok =	164.62	(deg)
13.400	0.27	DESPIN	FREE	Wobble Cone =	2.66,	Clok =	164.68	(deg)
13.500	0.35	DESPIN	FREE	Wobble Cone =	2.66,	Clok =	164.74	(deg)
13.600	0.38	DESPIN	FREE	Wobble Cone =	2.66,	Clok =	164.80	(deg)
13.700	0.40	DESPIN	FREE	Wobble Cone =	2.67,	Clok =	164.86	(deg)
13.800	0.36	DESPIN	FREE	Wobble Cone =	2.67,	Clok =	164.92	(deg)
13.900	0.32	DESPIN	FREE	Wobble Cone =	2.68,	Clok =	164.98	(deg)
14.000	0.23	DESPIN	FREE	Wobble Cone =	2.68,	Clok =	165.04	(deg)
14.100	0.15	DESPIN	FREE	Wobble Cone =	2.69,	Clok =	165.10	(deg)
14.200	0.05	DESPIN	FREE	Wobble Cone =	2.69,	Clok =	165.16	(deg)
14.300	-0.05	DESPIN	FREE	Wobble Cone =	2.70,	Clok =	165.22	(deg)
14.400	-0.14	DESPIN	FREE	Wobble Cone =	2.70,	Clok =	165.28	(deg)
14.500	-0.21	DESPIN	FREE	Wobble Cone =	2.70,	Clok =	165.34	(deg)
14.600	-0.26	DESPIN	FREE	Wobble Cone =	2.71,	Clok =	165.40	(deg)
14.700	-0.30	DESPIN	FREE	Wobble Cone =	2.71,	Clok =	165.46	(deg)
14.800	-0.30	DESPIN	FREE	Wobble Cone =	2.72,	Clok =	165.52	(deg)
14.900	-0.29	DESPIN	FREE	Wobble Cone =	2.72,	Clok =	165.58	(deg)
15.000	-0.24	DESPIN	FREE	Wobble Cone =	2.73,	Clok =	165.64	(deg)

Time = 15.000   DesMode = DESPIN   CurMode = FREE
PL Nomspin = 3.55   Wobble Cone = 2.73   Wobble Clok = 165.64

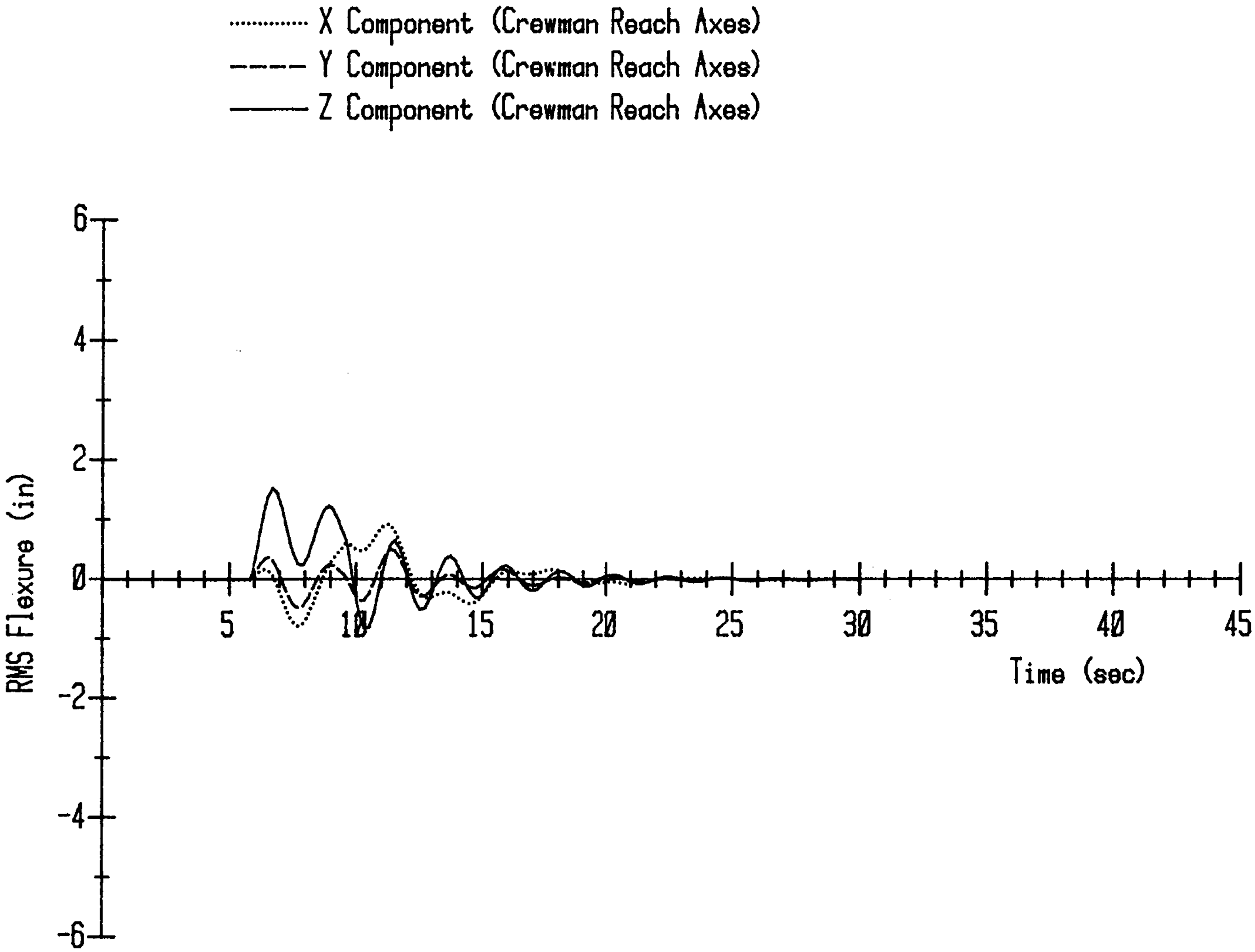
Cntrl@H: R	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
Cntrl@H:PB	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
PB Axes:PD	-0.10	2.63	29.68	PYR	3.560	0.099	0.177	Rate
PL Hndl: R	0.91	0.31	2.90	Pos	0.173	-0.015	0.257	Vel
PB Axes:OB	179.90	2.63	29.68	PYR	3.560	0.099	0.177	Rate
PL CM :OB	25.13	9.14	-19.34	Pos	-0.002	-0.004	-0.090	Vel
PL Rot K E	12.32							
PLAngmo: I	177.21	1.90	396.56	PYMag	-395.868	13.166	-19.299	Hxyz
PB Axes: I	179.84	2.62	29.64	PYR	3.553	0.092	0.180	Rate
PL CM : I	-0.50	7.15	-0.59	Pos	0.000	-0.002	-0.080	Vel
OB Axes: I	-0.06	0.00	0.04	PYR	0.006	-0.008	0.000	Rate
Orb CM : I	-25.65	-2.00	18.72	Pos	-0.000	0.000	0.006	Vel

Time	Stepsize
15.000	0.200

Time = 30.000 DesMode = DESPIN CurMode = FREE
PL Nomspin = 3.55 Wobble Cone = 3.34 Wobble Clok = 173.78

Cntrl@H: R	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
Cntrl@H:PB	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
PB Axes:PD	-1.86	5.16	83.20	PYR	3.557	0.064	0.224	Rate
PL Hndl: R	6.17	0.36	5.11	Pos	0.438	-0.007	-0.034	Vel
PB Axes:OB	178.14	5.16	83.20	PYR	3.557	0.064	0.224	Rate
PL CM :OB	25.10	9.07	-20.69	Pos	-0.003	-0.005	-0.090	Vel
PL Rot K E	12.32							
PLAngmo: I	177.21	1.90	396.56	PYMag	-395.868	13.166	-19.299	Hxyz
PB Axes: I	177.97	5.16	83.07	PYR	3.551	0.063	0.232	Rate
PL CM : I	-0.49	7.12	-1.79	Pos	0.000	-0.002	-0.080	Vel
OB Axes: I	-0.18	0.01	0.13	PYR	0.006	-0.008	0.000	Rate
Orb CM : I	-25.65	-2.00	18.81	Pos	-0.000	0.000	0.006	Vel

*** END OF SIMULATION ***
Syncom 10#/24" Despin Stroke Displaced Aft 6"
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:00:23 Mon 03 Nov 1986



Syncom 1Ø#/24" Despin Stroke Displaced Aft 6"

2.3. Syncom Attitude Correction Maneuver (Roll/Yaw/Pitch)

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:19:36 Mon 03 Nov 1986

"mvr" Position Parameters for Payload Flight Control

Nominal Orbiter STA of PL handling point...........(in)	800.00000
Nominal Orbiter BL of PL handling point...........(in)	24.00000
Nominal Orbiter WL of PL handling point...........(in)	600.00000
Payload STA of PL handle...........................(in)	-17.70378
Payload BL of PL handle............................(in)	-86.12500
Payload WL of PL handle............................(in)	0.00000
Crewman reach limit from RO in +/- Rx direction....(in)	12.00000
Crewman reach limit from RO in +/- Ry direction....(in)	12.00000
Crewman reach limit from RO in +/- Rz direction....(in)	12.00000
Rx position tolerance for PL handle................(in)	6.00000
Ry position tolerance for PL handle................(in)	6.00000
Rz position tolerance for PL handle................(in)	6.00000
Nominal vel for handle position correction.....(in/sec)	1.00000
Tolerance for corrective velocity...................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:20:08 Mon 03 Nov 1986

"mvr" Initial Conditions for the Simulation

Orbiter pitch wrt I axes..........................(deg)	0.00000
Orbiter yaw wrt I axes..........................(deg)	0.00000
Orbiter roll wrt I axes..........................(deg)	0.00000
Orbiter Bx component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter By component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter Bz component of ang vel wrt I axes....(deg/sec)	0.00000
Rx component of PL handle position.................(ft)	0.00000
Ry component of PL handle position.................(ft)	0.00000
Rz component of PL handle position.................(ft)	0.00000
Payload CM Xdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Ydot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Zdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload pitch wrt desired attitude................(deg)	10.00000
Payload yaw wrt desired attitude................(deg)	-10.00000
Payload roll wrt desired attitude................(deg)	10.00000
Payload nominal spin rate.....................(deg/sec)	0.00000
Payload wobble cone angle.........................(deg)	0.00000
Payload wobble clok angle.........................(deg)	0.00000

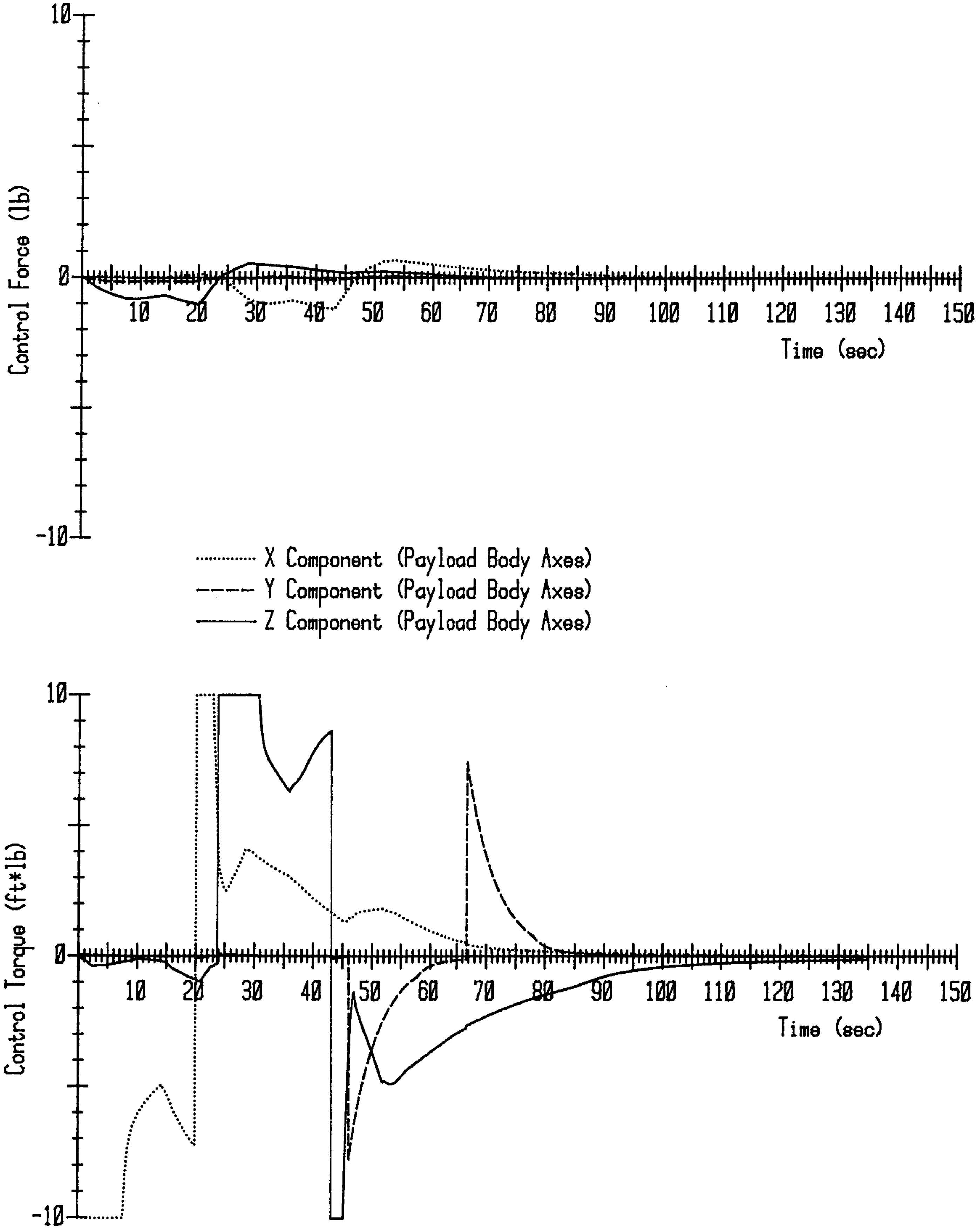
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:20:10 Mon 03 Nov 1986
Syncom Attitude Correction Maneuver (Roll/Yaw/Pitch)

```
Time =      0.000            DesMode =  PITCH            CurMode =  ROLL
PL Nomspin =    0.00         Wobble Cone =    0.00       Wobble Clok =     0.00

Cntrl@H: R        1.81     -9.69     -1.68 Torq     0.000     0.000     0.000 Forc
Cntrl@H:PB      -10.00      0.07     -0.05 Torq     0.000     0.000     0.000 Forc

PB Axes:PD       10.00    -10.00     10.00 PYR      0.000     0.000     0.000 Rate
PL Hndl: R        0.00      0.00      0.00 Pos      0.000     0.000     0.000 Vel

PB Axes:OB     -170.00    -10.00     10.00 PYR      0.000     0.000     0.000 Rate
PL CM  :OB       24.23      8.96    -19.68 Pos      0.000     0.000     0.000 Vel

PL Rot K E        0.00
PLAngmo: I        0.00      0.00      0.00 PYMag    0.000     0.000     0.000 Hxyz
PB Axes: I     -170.00    -10.00     10.00 PYR      0.000     0.000     0.000 Rate
PL CM  : I       -1.42      6.96     -1.01 Pos      0.000     0.000     0.000 Vel

OB Axes: I        0.00      0.00      0.00 PYR      0.000     0.000     0.000 Rate
Orb CM : I      -25.65     -2.00     18.68 Pos      0.000     0.000     0.000 Vel
```

Time	Stepsize
0.000	0.200

```
Time =  135.000          DesMode =  PITCH          CurMode =  HOLD
PL Nomspin =    0.00     Wobble Cone =  62.44      Wobble Clok =  -99.52

Cntrl@H: R       0.01     0.04     0.11 Torq    -0.002   0.015   -0.006 Forc
Cntrl@H:PB       0.04     0.00    -0.11 Torq     0.015  -0.002    0.006 Forc

PB Axes:PD      -0.78    -1.21     0.79 PYR      0.001  -0.004    0.001 Rate
PL Hndl: R      -0.09    -0.37     0.39 Pos      0.001  -0.002    0.002 Vel

PB Axes:OB     179.22    -1.21     0.79 PYR      0.001  -0.004    0.001 Rate
PL CM  :OB      25.87     9.08   -18.38 Pos      0.003   0.001    0.002 Vel

PL Rot K E       0.00
PLAngmo: I     163.18   -62.50     0.32 PYMag   -0.140  -0.281   -0.042 Hxyz
PB Axes: I     179.44    -1.25     1.13 PYR      0.001  -0.004    0.001 Rate
PL CM  : I       0.03     6.99     0.07 Pos      0.002   0.000    0.001 Vel

OB Axes: I       0.22     0.04    -0.34 PYR     -0.001   0.000    0.000 Rate
Orb CM : I     -25.76    -2.00    18.60 Pos     -0.000  -0.000   -0.000 Vel


*** END OF SIMULATION ***
Syncom Attitude Correction Maneuver (Roll/Yaw/Pitch)
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:20:10 Mon 03 Nov 1986
```

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 11:20:10 Mon 03 Nov 1986



Syncom Attitude Correction Maneuver (Roll/Yaw/Pitch)

C-3

2.4. Syncom Spinup Stroke Without Lateral Corrective Force

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:07:42 Mon 03 Nov 1986

"sncmd" Payload Inertia Data

Payload weight.......................................(lb)	15232.50000
CM STA (structural x coordinate of mass center)....(in)	-17.71284
CM BL (structural y coordinate of mass center)....(in)	0.15883
CM WL (structural z coordinate of mass center)....(in)	0.08977
Ixx about CM, structural coordinates.......(slug*ft*ft)	6331.54180
Iyy about CM, structural coordinates.......(slug*ft*ft)	4425.10827
Izz about CM, structural coordinates.......(slug*ft*ft)	5217.64945
Pxy about CM, structural coordinates.......(slug*ft*ft)	-38.70922
Pxz about CM, structural coordinates.......(slug*ft*ft)	-25.49521
Pyz about CM, structural coordinates.......(slug*ft*ft)	30.29916

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:08:15 Mon 03 Nov 1986

"fsu" Position Parameters for Payload Flight Control

Nominal Orbiter STA of PL handling point...........(in)	768.00000
Nominal Orbiter BL of PL handling point...........(in)	18.00000
Nominal Orbiter WL of PL handling point...........(in)	572.00000
Payload STA of PL handle..........................(in)	-17.71284
Payload BL of PL handle..........................(in)	-86.12500
Payload WL of PL handle..........................(in)	0.00000
Crewman reach limit from RO in +/- Rx direction....(in)	12.00000
Crewman reach limit from RO in +/- Ry direction....(in)	12.00000
Crewman reach limit from RO in +/- Rz direction....(in)	12.00000
Rx position tolerance for PL handle................(in)	6.00000
Ry position tolerance for PL handle................(in)	6.00000
Rz position tolerance for PL handle................(in)	6.00000
Nominal vel for handle position correction.....(in/sec)	0.00000
Tolerance for corrective velocity..................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:08:42 Mon 03 Nov 1986

"dploy" Attitude Parameters for Payload Flight Control

Crewman pitch wrt Orbiter body axes...............(deg)	0.00000
Crewman yaw wrt Orbiter body axes...............(deg)	90.00000
Crewman roll wrt Orbiter body axes...............(deg)	0.00000
Desired Payload pitch wrt Crewman body axes.......(deg)	0.00000
Desired Payload yaw wrt Crewman body axes.......(deg)	-90.00000
Desired Payload roll wrt Crewman body axes.......(deg)	0.00000
Positive PL pitch limit wrt desired attitude......(deg)	15.00000
Positive PL yaw limit wrt desired attitude......(deg)	15.00000
Positive PL roll limit wrt desired attitude......(deg)	10.00000
Negative PL pitch limit wrt desired attitude......(deg)	-15.00000
Negative PL yaw limit wrt desired attitude......(deg)	-15.00000
Negative PL roll limit wrt desired attitude......(deg)	-20.00000
PL pitch tolerance wrt desired attitude...........(deg)	2.00000
PL yaw tolerance wrt desired attitude...........(deg)	2.00000
PL roll tolerance wrt desired attitude...........(deg)	2.00000
Nominal maneuver rate about PL Bx axis........(deg/sec)	0.00000
Nominal maneuver rate about PL By axis........(deg/sec)	0.00000
Nominal maneuver rate about PL Bz axis........(deg/sec)	0.00000
Maneuver rate tolerance.............................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:09:04 Mon 03 Nov 1986

"lim15" Force & Torque Parameters for Payload Flight Control

Spinup axial (PL Bx) force....................(lb)	0.00000
Spinup normal (PL By) force....................(lb)	0.00000
Spinup tangential (PL Bz) force....................(lb)	-15.00000
Despin axial (PL Bx) force....................(lb)	0.00000
Despin normal (PL By) force....................(lb)	0.00000
Despin tangential (PL Bz) force....................(lb)	15.00000
Axial (PL Bx) force limit for capture.........(lb)	15.00000
Normal (PL By) force limit for capture.........(lb)	15.00000
Tangential (PL Bz) force limit for capture.........(lb)	15.00000
Roll (PL Bx) torque limit for capture..........(ft*lb)	15.00000
Pitch (PL By) torque limit for capture..........(ft*lb)	15.00000
Yaw (PL Bz) torque limit for capture..........(ft*lb)	15.00000
Time constant for computing desired accelerations (sec)	5.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:09:32 Mon 03 Nov 1986

"fsu" Initial Conditions for the Simulation

Orbiter pitch wrt I axes..........................(deg)	0.00000
Orbiter yaw wrt I axes..........................(deg)	0.00000
Orbiter roll wrt I axes..........................(deg)	0.00000
Orbiter Bx component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter By component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter Bz component of ang vel wrt I axes....(deg/sec)	0.00000
Rx component of PL handle position.................(ft)	0.00000
Ry component of PL handle position.................(ft)	0.00000
Rz component of PL handle position.................(ft)	0.99167
Payload CM Xdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Ydot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Zdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload pitch wrt desired attitude................(deg)	0.00000
Payload yaw wrt desired attitude................(deg)	0.00000
Payload roll wrt desired attitude................(deg)	-9.00000
Payload nominal spin rate.....................(deg/sec)	0.00000
Payload wobble cone angle........................(deg)	0.00000
Payload wobble clok angle........................(deg)	0.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:09:35 Mon 03 Nov 1986
Syncom 15#/24" Spinup Stroke Without Lateral Corrective Force

Time = 0.000		DesMode = SPINUP		CurMode = SPINUP		
PL Nomspin = 0.00		Wobble Cone = 0.00		Wobble Clok = 0.00		
Cntrl@H: R	0.00	0.00	0.00 Torq	-2.347	-0.000	-14.815 Forc
Cntrl@H:PB	0.00	0.00	0.00 Torq	0.000	0.000	-15.000 Forc
PB Axes:PD	0.00	0.00	-9.00 PYR	0.000	0.000	0.000 Rate
PL Hndl: R	0.00	0.00	0.99 Pos	0.000	0.000	0.000 Vel
PB Axes:OB	0.00	0.00	-9.00 PYR	0.000	0.000	0.000 Rate
PL CM :OB	28.32	8.60	-16.48 Pos	0.000	0.000	0.000 Vel
PL Rot K E	0.00					
PLAngmo: I	0.00	0.00	0.00 PYMag	0.000	0.000	0.000 Hxyz
PB Axes: I	0.00	0.00	-9.00 PYR	0.000	0.000	0.000 Rate
PL CM : I	0.00	7.10	-0.14 Pos	0.000	0.000	0.000 Vel
OB Axes: I	0.00	0.00	0.00 PYR	0.000	0.000	0.000 Rate
Orb CM : I	-28.32	-1.50	16.34 Pos	0.000	0.000	0.000 Vel

Time	Stepsize
0.000	0.100

Time (sec)	zFlex (in)	DesMode	CurMode	T_PBx (ft*lb)	T_PBy (ft*lb)	T_PBz (ft*lb)	F_PBx (lb)	F_PBy (lb)	F_PBz (lb)
0.000	0.00	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.050	0.00	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.100	0.04	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.150	0.09	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.200	0.17	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.250	0.25	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.300	0.37	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.350	0.48	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.400	0.62	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.450	0.75	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.500	0.89	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.550	1.04	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.600	1.18	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.650	1.32	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.700	1.45	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.750	1.58	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.800	1.69	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.850	1.80	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.900	1.88	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.950	1.96	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.000	2.01	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.050	2.06	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.100	2.07	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.150	2.08	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.200	2.06	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.250	2.04	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.300	1.99	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.350	1.94	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.400	1.86	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.450	1.79	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.500	1.69	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.550	1.59	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.600	1.49	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.650	1.38	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.700	1.27	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.750	1.17	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.800	1.06	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.850	0.96	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.900	0.87	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
1.950	0.79	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.000	0.72	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.050	0.65	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.100	0.60	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.150	0.56	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.200	0.54	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.250	0.52	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.300	0.53	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.350	0.53	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.400	0.56	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.450	0.59	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.500	0.65	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.550	0.70	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.600	0.77	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.650	0.83	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.700	0.91	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.750	0.99	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.800	1.07	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00

2.850	1.15	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.900	1.23	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
2.950	1.31	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.000	1.38	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.050	1.45	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.100	1.51	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.150	1.57	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.200	1.61	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.250	1.65	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.300	1.67	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.350	1.70	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.400	1.70	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.450	1.70	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.500	1.68	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.550	1.67	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.600	1.64	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.650	1.60	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.700	1.56	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.750	1.51	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.800	1.45	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.850	1.40	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.900	1.34	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
3.950	1.28	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.000	1.22	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.050	1.16	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.100	1.10	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.150	1.05	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.200	1.00	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.250	0.95	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.300	0.92	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.350	0.88	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.400	0.86	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.450	0.84	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.500	0.83	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.550	0.82	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.600	0.83	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.650	0.84	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.700	0.86	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.750	0.88	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.800	0.91	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.850	0.94	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.900	0.98	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
4.950	1.02	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
5.000	1.06	SPINUP	FREE	Wobble Cone =		0.08,	Clok =	151.90	(deg)
5.050	1.06	SPINUP	FREE	Wobble Cone =		0.09,	Clok =	151.93	(deg)
5.100	1.02	SPINUP	FREE	Wobble Cone =		0.09,	Clok =	151.96	(deg)
5.150	0.98	SPINUP	FREE	Wobble Cone =		0.09,	Clok =	151.98	(deg)
5.200	0.90	SPINUP	FREE	Wobble Cone =		0.09,	Clok =	152.01	(deg)
5.250	0.83	SPINUP	FREE	Wobble Cone =		0.10,	Clok =	152.03	(deg)
5.300	0.73	SPINUP	FREE	Wobble Cone =		0.10,	Clok =	152.06	(deg)
5.350	0.62	SPINUP	FREE	Wobble Cone =		0.10,	Clok =	152.09	(deg)
5.400	0.50	SPINUP	FREE	Wobble Cone =		0.10,	Clok =	152.11	(deg)
5.450	0.38	SPINUP	FREE	Wobble Cone =		0.11,	Clok =	152.14	(deg)
5.500	0.25	SPINUP	FREE	Wobble Cone =		0.11,	Clok =	152.17	(deg)
5.550	0.13	SPINUP	FREE	Wobble Cone =		0.11,	Clok =	152.19	(deg)
5.600	-0.00	SPINUP	FREE	Wobble Cone =		0.11,	Clok =	152.22	(deg)
5.650	-0.13	SPINUP	FREE	Wobble Cone =		0.12,	Clok =	152.25	(deg)
5.700	-0.25	SPINUP	FREE	Wobble Cone =		0.12,	Clok =	152.28	(deg)
5.750	-0.36	SPINUP	FREE	Wobble Cone =		0.12,	Clok =	152.30	(deg)
5.800	-0.46	SPINUP	FREE	Wobble Cone =		0.12,	Clok =	152.33	(deg)

5.850	-0.56	SPINUP	FREE	Wobble Cone =	0.13,	Clok =	152.36 (deg)
5.900	-0.63	SPINUP	FREE	Wobble Cone =	0.13,	Clok =	152.39 (deg)
5.950	-0.70	SPINUP	FREE	Wobble Cone =	0.13,	Clok =	152.42 (deg)
6.000	-0.74	SPINUP	FREE	Wobble Cone =	0.13,	Clok =	152.44 (deg)
6.050	-0.78	SPINUP	FREE	Wobble Cone =	0.14,	Clok =	152.47 (deg)
6.100	-0.79	SPINUP	FREE	Wobble Cone =	0.14,	Clok =	152.50 (deg)
6.150	-0.80	SPINUP	FREE	Wobble Cone =	0.14,	Clok =	152.53 (deg)
6.200	-0.78	SPINUP	FREE	Wobble Cone =	0.14,	Clok =	152.56 (deg)
6.250	-0.76	SPINUP	FREE	Wobble Cone =	0.15,	Clok =	152.59 (deg)
6.300	-0.72	SPINUP	FREE	Wobble Cone =	0.15,	Clok =	152.61 (deg)
6.350	-0.67	SPINUP	FREE	Wobble Cone =	0.15,	Clok =	152.64 (deg)
6.400	-0.60	SPINUP	FREE	Wobble Cone =	0.15,	Clok =	152.67 (deg)
6.450	-0.53	SPINUP	FREE	Wobble Cone =	0.16,	Clok =	152.70 (deg)
6.500	-0.44	SPINUP	FREE	Wobble Cone =	0.16,	Clok =	152.73 (deg)
6.550	-0.36	SPINUP	FREE	Wobble Cone =	0.16,	Clok =	152.76 (deg)
6.600	-0.26	SPINUP	FREE	Wobble Cone =	0.16,	Clok =	152.79 (deg)
6.650	-0.17	SPINUP	FREE	Wobble Cone =	0.17,	Clok =	152.81 (deg)
6.700	-0.07	SPINUP	FREE	Wobble Cone =	0.17,	Clok =	152.84 (deg)
6.750	0.03	SPINUP	FREE	Wobble Cone =	0.17,	Clok =	152.87 (deg)
6.800	0.12	SPINUP	FREE	Wobble Cone =	0.17,	Clok =	152.90 (deg)
6.850	0.21	SPINUP	FREE	Wobble Cone =	0.17,	Clok =	152.93 (deg)
6.900	0.29	SPINUP	FREE	Wobble Cone =	0.18,	Clok =	152.96 (deg)
6.950	0.37	SPINUP	FREE	Wobble Cone =	0.18,	Clok =	152.99 (deg)
7.000	0.43	SPINUP	FREE	Wobble Cone =	0.18,	Clok =	153.02 (deg)

Time = 7.000 DesMode = SPINUP CurMode = FREE
PL Nomspin = 4.78 Wobble Cone = 0.18 Wobble Clok = 153.02

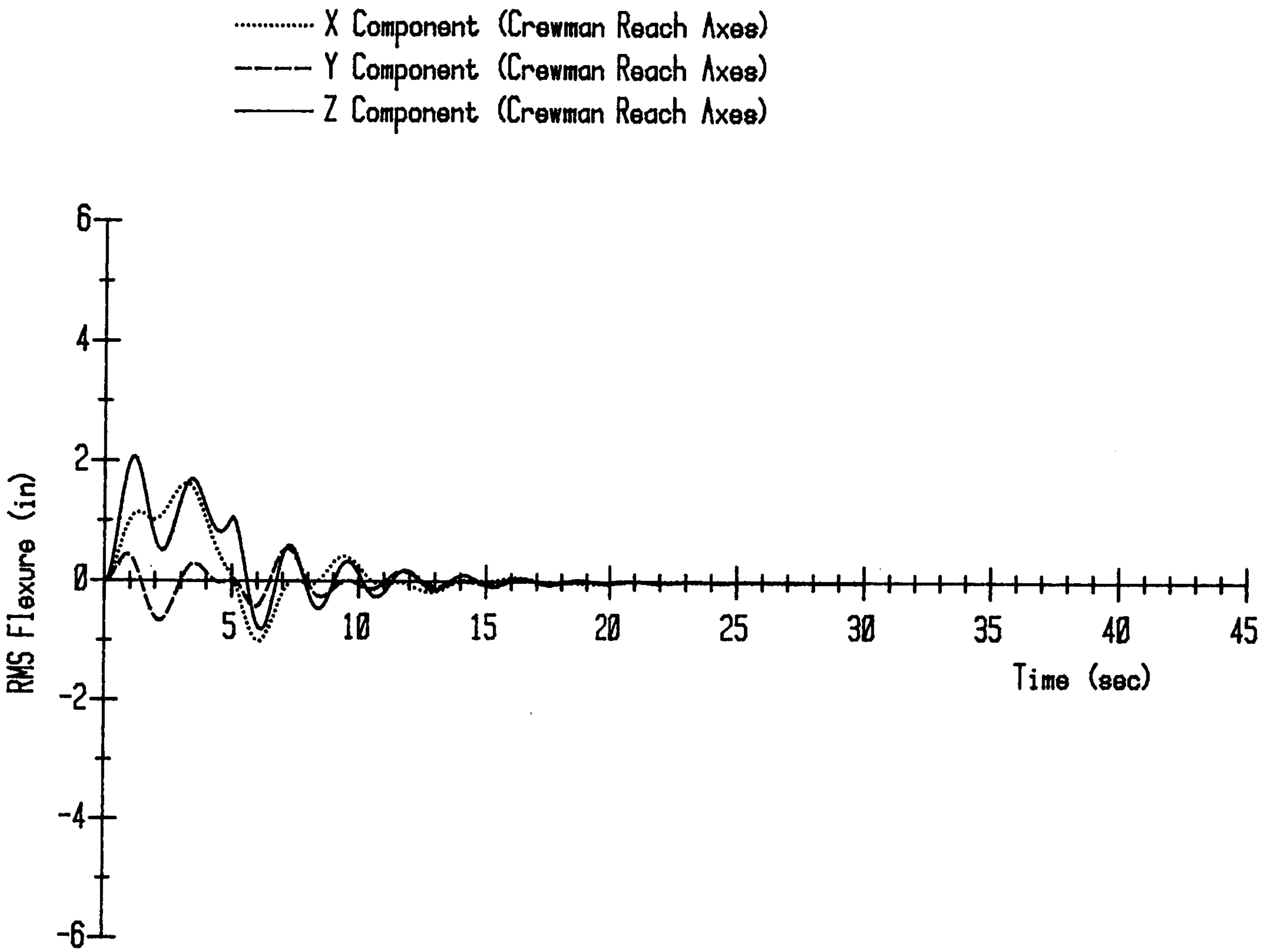
Cntrl@H: R	0.00	0.00	0.00 Torq	0.000	0.000	0.000	Forc
Cntrl@H:PB	0.00	0.00	0.00 Torq	0.000	0.000	0.000	Forc
PB Axes:PD	0.29	-0.08	12.70 PYR	4.769	0.068	-0.013	Rate
PL Hndl: R	-0.01	-0.00	-2.55 Pos	0.042	-0.030	-0.860	Vel
PB Axes:OB	0.29	-0.08	12.70 PYR	4.769	0.068	-0.013	Rate
PL CM :OB	28.29	8.49	-17.28 Pos	-0.005	-0.020	-0.176	Vel
PL Rot K E	22.06						
PLAngmo: I	0.03	-0.02	528.49 PYMag	528.488	-0.192	-0.308	Hxyz
PB Axes: I	0.21	-0.07	12.77 PYR	4.783	0.052	-0.007	Rate
PL CM : I	-0.00	7.03	-0.84 Pos	-0.000	-0.014	-0.154	Vel
OB Axes: I	-0.08	0.01	0.07 PYR	0.014	-0.017	0.002	Rate
Orb CM : I	-28.32	-1.49	16.39 Pos	0.000	0.001	0.011	Vel

Time	Stepsize
7.000	0.100

Time = 30.000 DesMode = SPINUP CurMode = FREE
PL Nomspin = 4.78 Wobble Cone = 1.36 Wobble Clok = 165.89

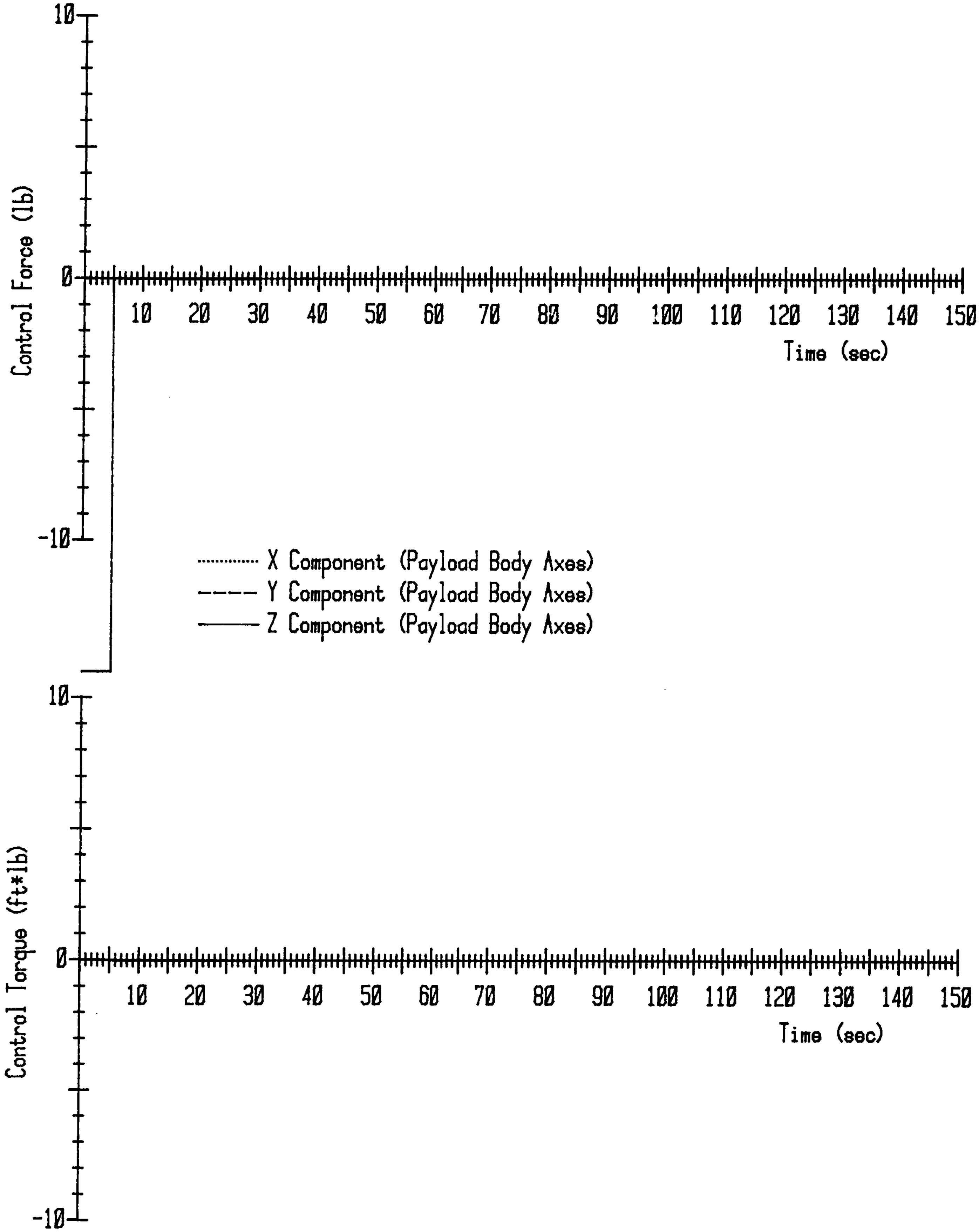
Cntrl@H: R	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
Cntrl@H:PB	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
PB Axes:PD	0.08	1.22	122.39	PYR	4.768	0.070	0.096	Rate
PL Hndl: R	10.36	0.26	-11.07	Pos	0.484	0.006	0.144	Vel
PB Axes:OB	0.08	1.22	122.39	PYR	4.768	0.070	0.096	Rate
PL CM :OB	28.15	8.01	-21.33	Pos	-0.008	-0.022	-0.176	Vel
PL Rot K E	22.06							
PLAngmo: I	0.03	-0.02	528.49	PYMag	528.488	-0.192	-0.308	Hxyz
PB Axes: I	-0.40	1.27	122.77	PYR	4.781	0.081	0.110	Rate
PL CM : I	-0.00	6.71	-4.39	Pos	-0.000	-0.014	-0.154	Vel
OB Axes: I	-0.48	0.05	0.38	PYR	0.014	-0.017	0.002	Rate
Orb CM : I	-28.32	-1.47	16.65	Pos	0.000	0.001	0.011	Vel

*** END OF SIMULATION ***
Syncom 15#/24" Spinup Stroke Without Lateral Corrective Force
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:09:35 Mon 03 Nov 1986



Syncom 15#/24" Spinup Stroke Without Lateral Corrective Force

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:09:35 Mon 03 Nov 1986



Syncom 15#/24" Spinup Stroke Without Lateral Corrective Force

2.5. Syncom Spinup Stroke With Lateral Corrective Force

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:36:32 Mon 03 Nov 1986

"fsulc" Position Parameters for Payload Flight Control

Nominal Orbiter STA of PL handling point...........(in)	768.00000
Nominal Orbiter BL of PL handling point...........(in)	18.00000
Nominal Orbiter WL of PL handling point...........(in)	572.00000
Payload STA of PL handle...........................(in)	-17.71284
Payload BL of PL handle...........................(in)	-86.12500
Payload WL of PL handle...........................(in)	0.00000
Crewman reach limit from RO in +/- Rx direction....(in)	12.00000
Crewman reach limit from RO in +/- Ry direction....(in)	12.00000
Crewman reach limit from RO in +/- Rz direction....(in)	12.00000
Rx position tolerance for PL handle................(in)	6.00000
Ry position tolerance for PL handle................(in)	1.00000
Rz position tolerance for PL handle................(in)	6.00000
Nominal vel for handle position correction.....(in/sec)	1.00000
Tolerance for corrective velocity..................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:36:40 Mon 03 Nov 1986
Syncom 15#/24" Spinup Stroke With Lateral Corrective Force

Time = 0.000		DesMode = SPINUP	CurMode = SPINUP
PL Nomspin = 0.00		Wobble Cone = 0.00	Wobble Clok = 0.00

Cntrl@H: R	0.00	0.00	0.00 Torq	-2.347	-0.000	-14.815	Forc
Cntrl@H:PB	0.00	0.00	0.00 Torq	0.000	0.000	-15.000	Forc
PB Axes:PD	0.00	0.00	-9.00 PYR	0.000	0.000	0.000	Rate
PL Hndl: R	0.00	0.00	0.99 Pos	0.000	0.000	0.000	Vel
PB Axes:OB	0.00	0.00	-9.00 PYR	0.000	0.000	0.000	Rate
PL CM :OB	28.32	8.60	-16.48 Pos	0.000	0.000	0.000	Vel
PL Rot K E	0.00						
PLAngmo: I	0.00	0.00	0.00 PYMag	0.000	0.000	0.000	Hxyz
PB Axes: I	0.00	0.00	-9.00 PYR	0.000	0.000	0.000	Rate
PL CM : I	0.00	7.10	-0.14 Pos	0.000	0.000	0.000	Vel
OB Axes: I	0.00	0.00	0.00 PYR	0.000	0.000	0.000	Rate
Orb CM : I	-28.32	-1.50	16.34 Pos	0.000	0.000	0.000	Vel

Time	Stepsize
0.000	0.100

Time (sec)	zFlex (in)	DesMode	CurMode	T_PBx (ft*lb)	T_PBy (ft*lb)	T_PBz (ft*lb)	F_PBx (lb)	F_PBy (lb)	F_PBz (lb)
0.000	0.00	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.050	0.00	SPINUP	SPINUP	0.00	0.00	0.00	0.00	0.00	-15.00
0.100	0.04	SPINUP	SPINUP	0.00	0.00	0.00	-0.25	0.01	-15.00
0.150	0.09	SPINUP	SPINUP	0.00	0.00	0.00	-0.45	0.02	-15.00
0.200	0.17	SPINUP	SPINUP	0.00	0.00	0.00	-0.63	0.02	-15.00
0.250	0.25	SPINUP	SPINUP	0.00	0.00	0.00	-0.76	0.03	-15.00
0.300	0.37	SPINUP	SPINUP	0.00	0.00	0.00	-0.87	0.03	-15.00
0.350	0.48	SPINUP	SPINUP	0.00	0.00	0.00	-0.92	0.03	-15.00
0.400	0.62	SPINUP	SPINUP	0.00	0.00	0.00	-0.96	0.03	-15.00
0.450	0.75	SPINUP	SPINUP	0.00	0.00	0.00	-0.93	0.03	-15.00
0.500	0.89	SPINUP	SPINUP	0.00	0.00	0.00	-0.90	0.03	-15.00
0.550	1.04	SPINUP	SPINUP	0.00	0.00	0.00	-0.81	0.03	-15.00
0.600	1.18	SPINUP	SPINUP	0.00	0.00	0.00	-0.72	0.02	-15.00
0.650	1.32	SPINUP	SPINUP	0.00	0.00	0.00	-0.58	0.02	-15.00
0.700	1.45	SPINUP	SPINUP	0.00	0.00	0.00	-0.44	0.01	-15.00
0.750	1.58	SPINUP	SPINUP	0.00	0.00	0.00	-0.26	0.01	-15.00
0.800	1.69	SPINUP	SPINUP	0.00	0.00	0.00	-0.04	0.00	-15.00
0.850	1.80	SPINUP	SPINUP	0.00	0.00	0.00	0.07	-0.00	-15.00
0.900	1.88	SPINUP	SPINUP	0.00	0.00	0.00	0.30	-0.01	-15.00
0.950	1.96	SPINUP	SPINUP	0.00	0.00	0.00	0.49	-0.02	-15.00
1.000	2.01	SPINUP	SPINUP	0.00	0.00	0.00	0.68	-0.02	-15.00
1.050	2.06	SPINUP	SPINUP	0.00	0.00	0.00	0.85	-0.03	-15.00
1.100	2.07	SPINUP	SPINUP	0.00	0.00	0.00	1.02	-0.03	-15.00
1.150	2.08	SPINUP	SPINUP	0.00	0.00	0.00	1.16	-0.04	-15.00
1.200	2.06	SPINUP	SPINUP	0.00	0.00	0.00	1.30	-0.04	-15.00
1.250	2.04	SPINUP	SPINUP	0.00	0.00	0.00	1.39	-0.04	-15.00
1.300	1.99	SPINUP	SPINUP	0.00	0.00	0.00	1.49	-0.05	-14.99
1.350	1.94	SPINUP	SPINUP	0.00	0.00	0.00	1.53	-0.05	-14.99
1.400	1.86	SPINUP	SPINUP	0.00	0.00	0.00	1.57	-0.05	-14.99
1.450	1.78	SPINUP	SPINUP	0.00	0.00	0.00	1.56	-0.05	-14.99
1.500	1.69	SPINUP	SPINUP	0.00	0.00	0.00	1.54	-0.05	-14.99
1.550	1.59	SPINUP	SPINUP	0.00	0.00	0.00	1.48	-0.05	-14.99
1.600	1.49	SPINUP	SPINUP	0.00	0.00	0.00	1.41	-0.04	-14.99
1.650	1.38	SPINUP	SPINUP	0.00	0.00	0.00	1.30	-0.04	-15.00
1.700	1.27	SPINUP	SPINUP	0.00	0.00	0.00	1.19	-0.04	-15.00
1.750	1.17	SPINUP	SPINUP	0.00	0.00	0.00	1.04	-0.03	-15.00
1.800	1.06	SPINUP	SPINUP	0.00	0.00	0.00	0.89	-0.03	-15.00
1.850	0.96	SPINUP	SPINUP	0.00	0.00	0.00	0.71	-0.02	-15.00
1.900	0.87	SPINUP	SPINUP	0.00	0.00	0.00	0.54	-0.02	-15.00
1.950	0.79	SPINUP	SPINUP	0.00	0.00	0.00	0.35	-0.01	-15.00
2.000	0.71	SPINUP	SPINUP	0.00	0.00	0.00	0.16	-0.01	-15.00
2.050	0.65	SPINUP	SPINUP	0.00	0.00	0.00	-0.01	0.00	-15.00
2.100	0.60	SPINUP	SPINUP	0.00	0.00	0.00	-0.22	0.01	-15.00
2.150	0.56	SPINUP	SPINUP	0.00	0.00	0.00	-0.39	0.01	-15.00
2.200	0.54	SPINUP	SPINUP	0.00	0.00	0.00	-0.56	0.02	-15.00
2.250	0.52	SPINUP	SPINUP	0.00	0.00	0.00	-0.71	0.02	-15.00
2.300	0.53	SPINUP	SPINUP	0.00	0.00	0.00	-0.85	0.03	-15.00
2.350	0.53	SPINUP	SPINUP	0.00	0.00	0.00	-0.97	0.03	-15.00
2.400	0.56	SPINUP	SPINUP	0.00	0.00	0.00	-1.08	0.04	-15.00
2.450	0.59	SPINUP	SPINUP	0.00	0.00	0.00	-1.15	0.04	-15.00
2.500	0.65	SPINUP	SPINUP	0.00	0.00	0.00	-1.21	0.04	-15.00
2.550	0.70	SPINUP	SPINUP	0.00	0.00	0.00	-1.24	0.04	-15.00
2.600	0.77	SPINUP	SPINUP	0.00	0.00	0.00	-1.26	0.04	-15.01
2.650	0.83	SPINUP	SPINUP	0.00	0.00	0.00	-1.24	0.04	-15.00
2.700	0.91	SPINUP	SPINUP	0.00	0.00	0.00	-1.22	0.04	-15.00
2.750	0.99	SPINUP	SPINUP	0.00	0.00	0.00	-1.16	0.04	-15.00
2.800	1.07	SPINUP	SPINUP	0.00	0.00	0.00	-1.10	0.04	-15.00

2.850	1.15	SPINUP	SPINUP	0.00	0.00	0.00	-1.00	0.03	-15.00
2.900	1.23	SPINUP	SPINUP	0.00	0.00	0.00	-0.91	0.03	-15.00
2.950	1.31	SPINUP	SPINUP	0.00	0.00	0.00	-0.78	0.03	-15.00
3.000	1.38	SPINUP	SPINUP	0.00	0.00	0.00	-0.66	0.02	-15.00
3.050	1.45	SPINUP	SPINUP	0.00	0.00	0.00	-0.52	0.02	-15.00
3.100	1.51	SPINUP	SPINUP	0.00	0.00	0.00	-0.38	0.01	-15.00
3.150	1.57	SPINUP	SPINUP	0.00	0.00	0.00	-0.24	0.01	-15.00
3.200	1.61	SPINUP	SPINUP	0.00	0.00	0.00	-0.06	0.00	-15.00
3.250	1.65	SPINUP	SPINUP	0.00	0.00	0.00	0.01	-0.00	-15.00
3.300	1.68	SPINUP	SPINUP	0.00	0.00	0.00	0.18	-0.01	-15.00
3.350	1.70	SPINUP	SPINUP	0.00	0.00	0.00	0.30	-0.01	-15.00
3.400	1.70	SPINUP	SPINUP	0.00	0.00	0.00	0.42	-0.01	-15.00
3.450	1.70	SPINUP	SPINUP	0.00	0.00	0.00	0.53	-0.02	-15.00
3.500	1.69	SPINUP	SPINUP	0.00	0.00	0.00	0.63	-0.02	-15.00
3.550	1.67	SPINUP	SPINUP	0.00	0.00	0.00	0.70	-0.02	-15.00
3.600	1.64	SPINUP	SPINUP	0.00	0.00	0.00	0.77	-0.02	-15.00
3.650	1.60	SPINUP	SPINUP	0.00	0.00	0.00	0.81	-0.03	-15.00
3.700	1.56	SPINUP	SPINUP	0.00	0.00	0.00	0.85	-0.03	-15.00
3.750	1.51	SPINUP	SPINUP	0.00	0.00	0.00	0.86	-0.03	-15.00
3.800	1.45	SPINUP	SPINUP	0.00	0.00	0.00	0.87	-0.03	-15.00
3.850	1.40	SPINUP	SPINUP	0.00	0.00	0.00	0.84	-0.03	-15.00
3.900	1.34	SPINUP	SPINUP	0.00	0.00	0.00	0.82	-0.03	-15.00
3.950	1.28	SPINUP	SPINUP	0.00	0.00	0.00	0.77	-0.02	-15.00
4.000	1.22	SPINUP	SPINUP	0.00	0.00	0.00	0.72	-0.02	-15.00
4.050	1.16	SPINUP	SPINUP	0.00	0.00	0.00	0.64	-0.02	-15.00
4.100	1.10	SPINUP	SPINUP	0.00	0.00	0.00	0.57	-0.02	-15.00
4.150	1.04	SPINUP	SPINUP	0.00	0.00	0.00	0.48	-0.02	-15.00
4.200	1.00	SPINUP	SPINUP	0.00	0.00	0.00	0.39	-0.01	-15.00
4.250	0.95	SPINUP	SPINUP	0.00	0.00	0.00	0.29	-0.01	-15.00
4.300	0.91	SPINUP	SPINUP	0.00	0.00	0.00	0.20	-0.01	-15.00
4.350	0.88	SPINUP	SPINUP	0.00	0.00	0.00	0.06	-0.00	-15.00
4.400	0.86	SPINUP	SPINUP	0.00	0.00	0.00	0.00	-0.00	-15.00
4.450	0.83	SPINUP	SPINUP	0.00	0.00	0.00	-0.04	0.00	-15.00
4.500	0.83	SPINUP	SPINUP	0.00	0.00	0.00	-0.17	0.01	-15.00
4.550	0.82	SPINUP	SPINUP	0.00	0.00	0.00	-0.25	0.01	-15.00
4.600	0.83	SPINUP	SPINUP	0.00	0.00	0.00	-0.32	0.01	-15.00
4.650	0.83	SPINUP	SPINUP	0.00	0.00	0.00	-0.37	0.01	-15.00
4.700	0.86	SPINUP	SPINUP	0.00	0.00	0.00	-0.43	0.01	-15.00
4.750	0.88	SPINUP	SPINUP	0.00	0.00	0.00	-0.46	0.01	-15.00
4.800	0.91	SPINUP	SPINUP	0.00	0.00	0.00	-0.49	0.02	-15.00
4.850	0.94	SPINUP	SPINUP	0.00	0.00	0.00	-0.49	0.02	-15.00
4.900	0.98	SPINUP	SPINUP	0.00	0.00	0.00	-0.50	0.02	-15.00
4.950	1.02	SPINUP	SPINUP	0.00	0.00	0.00	-0.48	0.02	-15.00
5.000	1.06	SPINUP	FREE	Wobble	Cone =	0.25,	Clok =	172.51	(deg)
5.050	1.06	SPINUP	FREE	Wobble	Cone =	0.26,	Clok =	172.36	(deg)
5.100	1.02	SPINUP	FREE	Wobble	Cone =	0.26,	Clok =	172.21	(deg)
5.150	0.98	SPINUP	FREE	Wobble	Cone =	0.26,	Clok =	172.07	(deg)
5.200	0.90	SPINUP	FREE	Wobble	Cone =	0.26,	Clok =	171.93	(deg)
5.250	0.83	SPINUP	FREE	Wobble	Cone =	0.26,	Clok =	171.79	(deg)
5.300	0.73	SPINUP	FREE	Wobble	Cone =	0.27,	Clok =	171.65	(deg)
5.350	0.63	SPINUP	FREE	Wobble	Cone =	0.27,	Clok =	171.52	(deg)
5.400	0.50	SPINUP	FREE	Wobble	Cone =	0.27,	Clok =	171.39	(deg)
5.450	0.38	SPINUP	FREE	Wobble	Cone =	0.27,	Clok =	171.26	(deg)
5.500	0.25	SPINUP	FREE	Wobble	Cone =	0.28,	Clok =	171.14	(deg)
5.550	0.13	SPINUP	FREE	Wobble	Cone =	0.28,	Clok =	171.02	(deg)
5.600	-0.00	SPINUP	FREE	Wobble	Cone =	0.28,	Clok =	170.90	(deg)
5.650	-0.13	SPINUP	FREE	Wobble	Cone =	0.28,	Clok =	170.78	(deg)
5.700	-0.25	SPINUP	FREE	Wobble	Cone =	0.28,	Clok =	170.66	(deg)
5.750	-0.36	SPINUP	FREE	Wobble	Cone =	0.29,	Clok =	170.55	(deg)
5.800	-0.46	SPINUP	FREE	Wobble	Cone =	0.29,	Clok =	170.44	(deg)

5.850	-0.56	SPINUP	FREE	Wobble Cone =	0.29,	Clok =	170.33	(deg)
5.900	-0.63	SPINUP	FREE	Wobble Cone =	0.29,	Clok =	170.22	(deg)
5.950	-0.70	SPINUP	FREE	Wobble Cone =	0.30,	Clok =	170.12	(deg)
6.000	-0.74	SPINUP	FREE	Wobble Cone =	0.30,	Clok =	170.02	(deg)
6.050	-0.78	SPINUP	FREE	Wobble Cone =	0.30,	Clok =	169.92	(deg)
6.100	-0.79	SPINUP	FREE	Wobble Cone =	0.30,	Clok =	169.82	(deg)
6.150	-0.80	SPINUP	FREE	Wobble Cone =	0.31,	Clok =	169.72	(deg)
6.200	-0.79	SPINUP	FREE	Wobble Cone =	0.31,	Clok =	169.63	(deg)
6.250	-0.77	SPINUP	FREE	Wobble Cone =	0.31,	Clok =	169.54	(deg)
6.300	-0.72	SPINUP	FREE	Wobble Cone =	0.31,	Clok =	169.44	(deg)
6.350	-0.67	SPINUP	FREE	Wobble Cone =	0.31,	Clok =	169.35	(deg)
6.400	-0.60	SPINUP	FREE	Wobble Cone =	0.32,	Clok =	169.27	(deg)
6.450	-0.53	SPINUP	FREE	Wobble Cone =	0.32,	Clok =	169.18	(deg)
6.500	-0.44	SPINUP	FREE	Wobble Cone =	0.32,	Clok =	169.10	(deg)
6.550	-0.36	SPINUP	FREE	Wobble Cone =	0.32,	Clok =	169.01	(deg)
6.600	-0.26	SPINUP	FREE	Wobble Cone =	0.33,	Clok =	168.93	(deg)
6.650	-0.17	SPINUP	FREE	Wobble Cone =	0.33,	Clok =	168.85	(deg)
6.700	-0.07	SPINUP	FREE	Wobble Cone =	0.33,	Clok =	168.78	(deg)
6.750	0.03	SPINUP	FREE	Wobble Cone =	0.33,	Clok =	168.70	(deg)
6.800	0.12	SPINUP	FREE	Wobble Cone =	0.34,	Clok =	168.62	(deg)
6.850	0.21	SPINUP	FREE	Wobble Cone =	0.34,	Clok =	168.55	(deg)
6.900	0.29	SPINUP	FREE	Wobble Cone =	0.34,	Clok =	168.48	(deg)
6.950	0.37	SPINUP	FREE	Wobble Cone =	0.34,	Clok =	168.41	(deg)
7.000	0.43	SPINUP	FREE	Wobble Cone =	0.35,	Clok =	168.34	(deg)

Time = 7.000   DesMode = SPINUP   CurMode = FREE
PL Nomspin = 4.78   Wobble Cone = 0.35   Wobble Clok = 168.34

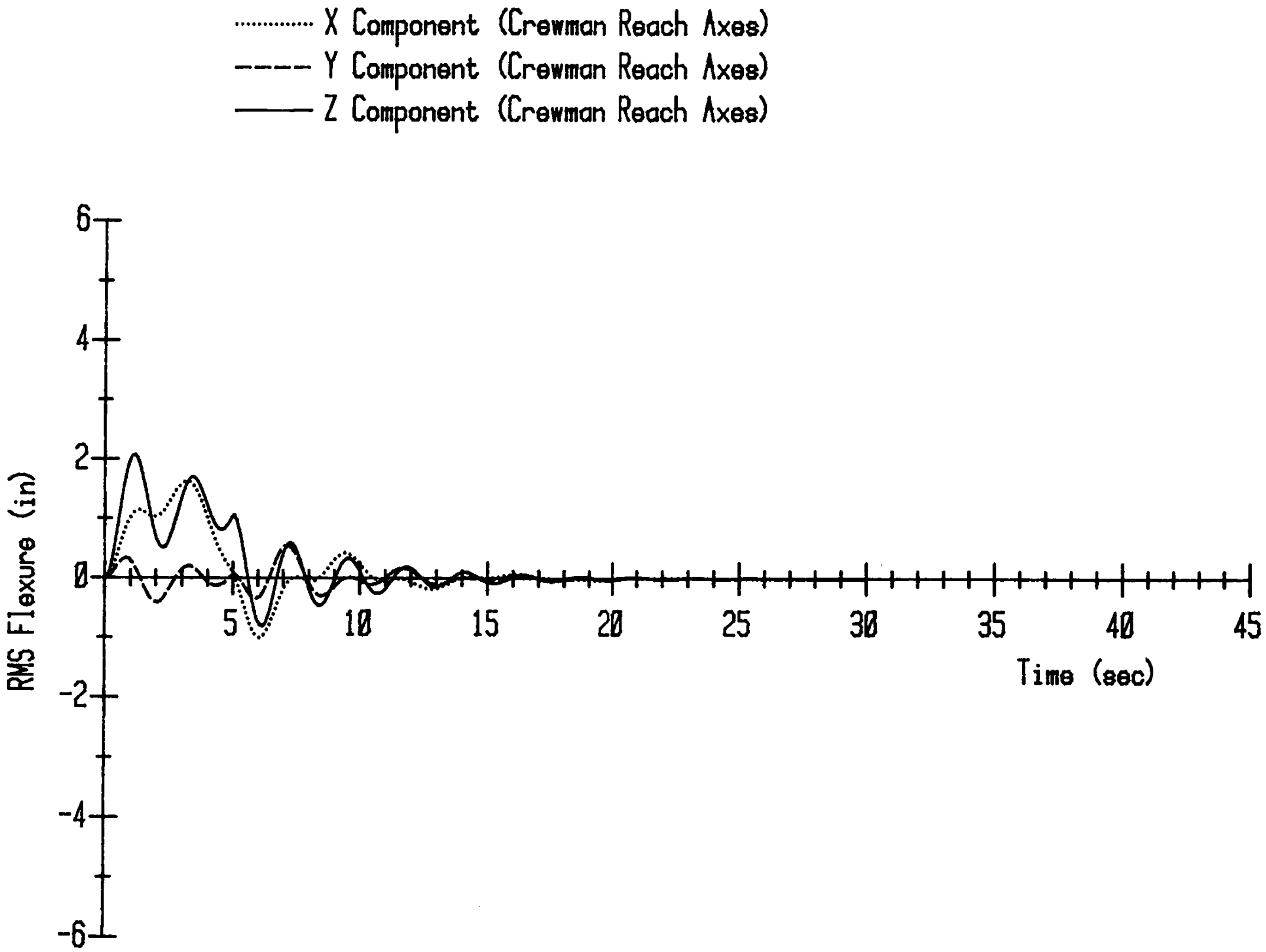
Cntrl@H: R	0.00	0.00	0.00 Torq	0.000	0.000	0.000 Forc
Cntrl@H:PB	0.00	0.00	0.00 Torq	0.000	0.000	0.000 Forc
PB Axes:PD	0.28	0.00	12.70 PYR	4.769	0.067	0.005 Rate
PL Hndl: R	-0.01	-0.02	-2.55 Pos	0.042	-0.025	-0.860 Vel
PB Axes:OB	0.28	0.00	12.70 PYR	4.769	0.067	0.005 Rate
PL CM :OB	28.30	8.49	-17.28 Pos	-0.005	-0.020	-0.176 Vel
PL Rot K E	22.06					
PLAngmo: I	-0.14	0.01	528.48 PYMag	528.483	0.075	1.315 Hxyz
PB Axes: I	0.20	0.01	12.77 PYR	4.783	0.050	0.011 Rate
PL CM : I	0.00	7.03	-0.84 Pos	0.000	-0.014	-0.154 Vel
OB Axes: I	-0.08	0.01	0.07 PYR	0.014	-0.017	0.002 Rate
Orb CM : I	-28.32	-1.49	16.39 Pos	-0.000	0.001	0.011 Vel

Time	Stepsize
7.000	0.100

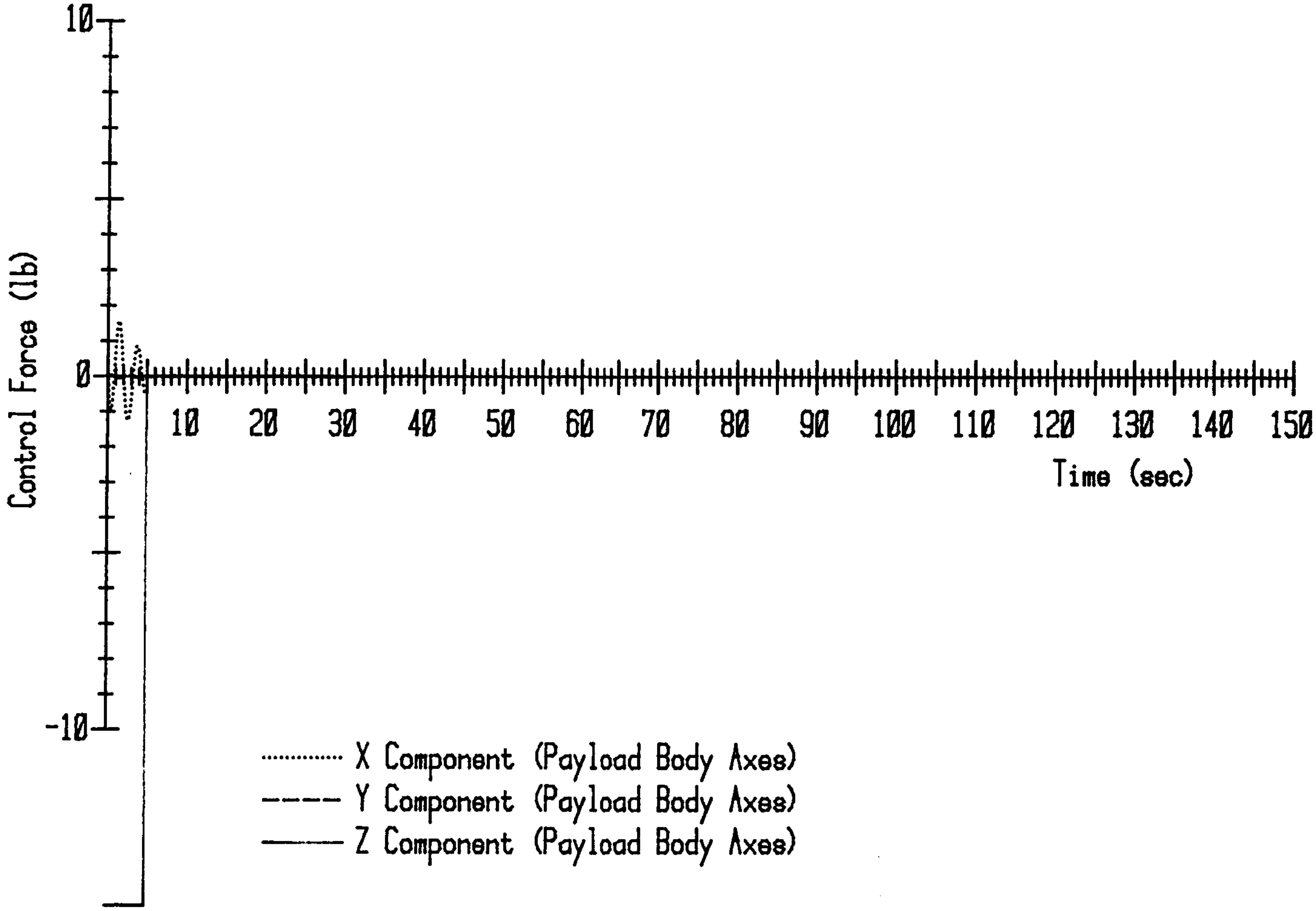
Time = 30.000   DesMode = SPINUP   CurMode = FREE
PL Nomspin = 4.78   Wobble Cone = 1.48   Wobble Clok = 170.13

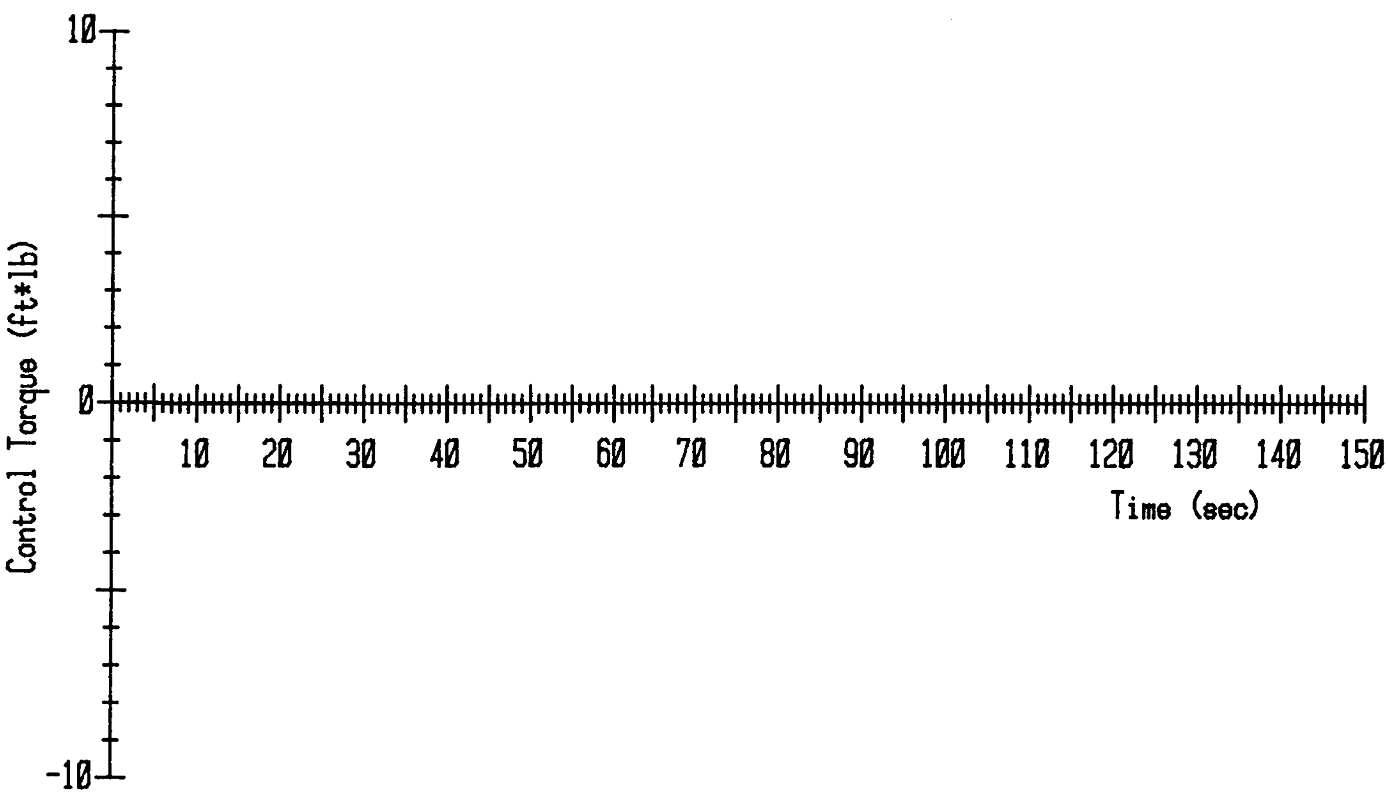
Cntrl@H: R	0.00	0.00	0.00 Torq	0.000	0.000	0.000 Forc
Cntrl@H:PB	0.00	0.00	0.00 Torq	0.000	0.000	0.000 Forc
PB Axes:PD	-0.23	1.32	122.39 PYR	4.768	0.060	0.110 Rate
PL Hndl: R	10.36	0.22	-11.07 Pos	0.484	0.006	0.144 Vel
PB Axes:OB	-0.23	1.32	122.39 PYR	4.768	0.060	0.110 Rate
PL CM :OB	28.16	8.01	-21.33 Pos	-0.007	-0.022	-0.176 Vel
PL Rot K E	22.06					
PLAngmo: I	-0.14	0.01	528.48 PYMag	528.483	0.075	1.315 Hxyz
PB Axes: I	-0.72	1.37	122.77 PYR	4.781	0.071	0.124 Rate
PL CM : I	0.01	6.71	-4.39 Pos	0.000	-0.014	-0.154 Vel
OB Axes: I	-0.48	0.05	0.38 PYR	0.014	-0.017	0.002 Rate
Orb CM : I	-28.32	-1.47	16.65 Pos	-0.000	0.001	0.011 Vel

*** END OF SIMULATION ***
Syncom 15#/24" Spinup Stroke With Lateral Corrective Force
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 13:36:40 Mon 03 Nov 1986



Syncom 15#/24" Spinup Stroke With Lateral Corrective Force





Syncom 15#/24" Spinup Stroke With Lateral Corrective Force

2.6. Syncom Coast After Anomalous Spinup Stroke

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:03:36 Mon 03 Nov 1986

"ssu" Position Parameters for Payload Flight Control

Nominal Orbiter STA of PL handling point...........(in)	800.00000
Nominal Orbiter BL of PL handling point...........(in)	24.00000
Nominal Orbiter WL of PL handling point...........(in)	600.00000
Payload STA of PL handle...........................(in)	-17.71284
Payload BL of PL handle...........................(in)	-86.12500
Payload WL of PL handle...........................(in)	0.00000
Crewman reach limit from RO in +/- Rx direction....(in)	12.00000
Crewman reach limit from RO in +/- Ry direction....(in)	12.00000
Crewman reach limit from RO in +/- Rz direction....(in)	12.00000
Rx position tolerance for PL handle................(in)	6.00000
Ry position tolerance for PL handle................(in)	6.00000
Rz position tolerance for PL handle................(in)	6.00000
Nominal vel for handle position correction.....(in/sec)	0.00000
Tolerance for corrective velocity..................(%)	0.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:04:21 Mon 03 Nov 1986

"adfra" Initial Conditions for the Simulation

Orbiter pitch wrt I axes..........................(deg)	0.00000
Orbiter yaw wrt I axes..........................(deg)	0.00000
Orbiter roll wrt I axes..........................(deg)	0.00000
Orbiter Bx component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter By component of ang vel wrt I axes....(deg/sec)	0.00000
Orbiter Bz component of ang vel wrt I axes....(deg/sec)	0.00000
Rx component of PL handle position.................(ft)	-0.08000
Ry component of PL handle position.................(ft)	0.62000
Rz component of PL handle position.................(ft)	-1.48000
Payload CM Xdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Ydot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Zdot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload pitch wrt desired attitude................(deg)	2.21000
Payload yaw wrt desired attitude................(deg)	-3.34000
Payload roll wrt desired attitude................(deg)	6.55000
Payload nominal spin rate.....................(deg/sec)	3.91600
Payload wobble cone angle.........................(deg)	10.98000
Payload wobble clok angle.........................(deg)	31.31000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:04:25 Mon 03 Nov 1986
Syncom Coast After First Spinup Stroke Applied 6" Fwd & Rotated 10 deg Aft

Time = 0.000 DesMode = FREE CurMode = FREE
PL Nomspin = 3.92 Wobble Cone = 10.98 Wobble Clok = 31.31

Cntrl@H: R	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
Cntrl@H:PB	0.00	0.00	0.00	Torq	0.000	0.000	0.000	Forc
PB Axes:PD	2.21	-3.34	6.55	PYR	3.923	0.605	-0.810	Rate
PL Hndl: R	-0.08	0.62	-1.48	Pos	0.061	0.117	-0.485	Vel
PB Axes:OB	2.21	-3.34	6.55	PYR	3.923	0.605	-0.810	Rate
PL CM :OB	25.48	9.05	-19.36	Pos	0.000	0.000	0.000	Vel
PL Rot K E	15.55							
PLAngmo: I	10.87	3.41	440.81	PYMag	432.130	26.230	-83.013	Hxyz
PB Axes: I	2.21	-3.34	6.55	PYR	3.923	0.605	-0.810	Rate
PL CM : I	-0.17	7.05	-0.68	Pos	0.000	0.000	0.000	Vel
OB Axes: I	0.00	0.00	0.00	PYR	0.000	0.000	0.000	Rate
Orb CM : I	-25.65	-2.00	18.68	Pos	0.000	0.000	0.000	Vel

Time	Stepsize
0.000	1.500
268.500	0.500

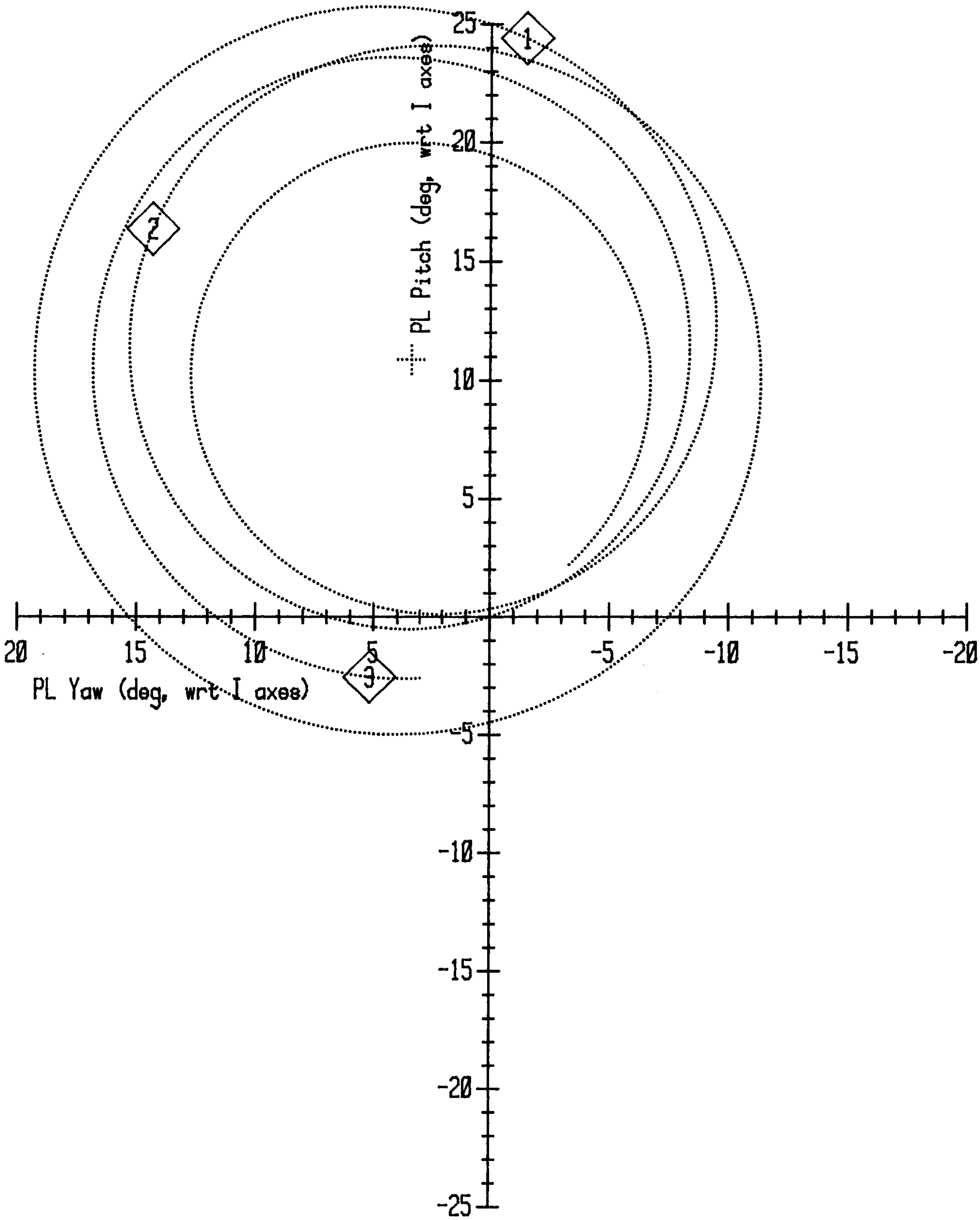
Time = 269.000 DesMode = FREE CurMode = FREE
PL Nomspin = 3.88 Wobble Cone = 13.44 Wobble Clok = -4.96

Cntrl@H: R	0.00	0.00	0.00 Torq	0.000	0.000	0.000	Forc
Cntrl@H:PB	0.00	0.00	0.00 Torq	0.000	0.000	0.000	Forc
PB Axes:PD	-2.58	2.88	7.58 PYR	3.884	-0.073	-1.140	Rate
PL Hndl: R	-0.07	-0.23	-1.61 Pos	0.056	0.124	-0.489	Vel
PB Axes:OB	-2.58	2.88	7.58 PYR	3.884	-0.073	-1.140	Rate
PL CM :OB	25.48	9.05	-19.36 Pos	0.000	0.000	0.000	Vel
PL Rot K E	15.55						
PLAngmo: I	10.87	3.41	440.81 PYMag	432.130	26.226	-83.012	Hxyz
PB Axes: I	-2.58	2.88	7.58 PYR	3.884	-0.073	-1.140	Rate
PL CM : I	-0.17	7.05	-0.68 Pos	0.000	0.000	0.000	Vel
OB Axes: I	0.00	0.00	0.00 PYR	0.000	0.000	0.000	Rate
Orb CM : I	-25.65	-2.00	18.68 Pos	0.000	0.000	0.000	Vel

*** END OF SIMULATION ***
Syncom Coast After First Spinup Stroke Applied 6" Fwd & Rotated 10 deg Aft
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:04:25 Mon 03 Nov 1986

Plot Symbol	Start Rev #	Time (sec)	Payload Attitude wrt I Axes Pitch	Yaw	Roll
"1"	1	87.750	24.41	-1.52	-1.03 (deg)
"2"	2	177.750	16.37	14.34	-0.44 (deg)
"3"	3	267.000	-2.54	5.17	-0.20 (deg)

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:04:25 Mon 03 Nov 1986



Syncom Coast After First Spinup Stroke Applied 6" Fwd & Rotated 10 deg Aft

2.7. EV2 / Rigid PFR Hold After Pitch Impulse From PRCS Jets

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:15:45 Mon 03 Nov 1986

"rigid" Translational Alacrity Matrix for RMS Flexure

Alacrity matrix, element [1,1]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [1,2]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [1,3]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [2,1]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [2,2]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [2,3]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [3,1]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [3,2]..........(ft/sec/sec/lb)	0.00000
Alacrity matrix, element [3,3]..........(ft/sec/sec/lb)	0.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:15:57 Mon 03 Nov 1986

"rigid" Translational Damping Constant Matrix for RMS Flexure

Damping constant matrix, element [1,1]......(lb/ft/sec)	0.00000
Damping constant matrix, element [1,2]......(lb/ft/sec)	0.00000
Damping constant matrix, element [1,3]......(lb/ft/sec)	0.00000
Damping constant matrix, element [2,1]......(lb/ft/sec)	0.00000
Damping constant matrix, element [2,2]......(lb/ft/sec)	0.00000
Damping constant matrix, element [2,3]......(lb/ft/sec)	0.00000
Damping constant matrix, element [3,1]......(lb/ft/sec)	0.00000
Damping constant matrix, element [3,2]......(lb/ft/sec)	0.00000
Damping constant matrix, element [3,3]......(lb/ft/sec)	0.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:16:11 Mon 03 Nov 1986

"rigid" Translational Spring Constant Matrix for RMS Flexure

Spring constant matrix, element [1,1]...........(lb/ft)	0.00000
Spring constant matrix, element [1,2]...........(lb/ft)	0.00000
Spring constant matrix, element [1,3]...........(lb/ft)	0.00000
Spring constant matrix, element [2,1]...........(lb/ft)	0.00000
Spring constant matrix, element [2,2]...........(lb/ft)	0.00000
Spring constant matrix, element [2,3]...........(lb/ft)	0.00000
Spring constant matrix, element [3,1]...........(lb/ft)	0.00000
Spring constant matrix, element [3,2]...........(lb/ft)	0.00000
Spring constant matrix, element [3,3]...........(lb/ft)	0.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:16:35 Mon 03 Nov 1986

"sncmh" Payload Inertia Data

Payload weight.....................................(lb)	15227.00000
CM STA (structural x coordinate of mass center)....(in)	-17.71000
CM BL (structural y coordinate of mass center)....(in)	0.19000
CM WL (structural z coordinate of mass center)....(in)	0.09000
Ixx about CM, structural coordinates.......(slug*ft*ft)	6322.70000
Iyy about CM, structural coordinates.......(slug*ft*ft)	4424.40000
Izz about CM, structural coordinates.......(slug*ft*ft)	5208.10000
Pxy about CM, structural coordinates.......(slug*ft*ft)	-38.70000
Pxz about CM, structural coordinates.......(slug*ft*ft)	-26.30000
Pyz about CM, structural coordinates.......(slug*ft*ft)	30.30000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:16:55 Mon 03 Nov 1986

"hold" Position Parameters for Payload Flight Control

Nominal Orbiter STA of PL handling point...........(in)	768.00000
Nominal Orbiter BL of PL handling point...........(in)	93.00000
Nominal Orbiter WL of PL handling point...........(in)	486.00000
Payload STA of PL handle..........................(in)	-17.71000
Payload BL of PL handle..........................(in)	86.12500
Payload WL of PL handle..........................(in)	0.00000
Crewman reach limit from RO in +/- Rx direction....(in)	12.00000
Crewman reach limit from RO in +/- Ry direction....(in)	12.00000
Crewman reach limit from RO in +/- Rz direction....(in)	12.00000
Rx position tolerance for PL handle...............(in)	6.00000
Ry position tolerance for PL handle...............(in)	6.00000
Rz position tolerance for PL handle...............(in)	6.00000
Nominal vel for handle position correction.....(in/sec)	1.00000
Tolerance for corrective velocity..................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:17:16 Mon 03 Nov 1986

"hold" Attitude Parameters for Payload Flight Control

Crewman pitch wrt Orbiter body axes...............(deg)	90.00000
Crewman yaw wrt Orbiter body axes...............(deg)	-45.00000
Crewman roll wrt Orbiter body axes...............(deg)	90.00000
Desired Payload pitch wrt Crewman body axes.......(deg)	0.00000
Desired Payload yaw wrt Crewman body axes.......(deg)	-90.00000
Desired Payload roll wrt Crewman body axes.......(deg)	180.00000
Positive PL pitch limit wrt desired attitude......(deg)	15.00000
Positive PL yaw limit wrt desired attitude......(deg)	15.00000
Positive PL roll limit wrt desired attitude......(deg)	10.00000
Negative PL pitch limit wrt desired attitude......(deg)	-15.00000
Negative PL yaw limit wrt desired attitude......(deg)	-15.00000
Negative PL roll limit wrt desired attitude......(deg)	-20.00000
PL pitch tolerance wrt desired attitude...........(deg)	2.00000
PL yaw tolerance wrt desired attitude...........(deg)	2.00000
PL roll tolerance wrt desired attitude...........(deg)	2.00000
Nominal maneuver rate about PL Bx axis........(deg/sec)	0.25000
Nominal maneuver rate about PL By axis........(deg/sec)	0.25000
Nominal maneuver rate about PL Bz axis........(deg/sec)	0.25000
Maneuver rate tolerance.............................(%)	10.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:17:34 Mon 03 Nov 1986

"hold" Force & Torque Parameters for Payload Flight Control

```
Spinup axial      (PL Bx) force....................(lb)          0.00000
Spinup normal     (PL By) force....................(lb)          0.00000
Spinup tangential (PL Bz) force....................(lb)        -10.00000
Despin axial      (PL Bx) force....................(lb)          0.00000
Despin normal     (PL By) force....................(lb)          0.00000
Despin tangential (PL Bz) force....................(lb)         10.00000
Axial      (PL Bx) force limit for capture.........(lb)         10.00000
Normal     (PL By) force limit for capture.........(lb)         10.00000
Tangential (PL Bz) force limit for capture.........(lb)         10.00000
Roll  (PL Bx) torque limit for capture..........(ft*lb)          2.00000
Pitch (PL By) torque limit for capture..........(ft*lb)         10.00000
Yaw   (PL Bz) torque limit for capture..........(ft*lb)         10.00000
Time constant for computing desired accelerations (sec)          5.00000
```

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:17:55 Mon 03 Nov 1986

"hold" Initial Conditions for the Simulation

Orbiter pitch wrt I axes..........................(deg)	0.00000
Orbiter yaw wrt I axes..........................(deg)	0.00000
Orbiter roll wrt I axes..........................(deg)	0.00000
Orbiter Bx component of ang vel wrt I axes....(deg/sec)	-0.00026
Orbiter By component of ang vel wrt I axes....(deg/sec)	-0.20331
Orbiter Bz component of ang vel wrt I axes....(deg/sec)	0.00007
Rx component of PL handle position.................(ft)	0.00000
Ry component of PL handle position.................(ft)	0.00000
Rz component of PL handle position.................(ft)	0.00000
Payload CM Xdot wrt Orbiter body axes..........(ft/sec)	-0.03686
Payload CM Ydot wrt Orbiter body axes..........(ft/sec)	0.00000
Payload CM Zdot wrt Orbiter body axes..........(ft/sec)	0.10081
Payload pitch wrt desired attitude................(deg)	0.00000
Payload yaw wrt desired attitude................(deg)	0.00000
Payload roll wrt desired attitude................(deg)	0.00000
Payload nominal spin rate.....................(deg/sec)	0.00000
Payload wobble cone angle........................(deg)	0.00000
Payload wobble clok angle........................(deg)	0.00000

MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:17:57 Mon 03 Nov 1986
EV2 / Rigid PFR Hold After -0.2 deg/sec Pitch Impulse from PRCS Tail Jets

Time = 0.000   DesMode = HOLD   CurMode = CAPTURE
PL Nomspin = 0.00   Wobble Cone = 0.00   Wobble Clok = 0.00

Cntrl@H: R	2.21	-0.04	10.00	Torq	-0.045	1.358	0.005	Forc
Cntrl@H:PB	0.04	-2.21	-10.00	Torq	-1.358	0.045	-0.005	Forc
PB Axes:PD	0.00	-3.9L-47	0.00	PYR	-0.000	0.144	0.144	Rate
PL Hndl: R	0.00	-0.00	0.00	Pos	-0.000	-0.069	0.000	Vel
PB Axes:OB	180.00	-0.00	-45.00	PYR	-0.000	0.144	0.144	Rate
PL CM :OB	28.32	2.68	-14.23	Pos	-0.087	0.000	0.000	Vel
PL Rot K E	0.00							
PLAngmo: I	0.00	0.00	0.00	PYMag	0.000	0.000	0.000	Hxyz
PB Axes: I	180.00	-0.00	-45.00	PYR	0.000	0.000	0.000	Rate
PL CM : I	0.00	-5.07	-5.06	Pos	-0.037	0.000	0.101	Vel
OB Axes: I	0.00	0.00	0.00	PYR	-0.000	-0.203	0.000	Rate
Orb CM : I	-28.32	-7.75	9.18	Pos	0.000	0.000	0.000	Vel

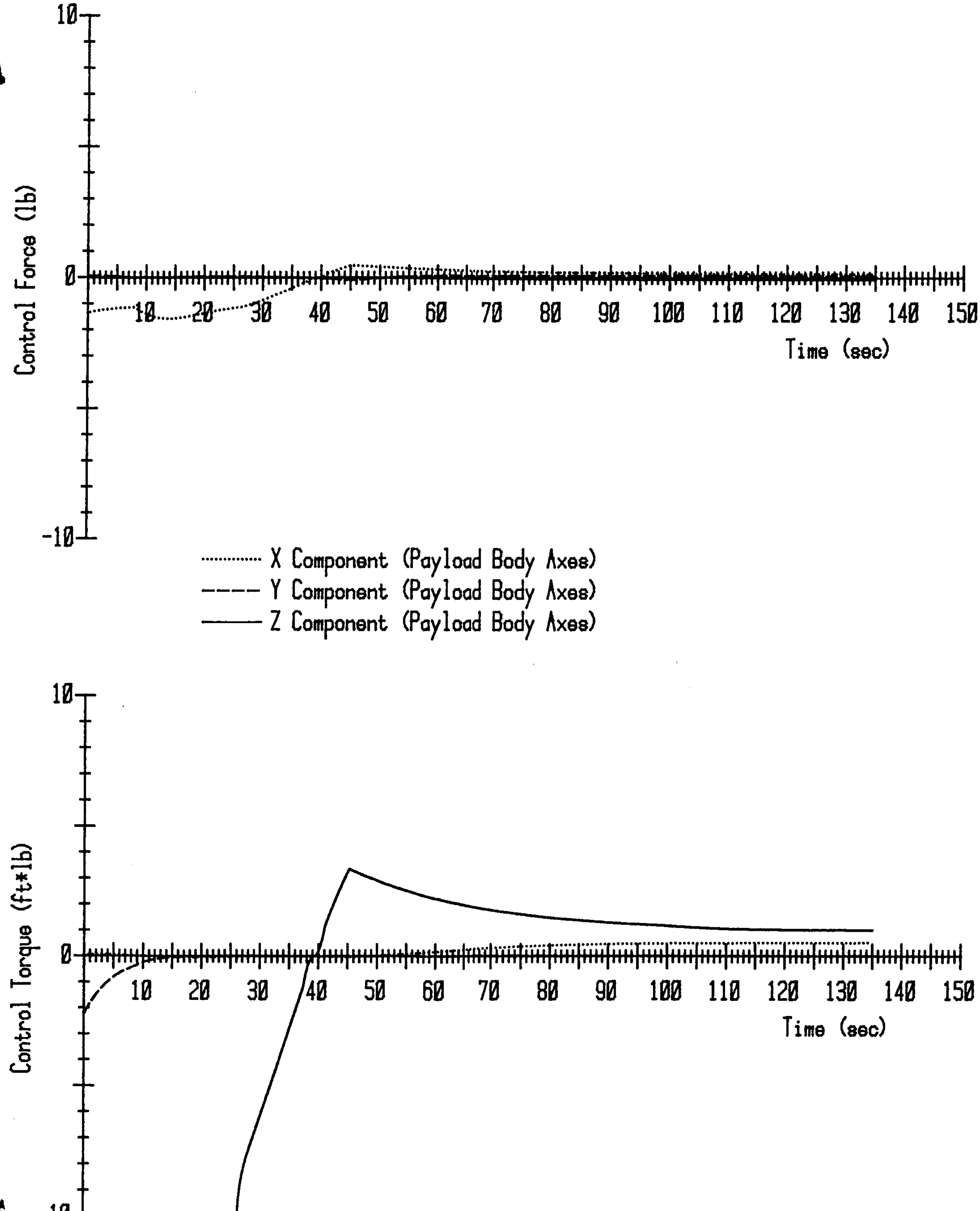
Time	Stepsize
0.000	0.200

Time = 135.000 DesMode = HOLD CurMode = HOLD
PL Nomspin = -0.00 Wobble Cone = 90.48 Wobble Clok = -40.08

Cntrl@H: R	-0.02	-0.52	-0.97	Torq	-0.069	-0.134	0.071	Forc
Cntrl@H:PB	0.50	-0.00	0.98	Torq	0.137	0.064	-0.069	Forc
PB Axes:PD	0.98	1.99	-0.11	PYR	-0.001	0.001	0.000	Rate
PL Hndl: R	0.14	0.49	-0.35	Pos	0.003	0.008	-0.006	Vel
PB Axes:OB	-177.90	0.71	-45.10	PYR	-0.001	0.001	0.000	Rate
PL CM :OB	28.56	2.84	-14.59	Pos	0.008	0.002	-0.006	Vel
PL Rot K E	0.03							
PLAngmo: I	-116.49	-85.07	16.83	PYMag	-0.645	-16.770	1.294	Hxyz
PB Axes: I	155.19	0.63	-44.99	PYR	-0.002	-0.139	-0.141	Rate
PL CM : I	3.16	-4.91	9.05	Pos	0.007	0.002	0.108	Vel
OB Axes: I	-26.91	0.08	-0.11	PYR	-0.002	-0.199	0.000	Rate
Orb CM : I	-28.91	-7.76	9.14	Pos	-0.003	-0.000	-0.001	Vel

*** END OF SIMULATION ***
EV2 / Rigid PFR Hold After -0.2 deg/sec Pitch Impulse from PRCS Tail Jets
MANHANDLE Version 04B (11:11:13 Tue 28 Oct 1986) Run @ 14:17:57 Mon 03 Nov 1986

MANHANDLE Version Ø4B (11:11:13 Tue 28 Oct 1986) Run @ 14:17:57 Mon Ø3 Nov 1986



EV2 / Rigid PFR Hold After -Ø.2 deg/sec Pitch Impulse from PRCS Tail Jets