

AD-A240 947



ANNUAL REPORT

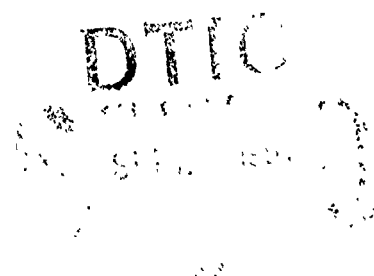
VOLUME 3

PART 3

TASK 3: SPECIAL STUDIES

REPORT NO. AR-0142-91-002

September 27, 1991



2

---

GUIDANCE, NAVIGATION AND CONTROL  
DIGITAL EMULATION TECHNOLOGY LABORATORY

Contract No. DASG60-89-C-0142

Sponsored By

The United States Army Strategic Defense Command

---

COMPUTER ENGINEERING RESEARCH LABORATORY

Georgia Institute of Technology

Atlanta, Georgia 30332-0540

---

Contract Data Requirements List Item A005

Period Covered: FY 91

Type Report: Annual

91-11307



## DISCLAIMER

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

## DISTRIBUTION CONTROL

(1) This material is approved for public release unlimited distribution.

(2) This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227-7013, October 1988.



|                    |                                             |
|--------------------|---------------------------------------------|
| Accession For      |                                             |
| Dist               | General <input checked="" type="checkbox"/> |
| Dist               | Tab <input type="checkbox"/>                |
| Dist               | Background <input type="checkbox"/>         |
| Justification      |                                             |
| By                 |                                             |
| Distribution/      |                                             |
| Availability Codes |                                             |
| Dist               | Availability                                |
| A-1                |                                             |

**ANNUAL REPORT**  
**VOLUME 3**  
**PART 3**  
**TASK 3: SPECIAL STUDIES**

September 27, 1991

---

**Authors**

**Cecil O. Alford, Richard M. Pitts and Philip R. Bingham**

**COMPUTER ENGINEERING RESEARCH LABORATORY**

**Georgia Institute of Technology**

**Atlanta, Georgia 30332-0540**

---

**Eugene L. Sanders**

**USASDC**

**Contract Monitor**

**Cecil O. Alford**

**Georgia Tech**

**Project Director**

---

**Copyright 1991**

**Georgia Tech Research Corporation**

**Centennial Research Building**

**Atlanta, Georgia 30332**

# TABLE OF CONTENTS

|                                                          | PAGE |
|----------------------------------------------------------|------|
| PART 1                                                   |      |
| 1. Introduction .....                                    | 1    |
| 2. Contract Interfaces .....                             | 12   |
| 2.1 AHAT.....                                            | 12   |
| 2.2 LATS.....                                            | 12   |
| 2.3 JAYCOR.....                                          | 12   |
| 2.4 KDEC.....                                            | 12   |
| 2.5 Other.....                                           | 13   |
| 2.6 Working Groups.....                                  | 13   |
| 3. Technical Issues .....                                | 14   |
| 3.1 LATS Seeker.....                                     | 14   |
| 3.1.1 Dithering                                          |      |
| 3.1.2 Delayed Gamma Model                                |      |
| 3.1.3 Staggered Row FPA                                  |      |
| 3.2 Discrimination Techniques.....                       | 15   |
| 3.2.1 Neural Network                                     |      |
| 3.2.2 Temperature                                        |      |
| 3.2.3 Multiple Sensors                                   |      |
| 3.3 Parallel EXOSIM .....                                | 16   |
| 3.3.1 Boost Phase                                        |      |
| 3.3.2 Midcourse/Teminal Phase                            |      |
| 3.4 Benchmarks .....                                     | 16   |
| 3.4.1 Signal Processing Benchmark                        |      |
| 3.4.2 Simulation Benchmarks                              |      |
| 4. Parallel Programming Methodology .....                | 18   |
| 4.1 Introduction.....                                    | 18   |
| 4.2 The EXOSIM Engagement.....                           | 19   |
| 4.3 Midcourse/Terminal Phase Simulation .....            | 19   |
| 4.3.1 PFP Test Results                                   |      |
| 4.4 Boost Phase Simulation.....                          | 26   |
| 4.5 Comparative Results .....                            | 26   |
| 5. References .....                                      | 33   |
| PART 2                                                   |      |
| 6. Appendix A:EXOSIM V1.0 Boost Phase.....               | 34   |
| PART 3                                                   |      |
| 7. Appendix B:EXOSIM V2.0 Midcourse/Terminal Phase ..... | 220  |

## List of Figures

| FIGURE      |                                                                             | PAGE |
|-------------|-----------------------------------------------------------------------------|------|
| Figure 1.1  | DETL Programmatic Tasks .....                                               | 2    |
| Figure 1.2  | GN&C and PFP Software Development .....                                     | 3    |
| Figure 1.3  | KEW Interceptor Emulation Status .....                                      | 4    |
| Figure 1.4  | Special Purpose Software Development Status .....                           | 5    |
| Figure 1.5  | VLSI Chip Set Design Status .....                                           | 6    |
| Figure 1.6  | GN&C Processor Prototype Development .....                                  | 7    |
| Figure 1.7  | Task 3 Schedules .....                                                      | 9    |
| Figure 4.1  | EXOSIM Engagement .....                                                     | 20   |
| Figure 4.2  | Functional System Blocks for EXOSIM Engagement .....                        | 21   |
| Figure 4.3  | EXOSIM Midcourse/Terminal Phase Block Diagram .....                         | 22   |
| Figure 4.4  | Programming Framework for Parallel Implementation of the Terminal Phase ... | 23   |
| Figure 4.5  | Parallel EXOSIM: Terminal Phase Implementation .....                        | 25   |
| Figure 4.6  | EXOSIM Boost Phase Block Diagram .....                                      | 27   |
| Figure 4.7  | Parallel EXOSIM Boost Phase: Fortran Version .....                          | 28   |
| Figure 4.8  | Parallel EXOSIM Boost Phase: Ada Version .....                              | 29   |
| Figure 4.9  | Comparison of Source Code and Intermediate C Code .....                     | 30   |
| Figure 4.10 | Comparison of Object Code Size .....                                        | 31   |
| Figure 4.11 | Benchmark Execution Times for EXOSIM End-to-End .....                       | 32   |

**VOLUME 3**  
**PART 3**  
**TASK 3: SPECIAL STUDIES**

7. Appendix B: EXOSIM v2.0 Midcourse and Terminal Phases

**B.1 Mainline (FORTRAN)****B.1.1 Uup00.for**

```

C      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / STORAG / XINT , TINT , XDOTL
      COMMON / RMASS / TLSTM , MASSL
      COMMON / RMISSL / XYZLCH

      REAL XINT(50) , TINT(50) , XDOTL(50)
      REAL XYZLCH(3) , MASSL

C      OUTPUTS

      REAL MXVCS , MYVCS , MZVCS
      REAL MXACS , MYACS , MZACS
      REAL MX , MY , MZ
      REAL MACH
      REAL MDO TV , MDO TA
      REAL CIM(9)

C      NAMELIST INPUTS

      REAL IXX , IYY , IZZ
      REAL MASS
      REAL IMPULS , QUAT(4) , MDOT
      REAL QUATD(4)

      INTEGER SEKTYP

      REAL TSTEP,DELT,LATLP,LOGLP
      REAL TMSUDRIV,TMSUSTEP

      double precision d_xd,d_yd,d_zd

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSDATA35.DAT')
$INCLUDE('^/INCLUDE/SSDATA38.DAT')
$INCLUDE('^/INCLUDE/SSDATA39.DAT')
$INCLUDE('^/INCLUDE/SSDATA42.DAT')
$INCLUDE('^/INCLUDE/SSDATA44.DAT')
$INCLUDE('^/INCLUDE/SSDATA45.DAT')
$INCLUDE('^/INCLUDE/SSDATA46.DAT')
$INCLUDE('^/INCLUDE/SSDATA47.DAT')
$INCLUDE('^/INCLUDE/SSDATA48.DAT')
$INCLUDE('^/INCLUDE/SSDATA49.DAT')
$INCLUDE('^/INCLUDE/SSDATA50.DAT')
$INCLUDE('^/INCLUDE/SSDATA01.DAT')
$INCLUDE('^/INCLUDE/SSDATA17.DAT')
$INCLUDE('^/INCLUDE/SSDATA18.DAT')
$INCLUDE('^/INCLUDE/SSDATA21.DAT')

```

```

$INCLUDE('^/INCLUDE/SSDATA22.DAT')
$INCLUDE('^/INCLUDE/SSDATA23.DAT')
$INCLUDE('^/INCLUDE/SSDATA28.DAT')
$INCLUDE('^/INCLUDE/SSDATA29.DAT')
$INCLUDE('^/INCLUDE/SSDATA30.DAT')
$INCLUDE('^/INCLUDE/SSDATA71.DAT')
$INCLUDE('^/INCLUDE/SSTIMING.DAT')
$INCLUDE('SSp00.DAT')

```

```

* INITIALIZE 80x87
  CALL CW87

```

```

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed  C
C           within this loop                      C
C-----C

```

```

c   call initialize_timing()

```

```

1000 CONTINUE

```

```

c   call start_timing(0)

```

```

C   WRITE(*,*) '-----BEGINNING OF LOOP-----'

```

```

C-----C
C----- MISSILE STATE UPDATE MODULE -----C
C-----C
C           Integrate missile states to current time  C
C-----C

```

```

C1001      format(1x,f7.4,3(a,lpe13.6))
C1002      format(1x,3(a,lpe13.6))
C          write(message,1001)t,'p ','p',' q ','q',' r ','r
C          call outmes(message)
C          write(message,1002)'          pd ','pd',' qd ','qd',' rd ','rd
C          call outmes(message)
C          write(message,1002)'          cim(1) ','cim(1),' (2) ','cim(2),
C          *                      ' (3) ','cim(3)
C          call outmes(message)
C          write(message,1002)'          (4) ','cim(4),' (5) ','cim(5),
C          *                      ' (6) ','cim(6)
C          call outmes(message)
C          write(message,1002)'          (7) ','cim(7),' (8) ','cim(8),
C          *                      ' (9) ','cim(9)
C          call outmes(message)
IF ( tstep .ge. tmsudriv ) THEN
      tmsudriv = tmsudriv + tmsustep

```

```

C-----C
C----- MASS PROPERTIES MODULE -----C
C-----C
C           Update mass flow rate, cg and inertia  C
C-----C

```

```

CALL MASSPR(T,MDOA,MDOIV,MASS,EISP,IMASS,

```

```

      .      MDOT,WEIGHT,WDOTTP,WDOTKV,WDOTTI,IXX,
      .      IYY,IZZ)

C-----C
C----- VEHICLE STATES MODULE -----C
C-----C
C      Compute missile state derivatives      C
C-----C

      CALL MISSIL2(T,QUAT,CIM,P,Q,R,IXX,IYY,IZZ,
      .      MXACS,MXVCS,MYACS,MYVCS,MZACS,
      .      MZVCS,XD,YD,ZD,NCLEAR,PD,QD,RD,
      .      MX,MY,MZ,U,V,W,QUATD,PHI,THT,PSI)

C-----C
C      MISSILE STATE INTEGRATION MODULE      C
C-----C
C      Revise missile states using derivatives C
C      just computed . Missile states must not C
C      be integrated if a table lookup index C
C      transition has occurred since the last C
C      integration step . The next integration C
C      step should be rescheduled to coincide C
C      with the earliest detected table lookup C
C      index transition instead . Otherwise C
C      schedule the next integration step to C
C      occur at the default step size .      C
C-----C

C      TRAPEZOIDAL INTEGRATION FOR SIMPLICITY

      CALL spINTEG ( MASS      , MDOT      , T , 1 )
      CALL spINTEG ( WKV      , WDOTKV     , T , 5 )
      CALL spINTEG ( P        , PD         , T , 12 )
      CALL spINTEG ( Q        , QD         , T , 13 )
      CALL spINTEG ( R        , RD         , T , 14 )
      CALL spINTEG ( QUAT(1)  , QUATD(1)   , T , 15 )
      CALL spINTEG ( QUAT(2)  , QUATD(2)   , T , 16 )
      CALL spINTEG ( QUAT(3)  , QUATD(3)   , T , 17 )
      CALL spINTEG ( QUAT(4)  , QUATD(4)   , T , 18 )

C      SAVE TIME OF LAST MISSILE STATE UPDATE

      TLMSU = T

      ENDIF

C-----C
C----- SEPARATION MODULE -----C
C-----C
C      Models discontinuities occuring during C
C      stage separation                        C
C-----C

C      NOSE FAIRING / BOOST ADAPTER SEPARATION

```

```

      IF ( IDROP.EQ.1 .OR. (ABS(T-TDROP).LE.DTEPS
*      .AND. IGIT.EQ.1 ) ) THEN
      WKV      = WKV - WBANF
      MASS     = WKV/XMTOF
      WRITE(MESSAGE,155) T
      CALL OUTMES(MESSAGE)
155      FORMAT(1X,E16.9,' DROP NOSE FAIRING AND BOOST ADAPTER')

C      REINITIALIZE PERTINENT INTEGRALS

      CALL spINTEGI ( MASS      , 0.0e0 , T      , 1 )
      CALL spINTEGI ( WPROP     , 0.0e0 , T      , 2 )
      CALL spINTEGI ( IMPULS    , 0.0e0 , T      , 3 )
      CALL spINTEGI ( WKV       , 0.0e0 , T      , 5 )
      ENDIF

C-----C
C----- Processor communication -----C
C-----C

c      call switch_timing()

C----- Communicate with p01 -----C

      CALL SEND_REAL_32BIT( IXX )
      CALL SEND_REAL_32BIT( IYY )
      CALL SEND_REAL_32BIT( IZZ )
      CALL SEND_REAL_32BIT( MASS )

C----- Communicate with p03 -----C

      CALL SEND_REAL_32BIT( P )
      CALL SEND_REAL_32BIT( Q )
      CALL SEND_REAL_32BIT( R )
c      CALL RECEIVE_REAL_64BIT( d_XD )
c      XD = d_XD
c      CALL RECEIVE_REAL_64BIT( d_YD )
c      YD = d_YD
c      CALL RECEIVE_REAL_64BIT( d_ZD )
c      ZD = d_ZD
      CALL RECEIVE_REAL_32BIT( XD )
      CALL RECEIVE_REAL_32BIT( YD )
      CALL RECEIVE_REAL_32BIT( ZD )
      CALL SEND_REAL_32BIT( CIM(1) )
      CALL SEND_REAL_32BIT( CIM(2) )
      CALL SEND_REAL_32BIT( CIM(3) )
      CALL SEND_REAL_32BIT( CIM(4) )
      CALL SEND_REAL_32BIT( CIM(5) )
      CALL SEND_REAL_32BIT( CIM(6) )
      CALL SEND_REAL_32BIT( CIM(7) )
      CALL SEND_REAL_32BIT( CIM(8) )
      CALL SEND_REAL_32BIT( CIM(9) )

C----- Communicate with p01 -----C

      CALL RECEIVE_SIGNED_16BIT( IDROP )

C----- Receive from ACSTHR and VCSTHR -----C

      CALL receive_REAL_32BIT( mdotV )
      CALL receive_REAL_32BIT( mdotA )
      CALL receive_REAL_32BIT( mxvcs )
      CALL receive_REAL_32BIT( myvcs )

```

```

CALL receive_REAL_32BIT( mzvcs )
CALL receive_REAL_32BIT( mxacs )
CALL receive_REAL_32BIT( myacs )
CALL receive_REAL_32BIT( mzacs )

CALL SEND_REAL_32BIT( PD )
CALL SEND_REAL_32BIT( QD )
CALL SEND_REAL_32BIT( RD )

c    call switch_timing()

C-----C
C----- OUTPUT MODULE -----C
C-----C
C          Creates print and plot output data
C          files
C-----C
c    call stop_timing()

c    if ( mod(int(tstep),int(dtpert)).eq.0 ) then
c        call output_timing()
c        call initialize_timing()
c    endif

C-----C
C----- TERMINATION LOGIC -----C
C-----C
C          Defines the simulation termination
C          conditions
C-----C

C    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

    IEXIT = 0

C    ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
C    EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED

    IF ( T.GE.TFINAL ) THEN
        IEXIT = 1
    endif

C    increment time

    TSTEP = TSTEP + 1.0e0
    T = TSTEP * DELT

C    CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

    IF ( IEXIT.EQ.0 ) GO TO 1000

CALL OUTMES('ERROR: Exit from P00')
END

```

## B.1.2 Up01.for

```

C      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C

      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / ROBTRG / FIRST2 , TL2      , GRTPST
      COMMON / RNAVIG / GRLAST , TONAV , MNAV , DTX0 , DTY0 ,
      DTZ0

      DOUBLE PRECISION TSTEP,DELT
      DOUBLE PRECISION TIMUDRIV,TGPUDRIV
      DOUBLE PRECISION TIMUSTEP,TGPUSTEP

      DOUBLE PRECISION GRLAST(3)      , GRTPST(3)
      DOUBLE PRECISION VTIC(5,3)      , rtic(5,3)
      DOUBLE PRECISION GRT(5,3),GRTEST(3),RTEST(3),VTEST(3)
      DOUBLE PRECISION RREL(3),VREL(3)

      INTEGER          FIRST1      , FIRST2
      INTEGER SEKTYP

C      OUTPUTS

      DOUBLE PRECISION MASS
      DOUBLE PRECISION TI2M(9)
      DOUBLE PRECISION QS1(4)      , VMI(3)      , RMI(3)
      DOUBLE PRECISION VMIR(3)
      DOUBLE PRECISION AT(3)
      DOUBLE PRECISION GR(3)      , CIE(9)

C      NAMELIST INPUTS

      DOUBLE PRECISION XYZE(3)      , XYZED(3)
      DOUBLE PRECISION PULSEG(3)
      DOUBLE PRECISION PULSEA(3)

      DOUBLE PRECISION LATLP, LONGLP
      DOUBLE PRECISION RMIR(3)

      real s_mass,s_pulsea(3),s_pulseseg(3)
      real S_XD,S_YD,S_ZD,S_GR(3)

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSDATA35.DAT')
$INCLUDE('^/INCLUDE/SSDATA38.DAT')
$INCLUDE('^/INCLUDE/SSDATA39.DAT')
$INCLUDE('^/INCLUDE/SSDATA42.DAT')
$INCLUDE('^/INCLUDE/SSDATA44.DAT')
$INCLUDE('^/INCLUDE/SSDATA45.DAT')
$INCLUDE('^/INCLUDE/SSDATA46.DAT')
$INCLUDE('^/INCLUDE/SSDATA47.DAT')
$INCLUDE('^/INCLUDE/SSDATA48.DAT')
$INCLUDE('^/INCLUDE/SSDATA49.DAT')
$INCLUDE('^/INCLUDE/SSDATA50.DAT')

```

```

$INCLUDE('^/INCLUDE/SSDATA01.DAT')
$INCLUDE('^/INCLUDE/SSDATA17.DAT')
$INCLUDE('^/INCLUDE/SSDATA18.DAT')
$INCLUDE('^/INCLUDE/SSDATA21.DAT')
$INCLUDE('^/INCLUDE/SSDATA22.DAT')
$INCLUDE('^/INCLUDE/SSDATA23.DAT')
$INCLUDE('^/INCLUDE/SSDATA28.DAT')
$INCLUDE('^/INCLUDE/SSDATA29.DAT')
$INCLUDE('^/INCLUDE/SSDATA30.DAT')
$INCLUDE('^/INCLUDE/SSDATA71.DAT')
$INCLUDE('^/INCLUDE/SSTIMING.DAT')
$INCLUDE('SSp01.DAT')

```

```

* INITIALIZE 80x87
  CALL CW87

```

```

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed   C
C           within this loop                       C
C-----C

```

```

1000 CONTINUE

```

```

C   WRITE(*,*) '-----BEGINNING OF LOOP-----'

```

```

C-----C
C----- Processor communication -----C
C-----C

```

```

C----- COMMUNICATION WITH P00 -----C

```

```

CALL RECEIVE_REAL_32BIT( S_GR(01) )
CALL RECEIVE_REAL_32BIT( S_GR(02) )
CALL RECEIVE_REAL_32BIT( S_GR(03) )
GR(1) = S_GR(1)
GR(2) = S_GR(2)
GR(3) = S_GR(3)
CALL RECEIVE_REAL_32BIT( s_MASS )
MASS = s_MASS
CALL RECEIVE_REAL_32BIT( s_PULSEA(01) )
PULSEA(01) = s_PULSEA(01)
CALL RECEIVE_REAL_32BIT( s_PULSEA(02) )
PULSEA(02) = s_PULSEA(02)
CALL RECEIVE_REAL_32BIT( s_PULSEA(03) )
PULSEA(03) = s_PULSEA(03)
CALL RECEIVE_REAL_32BIT( s_PULSEG(01) )
PULSEG(01) = s_PULSEG(01)
CALL RECEIVE_REAL_32BIT( s_PULSEG(02) )
PULSEG(02) = s_PULSEG(02)
CALL RECEIVE_REAL_32BIT( s_PULSEG(03) )
PULSEG(03) = s_PULSEG(03)
CALL RECEIVE_REAL_64BIT( XYZE(01) )
CALL RECEIVE_REAL_64BIT( XYZE(02) )
CALL RECEIVE_REAL_64BIT( XYZE(03) )
CALL RECEIVE_REAL_64BIT( XYZED(01) )
CALL RECEIVE_REAL_64BIT( XYZED(02) )
CALL RECEIVE_REAL_64BIT( XYZED(03) )

```

C----- COMMUNICATION WITH SEEKER -----C

```

CALL RECEIVE_REAL_64BIT( X )
CALL RECEIVE_REAL_64BIT( Y )
CALL RECEIVE_REAL_64BIT( Z )
CALL RECEIVE_REAL_32BIT( S_XD )
CALL RECEIVE_REAL_32BIT( S_YD )
CALL RECEIVE_REAL_32BIT( S_ZD )
XD = S_XD
YD = S_YD
ZD = S_ZD

```

C----- COMMUNICATION WITH CORVEL -----C

```

CALL SEND_REAL_32BIT( snl(RMIR(1) ) )
CALL SEND_REAL_32BIT( snl(RMIR(2) ) )
CALL SEND_REAL_32BIT( snl(RMIR(3) ) )
CALL SEND_REAL_32BIT( snl(VMIR(1) ) )
CALL SEND_REAL_32BIT( snl(VMIR(2) ) )
CALL SEND_REAL_32BIT( snl(VMIR(3) ) )

CALL RECEIVE_REAL_64BIT( GRT(01,01) )
CALL RECEIVE_REAL_64BIT( GRT(01,02) )
CALL RECEIVE_REAL_64BIT( GRT(01,03) )
CALL RECEIVE_REAL_64BIT( RTIC(01,01) )
CALL RECEIVE_REAL_64BIT( RTIC(01,02) )
CALL RECEIVE_REAL_64BIT( RTIC(01,03) )
CALL RECEIVE_REAL_64BIT( VTIC(01,01) )
CALL RECEIVE_REAL_64BIT( VTIC(01,02) )
CALL RECEIVE_REAL_64BIT( VTIC(01,03) )

```

C----- COMMUNICATE WITH CORVEL -----C

```

CALL SEND_REAL_32BIT( snl(AT(01) ) )
CALL SEND_REAL_32BIT( snl(AT(02) ) )
CALL SEND_REAL_32BIT( snl(AT(03) ) )

```

C----- DAISY CHAIN -----C

```

CALL SEND_REAL_32BIT( SNGL(TI2M(1) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(2) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(3) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(4) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(5) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(6) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(7) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(8) ) )
CALL SEND_REAL_32BIT( SNGL(TI2M(9) ) )

CALL SEND_REAL_32BIT( SNGL(VREL(1) ) )
CALL SEND_REAL_32BIT( SNGL(VREL(2) ) )
CALL SEND_REAL_32BIT( SNGL(VREL(3) ) )
CALL SEND_REAL_32BIT( SNGL(RREL(1) ) )
CALL SEND_REAL_32BIT( SNGL(RREL(2) ) )
CALL SEND_REAL_32BIT( SNGL(RREL(3) ) )

CALL SEND_REAL_32BIT( SNGL(SP) )
CALL SEND_REAL_32BIT( SNGL(SQ) )
CALL SEND_REAL_32BIT( SNGL(SR) )

```

C-----

C----- INERTIAL MEASUREMENT UPDATE -----C

C-----

```

C                               Get inertial measurement data needed   C
C                               for guidance calculations .             C
C                               -----C

```

```

IF ( TSTEP .GE. TIMUDRIV ) THEN

```

```

    TIMUDRIV = TIMUDRIV + TIMUSTEP

```

```

C-----C
C-----IMU PROCESSOR MODULE -----C
C-----C
C                               Convert gyro and accelerometer outputs C
C                               to delta angle and delta velocity         C
C                               -----C

```

```

    CALL IMUPRO(T,PULSESEG,PULSEA,DELPHI,DELTHT,DELPSI,
    .           DELU,DELV,DELW)

```

```

C-----C
C-----NAVIGATION MODULE -----C
C-----C
C                               This module calculates the quaternions C
C                               and transformation matrices using delta C
C                               angles sensed by the gyro and calculates C
C                               the interceptor velocity and position    C
C                               using delta velocity sensed by the       C
C                               accelerometer                               C
C                               -----C

```

```

    CALL NAVIG(T,MASS,DELPHI,DELTHT,DELPSI,DELU,DELV,DELW,GR,
    .           QS1,CIE,SP,SQ,SR,SUD,SVD,SWL,VMIR,RMIR,TI2M,SPHI,STHT,
    .           SPSI,SU,SV,SW,AT,VMI,RMI)

```

```

ENDIF

```

```

C-----C
C-----MIDCOURSE CORRECTION -----C
C-----C
C                               Models uplink of interceptor,          C
C                               target, and intercept conditions         C
C                               -----C

```

```

IF ( ( DABS(T-TUPLK1).LE.DTEPS ) .OR.
*   ( DABS(T-TUPLK2).LE.DTEPS ) ) THEN

```

```

C    REVISE ESTIMATED MISSILE STATES

```

```

    VMI(1)    = XYZED(1)
    VMI(2)    = XYZED(2)
    VMI(3)    = XYZED(3)

```

```

    RMI(1)    = XYZE(1)
    RMI(2)    = XYZE(2)
    RMI(3)    = XYZE(3)

```

```

VMIR(1)  = XD
VMIR(2)  = YD
VMIR(3)  = ZD

```

```

RMIR(1)  = X
RMIR(2)  = Y
RMIR(3)  = Z

```

```

TONAV    = 'T'

```

```

ENDIF

```

```

C-----C
C-----MIDCOURSE CORRECTION-----C
C-----C
C          Models uplink of interceptor,      C
C          target, and intercept conditions    C
C-----C
C-----C

```

```

IF ( ( DABS(T-TUPLK1).LE.DTEPS ) .OR.
*   ( DABS(T-TUPLK2).LE.DTEPS ) ) THEN

```

```

C      REVISE ESTIMATED TARGET STATES

```

```

RTEST(1) = RTIC(1,1)
RTEST(2) = RTIC(1,2)
RTEST(3) = RTIC(1,3)

```

```

VTEST(1) = VTIC(1,1)
VTEST(2) = VTIC(1,2)
VTEST(3) = VTIC(1,3)

```

```

GRTEST(1) = GRT(1,1)
GRTEST(2) = GRT(1,2)
GRTEST(3) = GRT(1,3)

```

```

TL2      = T

```

```

ENDIF

```

```

C-----C
C          ON BOARD GUIDANCE PROCESSING      C
C-----C
C          Determine guidance commands        C
C-----C
C-----C

```

```

IF ( TSTEP .GE. TGPUDRIV ) THEN

```

```

    TGPUDRIV = TGPUDRIV + TGPUSTEP

```

```

C-----C
C-----ON BOARD TARGET MODULE-----C
C-----C
C          Estimate target position based on  C
C          predicted intercept conditions      C
C-----C
C-----C

```

```

C      GRTEST TEMPORARILY EQUAL TO GRT

      GRTEST(1) = GRT(1,1)
      GRTEST(2) = GRT(1,2)
      GRTEST(3) = GRT(1,3)

      CALL OBARG(T,GRTEST,RTEST,VTEST)
      CALL ESTREL2(RTEST,VTEST,RMIR,VMIR,RREL,VREL)

      ENDIF

C-----C
C-----TERMINATION LOGIC-----C
C-----C
C      Defines the simulation termination conditions
C
C-----C
C
C      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

      IEXIT = 0

C      ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
C      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED

      IF ( T.GE.TFINAL ) THEN
        IEXIT = 1
      ENDIF

C      increment time

      TSTEP = TSTEP + 1.0D0
      T = TSTEP * DELT

C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

      IF ( IEXIT.EQ.0 ) GO TO 1000

      END

```

## B.1.3 Up02.for

```

c      PROGRAM EXOSIM
C-----C
C-----C Declare and initialize variables -----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      CHARACTER*128 MESSAGE

C      THE FOLLOWING COMMON BLOCK USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RVCSTR / TREFLV , TLSTV , TMVCS , THVCS , LENVCS

      REAL  TMVCS(6,4) , THVCS(6,4) , DTOFFV(4)
      REAL  CG(3)
      REAL  TOFFLT(4)
      REAL  FOFF1(4) , FOFF2(4)
      REAL  MXVCS , MYVCS , MZVCS
      REAL  MDOTV , latlp, longlp

      INTEGER      LENVCS(4)
      INTEGER      SEKTYP

      REAL  TSTEP, DELT
      REAL  TMSUDRIV, TMSUSTEP

      double precision  d.DTOFFV(4)
      double precision  d.CG(3)
      double precision  d.TOFFLT(4), d.tvtab
      double precision  d.MXVCS , d.MYVCS , d.MZVCS
      double precision  d.FXVCS , d.FYVCS , d.FZVCS
      double precision  d.MDOTV , d.tburnm, d.timonv

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSDATA35.DAT')
$INCLUDE('~/INCLUDE/SSDATA38.DAT')
$INCLUDE('~/INCLUDE/SSDATA39.DAT')
$INCLUDE('~/INCLUDE/SSDATA42.DAT')
$INCLUDE('~/INCLUDE/SSDATA44.DAT')
$INCLUDE('~/INCLUDE/SSDATA45.DAT')
$INCLUDE('~/INCLUDE/SSDATA46.DAT')
$INCLUDE('~/INCLUDE/SSDATA47.DAT')
$INCLUDE('~/INCLUDE/SSDATA48.DAT')
$INCLUDE('~/INCLUDE/SSDATA49.DAT')
$INCLUDE('~/INCLUDE/SSDATA50.DAT')
$INCLUDE('~/INCLUDE/SSDATA01.DAT')
$INCLUDE('~/INCLUDE/SSDATA17.DAT')
$INCLUDE('~/INCLUDE/SSDATA18.DAT')
$INCLUDE('~/INCLUDE/SSDATA21.DAT')
$INCLUDE('~/INCLUDE/SSDATA22.DAT')
$INCLUDE('~/INCLUDE/SSDATA23.DAT')
$INCLUDE('~/INCLUDE/SSDATA28.DAT')
$INCLUDE('~/INCLUDE/SSDATA29.DAT')
$INCLUDE('~/INCLUDE/SSDATA30.DAT')
$INCLUDE('~/INCLUDE/SSDATA71.DAT')
$INCLUDE('~/INCLUDE/SSTIMING.DAT')
$INCLUDE('SSp02.DAT')

* INITIALIZE 80x87
      CALL CW87

```

```

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed   C
C           within this loop                       C
C-----C

1000 CONTINUE

C   WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C----- MISSILE STATE UPDATE MODULE -----C
C-----C
C           Integrate missile states to current time C
C-----C

C----- receive from masspr (P00) -----C

      CALL RECEIVE_REAL_32BIT( cg(01) )
      CALL RECEIVE_REAL_32BIT( cg(02) )
      CALL RECEIVE_REAL_32BIT( cg(03) )

C----- Send variables to masspr and missil (p00) -----C

      CALL send_REAL_32BIT( mdotV )

      CALL send_REAL_32BIT( fxvcs )
      CALL send_REAL_32BIT( fyvcs )
      CALL send_REAL_32BIT( fzvcs )

      CALL send_REAL_32BIT( mxvcs )
      CALL send_REAL_32BIT( myvcs )
      CALL send_REAL_32BIT( mzvcs )

C----- Communication with p01 -----C
      CALL RECEIVE_REAL_32BIT( DTOFFV(01) )
      CALL RECEIVE_REAL_32BIT( DTOFFV(02) )
      CALL RECEIVE_REAL_32BIT( DTOFFV(03) )
      CALL RECEIVE_REAL_32BIT( DTOFFV(04) )
      CALL RECEIVE_SIGNED_16BIT( IVCS )
      CALL RECEIVE_SIGNED_16BIT( IVTAB )
      CALL RECEIVE_REAL_32BIT( TBURNM )
      CALL RECEIVE_REAL_32BIT( TIMONV )
      CALL RECEIVE_REAL_32BIT( TOFFLT(01) )
      CALL RECEIVE_REAL_32BIT( TOFFLT(02) )
      CALL RECEIVE_REAL_32BIT( TOFFLT(03) )
      CALL RECEIVE_REAL_32BIT( TOFFLT(04) )
      CALL RECEIVE_REAL_32BIT( TVTAB )

      IF ( tstep .ge. tmsudriv ) THEN

          tmsudriv = tmsudriv + tmsustep

C-----C
C----- VCS THRUSTER RESPONSE MODULE -----C
C-----C

```

```

C                                     Determines the forces and moments      C
C                                     imparted by the VCS thrusters           C
C                                     -----C
C
C      IF ( T.GE.TKVON ) THEN
C
C          CALL VCSTHR(T,CG,TBURNM,IVCS,TOFFLT,
C                      timonv,DTOFFV,TVTAB,FOFF1,FOFF2,IVTAB,TBRK,
C                      FXVCS,FYVCS,FZVCS,MXVCS,MYVCS,MZVCS,MDOTV)
C
C      ENDIF
C
C  ENDIF
C
C-----C
C-----C----- TERMINATION LOGIC -----C
C-----C
C                                     Defines the simulation termination    C
C                                     conditions                             C
C                                     -----C
C
C      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )
C
C      IEXIT = 0
C
C      increment time
C
C      TSTEP = TSTEP + 1.0e0
C      T = TSTEP * DELT
C
C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
C
C      IF ( IEXIT.EQ.0 ) GO TO 1000
C
C  END

```

## B.1.4 Uup03.for

```

C      PROGRAM EXOSIM
C-----
C----- Declare and initialize variables -----C
C-----

```

```

      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

```

```

C      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

```

```

      COMMON / PTARG / TL1      , GRTLST , FIRST1

```

```

      REAL CER(9) , CIM(9) , Q, R, ZD, YD, XD, PTARG, QTARG, RTARG
      REAL TPHI, TTHT, TPSI, TPHID, TTHTD, TPSID, CIT(9) , CTI(9)

```

```

      DOUBLE PRECISION XINT(50)      , TINT(50)      , XDOTL(50)
      DOUBLE PRECISION SF1G(3)      , SF2G(3)      , DCG(3)
      DOUBLE PRECISION SF1A(3)      , SF2A(3)      , DCA(3)
      DOUBLE PRECISION OMEGA0(3)      , XYZLCH(3)      , EVTIME(20)
      DOUBLE PRECISION MASSL      , MACHL      , ANGACL(3,4,10)
      DOUBLE PRECISION OMEGAI(3)      , GRTLST(5,3)      , CIMO(9)
      DOUBLE PRECISION WBI2(3)      , WBI1(3)      , WBO2(3)
      DOUBLE PRECISION WBO1(3)      , GRLST(3)      , XYZDP(3)
      DOUBLE PRECISION ABI2(3)      , ABI1(3)      , ABO2(3)
      DOUBLE PRECISION ABO1(3)      , GRLAST(3)      , GRTPST(3)
      DOUBLE PRECISION TLATCH(10)      , LAMMSV(2,10)      , RRELSV(3,10)
      DOUBLE PRECISION VRELSV(3,10)      , TI2MSV(9,10)      , SNRSV(10)
      DOUBLE PRECISION AACCEL(3,4)

```

```

      DOUBLE PRECISION CGX(20)      , CGY(20)      , CGZ(20)
      DOUBLE PRECISION MASST1(20)      , MASST2(20)

```

```

      REAL      RANSEQ(97)      , RANLST      , RAND1(98)

```

```

      INTEGER      IEVFLG(20)      , FIRST1      , FIRST2
      INTEGER      ISEQ(4)      , IMCPAS(3,4)      , FLIP
      INTEGER      VCOD(4)      , GATE

```

```

C      OUTPUTS

```

```

      DOUBLE PRECISION MXA      , MYA      , MZA
      DOUBLE PRECISION MXT      , MYI      , MZT
      DOUBLE PRECISION MRCX      , MRCY      , MRCZ
      DOUBLE PRECISION MXVCS      , MYVCS      , MZVCS
      DOUBLE PRECISION MXACS      , MYACS      , MZACS
      DOUBLE PRECISION MX      , MY      , MZ
      DOUBLE PRECISION MACH      , MDOTT      , MDOTF
      DOUBLE PRECISION MDOTV      , MDOTA      , LFRACS
      DOUBLE PRECISION KN      , KM      , MDLTFR
      DOUBLE PRECISION KTHT      , KTHTD      , MDELTA
      DOUBLE PRECISION KNE      , KME      , MALPHA
      DOUBLE PRECISION LATT      , LONGT      , MVS
      DOUBLE PRECISION KA      , KV      , MAGRTR
      DOUBLE PRECISION MAGLOS      , MGRDTR      , MAGR
      DOUBLE PRECISION MAGV      , MGRDOT      , MXYZDD
      DOUBLE PRECISION MGR      , LAT      , LONG
      DOUBLE PRECISION MISS      , MVR      , MVRWM
      DOUBLE PRECISION ATHRF(4)      , CMMD(2)      , VCMDL(4)
      DOUBLE PRECISION PTER(3)      , CMS(9)

```

|                  |            |                 |              |
|------------------|------------|-----------------|--------------|
| DOUBLE PRECISION | GRT (5, 3) | , ADISTT (4, 3) |              |
| DOUBLE PRECISION | RTAR (3)   | , VTAR (3)      | , UVS (3)    |
| DOUBLE PRECISION | VC (3)     | , DLV (3)       | , VTTP (3)   |
| DOUBLE PRECISION | VS (3)     | , US (3)        | , AC (3)     |
| DOUBLE PRECISION | CIR (9)    | , CMI (9)       |              |
| DOUBLE PRECISION | VW (3)     | , WC (3)        | , PM (3)     |
| DOUBLE PRECISION | CGEST (3)  | , RRELTR (3)    | , VRELTP (3) |
| DOUBLE PRECISION | LAMTRU (2) | , LAMDXX (2)    | , LAMDTR (2) |
| DOUBLE PRECISION | LAMSEK (2) | , LAMDSK (2)    | , LAMM (2)   |
| DOUBLE PRECISION | RTEST (3)  | , RREL (3)      | , URREL (3)  |
| DOUBLE PRECISION | VREL (3)   | , TI2M (9)      | , USI (3)    |
| DOUBLE PRECISION | QS1 (4)    | , VMI (3)       | , RMI (3)    |
| DOUBLE PRECISION | VMIR (3)   | , RMIR (3)      | , VTEST (3)  |
| DOUBLE PRECISION | AT (3)     | , XYZR (3)      | , GB (3)     |
| DOUBLE PRECISION | GR (3)     | , CRI (9)       |              |
| DOUBLE PRECISION | VRWM (3)   | , CEI (9)       | , CIE (9)    |
| DOUBLE PRECISION | PG (3)     | , VTT (3)       | , US0 (3)    |
| DOUBLE PRECISION | PG0 (3)    | , USF (3)       | , QUATIC (4) |
| DOUBLE PRECISION | TI2MO (9)  | , GREST (3)     |              |
| DOUBLE PRECISION | LAMMO (2)  | , RRELO (3)     | , VRELO (3)  |
| DOUBLE PRECISION | RRELM (3)  | , VRELM (3)     | , GRTEST (3) |
| DOUBLE PRECISION | FOFF1 (4)  | , FOFF2 (4)     |              |

## C NAMELIST INPUTS

|                  |               |                 |                |
|------------------|---------------|-----------------|----------------|
| DOUBLE PRECISION | IXX           | , IYY           | , IZZ          |
| DOUBLE PRECISION | CG (3)        | , MASS          | , PQR (3)      |
| DOUBLE PRECISION | IMPULS        | , QUAT (4)      | , MDOT         |
| DOUBLE PRECISION | QUATD (4)     | , BOFF2 (2)     | , TMVCS (6, 4) |
| DOUBLE PRECISION | THVCS (6, 4)  | , DTOFFV (4)    | , VG (3)       |
| DOUBLE PRECISION | TMACSA (8, 4) | , THACSA (8, 4) | , DTACSA (4)   |
| DOUBLE PRECISION | TMACSB (8, 4) | , THACSB (8, 4) | , DTACSB (4)   |
| DOUBLE PRECISION | XYZE (3)      | , XYZED (3)     | , RTIC (5, 3)  |
| DOUBLE PRECISION | VTIC (5, 3)   | , PULSEG (3)    | , QFRACG (3)   |
| DOUBLE PRECISION | PULSEA (3)    | , QFRACA (3)    | , XYZEDD (3)   |
| DOUBLE PRECISION | LAM (2)       | , LAMD (2)      | , VGM (3)      |
| DOUBLE PRECISION | DTVCSY (3)    | , DTVCSY (3)    | , FLTC (4)     |
| DOUBLE PRECISION | TOFFLT (4)    | , TMF (8, 4)    | , THF (8, 4)   |
| DOUBLE PRECISION | DTOFF (4)     | , VWIC (3)      | , AOFF1 (4)    |
| DOUBLE PRECISION | VTTIC (3)     | , USD (3)       | , VCMD (4)     |
| DOUBLE PRECISION | PGD (3)       | , VWD (3)       | , MASS0        |
| DOUBLE PRECISION | MSSTG2        | , LATLP         | , LONGLP       |
| DOUBLE PRECISION | IMPLS0        | , MVRDOT        |                |
| DOUBLE PRECISION | RJ (5)        |                 |                |
| DOUBLE PRECISION | AOFF2 (4)     |                 |                |

|         |            |            |            |
|---------|------------|------------|------------|
| INTEGER | LENVCS (4) | , LENA (4) | , LENB (4) |
| INTEGER | LENF (4)   | , GYSEED   | , FRMCNT   |
| INTEGER | SKSEED     | , SEKTYP   | , ACQD     |
| INTEGER | TERM       | , TOSEED   | , VLVCMS   |
| INTEGER | ESTATE     |            |            |
| INTEGER | TRACK      |            |            |
| INTEGER | ROWBEG     | , COLBEG   | , PLOTNO   |

DOUBLE PRECISION TSTEP, DELT

DOUBLE PRECISION TMSUDRIV, TTSUDRIV, TRSUDRIV, TIMUDRIV,

\* TGPUDRIV, TAPUDRIV, TSPUDRIV, TKFUDRIV

DOUBLE PRECISION TMSUSTEP, TTSUSTEP, TRSUSTEP, TIMUSTEP,

\* TGPUSTEP, TAPUSTEP, TSPUSTEP, TKFUSTEP

Integer first, jrst, krst

## \* DATA INITIALIZATION

\$INCLUDE ('~/INCLUDE/SSDATA35.DAT')

\$INCLUDE ('~/INCLUDE/SSDATA38.DAT')

```

$INCLUDE('^/INCLUDE/SSDATA39.DAT')
$INCLUDE('^/INCLUDE/SSDATA42.DAT')
$INCLUDE('^/INCLUDE/SSDATA44.DAT')
$INCLUDE('^/INCLUDE/SSDATA45.DAT')
$INCLUDE('^/INCLUDE/SSDATA46.DAT')
$INCLUDE('^/INCLUDE/SSDATA47.DAT')
$INCLUDE('^/INCLUDE/SSDATA48.DAT')
$INCLUDE('^/INCLUDE/SSDATA49.DAT')
$INCLUDE('^/INCLUDE/SSDATA50.DAT')
$INCLUDE('^/INCLUDE/SSDATA01.DAT')
$INCLUDE('^/INCLUDE/SSDATA17.DAT')
$INCLUDE('^/INCLUDE/SSDATA18.DAT')
$INCLUDE('^/INCLUDE/SSDATA21.DAT')
$INCLUDE('^/INCLUDE/SSDATA22.DAT')
$INCLUDE('^/INCLUDE/SSDATA23.DAT')
$INCLUDE('^/INCLUDE/SSDATA28.DAT')
$INCLUDE('^/INCLUDE/SSDATA29.DAT')
$INCLUDE('^/INCLUDE/SSDATA30.DAT')
$INCLUDE('^/INCLUDE/SSDATA71.DAT')
$INCLUDE('^/INCLUDE/SSTIMING.DAT')

```

```

$INCLUDE('^/INCLUDE/SSMAS_cg.DAT')

```

```

      DATA IMASS , IAERC , IBTHR , IBAUTO / 1 , 1 , 1 , 1 /

```

```

* INITIALIZE 80x87
  CALL CW87

```

```

$INCLUDE('SSp03.DAT')

```

```

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed   C
C           within this loop                       C
C-----C

```

```

1000 CONTINUE

```

```

C-----C
C----- Processor communication -----C
C-----C

```

```

C----- Communicate with p00 -----C

```

```

  CALL RECEIVE_REAL_32BIT( Q )
  CALL RECEIVE_REAL_32BIT( R )
  CALL RECEIVE_REAL_64BIT( X )
  CALL RECEIVE_REAL_64BIT( Y )
  CALL RECEIVE_REAL_64BIT( Z )
  CALL RECEIVE_REAL_32BIT( XD )
  CALL RECEIVE_REAL_32BIT( YD )
  CALL RECEIVE_REAL_32BIT( ZD )
  CALL RECEIVE_REAL_32BIT( CIM(1) )
  CALL RECEIVE_REAL_32BIT( CIM(2) )
  CALL RECEIVE_REAL_32BIT( CIM(3) )
  CALL RECEIVE_REAL_32BIT( CIM(4) )
  CALL RECEIVE_REAL_32BIT( CIM(5) )
  CALL RECEIVE_REAL_32BIT( CIM(6) )
  CALL RECEIVE_REAL_32BIT( CIM(7) )
  CALL RECEIVE_REAL_32BIT( CIM(8) )

```

```

CALL RECEIVE_REAL_32BIT( CIM(9) )

C----- Communicate with p01 -----C

CALL SEND_REAL_64BIT( GRT(01,01) )
CALL SEND_REAL_64BIT( GRT(01,02) )
CALL SEND_REAL_64BIT( GRT(01,03) )
CALL SEND_SIGNED_16BIT( IRESLV )
CALL SEND_REAL_32BIT( SNGL(LAMDXX(01)) )
CALL SEND_REAL_32BIT( SNGL(LAMDXX(02)) )
CALL SEND_REAL_32BIT( SNGL(LAMSEK(01)) )
CALL SEND_REAL_32BIT( SNGL(LAMSEK(02)) )
CALL SEND_REAL_32BIT( SNGL(MAGRTR) )
CALL SEND_REAL_64BIT( RTIC(01,01) )
CALL SEND_REAL_64BIT( RTIC(01,02) )
CALL SEND_REAL_64BIT( RTIC(01,03) )
CALL SEND_REAL_64BIT( VTIC(01,01) )
CALL SEND_REAL_64BIT( VTIC(01,02) )
CALL SEND_REAL_64BIT( VTIC(01,03) )

C   WRITE(*,*) '-----BEGINNING OF LOOP-----'

   IF ( tstep .ge. tmsudriv ) THEN

       tmsudriv = tmsudriv + tmsustep

C   ROTATING EARTH MODEL

       CALL SPMMK(0.0E0,1,0.0E0,2,SNGL(OMEGAE*T),3,CER)
   ENDIF

C-----C
C----- RELATIVE STATES MODULE -----C
C-----C
C   Calculate relative range, range rate, C
C   time-to-go, LOS angles and rates C
C   C
C-----C

   IF ( TSTEP .GE. TRSUDRIV ) THEN

       TRSUDRIV = TRSUDRIV + TRSUSTEP

       CALL RELAT(RTIC,VTIC,X,Y,Z,XD,YD,ZD,Q,R,CIM,CMS,RRELTR,
.           MAGRTR,VRELTR,MGRDTR,MAGLOS,LAMTRU,LAMDXX,
.           LAMDTR,LAMSEK,LAMDSK,TGOTR,RRELM,VRELM)

C   EXTRAPOLATE POINT OF CLOSEST APPROACH

       XMISS = RRELTR(1) + TGOTR*VRELTR(1)
       YMISS = RRELTR(2) + TGOTR*VRELTR(2)
       ZMISS = RRELTR(3) + TGOTR*VRELTR(3)

       MISS = DSQRT ( XMISS**2 + YMISS**2 + ZMISS**2 )

   ENDIF

C-----C
C----- TARGET STATES MODULE -----C
C-----C
C   This module calculates the true exo- C
C   atmospheric trajectory data for C

```

```

C                                     the target                                     C
C                                                                                   C
C-----C

```

```

IF ( TSTEP .GE. TTSUDRIV ) THEN

```

```

    TTSUDRIV = TTSUDRIV + TTSUSTEP

```

```

    CALL TARGET( T,MAGRTR,CER,CIE,PTARG,QTARG,RTARG,
.               TPHI,TTHT,TPSI,GRT,TPHID,TTHTD,TPSID,CIT,RTIC,VTIC,
.               RTAR,RTER,IRESLV,RJ,CTI,VTAR,LATT,LONGT )

```

```

ENDIF

```

```

C-----C
C-----C----- TERMINATION LOGIC -----C
C-----C
C               Defines the simulation termination   C
C               conditions                             C
C-----C

```

```

C    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

```

```

    IEXIT = 0

```

```

C    ENABLE EXIT IF INTERCEPT HAS OCCURRED AND ALL EVENTS SCHEDULED FOR
C    THIS TIME HAVE BEEN EXECUTED

```

```

    IF ( TGOTR.LE.TGOMN ) THEN

```

```

        IEXIT = 1

```

```

    ENDIF

```

```

C    increment time

```

```

    TSTEP = TSTEP + 1.0D0

```

```

    T = TSTEP * DELT

```

```

C    CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

```

```

    IF ( IEXIT.EQ.0 ) GO TO 1000

```

```

C-----C
C-----C----- POINT OF CLOSEST APPROACH CALCULATION -----C
C-----C
C               Determines the miss distance at the   C
C               point of closest approach               C
C-----C

```

```

    MISS = DSQRT ( (RRELTR(1) + VRELTR(1)*TGOTR)**2
.               + (RRELTR(2) + VRELTR(2)*TGOTR)**2
.               + (RRELTR(3) + VRELTR(3)*TGOTR)**2 )

```

```

    WRITE(MESSAGE,889) T,MISS

```

```

    CALL OUTMES(MESSAGE)

```

```

889 FORMAT(1X,E16.9,' MISS = ',E16.9)

```

END

## B.1.5 Uup04.for

```

C      PROGRAM EXOSIM
C-----
C----- Declare and initialize variables -----C
C-----

      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / NORCOM / GSET , ISET
      COMMON / RANCOM / RANSEQ , RANLST
      COMMON / RGYRO / PSIG , THTG , PHIG , THXZG , THXYG ,
      .           THYZG , THYXG , THZYG , THZXG , SF1G ,
      .           SF2G , DCG , T0GYRO , CIMO , WBI2 ,
      .           WBI1 , WBO2 , WBO1 , DRSIGG

      REAL S_CIM(9)
      REAL S_P,S_Q,S_R
      DOUBLE PRECISION SF1G(3) , SF2G(3) , DCG(3)
      REAL RANSEQ(97) , RANLST
      DOUBLE PRECISION WBI2(3) , WBI1(3) , WBO2(3)
      DOUBLE PRECISION WBO1(3)
      DOUBLE PRECISION CIM(9)
      DOUBLE PRECISION PULSEG(3)
      DOUBLE PRECISION QFRACG(3)
      DOUBLE PRECISION LONGLP, LATLP, CIMO(9)

      INTEGER*4 GYSEED
      INTEGER SEKTYP

      DOUBLE PRECISION TSTEP,DELT
      DOUBLE PRECISION TIMUDRIV,TIMUSTEP

* DATA INITIALIZATION

$include ('~/include/ssdata35.dat')
$include ('~/include/ssdata38.dat')
$include ('~/include/ssdata39.dat')
$include ('~/include/ssdata42.dat')
$include ('~/include/ssdata44.dat')
$include ('~/include/ssdata45.dat')
$include ('~/include/ssdata46.dat')
$include ('~/include/ssdata47.dat')
$include ('~/include/ssdata48.dat')
$include ('~/include/ssdata49.dat')
$include ('~/include/ssdata50.dat')
$include ('~/include/ssdata01.dat')
$include ('~/include/ssdata17.dat')
$include ('~/include/ssdata18.dat')
$include ('~/include/ssdata21.dat')
$include ('~/include/ssdata22.dat')
$include ('~/include/ssdata23.dat')
$include ('~/include/ssdata28.dat')
$include ('~/include/ssdata29.dat')
$include ('~/include/ssdata30.dat')
$include ('~/include/ssdata71.dat')
$include ('~/include/sstiming.dat')

```

```

call cw87

$include ('ssp04.dat')

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed   C
C           within this loop                       C
C-----C

1000 CONTINUE

C   WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C----- Processor communication -----C
C-----C
C----- Communicate with p01 -----C

CALL SEND_REAL_32BIT( SNGL(PULSESEG(01)) )
CALL SEND_REAL_32BIT( SNGL(PULSESEG(02)) )
CALL SEND_REAL_32BIT( SNGL(PULSESEG(03)) )

CALL RECEIVE_REAL_32BIT( S_P )
CALL RECEIVE_REAL_32BIT( S_Q )
CALL RECEIVE_REAL_32BIT( S_R )
P = DBLE(S_P)
Q = DBLE(S_Q)
R = DBLE(S_R)

CALL RECEIVE_REAL_32BIT( S_CIM(1) )
CALL RECEIVE_REAL_32BIT( S_CIM(2) )
CALL RECEIVE_REAL_32BIT( S_CIM(3) )
CALL RECEIVE_REAL_32BIT( S_CIM(4) )
CALL RECEIVE_REAL_32BIT( S_CIM(5) )
CALL RECEIVE_REAL_32BIT( S_CIM(6) )
CALL RECEIVE_REAL_32BIT( S_CIM(7) )
CALL RECEIVE_REAL_32BIT( S_CIM(8) )
CALL RECEIVE_REAL_32BIT( S_CIM(9) )
CIM(1) = DBLE(S_CIM(1))
CIM(2) = DBLE(S_CIM(2))
CIM(3) = DBLE(S_CIM(3))
CIM(4) = DBLE(S_CIM(4))
CIM(5) = DBLE(S_CIM(5))
CIM(6) = DBLE(S_CIM(6))
CIM(7) = DBLE(S_CIM(7))
CIM(8) = DBLE(S_CIM(8))
CIM(9) = DBLE(S_CIM(9))

C-----C
C----- INERTIAL MEASUREMENT UPDATE -----C
C-----C
C           Get inertial measurement data needed   C
C           for guidance calculations .             C
C-----C

```

```

      IF ( TSTEP .GE. TIMUDRIV ) THEN
        TIMUDRIV = TIMUDRIV + TIMUSTEP

C-----C
C-----GYRO MODULE -----C
C-----C
C          Determine sensed body rates .      C
C-----C
C-----C

      CALL GYRO(T,P,Q,R,CIM,GYSEED,QFRACG,PULSEG)

      ENDIF

C-----C
C-----TERMINATION LOGIC -----C
C-----C
C          Defines the simulation termination  C
C          conditions                          C
C-----C

C      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )
      IEXIT = 0

C      increment time

      TSTEP = TSTEP + 1.0d0
      T = TSTEP * DELT

C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

      IF ( IEXIT.EQ.0 ) GO TO 1000

      END

```

## B.1.6 Uup05.for

```

C      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      CHARACTER*128 MESSAGE

C      THE FOLLOWING COMMON BLOCK USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / NORCOM / GSET , ISET
      COMMON / RANCOM / RANSEQ , RANLST
      COMMON / RACSTR / TREFLA , TLSTC , ACSF , AOFF1 , AOFF2 ,
      .               TMACSA , THACSA , LENA , TMACSB , THACSB ,
      .               LENB

      REAL CG(3)
      REAL MXACS , MYACS , MZACS , MDOTA
      REAL RANSEQ(97) , RANLST, AOFF2(4) , AOFF1(4)
      REAL TMACSA(8,4) , THACSA(8,4) , DTACSA(4)
      REAL TMACSB(8,4) , THACSB(8,4) , DTACSB(4)

      double precision d_cg(3)
      double precision d_MXACS, d_MYACS, d_MZACS, d_MDOTA
      double precision d_fxacs,d_fyacs,d_fzacs
      double precision d_acslev,d_dtacsa(4),d_dtacsb(4)
      double precision d_tatab

      INTEGER      LENA(4) , LENB(4)
      INTEGER      SEKTYP
      INTEGER*4     TOSEED

      REAL TSTEP,DELT,latlp,longlp
      REAL TMSUDRIV, TMSUSTEP

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSDATA35.DAT')
$INCLUDE('^/INCLUDE/SSDATA38.DAT')
$INCLUDE('^/INCLUDE/SSDATA39.DAT')
$INCLUDE('^/INCLUDE/SSDATA42.DAT')
$INCLUDE('^/INCLUDE/SSDATA44.DAT')
$INCLUDE('^/INCLUDE/SSDATA45.DAT')
$INCLUDE('^/INCLUDE/SSDATA46.DAT')
$INCLUDE('^/INCLUDE/SSDATA47.DAT')
$INCLUDE('^/INCLUDE/SSDATA48.DAT')
$INCLUDE('^/INCLUDE/SSDATA49.DAT')
$INCLUDE('^/INCLUDE/SSDATA50.DAT')
$INCLUDE('^/INCLUDE/SSDATA01.DAT')
$INCLUDE('^/INCLUDE/SSDATA17.DAT')
$INCLUDE('^/INCLUDE/SSDATA18.DAT')
$INCLUDE('^/INCLUDE/SSDATA21.DAT')
$INCLUDE('^/INCLUDE/SSDATA22.DAT')
$INCLUDE('^/INCLUDE/SSDATA23.DAT')
$INCLUDE('^/INCLUDE/SSDATA28.DAT')
$INCLUDE('^/INCLUDE/SSDATA29.DAT')
$INCLUDE('^/INCLUDE/SSDATA30.DAT')
$INCLUDE('^/INCLUDE/SSDATA71.DAT')
$INCLUDE('^/INCLUDE/SSTIMING.DAT')

```

```
* INITIALIZE 80x87
  CALL CW87
```

```
C      DETERMINE IF MIDFLIGHT RESTART
$INCLUDE('SSp05.DAT')
```

```
C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C      Execution of all events is performed C
C      within this loop C
C C
C-----C
```

```
1000 CONTINUE
```

```
C      WRITE(*,*) '-----BEGINNING OF LOOP-----'
```

```
C-----C
C----- MISSILE STATE UPDATE MODULE -----C
C-----C
C      Integrate missile states to current time C
C C
C-----C
```

```
C----- recieve from masspr (P00) -----C
```

```
CALL RECEIVE_REAL_32BIT( cg(01) )
CALL RECEIVE_REAL_32BIT( cg(02) )
CALL RECEIVE_REAL_32BIT( cg(03) )
```

```
C----- Send variables to masspr and missil (p00) -----C
```

```
CALL send_REAL_32BIT( mdota )

CALL send_REAL_32BIT( fxacs )
CALL send_REAL_32BIT( fyacs )
CALL send_REAL_32BIT( fzacs )

CALL send_REAL_32BIT( mxacs )
CALL send_REAL_32BIT( myacs )
CALL send_REAL_32BIT( mzacs )
```

```
C----- Communication with p01 -----C
```

```
CALL RECEIVE_REAL_32BIT( ACSLEV )
CALL RECEIVE_REAL_32BIT( DTACSA(01) )
CALL RECEIVE_REAL_32BIT( DTACSA(02) )
CALL RECEIVE_REAL_32BIT( DTACSA(03) )
CALL RECEIVE_REAL_32BIT( DTACSA(04) )
CALL RECEIVE_REAL_32BIT( DTACSB(01) )
CALL RECEIVE_REAL_32BIT( DTACSB(02) )
CALL RECEIVE_REAL_32BIT( DTACSB(03) )
CALL RECEIVE_REAL_32BIT( DTACSB(04) )
CALL RECEIVE_SIGNED_16BIT( ITHRES )
CALL RECEIVE_REAL_32BIT( TATAB )
```

```
CALL SEND_SIGNED_16BIT( IACSON )
```

```
C-----
```

```

      IF ( tstep .ge. tmsudriv ) THEN
        tmsudriv = tmsudriv + tmsustep

        IF ( T.GE.TKVON ) THEN

C-----C
C----- ACS THRUSTER RESPONSE MODULE -----C
C-----C
C          Determines the forces and moments      C
C          imparted by the ACS thrusters           C
C-----C

          CALL ACSTHR(T,CG,ACSLEV,DTACSA,DTACSB,TATAB,TOSEED,
.            tbrk,ITHRES,FXACS,FYACS,FZACS,MXACS,MYACS,MZACS,
.            MDOTA,IACSON,TIMONA)

          ENDIF
        ENDIF

C-----C
C----- TERMINATION LOGIC -----C
C-----C
C          Defines the simulation termination      C
C          conditions                             C
C-----C

C    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )
      IEXIT = 0

C    increment time
      TSTEP = TSTEP + 1.0e0
      T = TSTEP * DELT

C    CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
      IF ( IEXIT.EQ.0 ) GO TO 1000
    END

```

## B.1.7 Uup06.for

```

C      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      CHARACTER*128 MESSAGE

      real  CGX(20)      , CGY(20)      , CGZ(20)
      real  MASST1(20)

      real  CG(3)        , MASS
      INTEGER      iexit      , icg

      real  TSTEP,DELT
      real  TMSUDRIV,TMSUSTEP

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSMAS_cg.DAT')

* INITIALIZE 80x87
  CALL CW87

C      RESTARTING FROM MIDFLIGHT DATA FILE
$INCLUDE('SSp06.DAT')

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C      Execution of all events is performed
C      within this loop
C-----C

      1000 CONTINUE

C      WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C----- MISSILE STATE UPDATE MODULE -----C
C-----C
C      Integrate missile states to current time
C-----C

      IF ( tstep .ge. tmsudriv ) THEN
        tmsudriv = tmsudriv + tmsustep
        dt      = tmsustep * delt

C-----C
C----- MASS PROPERTIES MODULE -----C
C-----C
C      Update cg
C-----C

```

```

C      CALCULATE MISSILE CENTER OF GRAVITY COMPONENTS

          CALL spTABLE(MASST1,CGX,MASS,CG(1),20,ICG)
          CALL spTABLE(MASST1,CGY,MASS,CG(2),20,ICG)
          CALL spTABLE(MASST1,CGZ,MASS,CG(3),20,ICG)

      ENDIF

C-----
C----- Processor communication -----C
C-----

c----- communication with missil model

      call receive_real_32bit( mass )

C----- send to ACSTHR and VCSTHR and ACCEL

      CALL send_REAL_32BIT( cg(01) )
      CALL send_REAL_32BIT( cg(02) )
      CALL send_REAL_32BIT( cg(03) )

C-----
C----- OUTPUT MODULE -----C
C-----
C      Creates print and plot output data  C
C      files                               C
C                                           C
C-----

C      if ( mod(idnint(tstep),idnint(dtpert)).eq.0 ) then
C      ENDIF

C-----
C----- TERMINATION LOGIC -----C
C-----
C      Defines the simulation termination  C
C      conditions                         C
C                                           C
C-----

C      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

      IEXIT = 0

C      increment time

      TSTEP = TSTEP + 1.0
      T = TSTEP * DELT

C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

      IF ( IEXIT.EQ.0 ) GO TO 1000

      END

```

## B.1.8 Uup07.for

```

C      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      CHARACTER*128 MESSAGE

      REAL TSTEP, DELT
      REAL TIMUDRIV, TIMUSTEP, TGPUDRIV, TGPUSTEP

      REAL MVR           , MVS           , VTP (3) , AT (3)
      REAL VS (3)        , VTT (3)        , VG (3) , US (3)
      real vc (3)         , uvs (3)        , dlV (3)
      REAL X, Y, Z, XD, YD, ZD
      REAL RMIR (3), VMIR (3)

      REAL LONGLP, LATLP
      INTEGER SEKTP

* DATA INITIALIZATION
$INCLUDE ('^/INCLUDE/SSDATA35.DAT')
$INCLUDE ('^/INCLUDE/SSDATA38.DAT')
$INCLUDE ('^/INCLUDE/SSDATA39.DAT')
$INCLUDE ('^/INCLUDE/SSDATA42.DAT')
$INCLUDE ('^/INCLUDE/SSDATA44.DAT')
$INCLUDE ('^/INCLUDE/SSDATA45.DAT')
$INCLUDE ('^/INCLUDE/SSDATA46.DAT')
$INCLUDE ('^/INCLUDE/SSDATA47.DAT')
$INCLUDE ('^/INCLUDE/SSDATA48.DAT')
$INCLUDE ('^/INCLUDE/SSDATA49.DAT')
$INCLUDE ('^/INCLUDE/SSDATA50.DAT')
$INCLUDE ('^/INCLUDE/SSDATA01.DAT')
$INCLUDE ('^/INCLUDE/SSDATA17.DAT')
$INCLUDE ('^/INCLUDE/SSDATA18.DAT')
$INCLUDE ('^/INCLUDE/SSDATA21.DAT')
$INCLUDE ('^/INCLUDE/SSDATA22.DAT')
$INCLUDE ('^/INCLUDE/SSDATA23.DAT')
$INCLUDE ('^/INCLUDE/SSDATA28.DAT')
$INCLUDE ('^/INCLUDE/SSDATA29.DAT')
$INCLUDE ('^/INCLUDE/SSDATA30.DAT')
$INCLUDE ('^/INCLUDE/SSDATA71.DAT')
$INCLUDE ('^/INCLUDE/SSTIMING.DAT')

* INITIALIZE 80x87
  CALL CW87

$INCLUDE ('SSp07.DAT')

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C      Execution of all events is performed  C
C      within this loop                      C
C-----C

```

```

C      CALL INITIALIZE_TIMING()

1000 CONTINUE

C      CALL START_TIMING(0)

C      WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C-----C      Processor communication -----C
C-----C

C      CALL SWITCH_TIMING()

C-----C      COMMUNICATION WITH P00 -----C

      CALL RECEIVE_REAL_32BIT( X )
      CALL RECEIVE_REAL_32BIT( Y )
      CALL RECEIVE_REAL_32BIT( Z )
      CALL RECEIVE_REAL_32BIT( XD )
      CALL RECEIVE_REAL_32BIT( YD )
      CALL RECEIVE_REAL_32BIT( ZD )

      CALL RECEIVE_REAL_32BIT( RMIR(1) )
      CALL RECEIVE_REAL_32BIT( RMIR(2) )
      CALL RECEIVE_REAL_32BIT( RMIR(3) )
      CALL RECEIVE_REAL_32BIT( VMIR(1) )
      CALL RECEIVE_REAL_32BIT( VMIR(2) )
      CALL RECEIVE_REAL_32BIT( VMIR(3) )

C-----C      COMMUNICATION WITH P01 -----C

      CALL RECEIVE_REAL_32BIT( at(1) )
      CALL RECEIVE_REAL_32BIT( at(2) )
      CALL RECEIVE_REAL_32BIT( at(3) )
      CALL SEND_REAL_32BIT( VG(1) )
      CALL SEND_REAL_32BIT( VG(2) )
      CALL SEND_REAL_32BIT( VG(3) )

C      CALL SWITCH_TIMING()

C-----C
C-----C      INERTIAL MEASUREMENT UPDATE -----C
C-----C
C      Get inertial measurement data needed      C
C      for guidance calculations .                  C
C-----C

      IF ( TSTEP .GE. TIMUDRIV ) THEN

          TIMUDRIV = TIMUDRIV + TIMUSTEP

C      TIME SINCE LAST INERTIAL MEASUREMENT UPDATE

          DT      = TIMUSTEP * DELT

C      INTEGRATE GRAVITY COMPENSATED ACCELERATION

          VTT(1) = VTT(1) + DT*AT(1)
          VTT(2) = VTT(2) + DT*AT(2)
          VTT(3) = VTT(3) + DT*AT(3)

```

```

ENDIF

C-----C
C-----MIDCOURSE CORRECTION -----C
C-----C
C          Models uplink of interceptor,      C
C          target, and intercept conditions    C
C-----C

      IF ( ( ABS(T-TUPLK1).LE.DTEPS ) .OR.
*        ( ABS(T-TUPLK2).LE.DTEPS ) ) THEN

C      REVISE ESTIMATED MISSILE STATES

          VMIR(1)  = XD
          VMIR(2)  = YD
          VMIR(3)  = ZD

          RMIR(1)  = X
          RMIR(2)  = Y
          RMIR(3)  = Z

      ENDIF

C-----C
C-----ON BOARD GUIDANCE PROCESSING -----C
C-----C
C          Determine guidance commands          C
C-----C

      IF ( TSTEP .GE. TGPUDRIV ) THEN

          TGPUDRIV = TGPUDRIV + TGPUSTEP

C-----C
C-----CORRELATED VELOCITY MODULE -----C
C-----C
C          This section calculates the correlated C
C          velocity vector (VC) through an iter- C
C          ative process. From VC, the steering C
C          velocity vector is produced by sub- C
C          tracting a bias velocity (VD0) from the C
C          velocity to be gained (VG).          C
C-----C

      IF ( T.GE.TCORV .AND. T.LE.(TTF-DTSPVC) ) THEN

          CALL CORVEL(T,MVR,VTT,RMIR,VMIR,VTP,VG,VS,MVS,UVS,VC,
              DLV,TFFE,TTFE)

          DTMP1 = DTCVU * ANINT ( (T+DTCVU) / DTCVU )
          TCORV = DTMP1
      ENDIF

```

```

ENDIF

C-----C
C----- OUTPUT MODULE -----C
C-----C

c      call stop_timing()

c      if ( mod(int(tstep),int(dtprt)).eq.0 ) then
c          call output_timing()
c          call INITIALIZE_TIMING()
c      ENDIF

C-----C
C----- TERMINATION LOGIC -----C
C-----C
C          Defines the simulation termination
C          conditions
C-----C

C      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

      IEXIT = 0

C      ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
C      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED

      IF ( T.GE.TFINAL ) THEN
          IEXIT = 1
      ENDIF

C      increment time

      TSTEP = TSTEP + 1.0D0
      T = TSTEP * DELT

C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

      IF ( IEXIT.EQ.0 ) GO TO 1000

END

```

## B.1.9 Uup08.for

```

C      PROGRAM EXOSIM
C-----
C----- Declare and initialize variables -----
C-----
C-----

      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / STORAG / XINT      , TINT      , XDOTL
      COMMON / RMISSL / XYZLCH

      REAL S_MASS,S_CIM(9),S_FXACS,S_FYACS,S_FZACS
      REAL S_FXVCS,S_FYVCS,S_FZVCS

      DOUBLE PRECISION XINT(50)      , TINT(50)      , XDOTL(50)
      DOUBLE PRECISION SF1G(3)      , SF2G(3)      , DCG(3)
      DOUBLE PRECISION SF1A(3)      , SF2A(3)      , DCA(3)
      DOUBLE PRECISION OMEGA0(3)    , XYZLCH(3)    , EVTIME(20)
      DOUBLE PRECISION MASSL      , MACHL      , ANGACL(3,4,10)
      DOUBLE PRECISION OMEGAI(3)    , GRTLST(5,3)   , CIMO(9)
      DOUBLE PRECISION WBI2(3)      , WBI1(3)      , WBO2(3)
      DOUBLE PRECISION WBO1(3)      , GRLST(3)      , XYZDP(3)
      DOUBLE PRECISION ABI2(3)      , ABI1(3)      , ABO2(3)
      DOUBLE PRECISION ABO1(3)      , GRLAST(3)     , GRTPST(3)
      DOUBLE PRECISION TLATCH(10)   , LAMMSV(2,10)  , RRELSV(3,10)
      DOUBLE PRECISION VRELSV(3,10) , TI2MSV(9,10)  , SNRSV(10)
      DOUBLE PRECISION AACCEL(3,4)

      DOUBLE PRECISION CGX(20)      , CGY(20)      , CGZ(20)
      DOUBLE PRECISION MASST1(20)   , MASST2(20)

      REAL RANSEQ(97)      , RANLST      , RAND1(98)

      INTEGER IEVFLG(20)      , FIRST1      , FIRST2
      INTEGER ISEQ(4)      , IMCPAS(3,4)      , FLIP
      INTEGER VCOD(4)      , GATE

C      OUTPUTS

      DOUBLE PRECISION MXA      , MYA      , MZA
      DOUBLE PRECISION MXT      , MYT      , MZT
      DOUBLE PRECISION MRCX      , MRCY      , MRCZ
      DOUBLE PRECISION MXVCS      , MYVCS      , MZVCS
      DOUBLE PRECISION MXACS      , MYACS      , MZACS
      DOUBLE PRECISION MX      , MY      , MZ
      DOUBLE PRECISION MACH      , MDOTT      , MDOTF
      DOUBLE PRECISION MDOTV      , MDOTA      , LFRACS
      DOUBLE PRECISION KN      , KM      , MDLTFR
      DOUBLE PRECISION KTHT      , KTHTD      , MDELTA
      DOUBLE PRECISION KNE      , KME      , MALPHA
      DOUBLE PRECISION LATT      , LONGT      , MVS
      DOUBLE PRECISION KA      , KV      , MAGRTR
      DOUBLE PRECISION MAGLOS      , MGRD'TR      , MAGR
      DOUBLE PRECISION MAGV      , MGRDOT      , MXYZDD
      DOUBLE PRECISION MGR      , LAT      , LONG
      DOUBLE PRECISION MISS      , MVR      , MVRWM
      DOUBLE PRECISION ATHRF(4)    , CMMD(2)      , VCMDL(4)

```

|                  |            |              |                 |
|------------------|------------|--------------|-----------------|
| DOUBLE PRECISION | RTER (3)   | , CTI (9)    | , CMS (9)       |
| DOUBLE PRECISION | CIT (9)    | , GRT (5, 3) | , ADISTT (4, 3) |
| DOUBLE PRECISION | RTAR (3)   | , VTAR (3)   | , UVS (3)       |
| DOUBLE PRECISION | VC (3)     | , DLV (3)    | , VTTP (3)      |
| DOUBLE PRECISION | VS (3)     | , US (3)     | , AC (3)        |
| DOUBLE PRECISION | CIR (9)    | , CIM (9)    | , CMI (9)       |
| DOUBLE PRECISION | VW (3)     | , WC (3)     | , PM (3)        |
| DOUBLE PRECISION | CGEST (3)  | , RRELTR (3) | , VRELTR (3)    |
| DOUBLE PRECISION | LAMTRU (2) | , LAMDXX (2) | , LAMDTR (2)    |
| DOUBLE PRECISION | LAMSEK (2) | , LAMDSK (2) | , LAMM (2)      |
| DOUBLE PRECISION | RTEST (3)  | , RREL (3)   | , URREL (3)     |
| DOUBLE PRECISION | VREL (3)   | , TI2M (9)   | , USI (3)       |
| DOUBLE PRECISION | QS1 (4)    | , VMI (3)    | , RMI (3)       |
| DOUBLE PRECISION | VMIR (3)   | , KMIR (3)   | , VTEST (3)     |
| DOUBLE PRECISION | AT (3)     | , XYZR (3)   | , GB (3)        |
| DOUBLE PRECISION | GR (3)     | , CER (9)    | , CRI (9)       |
| DOUBLE PRECISION | VRWM (3)   | , CEI (9)    | , CIE (9)       |
| DOUBLE PRECISION | PG (3)     | , VTT (3)    | , US0 (3)       |
| DOUBLE PRECISION | PG0 (3)    | , USF (3)    | , QUATIC (4)    |
| DOUBLE PRECISION | TI2MO (9)  | , GREST (3)  |                 |
| DOUBLE PRECISION | LAMMO (2)  | , RRELO (3)  | , VRELO (3)     |
| DOUBLE PRECISION | RRELM (3)  | , VRELM (3)  | , GRTEST (3)    |
| DOUBLE PRECISION | FOFF1 (4)  | , FOFF2 (4)  |                 |

## C NAMELIST INPUTS

|                  |               |                 |                |
|------------------|---------------|-----------------|----------------|
| DOUBLE PRECISION | IXX           | , IYY           | , IZZ          |
| DOUBLE PRECISION | CG (3)        | , MASS          | , PQR (3)      |
| DOUBLE PRECISION | IMPULS        | , QUAT (4)      | , MDOT         |
| DOUBLE PRECISION | QUATD (4)     | , BOFF2 (2)     | , TMVCS (6, 4) |
| DOUBLE PRECISION | THVCS (6, 4)  | , DTOFFV (4)    | , VG (3)       |
| DOUBLE PRECISION | TMACSA (8, 4) | , THACSA (8, 4) | , DTACSA (4)   |
| DOUBLE PRECISION | TMACSB (8, 4) | , THACSB (8, 4) | , DTACSB (4)   |
| DOUBLE PRECISION | XYZE (3)      | , XYZED (3)     | , RTIC (5, 3)  |
| DOUBLE PRECISION | VTIC (5, 3)   | , PULSEG (3)    | , QFRACG (3)   |
| DOUBLE PRECISION | PULSEA (3)    | , QFRACA (3)    | , XYZEDD (3)   |
| DOUBLE PRECISION | LAM (2)       | , LAMD (2)      | , VGM (3)      |
| DOUBLE PRECISION | DTVCSY (3)    | , DTVCSY (3)    | , FLTC (4)     |
| DOUBLE PRECISION | TOFFLT (4)    | , TMF (8, 4)    | , THF (8, 4)   |
| DOUBLE PRECISION | DTOFF (4)     | , VWIC (3)      | , AOFF1 (4)    |
| DOUBLE PRECISION | VTTIC (3)     | , USD (3)       | , VCMD (4)     |
| DOUBLE PRECISION | PGD (3)       | , VWD (3)       | , MASS0        |
| DOUBLE PRECISION | MSSTG2        | , LATLP         | , LONGLP       |
| DOUBLE PRECISION | IMPLS0        | , MVRDOT        | , CAZ (100)    |
| DOUBLE PRECISION | CEL (100)     | , RJ (5)        |                |
| DOUBLE PRECISION | AZSUB (100)   | , ELSUB (100)   | , RJSUB (100)  |
| DOUBLE PRECISION | AOFF2 (4)     |                 |                |

|         |            |            |            |
|---------|------------|------------|------------|
| INTEGER | LENVCS (4) | , LENA (4) | , LENB (4) |
| INTEGER | LENF (4)   | , GYSEED   | , FRMCNT   |
| INTEGER | SKSEED     | , SEKTYP   | , ACQD     |
| INTEGER | TERM       | , TOSEED   | , VLVCM5   |
| INTEGER | ESTATE     |            |            |
| INTEGER | TRACK      |            |            |
| INTEGER | ROWBEG     | , COLBEG   | , PLOTNO   |

DOUBLE PRECISION TSTEP, DELT

DOUBLE PRECISION TMSUDRIV, TTSUDRIV, TRSUDRIV, TIMUDRIV,

\* TGPUDRIV, TAPUDRIV, TSPUDRIV, TKFUDRIV

DOUBLE PRECISION TMSUSTEP, TTSUSTEP, TRSUSTEP, TIMUSTEP,

\* TGPUSTEP, TAPUSTEP, TSPUSTEP, TKFUSTEP

Integer first, jrst, krst

\* DATA INITIALIZATION

```

$INCLUDE ('^/INCLUDE/SSDATA35.DAT')
$INCLUDE ('^/INCLUDE/SSDATA38.DAT')
$INCLUDE ('^/INCLUDE/SSDATA39.DAT')
$INCLUDE ('^/INCLUDE/SSDATA42.DAT')
$INCLUDE ('^/INCLUDE/SSDATA44.DAT')
$INCLUDE ('^/INCLUDE/SSDATA45.DAT')
$INCLUDE ('^/INCLUDE/SSDATA46.DAT')
$INCLUDE ('^/INCLUDE/SSDATA47.DAT')
$INCLUDE ('^/INCLUDE/SSDATA48.DAT')
$INCLUDE ('^/INCLUDE/SSDATA49.DAT')
$INCLUDE ('^/INCLUDE/SSDATA50.DAT')
$INCLUDE ('^/INCLUDE/SSDATA01.DAT')
$INCLUDE ('^/INCLUDE/SSDATA17.DAT')
$INCLUDE ('^/INCLUDE/SSDATA18.DAT')
$INCLUDE ('^/INCLUDE/SSDATA21.DAT')
$INCLUDE ('^/INCLUDE/SSDATA22.DAT')
$INCLUDE ('^/INCLUDE/SSDATA23.DAT')
$INCLUDE ('^/INCLUDE/SSDATA28.DAT')
$INCLUDE ('^/INCLUDE/SSDATA29.DAT')
$INCLUDE ('^/INCLUDE/SSDATA30.DAT')
$INCLUDE ('^/INCLUDE/SSDATA71.DAT')
$INCLUDE ('^/INCLUD. /SSTIMING.DAT')

$INCLUDE ('^/INCLUDE/SSMAS_cg.DAT')

      DATA IMASS , IAERO , IBTHR , IBAUTO / 1 , 1 , 1 , 1 /

* INITIALIZE 80x87
  CALL CW87
$INCLUDE ('SSp08.DAT')

C-----C
C-----C MAIN EXECUTION LOOP -----C
C-----C
C      Execution of all events is performed C
C      within this loop C
C C
C-----C

1000 CONTINUE

C      WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C-----C MISSILE STATE UPDATE MODULE -----C
C-----C
C      Integrate missile states to current time C
C C
C-----C

      IF ( tstep .ge. tmsudriv ) THEN

          tmsudriv = tmsudriv + tmsustep

C-----C
C-----C VEHICLE STATES MODULE -----C
C-----C
C      Compute missile state derivatives C
C C
C-----C

```

```

CALL MISSIL(T,CIM,MASS,
.          FXACS,FXVCS,FYACS,FYVCS,
.          FZACS,FZVCS,
.          X,Y,Z,NCLEAR,UD,VD,WD,
.          GB,GR,MGR,FX,FY,FZ,XDD,YDD,ZDD,MXYZDD)

```

```

C-----C
C          MISSILE STATE INTEGRATION MODULE          C
C-----C
C          Revise missile states using derivatives C
C          just computed . Missile states must not C
C          be integrated if a table lookup index C
C          transition has occurred since the last C
C          integration step . The next integration C
C          step should be rescheduled to coincide C
C          with the earliest detected table lookup C
C          index transition instead . Otherwise C
C          schedule the next integration step to C
C          occur at the default step size . C
C-----C

```

#### C TRAPEZOIDAL INTEGRATION FOR SIMPLICITY

```

CALL INTEG ( XD      , XDD      , T , 6 )
CALL INTEG ( YD      , YDD      , T , 7 )
CALL INTEG ( ZD      , ZDD      , T , 8 )
CALL INTEG ( X       , XD       , T , 9 )
CALL INTEG ( Y       , YD       , T , 10 )
CALL INTEG ( Z       , ZD       , T , 11 )

```

#### C TRANSFORM INERTIAL POSITION AND VELOCITY TO EARTH FRAME

```

XYZE(1) = CIE(1)*X + CIE(4)*Y + CIE(7)*Z
XYZE(2) = CIE(2)*X + CIE(5)*Y + CIE(8)*Z
XYZE(3) = CIE(3)*X + CIE(6)*Y + CIE(9)*Z

XYZED(1) = CIE(1)*XD + CIE(4)*YD + CIE(7)*ZD
XYZED(2) = CIE(2)*XD + CIE(5)*YD + CIE(8)*ZD
XYZED(3) = CIE(3)*XD + CIE(6)*YD + CIE(9)*ZD

XYZEDD(1) = CIE(1)*XDD + CIE(4)*YDD + CIE(7)*ZDD
XYZEDD(2) = CIE(2)*XDD + CIE(5)*YDD + CIE(8)*ZDD
XYZEDD(3) = CIE(3)*XDD + CIE(6)*YDD + CIE(9)*ZDD

```

#### C ROTATING EARTH MODEL

```
CALL MMK(0.0D0,1,0.0D0,2,OMEGAE*T,3,CER)
```

```

XYZR(1) = CER(1)*XYZE(1) + CER(4)*XYZE(2) + CER(7)*XYZE(3)
XYZR(2) = CER(2)*XYZE(1) + CER(5)*XYZE(2) + CER(8)*XYZE(3)
XYZR(3) = CER(3)*XYZE(1) + CER(6)*XYZE(2) + CER(9)*XYZE(3)

```

```

CIR(1) = CER(1)*CIE(1) + CER(4)*CIE(2) + CER(7)*CIE(3)
CIR(2) = CER(2)*CIE(1) + CER(5)*CIE(2) + CER(8)*CIE(3)
CIR(3) = CER(3)*CIE(1) + CER(6)*CIE(2) + CER(9)*CIE(3)
CIR(4) = CER(1)*CIE(4) + CER(4)*CIE(5) + CER(7)*CIE(6)
CIR(5) = CER(2)*CIE(4) + CER(5)*CIE(5) + CER(8)*CIE(6)
CIR(6) = CER(3)*CIE(4) + CER(6)*CIE(5) + CER(9)*CIE(6)
CIR(7) = CER(1)*CIE(7) + CER(4)*CIE(8) + CER(7)*CIE(9)
CIR(8) = CER(2)*CIE(7) + CER(5)*CIE(8) + CER(8)*CIE(9)
CIR(9) = CER(3)*CIE(7) + CER(6)*CIE(8) + CER(9)*CIE(9)

```

```

CRI(1)  = CIR(1)
CRI(2)  = CIR(4)
CRI(3)  = CIR(7)
CRI(4)  = CIR(2)
CRI(5)  = CIR(5)
CRI(6)  = CIR(8)
CRI(7)  = CIR(3)
CRI(8)  = CIR(6)
CRI(9)  = CIR(9)

```

C        CALCULATE CURRENT LATITUDE AND LONGITUDE

```

LAT      = DATAN2(XYZR(3),DSQRT(XYZR(1)**2+XYZR(2)**2))/DTR
LONG     = DATAN2(XYZR(2),XYZR(1))/DTR

```

C        CALCULATE CURRENT MISSILE ALTITUDE

```

ALT      = DSQRT ( X**2 + Y**2 + Z**2 ) - RADE

```

C        SAVE TIME OF LAST MISSILE STATE UPDATE

```

TLMSU   = T

```

ENDIF

```

C-----C
C----- Processor communication -----C
C-----C

```

C----- Communicate with p01 -----C

```

CALL SEND_REAL_32BIT( SNGL(GR(01)) )
CALL SEND_REAL_32BIT( SNGL(GR(02)) )
CALL SEND_REAL_32BIT( SNGL(GR(03)) )
CALL RECEIVE_REAL_32BIT( S_MASS )
MASS = S_MASS
CALL SEND_REAL_64BIT( XYZE(01) )
CALL SEND_REAL_64BIT( XYZE(02) )
CALL SEND_REAL_64BIT( XYZE(03) )
CALL SEND_REAL_64BIT( XYZED(01) )
CALL SEND_REAL_64BIT( XYZED(02) )
CALL SEND_REAL_64BIT( XYZFD(03) )

```

C----- Communicate with p03 -----C

```

CALL SEND_REAL_64BIT( X )
CALL SEND_REAL_64BIT( Y )
CALL SEND_REAL_64BIT( Z )
CALL SEND_REAL_32BIT( SNGL(X) )
CALL SEND_REAL_32BIT( SNGL(Y) )
CALL SEND_REAL_32BIT( SNGL(Z) )
CALL SEND_REAL_32BIT( SNGL(XD) )
CALL SEND_REAL_32BIT( SNGL(YD) )
CALL SEND_REAL_32BIT( SNGL(ZD) )
CALL RECEIVE_REAL_32BIT( S_CIM(1) )
CALL RECEIVE_REAL_32BIT( S_CIM(2) )
CALL RECEIVE_REAL_32BIT( S_CIM(3) )
CALL RECEIVE_REAL_32BIT( S_CIM(4) )
CALL RECEIVE_REAL_32BIT( S_CIM(5) )
CALL RECEIVE_REAL_32BIT( S_CIM(6) )
CALL RECEIVE_REAL_32BIT( S_CIM(7) )
CALL RECEIVE_REAL_32BIT( S_CIM(8) )

```

```

CALL RECEIVE_REAL_32BIT( S_CIM(9) )
CIM(1) = S_CIM(1)
CIM(2) = S_CIM(2)
CIM(3) = S_CIM(3)
CIM(4) = S_CIM(4)
CIM(5) = S_CIM(5)
CIM(6) = S_CIM(6)
CIM(7) = S_CIM(7)
CIM(8) = S_CIM(8)
CIM(9) = S_CIM(9)

C----- Receive from ACSTHR and VCSTHR -----C

CALL receive_REAL_32BIT( S_fxvcs )
CALL receive_REAL_32BIT( S_fyvcs )
CALL receive_REAL_32BIT( S_fzvcs )
CALL receive_REAL_32BIT( S_fxacs )
CALL receive_REAL_32BIT( S_fyacs )
CALL receive_REAL_32BIT( S_fzacs )
FXVCS = S_FXVCS
FYVCS = S_FYVCS
FZVCS = S_FZVCS
FXACS = S_FXACS
FYACS = S_FYACS
FZACS = S_FZACS

CALL SEND_REAL_32BIT( SNGL(UD) )
CALL SEND_REAL_32BIT( SNGL(VD) )
CALL SEND_REAL_32BIT( SNGL(WD) )

C-----C
C----- OUTPUT MODULE -----C
C-----C
C          Creates print and plot output data   C
C          files                               C
C                                           C
C-----C

    iprint = iprint + 1

    if ( iprint .eq. int(dtpnt) ) then
      WRITE(MESSAGE,202) T,ALT,X,Y,Z
      CALL OUTMES(MESSAGE)
202  FORMAT(1X, f8.4, 4E14.7)
      iprint = 0
    ENDIF

C-----C
C----- TERMINATION LOGIC -----C
C-----C
C          Defines the simulation termination   C
C          conditions                           C
C                                           C
C-----C

C    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

    IEXIT = 0

C    ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
C    EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED

    IF ( T.GE.TFINAL ) THEN
      IEXIT = 1

```

```

ENDIF

C  ENABLE EXIT IF MISSILE HAS IMPACTED AND ALL EVENTS SCHEDULED FOR
C  THIS TIME HAVE BEEN EXECUTED

    IF ( ALT.LT.0.0 ) THEN
        IEXIT = 1
    ENDIF

C  increment time

    TSTEP = TSTEP + 1.0D0
    T = TSTEP * DELT

C  CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

    IF ( IEXIT.EQ.0 ) GO TO 1000

    CALL OUTMES('ERROR: Exit from P08')
    END

```

## B.1.10 Up09.for

```

c      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      CHARACTER*128 MESSAGE

      COMMON / NORCOM / GSET      , ISET
      COMMON / RANCOM /          RANSEQ(97), RANLST

      REAL  MAGRTR      , snr
      REAL  LAMSEK(2)   , LAMM(2)
      real  gset, ranlst

      INTEGER      FRMCNT      , iset
      INTEGER      SEKTYP      , ACQD
      INTEGER      TERM        , TRACK
      INTEGER*4     SKSEED

      REAL TSTEP,DELT
      REAL TSPUDRIV,TSPUSTEP
      real delt_time
$INCLUDE('pfp:INCLUDE/target.for')

* INITIALIZE 80x87
  CALL CW87

$INCLUDE('ssp09.dat')

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C      Execution of all events is performed  C
C      within this loop                      C
C-----C

c      CALL INITIALIZE_TIMING()

      1000 CONTINUE

c      call reset_timer()
c      timer = read_timer()

c      CALL START_TIMING(0)

C      WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C----- Processor communication -----C
C-----C

c      CALL SWITCH_TIMING()

C----- COMMUNICATION WITH KALMAN -----C

      call send_real_32bit( lamm(1) )

```

```

call send_real_32bit( lamm(2) )
call send_real_32bit( snr )
call send_real_32bit( frmrat )

C----- COMMUNICATION WITH RELAT -----C

CALL RECEIVE_REAL_32BIT( LAMSEK(01) )
CALL RECEIVE_REAL_32BIT( LAMSEK(02) )
CALL RECEIVE_REAL_32BIT( MAGRTR )

c    CALL SWITCH_TIMING()

C----- SEEKER MODULE -----C
C----- Calculates LOS angles measured by the seeker -----C
C-----
C-----

IF ( TSTEP .GE. TSPUDRIV ) THEN
c    TSPUDRIV = TSPUDRIV + TSPUSTEP

    CALL SEEKER(T,ACQD,LAMSEK,MAGRTR,SKSEED,FRMRAT,FRMCNT,
                SAMRAT,TRACK,TERM,SNR,LAMM)

    tspudriv = tspudriv + int(1000.0/frmrat)
ENDIF

c    delt_time = (timer -(read_timer() + 18))/1.229e6
c    CALL output_message( %VAL(real_32bit), delt_time,
c    & %VAL(int2(1)) )
c    call output_n1

C----- OUTPUT MODULE -----C
C-----
C-----

c    call stop_timing()

c    if ( mod(idnint(tstep),idnint(dtpert)).eq.0 ) then
c        call output_timing()
c        call INITIALIZE_TIMING()
c    ENDIF

C----- TERMINATION LOGIC -----C
C----- Defines the simulation termination conditions -----C
C-----
C-----

C    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

IEXIT = 0

C    increment time

```

```
TSTEP = TSTEP + 1.0  
T = TSTEP * DELT  
C   CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET  
    IF ( IEXIT.EQ.0 ) GO TO 1000  
    END
```

## B.1.11 Uup10.for

```

C      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C

      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / NORCOM / GSET , ISET
      COMMON / RANCOM / RANSEQ , RANLST
      COMMON / RACCEL / DRSIGA , PSIA , THTA , PHIA , THXZA ,
      .              THXYA , THYZA , THYXA , THZYA , THZXA ,
      .              SF1A , SF2A , DCA , TOACCE , GRLST ,
      .              XYZDP , ABI2 , ABI1 , ABO2 , ABO1

      REAL S_PD,S_QD,S_RD,S_UD,S_VD,S_WD,S_CIM(9)
      REAL S_CG(3),S_P,S_Q,S_R,S_XD,S_YD,S_ZD,S_GR(3)
      DOUBLE PRECISION SF1A(3) , SF2A(3) , DCA(3)
      REAL RANSEQ(97) , RANLST
      DOUBLE PRECISION GRLST(3) , XYZDP(3)
      DOUBLE PRECISION ABI2(3) , ABI1(3) , ABO2(3)
      DOUBLE PRECISION ABO1(3) , CIM(9)
      DOUBLE PRECISION GR(3) , PULSEA(3)
      DOUBLE PRECISION QFRACA(3) , CG(3)
      DOUBLE PRECISION LONGLP, LATLP, CIMO(9)

      INTEGER*4 GYSEED
      INTEGER SEKTYP

      DOUBLE PRECISION TSTEP,DELT
      DOUBLE PRECISION TIMUDRIV,TIMUSTEP

* DATA INITIALIZATION

$include ('^/include/ssdata35.dat')
$include ('^/include/ssdata38.dat')
$include ('^/include/ssdata39.dat')
$include ('^/include/ssdata42.dat')
$include ('^/include/ssdata44.dat')
$include ('^/include/ssdata45.dat')
$include ('^/include/ssdata46.dat')
$include ('^/include/ssdata47.dat')
$include ('^/include/ssdata48.dat')
$include ('^/include/ssdata49.dat')
$include ('^/include/ssdata50.dat')
$include ('^/include/ssdata01.dat')
$include ('^/include/ssdata17.dat')
$include ('^/include/ssdata18.dat')
$include ('^/include/ssdata21.dat')
$include ('^/include/ssdata22.dat')
$include ('^/include/ssdata23.dat')
$include ('^/include/ssdata28.dat')
$include ('^/include/ssdata29.dat')
$include ('^/include/ssdata30.dat')
$include ('^/include/ssdata71.dat')
$include ('^/include/sstiming.dat')

```

```

      call cw87
$include ('sspl0.dat')

```

```

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed  C
C           within this loop                      C
C-----C

```

```

1000 CONTINUE

```

```

C      WRITE(*,*) '-----BEGINNING OF LOOP-----'

```

```

C-----C
C----- Processor communication -----C
C-----C
C----- Communicate with p01 -----C

```

```

      CALL RECEIVE_REAL_32BIT( S_GR(1) )
      CALL RECEIVE_REAL_32BIT( S_GR(2) )
      CALL RECEIVE_REAL_32BIT( S_GR(3) )
      GR(1) = DBLE(S_GR(1))
      GR(2) = DBLE(S_GR(2))
      GR(3) = DBLE(S_GR(3))

```

```

      CALL SEND_REAL_32BIT( SNGL(PULSEA(01)) )
      CALL SEND_REAL_32BIT( SNGL(PULSEA(02)) )
      CALL SEND_REAL_32BIT( SNGL(PULSEA(03)) )

```

```

      CALL RECEIVE_REAL_32BIT( S_CG(1) )
      CALL RECEIVE_REAL_32BIT( S_CG(2) )
      CALL RECEIVE_REAL_32BIT( S_CG(3) )
      CG(1) = DBLE(S_CG(1))
      CG(2) = DBLE(S_CG(2))
      CG(3) = DBLE(S_CG(3))
      CALL RECEIVE_REAL_32BIT( S_P )
      CALL RECEIVE_REAL_32BIT( S_Q )
      CALL RECEIVE_REAL_32BIT( S_R )
      P = DBLE(S_P)
      Q = DBLE(S_Q)
      R = DBLE(S_R)

```

```

      CALL RECEIVE_REAL_32BIT( S_XD )
      CALL RECEIVE_REAL_32BIT( S_YD )
      CALL RECEIVE_REAL_32BIT( S_ZD )
      XD = DBLE(S_XD)
      YD = DBLE(S_YD)
      ZD = DBLE(S_ZD)

```

```

      CALL RECEIVE_REAL_32BIT( S_CIM(1) )
      CALL RECEIVE_REAL_32BIT( S_CIM(2) )
      CALL RECEIVE_REAL_32BIT( S_CIM(3) )
      CALL RECEIVE_REAL_32BIT( S_CIM(4) )
      CALL RECEIVE_REAL_32BIT( S_CIM(5) )
      CALL RECEIVE_REAL_32BIT( S_CIM(6) )
      CALL RECEIVE_REAL_32BIT( S_CIM(7) )
      CALL RECEIVE_REAL_32BIT( S_CIM(8) )
      CALL RECEIVE_REAL_32BIT( S_CIM(9) )

```

```

CIM(1) = DBLE(S_CIM(1))
CIM(2) = DBLE(S_CIM(2))
CIM(3) = DBLE(S_CIM(3))
CIM(4) = DBLE(S_CIM(4))
CIM(5) = DBLE(S_CIM(5))
CIM(6) = DBLE(S_CIM(6))
CIM(7) = DBLE(S_CIM(7))
CIM(8) = DBLE(S_CIM(8))
CIM(9) = DBLE(S_CIM(9))

```

```

CALL RECEIVE_REAL_32BIT( S_PD )
CALL RECEIVE_REAL_32BIT( S_QD )
CALL RECEIVE_REAL_32BIT( S_RD )
CALL RECEIVE_REAL_32BIT( S_UD )
CALL RECEIVE_REAL_32BIT( S_VD )
CALL RECEIVE_REAL_32BIT( S_WD )
PD = DBLE(S_PD)
QD = DBLE(S_QD)
RD = DBLE(S_RD)
UD = DBLE(S_UD)
VD = DBLE(S_VD)
WD = DBLE(S_WD)

```

```

C-----C
C----- INERTIAL MEASUREMENT UPDATE -----C
C-----C
C           Get inertial measurement data needed   C
C           for guidance calculations .             C
C                                                    C
C-----C

```

```

IF ( TSTEP .GE. TIMUDRIV ) THEN

```

```

    TIMUDRIV = TIMUDRIV + TIMUSTEP

```

```

C-----C
C----- ACCELEROMETER MODULE -----C
C-----C
C           Determine sensed accelerations          C
C                                                    C
C-----C

```

```

    CALL ACCEL(T,UD,VD,WD,P,Q,R,PD,QD,RD,CG,CIM,XD,YD,ZD,
    GR,GYSEED,QFRACA,PULSEA)

```

```

ENDIF

```

```

C-----C
C----- TERMINATION LOGIC -----C
C-----C
C           Defines the simulation termination      C
C           conditions                             C
C                                                    C
C-----C

```

```

C    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

```

```

    IEXIT = 0

```

```

C    increment time

```

Appendix B - Exosim v2.0 Midcourse and Terminal Phases

TSTEP = TSTEP + 1.0d0  
T = TSTEP \* DELT

C    CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET  
     IF ( IEXIT.EQ.0 ) GO TO 1000  
     END

## B.1.12 Uup11.for

```

c      PROGRAM EXOSIM
C-----C
C----- Declare and initialize variables -----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      real rdum
      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RMGUID / ISEQ , TVCOMP , OMEGA0 , IMIDB2 , TMIDB2 ,
      .             ISK3ON
      COMMON / RMAUTO / ANGACL , IMCPAS , TP2END , TP3END , IP2END ,
      .             TCOAST , ICOAST , TRDONE , IRATE , IACSB1 ,
      .             IACSB2 , ICNT , IVPFL , IVPFLN , TBURN2 ,
      .             OMEGAI , TLSTMA , AACCEL
      COMMON / RKVAUT / SW17 , SW18 , SW18P , SW18Y , SW19 ,
      .             SW19P , SW19Y , IROLL , TPTON2 , TYTON2 ,
      .             TNEXTP , TNEXTY , FLTCLPL , FLTCYL

      REAL T,TSTEP,DELT
      REAL TMSUDRIV,TIMUDRIV,TGPUDRIV,TAPUDRIV,TSPUDRIV,TKFUDRIV
      REAL TMSUSTEP,TIMUSTEP,TGPUSTEP,TAPUSTEP,TSPUSTEP,TKFUSTEP
      REAL FLTC(4)

      REAL dtacsa_s(4),dtacsb_s(4),dtoffv_s(4),tofflt_s(4)
      REAL ANGACL(3,4,10)
      REAL OMEGAI(3) , GRLAST(3)
      REAL OMEGA0(3) , AACCEL(3,4)

      INTEGER      FIRST1 , FIRST2
      INTEGER      ISEQ(4) , IMCPAS(3,4) , FLIP

C      OUTPUTS

      REAL MAGRTR , MAGR , MASS
      REAL MAGV , MGRDOT , CMS(9)
      REAL ADISTT(4,3) , LAMDXX(2)
      REAL LAMSEK(2) , LAMM(2)
      REAL RREL(3) , URREL(3)
      REAL VREL(3) , TI2M(9)
      REAL QS1(4) , VMI(3) , RMI(3)
      REAL VTEST(3) , VMIR(3) , GRTEST(3)
      REAL AT(3)
      REAL CIE(9)

C      NAMELIST INPUTS

      REAL IXX , IYY , IZZ
      REAL CG(3) , DTOFFV(4) , VG(3)
      REAL DTACSA(4) , DTACSB(4)
      REAL XYZE(3) , XYZED(3)
      REAL GRT(5,3) , VTIC(5,3) , rtic(5,3)
      REAL PULSEA(3) , PULSEG(3)
      REAL LAM(2) , LAMD(2) , VGM(3)
      REAL DTVCS(3) , DTVCSY(3)
      REAL TOFFLT(4) , LATLP , LONGLP

      INTEGER      SEKTYP , ACQD

```

```

      INTEGER          TERM          ,  TOSEED          , ESTATE
      INTEGER          TRACK
      REAL    RTEST(3)          ,  RMIR(3)

```

## \* DATA INITIALIZATION

```

$INCLUDE('^/INCLUDE/SSDATA35.DAT')
$INCLUDE('^/INCLUDE/SSDATA38.DAT')
$INCLUDE('^/INCLUDE/SSDATA39.DAT')
$INCLUDE('^/INCLUDE/SSDATA42.DAT')
$INCLUDE('^/INCLUDE/SSDATA44.DAT')
$INCLUDE('^/INCLUDE/SSDATA45.DAT')
$INCLUDE('^/INCLUDE/SSDATA46.DAT')
$INCLUDE('^/INCLUDE/SSDATA47.DAT')
$INCLUDE('^/INCLUDE/SSDATA48.DAT')
$INCLUDE('^/INCLUDE/SSDATA49.DAT')
$INCLUDE('^/INCLUDE/SSDATA50.DAT')
$INCLUDE('^/INCLUDE/SSDATA01.DAT')
$INCLUDE('^/INCLUDE/SSDATA17.DAT')
$INCLUDE('^/INCLUDE/SSDATA18.DAT')
$INCLUDE('^/INCLUDE/SSDATA21.DAT')
$INCLUDE('^/INCLUDE/SSDATA22.DAT')
$INCLUDE('^/INCLUDE/SSDATA23.DAT')
$INCLUDE('^/INCLUDE/SSDATA28.DAT')
$INCLUDE('^/INCLUDE/SSDATA29.DAT')
$INCLUDE('^/INCLUDE/SSDATA30.DAT')
$INCLUDE('^/INCLUDE/SSDA1A71.DAT')
$INCLUDE('^/INCLUDE/SSTIMING.DAT')
$INCLUDE(':pfp:INCLUDE/target.for')

```

## \* INITIALIZE 80x87

CALL CW87

\$INCLUDE('SSp11.DAT')

```

      CALL MCAUTO(T, IXX, IYY, IZZ, SP, SQ, SR, ROLLER, PITER,
.           YAWER, IDIS, IACSON, IBURND, IBURNM, IDMEAS, IPASSM,
.           ICMD, IRATON, TPATON, TYATON, DTSAMP, TSAL, TSAH,
.           TLAPS, ITHRES, ANVP, ACSLEV, TMAUTO, 0)

```

```

      idrop_s = idrop
      acslev_s = acslev
      dtacsa_s(1) = dtacsa(1)
      dtacsa_s(2) = dtacsa(2)
      dtacsa_s(3) = dtacsa(3)
      dtacsa_s(4) = dtacsa(4)
      dtacsb_s(1) = dtacsb(1)
      dtacsb_s(2) = dtacsb(2)
      dtacsb_s(3) = dtacsb(3)
      dtacsb_s(4) = dtacsb(4)
      dtoffv_s(1) = dtoffv(1)
      dtoffv_s(2) = dtoffv(2)
      dtoffv_s(3) = dtoffv(3)
      dtoffv_s(4) = dtoffv(4)
      ithres_s = ithres
      ivcs_s = ivcs
      ivtab_s = ivtab
      tatab_s = tatab
      tburnm_s = tburnm
      timonv_s = timonv
      tofflt_s(1) = tofflt(1)
      tofflt_s(2) = tofflt(2)
      tofflt_s(3) = tofflt(3)
      tofflt_s(4) = tofflt(4)
      tvtab_s = tvtab

```

```

C-----C
C----- - MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed   C
C           within this loop                       C
C-----C

1000 CONTINUE

C   WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C----- Processor communication -----C
C-----C

C----- COMMUNICATION WITH P00 -----C

CALL RECEIVE_REAL_32BIT( IXX )
CALL RECEIVE_REAL_32BIT( IYY )
CALL RECEIVE_REAL_32BIT( IZZ )
CALL RECEIVE_REAL_32BIT( MASS )

C----- COMMUNICATION WITH P00 -----C

CALL SEND_SIGNED_16BIT( IDROP_s )

C----- COMMUNICATION WITH P02 -----C
CALL SEND_REAL_32BIT( ACSLEV_s )
CALL SEND_REAL_32BIT( DTACSA_s(01) )
CALL SEND_REAL_32BIT( DTACSA_s(02) )
CALL SEND_REAL_32BIT( DTACSA_s(03) )
CALL SEND_REAL_32BIT( DTACSA_s(04) )
CALL SEND_REAL_32BIT( DTACSB_s(01) )
CALL SEND_REAL_32BIT( DTACSB_s(02) )
CALL SEND_REAL_32BIT( DTACSB_s(03) )
CALL SEND_REAL_32BIT( DTACSB_s(04) )
CALL SEND_REAL_32BIT( DTOFFV_s(01) )
CALL SEND_REAL_32BIT( DTOFFV_s(02) )
CALL SEND_REAL_32BIT( DTOFFV_s(03) )
CALL SEND_REAL_32BIT( DTOFFV_s(04) )
CALL SEND_SIGNED_16BIT( ITHRES_s )
CALL SEND_SIGNED_16BIT( IVCS_s )
CALL SEND_SIGNED_16BIT( IVTAB_s )
CALL SEND_REAL_32BIT( TATAB_s )
CALL SEND_REAL_32BIT( TBURNM_s )
CALL SEND_REAL_32BIT( TIMONV_s )
CALL SEND_REAL_32BIT( TOFFLT_s(01) )
CALL SEND_REAL_32BIT( TOFFLT_s(02) )
CALL SEND_REAL_32BIT( TOFFLT_s(03) )
CALL SEND_REAL_32BIT( TOFFLT_s(04) )
CALL SEND_REAL_32BIT( TVTAB_s )

C----- COMMUNICATION WITH P02 -----C

CALL RECEIVE_SIGNED_16BIT( IACSON )

C----- COMMUNICATE WITH CORVEL -----C

CALL RECEIVE_REAL_32BIT( VG(01) )
CALL RECEIVE_REAL_32BIT( VG(02) )
CALL RECEIVE_REAL_32BIT( VG(03) )

C----- DAISY CHAIN WITH IMUPRO AND NAVIG -----C

```

```

CALL RECEIVE_REAL_32BIT( TI2M(1) )
CALL RECEIVE_REAL_32BIT( TI2M(2) )
CALL RECEIVE_REAL_32BIT( TI2M(3) )
CALL RECEIVE_REAL_32BIT( TI2M(4) )
CALL RECEIVE_REAL_32BIT( TI2M(5) )
CALL RECEIVE_REAL_32BIT( TI2M(6) )
CALL RECEIVE_REAL_32BIT( TI2M(7) )
CALL RECEIVE_REAL_32BIT( TI2M(8) )
CALL RECEIVE_REAL_32BIT( TI2M(9) )

```

```

CALL RECEIVE_REAL_32BIT( VREL(1) )
CALL RECEIVE_REAL_32BIT( VREL(2) )
CALL RECEIVE_REAL_32BIT( VREL(3) )
CALL RECEIVE_REAL_32BIT( RREL(1) )
CALL RECEIVE_REAL_32BIT( RREL(2) )
CALL RECEIVE_REAL_32BIT( RREL(3) )

```

```

CALL RECEIVE_REAL_32BIT( SP )
CALL RECEIVE_REAL_32BIT( SQ )
CALL RECEIVE_REAL_32BIT( SR )

```

```

call send_real_32bit( magr )
call send_real_32bit( magv )
call send_real_32bit( tgo )
call send_real_32bit( piter )
call send_real_32bit( roller )
call send_real_32bit( yawer )
call send_signed_16bit( iburn1 )
call send_real_32bit( lamd(1) )
call send_real_32bit( lamd(2) )
call send_signed_16bit( acqd )

```

```

call receive_signed_16bit( estate )
call receive_real_32bit( piter )
call receive_real_32bit( roller )
call receive_real_32bit( yawer )
call receive_signed_16bit( iburn1 )
call receive_real_32bit( lamd(1) )
call receive_real_32bit( lamd(2) )
call receive_signed_16bit( acqd )
call receive_real_32bit( tge1 )
call receive_real_32bit( tge2a1 )
call receive_real_32bit( trmtgo )

```

```

C-----C
C          ON BOARD GUIDANCE PROCESSING          C
C-----C
C          Determine guidance commands             C
C-----C
C

```

```

IF ( TSTEP .GE. TGPUDRIV ) THEN

```

```

C          TGPUDRIV = TGPUDRIV + TGPUSTEP

```

```

C-----C
C-----ESTIMATED RELATIVE STATES MODULE-----C
C-----C
C          Estimate range, range rate, and time-to- C
C          go based on navigation output and target C

```

# Appendix B - Exosim v2.0 Midcourse and Terminal Phases

```

C                                     model estimates                                     C
C
C-----C

      CALL ESTREL(TI2M,CMS,ESTATE,RREL,VREL,
      MAGR,MAGV,URREL,MGRDOT,TGO,PITER,YAWER,LAMD)

      ENDIF

C-----C
C----- Processor communication -----C
C-----C

C----- COMMUNICATION WITH P00 -----C

      CALL RECEIVE_REAL_32BIT( IXX )
      CALL RECEIVE_REAL_32BIT( IYY )
      CALL RECEIVE_REAL_32BIT( IZZ )
      CALL RECEIVE_REAL_32BIT( MASS )

C----- COMMUNICATION WITH P00 -----C

      CALL SEND_SIGNED_16BIT( IDROP_s )
C----- COMMUNICATION WITH P02 -----C
      CALL SEND_REAL_32BIT( ACSLEV_s )
      CALL SEND_REAL_32BIT( DTACSA_s(01) )
      CALL SEND_REAL_32BIT( DTACSA_s(02) )
      CALL SEND_REAL_32BIT( DTACSA_s(03) )
      CALL SEND_REAL_32BIT( DTACSA_s(04) )
      CALL SEND_REAL_32BIT( DTACSB_s(01) )
      CALL SEND_REAL_32BIT( DTACSB_s(02) )
      CALL SEND_REAL_32BIT( DTACSB_s(03) )
      CALL SEND_REAL_32BIT( DTACSB_s(04) )
      CALL SEND_REAL_32BIT( DTOFFV_s(01) )
      CALL SEND_REAL_32BIT( DTOFFV_s(02) )
      CALL SEND_REAL_32BIT( DTOFFV_s(03) )
      CALL SEND_REAL_32BIT( DTOFFV_s(04) )
      CALL SEND_SIGNED_16BIT( ITHRES_s )
      CALL SEND_SIGNED_16BIT( IVCS_s )
      CALL SEND_SIGNED_16BIT( IVTAB_s )
      CALL SEND_REAL_32BIT( TATAB_s )
      CALL SEND_REAL_32BIT( TBURNM_s )
      CALL SEND_REAL_32BIT( TIMONV_s )
      CALL SEND_REAL_32BIT( TOFFLT_s(01) )
      CALL SEND_REAL_32BIT( TOFFLT_s(02) )
      CALL SEND_REAL_32BIT( TOFFLT_s(03) )
      CALL SEND_REAL_32BIT( TOFFLT_s(04) )
      CALL SEND_REAL_32BIT( TVTAB_s )

C----- COMMUNICATION WITH P02 -----C

      CALL RECEIVE_SIGNED_16BIT( IACSON )

C----- COMMUNICATE WITH CORVEL -----C

      CALL RECEIVE_REAL_32BIT( VG(01) )
      CALL RECEIVE_REAL_32BIT( VG(02) )
      CALL RECEIVE_REAL_32BIT( VG(03) )

C----- DAISY CHAIN WITH IMUPRO AND NAVIG -----C

      CALL RECEIVE_REAL_32BIT( TI2M(1) )

```

```

CALL RECEIVE_REAL_32BIT( TI2M(2) )
CALL RECEIVE_REAL_32BIT( TI2M(3) )
CALL RECEIVE_REAL_32BIT( TI2M(4) )
CALL RECEIVE_REAL_32BIT( TI2M(5) )
CALL RECEIVE_REAL_32BIT( TI2M(6) )
CALL RECEIVE_REAL_32BIT( TI2M(7) )
CALL RECEIVE_REAL_32BIT( TI2M(8) )
CALL RECEIVE_REAL_32BIT( TI2M(9) )

```

```

CALL RECEIVE_REAL_32BIT( VREL(1) )
CALL RECEIVE_REAL_32BIT( VREL(2) )
CALL RECEIVE_REAL_32BIT( VREL(3) )
CALL RECEIVE_REAL_32BIT( RREL(1) )
CALL RECEIVE_REAL_32BIT( RREL(2) )
CALL RECEIVE_REAL_32BIT( RREL(3) )

```

```

CALL RECEIVE_REAL_32BIT( SP )
CALL RECEIVE_REAL_32BIT( SQ )
CALL RECEIVE_REAL_32BIT( SR )

```

```

C-----C
C----- MISSILE STATE UPDATE MODULE -----C
C-----C
C          Integrate missile states to current time C
C-----C

```

```

C-----C
C----- VCS THRUSTER RESPONSE MODULE -----C
C-----C
C          Determines the forces and moments C
C          imparted by the VCS thrusters C
C-----C

```

```

IF ( T.GE.TKVON ) THEN

```

```

    CALL VCSTHR2(T,FLTIC,FLTCP,FLTCY,TBURNM,TOFFLT,
    ... onv, IVTAB)

```

```

C-----C
C----- ACS THRUSTER RESPONSE MODULE -----C
C-----C
C          Determines the forces and moments C
C          imparted by the ACS thrusters C
C-----C

```

```

    CALL ACSTHR2(ITHRES)

```

```

ENDIF

```

```

C-----C
C----- SEPARATION MODULE -----C
C-----C
C          Models discontinuities occurring during C
C          stage separation C
C-----C

```

```

C-----C

C      NOSE FAIRING / BOOST ADAPTER SEPARATION

      IF ( IDROP.EQ.1 .OR. (ABS(T-TDROP).LE.DTEPS
*      .AND. IGIT.EQ.1 ) ) THEN
          IDROP = 2
          IPASSM = 0
      ENDIF

      IF ( TSTEP .GE. TGPUDRIV ) THEN
          TGPUDRIV = TGPUDRIV + TGPUSTEP
      ENDIF

C-----C
C-----MIDCOURSE GUIDANCE MODULE-----C
C-----C
C      Calculates roll error, controls      C
C      midcourse sequencing, and issues      C
C      midcourse diverts                      C
C-----C

      IF ( T.GT.TSTG2 .AND.
*      T.GE.TMGUID .AND. ACQD.EQ.0 ) THEN

          CALL MCGUID(T, TI2M, VG, URREL, MASS, IDIST, MIDBRN, MAGR,
.      MAGV, SP, SQ, SR, PITER, YAWER, FLIP, IVCS, ICMD, IDMEAS, IDPASS,
.      IDROP, IMCEND, IBURND, IBURNM, VGM, ADISTT, ROLLER,
.      TMGUID)

      ENDIF

      ENDIF

C-----C
C-----KALMAN FILTER MODULE-----C
C-----C

      call send_real_32bit( magr )
      call send_real_32bit( magv )
      call send_real_32bit( tgo )
      call send_real_32bit( piter )
      call send_real_32bit( roller )
      call send_real_32bit( yawer )
      call send_signed_16bit( iburn1 )
      call send_real_32bit( lamd(1) )
      call send_real_32bit( lamd(2) )
      call send_signed_16bit( acqd )

      call receive_signed_16bit( estate )
      call receive_real_32bit( piter )
      call receive_real_32bit( roller )
      call receive_real_32bit( yawer )
      call receive_signed_16bit( iburn1 )
      call receive_real_32bit( lamd(1) )
      call receive_real_32bit( lamd(2) )
      call receive_signed_16bit( acqd )
      call receive_real_32bit( tge1 )
      call receive_real_32bit( tge2a1 )
      call receive_real_32bit( trmtgo )

C-----C

```

```

C----- Processor communication -----C
C-----C

C----- COMMUNICATION WITH P00 -----C

    CALL RECEIVE_REAL_32BIT( IXX )
    CALL RECEIVE_REAL_32BIT( IYY )
    CALL RECEIVE_REAL_32BIT( IZZ )
    CALL RECEIVE_REAL_32BIT( MASS )

C----- COMMUNICATION WITH P00 -----C

    CALL SEND_SIGNED_16BIT( IDROP_s )
C----- COMMUNICATION WITH P02 -----C
    CALL SEND_REAL_32BIT( ACSLEV_s )
    CALL SEND_REAL_32BIT( DTACSA_s(01) )
    CALL SEND_REAL_32BIT( DTACSA_s(02) )
    CALL SEND_REAL_32BIT( DTACSA_s(03) )
    CALL SEND_REAL_32BIT( DTACSA_s(04) )
    CALL SEND_REAL_32BIT( DTACSB_s(01) )
    CALL SEND_REAL_32BIT( DTACSB_s(02) )
    CALL SEND_REAL_32BIT( DTACSB_s(03) )
    CALL SEND_REAL_32BIT( DTACSB_s(04) )
    CALL SEND_REAL_32BIT( DTOFFV_s(01) )
    CALL SEND_REAL_32BIT( DTOFFV_s(02) )
    CALL SEND_REAL_32BIT( DTOFFV_s(03) )
    CALL SEND_REAL_32BIT( DTOFFV_s(04) )
    CALL SEND_SIGNED_16BIT( ITHRES_s )
    CALL SEND_SIGNED_16BIT( IVCS_s )
    CALL SEND_SIGNED_16BIT( IVTAB_s )
    CALL SEND_REAL_32BIT( TATAB_s )
    CALL SEND_REAL_32BIT( TBURNM_s )
    CALL SEND_REAL_32BIT( TIMONV_s )
    CALL SEND_REAL_32BIT( TOFFLT_s(01) )
    CALL SEND_REAL_32BIT( TOFFLT_s(02) )
    CALL SEND_REAL_32BIT( TOFFLT_s(03) )
    CALL SEND_REAL_32BIT( TOFFLT_s(04) )
    CALL SEND_REAL_32BIT( TVTAB_s )

C----- COMMUNICATION WITH P02 -----C

    CALL RECEIVE_SIGNED_16BIT( IACSON )

C----- COMMUNICATE WITH CORVEL -----C

    CALL RECEIVE_REAL_32BIT( VG(01) )
    CALL RECEIVE_REAL_32BIT( VG(02) )
    CALL RECEIVE_REAL_32BIT( VG(03) )

C----- DAISY CHAIN WITH IMUPRO AND NAVIG -----C

    CALL RECEIVE_REAL_32BIT( TI2M(1) )
    CALL RECEIVE_REAL_32BIT( TI2M(2) )
    CALL RECEIVE_REAL_32BIT( TI2M(3) )
    CALL RECEIVE_REAL_32BIT( TI2M(4) )
    CALL RECEIVE_REAL_32BIT( TI2M(5) )
    CALL RECEIVE_REAL_32BIT( TI2M(6) )
    CALL RECEIVE_REAL_32BIT( TI2M(7) )
    CALL RECEIVE_REAL_32BIT( TI2M(8) )
    CALL RECEIVE_REAL_32BIT( TI2M(9) )

    CALL RECEIVE_REAL_32BIT( VREL(1) )
    CALL RECEIVE_REAL_32BIT( VREL(2) )

```

```

CALL RECEIVE_REAL_32BIT( VREL(3) )
CALL RECEIVE_REAL_32BIT( RREL(1) )
CALL RECEIVE_REAL_32BIT( RREL(2) )
CALL RECEIVE_REAL_32BIT( RREL(3) )

```

```

CALL RECEIVE_REAL_32BIT( SP )
CALL RECEIVE_REAL_32BIT( SQ )
CALL RECEIVE_REAL_32BIT( SR )

```

```

call send_real_32bit( magr )
call send_real_32bit( magv )
call send_real_32bit( tgo )
call send_real_32bit( piter )
call send_real_32bit( roller )
call send_real_32bit( yawer )
call send_signed_16bit( iburn1 )
call send_real_32bit( lamd(1) )
call send_real_32bit( lamd(2) )
call send_signed_16bit( acqd )

```

```

call receive_signed_16bit( estate )
call receive_real_32bit( piter )
call receive_real_32bit( roller )
call receive_real_32bit( yawer )
call receive_signed_16bit( iburn1 )
call receive_real_32bit( lamd(1) )
call receive_real_32bit( lamd(2) )
call receive_signed_16bit( acqd )
call receive_real_32bit( tge1 )
call receive_real_32bit( tge2a1 )
call receive_real_32bit( trmtgo )

```

```

C-----C
C----- AUTOPILOTS -----C
C-----C
C-----C
C-----C

```

```

IF ( TSTEP .GE. TAPUDRIV ) THEN

```

```

C-----C
C----- MIDCOURSE AUTOPILOT MODULE -----C
C-----C
C               Performs large angle reorients and rate C
C               control during midcourse C
C-----C

```

```

IF ( T.GE.TKVON ) THEN

```

```

IF ( T.GT.TSTG2 .AND. T.GE.TMAUTO .AND.
      ( ICMD.NE.0 .OR. ACQD.EQ.0 ) ) THEN

```

```

      CALL MCAUTO(T, IXX, IYY, IZZ, SP, SQ, SR, ROLLER, PITER,
      YAWER, IDIST, IACSON, IBURND, IBURNM, IDMEAS, IPASSM,
      ICMD, TRATON, TPATON, TYATON, DTSAMP, TSAL, TSAH,
      TLAPS, ITHRES, ANVP, ACSLEV, TMAUTO, 1)

```

```

      ENDIF
    ENDIF
  ENDIF

```

```

C-----C
C----- Processor communication -----C

```

```

C-----C
      idrop_s = idrop
C----- COMMUNICATION WITH P00 -----C
      CALL RECEIVE_REAL_32BIT( IXX )
      CALL RECEIVE_REAL_32BIT( IYY )
      CALL RECEIVE_REAL_32BIT( IZZ )
      CALL RECEIVE_REAL_32BIT( MASS )

C----- COMMUNICATION WITH P00 -----C
      CALL SEND_SIGNED_16BIT( IDROP_s )
C----- COMMUNICATION WITH P02 -----C
      CALL SEND_REAL_32BIT( ACSLEV_s )
      CALL SEND_REAL_32BIT( DTACSA_s(01) )
      CALL SEND_REAL_32BIT( DTACSA_s(02) )
      CALL SEND_REAL_32BIT( DTACSA_s(03) )
      CALL SEND_REAL_32BIT( DTACSA_s(04) )
      CALL SEND_REAL_32BIT( DTACSB_s(01) )
      CALL SEND_REAL_32BIT( DTACSB_s(02) )
      CALL SEND_REAL_32BIT( DTACSB_s(03) )
      CALL SEND_REAL_32BIT( DTACSB_s(04) )
      CALL SEND_REAL_32BIT( DTOFFV_s(01) )
      CALL SEND_REAL_32BIT( DTOFFV_s(02) )
      CALL SEND_REAL_32BIT( DTOFFV_s(03) )
      CALL SEND_REAL_32BIT( DTOFFV_s(04) )
      CALL SEND_SIGNED_16BIT( ITHRES_s )
      CALL SEND_SIGNED_16BIT( IVCS_s )
      CALL SEND_SIGNED_16BIT( IVTAB_s )
      CALL SEND_REAL_32BIT( TATAB_s )
      CALL SEND_REAL_32BIT( TBURNM_s )
      CALL SEND_REAL_32BIT( TIMONV_s )
      CALL SEND_REAL_32BIT( TOFFLT_s(01) )
      CALL SEND_REAL_32BIT( TOFFLT_s(02) )
      CALL SEND_REAL_32BIT( TOFFLT_s(03) )
      CALL SEND_REAL_32BIT( TOFFLT_s(04) )
      CALL SEND_REAL_32BIT( TVTAB_s )

C----- COMMUNICATION WITH P02 -----C
      CALL RECEIVE_SIGNED_16BIT( IACSON )

C----- COMMUNICATE WITH CORVEL -----C
      CALL RECEIVE_REAL_32BIT( VG(01) )
      CALL RECEIVE_REAL_32BIT( VG(02) )
      CALL RECEIVE_REAL_32BIT( VG(03) )

C----- DAISY CHAIN WITH IMUPRO AND NAVIG -----C
      CALL RECEIVE_REAL_32BIT( TI2M(1) )
      CALL RECEIVE_REAL_32BIT( TI2M(2) )
      CALL RECEIVE_REAL_32BIT( TI2M(3) )
      CALL RECEIVE_REAL_32BIT( TI2M(4) )
      CALL RECEIVE_REAL_32BIT( TI2M(5) )
      CALL RECEIVE_REAL_32BIT( TI2M(6) )
      CALL RECEIVE_REAL_32BIT( TI2M(7) )
      CALL RECEIVE_REAL_32BIT( TI2M(8) )
      CALL RECEIVE_REAL_32BIT( TI2M(9) )

      CALL RECEIVE_REAL_32BIT( VREL(1) )
      CALL RECEIVE_REAL_32BIT( VREL(2) )
      CALL RECEIVE_REAL_32BIT( VREL(3) )

```

```
CALL RECEIVE_REAL_32BIT( RREL(1) )
CALL RECEIVE_REAL_32BIT( RREL(2) )
CALL RECEIVE_REAL_32BIT( RREL(3) )
```

```
CALL RECEIVE_REAL_32BIT( SP )
CALL RECEIVE_REAL_32BIT( SQ )
CALL RECEIVE_REAL_32BIT( SR )
```

```
call send_real_32bit( magr )
call send_real_32bit( magv )
call send_real_32bit( tgo )
call send_real_32bit( piter )
call send_real_32bit( roller )
call send_real_32bit( yawer )
call send_signed_16bit( iburn1 )
call send_real_32bit( lamd(1) )
call send_real_32bit( lamd(2) )
call send_signed_16bit( acqd )
```

```
call receive_signed_16bit( estate )
call receive_real_32bit( piter )
call receive_real_32bit( roller )
call receive_real_32bit( yawer )
call receive_signed_16bit( iburn1 )
call receive_real_32bit( lamd(1) )
call receive_real_32bit( lamd(2) )
call receive_signed_16bit( acqd )
call receive_real_32bit( tgel )
call receive_real_32bit( tge2a1 )
call receive_real_32bit( trmtgo )
```

```
C-----C
C----- AUTOPILOTS -----C
C-----C
C-----C
C-----C
```

```
IF ( TSTEP .GE. TAPUDRIV ) THEN
```

```
IF ( T.GE.TKVON ) THEN
```

```
C-----C
C----- KV AUTOPILOT MODULE -----C
C-----C
C      Calls the various ACS autopilot      C
C      modes used for controlling the      C
C      kill vehicle attitude during flight.  C
C      Its purpose is to define which      C
C      thruster to burn, for how long, and at C
C      what thrust level.                  C
C-----C
```

```
CALL KVAUTO(T, SP, SQ, SR, FLTCP, FLTCY, IXX, IYY, IZZ, ADISTT,
.          ROLLER, PITER, YAWER, TCWAIT, IDIST, SW80, TSAL, TSAH,
.          TNEXT, TLAPS, ANVP, DTSAMP, ACSLEV, TRATON, TPATON,
.          TYATON, ITHRES)
```

```
ENDIF
ENDIF
```

```
C----- COMMUNICATION WITH P00 -----C
```

```

CALL RECEIVE_REAL_32BIT( IXX )
CALL RECEIVE_REAL_32BIT( IYY )
CALL RECEIVE_REAL_32BIT( IZZ )
CALL RECEIVE_REAL_32BIT( MASS )

```

C----- COMMUNICATION WITH P00 -----C

```

CALL SEND_SIGNED_16BIT( IDROP_s )

```

C----- COMMUNICATION WITH P02 -----C

```

CALL SEND_REAL_32BIT( ACSLEV_s )
CALL SEND_REAL_32BIT( DTACSA_s(01) )
CALL SEND_REAL_32BIT( DTACSA_s(02) )
CALL SEND_REAL_32BIT( DTACSA_s(03) )
CALL SEND_REAL_32BIT( DTACSA_s(04) )
CALL SEND_REAL_32BIT( DTACSB_s(01) )
CALL SEND_REAL_32BIT( DTACSB_s(02) )
CALL SEND_REAL_32BIT( DTACSB_s(03) )
CALL SEND_REAL_32BIT( DTACSB_s(04) )
CALL SEND_REAL_32BIT( DTOFFV_s(01) )
CALL SEND_REAL_32BIT( DTOFFV_s(02) )
CALL SEND_REAL_32BIT( DTOFFV_s(03) )
CALL SEND_REAL_32BIT( DTOFFV_s(04) )
CALL SEND_SIGNED_16BIT( ITHRES_s )
CALL SEND_SIGNED_16BIT( IVCS_s )
CALL SEND_SIGNED_16BIT( IVTAB_s )
CALL SEND_REAL_32BIT( TATAB_s )
CALL SEND_REAL_32BIT( TBURNM_s )
CALL SEND_REAL_32BIT( TIMONV_s )
CALL SEND_REAL_32BIT( TOFFLT_s(01) )
CALL SEND_REAL_32BIT( TOFFLT_s(02) )
CALL SEND_REAL_32BIT( TOFFLT_s(03) )
CALL SEND_REAL_32BIT( TOFFLT_s(04) )
CALL SEND_REAL_32BIT( TVTAB_s )

```

C----- COMMUNICATION WITH P02 -----C

```

CALL RECEIVE_SIGNED_16BIT( IACSON )

```

C----- COMMUNICATE WITH CORVEL -----C

```

CALL RECEIVE_REAL_32BIT( VG(01) )
CALL RECEIVE_REAL_32BIT( VG(02) )
CALL RECEIVE_REAL_32BIT( VG(03) )

```

C----- DAISY CHAIN WITH IMUPRO AND NAVIG -----C

```

CALL RECEIVE_REAL_32BIT( TI2M(1) )
CALL RECEIVE_REAL_32BIT( TI2M(2) )
CALL RECEIVE_REAL_32BIT( TI2M(3) )
CALL RECEIVE_REAL_32BIT( TI2M(4) )
CALL RECEIVE_REAL_32BIT( TI2M(5) )
CALL RECEIVE_REAL_32BIT( TI2M(6) )
CALL RECEIVE_REAL_32BIT( TI2M(7) )
CALL RECEIVE_REAL_32BIT( TI2M(8) )
CALL RECEIVE_REAL_32BIT( TI2M(9) )

CALL RECEIVE_REAL_32BIT( VREL(1) )
CALL RECEIVE_REAL_32BIT( VREL(2) )
CALL RECEIVE_REAL_32BIT( VREL(3) )
CALL RECEIVE_REAL_32BIT( RREL(1) )
CALL RECEIVE_REAL_32BIT( RREL(2) )
CALL RECEIVE_REAL_32BIT( RREL(3) )

```

```

CALL RECEIVE_REAL_32BIT( SP )
CALL RECEIVE_REAL_32BIT( SQ )
CALL RECEIVE_REAL_32BIT( SR )

call send_real_32bit( magr )
call send_real_32bit( magv )
call send_real_32bit( tgo )
call send_real_32bit( piter )
call send_real_32bit( roller )
call send_real_32bit( yawer )
call send_signed_16bit( iburn1 )
call send_real_32bit( lamd(1) )
call send_real_32bit( lamd(2) )
call send_signed_16bit( acqd )

call receive_signed_16bit( estate )
call receive_real_32bit( piter )
call receive_real_32bit( roller )
call receive_real_32bit( yawer )
call receive_signed_16bit( iburn1 )
call receive_real_32bit( lamd(1) )
call receive_real_32bit( lamd(2) )
call receive_signed_16bit( acqd )
call receive_real_32bit( tge1 )
call receive_real_32bit( tge2al )
call receive_real_32bit( trmtgo )

C-----C
C----- AUTOPILOTS -----C
C-----C
C-----C
C-----C

IF ( TSTEP .GE. TAPUDRIV ) THEN

    TAPUDRIV = TAPUDRIV + TAPUSTEP

    IF ( T.GE.TKVON ) THEN

C-----C
C----- VCS LOGIC MODULE -----C
C-----C
C          Controls the kill vehicle velocity by      C
C          determining the appropriate VCS thruster   C
C          on and off times.                          C
C-----C

        CALL VCSLOG(T,MASS,LAMD,TGO,MAGV,TGIL,TRMTGO,TGE2AL,
        .           TGE1,VGM,IVCS,IDMEAS,IBURNM,MIDBRN,IBURN1,IBURN2,
        .           IBURN3,IDIST,FLTC,FLTCP,FLTCY,TSAL,TAH,TOFFLT,
        .           TOFLTM,TBURNP,TBURNY,TGE2,TGI1P,TGI2P,TGI3P,
        .           TGI1Y,TGI2Y,TGI3Y,TIMONV,TGOFLM,TCWAIT,DTVCSF,
        .           DTVCSY,DTOFFV,TBURNM)

C          SET FLAG TO COMPUTE VCS THRUSTER RESPONSE TABLE

        IVTAB = 1
        TVTAB = T

C-----C
C----- ACS RESOLVING LOGIC MODULE -----C
C-----C

```

```

      IF ( ITHRES.EQ.1 ) THEN
          CALL RESTHR(T, IDIST, ANVP, DTSAMP, TOFLTM, TRATON,
                    TPATON, TYATON, DTACSA, DTACSB)
C      BEGINNING TIME OF ACS THRUSTER RESPONSE TABLE
          TATAB = T
      ENDIF
  ENDIF
ENDIF

ithres_s = ithres
acslev_s = acslev

dtacsa_s(1) = dtacsa(1)
dtacsa_s(2) = dtacsa(2)
dtacsa_s(3) = dtacsa(3)
dtacsa_s(4) = dtacsa(4)
dtacsb_s(1) = dtacsb(1)
dtacsb_s(2) = dtacsb(2)
dtacsb_s(3) = dtacsb(3)
dtacsb_s(4) = dtacsb(4)
dtoffv_s(1) = dtoffv(1)
dtoffv_s(2) = dtoffv(2)
dtoffv_s(3) = dtoffv(3)
dtoffv_s(4) = dtoffv(4)
ivcs_s = ivcs
ivtab_s = ivtab
tatab_s = tatab
tburnm_s = tburnm
timonv_s = timonv
tofflt_s(1) = tofflt(1)
tofflt_s(2) = tofflt(2)
tofflt_s(3) = tofflt(3)
tofflt_s(4) = tofflt(4)
tvtab_s = tvtab

C-----C
C----- TERMINATION LOGIC -----C
C-----C
C          Defines the simulation termination      C
C          conditions                               C
C-----C

C      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )
          IEXIT = 0

C      ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
C      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED

      IF ( T.GE.TFINAL ) THEN
          IEXIT = 1
      ENDIF

C      increment time

      TSTEP = TSTEP + 5.0E0

```

```
T = TSTEP * DELT  
C  CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET  
  IF ( IEXIT.EQ.0 ) GO TO 1000  
  END
```

## B.1.13 Uup12.for

```

C      PROGRAM EXOSIM
C-----C
C-----C      Declare and initialize variables      C-----C
C-----C

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      real rdum
      CHARACTER*128 MESSAGE
C      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RKALMN / TKF      , IDRTOK , PP11      , PP12      , PP22      ,
      .                          PY11      , PY12      , PY22      , PLMDFP , YLMDFP ,
      .                          PLAMH      , YLAMH      , PLAMDH      , YLAMDH      , PLAMDF ,
      .                          YLAMDF , TGIL      , KFMODE , IFPAS

      REAL T,TSTEP,DELT
      REAL TMSUDRIV,TIMUDRIV,TGPUDRIV,TAPUDRIV,TSPUDRIV,TKFUDRIV
      REAL TMSUSTEP,TIMUSTEP,TGPUSTEP,TAPUSTEP,TSPUSTEP,TKFUSTEP
      REAL FLTC(4)

      REAL dtacsa_s(4),dtacsb_s(4),dtoffv_s(4),tofflt_s(4)
      REAL ANGACL(3,4,10)
      REAL OMEGAI(3)      , GRLAST(3)
      REAL OMEGA0(3)      , AACCEL(3,4)

      INTEGER      FIRST1      , FIRST2
      INTEGER      ISEQ(4)      , IMCPAS(3,4) , FLIP

C      OUTPUTS

      REAL MAGRTR      , MAGR      , MASS
      REAL MAGV      , MGRDOT      , CMS(9)
      REAL ADISTT(4,3) , LAMDXX(2)
      REAL LAMSEK(2)   , LAMM(2)
      REAL RREL(3)     , URREL(3)
      REAL VREL(3)     , TI2M(9)
      REAL QS1(4)      , VMI(3)      , RMI(3)
      REAL VTEST(3) , VMIR(3) , GRTEST(3)
      REAL AT(3)
      REAL CIE(9)

C      NAMELIST INPUTS

      REAL IXX      , IYY      , IZZ
      REAL CG(3)    , DTOFFV(4) , VG(3)
      REAL DTACSA(4) , DTACSB(4)
      REAL XYZE(3)   , XYZED(3)
      REAL GRT(5,3)  , VTIC(5,3) , rtic(5,3)
      REAL PULSEA(3) , PULSEG(3)
      REAL LAM(2)    , LAMD(2)   , VGM(3)
      REAL DTVCSF(3) , DTVCSY(3)
      REAL TOFFLT(4) , LATLP      , LONGLP

      INTEGER      SEKTYP      , ACQD
      INTEGER      TERM      , TOSEED      , ESTATE
      INTEGER      TRACK

      REAL RTEST(3)      , RMIR(3)

```

## \* DATA INITIALIZATION

```

$INCLUDE('^/INCLUDE/SSDATA35.DAT')
$INCLUDE('^/INCLUDE/SSDATA38.DAT')
$INCLUDE('^/INCLUDE/SSDATA39.DAT')
$INCLUDE('^/INCLUDE/SSDATA42.DAT')
$INCLUDE('^/INCLUDE/SSDATA44.DAT')
$INCLUDE('^/INCLUDE/SSDATA45.DAT')
$INCLUDE('^/INCLUDE/SSDATA46.DAT')
$INCLUDE('^/INCLUDE/SSDATA47.DAT')
$INCLUDE('^/INCLUDE/SSDATA48.DAT')
$INCLUDE('^/INCLUDE/SSDATA49.DAT')
$INCLUDE('^/INCLUDE/SSDATA50.DAT')
$INCLUDE('^/INCLUDE/SSDATA01.DAT')
$INCLUDE('^/INCLUDE/SSDATA17.DAT')
$INCLUDE('^/INCLUDE/SSDATA18.DAT')
$INCLUDE('^/INCLUDE/SSDATA21.DAT')
$INCLUDE('^/INCLUDE/SSDATA22.DAT')
$INCLUDE('^/INCLUDE/SSDATA23.DAT')
$INCLUDE('^/INCLUDE/SSDATA28.DAT')
$INCLUDE('^/INCLUDE/SSDATA29.DAT')
$INCLUDE('^/INCLUDE/SSDATA30.DAT')
$INCLUDE('^/INCLUDE/SSDATA71.DAT')
$INCLUDE('^/INCLUDE/SSTIMING.DAT')
$INCLUDE(':pfp:INCLUDE/target.for')

```

```

* INITIALIZE 80x87
  CALL CW87

```

```

$INCLUDE('SS_12.DAT')

```

```

  idrop_s = idrop
  acslev_s = acslev
  dtacsa_s(1) = dtacsa(1)
  dtacsa_s(2) = dtacsa(2)
  dtacsa_s(3) = dtacsa(3)
  dtacsa_s(4) = dtacsa(4)
  dtacsb_s(1) = dtacsb(1)
  dtacsb_s(2) = dtacsb(2)
  dtacsb_s(3) = dtacsb(3)
  dtacsb_s(4) = dtacsb(4)
  dtoffv_s(1) = dtoffv(1)
  dtoffv_s(2) = dtoffv(2)
  dtoffv_s(3) = dtoffv(3)
  dtoffv_s(4) = dtoffv(4)
  ithres_s = ithres
  ivcs_s = ivcs
  ivtab_s = ivtab
  tatab_s = tatab
  tburnm_s = tburnm
  timonv_s = timonv
  tofflt_s(1) = tofflt(1)
  tofflt_s(2) = tofflt(2)
  tofflt_s(3) = tofflt(3)
  tofflt_s(4) = tofflt(4)
  tvtab_s = tvtab

```

```

C-----C
C----- MAIN EXECUTION LOOP -----C
C-----C
C           Execution of all events is performed   C
C           wit'in this loop                       C
C-----C

```

```

1000 CONTINUE

C      WRITE(*,*) '-----BEGINNING OF LOOP-----'

C-----C
C----- Processor communication -----C
C-----C

C----- COMMUNICATION WITH SEEKER -----C

      call receive_real_32bit( lamm(1) )
      call receive_real_32bit( lamm(2) )
      call receive_real_32bit( snr )
      call receive_real_32bit( frmrat )

C----- COMMUNICATION WITH P03 -----C

      CALL RECEIVE_SIGNED_16BIT( IRESLV )
      CALL RECEIVE_REAL_32BIT( LAMDXX(01) )
      CALL RECEIVE_REAL_32BIT( LAMDXX(02) )
      call receive_real_32bit( lamsek(1) )
      call receive_real_32bit( lamsek(2) )
      call receive_real_32bit( magrtr )

C----- DAISY CHAIN WITH IMUPRO AND NAVIG -----C

      CALL RECEIVE_REAL_32BIT( TI2M(1) )
      CALL RECEIVE_REAL_32BIT( TI2M(2) )
      CALL RECEIVE_REAL_32BIT( TI2M(3) )
      CALL RECEIVE_REAL_32BIT( TI2M(4) )
      CALL RECEIVE_REAL_32BIT( TI2M(5) )
      CALL RECEIVE_REAL_32BIT( TI2M(6) )
      CALL RECEIVE_REAL_32BIT( TI2M(7) )
      CALL RECEIVE_REAL_32BIT( TI2M(8) )
      CALL RECEIVE_REAL_32BIT( TI2M(9) )

      CALL RECEIVE_REAL_32BIT( VREL(1) )
      CALL RECEIVE_REAL_32BIT( VREL(2) )
      CALL RECEIVE_REAL_32BIT( VREL(3) )
      CALL RECEIVE_REAL_32BIT( RREL(1) )
      CALL RECEIVE_REAL_32BIT( RREL(2) )
      CALL RECEIVE_REAL_32BIT( RREL(3) )

C-----C
C----- KALMAN FILTER MODULE -----C
C-----C
C      Filter LOS angles      C
C      C      C
C-----C

      call receive_real_32bit( magr )
      call receive_real_32bit( magv )
      call receive_real_32bit( tgo )
      call receive_real_32bit( piter )
      call receive_real_32bit( roller )
      call receive_real_32bit( yawer )
      call receive_signed_16bit( iburn1 )
      call receive_real_32bit( lamd(1) )
      call receive_real_32bit( lamd(2) )
      call receive_signed_16bit( acqd )

      IF ( TSTEP .GE. TKFUDRIV ) THEN

```

```

C      TKFUDRIV = TKFUDRIV + TKFUSTEP
      TKFUDRIV = TKFUDRIV + int(1000.0/frmrat)

C      write(message,103)t
C103    format(' kalman',f10.4)
C      call outmes(message)

C      CALL FILTER IF SNR IS SUFFICIENT

      IF ( SNR.GE.SNRACQ .OR. SEKTYP.NE.2 ) THEN

        IF (SEKTYP.EQ.1 .OR. SEKTYP.EQ.2) THEN
          ASIG = (32.56*SNR**(-0.29912))*1.0E-6
        ENDIF

        CALL KALMAN(T, TI2M, LAMM, ASIG, SNR, TGO, RREL, VREL,
.          TI2M, RACQ, MAGRTR, MAGR, MAGV, LAMSEK, LAMDXX, FRMRAT, CMS,
.          MACQ, MCSO, MTERM, IRESLV, TRACK, TERM, TRMTGO, TGE1,
.          TGE2AL, WFILT, ZFILT, LAM, LAMD, IBURN1, ACQD, ESTATE,
.          PITER, YAWER, ROLLER)

      ENDIF
    ENDIF

    call send_signed_16bit( estate )
    call send_real_32bit( piter )
    call send_real_32bit( roller )
    call send_real_32bit( yawer )
    call send_signed_16bit( iburn1 )
    call send_real_32bit( lamd(1) )
    call send_real_32bit( lamd(2) )
    call send_signed_16bit( acqd )
    call send_real_32bit( tge1 )
    call send_real_32bit( tge2al )
    call send_real_32bit( trmtgo )

C-----C
C----- TERMINATION LOGIC -----C
C-----C
C      Defines the simulation termination    C
C      conditions                            C
C-----C

C      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT )

      IEXIT = 0

C      ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
C      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED

      IF ( T.GE.TFINAL ) THEN
        IEXIT = 1
      ENDIF

C      increment time

      TSTEP = TSTEP + 1.0E0
      T = TSTEP * DELT

C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET

      IF ( IEXIT.EQ.0 ) GO TO 1000

```

END

B.2 Utilities (FORTRAN)

B.2.1 Sskvauto.for

Omitted - Classified

### B.2.2 Ssvcslog.for

Omitted - Classified

## B.2.3 Uuaccel.1.r

```

C-----
C      SUBROUTINE ACCEL (T,UD,VD,WD,P,Q,R,PD,QD,RD,CG,CIM,XD,YD,ZD,GR,
C      .                GYSEED,QFRACA,PULSEA)
C-----
C
C      SUBROUTINE NAME :      ACCEL
C
C      AUTHOR(S) :          D. C. FOREMAN
C
C      FUNCTION :           ACCELEROMETER MODEL COMPUTES SENSED DELTA
C                           VELOCITY COUNTS.  INCLUDES ROTATIONAL
C                           EFFECTS, AXIS MISALIGNMENT AND NONORTHOGON-
C                           ALITY ERRORS, SCALE FACTOR ERRORS, RANDOM
C                           AND CONSTANT DRIFT AND QUANTIZATION.
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NORM      ,  RESP2R
C
C      INPUTS :             T,UD,VD,WD,P,Q,R,PD,QD,RD,CG,CIM,XD,
C                           YD,ZD,GR
C
C      OUTPUTS :            NONE
C
C      BOTH :               GYSEED,QFRACA,PULSEA
C
C      UPDATES :            T. THORNTON - CR # 004
C                           T. THORNTON - CR # 016
C                           B. HILL     - CR # 020
C                           D. SMITH    - CR # 021
C                           B. HILL     - CR # 022
C                           B. HILL     - CR # 030
C                           T. THORNTON - CR # 037
C                           B. HILL     - CR # 038
C                           D. SMITH    - CR # 059
C                           D. SISSOM   - CR # 069
C                           D. SMITH    - CR # 070
C                           D. SMITH    - CR # 075
C                           D. SMITH    - CR # 076
C                           B. HILL /   - CR # 081
C                           R. RHYNE    - CR # 084
C                           R. RHYNE    - CR # 087
C                           B. HILL     - CR # 093
C-----

```

```

IMPLICIT DOUBLE PRECISION      (A-H)
IMPLICIT DOUBLE PRECISION      (O-Z)

DOUBLE PRECISION  ABI0(3)      ,  ABI1(3)      ,  ABI2(3)
DOUBLE PRECISION  ABO0(3)      ,  ABO1(3)      ,  ABO2(3)
DOUBLE PRECISION  CG(3)        ,  CIM(9)        ,  DCA(3)
DOUBLE PRECISION  DUM1(3)      ,  DUM2(3)      ,  DUM3(3)
DOUBLE PRECISION  DVEL(3)      ,  GR(3)         ,  GRAVC(3)
DOUBLE PRECISION  GRLST(3)     ,  LIMU(3)       ,  PULSEA(3)
DOUBLE PRECISION  QFRACA(3)    ,  SF1A(3)       ,  SF2A(3)
DOUBLE PRECISION  SFEA(3)      ,  WDRA(3)
DOUBLE PRECISION  XIMU(3)      ,  XYZDP(3)

```

```

      INTEGER*4          GYSEED

C      LOCAL COMMON USED FOR CONSTANTS, LOCAL VARIABLES AND
C      INITIALIZATION FLAG

      SAVE              IACCEL

C      COMMON "RACCEL" USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RACCEL / DRSIGA, PSIA , THTA , PHIA , THXZA ,
      .                THXYA , THYZA , THYXA , THZYA , THZXA ,
      .                SF1A , SF2A , DCA , TOACCE , GRLST ,
      .                XYZDP , ABI2 , ABI1 , ABO2 , ABO1

* DATA INITIALIZATION
$include('^/include/ssdata15.dat')
$include('^/include/ssdata16.dat')

      DATA IACCEL / 1 /

      IF (IACCEL .EQ. 1) THEN

        IACCEL = 0

C      INITIALIZE ACCELEROMETER PARAMETERS

      IF ( T .EQ. 0.0 ) THEN
        DRSIGA = DRSGAI/(60.0*DSQRT(DTIMU))
        CALL NORM(ALNSGA,ALNMNA,GYSEED,PSIA)
        CALL NORM(ALNSGA,ALNMNA,GYSEED,THTA)
        CALL NORM(ALNSGA,ALNMNA,GYSEED,PHIA)
        CALL NORM(AORSGA,AORMNA,GYSEED,THXZA)
        CALL NORM(AORSGA,AORMNA,GYSEED,THXYA)
        CALL NORM(AORSGA,AORMNA,GYSEED,THYZA)
        CALL NORM(AORSGA,AORMNA,GYSEED,THYXA)
        CALL NORM(AORSGA,AORMNA,GYSEED,THZYA)
        CALL NORM(AORSGA,AORMNA,GYSEED,THZXA)
        CALL NORM(SF1SGA,SF1MNA,GYSEED,SF1A(1))
        CALL NORM(SF1SGA,SF1MNA,GYSEED,SF1A(2))
        CALL NORM(SF1SGA,SF1MNA,GYSEED,SF1A(3))
        CALL NORM(SF2SGA,SF2MNA,GYSEED,SF2A(1))
        CALL NORM(SF2SGA,SF2MNA,GYSEED,SF2A(2))
        CALL NORM(SF2SGA,SF2MNA,GYSEED,SF2A(3))
        CALL NORM(DCSIGA,DCMENA,GYSEED,DCA(1))
        CALL NORM(DCSIGA,DCMENA,GYSEED,DCA(2))
        CALL NORM(DCSIGA,DCMENA,GYSEED,DCA(3))
        DO 10 I = 1,3
          ABI2(I) = 0.0D0
          ABI1(I) = 0.0D0
          ABO2(I) = 0.0D0
          ABO1(I) = 0.0D0
10      CONTINUE
      ENDIF

C      COMPUTE SECOND ORDER RESPONSE DIFFERENCE EQUATION COEFFICIENTS

      IF ( IARTYP.EQ.2 ) THEN
        CALL RESP2R ( DTIMU,WACC,ZACC,CABI2,CABI1,CABIO,CABO2,
        .            CABO1,CABO0 )
      ENDIF
    ENDIF

C      CALCULATE TIME SINCE LAST CALL TO ACCEL

```

```

DTDEL = T - TOACCE
TOACCE = T

C   DETERMINE INERTIAL FRAME DELTA VELOCITY OVER PREVIOUS INTERVAL
WITH
C   GRAVITATIONAL CONTRIBUTION REMOVED

    IF ( DTDEL.NE.0.0D0 ) THEN
        GRAVG(1) = 0.5D0 * ( GR(1) + GRLST(1) )
        GRAVG(2) = 0.5D0 * ( GR(2) + GRLST(2) )
        GRAVG(3) = 0.5D0 * ( GR(3) + GRLST(3) )
        DLVXI   = XD - XYZDP(1) - DTDEL*GRAVG(1)
        DLVYI   = YD - XYZDP(2) - DTDEL*GRAVG(2)
        DLVZI   = ZD - XYZDP(3) - DTDEL*GRAVG(3)
    ENDIF

C   SAVE GRAVITY VECTOR FOR USE ON NEXT PASS

    GRLST(1) = GR(1)
    GRLST(2) = GR(2)
    GRLST(3) = GR(3)

C   ROTATE DELTA VELOCITY INTO MISSILE FRAME

    IF ( DTDEL.NE.0.0D0 ) THEN
        DLVXB = CIM(1)*DLVXI + CIM(4)*DLVYI + CIM(7)*DLVZI
        DLVYB = CIM(2)*DLVXI + CIM(5)*DLVYI + CIM(8)*DLVZI
        DLVZB = CIM(3)*DLVXI + CIM(6)*DLVYI + CIM(9)*DLVZI
    ENDIF

C   CONVERT DELTA VELOCITY TO AVERAGE ACCELERATION

    IF ( DTDEL.NE.0.0D0 ) THEN
        UDAVG = DLVXB / DTDEL
        VDAVG = DLVYB / DTDEL
        WDAVG = DLVZB / DTDEL
    ELSE
        UDAVG = UD
        VDAVG = VD
        WDAVG = WD
    ENDIF

C   SAVE PREVIOUS INERTIAL FRAME VELOCITY

    XYZDP(1) = XD
    XYZDP(2) = YD
    XYZDP(3) = ZD

C   SENSOR ACCELERATION DUE TO PACKAGE OFFSET FROM THE CG

    IF ( IMUOFF.EQ.0 ) THEN
        UDR = UDAVG
        VDR = VDAVG
        WDR = WDAVG
    ELSE
        XIMU(1) = CG(1) - LIMU(1)
        XIMU(2) = CG(2) - LIMU(2)
        XIMU(3) = CG(3) - LIMU(3)

        DUM1(1) = QD*XIMU(3) - RD*XIMU(2)
        DUM1(2) = RD*XIMU(1) - PD*XIMU(3)
        DUM1(3) = PD*XIMU(2) - QD*XIMU(1)

        DUM2(1) = Q*XIMU(3) - R*XIMU(2)

```

```

DUM2(2) = R*XIMU(1) - P*XIMU(3)
DUM2(3) = P*XIMU(2) - Q*XIMU(1)

DUM3(1) = Q*DUM2(3) - R*DUM2(2)
DUM3(2) = R*DUM2(1) - P*DUM2(3)
DUM3(3) = P*DUM2(2) - Q*DUM2(1)

UDR      = UDAVG + DUM1(1) + DUM3(1)
VDR      = VDAVG + DUM1(2) + DUM3(2)
WDR      = WDAVG + DUM1(3) + DUM3(3)
ENDIF

C  ACCELEROMETER AXIS MISALIGNMENT EFFECTS

UDM      =  UDR      + VDR*PSIA - WDR*THTA
VDM      = - UDR*PSIA + VDR      + WDR*PHIA
WDM      =  UDR*THTA - VDR*PHIA + WDR

C  ACCELEROMETER AXIS NONORTHOGONALITY EFFECTS

UDN      =  UDM      + VDM*THXZA - WDM*THXYA
VDN      = - UDM*THYZA + VDM      + WDM*THYXA
WDN      =  UDM*THZYA - VDM*THZXA + WDM

C  ADD LINEAR AND QUADRATIC SCALE FACTOR ERRORS

SFEA(1) = UDN + SF1A(1)*UDN + SF2A(1)*UDN**2
SFEA(2) = VDN + SF1A(2)*VDN + SF2A(2)*VDN**2
SFEA(3) = WDN + SF1A(3)*WDN + SF2A(3)*WDN**2

C  FOR EACH AXIS ...

DO 20 I=1,3

C  MAKE A GAUSSIAN DRAW FOR RANDOM DRIFT AND ADD TO CONSTANT DRIFT

IF ( DRSIGA.GT.0.0D0 ) THEN
  CALL NORM(DRSIGA,DRMENA,GYSEED,DRA)
ENDIF

WDRA(I) = DRA + DCA(I)

C  COMPUTE INPUT TO ACCELEROMETER RESPONSE MODEL

ABI0(I) = SFEA(I) + WDRA(I)

C  SECOND ORDER RESPONSE MODEL

IF ( IARTYP.EQ.2 ) THEN
  ABO0(I) = ( CABI0*ABI0(I) + CABI1*ABI1(I)
             + CABI2*ABI2(I) - CABO1*ABO1(I)
             - CABO2*ABO2(I) )/CABO0
  ABI2(I) = ABI1(I)
  ABI1(I) = ABI0(I)
  ABO2(I) = ABO1(I)
  ABO1(I) = ABO0(I)
ENDIF

C  INSTANTANEOUS RESPONSE MODEL

IF ( IARTYP.EQ.0 ) THEN
  ABO0(I) = ABI0(I)
ENDIF

```

```

C      COMPUTE SENSED DELTA VELOCITY
      DVEL(I) = DTDEL * ABOO(I)
      IF ( SPPA.GT.0.0 ) THEN
C      UNQUANTIZED OUTPUT IN COUNTS
      QFRACA(I) = QFRACA(I) - PULSEA(I) + DVEL(I)/SPPA
C      QUANTIZED OUTPUT IN COUNTS
      PULSEA(I) = DINT(QFRACA(I))
      ELSE
      PULSEA(I) = DVEL(I)
      ENDIF
20 CONTINUE
      RETURN
      END

```

## B.2.4 Uuacsth2.for

```

C-----
C      SUBROUTINE ACSTHR2 (ITHRES)
C-----
C
C      SUBROUTINE NAME :      ACSTHR2
C
C      AUTHOR   ) :          B. HILL
C
C      FUNCTION  :          RESOLVES THE ACS THRUSTER BURN TIMES INTO
C                          THE APPROPRIATE FORCES AND MOMENTS
C
C      CALLED FROM :          FORTRAN MAIN
C
C      SUBROUTINES CALLED :   none
C
C      BOTH :                ITHRES
C
C      UPDATES :
C          D. SISSOM      - CR # 017
C          D. SISSOM      - CR # 032
C          B. HILL        - CR # 038
C          T. THORNTON    - CR # 043
C          B. HILL        - CR # 051
C          D. SMITH       - CR # 059
C          D. SISSOM      - CR # 069
C          D. SMITH       - CR # 074
C          D. SMITH       - CR # 076
C          D. SMITH       - CR # 080
C          B. HILL /      - CR # 081
C          R. RHYNE
C          D. SMITH       - CR # 082
C          R. RHYNE       - CR # 083
C          R. RHYNE       - CR # 084
C          B. HILL        - CR # 086
C          R. RHYNE       - CR # 087
C          B. HILL        - CR # 089
C          B. HILL        - CR # 093
C-----
C
C      IMPLICIT REAL      (A-H)
C      IMPLICIT REAL      (O-Z)
C
C
C      IF (ITHRES .EQ. 1) THEN
C
C          ITHRES = 0
C
C      ENDIF
C
C      RETURN
C      END

```

## B.2.5 Uuacsthr.for

```

C-----
C      SUBROUTINE ACSTHR(T,CG,ACSLEV,DTACSA,DTACSB,TATAB,TOSEED,TBRK,
C      .               ITHRES,FXACS,FYACS,FZACS,MXACS,MYACS,MZACS,
C      .               MDOTA,IACSON,TIMONA)
C-----
C
C      SUBROUTINE NAME :      ACSTHR
C
C      AUTHOR(S) :          B. HILL
C
C      FUNCTION :           RESOLVES THE ACS THRUSTER BURN TIMES INTO
C                           THE APPROPRIATE FORCES AND MOMENTS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NORM, TABLE
C
C      INPUTS :             T,CG,ACSLEV,DTACSA,DTACSB,TATAB
C
C      OUTPUTS :            FXACS,FYACS,FZACS,MXACS,MYACS,MZACS,MDOTA,
C                           IACSON,TIMONA
C
C      BOTH :              TOSEED,TBRK,ITHRES
C
C      UPDATES :            D. SISSOM - CR # 017
C                           D. SISSOM - CR # 032
C                           B. HILL - CR # 038
C                           T. THORNTON - CR # 043
C                           B. HILL - CR # 051
C                           D. SMITH - CR # 059
C                           D. SISSOM - CR # 069
C                           D. SMITH - CR # 074
C                           D. SMITH - CR # 076
C                           D. SMITH - CR # 080
C                           B. HILL / - CR # 081
C                           R. RHYNE
C                           D. SMITH - CR # 082
C                           R. RHYNE - CR # 083
C                           R. RHYNE - CR # 084
C                           B. HILL - CR # 086
C                           R. RHYNE - CR # 087
C                           B. HILL - CR # 089
C                           B. HILL - CR # 093
C-----

```

```

IMPLICIT REAL      (A-H)
IMPLICIT REAL      (O-Z)

REAL  ACSDIR(3,4) , ACSLOC(3,4) , ACSMA(9,4)
REAL  AOFF1(4) , AOFF2(4) , ATHRA(4)
REAL  ATHRB(4) , CG(3) , DTACSA(4)
REAL  DTACSB(4) , F(3) , F0(3)
REAL  ISPACS , M(3) , MDOTA
REAL  MXACS , MYACS , MZACS
REAL  THACSA(8,4) , THACSB(8,4) , TMACSA(8,4)
REAL  TMACSB(8,4) , XMOM(3)

INTEGER  INDXA(4)
INTEGER  INDXB(4)
INTEGER  LENA(4) , LENB(4)

```

```

      INTEGER*4          TOSEED

C      LOCAL COMMON USED FOR CONSTANTS, LOCAL VARIABLES AND
C      INITIALIZATION FLAG

      SAVE              IACSTH , ACSMA

C      COMMON "RACSTR" USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RACSTR / TREFLA , TLSTC , ACSF , AOFF1 , AOFF2 ,
      .              TMACSA , THACSA , LENA , TMACSB , THACSB ,
      .              LENB

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSDATA01.DAT')
$INCLUDE('^/INCLUDE/SSDATA02.DAT')
$INCLUDE('^/INCLUDE/SSDATA03.DAT')
$INCLUDE('^/INCLUDE/SSDATA17.DAT')
$INCLUDE('^/INCLUDE/SSDATA18.DAT')
$INCLUDE('^/INCLUDE/SSDATA19.DAT')
$INCLUDE('^/INCLUDE/SSDATA20.DAT')

      DATA IACSTH / 1 /

      IF ( IACSTH.EQ.1 ) THEN

        IACSTH = 0

        IF (T .LT. TKVON+EPSL) THEN

C          ACS MISALIGNMENT DIRECTIONS
C          AOFF1 = CONE ANGLE OFF NORMAL
C          AOFF2 = POLAR ANGLE

          CALL spNORM(AOFFSD,0.0e0,TOSEED,AOFF1(1))
          CALL spNORM(AOFFSD,0.0e0,TOSEED,AOFF1(2))
          CALL spNORM(AOFFSD,0.0e0,TOSEED,AOFF1(3))
          CALL spNORM(AOFFSD,0.0e0,TOSEED,AOFF1(4))

          AOFF2(1) = 2.0*PI*spRANO(TOSEED)
          AOFF2(2) = 2.0*PI*spRANO(TOSEED)
          AOFF2(3) = 2.0*PI*spRANO(TOSEED)
          AOFF2(4) = 2.0*PI*spRANO(TOSEED)

        ENDIF

      DO 300 I = 1 , 4
        CAOFF1 = COS(AOFF1(I))
        SAOFF1 = SIN(AOFF1(I))
        CAOFF2 = COS(AOFF2(I))
        SAOFF2 = SIN(AOFF2(I))
        ACSMA(1,I) = CAOFF1
        ACSMA(2,I) = SAOFF1*CAOFF2
        ACSMA(3,I) = SAOFF1*SAOFF2
        ACSMA(4,I) = SAOFF1*SAOFF2
        ACSMA(5,I) = CAOFF1
        ACSMA(6,I) = SAOFF1*CAOFF2
        ACSMA(7,I) = SAOFF1*CAOFF2
        ACSMA(8,I) = SAOFF1*SAOFF2
        ACSMA(9,I) = CAOFF1
300    CONTINUE

      ENDIF

```

```

C      RESET THE FORCE AND MOMENT COUNTERS TO ZERO

FXACS  = 0.0
FYACS  = 0.0
FZACS  = 0.0
MXACS  = 0.0
MYACS  = 0.0
MZACS  = 0.0
MDOTA  = 0.0

      IF (ITHRES .EQ. 1) THEN

C      CALCULATE TIME FOR PULSE TO COME ON AND TIME FOR PULSE TO
C      REACH FULL FORCE LEVEL

      TIMONA = TATAB + TLAGA
      TUPA = TIMONA + TRUPA

C      DETERMINE APPROPRIATE MAXIMUM THRUST LEVEL

      IF (ACSLEV .GT. 1.5) THEN
        ACSF = ACSFH
      ELSE
        ACSF = ACSFL
      ENDIF

C      DO 101 I=1,4

C      INITIALIZE TABLE POINTERS

      INDXA(I) = 1
      INDXB(I) = 1

C      CALCULATE THRUSTER RESPONSE TABLE FOR "A" THRUSTERS

      CALL spTABLE(TMACSA(1,I),THACSA(1,I),TATAB,THA1,LENA(I),
        INDXA(I))
      IF (DTACSA(I) .GE. TCMINA) THEN
        IF (THA1 .LT. EPSL) THEN

C          PREVIOUS VALVE STATE WAS LOW

          TMACSA(1,I) = TATAB
          THACSA(1,I) = 0.0
          TMACSA(2,I) = TIMONA
          THACSA(2,I) = 0.0
          TMACSA(3,I) = TUPA
          THACSA(3,I) = ACSF
          IPTR = 4
        ELSE
          CALL spTABLE(TMACSA(1,I),THACSA(1,I),TIMONA,THA2,
            LENA(I),INDXA(I))
          IF (THA2 .LT. EPSL) THEN

C          PREVIOUS VALVE STATE WAS EITHER DELAY OR RAMP,
C          AND NO CROSS-OVER IS PRESENT

          TMACSA(1,I) = TMACSA(LENA(I)-3,I)
          THACSA(1,I) = THACSA(LENA(I)-3,I)
          TMACSA(2,I) = TMACSA(LENA(I)-2,I)
          THACSA(2,I) = THACSA(LENA(I)-2,I)
          TMACSA(3,I) = TMACSA(LENA(I)-1,I)
          THACSA(3,I) = THACSA(LENA(I)-1,I)
          TMACSA(4,I) = TIMONA
        
```

```

      THACSA(4,I) = 0.0
      TMACSA(5,I) = TUPA
      THACSA(5,I) = ACSF
      IPTR = 6
    ELSE
      CALL spTABLE(TMCA(1,I),THACSA(1,I),TUPA,THA3,
                  LENA(I),INDXA(I))
      IF (THA3 .GE. (ACSF-EPST)) THEN
C          PREVIOUS VALVE STATE WAS HIGH

          TMACSA(1,I) = TATAB
          THACSA(1,I) = ACSF
          IPTR = 2
        ELSE
C          PREVIOUS VALVE STATE WAS DELAY, AND A
C          CROSS-OVER CONDITION HAS OCCURED

          TMACSA(1,I) = TMACSA(LENA(I)-3,I)
          THACSA(1,I) = THACSA(LENA(I)-3,I)
          TMACSA(2,I) = TMACSA(LENA(I)-2,I)
          THACSA(2,I) = THACSA(LENA(I)-2,I)
          TMACSA(3,I) = (TMACSA(LENA(I)-1,I) + TIMONA)/2.0
          THACSA(3,I) = (TMACSA(3,I) - TIMONA)*ACSF/TRDNA
          TMACSA(4,I) = TUPA
          THACSA(4,I) = ACSF
          IPTR = 5
        ENDIF
      ENDIF
    ENDIF
    TMACSA(IPTR,I) = TIMONA + DIACSA(I)
    THACSA(IPTR,I) = ACSF
    TMACSA(IPTR+1,I) = TMACSA(IPTR,I) + TRDNA
    THACSA(IPTR+1,I) = 0.0
    TMACSA(IPTR+2,I) = 999.0
    THACSA(IPTR+2,I) = 0.0
    LENA(I) = IPTR+2
  ELSE
C    MAKE SURE VALVE IS OFF

    IF (THA1 .LT. EPST) THEN
C      PREVIOUS VALVE STATE WAS LOW

      TMACSA(1,I) = TATAB
      THACSA(1,I) = 0.0
      TMACSA(2,I) = 999.0
      THACSA(2,I) = 0.0
      LENA(I) = 2
    ELSE
      CALL spTABLE(TMCA(1,I),THACSA(1,I),TUPA,THA3,
                  LENA(I),INDXA(I))
      IF (THA3 .LT. EPST) THEN
C          PREVIOUS VALVE STATE WAS EITHER DELAY OR RAMP, WITH
C          NO CROSSOVER POSSIBLE

          TMACSA(1,I) = TMACSA(LENA(I)-3,I)
          THACSA(1,I) = THACSA(LENA(I)-3,I)
          TMACSA(2,I) = TMACSA(LENA(I)-2,I)
          THACSA(2,I) = THACSA(LENA(I)-2,I)
          TMACSA(3,I) = TMACSA(LENA(I)-1,I)

```

```

      THACSA(3,I) = THACSA(LENA(I)-1,I)
      TMACSA(4,I) = 999.0
      THACSA(4,I) = 0.0
      LENA(I) = 4
ELSE
C      PREVIOUS VALVE STATE WAS DELAY, AND CROSSOVER COULD
C      OCCUR

      TMACSA(1,I) = TATAB
      THACSA(1,I) = ACSF
      TMACSA(2,I) = TIMONA
      THACSA(2,I) = ACSF
      TMACSA(3,I) = TIMONA + TRDNA
      THACSA(3,I) = 0.0
      TMACSA(4,I) = 999.0
      THACSA(4,I) = 0.0
      LENA(I) = 4
ENDIF
ENDIF
ENDIF
C      CALCULATE THRUSTER RESPONSE TABLE FOR "B" THRUSTERS
CALL spTABLE(TMCSB(1,I),THACSB(1,I),TATAB,THB1,LENB(I),
             INDXB(I))
IF (DTACSB(I) .GE. TCMINA) THEN
  IF (THB1 .LT. EPSL) THEN
C      PREVIOUS VALVE STATE WAS LOW

      TMCSB(1,I) = TATAB
      THACSB(1,I) = 0.0
      TMCSB(2,I) = TIMONA
      THACSB(2,I) = 0.0
      TMCSB(3,I) = TUPA
      THACSB(3,I) = ACSF
      IPTR = 4
ELSE
      CALL spTABLE(TMCSB(1,I),THACSB(1,I),TIMONA,THB2,
                  LENB(I),INDXB(I))
      IF (THB2 .LT. EPSL) THEN
C      PREVIOUS VALVE STATE WAS EITHER DELAY OR RAMP,
C      AND NO CROSS-OVER IS PRESENT

      TMCSB(1,I) = TMCSB(LENB(I)-3,I)
      THACSB(1,I) = THACSB(LENB(I)-3,I)
      TMCSB(2,I) = TMCSB(LENB(I)-2,I)
      THACSB(2,I) = THACSB(LENB(I)-2,I)
      TMCSB(3,I) = TMCSB(LENB(I)-1,I)
      THACSB(3,I) = THACSB(LENB(I)-1,I)
      TMCSB(4,I) = TIMONA
      THACSB(4,I) = 0.0
      TMCSB(5,I) = TUPA
      THACSB(5,I) = ACSF
      IPTR = 6
ELSE
      CALL spTABLE(TMCSB(1,I),THACSB(1,I),TUPA,THB3,
                  LENB(I),INDXB(I))
      IF (THB3 .GE. (ACSF-EPSL)) THEN
C      PREVIOUS VALVE STATE WAS HIGH

```

```

      TMACSB(1,I) = TATAB
      THACSB(1,I) = ACSF
      IPTR = 2
    ELSE
C      PREVIOUS VALVE STATE WAS DELAY, AND A
C      CROSS-OVER CONDITION HAS OCCURED

      TMACSB(1,I) = TMACSB(LENB(I)-3,I)
      THACSB(1,I) = THACSB(LENB(I)-3,I)
      TMACSB(2,I) = TMACSB(LENB(I)-2,I)
      THACSB(2,I) = THACSB(LENB(I)-2,I)
      TMACSB(3,I) = (TMACSB(LENB(I)-1,I) + TIMONA)
                    /2.0
      THACSB(3,I) = (TMACSB(3,I) - TIMONA)*ACSF/TRDNA
      TMACSB(4,I) = TUPA
      THACSB(4,I) = ACSF
      IPTR = 5
    ENDIF
  ENDIF
ENDIF
TMACSB(IPTR,I) = TIMONA + DTACSB(I)
THACSB(IPTR,I) = ACSF
TMACSB(IPTR+1,I) = TMACSB(IPTR,I) + TRDNA
THACSB(IPTR+1,I) = 0.0
TMACSB(IPTR+2,I) = 999.0
THACSB(IPTR+2,I) = 0.0
LENB(I) = IPTR+2
ELSE
C      MAKE SURE VALVE IS OFF

      IF (THB1 .LT. EPSL) THEN
C      PREVIOUS VALVE STATE WAS LOW

      TMACSB(1,I) = TATAB
      THACSB(1,I) = 0.0
      TMACSB(2,I) = 999.0
      THACSB(2,I) = 0.0
      LENB(I) = 2
    ELSE
      CALL spTABLE(TMACSB(1,I),THACSB(1,I),TUPA,THB3,
                  LENB(I),INDXB(I))
      IF (THB3 .LT. EPSL) THEN
C      PREVIOUS VALVE STATE WAS EITHER DELAY OR RAMP, WITH
C      NO CROSSOVER POSSIBLE

      TMACSB(1,I) = TMACSB(LENB(I)-3,I)
      THACSB(1,I) = THACSB(LENB(I)-3,I)
      TMACSB(2,I) = TMACSB(LENB(I)-2,I)
      THACSB(2,I) = THACSB(LENB(I)-2,I)
      TMACSB(3,I) = TMACSB(LENB(I)-1,I)
      THACSB(3,I) = THACSB(LENB(I)-1,I)
      TMACSB(4,I) = 999.0
      THACSB(4,I) = 0.0
      LENB(I) = 4
    ELSE
C      PREVIOUS VALVE STATE WAS DELAY, AND CROSSOVER COULD
C      OCCUR

      TMACSB(1,I) = TATAB

```

```

        THACSB(1,I) = ACSF
        TMACSB(2,I) = TIMONA
        THACSB(2,I) = ACSF
        TMACSB(3,I) = TIMONA + TRDNA
        THACSB(3,I) = 0.0
        TMACSB(4,I) = 999.0
        THACSB(4,I) = 0.0
        LENB(I) = 4
    ENDIF
  ENDIF
ENDIF
101  CONTINUE

ENDIF

C    SET REFERENCE TIME FOR TABLE LOOKUPS AND RESET ACS "ON" FLAG
TREF = T
IACSON = 0

C    CALCULATE AVERAGE THRUST LEVELS FOR EACH "A" THRUSTER
C    DURING NEXT CYCLE
DO 20 I = 1 , 4

C      INITIALIZE TABLE POINTER
INDXA(I) = 1

C      COMPUTE INSTANTANEOUS THRUST LEVEL VIA TABLE LOOKUP IF ACS "A"
C      CYCLE IS SCHEDULED FOR THIS THRUSTER . ALSO EXTRAPOLATE TIME OF
C      NEXT ACS "A" TABLE LOOKUP INDEX TRANSITION .
      IF ( TMACSA(1,I).GT.0.0e0 ) THEN
        CALL spTABLE(TMCA(1,I),THACSA(1,I),TREF,ATHRA(I),
                     LENA(I),INDXA(I))
        IF ( ATHRA(I) .GE. ACSF-EPSL ) IACSON = 1
      ELSE
        ATHRA(I) = 0.0e0
        INDXA(I) = 0
      ENDIF

C      CALCULATE THE FORCES AND MOMENTS PRODUCED BY THE "A"
C      ACS THRUSTERS :
      F(I) IS THE FORCE ALONG THE Ith AXIS.
      XMOM(I) IS THE EFFECTIVE MOMENT ARM.
      FORCES ARE ADJUSTED FOR MISALIGNMENT EFFECTS.
      THE MOMENT GENERATED IS ( F x XMOM ) .

      DO 10 J=1,3
        F0(J) = ACSDIR(J,I)*ATHRA(I)
        XMOM(J) = CG(J) - ACSLOC(J,I)
      CONTINUE
10    F(1) = ACSMA(1,I)*F0(1) +ACSMA(4,I)*F0(2) +ACSMA(7,I)*F0(3)
      F(2) = ACSMA(2,I)*F0(1) +ACSMA(5,I)*F0(2) +ACSMA(8,I)*F0(3)
      F(3) = ACSMA(3,I)*F0(1) +ACSMA(6,I)*F0(2) +ACSMA(9,I)*F0(3)

      M(1) = F(2)*XMOM(3) - F(3)*XMOM(2)
      M(2) = F(3)*XMOM(1) - F(1)*XMOM(3)
      M(3) = F(1)*XMOM(2) - F(2)*XMOM(1)

      FXACS = FXACS + F(1)
      FYACS = FYACS + F(2)
      FZACS = FZACS + F(3)

```

```

MXACS = MXACS + M(1)
MYACS = MYACS + M(2)
MZACS = MZACS + M(3)
MDOTA = MDOTA + ATHRA(I)/ISPACS
20 CONTINUE

C   CALCULATE AVERAGE THRUST LEVELS FOR EACH "B" THRUSTER
C   DURING NEXT CYCLE

DO 40 I = 1, 4

C   INITIALIZE TABLE POINTERS

INDXB(I) = 1

C   COMPUTE INSTANTANEOUS THRUST LEVEL VIA TABLE LOOKUP IF ACS "B"
C   CYCLE IS SCHEDULED FOR THIS THRUSTER. ALSO EXTRAPOLATE TIME OF
C   NEXT ACS "B" TABLE LOOKUP INDEX TRANSITION.

IF ( TMACSB(1,I).GT.0.0e0 ) THEN
    CALL spTABLE(TMACSB(1,I),THACSB(1,I),TREF,ATHRB(I),
                LENB(I),INDXB(I))
    IF ( ATHRB(I) .GE. ACSF-EPSL ) IACSON = 1
ELSE
    ATHRB(I) = 0.0e0
    INDXB(I) = 0
ENDIF

C   CALCULATE THE FORCES AND MOMENTS PRODUCED BY THE "B"
C   ACS THRUSTERS :
C       F(I) IS THE FORCE ALONG THE Ith AXIS.
C       XMOM(I) IS THE EFFECTIVE MOMENT ARM.
C       FORCES ARE ADJUSTED FOR MISALIGNMENT EFFECTS.
C       THE MOMENT GENERATED IS ( F x XMOM ).

DO 30 J=1,3
    F0(J) = -ACSDIR(J,I)*ATHRB(I)
    XMOM(J) = CG(J) - ACSLOC(J,I)
30 CONTINUE

F(1) = ACSMA(1,I)*F0(1) +ACSMA(4,I)*F0(2) +ACSMA(7,I)*F0(3)
F(2) = ACSMA(2,I)*F0(1) +ACSMA(5,I)*F0(2) +ACSMA(8,I)*F0(3)
F(3) = ACSMA(3,I)*F0(1) +ACSMA(6,I)*F0(2) +ACSMA(9,I)*F0(3)

M(1) = F(2)*XMOM(3) - F(3)*XMOM(2)
M(2) = F(3)*XMOM(1) - F(1)*XMOM(3)
M(3) = F(1)*XMOM(2) - F(2)*XMOM(1)

FXACS = FXACS + F(1)
FYACS = FYACS + F(2)
FZACS = FZACS + F(3)
MXACS = MXACS + M(1)
MYACS = MYACS + M(2)
MZACS = MZACS + M(3)
MDOTA = MDOTA + ATHRB(I)/ISPACS
40 CONTINUE

RETURN
END

```

## B.2.6 Uublkdat.for

```

C-----
  BLOCKDATA BLKDAT

  IMPLICIT DOUBLE PRECISION      (A-H)
  IMPLICIT DOUBLE PRECISION      (O-Z)

  COMMON / NORCOM / GSET , ISET

C
C  COMMON "RSPLAG" USED FOR MIDFLIGHT CAPABILITIES ONLY
C
  PARAMETER      (NSAVMX=10)

  DOUBLE PRECISION TLATCH(NSAVMX)
  DOUBLE PRECISION LAMMSV(2,NSAVMX)
  DOUBLE PRECISION RRELSV(3,NSAVMX)
  DOUBLE PRECISION VRELSV(3,NSAVMX)
  DOUBLE PRECISION TI2MSV(9,NSAVMX)
  DOUBLE PRECISION SNRSV(NSAVMX)

  COMMON / RSPLAG / NLATCH , TLATCH , LAMMSV , RRELSV , VRELSV ,
  .              TI2MSV , SNRSV

  DATA ISET / 0 /

  DATA NLATCH/0/
  END

```

## B.2.7 Uubrtavg.for

```

C-----
C      SUBROUTINE BRTAVG (TN,TA,DT,W)
C-----
C
C      SUBROUTINE NAME :      BRTAVG
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           Compute the average body rates over the last
C                          interval using the current and previous
C                          inertial to missile transformation matrices
C
C      CALLED FROM :        GYRO
C
C      SUBROUTINES CALLED :  M3X3I
C
C      INPUTS :             TN,TA,DT
C
C      OUTPUTS :            W
C
C      UPDATES :            D. SMITH      - CR # 076
C-----
C
C      IMPLICIT DOUBLE PRECISION      (A-H)
C      IMPLICIT DOUBLE PRECISION      (O-Z)
C
C      DOUBLE PRECISION  TN(9),      TA(9),      W(3)
C      DOUBLE PRECISION  TD(9),      TI(9),      TE(9)
C
C      COMPUTE INVERSE OF PREVIOUS TRANSFORMATION MATRIX
C
C      CALL M3X3I ( TA , TI )
C
C      COMPUTE DELTA ROTATION MATRIX FROM PREVIOUS MISSILE ATTITUDE TO
CURRENT
C      MISSILE ATTITUDE
C
C      TD(1) = TN(1)*TI(1) + TN(4)*TI(2) + TN(7)*TI(3)
C      TD(2) = TN(2)*TI(1) + TN(5)*TI(2) + TN(8)*TI(3)
C      TD(3) = TN(3)*TI(1) + TN(6)*TI(2) + TN(9)*TI(3)
C      TD(4) = TN(1)*TI(4) + TN(4)*TI(5) + TN(7)*TI(6)
C      TD(5) = TN(2)*TI(4) + TN(5)*TI(5) + TN(8)*TI(6)
C      TD(6) = TN(3)*TI(4) + TN(6)*TI(5) + TN(9)*TI(6)
C      TD(7) = TN(1)*TI(7) + TN(4)*TI(8) + TN(7)*TI(9)
C      TD(8) = TN(2)*TI(7) + TN(5)*TI(8) + TN(8)*TI(9)
C      TD(9) = TN(3)*TI(7) + TN(6)*TI(8) + TN(9)*TI(9)
C
C      DETERMINE DELTA EULER ANGLES FROM PREVIOUS ORIENTATION ( EULER
ROTATION
C      SEQUENCE IS PSI-THETA-PHI )
C
C      DLPSI = DATAN2 ( TD(4) , TD(1) )
C      DLTHE = DASIN ( -TD(7) )
C      DLPHI = DATAN2 ( TD(8) , TD(9) )
C
C      CDLPSI = DCOS ( DLPSI )
C      SDLPSI = DSIN ( DLPSI )
C      CDLTHE = DCOS ( DLTHE )
C      SDLTHE = DSIN ( DLTHE )
C      CDLPHI = DCOS ( DLPHI )

```

```

SDLPHI = DSIN ( DLPHI )

C   COMPUTE MATRIX RELATING EULER ANGULAR RATES TO BODY RATES ( [TE]
IS
C   USED FOR TEMPORARY STORAGE )

TE(1)  =  1.0D0
TE(2)  =  0.0D0
TE(3)  =  0.0D0
TE(4)  =  0.0D0
TE(5)  =  CDLPSI
TE(6)  =  - SDLPSI
TE(7)  =  - SDLTHE
TE(8)  =  CDLTHE*SDLPHI
TE(9)  =  CDLTHE*CDLPHI

C   ADD IDENTITY MATRIX TO [TE] AND INVERT THE RESULTANT MATRIX

TD(1)  =  TE(1) + 1.0D0
TD(2)  =  TE(2)
TD(3)  =  TE(3)
TD(4)  =  TE(4)
TD(5)  =  TE(5) + 1.0D0
TD(6)  =  TE(6)
TD(7)  =  TE(7)
TD(8)  =  TE(8)
TD(9)  =  TE(9) + 1.0D0

CALL M3X3I ( TD , TI )

C   CALCULATE AVERAGE BODY RATES OVER LAST INTERVAL

TD(1)  =  TI(1)*TE(1) + TI(4)*TE(2) + TI(7)*TE(3)
TD(2)  =  TI(2)*TE(1) + TI(5)*TE(2) + TI(8)*TE(3)
TD(3)  =  TI(3)*TE(1) + TI(6)*TE(2) + TI(9)*TE(3)
TD(4)  =  TI(1)*TE(4) + TI(4)*TE(5) + TI(7)*TE(6)
TD(5)  =  TI(2)*TE(4) + TI(5)*TE(5) + TI(8)*TE(6)
TD(6)  =  TI(3)*TE(4) + TI(6)*TE(5) + TI(9)*TE(6)
TD(7)  =  TI(1)*TE(7) + TI(4)*TE(8) + TI(7)*TE(9)
TD(8)  =  TI(2)*TE(7) + TI(5)*TE(8) + TI(8)*TE(9)
TD(9)  =  TI(3)*TE(7) + TI(6)*TE(8) + TI(9)*TE(9)

W(1)   =  2.0D0 * ( TD(1)*DLPHI + TD(4)*DLTHE + TD(7)*DLPSI ) / DT
W(2)   =  2.0D0 * ( TD(2)*DLPHI + TD(5)*DLTHE + TD(8)*DLPSI ) / DT
W(3)   =  2.0D0 * ( TD(3)*DLPHI + TD(6)*DLTHE + TD(9)*DLPSI ) / DT

RETURN
END

```

## B.2.8 Uucorvel.for

```

C-----
C      SUBROUTINE CORVEL (T,MVR,VTT,RMIR,VMIR,VTP,VG,VS,MVS,UVS,VC,DLV,
C      TFFE,TTFE)
C-----
C
C      SUBROUTINE NAME :      CORVEL
C
C      AUTHOR(S) :          M. K. DOUBLEDAY, L. C. HECK
C
C      FUNCTION :           CALCULATES THE CORRELATED VELOCITY
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             T,MVR,VTT,RMIR,VMIR
C
C      OUTPUTS :            VS,MVS,UVS,VC,DLV,TFFE,TTFE
C
C      BOTH :               VTP,VG
C
C      UPDATES :            T. THORNTON - CR # 025
C                          D. SMITH   - CR # 013
C                          B. HILL    - CR # 030
C                          T. THORNTON - CR # 033
C                          T. THORNTON - CR # 042
C                          T. THORNTON - CR # 043
C                          T. THORNTON - CR # 044
C                          D. SMITH   - CR # 059
C                          D. SMITH   - CR # 072
C                          B. HILL /  - CR # 081
C                          R. RHYNE   - CR # 093
C                          B. HILL    - CR # 093
C-----
C
C      IMPLICIT REAL        (A-H)
C      IMPLICIT REAL        (O-Z)
C
C      REAL  VMIR(3)        ,  RMIR(3)
C
C      REAL  DLV(3)          ,  MDVT          ,  MRB
C      REAL  MRT             ,  MTMPV         ,  MVCE
C      REAL  MVR             ,  MVS           ,  MVSE
C      REAL  RB(3)           ,  RTPRED(3)
C      REAL  TMPV(3)         ,  URB(3)         ,  URT(3)
C      REAL  UTHP(3)         ,  UTMPV(3)       ,  UVS(3)
C      REAL  VC(3)           ,  VCE(3)
C      REAL  VD0(3)          ,  VG(3)          ,  VGE(3)
C      REAL  VPHI(3)         ,  VS(3)
C      REAL  VSE(3)          ,  VTT(3)         ,  VTP(3)
C
C      LOCAL COMMON USED FOR CONSTANTS AND INITIALIZATION FLAG
C
C      SAVE                  ICORV
C
C      * DATA INITIALIZATION
C      $INCLUDE('~/INCLUDE/SSDATA39.DAT')
C      $INCLUDE('~/INCLUDE/SSDATA42.DAT')
C      $INCLUDE('~/INCLUDE/SSDATA43.DAT')
C      $INCLUDE('~/INCLUDE/SSDATA23.DAT')

```

```

DATA ICORV / 1 /

IF (ICORV .EQ. 1) THEN

    ICORV = 0

    IF (T .EQ. 0.0) THEN
        ILOOP = 50
    ELSE
        ILOOP = 1
    ENDIF
ELSE
    ILOOP = 1
ENDIF

C  ESTIMATE VELOCITY TO BE GAINED (VGE) , CORRELATED VELOCITY (VCE) ,
C  AND STEERING VELOCITY (VSE)

DO 10 I=1,3
    DLV(I) = VTT(I) - VTTP(I)
    VGE(I) = VG(I) - DLV(I)
    VCE(I) = VGE(I) + VMIR(I)
    VSE(I) = VGE(I) - VDO(I)
    VTTP(I) = VTT(I)
10 CONTINUE

MVSE = SQRT ( VSE(1)**2 + VSE(2)**2 + VSE(3)**2 )
MDVT = SQRT ( DLV(1)**2 + DLV(2)**2 + DLV(3)**2 )

C  CALCULATE POSITION BIAS SCALE FACTOR

IF ( MVSE.GT.MVR ) THEN
    SCALE3 = MVR/MVSE
ELSE
    SCALE3 = 1.0
END IF

SCALAR = F2 * MVR * SCALE3 / ( F1 + MDVT )

C  CALCULATE OFFSET POSITION VECTOR

IF ( T.GE.TSTG2 ) THEN
    RB(1) = RMIR(1)
    RB(2) = RMIR(2)
    RB(3) = RMIR(3)
ELSE
    RB(1) = RMIR(1) + SCALAR*VSE(1)
    RB(2) = RMIR(2) + SCALAR*VSE(2)
    RB(3) = RMIR(3) + SCALAR*VSE(3)
END IF

DO 30 I = 1, ILOOP

C  COMPUTE UNIT VECTORS

MRB = SQRT(RB(1)**2 + RB(2)**2 + RB(3)**2)
URB(1) = RB(1)/MRB
URB(2) = RB(2)/MRB
URB(3) = RB(3)/MRB

MRT = SQRT(RTPRED(1)**2 + RTPRED(2)**2 + RTPRED(3)**2)
URT(1) = RTPRED(1)/MRT
URT(2) = RTPRED(2)/MRT

```

$$URT(3) = RTPRED(3) / MRT$$

$$TMPV(1) = URB(2) * URT(3) - URB(3) * URT(2)$$

$$TMPV(2) = URB(3) * URT(1) - URB(1) * URT(3)$$

$$TMPV(3) = URB(1) * URT(2) - URB(2) * URT(1)$$

$$MTMPV = \sqrt{TMPV(1)^2 + TMPV(2)^2 + TMPV(3)^2}$$

$$UTMPV(1) = TMPV(1) / MTMPV$$

$$UTMPV(2) = TMPV(2) / MTMPV$$

$$UTMPV(3) = TMPV(3) / MTMPV$$

$$UTHP(1) = UTMPV(2) * URB(3) - UTMPV(3) * URB(2)$$

$$UTHP(2) = UTMPV(3) * URB(1) - UTMPV(1) * URB(3)$$

$$UTHP(3) = UTMPV(1) * URB(2) - UTMPV(2) * URB(1)$$

C ESTIMATE HORIZONTAL AND RADIAL COMPONENTS OF VC

$$VHC = VCE(1) * UTHP(1) + VCE(2) * UTHP(2) + VCE(3) * UTHP(3)$$

$$VCR = VCE(1) * URB(1) + VCE(2) * URB(2) + VCE(3) * URB(3)$$

C COMPUTE SIN AND COS OF RANGE ANGLE

$$VPHI(1) = URB(2) * URT(3) - URB(3) * URT(2)$$

$$VPHI(2) = URB(3) * URT(1) - URB(1) * URT(3)$$

$$VPHI(3) = URB(1) * URT(2) - URB(2) * URT(1)$$

$$SINPHI = \sqrt{VPHI(1)^2 + VPHI(2)^2 + VPHI(3)^2}$$

$$COSPHI = URB(1) * URT(1) + URB(2) * URT(2) + URB(3) * URT(3)$$

C COMPUTE INTERMEDIATE VARIABLES

$$MVCE = \sqrt{VCE(1)^2 + VCE(2)^2 + VCE(3)^2}$$

$$W = VHC / MRB$$

$$EL = MRB * VHC^2 / GMU$$

$$AR = MRB / MRT$$

$$TP1 = MVCE^2 * MRB / GMU$$

$$HHH = EL * SINPHI^2 * (2.0 - TP1)$$

$$SQRHHH = \sqrt{HHH}$$

C COMPUTE TIME OF FLIGHT ESTIMATE

$$T1 = EL * SINPHI / (HHH * W)$$

$$T2A = (1.0 - EL) / AR + 1.0 - AR * EL$$

$$T2B = (2.0 * EL - 1.0 - 1.0 / AR) * COSPHI$$

$$T2 = T2A + T2B$$

$$T3 = 2.0 * EL^2 * SINPHI^3 / (W * HHH * SQRHHH)$$

$$T4A = SQRHHH$$

$$T4B = EL + AR * EL + COSPHI - 1.0$$

$$T4 = \text{ATAN2}(T4A, T4B)$$

$$TFFE = T1 * T2 + T3 * T4$$

C ESTIMATE TOTAL TIME OF FLIGHT

$$TTFE = T + TFFE$$

C COMPUTE TIME OF FREE FALL AND TIME OF FLIGHT ERROR

$$TFF = TTF - T$$

$$\text{DELTF} = TFF - TFFE$$

C COMPUTE PARTIAL OF TFF W/RESPECT TO VC

$$A = 2.0 * (AR - COSPHI) / SINPHI + (VCR / VHC)$$

```

B      = A*VCR - VHC
C      = B * MRB / GMU
D      = C * EL * SINPHI**2
E      = D + HHH/VHC
PARHV  = E * 2.0

PART1V = ( 1.0/VHC - PARHV/HHH ) * T1
PART2V = ( 2.0*EL/VHC ) * ( 2.0*COSPHI - (1.0+AR**2)/AR )
PART3V = ( 1.0/VHC - PARHV/(2.0*HHH) ) * 3.0 * T3

SUBEQ1 = ( EL + AR*EL + COSPHI - 1.0 ) * VHC * PARHV
SUBEQ2 = 4.0 * HHH * EL * ( 1.0 + AR )
SUBEQ3 = ( EL + AR*EL + COSPHI-1.0 )**2 + HHH
SUBEQ4 = 2.0 * SQRHHH * VHC

PART4V = ( SUBEQ1 - SUBEQ2 ) / ( SUBEQ3 * SUBEQ4 )
PTFFV  = T1*PART2V + T2*PART1V + T3*PART4V + T4*PART3V

VCOPK  = VHC + DELTF/PTFFV

C      COMPUTE CORRELATED VELOCITY VECTOR

C      HIT EQUATION FOR RADIAL COMPONENT OF VCP

VCRPK  = VCOPK/(EL*SINPHI) * ( 1.0 - AR*EL - (1.0-EL)*COSPHI )

C      COMPUTE VC, VG, VS

DO 20 J = 1 , 3
    VC(J) = VCRPK*URB(J) + VCOPK*UTHP(J)
    VG(J) = VC(J) - VMIR(J)
    VS(J) = VG(J) - VD0(J)
20  CONTINUE

30  CONTINUE

MVS = SQRT(VS(1)**2 + VS(2)**2 + VS(3)**2)
UVS(1) = VS(1)/MVS
UVS(2) = VS(2)/MVS
UVS(3) = VS(3)/MVS

RETURN
END

```

## B.2.9 Uudnorm.for

```

C-----
C      SUBROUTINE NORM(SD,MN, ISEED,RDN)
C-----
C
C      SUBROUTINE NAME :      NORM
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           GENERATES NORMALLY DISTRIBUTED RANDOM
C                          NUMBERS USING THE BOX-MULLER TRANSFORMATION
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  RAN0
C
C      INPUTS :             SD,MN
C
C      OUTPUTS :            RDN
C
C      BOTH :              ISEED
C
C      UPDATES :           D. SMITH    - CR # 062
C                          R. RHYNE    - CR # 087
C-----
C
C      IMPLICIT DOUBLE PRECISION      (A-H)
C      IMPLICIT DOUBLE PRECISION      (O-Z)
C
C      DOUBLE PRECISION  MN
C      INTEGER*4 ISEED
C
C      COMMON / NORCOM / GSET , ISET
C
C      DATA ONE / 1.0D0 /
C      DATA TWO / 2.0D0 /
C
C      IF A SPARE RANDOM NUMBER IS NOT AVAILABLE FROM THE PREVIOUS PASS
C      GENERATE TWO NEW ONES
C
C      IF ( ISET.EQ.0 ) THEN
C
C          GET TWO UNIFORM RANDOM NUMBERS WITHIN THE SQUARE EXTENDING
C          FROM -1 TO 1 IN EACH DIRECTION
C
C      1  V1      = TWO*RAN0(ISEED) - ONE
C          V2      = TWO*RAN0(ISEED) - ONE
C
C      SEE IF THEY ARE WITHIN THE UNIT CIRCLE . IF NOT , TRY AGAIN .
C
C          R      = V1*V1 + V2*V2
C          IF ( R.GE.ONE ) GO TO 1
C
C      PERFORM BOX-MULLER TRANSFORMATION TO GENERATE TWO GAUSSIAN
C      RANDOM NUMBERS . RETURN ONE AND SAVE THE OTHER FOR THE NEXT
C      PASS .
C
C          FAC      = DSQRT ( -TWO*DLOG(R)/R )
C          GSET      = FAC*V1
C          RDN       = MN + SD*FAC*V2
C          ISET      = 1

```

```
C      USE GAUSSIAN RANDOM NUMBER CARRIED OVER FROM PREVIOUS PASS .  
      ELSE IF ( ISET.EQ.1 ) THEN  
          RDN      = MN + SD*GSET  
          ISET      = 0  
      ENDIF  
  
      RETURN  
      END
```

## B.2.10 Uuestrel.for

```

C-----
      SUBROUTINE ESTREL (TI2M,CMS,ESTATE,RREL,
      .                  VREL,MAGR,MAGV,URREL,MGRDOT,TGO,PITER,YAWER,
      .                  LAMD)
C-----
C
C      SUBROUTINE NAME :      ESTREL
C
C      AUTHOR(S) :          T. THORNTON
C
C      FUNCTION :           COMPUTES ESTIMATED RELATIVE RANGE, RANGE
C                          RATE, AND TIME-TO-GO
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             RREL,VREL,TI2M,CMS,ESTATE
C
C      OUTPUTS :            MAGR,MAGV,URREL,MGRDOT,TGO,
C                          PITER,YAWER,LAMD
C
C      UPDATES :            D. SMITH    - CR # 059
C                          R. RHYNE    - CR # 068
C                          D. SISSOM   - CR # 069
C                          B. HILL /   - CR # 081
C                          R. RHYNE    - CR # 088
C                          R. RHYNE    - CR # 093
C-----
C
      IMPLICIT REAL (A-H)
      IMPLICIT REAL (O-Z)

      REAL CMS(9)          , LAMD(2)          , LAMSKE(2)
      REAL MAGR            , MAGV            , MGRDOT
      REAL RELM(3)         , RELS(3)
      REAL RREL(3)         , TI2M(9)
      REAL URREL(3)        , VELM(3)          , VEL(3)
      REAL VMIR(3)         , VREL(3)          , VTEST(3)
      DOUBLE PRECISION RTEST(3)
      DOUBLE PRECISION RMIR(3)
      INTEGER           ESTATE

C      COMPUTE ESTIMATED RELATIVE STATES AND ESTIMATED TIME-TO-GO

      MAGR = SQRT(RREL(1)**2 + RREL(2)**2 + RREL(3)**2)
      URREL(1) = RREL(1)/MAGR
      URREL(2) = RREL(2)/MAGR
      URREL(3) = RREL(3)/MAGR

      MAGV = SQRT(VREL(1)**2 + VREL(2)**2 + VREL(3)**2)

      MGRDOT = VREL(1)*URREL(1) + VREL(2)*URREL(2) + VREL(3)*URREL(3)
      VRDRR = VREL(1)*RREL(1) + VREL(2)*RREL(2) + VREL(3)*RREL(3)
      TGO = -VRDRR/(MAGV**2)

      IF ( ESTATE.EQ.1 ) THEN

C      COMPUTE ESTIMATED RELATIVE STATES MISSILE FRAME

```

```

RELM(1) = RREL(1)*TI2M(1) + RREL(2)*TI2M(4) + RREL(3)*TI2M(7)
RELM(2) = RREL(1)*TI2M(2) + RREL(2)*TI2M(5) + RREL(3)*TI2M(8)
RELM(3) = RREL(1)*TI2M(3) + RREL(2)*TI2M(6) + RREL(3)*TI2M(9)

```

```

VELM(1) = VREL(1)*TI2M(1) + VREL(2)*TI2M(4) + VREL(3)*TI2M(7)
VELM(2) = VREL(1)*TI2M(2) + VREL(2)*TI2M(5) + VREL(3)*TI2M(8)
VELM(3) = VREL(1)*TI2M(3) + VREL(2)*TI2M(6) + VREL(3)*TI2M(9)

```

C      COMPUTE ESTIMATED RELATIVE STATES IN SEEKER FRAME

```

RELS(1) = RELM(1)*CMS(1) + RELM(2)*CMS(4) + RELM(3)*CMS(7)
RELS(2) = RELM(1)*CMS(2) + RELM(2)*CMS(5) + RELM(3)*CMS(8)
RELS(3) = RELM(1)*CMS(3) + RELM(2)*CMS(6) + RELM(3)*CMS(9)

```

```

VELS(1) = VELM(1)*CMS(1) + VELM(2)*CMS(4) + VELM(3)*CMS(7)
VELS(2) = VELM(1)*CMS(2) + VELM(2)*CMS(5) + VELM(3)*CMS(8)
VELS(3) = VELM(1)*CMS(3) + VELM(2)*CMS(6) + VELM(3)*CMS(9)

```

C      COMPUTE ESTIMATED LINE OF SIGHT ERRORS

```

LAMSKE(1) = ATAN2(-RELS(3),RELS(1))
LAMSKE(2) = ATAN2( RELS(2),RELS(1))

```

```

PITER =  LAMSKE(1)
YAWER = -LAMSKE(2)

```

C      COMPUTE ESTIMATED LINE OF SIGHT RATE ERRORS

```

LAMD(1) = (RELS(3)*VELS(1) - RELS(1)*VELS(3)) /
          (RELS(1)**2 + RELS(3)**2)
LAMD(2) = (RELS(1)*VELS(2) - RELS(2)*VELS(1)) /
          (RELS(1)**2 + RELS(2)**2)

```

```

ENDIF

```

```

RETURN
END

```

## B.2.11 Uuestr12.for

```

C-----
C      SUBROUTINE ESTREL2 (RTEST, VTEST, RMIR, VMIR, RREL, VREL)
C-----
C
C      SUBROUTINE NAME :      ESTREL2
C
C      AUTHOR(S) :          T. THORNTON
C
C      FUNCTION :           COMPUTES ESTIMATED RELATIVE RANGE, RANGE
C                          RATE, AND TIME-TO-GO
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             RTEST, VTEST, RMIR, VMIR
C
C      OUTPUTS :            RREL, VREL
C
C      UPDATES :            D. SMITH      - CR # 059
C                          R. RHYNE      - CR # 068
C                          D. SISSOM     - CR # 069
C                          B. HILL /    - CR # 081
C                          R. RHYNE
C                          R. RHYNE      - CR # 088
C                          R. RHYNE      - CR # 093
C-----
C
C      IMPLICIT DOUBLE PRECISION (A-H)
C      IMPLICIT DOUBLE PRECISION (O-Z)
C
C      DOUBLE PRECISION  RREL(3)
C      DOUBLE PRECISION  VMIR(3)          , VREL(3)          , VTEST(3)
C      DOUBLE PRECISION  RTEST(3)
C      DOUBLE PRECISION  RMIR(3)
C
C      COMPUTE ESTIMATED RELATIVE STATES AND ESTIMATED TIME-TO-GO
C
C      RREL(1) = RTEST(1) - RMIR(1)
C      RREL(2) = RTEST(2) - RMIR(2)
C      RREL(3) = RTEST(3) - RMIR(3)
C
C      VREL(1) = VTEST(1) - VMIR(1)
C      VREL(2) = VTEST(2) - VMIR(2)
C      VREL(3) = VTEST(3) - VMIR(3)
C
C      RETURN
C      END

```

## B.2.12 Uufv2bxi.for

```

C-----
C      SUBROUTINE FV2BXI ( FV, FVSQ, B )
C-----
C
C      SUBROUTINE NAME :      FV2BXI
C
C      AUTHOR(S) :           W. E. EXELY
C
C      FUNCTION :             COMPUTE DIRECTION COSINE MATRIX (B) FROM
C                             THE QUATERNION ATTITUDE VECTOR (FV) AND
C                             COMPUTE THE SQUARE (FVSQ) OF THE MAGNITUDE
C                             OF THE QUATERNION (FV)
C
C      CALLED FROM :          MISSIL
C
C      SUBROUTINES CALLED :    NONE
C
C      INPUTS :                FV
C
C      OUTPUTS :               FVSQ,B
C
C      UPDATES :               D. SMITH      - CR # 59
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C
C      DIMENSION  FV ( 4 ),    B ( 9 )
C
C      DATA      R1,  R2  /    1.0,  2.0  /
C
C      F1  = FV(1)
C      F2  = FV(2)
C      F3  = FV(3)
C      F4  = FV(4)
C      F1S = F1*F1
C      F2S = F2*F2
C      F3S = F3*F3
C      F4S = F4*F4
C      TT  = F1S + F2S + F3S + F4S
C
C      IF( TT ) 20, 20, 10
C
C      10 CONTINUE
C
C      T1  = R2/TT
C      T2  = F3*F4
C      T3  = F1*F2
C      B(2) = T1*( T3 + T2 )
C      B(4) = T1*( T3 - T2 )
C
C      T2  = F2*F4
C      T3  = F1*F3
C      B(7) = T1*( T3 + T2 )
C      B(3) = T1*( T3 - T2 )
C
C      T2  = F1*F4
C      T3  = F2*F3
C      B(6) = T1*( T3 + T2 )
C      B(8) = T1*( T3 - T2 )

```

```
C      T2      = T1*F4S - R1
      B(1) = T1*F1S + T2
      B(5) = T1*F2S + T2
      B(9) = T1*F3S + T2
C
C      20 CONTINUE
C
C      FVSQ = TT
C
C      RETURN
C      END
```

## B.2.13 Uufvdot.for

```

C-----
C      SUBROUTINE FVDOT ( W, WD, F, FD )
C-----
C
C      SUBROUTINE NAME :      FVDOT
C
C      AUTHOR(S) :           W. E. EXELY
C
C      FUNCTION :             COMPUTE THE QUATERNION DERIVATIVES (FD)
C                             USING BODY RATES (W) AND LATENT INTEGRAL
C                             DERIVATIVE (WD) AND THE QUATERNION (F)
C
C      CALLED FROM :         FORTRAN MAIN, MISSIL
C
C      SUBROUTINES CALLED :   NONE
C
C      INPUTS :              W,WD,F
C
C      OUTPUTS :             FD
C
C      UPDATES :             D. SMITH      - CR # 59
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C
C      DIMENSION W(3), F(4), FD(4)
C
C      W1 = W(1)
C      W2 = W(2)
C      W3 = W(3)
C      W4 = WD
C      F1 = F(1)
C      F2 = F(2)
C      F3 = F(3)
C      F4 = F(4)
C
C      FD(1) = ( W4*F1 + W1*F4 - W2*F3 + W3*F2 ) *0.5
C      FD(2) = ( W4*F2 + W1*F3 + W2*F4 - W3*F1 ) *0.5
C      FD(3) = ( W4*F3 - W1*F2 + W2*F1 + W3*F4 ) *0.5
C      FD(4) = ( W4*F4 - W1*F1 - W2*F2 - W3*F3 ) *0.5
C
C      RETURN
C      END

```

## B.2.14 Uugyro.for

```

C-----
C      SUBROUTINE GYRO (T,P,Q,R,CIM,GYSEED,QFRACG,PULSEG)
C-----
C
C      SUBROUTINE NAME :      GYRO
C
C      AUTHOR(S) :          A. P. BUKLEY, M. K. DOUBLEDAY
C
C      FUNCTION :           GYRO MODEL COMPUTES SENSED DELTA ANGLE
C                          COUNTS. INCLUDES AXIS MISALIGNMENT AND
C                          NONORTHOGONALITY ERRORS, SCALE FACTOR
C                          ERRORS, RANDOM AND CONSTANT DRIFT, AND
C                          QUANTIZATION.
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NORM,BRTAVG,RESP2P
C
C      INPUTS :             T,P,Q,R,CIM
C
C      OUTPUTS :            NONE
C
C      BOTH :              GYSEED,QFRACG,PULSEG
C
C      UPDATES :            T. THORNTON - CR # 004
C                          T. THORNTON - CR # 016
C                          B. HILL      - CR # 020
C                          D. SMITH     - CR # 021
C                          B. HILL      - CR # 022
C                          B. HILL      - CR # 030
C                          B. HILL      - CR # 038
C                          D. SMITH     - CR # 059
C                          D. SISSOM    - CR # 069
C                          D. SMITH     - CR # 070
C                          D. SMITH     - CR # 075
C                          D. SMITH     - CR # 077
C                          D. SMITH     - CR # 078
C                          B. HILL /    - CR # 081
C                          R. RHYNE
C                          R. RHYNE     - CR # 083
C                          R. RHYNE     - CR # 084
C                          R. RHYNE     - CR # 087
C                          B. HILL      - CR # 093
C
C-----
C
C      IMPLICIT DOUBLE PRECISION      (A-H)
C      IMPLICIT DOUBLE PRECISION      (O-Z)
C
C      DOUBLE PRECISION  CIM(9)        , CIMO(9)        , DCG(3)
C      DOUBLE PRECISION  DTHET(3)      , PULSEG(3)      , PQRAVG(3)
C      DOUBLE PRECISION  QFRACG(3)     , SF1G(3)        , SF2G(3)
C      DOUBLE PRECISION  SFEG(3)       , WBI0(3)        ,
C      DOUBLE PRECISION  WBI1(3)       , WBI2(3)        , WBO0(3)
C      DOUBLE PRECISION  WBO1(3)       , WBO2(3)        , WDRG(3)
C
C      INTEGER*4          GYSEED
C
C      LOCAL COMMON USED FOR CONSTANTS, LOCAL VARIABLES AND
C      INITIALIZATION FLAG

```

```

SAVE                IGYRO

C    COMMON "RGYRO" USED FOR MIDFLIGHT CAPABILITIES ONLY

COMMON / RGYRO / PSIG , THTG , PHIG , THXZG , THXYG ,
.              THYZG , THYXG , THZYG , THZXG , SF1G ,
.              SF2G , DCG , T0GYRO , TMO , WBI2 ,
.              WBI1 , WBO2 , WBO1 , DRSIGG

* DATA INITIALIZATION
$INCLUDE('~/include/ssdata53.dat')
$INCLUDE('~/include/ssdata16.dat')
$INCLUDE('~/include/ssdata21.dat')

DATA IGYRO / 1 /

IF (IGYRO .EQ. 1) THEN

    IGYRO = 0

C    INITIALIZE GYRO PARAMETERS

    IF (T .EQ. 0.0) THEN
        DRSIGG = DRSIGI / (60.0*DSQRT(DTIMU)*DTR)
        CALL NORM(ALNSGG,ALNMNG,GYSEED,PSIG)
        CALL NORM(ALNSGG,ALNMNG,GYSEED,THTG)
        CALL NORM(ALNSGG,ALNMNG,GYSEED,PHIG)
        CALL NORM(AORSGG,AORMNG,GYSEED,THXZG)
        CALL NORM(AORSGG,AORMNG,GYSEED,THXYG)
        CALL NORM(AORSGG,AORMNG,GYSEED,THYZG)
        CALL NORM(AORSGG,AORMNG,GYSEED,THYXG)
        CALL NORM(AORSGG,AORMNG,GYSEED,THZYG)
        CALL NORM(AORSGG,AORMNG,GYSEED,THZXG)
        CALL NORM(SF1SGG,SF1MNG,GYSEED,SF1G(1))
        CALL NORM(SF1SGG,SF1MNG,GYSEED,SF1G(2))
        CALL NORM(SF1SGG,SF1MNG,GYSEED,SF1G(3))
        CALL NORM(SF2SGG,SF2MNG,GYSEED,SF2G(1))
        CALL NORM(SF2SGG,SF2MNG,GYSEED,SF2G(2))
        CALL NORM(SF2SGG,SF2MNG,GYSEED,SF2G(3))
        CALL NORM(DCSIGG,DCMENG,GYSEED,DCG(1))
        CALL NORM(DCSIGG,DCMENG,GYSEED,DCG(2))
        CALL NORM(DCSIGG,DCMENG,GYSEED,DCG(3))
        DO 10 I = 1,3
            WBI2(I) = 0.0D0
            WBI1(I) = 0.0D0
            WBO2(I) = 0.0D0
            WBO1(I) = 0.0D0
10    CONTINUE
    ENDIF

C    COMPUTE SECOND ORDER RESPONSE DIFFERENCE EQUATION COEFFICIENTS

    IF (IGRTYP.EQ.2) THEN
        CALL RESP2R ( DTIMU,WGYR,ZGYR,CWBI2,CWBI1,CWBI0,CWBO2,CWBO1,
.                  CWBO0 )
    ENDIF

ENDIF

C    COMPUTE DELTA TIME SINCE LAST PASS THROUGH GYRO

DTDEL = T - T0GYRO
T0GYRO = T

```

```

C      DETERMINE AVERAGE BODY RATE OVER LAST INTERVAL

      IF ( DTDEL.NE.0.0D0 ) THEN
        CALL BRTAVG ( CIM , CIMO , DTDEL , PQRAVG )
      ELSE
        PQRAVG(1) = P
        PQRAVG(2) = Q
        PQRAVG(3) = R
      ENDIF

C      SAVE INERTIAL-TO-MISSILE ROTATION MATRIX FOR NEXT PASS

      DO 20 I = 1 , 9
        CIMO(I) = CIM(I)
20    CONTINUE

C      GYRO AXIS MISALIGNMENT EFFECTS

      PM      = PQRAVG(1) + PQRAVG(2)*PSIG - PQRAVG(3)*THTG
      QM      = PQRAVG(2) - PQRAVG(1)*PSIG + PQRAVG(3)*PHIG
      RM      = PQRAVG(3) + PQRAVG(1)*THTG - PQRAVG(2)*PHIG

C      GYRO AXIS NONORTHOGONALITY EFFECTS

      PN      = PM + QM*THXZG - RM*THXYG
      QN      = QM - PM*THYZG + RM*THYXG
      RN      = RM + PM*THZYG - QM*THZXG

C      ADD LINEAR AND QUADRATIC SCALE FACTOR ERRORS

      SFEG(1) = PN + SF1G(1)*PN + SF2G(1)*PN**2
      SFEG(2) = QN + SF1G(2)*QN + SF2G(2)*QN**2
      SFEG(3) = RN + SF1G(3)*RN + SF2G(3)*RN**2

C      FOR EACH AXIS ...

      DO 30 I = 1,3

C      MAKE A GAUSSIAN DRAW FOR RANDOM DRIFT AND ADD TO CONSTANT
C      DRIFT

      IF ( DRSIGG.GT.0.0D0 ) THEN
        CALL NORM(DRSIGG,DRMENG,GYSEED,DRG)
      ENDIF

      WDRG(I) = DRG + DCG(I)

C      COMPUTE INPUT TO GYRO RESPONSE MODEL

      WBI0(I) = SFEG(I) + WDRG(I)

C      SECOND ORDER RESPONSE MODEL

      IF ( IGRTP.EQ.2 ) THEN
        WBO0(I) = ( CWBI0*WBI0(I) + CWBI1*WBI1(I)
                  + CWBI2*WBI2(I) - CWBO1*WBO1(I)
                  - CWBO2*WBO2(I) ) /CWBO0
        WBI2(I) = WBI1(I)
        WBI1(I) = WBI0(I)
        WBO2(I) = WBO1(I)
        WBO1(I) = WBO0(I)
      ENDIF

```

```

C      INSTANTANEOUS RESPONSE MODEL

      IF ( IGRTYP.EQ.0 ) THEN
        WBO0(I) = WBIO(I)
      ENDIF

C      COMPUTE DELTA THETA

      DTHET(I) = DTDEL * WBO0(I)

      IF ( SPPG.GT.0.0 ) THEN

C      UNQUANTIZED OUTPUT IN COUNTS

        QFRACG(I) = QFRACG(I) - PULSEG(I) + DTHET(I)/SPPG

C      QUANTIZED OUTPUT IN COUNTS

        PULSEG(I) = DINT(QFRACG(I))
      ELSE
        PULSEG(I) = DTHET(I)
      ENDIF

30 CONTINUE

      RETURN
      END

```

## B.2.15 Uuimupro.for

```

C-----
      SUBROUTINE IMUPRO(T,PULSEG,PULSEA,DELPHI,DELTHT,DELPSE,DELU,
      .                DELV,DELW)
C-----
C
C      SUBROUTINE NAME :      IMUPRO
C
C      AUTHOR(S) :          T. THORNTON
C
C      FUNCTION :           COMPUTES THE IMU PROCESSOR RELATED FUNCTIONS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             T,PULSEG,PULSEA
C
C      OUTPUTS :            DELPHI,DELTHT,DELPSE,DELU,DELV,DELW
C
C      UPDATES :            T. THORNTON - CR # 004
C                          T. THORNTON - CR # 016
C                          B. HILL     - CR # 022
C                          T. THORNTON - CR # 037
C                          D. SMITH    - CR # 059
C                          D. SMITH    - CR # 070
C                          D. SMITH    - CR # 075
C                          B. HILL /   - CR # 081
C                          R. RHYNE    - CR # 081
C                          B. HILL     - CR # 093
C-----
      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      DOUBLE PRECISION  PULSEA(3)    , PULSEG(3)

      * DATA INITIALIZATION
      $INCLUDE('~/INCLUDE/SSDATA54.DAT')

C      GYRO OUTPUT COMPENSATION

C      CALCULATE DELTA ANGLES

      IF ( PERPG.GT.0.0 ) THEN
        DELPHS = PULSEG(1)*PERPG
        DELTHS = PULSEG(2)*PERPG
        DELPSS = PULSEG(3)*PERPG
      ELSE
        DELPHS = PULSEG(1)
        DELTHS = PULSEG(2)
        DELPSS = PULSEG(3)
      END IF

C      COMPENSATE SENSED DELTA ANGLES FOR SCALE FACTOR ERRORS

      DELPH = DELPHS*SFCGX
      DELTH = DELTHS*SFCGY
      DELPS = DELPSS*SFCGZ

C      COMPENSATE SENSED DELTA ANGLES FOR GYRO MISALIGNMENTS

```

```

DELPHI = DELPH      - DELTH*PSIGP + DELPS*THTGP
DELTHT = DELPH*PSIGP + DELTH      - DELPS*PHIGP
DELPST = -DELPH*THTGP + DELTH*PHIGP + DELPS

```

C ACCELEROMETER OUTPUT COMPENSATION

C CALCULATE DELTA VELOCITY

```

IF ( PERPA.GT.0.0 ) THEN
  DELUS = PULSEA(1)*PERPA
  DELVS = PULSEA(2)*PERPA
  DELWS = PULSEA(3)*PERPA
ELSE
  DELUS = PULSEA(1)
  DELVS = PULSEA(2)
  DELWS = PULSEA(3)
END IF

```

C COMPENSATE SENSED VELOCITY FOR SCALE FACTOR ERRORS

```

DELXS = DELUS*SFCAX
DELYS = DELVS*SFCAY
DELZS = DELWS*SFCAZ

```

C COMPENSATE SENSED VELOCITY FOR ACCELEROMETER MISALIGNMENTS

```

DELUM = DELXS      - DELYS*PSIAP + DELZS*THTAP
DELVM = DELXS*PSIAP + DELYS      - DELZS*PHIAP
DELWM = -DELXS*THTAP + DELYS*PHIAP + DELZS

```

C SKULLING COMPENSATION

```

IF ( ISKULL.EQ.0 ) THEN
  DELU = DELUM
  DELV = DELVM
  DELW = DELWM
ELSE
  DELU = DELUM - 0.5 * ( DELPSI*DELVM - DELTHT*DELWM )
  DELV = DELVM - 0.5 * ( DELPHI*DELWM - DELPSI*DELU )
  DELW = DELWM - 0.5 * ( DELTHT*DELU - DELPHI*DELVM )
END IF

```

```

RETURN
END

```

## B.2.16 Uuinteg.for

```

C-----
C      SUBROUTINE INTEG ( X , XDOT , T , I )
C-----
C
C      SUBROUTINE NAME :      INTEG
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           Perform simple trapezoidal integration of
C                          XDOT to yield X. DTD is the time since
C                          the last integration and I is the array
C                          index where X is stored
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             XDOT,T,I
C
C      OUTPUTS :            X
C
C      UPDATES :            D. SISSOM   - CR # 58
C                          D. SMITH    - CR # 59
C-----
C
C      COMMON/STORAG/      XINT,          TINT,          XDOTL
C      DOUBLE PRECISION    XINT(50),      TINT(50),      XDOTL(50)
C      DOUBLE PRECISION    DT,            DTMP,          X
C      DOUBLE PRECISION    XDOT,          T
C
C      DT      = T - TINT(I)
C
C      XINT(I) = XINT(I) + 0.5D0*DT*(XDOT+XDOTL(I))
C      X      = XINT(I)
C
C      TINT(I) = T
C      XDOTL(I) = XDOT
C
C      TEMPORARY CODE TO NORMALIZE QUATERNION AFTER 4TH COMPONENT IS
C      REVISED
C
C      IF ( I.EQ.18 ) THEN
C          DTMP = DSQRT ( XINT(15)**2 + XINT(16)**2 + XINT(17)**2 +
C                          XINT(18)**2 )
C          XINT(15) = XINT(15) / DTMP
C          XINT(16) = XINT(16) / DTMP
C          XINT(17) = XINT(17) / DTMP
C          XINT(18) = XINT(18) / DTMP
C      END IF
C
C      RETURN
C      END

```

## B.2.17 Uuintegi.for

```

C-----
C      SUBROUTINE INTEGI ( X , XDOT , T , I )
C-----
C
C      SUBROUTINE NAME :      INTEGI
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           Initialize integral of X which is stored
C                           in position I of the integral array
C
C      CALLED FROM :        MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             X,XDOT,T,I
C
C      OUTPUTS :            NONE
C
C      UPDATES :            D. SISSOM   - CR # 58
C                           D. SMITH    - CR # 59
C-----
C
C      COMMON/STORAG/      XINT,          TINT,          XDOTL
C      DOUBLE PRECISION    XINT(50),      TINT(50),      XDOTL(50)
C      DOUBLE PRECISION    X,              T,              XDOT
C
C      XINT(I) = X
C      XDOTL(I) = XDOT
C      TINT(I) = T
C
C      RETURN
C      END

```

## B.2.18 Uukalman.for

```

C-----
C      SUBROUTINE KALMAN (T, TI2M, LAMMO, ASIG, SNRO, TGO, RRELO, VRELO, TI2MO,
C      .                  RACQ, MAGRTR, MAGR, MAGV, LAMSEK, LAMDXX, FRMRAT, CMS,
C      .                  MACQ, MCSO, MTERM, IRESLV, TRACK, TERM, TRMTGO, TGE1,
C      .                  TGE2AL, WFILT, ZFILT, LAM, LAMD, IBURN1, ACQD, ESTATE,
C      .                  PITER, YAWER, ROLLER)
C-----
C
C      SUBROUTINE NAME :      KALMAN
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           2-STATE EXTENDED KALMAN FILTER
C                          ESTIMATES LOS ANGLES AND RATES
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             T, TI2M, LAMMO, ASIG, SNRO, TGO, RRELO, VRELO,
C                          TI2MO, RACQ, MAGRTR, MAGR, MAGV, LAMSEK, LAMDXX,
C                          FRMRAT, CMS, MACQ, MCSO, MTERM, IRESLV
C
C      OUTPUTS :            TRMTGO, TGE1, TGE2AL, WFILT, ZFILT, LAM,
C                          LAMD, IBURN1, ACQD, PITER, YAWER
C
C      BOTH :               ESTATE, TRACK, TERM
C
C      UPDATES :            D. SISSOM   - CR # 032
C                          B. HILL     - CR # 030
C                          B. HILL     - CR # 038
C                          T. THORNTON - CR # 043
C                          T. THORNTON - CR # 048
C                          D. SMITH    - CR # 059
C                          D. SMITH    - CR # 064
C                          R. RHYNE    - CR # 068
C                          D. SISSOM   - CR # 069
C                          D. SMITH    - CR # 070
C                          D. SMITH    - CR # 074
C                          R. RHYNE    - CR # 079
C                          B. HILL /   - CR # 081
C                          R. RHYNE
C                          B. HILL     - CR # 086
C                          R. RHYNE    - CR # 087
C                          R. RHYNE    - CR # 088
C                          D. SISSOM   - CR # 091
C                          B. HILL     - CR # 093
C-----

```

```

IMPLICIT REAL      (A-H)
IMPLICIT REAL      (O-Z)

```

```

CHARACTER*128 MESSAGE

```

```

REAL  CSSHFT(3)      , TMSHFT(3)      , TKSHT(3)
REAL  LAMSEK(2)      , LAMDXX(2)      , MAGRSQ
REAL  LAM(2)         , LAMD(2)        , MAGRO
REAL  RRELO(3)       , VRELO(3)       , RATE(6)
REAL  LAMMO(2)       , TI2MO(9)      , TI2M(9)
REAL  MAGR           , MAGV          , MAGRTR
REAL  CMS(9)

```

# Appendix B - Exosim v2.0 Midcourse and Terminal Phases

```

      INTEGER          SEKTYP          , ACQD
      INTEGER          ESTATE          , TRACK          , TERM

C      LOCAL COMMON USED FOR CONSTANTS, LOCAL VARIABLES AND
C      INITIALIZATION FLAG

      SAVE              IKALMN

C      COMMON "RKALMN" USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RKALMN / TKF          , IDRTOK , PP11      , PP12      , PP22      ,
      .                      PY11      , PY12      , PY22      , PLMDFP , YLMDFP ,
      .                      PLAMH      , YLAMH      , PLAMDH , YLAMDH , PLAMDF ,
      .                      YLAMDF , TGIL      , KFMODE , IFPAS

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSDATA48.DAT')
$INCLUDE('~/INCLUDE/SSDATA50.DAT')
$INCLUDE('~/INCLUDE/SSDATA55.DAT')
$INCLUDE('~/INCLUDE/SSDATA56.DAT')
$INCLUDE('~/INCLUDE/SSDATA57.DAT')
$INCLUDE('~/INCLUDE/SSDATA11.DAT')
$INCLUDE('~/INCLUDE/SSDATA12.DAT')

      DATA IKALMN / 1 /

      IF (IKALMN .EQ. 1) THEN

        IKALMN = 0

        IF (IFPAS .EQ. 0) THEN

C          INITIALIZE FILTER PARAMETERS

          KFMODE = 1
          TKF    = T

C          INITIALIZE FILTER ESTIMATES OF INERTIAL FRAME LAMBDA AND
C          LAMBDA DOT

          PLMDH1 = (RRELO(3)*VRELO(1) - RRELO(1)*VRELO(3))/
          .      (RRELO(1)**2 + RRELO(3)**2)
          PLAMDH = PLMDH1
          YLMDH1 = (RRELO(1)*VRELO(2) - RRELO(2)*VRELO(1))/
          .      (RRELO(1)**2 + RRELO(2)**2)
          YLAMDH = YLMDH1

C          INITIALIZE COVARIANCE MATRIX ELEMENTS

          PP22 = SGP22**2
          PY22 = SGP22**2
          PP12 = SGP12**2
          PY12 = SGP12**2
          PP11 = SGP11**2
          PY11 = SGP11**2

C          INITIALIZE PROCESS NOISE COVARIANCE

          RW = SGW**2

C          INITIALIZE MEASUREMENT NOISE MATRIX

          RV = AKSGME*ASIG**2

```

```

        ENDIF

ENDIF

C    INCREMENT FILTER PASS COUNTER

    IFPAS = IFPAS + 1

C    PERFORM EXECUTIVE FUNCTION FOR SEEKER TYPES 0 AND 1

    IF ( SEKTYP.EQ.0 .OR. SEKTYP.EQ.1 ) THEN

C        INITIATE ACQUISTION MODE

        IF ( ACQD.EQ.0 .AND. MAGRTR.LE.RACQ ) THEN
            ESTATE = 0
            ACQD = 1
            TRMTGO = TGO - (MAGR - RNGTRM)/MAGV
            TGIL = TRMTGO + TWAIT
            TGE2AL = TGIL + DTVCS2
            CALL OUTMES(' INITIATE ACQUISITION PHASE')
        ENDIF

C        COMPUTE THE SEEKER DATA RATE

        IF ( TRACK .EQ. 1 ) THEN
            TRACK = 2
            TGE1 = TGO - (RNHITS + ILAG)/FRMRAT
            IBURN1 = 0
            CALL OUTMES(' INITIATE TRACK PHASE')
        ELSEIF ( TERM .EQ. 1 ) THEN
            TERM = 2
            CALL OUTMES(' INITIATE TERMINAL PHASE')
        ENDIF

    ENDIF

C    USE TRUE LOS ANGLES AND RATES WITH PERFECT SEEKER MODEL

    IF ( SEKTYP.EQ.0 .AND. ESTATE.EQ.0) THEN
        LAMD(1) = LAMDXX(1)
        LAMD(2) = LAMDXX(2)
        PITER = LAMMO(1)
        YAWER = LAMMO(2)
        ROLLER = 0.0
        RETURN
    ENDIF

C    DETERMINE APPARENT RELATIVE INERTIAL FRAME STATES FOR LOCAL USE

    RXI = RRELO(1)
    RYI = RRELO(2)
    RZI = RRELO(3)

    VXI = VRELO(1)
    VYI = VRELO(2)
    VZI = VRELO(3)

    MAGRO = SQRT ( RXI**2 + RYI**2 + RZI**2 )

C    RECONSTRUCT MEASURED LOS VECTOR IN SEEKER FRAME

    TANPCH = TAN ( LAMMO(1) )

```

```

TANYAW = TAN ( LAMMO(2) )

XLOSS = 1.0E0 / SQRT ( 1.0D0 + TANPCH**2 + TANYAW**2 )
YLOSS = XLOSS * TANYAW
ZLOSS = - XLOSS * TANPCH

C ROTATE MEASURED LOS VECTOR INTO MISSILE FRAME

XLOSM = CMS(1)*XLOSS + CMS(2)*YLOSS + CMS(3)*ZLOSS
YLOSM = CMS(4)*XLOSS + CMS(5)*YLOSS + CMS(6)*ZLOSS
ZLOSM = CMS(7)*XLOSS + CMS(8)*YLOSS + CMS(9)*ZLOSS

C ROTATE MEASURED LOS VECTOR INTO INERTIAL FRAME

XLOSI = TI2MO(1)*XLOSM + TI2MO(2)*YLOSM + TI2MO(3)*ZLOSM
YLOSI = TI2MO(4)*XLOSM + TI2MO(5)*YLOSM + TI2MO(6)*ZLOSM
ZLOSI = TI2MO(7)*XLOSM + TI2MO(8)*YLOSM + TI2MO(9)*ZLOSM

C DETERMINE MEASURED LOS ANGLES IN INERTIAL FRAME

PLAMM = ATAN2 ( -ZLOSI , XLOSI )
YLAMM = ATAN2 ( YLOSI , XLOSI )

C EXECUTE FILTER INITIALIZATION LOGIC ON FIRST FILTER PASS

C THE FOLLOWING INITIALIZATION IS DONE HERE, RATHER THAN IN THE
C INITIAL SECTION TO AVOID REPETITIVE CALCULATIONS TO OBTAIN THE
C VALUES OF PLAMM AND YLAMM

IF ( IFPAS.EQ.1 ) THEN

    PLAMH1 = PLAMM
    PLAMH = PLAMH1
    YLAMH1 = YLAMM
    YLAMH = YLAMH1

ENDIF

C DETERMINE TIME SINCE LAST FILTER UPDATE

IF ( T.GT.TKF ) THEN
    DTKF = T - TKF
ELSE
    DTKF = 0.0E0
ENDIF
TKF = T

C ENABLE FIRST BURN WHEN DATA RATE IS SUFFICIENT (SEEKER TYPE 2)
C OR WHEN IN TERMINAL MODE (SEEKER TYPE 3)

IF ( (SEKTYP.EQ.2.AND.FRMRAT.GE.RATE(5).AND.IDRTOK.EQ.0) .OR.
    (SEKTYP.EQ.3.AND.IDRTOK.EQ.0.AND.MTERM.EQ.1) ) THEN
    TGE1 = TGO - RNHITS/FRMRAT
    IBURN1 = 0
    IDRTOK = 1
ENDIF

C ENABLE ACQUISITION MODE ON FIRST PASS

IF ( (SEKTYP.NE.3.AND.KFMODE.EQ.1.AND.SNRO.GE.SNRACQ) .OR.
    (SEKTYP.EQ.3.AND.KFMODE.EQ.1.AND.MACQ.EQ.1) ) THEN
    WRITE(MESSAGE,101) T,MAGRO
    CALL OUTMES(MESSAGE)

```

```

101  FORMAT(1X,E16.9,' ACQUISITION MODE ENABLED:  MAGRO = ',E16.9)
      KFMODE = 2
      ACQD   = 1
      ELSEIF ((SEKTYP.NE.3 .AND. KFMODE.EQ.2 .AND. SNRO.GE.SNRTRK) .OR.
      . (SEKTYP.EQ.3 .AND. KFMODE.EQ.2 .AND. MACQ.EQ.1) ) THEN

C      REINITIALIZE ERROR COVARIANCE DIAGONAL ELEMENTS SWITCH FROM
C      ACQUISITION TO TRACK MODE

      WRITE(MESSAGE,102) T,MAGRO
      CALL OUTMES(MESSAGE)
102  FORMAT(1X,E16.9,' TRACK MODE ENABLED:  MAGRO = ',E16.9)
      KFMODE = 3
      MAGRSQ = MAGRO**2
      TGOSQ  = TGO**2
      PP11   = PP11 + TKSHT(3)**2/MAGRSQ
      PY11   = PY11 + TKSHT(2)**2/MAGRSQ
      PP22   = PP22 + TKSHT(3)**2/(MAGRSQ*TGOSQ)
      PY22   = PY22 + TKSHT(2)**2/(MAGRSQ*TGOSQ)
      ENDIF

      IF ( KFMODE.GE.3 .AND. IFPAS.GE.NINT(RNHITS) ) ESTATE = 0

C      REINITIALIZE ERROR COVARIANCE DIAGONAL ELEMENTS AT SWITCH FROM
C      TRACK TO DISCRIMINATION MODE

      IF ( (SEKTYP.NE.3 .AND. KFMODE.EQ.3 .AND. SNRO.GE.SNRCSO) .OR.
      . (SEKTYP.EQ.3 .AND. KFMODE.EQ.3 .AND. MCSO.EQ.1) ) THEN
      WRITE(MESSAGE,103) T,MAGRO
      CALL OUTMES(MESSAGE)
103  FORMAT(1X,E16.9,' CSO MODE ENABLED:  MAGRO = ',E16.9)
      KFMODE = 4
      MAGRSQ = MAGRO**2
      TGOSQ  = TGO**2
      PP11   = PP11 + CSSHT(3)**2/MAGRSQ
      PY11   = PY11 + CSSHT(2)**2/MAGRSQ
      PP22   = PP22 + CSSHT(3)**2/(MAGRSQ*TGOSQ)
      PY22   = PY22 + CSSHT(2)**2/(MAGRSQ*TGOSQ)
      ENDIF

C      REINITIALIZE ERROR COVARIANCE DIAGONAL ELEMENTS AT SWITCH FROM
C      DISCRIMINATION TO TERMINAL MODE (SEEKER TYPE 2) OR FRAME RATE
C      EQUALS 12.5 (SEEKER TYPE 3) AND ENABLE SECOND BURN

      IF ( (SEKTYP.NE.3 .AND. KFMODE.EQ.4 .AND. SNRO.GE.SNRTRM) .OR.
      . (SEKTYP.EQ.3 .AND. KFMODE.EQ.4 .AND. FRMRAT.GE.RATE(3)) ) THEN
      WRITE(MESSAGE,104) T,MAGRO
      CALL OUTMES(MESSAGE)
104  FORMAT(1X,E16.9,' TERMINAL MODE ENABLED:  MAGRO = ',E16.9)
      KFMODE = 5
      TGE2AL = TGO - RNHITS/FRMRAT
      TRMTGO = TGO - RNHITS/FRMRAT
      MAGRSQ = MAGRO**2
      TGOSQ  = TGO**2
      PP11   = PP11 + TMSHT(3)**2/MAGRSQ
      PY11   = PY11 + TMSHT(2)**2/MAGRSQ
      PP22   = PP22 + TMSHT(3)**2/(MAGRSQ*TGOSQ)
      PY22   = PY22 + TMSHT(2)**2/(MAGRSQ*TGOSQ)
      ENDIF

C      COMPUTE R ( MEASUREMENT NOISE MATRIX ) FOR CURRENT TIME

      RV      = AKSGME * ASIG**2

```

```

C   PROCESS NOISE TERMS AS A FUNCTION OF HOMING PHASE

      IF ( KFMODE.GT.2 .AND. KFMODE.LT.5 ) THEN
            RW      = SGWH**2
      ELSE IF ( KFMODE.EQ.5 ) THEN
            RW      = SGWT**2
      ENDIF

C   COMPUTE Q ( PROCESS NOISE MATRIX ) FOR CURRENT TIME

      Q11      = RW * DTKF**2 / 4.0E0
      Q12      = RW * DTKF / 2.0E0
      Q22      = RW

C   EXTRAPOLATE COVARIANCE MATRIX TO CURRENT TIME
C   P(N+1) = PHI(N)*P(N)*PHI(N)T + Q

      PPX      = PP12 + DTKF*PP22
      PYX      = PY12 + DTKF*PY22
      PP11      = Q11 + PP11 + DTKF*(PP12+PPX)
      PY11      = Q11 + PY11 + DTKF*(PY12+PYX)
      PP12      = Q12 + PPX
      PY12      = Q12 + PYX
      PP22      = Q22 + PP22
      PY22      = Q22 + PY22

C   COMPUTE KALMAN FILTER GAIN MATRIX :
C
C   K(N)      = P(N) * HT* ( H*P(N) * HT + RV )**-1

      DNP      = PP11 + RV
      DNY      = PY11 + RV
      AKP11     = PP11 / DNP
      AKY11     = PY11 / DNY
      AKP21     = PP12 / DNP
      AKY21     = PY12 / DNY

      IF ( AKP11.GT.GFLIM ) AKP11 = GFLIM
      IF ( AKY11.GT.GFLIM ) AKY11 = GFLIM
      IF ( AKP21.GT.GFDLIM ) AKP21 = GFDLIM
      IF ( AKY21.GT.GFDLIM ) AKY21 = GFDLIM

C   COMPUTE FILTER BANDWIDTH AND DAMPING

      IF ( AKP21.GT.0.0E0 .AND. DTKF.GT.0.0E0 ) THEN
            WFILT = SQRT ( AKP21 / DTKF )
            ZFILT = AKP11 * WFILT / ( 2.0E0 * AKP21 )
      ENDIF

C   UPDATE COVARIANCE MATRIX :
C   +
C   P(N)      = ( I - K(N)*H ) * P(N)

      PP22      = PP22 - AKP21*PP12
      PY22      = PY22 - AKY21*PY12
      PP12      = PP12 - AKP21*PP11
      PY12      = PY12 - AKY21*PY11
      PP11      = PP11 - AKP11*PP11
      PY11      = PY11 - AKY11*PY11

C   ESTIMATE DELTA LOS ANGULAR RATE DUE TO MISSILE MOTION ( 'PLANT'
C   INPUT OR FORCING FUNCTION )

      PLMDF     = ( RZI*VXI - RXI*VZI ) / ( RXI**2 + RZI**2 )

```

```

YLMDF = ( RXI*VYI - RYI*VXI ) / ( RXI**2 + RYI**2 )

IF ( DTKF.NE.0.0E0 ) THEN
    DLPLMD = ( PLMDF - PLMDFP )
    DLYLMD = ( YLMDF - YLMDFP )
ELSE
    DLPLMD = 0.0E0
    DLYLMD = 0.0E0
ENDIF

PLMDFP = PLMDF
YLMDFP = YLMDF

C    EXTRAPOLATE FILTERED INERTIAL FRAME STATES TO CURRENT TIME

PLAMH1 = PLAMH + DTKF * ( PLAMDH + 0.5E0*DTKF*DLPLMD )
YLAMH1 = YLAMH + DTKF * ( YLAMDH + 0.5E0*DTKF*DLYLMD )

PLMDH1 = PLAMDH + DLPLMD
YLM DH1 = YLAMDH + DLYLMD

C    REVISE FILTER ESTIMATES OF INERTIAL FRAME LAMBDA AND LAMBDA DOT :
C    ^      + ^
C    X(N) = X(N)^ + K(N)*( Y(N) - H*X(N)^ )

ERRP = PLAMM - PLAMH1
ERRY = YLAMM - YLAMH1
PLAMH = PLAMH1 + AKP11*ERRP
PLAMDH = PLMDH1 + AKP21*ERRP
YLAMH = YLAMH1 + AKY11*ERRY
YLAMDH = YLMDH1 + AKY21*ERRY

C    EXTRAPOLATE LOS ANGLES AHEAD TO ACCOUNT FOR SIGNAL PROCESSING LAG

IF ( DTKF.NE.0.0E0 ) THEN
    DLPLMD = DLPLMD * SPLAG / DTKF
    DLYLMD = DLYLMD * SPLAG / DTKF
ELSE
    DLPLMD = 0.0E0
    DLYLMD = 0.0E0
ENDIF

PLAMF = PLAMH + SPLAG * ( PLAMDH + 0.5E0*SPLAG*DLPLMD )
YLAMF = YLAMH + SPLAG * ( YLAMDH + 0.5E0*SPLAG*DLYLMD )

PLAMDF = PLAMDH + DLPLMD
YLAMDF = YLAMDH + DLYLMD

C    RECONSTRUCT FILTERED LOS VECTOR IN INERTIAL FRAME

TANPCH = TAN ( PLAMF )
TANYAW = TAN ( YLAMF )
COSPSQ = COS ( PLAMF ) **2
COSYSQ = COS ( YLAMF ) **2

XLOSI = 1.0E0 / SQRT ( 1.0E0 + TANPCH**2 + TANYAW**2 )
YLOSI = XLOSI * TANYAW
ZLOSI = - XLOSI * TANPCH

C    DETERMINE FILTERED LOS VECTOR RATES IN INERTIAL FRAME

XLOSDI = - ( PLAMDF*TANPCH/COSPSQ
            + YLAMDF*TANYAW/COSYSQ ) * XLOSI**3
YLOSDI = YLAMDF*XLOSI /COSYSQ + XLOSDI*TANYAW

```

```

ZLOSDI = - PLAMDF*XLOSI /COSPSQ - XLOSDI*TANPCH

C ROTATE LOS VECTOR INTO MISSILE FRAME

XLOSM = TI2M(1)*XLOSI + TI2M(4)*YLOSI + TI2M(7)*ZLOSI
YLOSM = TI2M(2)*XLOSI + TI2M(5)*YLOSI + TI2M(8)*ZLOSI
ZLOSM = TI2M(3)*XLOSI + TI2M(6)*YLOSI + TI2M(9)*ZLOSI

C ROTATE LOS VECTOR RATES INTO MISSILE FRAME

XLOSDM = TI2M(1)*XLOSDI + TI2M(4)*YLOSDI + TI2M(7)*ZLOSDI
YLOSDM = TI2M(2)*XLOSDI + TI2M(5)*YLOSDI + TI2M(8)*ZLOSDI
ZLOSDM = TI2M(3)*XLOSDI + TI2M(6)*YLOSDI + TI2M(9)*ZLOSDI

C ROTATE LOS VECTOR INTO SEEKER FRAME

XLOSS = CMS(1)*XLOSM + CMS(4)*YLOSM + CMS(7)*ZLOSM
YLOSS = CMS(2)*XLOSM + CMS(5)*YLOSM + CMS(8)*ZLOSM
ZLOSS = CMS(3)*XLOSM + CMS(6)*YLOSM + CMS(9)*ZLOSM

C ROTATE LOS VECTOR RATES INTO SEEKER FRAME

XLOS DS = CMS(1)*XLOSDM + CMS(4)*YLOSDM + CMS(7)*ZLOSDM
YLOS DS = CMS(2)*XLOSDM + CMS(5)*YLOSDM + CMS(8)*ZLOSDM
ZLOS DS = CMS(3)*XLOSDM + CMS(6)*YLOSDM + CMS(9)*ZLOSDM

C DETERMINE LOS ANGLES IN SEEKER FRAME

LAM(1) = ATAN2 ( -ZLOSS , XLOSS )
LAM(2) = ATAN2 ( YLOSS , XLOSS )

C DETERMINE LOS ANGULAR RATES IN SEEKER FRAME

TANPCH = TAN ( LAM(1) )
TANYAW = TAN ( LAM(2) )
COSPSQ = COS ( LAM(1) ) **2
COSYSQ = COS ( LAM(2) ) **2

LAM D(1) = ( - ZLOS DS - XLOS DS*TANPCH ) * COSPSQ / XLOSS
LAM D(2) = ( YLOS DS - XLOS DS*TANYAW ) * CCSYSQ / XLOSS

C DETERMINE ATTITUDE ERRORS

IF ( ESTATE .EQ. 0 ) THEN
  PITER = LAM(1)
  YAWER = -LAM(2)
  ROLLER = 0.0
ENDIF

RETURN
END

```

## B.2.19 Uum3x3i.for

```

C-----
C
C      SUBROUTINE M3X3I ( A , B )
C-----
C
C      SUBROUTINE NAME :      M3X3I
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           Compute the inverse of a 3 by 3 matrix .
C
C      CALLED FROM :        UTILITY ROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             A
C
C      OUTPUTS :            B
C
C      UPDATES :            NONE
C-----
C-----

      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      DOUBLE PRECISION  A(3,3),      B(3,3)

      DET = A(1,1)*A(2,2)*A(3,3) - A(1,1)*A(2,3)*A(3,2)
      .    + A(1,2)*A(2,3)*A(3,1) - A(1,2)*A(2,1)*A(3,3)
      .    + A(1,3)*A(2,1)*A(3,2) - A(1,3)*A(2,2)*A(3,1)

      IF ( DET.NE.0.0D0 ) THEN
        B(1,1) = ( A(2,2)*A(3,3) - A(2,3)*A(3,2) ) / DET
        B(2,1) = ( A(2,3)*A(3,1) - A(2,1)*A(3,3) ) / DET
        B(3,1) = ( A(2,1)*A(3,2) - A(2,2)*A(3,1) ) / DET
        B(1,2) = ( A(1,3)*A(3,2) - A(1,2)*A(3,3) ) / DET
        B(2,2) = ( A(1,1)*A(3,3) - A(1,3)*A(3,1) ) / DET
        B(3,2) = ( A(1,2)*A(3,1) - A(1,1)*A(3,2) ) / DET
        B(1,3) = ( A(1,2)*A(2,3) - A(1,3)*A(2,2) ) / DET
        B(2,3) = ( A(1,3)*A(2,1) - A(1,1)*A(2,3) ) / DET
        B(3,3) = ( A(1,1)*A(2,2) - A(1,2)*A(2,1) ) / DET
      ELSE
        B(1,1) = 0.0D0
        B(2,1) = 0.0D0
        B(3,1) = 0.0D0
        B(1,2) = 0.0D0
        B(2,2) = 0.0D0
        B(3,2) = 0.0D0
        B(1,3) = 0.0D0
        B(2,3) = 0.0D0
        B(3,3) = 0.0D0
      END IF

      RETURN
      END

```

## B.2.20 Uumasspr.for

```

C-----
      SUBROUTINE MASSPR(T,MDOTA,MDOTV,MASS,EISP,IMASS,
      .                MDOT,WEIGHT,WDOTTP,WDOTKV,WDOTTI,IXX,
      .                IYY,IZZ)
C-----
C
C      SUBROUTINE NAME :      MASSPR
C
C      AUTHOR(S) :          B. HILL
C
C      FUNCTION :           CALCULATE MISSILE MASS PROPERTIES
C
C      CALLED FROM :        MAIN
C
C      SUBROUTINES CALLED :  TABLE
C
C      INPUTS :              T,MDOTT,MDOTF,MDOTA,MDOTV,MASS,EISP
C
C      OUTPUTS :             MDOT,WEIGHT,WDOTTP,WDOTFR,WDOTKV,WDOTTI,CG,
C                           IXX,IYY,IZZ
C
C      BOTH :                TBRK,IMASS
C
C      UPDATES :             D. SMITH   - CR # 059
C                           D. SISSOM  - CR # 069
C                           D. SMITH   - CR # 076
C                           D. SMITH   - CR # 080
C                           B. HILL /  - CR # 081
C                           R. RHYNE
C                           R. RHYNE   - CR # 087
C                           B. HILL   - CR # 089
C                           B. HILL   - CR # 093
C-----
      IMPLICIT REAL          (A-H)
      IMPLICIT REAL          (O-Z)

      REAL INERXX(20)        , INERY(20)
      REAL INERZZ(20)        , IXX          , IYY
      REAL IZZ              , MASS          , MASSL
      REAL MASST1(20)        , MASST2(20)   , MDOT
      REAL MDOTA            , MDOTV

C      LOCAL COMMON USED TO HOLD CONSTANTS AND INITIALIZATION FLAG

      SAVE                  IDATIN , BISP

C      COMMON "RMASS" USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RMASS / TLSTM , MASSL

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSDATA45.DAT')
$INCLUDE('~/INCLUDE/SSDATA58.DAT')
$INCLUDE('~/INCLUDE/SSDATA22.DAT')
$INCLUDE('~/INCLUDE/SSDATA23.DAT')

      DATA IDATIN / 1 /
      DATA ICG / 1 /, III / 1 /

```

```

IF (IMASS .EQ. 1) THEN
    IMASS = 0
    IF (IDATIN .EQ. 1) THEN
        IDATIN = 0
C        ZERC BOOSTER SPECIFIC IMPULSE AFTER SECOND STAGE
        BISP = 0.0
        EISP = 0.0
    ENDIF
ENDIF

C    CALCULATE TOTAL MASS FLOW RATE
MDOT = - MDOTA - MDOTV

C    CONVERT MASS TO WEIGHT
WEIGHT = MASS*XMTOF

C    CALCULATE WEIGHT EXPULSION RATES
WDOTTP = 0.0
WDOTTI = 0.0
WDOTKV = (- MDOTA - MDOTV)*XMTOF

C    CALCULATE MISSILE MOMENT OF INERTIA
CALL spTABLE(MASST2, INERXX, MASS, IXX, 20, III)
CALL spTABLE(MASST2, INERYY, MASS, IYY, 20, III)
CALL spTABLE(MASST2, INERZZ, MASS, IZZ, 20, III)

RETURN
END

```

## B.2.21 Uumcauto.for

```

C-----
      SUBROUTINE MCAUTO(T, IXX, IYY, IZZ, SP, SQ, SR, ROLLER, PITER, YAWER, IDIST,
      .               IACSON, IBURND, IBURNM, IDMEAS, IPASSM, ICMD, TRATON,
      .               TPATON, TYATON, DTSAMP, TSAL, TSAH, TLAPS, ITHRES,
      .               ANVP, ACSLEV, TMAUTO, initflag)
C-----

```

```

C
C      SUBROUTINE NAME :      MCAUTO
C
C      AUTHOR      :      R. RHYNE
C
C      FUNCTION    :      GENERATES ACS COMMANDS TO NULL LARGE
C                          ATTITUDE ERRORS AND RATES DURING MIDCOURSE
C
C      CALLED FROM :      FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :      T, IXX, IYY, IZZ, SP, SQ, SR, ROLLER, PITER,
C                      YAWER, IDIST, IACSON, IBURND, IBURNM, IDMEAS
C
C      OUTPUTS :      ICMD, TRATON, TPATON, TYATON, DTSAMP, TSAL, TSAH,
C                      TLAPS, ITHRES, ANVP, ACSLEV, TMAUTO
C
C      BOTH :      IPASSM
C
C      UPDATES :      B. HILL /      - CR # 081
C                      R. RHYNE
C                      D. SMITH      - CR # 082
C                      R. RHYNE      - CR # 083
C                      R. RHYNE      - CR # 087
C                      R. RHYNE      - CR # 090
C                      D. SMITH      - CR # 092
C                      B. HILL      - CR # 093
C-----

```

```

      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

```

```

      REAL  I1(3)          , ANGACL(3,4,10), OMEGAI(3)
      REAL  OMEGA(3)        , TBURNA(3)    , MOMARM(3)
      REAL  AERROR(3)       , OMEGAD       , AACCEL(3,4)
      REAL  IXX             , IYY          , IZZ
      INTEGER IMCPAS(3,4), initflag

```

```

C      COMMON "RMAUTO" USED FOR MIDFLIGHT CAPABILITIES ONLY

```

```

      COMMON / RMAUTO / ANGACL , IMCPAS , TP2END , TP3END , IP2END ,
      .               TCOAST , ICOAST , TRDONE , IRATE , IACSB1 ,
      .               IACSB2 , ICNT , IVPFL , IVPFLN , TBURN2 ,
      .               OMEGAI , TLSTMA , AACCEL

```

```

* DATA INITIALIZATION

```

```

$INCLUDE ('^/INCLUDE/SSDATA59.DAT')
$INCLUDE ('^/INCLUDE/SSDATA60.DAT')
$INCLUDE ('^/INCLUDE/SSDATA01.DAT')
$INCLUDE ('^/INCLUDE/SSDATA02.DAT')
$INCLUDE ('^/INCLUDE/SSDATA05.DAT')
$INCLUDE ('^/INCLUDE/SSDATA07.DAT')
$INCLUDE ('^/INCLUDE/SSDATA08.DAT')

```

```

$INCLUDE ('^/INCLUDE/SSDATA19.DAT')

IF ( IPASSM.EQ.0 ) THEN

C      INITIALIZE ACCELERATION TABLE, PULSE FLAGS, AND PULSE TIMES

      MOMARM(1) = RJARM
      MOMARM(2) = PIARM
      MOMARM(3) = YIARM
      II(1)     = IXX
      II(2)     = IYY
      II(3)     = IZZ
      DO 10 I = 1,3
        ANGACL(I,1,1) = 2.*ACSFL*MOMARM(I)/II(I)
        ANGACL(I,2,1) = 2.*ACSFH*MOMARM(I)/II(I)
        IF ( I.EQ.1 ) THEN
          ANGACL(I,3,1) = 4.*ACSFL*MOMARM(I)/II(I)
          ANGACL(I,4,1) = 4.*ACSFH*MOMARM(I)/II(I)
        ELSE
          ANGACL(I,3,1) = 0.
          ANGACL(I,4,1) = 0.
        ENDIF
        DO 4 J = 1,4
          IMCPAS(I,J) = 1
          AACCEL(I,J) = ANGACL(I,J,1)
          DO 3 K = 2,10
            ANGACL(I,J,K) = 0.
          CONTINUE
        CONTINUE
      CONTINUE
      IPASSM = 1
      ICNT   = 0
      IP2END = 1
      ICOAST = 1
      TP2END = 1000.0
      TP3END = 1000.0
      TCOAST = 1000.0
      TRDONE = 1000.0
    ENDIF

    if (initflag .ne. 0 ) then

C      TIME SINCE LAST CALL

      DTMCA = T - TLSTMA
      TLSTMA = T

C      DETERMINE IF CORRECTION REQUIRED AND ISSUE APPROPRIATE COMMAND

      IF ( ICMD.EQ.0 .AND. IDIST.EQ.0
        .AND. IBURNM.NE.0 .AND. IBURND.EQ.0 ) THEN

        IF ( ABS(ROLLER).GE.CAPHL ) THEN

C          COMPUTE INITIAL ROLL CORRECTION BURN TIME

          ICMD   = 1
          IVPFL  = 3
          IACSB1 = 1
          IF ( ABS(ROLLER).GE.4.*CAPHL ) IVPFL = 2
          OMEGAD = ROLLER*AACCEL(1,IVPFL)/ABS(ROLLER)
          IF ( SP/ROLLER.LT.0. ) THEN
            RLLER0 = ROLLER + SP**2/(2.*OMEGAD)
          ELSE

```

```

        RLLERO = ROLLER
    ENDIF
    TBACS = SQRT (ABS (RLLERO) / (2.*AACCEL(1,IVPFL))) - SP/OMEGAD

ELSEIF ( ABS(SP).GT.CRPHL ) THEN

C      DEFINE ROLL RATE CORRECTION COMMAND

        ICMD   = 1
        IRATE  = 1
        IACSB1 = 1
        IF ( ABS(SP).GT.750.*CRPH ) THEN
            IVPFL = 4
        ELSEIF ( ABS(SP).GT.375.*CRPH ) THEN
            IVPFL = 2
        ELSEIF ( ABS(SP).GT.15.*CRPH ) THEN
            IVPFL = 3
        ELSE
            IVPFL = 1
        ENDIF

ELSEIF ( IDMEAS.NE.2 ) THEN

        IF ( ABS(PITER).GT.CATHL ) THEN

C      COMPUTE INITIAL PITCH CORRECTION BURN TIME

            OMEGAD = PITER*AACCEL(2,2)/ABS(PITER)
            IF ( SQ/PITER.LT.0. ) THEN
                PITER0 = PITER + SQ**2/(2.*OMEGAD)
            ELSE
                PITER0 = PITER
            ENDIF
            TBACS = SQRT (ABS (PITER0) / (2.*AACCEL(2,2))) - SQ/OMEGAD

C      ISSUE PITCH COMMAND

            ICMD   = 2
            IVPFL  = 2
            IACSB1 = 1

        ELSEIF ( ABS(YAWER).GT.CAPSL ) THEN

            OMEGAD = YAWER*AACCEL(3,2)/ABS(YAWER)
            IF ( SR/YAWER.LT.0. ) THEN
                YAWER0 = YAWER + SR**2/(2.*OMEGAD)
            ELSE
                YAWER0 = YAWER
            ENDIF
            TBACS = SQRT (ABS (YAWER0) / (2.*AACCEL(3,2))) - SR/OMEGAD

C      ISSUE YAW COMMAND

            ICMD   = 3
            IVPFL  = 2
            IACSB1 = 1

        ELSEIF ( TSAH.GT.T+TSMPPH+EPSL .AND. IDMEAS.EQ.1 ) THEN

C      ENABLE KV AUTOPILOT

            TSAL   = T
            TSAH   = T
            TLAPS   = T

```

```

ENDIF
ELSEIF ( IBURND.EQ.0 ) THEN
C      NULL BODY RATES BEFORE DISTURBANCE PULSE ISSUED
      IF ( ABS(SQ).GE.CRTH ) THEN
        ICMD = 2
        IVPFL = 1
        IF ( ABS(SQ).GT.35.*CRTH ) IVPFL = 2
        IRATE = 1
        IACSB1 = 1
      ELSEIF ( ABS(SR).GE.CRPS ) THEN
        ICMD = 3
        IVPFL = 1
        IF ( ABS(SR).GT.35.*CRPS ) IVPFL = 2
        IRATE = 1
        IACSB1 = 1
      ENDIF
    ENDIF
  ENDIF
C      EXECUTE CONTROL LOGIC IF ATTITUDE/RATE CORRECTION REQUIRED
      IF ( ICMD.NE.0 ) THEN
C      ZERO ACS BURN VECTOR AND FORM ANGULAR RATE AND ERROR VECTORS
        TBURNA(1) = 0.
        TBURNA(2) = 0.
        TBURNA(3) = 0.

        OMEGA(1) = SP
        OMEGA(2) = SQ
        OMEGA(3) = SR

        AERROR(1) = ROLLER
        AERROR(2) = PITER
        AERROR(3) = YAWER
C      UPDATE ANGULAR ACCELERATION TABLE
        IF ( IACSON.EQ.1 ) THEN
          ICNT = ICNT + 1
          IF ( ICNT.EQ.1 ) OMEGAI(ICMD) = OMEGA(ICMD)
          IF ( ICNT.GE.2 ) THEN
            DO 12 I = IMCPAS(ICMD,IVPFL),1,-1
              IF (I.LT.10) ANGACL(ICMD,IVPFL,I+1) =
                ANGACL(ICMD,IVPFL,I)
12          CONTINUE
            ANGACL(ICMD,IVPFL,1)=ABS(OMEGAI(ICMD)-OMEGA(ICMD))/DTMCA
            OMEGAI(ICMD) = OMEGA(ICMD)
            IMCPAS(ICMD,IVPFL) = IMCPAS(ICMD,IVPFL) + 1
            IF (IMCPAS(ICMD,IVPFL).GE.ISAMP) IMCPAS(ICMD,IVPFL)=ISAMP
          ENDIF
        ELSE
          ICNT = 0
        ENDIF
C      COMPUTE EXPECTED ANGULAR ACCELERATION
        AACCEL(ICMD,IVPFL) = 0.0
        DO 20 I = 1,IMCPAS(ICMD,IVPFL)

```

```

20      AACCEL(ICMD,IVPFL) = AACCEL(ICMD,IVPFL)+ANGACL(ICMD,IVPFL,I)
      CONTINUE
      AACCEL(ICMD,IVPFL) = AACCEL(ICMD,IVPFL)/
                              (IMCPAS(ICMD,IVPFL))

C      EXECUTE BURN LOGIC

      IF ( IRATE.EQ.1 ) THEN

C          RATE CORRECTION

          IF ( IACSB1.EQ.1 ) THEN
              TBURNA(ICMD) = -OMEGA(ICMD)/AACCEL(ICMD,IVPFL)
              DTSAMP = ABS(TBURNA(ICMD))
              TRDONE = T + DTSAMP + TLAGA + TRDNA
              ITHRES = 1
              IACSB1 = 0
              ICNT = 0
              TSAL = 1000.
              TSAH = 1000.
              TLAPS = 1000.
          ELSEIF ( T.GE.TRDONE ) THEN
              TRDONE = 1000.
              IRATE = 0
              ICMD = 0
          ENDIF

      ELSEIF ( IACSB1.EQ.1 ) THEN

C          ENABLE FIRST ATTITUDE CONTROL PULSE

              TBURNA(ICMD) = AERROR(ICMD)*TBACS/ABS(AERROR(ICMD))
              DTSAMP = ABS(TBURNA(ICMD))
              ITHRES = 1
              TCOAST = T + DTSAMP + TLAGA + TRDNA
              ICOAST = 0
              IACSB1 = 0
              ICNT = 0
              TSAL = 1000.
              TSAH = 1000.
              TLAPS = 1000.

      ELSEIF ( T.GE.TCOAST .AND. ICOAST.EQ.0 ) THEN

C          COMPUTE SECOND BURN TO LEAVE DESIRED LOW LEVEL BURN

              ICOAST = 1
              IACSB2 = 1
              IF ( OMEGA(ICMD).LT.0. ) THEN
                  DIRECT = -1.
              ELSE
                  DIRECT = 1.
              ENDIF
              IF ( ICMD.EQ.1 .AND. IVPFL.EQ.2 ) THEN
                  IVPFLN = 3
              ELSE
                  IVPFLN = 1
              ENDIF
              TBURN2=(OMEGA(ICMD)-DIRECT*AACCEL(ICMD,IVPFLN)*TBURN3)
                              /AACCEL(ICMD,IVPFL)

      ELSEIF ( T.GE.TCOAST .AND. IACSB2.EQ.1 ) THEN

C          ENABLE ACS BURN WHEN ATTITUDE ERROR EQUALS EXPECTED

```

```

C          DISTANCE FROM DESIRED LOW LEVEL THIRD PULSE ERROR

          THET2D = OMEGA(ICMD) - AACCEL(ICMD, IVPFL) * TBURN2
          THT2DD = -DIRECT * AACCEL(ICMD, IVPFLN)
          THT1DD = -DIRECT * AACCEL(ICMD, IVPFL)
          DELANG = 0.5 * (THET2D**2 - OMEGA(ICMD)**2) / THT1DD +
          2. * THET2D * TBURN3 - 0.5 * THET2D**2 / THT2DD
          DELNXT = AERROR(ICMD) - OMEGA(ICMD) * DTMCU
          IF ( ABS(DELANG) .GE. ABS(DELNXT) ) THEN
              IACSB2 = 0
              ICNT = 0
              TBURNA(ICMD) = -TBURN2
              DTSAMP = ABS(TBURNA(ICMD))
              ITHRES = 1
              IP2END = 1
              TP2END = T + DTSAMP + TLAGA + TRDNA
              DELANG = 0.
          ENDIF

          ELSEIF ( T.GE.TP2END .AND. IP2END.EQ.1 ) THEN

C          DEFINE LOW LEVEL ACS PULSE FOR 'FINE TUNING'

          DELANG = 0.5 * OMEGA(ICMD)**2 / AACCEL(ICMD, IVPFLN)
          DELNXT = AERROR(ICMD) - OMEGA(ICMD) * DTMCU
          TDELAN = (ABS(AERROR(ICMD)) - DELANG) / ABS(OMEGA(ICMD))
          IF ( DELANG.GE.ABS(DELNXT) .OR. TDELAN.GT.2.5 * TBURN3 .OR.
              OMEGA(ICMD) / AERROR(ICMD) .LT.0. ) THEN
              IP2END = 0
              ICNT = 0
              TBURNA(ICMD) = -OMEGA(ICMD) / AACCEL(ICMD, IVPFLN)
              DTSAMP = ABS(TBURNA(ICMD))
              ITHRES = 1
              IVPFL = IVPFLN
              TP3END = T + DTSAMP + TLAGA + TRDNA
          ENDIF

          ELSEIF ( T.GE.TP3END ) THEN

C          CORRECTION COMPLETE FOR Ith AXIS

          TP3END = 1000.
          DELANG = 0.
          ICMD = 0

          ENDIF
        ENDIF

C      DEFINE ACS LEVEL AND VALVE PAIR CONFIGURATION BASED ON
C      ACCELERATION TABLE INDEX USED

        IF ( IVPFL.EQ.4 ) THEN
            ACSLEV = 2.
            ANVP = 2.
        ELSEIF ( IVPFL.EQ.3 ) THEN
            ACSLEV = 1.
            ANVP = 2.
        ELSEIF ( IVPFL.EQ.2 ) THEN
            ACSLEV = 2.
            ANVP = 1.
        ELSE
            ACSLEV = 1.
            ANVP = 1.
        ENDIF

```

```
C      UPDATE ACS BURN COMMANDS

      TRATON = TBURNA(1)
      TPATON = TBURNA(2)
      TYATON = TBURNA(3)

C      CALCULATE NEXT TIME TO CALL

      TMAUTO = T + DTMCU - EPSL
      endif

      RETURN
      END
```

## B.2.22 Uumcguid.for

```

C-----
C      SUBROUTINE MCGUID (T, TI2M, VG, URREL, MASS, IDIST, MIDBRN, MAGR, MAGV, SP,
C      .                  SQ, SR, PITER, YAWER, FLIP, IVCS, ICMD, IDMEAS, IDPASS,
C      .                  IDROP, IMCEND, IBURND, IBURNM, VGM, ADISTT, ROLLER,
C      .                  TMGUID)
C-----
C
C      SUBROUTINE NAME :      MCGUID
C
C      AUTHOR      :      R. RHYNE
C
C      FUNCTION    :      DEFINES ROLL ERROR, SEQUENCES MIDCOURSE
C      .            .      EVENTS, AND ENABLES MIDCOURSE DIVERTS
C
C      CALLED FROM :      FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS      :      T, TI2M, VG, URREL, MASS, IDIST, MIDBRN, MAGR,
C      .            .      MAGV, SP, SQ, SR, PITER, YAWER, FLIP, ICMD
C
C      OUTPUTS     :      IDMEAS, IDPASS, IMCEND, IBURND, IBURNM, VGM,
C      .            .      ADISTT, ROLLER, TMGUID
C
C      BOTH       :      IDROP
C
C      UPDATES    :      B. HILL /   - CR # 081
C      .            .      R. RHYNE
C      .            .      R. RHYNE   - CR # 083
C      .            .      R. RHYNE   - CR # 084
C      .            .      R. RHYNE   - CR # 087
C      .            .      R. RHYNE   - CR # 090
C      .            .      B. HILL    - CR # 093
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C
C      CHARACTER*128 MESSAGE
C      REAL TI2M(9)      , VG(3)      , URREL(3)
C      REAL MASS      , MAGR      , MAGV
C      REAL VGM(3)      , ADISTT(4,3) , OMEGA0(3)
C      REAL VGP(3)      , VGPM(3)    , ACQRNG(4,4)
C      REAL RATE(6)     , TRGSIG(4)
C      INTEGER      ISEQ(4)      , FLIP      , SEKTYP
C      INTEGER      BCKGRD
C
C      LOCAL COMMON USED FOR CONSTANTS AND INITIALIZATION FLAG
C
C      SAVE      IMGUID
C
C      COMMON "RMGUID" USED FOR MIDFLIGHT CAPABILITIES ONLY
C
C      COMMON / RMGUID / ISEQ      , TVCOMP , OMEGA0 , IMIDB2 , TMIDB2 ,
C      .            .      ISK3ON
C
C      * DATA INITIALIZATION
C      $INCLUDE ('~/INCLUDE/SSDATA46.DAT')
C      $INCLUDE ('~/INCLUDE/SSDATA48.DAT')
C      $INCLUDE ('~/INCLUDE/SSDATA50.DAT')

```

```

$INCLUDE ('^/INCLUDE/SSDATA55.DAT')
$INCLUDE ('^/INCLUDE/SSDATA60.DAT')
$INCLUDE ('^/INCLUDE/SSDATA61.DAT')
$INCLUDE ('^/INCLUDE/SSDATA62.DAT')
$INCLUDE ('^/INCLUDE/SSDATA01.DAT')
$INCLUDE ('^/INCLUDE/SSDATA04.DAT')
$INCLUDE ('^/INCLUDE/SSDATA05.DAT')
$INCLUDE ('^/INCLUDE/SSDATA09.DAT')
$INCLUDE ('^/INCLUDE/SSDATA12.DAT')
$INCLUDE ('^/INCLUDE/SSDATA13.DAT')
$INCLUDE ('^/INCLUDE/SSDATA17.DAT')

DATA IMGUID / 1 /

IF ( IMGUID .EQ. 1 ) THEN
  IMGUID = 0
  IF ( SEKTYP.EQ.2 ) THEN
    TSIG = TRGSIG(ITRGSG)
    TSGACQ = TSIG
    RAQREF = ACQRNG(BCKGRD, ITRGSG)
    RNGAQ = SQRT((TSGACQ/TSIG)*(6.0/SNRACQ)*
                  (SQRT(1./RATE(1))))*RAQREF
  ELSE IF ( SEKTYP.EQ.3 ) THEN
    RNGAQ = ACQR3
  ELSE
    RNGAQ = RNGACQ
  ENDIF
ENDIF

C GET VG IN BODY FRAME

VGM(1) = TI2M(1)*VG(1) + TI2M(4)*VG(2) + TI2M(7)*VG(3)
VGM(2) = TI2M(2)*VG(1) + TI2M(5)*VG(2) + TI2M(8)*VG(3)
VGM(3) = TI2M(3)*VG(1) + TI2M(6)*VG(2) + TI2M(9)*VG(3)

C CALCULATE ROLL ERROR IF KV REORIENTATION AND UPLINK HAVE OCCURRED

IF ( FLIP.EQ.0 .AND. T.GE.TUPLK1 .AND. IMCEND.EQ.0 ) THEN

  VGDLOS = URREL(1)*VG(1) + URREL(2)*VG(2) + URREL(3)*VG(3)

C DETERMINE PERPENDICULAR COMPONENT OF VG

VGP(1) = VG(1) - VGDLOS*URREL(1)
VGP(2) = VG(2) - VGDLOS*URREL(2)
VGP(3) = VG(3) - VGDLOS*URREL(3)

C GET VGP IN BODY FRAME

VGPM(1) = TI2M(1)*VGP(1) + TI2M(4)*VGP(2) + TI2M(7)*VGP(3)
VGPM(2) = TI2M(2)*VGP(1) + TI2M(5)*VGP(2) + TI2M(8)*VGP(3)
VGPM(3) = TI2M(3)*VGP(1) + TI2M(6)*VGP(2) + TI2M(9)*VGP(3)

IF ( VGPM(3).NE.0.0 ) THEN
  RERR = -ATAN2(VGPM(2),VGPM(3))
ELSE
  PIO2 = PI/2.
  RERR = -SIGN(PIO2,X)
ENDIF

C ESTIMATE REQUIRED DIVERT DURATION

ACM = FLATM/MASS
TBURNY = ABS(VGPM(2)/ACM)

```

```

TBURNZ = ABS(VGPM(3)/ACM)
TBURN  = AMAX1(TBURNY, TBURNZ)

C      BYPASS MAJOR ROLL CORRECTION IF BURN TIME ALONG EITHER
C      AXIS IS BELOW VCS BURN THRESHOLD

      IF ( TBURN.LT.TCMINV .AND. ICMD.EQ.0 ) THEN
        ROLLER = 0.
        IVCS   = 0
      ELSE IF ( ABS(TBURNY).LT.TCMINV .AND. ICMD.EQ.0 ) THEN
        ROLLER = 0.
        IF ( VGPM(3) .GT. 0. ) THEN
          IVCS = 4
        ELSE
          IVCS = 2
        ENDIF
      ELSE IF ( ABS(TBURNZ).LT.TCMINV .AND. ICMD.EQ.0 ) THEN
        ROLLER = 0.
        IF ( VGPM(2) .GT. 0. ) THEN
          IVCS = 3
        ELSE
          IVCS = 1
        ENDIF
      ENDIF

C      DEFINE ROLL ERROR TO ALIGN VGPM WITH NEAREST VCS THRUSTER

      ELSE IF ( ICMD .EQ. 0 ) THEN
        IF ( ABS(RERR) .LE. PI/4. ) THEN
          ROLLER = RERR
          IVCS   = 4
        ELSE IF ( RERR .LE. -3.*PI/4. ) THEN
          ROLLER = PI + RERR
          IVCS   = 2
        ELSE IF ( RERR .GE. 3.*PI/4. ) THEN
          ROLLER = RERR - PI
          IVCS   = 2
        ELSE IF ( RERR.LT.3.*PI/4. .AND. RERR.GT.PI/4. ) THEN
          ROLLER = RERR - PI/2.
          IVCS   = 1
        ELSE
          ROLLER = RERR + PI/2.
          IVCS   = 3
        ENDIF
      ENDIF

C      IF ATTITUDE CORRECTION IN PROGRESS, USE SAME
C      ROLL ERROR CALCULATION

      ELSE
        IF ( IVCS .EQ. 1 ) THEN
          ROLLER = RERR - PI/2.
        ELSE IF ( IVCS .EQ. 2 ) THEN
          IF ( RERR .LT. 0. ) THEN
            ROLLER = PI + RERR
          ELSE
            ROLLER = RERR - PI
          ENDIF
        ELSE IF ( IVCS .EQ. 3 ) THEN
          ROLLER = RERR + PI/2.
        ELSE
          ROLLER = RERR
        ENDIF
      ENDIF
    ELSE

```

```

C      ZERO ROLL ERROR IF PITCHOVER AND FIRST UPLINK HAVE NOT OCCURRED
      ROLLER = 0.
    ENDIF

    IF ( IDMEAS.EQ.0 .AND. ICMD.EQ.0 .AND. ABS(PITER).LE.CATHL
      .AND. ABS(YAWER).LE.CAPSL .AND. (IGIT.EQ.0 .OR.
      . (IGIT.EQ.1 .AND. T.GE.TDROP)) ) THEN

C      ENTER DISTURBANCE MEASUREMENT MODE

      WRITE(MESSAGE,10) T
      CALL OUTMES(MESSAGE)
10     FORMAT(1X,E16.9,' KV PITCHOVER COMPLETE - BEGIN',
      . ' DISTURBANCE MEASUREMENT')
      IDMEAS = 2
    ENDIF

    IF ( IDMEAS.EQ.2 .AND. ABS(SP).LE.CRPHL .AND. ABS(SQ).LE.CRTH
      .AND. ABS(SR).LE.CRPS .AND. ICMD.EQ.0 ) THEN

      IF ( IDPASS .EQ. 0 ) THEN

C      DEFINE VCS DISTURBANCE SEQUENCE

        IF ( ABS(VGM(2)) .GE. ABS(VGM(3)) ) THEN
          INDEXY = 1
          INDEXZ = 3
        ELSE
          INDEXY = 3
          INDEXZ = 1
        ENDIF
        IF ( VGM(2) .GE. 0. ) THEN
          ISEQ(INDEXY) = 3
          ISEQ(INDEXY+1) = 1
        ELSE
          ISEQ(INDEXY) = 1
          ISEQ(INDEXY+1) = 3
        ENDIF
        IF ( VGM(3) .GE. 0. ) THEN
          ISEQ(INDEXZ) = 4
          ISEQ(INDEXZ+1) = 2
        ELSE
          ISEQ(INDEXZ) = 2
          ISEQ(INDEXZ+1) = 4
        ENDIF
        IDPASS = 1
      ENDIF

      IF ( IBURND .EQ. 0 ) THEN

C      DROP BOOST ADAPTER AND NOSE FAIRING PRIOR TO FIRST
C      DISTURBANCE BURN - IF EVENT DRIVEN LOGIC, SCHEDULE
C      SEPARATION HERE - OTHERWISE, SEPARATION WILL OCCUR
C      AT T=TDROP IN MAIN ROUTINE

        IF ( IDROP.EQ.0 .AND. IGIT.EQ.0 ) THEN
          IDROP = 1
        ELSE

C      DEFINE Ith DISTURBANCE BURN

```

```

        IBURND = 1
        IBURNM = 0
        TVCOMP = T + TLAGV + TBURND + TRDNV + TIWAIT
        IVCS = ISEQ(IDPASS)
        OMEGA0(1) = SP
        OMEGA0(2) = SQ
        OMEGA0(3) = SR
    ENDIF

ELSE IF ( T .GT. TVCOMP ) THEN

C      COMPUTE ANGULAR ACCEL INDUCED BY PREVIOUS DISTURBANCE BURN

        IBURND = 0
        ADISTT(ISEQ(IDPASS),1) = (SP - OMEGA0(1))/TBURND
        ADISTT(ISEQ(IDPASS),2) = (SQ - OMEGA0(2))/TBURND
        ADISTT(ISEQ(IDPASS),3) = (SR - OMEGA0(3))/TBURND
        IDPASS = IDPASS + 1
        TVCOMP = 1000.
        IF ( IDPASS .GT. 4 ) THEN
            IDMEAS = 1
            WRITE(MESSAGE,15) T
            CALL OUTMES(MESSAGE)
15      FORMAT(1X,E16.9,' DISTURBANCE MEASUREMENT COMPLETE -',
              ' ORIENT KV TO LOS')
        ENDIF
    ENDIF
ENDIF

C      ENABLE SEEKER AFTER PITCHOVER AND DISTURBANCE
C      MEASUREMENT COMPLETED

        IF ( ABS(PITER).LE.CATH .AND. ABS(YAWER).LE.CAPS
          .AND. ABS(SQ).LE.CRTH .AND. ABS(SR).LE.CRPS
          .AND. FLIP.EQ.1 .AND. IDMEAS.EQ.1 ) THEN

C      ENABLE SEEKER (TYPES 0,1,&2) IF EVENT DRIVEN LOGIC -
C      OTHERWISE WILL BE ENABLED BY MAIN ROUTINE AT SECOND
C      STAGE SEPARATION - SEEKER TYPE 3 HANDLED BELOW -
C      TYPE 3 ENABLED BY MAIN ROUTINE AT T=TSK3ON IF EVENT
C      LOGIC NOT USED

        FLIP = 0
        WRITE(MESSAGE,20) T
        CALL OUTMES(MESSAGE)
20      FORMAT(1X,E16.9,' KV ORIENTATION COMPLETE')
    ENDIF

        IF ( SEKTYP.EQ.3 .AND. IGIT.EQ.0 .AND. MAGR.LE.ACQR3
          .AND. ISK3ON.EQ.0 ) THEN
            ISK3ON = 1
            WRITE(MESSAGE,30) T
            CALL OUTMES(MESSAGE)
30      FORMAT(1X,E16.9,' SEEKER 3 ENABLED')
        ENDIF

C      DEFINE THREE MIDCOURSE DIVERTS

        IF ( ABS(ROLLER).LE.CAPH .AND. ABS(SP).LE.CRPH
          .AND. ICMD.EQ.0 .AND. T.GT.TUPLK1 ) THEN
            DELMID = ( MAGR - RNGAQ )/MAGV
            IF ( ICMD.EQ.0 .AND. MIDBRN.EQ.0 ) THEN
                IBURNM = 0
                IMIDB2 = 1
            
```

```

ELSE IF ( IDIST.EQ.0 .AND. MIDBRN.EQ.1 .AND. IMIDB2.EQ.1 ) THEN
  TMIDB2 = T + 0.5*DELMID
  IMIDB2 = 0
ELSE IF ( T.GE.TMIDB2 .AND. MIDBRN.EQ.1 ) THEN
  IBURNM = 0
ELSE IF ( IDIST.EQ.0 .AND. MIDBRN.EQ.2 ) THEN
  TMAX = TBURN + TBWAIT
  IF ( DELMID .LE. TMAX+DTMCU ) THEN
    IBURNM = 0
    ROLLER = 0.
    IMCEND = 1
  ENDIF
ENDIF
ENDIF

```

```

C      COMPUTE TIME OF NEXT CALL

      TMGUID = T + DTMCU - EPSL

      RETURN
      END

```

## B.2.23 Uumissil.for

```

C-----
      SUBROUTINE MISSIL(T,CIM,MASS,
      .             FXACS,FXVCS,FYACS,FYVCS,
      .             FZACS,FZVCS,
      .             X,Y,Z,NCLEAR,UD,VD,WD,
      .             GB,GR,MGR,FX,FY,FZ,XDD,YDD,ZDD,MXYZDD)
C-----
C
C      SUBROUTINE NAME :      MISSIL
C
C      AUTHOR(S) :          D. C. FOREMAN, A. P. BUKLEY
C
C      FUNCTION :           COMPUTES THE ROTATIONAL AND TRANSLATIONAL
C                           MISSILE ACCELERATIONS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  FVDOT,FV2BXI
C
C      INPUTS :             T,QUAT,CIM,P,Q,R,IXX,IYY,IZZ,MASS,FXA,
C                           FXT,FRCX,FXACS,FXVCS,FYA,FYT,FRCY,FYACS,
C                           FYVCS,FZA,FZT,FRCZ,FZACS,FZVCS,MXA,MXT,
C                           MRCX,MXACS,MXVCS,MYA,MYT,MRCY,MYACS,MYVCS,
C                           MZA,MZT,MRCZ,MZACS,MZVCS,X,Y,Z,XD,YD,ZD
C
C      OUTPUTS :            UD,VD,WD,PD,QD,RD,GB,GR,MGR,MX,MY,MZ,FX,FY,
C                           FZ,XDD,YDD,ZDD,MXYZDD,U,V,W,QUATD,PHI,THT,
C                           PSI
C
C      BOTH :               NCLEAR
C
C      UPDATES :            D. SISSOM   - CR # 011
C                           T. THORNTON - CR # 012
C                           T. THORNTON - CR # 018
C                           B. HILL     - CR # 030
C                           T. THORNTON - CR # 031
C                           T. THORNTON - CR # 033
C                           T. THORNTON - CR # 035
C                           T. THORNTON - CR # 037
C                           T. THORNTON - CR # 049
C                           T. THORNTON - CR # 050
C                           D. SMITH    - CR # 059
C                           D. SMITH    - CR # 060
C                           B. HILL     - CR # 062
C                           D. SMITH    - CR # 076
C                           R. RHYNE    - CR # 079
C                           B. HILL /   - CR # 081
C                           R. RHYNE
C                           R. RHYNE    - CR # 087
C                           B. HILL     - CR # 093
C-----

```

```

      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      DOUBLE PRECISION  CIM(9)        , CMT(9)        , GB(3)
      DOUBLE PRECISION  GR(3)         , MASS          , MGR
      DOUBLE PRECISION  MXYZ
      DOUBLE PRECISION  MXYZDD
      DOUBLE PRECISION  UXYZ(3)

```

```

DOUBLE PRECISION UXYZDD(3)      , XYZLCH(3)

C   LOCAL COMMON USED FOR CONSTANTS, LOCAL VARIABLES AND
C   INITIALIZATION FLAG

      SAVE                IMISL

C   COMMON "RMISSL" USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RMISSL / XYZLCH

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSDATA39.DAT')
$INCLUDE('~/INCLUDE/SSDATA63.DAT')
$INCLUDE('~/INCLUDE/SSDATA28.DAT')

C   COMPUTE MISSILE LAUNCH POSITION IN INERTIAL FRAME

      CMI(1) = CIM(1)
      CMI(2) = CIM(4)
      CMI(3) = CIM(7)
      CMI(4) = CIM(2)
      CMI(5) = CIM(5)
      CMI(6) = CIM(8)
      CMI(7) = CIM(3)
      CMI(8) = CIM(6)
      CMI(9) = CIM(9)

C   DETERMINE LOCAL GRAVITY VECTOR

      MXYZ = DSQRT ( X**2 + Y**2 + Z**2 )
      MGR = GMU / MXYZ**2

      IF ( MXYZ.GT.0.0D0 ) THEN
        UXYZ(1) = X / MXYZ
        UXYZ(2) = Y / MXYZ
        UXYZ(3) = Z / MXYZ
      ELSE
        UXYZ(1) = 0.0D0
        UXYZ(2) = 0.0D0
        UXYZ(3) = 0.0D0
      ENDIF

C   CALCULATE GRAVITY VECTOR IN INERTIAL AND BODY FRAMES

      GR(1) = - MGR*UXYZ(1)
      GR(2) = - MGR*UXYZ(2)
      GR(3) = - MGR*UXYZ(3)

      GB(1) = CIM(1)*GR(1) + CIM(4)*GR(2) + CIM(7)*GR(3)
      GB(2) = CIM(2)*GR(1) + CIM(5)*GR(2) + CIM(8)*GR(3)
      GB(3) = CIM(3)*GR(1) + CIM(6)*GR(2) + CIM(9)*GR(3)

C   CALCULATE TOTAL FORCES AND MOMENTS

      FX = FXACS + FXVCS
      FY = FYACS + FYVCS
      FZ = FZACS + FZVCS

C   MISSILE CLEARED THE LAUNCHER

      IF ( NCLEAR.EQ.1 ) THEN
        UD = FX/MASS + GB(1)

```

```

      VD      = FY/MASS + GB(2)
      WD      = FZ/MASS + GB(3)
ENDIF

```

C      TRANSFORM BODY ACCELERATIONS TO INERTIAL FRAME

```

XDD = CMI(1)*UD + CMI(4)*VD + CMI(7)*WD
YDD = CMI(2)*UD + CMI(5)*VD + CMI(8)*WD
ZDD = CMI(3)*UD + CMI(6)*VD + CMI(9)*WD

MXYZDD = DSQRT ( XDD**2 + YDD**2 + ZDD**2 )
IF ( MXYZDD.GT.0.0D0 ) THEN
  UXYZDD(1) = XDD / MXYZDD
  UXYZDD(2) = YDD / MXYZDD
  UXYZDD(3) = ZDD / MXYZDD
ELSE
  UXYZDD(1) = 0.0D0
  UXYZDD(2) = 0.0D0
  UXYZDD(3) = 0.0D0
ENDIF

RETURN
END

```

## B.2.24 Uumissl2.for

```

C-----
C      SUBROUTINE MISSIL2 (T, QUAT, CIM, P, Q, R, IXX, IYY, IZZ,
C      .                   MXACS, MXVCS, MYACS, MYVCS, MZACS,
C      .                   MZVCS, XD, YD, ZD, NCLEAR, PD, QD, RD,
C      .                   MX, MY, MZ, U, V, W, QUATD, PHI, THT, PSI)
C-----
C
C      SUBROUTINE NAME :      MISSIL
C
C      AUTHOR(S) :          D. C. FOREMAN, A. P. BUKLEY
C
C      FUNCTION :           COMPUTES THE ROTATIONAL AND TRANSLATIONAL
C                           MISSILE ACCELERATIONS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  FVDOT, FV2BXI
C
C      INPUTS :             T, QUAT, CIM, P, Q, R, IXX, IYY, IZZ,
C                           MXA,
C                           MXACS, MXVCS, MYACS, MYVCS,
C                           MZACS, MZVCS, XD, YD, ZD
C
C      OUTPUTS :            PD, QD, RD, MX, MY, MZ,
C                           U, V, W, QUATD, PHI, THT,
C                           PSI
C
C      BOTH :               NCLEAR
C
C      UPDATES :            D. SISSOM   - CR # 011
C                           T. THORNTON - CR # 012
C                           T. THORNTON - CR # 018
C                           B. HILL    - CR # 030
C                           T. THORNTON - CR # 031
C                           T. THORNTON - CR # 033
C                           T. THORNTON - CR # 035
C                           T. THORNTON - CR # 037
C                           T. THORNTON - CR # 049
C                           T. THORNTON - CR # 050
C                           D. SMITH   - CR # 059
C                           D. SMITH   - CR # 060
C                           B. HILL    - CR # 062
C                           D. SMITH   - CR # 076
C                           R. RHYNE   - CR # 079
C                           B. HILL /  - CR # 081
C                           R. RHYNE   -
C                           R. RHYNE   - CR # 087
C                           B. HILL    - CR # 093
C-----
C
C      IMPLICIT REAL      (A-H)
C      IMPLICIT REAL      (O-Z)
C
C      REAL  CIM(9)        , CMI(9)
C      REAL  IXX           , IYY
C      REAL  IZZ
C      REAL  MX            , MXACS
C      REAL  MXVCS
C      REAL  MY
C      REAL  MYACS         , MYVCS

```

```

REAL  MZ          , MZACS
REAL  MZVCS       , PQR(3)
REAL  QUAT(4)     , QUATD(4)
REAL  XYZLCH(3)

C      LOCAL COMMON USED FOR CONSTANTS, LOCAL VARIABLES AND
C      INITIALIZATION FLAG

      SAVE          IMISL

C      COMMON "RMISSL" USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RMISSL / XYZLCH

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSDATA39.DAT')
$INCLUDE('~/INCLUDE/SSDATA63.DAT')
$INCLUDE('~/INCLUDE/SSDATA28.DAT')

      DATA IMISL / 1 /

      IF (IMISL .EQ. 1) THEN

        IMISL = 0

C      COMPUTE MISSILE LAUNCH POSITION IN INERTIAL FRAME

        CMI(1) = CIM(1)
        CMI(2) = CIM(4)
        CMI(3) = CIM(7)
        CMI(4) = CIM(2)
        CMI(5) = CIM(5)
        CMI(6) = CIM(8)
        CMI(7) = CIM(3)
        CMI(8) = CIM(6)
        CMI(9) = CIM(9)

      ENDIF

C      CALCULATE TOTAL FORCES AND MOMENTS

      MX      = MXACS + MXVCS
      MY      = MYACS + MYVCS
      MZ      = MZACS + MZVCS

C      MISSILE CLEARED THE LAUNCHER

      IF ( NCLEAR.EQ.1 ) THEN
        PD      = MX/IXX + Q*R*((IYY-IZZ)/IXX)
        QD      = MY/IYY + R*P*((IZZ-IXX)/IYY)
        RD      = MZ/IZZ + P*Q*((IXX-IYY)/IZZ)
      ENDIF

C      COMPUTE QUATERNION DERIVATIVES

      PQR(1) = P
      PQR(2) = Q
      PQR(3) = R

      TMP1    = 0.0
      CALL FVDOT(PQR,TMP1,QUAT,QUATD)

C      COMPUTE BODY-TO-INERTIAL TRANSFORMATION MATRIX

```

```

CALL FV2BXI (QUAT, TMP1, CMI)

CIM(1) = CMI(1)
CIM(2) = CMI(4)
CIM(3) = CMI(7)
CIM(4) = CMI(2)
CIM(5) = CMI(5)
CIM(6) = CMI(8)
CIM(7) = CMI(3)
CIM(8) = CMI(6)
CIM(9) = CMI(9)

C   COMPUTE EULER ANGLES

PHI   = ATAN2 (CIM(8), CIM(9))
THT   = -ASIN (CIM(7))
PSI   = ATAN2 (CIM(4), CIM(1))

C   TRANSFORM INERTIAL VELOCITY TO BODY FRAME

U      = CIM(1)*XD + CIM(4)*YD + CIM(7)*ZD
V      = CIM(2)*XD + CIM(5)*YD + CIM(8)*ZD
W      = CIM(3)*XD + CIM(6)*YD + CIM(9)*ZD

RETURN
END

```

## B.2.25 Uummk.for

```

C-----
C      SUBROUTINE MMK (A, NA, B, NB, C, NC, RM)
C-----
C
C      SUBROUTINE NAME :      MMK
C
C      AUTHOR(S) :          J. SHEEHAN
C
C      FUNCTION :           GENERATES A DIRECTION COSINE MATRIX
C                          BY ROTATING IN ORDER:
C                          1) ANGLE C ABOUT THE NC AXIS
C                          2) ANGLE B ABOUT THE NB AXIS
C                          3) ANGLE A ABOUT THE NA AXIS
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  ROTMX, MMLXY
C
C      INPUTS :             A, NA, B, NB, C, NC
C
C      OUTPUTS :            RM
C
C      UPDATES :            D. SMITH    - CR # 59
C-----
C
C      IMPLICIT DOUBLE PRECISION (A-H)
C      IMPLICIT DOUBLE PRECISION (O-Z)
C
C      DIMENSION AM(3,3), BM(3,3), CM(3,3), RM(3,3), T(9)
C
C      CALL ROTMX(A, NA, AM)
C      CALL ROTMX(B, NB, BM)
C      CALL ROTMX(C, NC, CM)
C
C      CALL MMLXY(BM, CM, T)
C      CALL MMLXY(AM, T, RM)
C
C      RETURN
C      END

```

## B.2.26 Uummlxy.for

```

C-----
C      SUBROUTINE MMLXY(X,Y,Z)
C-----
C
C      SUBROUTINE NAME :      MMLXY
C
C      AUTHOR(S) :          J. SHEEHAN
C
C      FUNCTION :            MULTIPLY TWO 3X3 MATRICES
C
C      CALLED FROM :         UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :              X, Y
C
C      OUTPUTS :             Z
C
C      UPDATES :             D. SMITH    - CR # 59
C-----
C
C      IMPLICIT DOUBLE PRECISION (A-H)
C      IMPLICIT DOUBLE PRECISION (O-Z)
C
C      DIMENSION X(3,3), Y(3,3), Z(3,3)
C
C      Z(I,J) = X(I,1)*Y(1,J) + X(I,2)*Y(2,J) + X(I,3)*Y(3,J)
C
C      Z(1,1) = X(1,1)*Y(1,1) + X(1,2)*Y(2,1) + X(1,3)*Y(3,1)
C      Z(2,1) = X(2,1)*Y(1,1) + X(2,2)*Y(2,1) + X(2,3)*Y(3,1)
C      Z(3,1) = X(3,1)*Y(1,1) + X(3,2)*Y(2,1) + X(3,3)*Y(3,1)
C      Z(1,2) = X(1,1)*Y(1,2) + X(1,2)*Y(2,2) + X(1,3)*Y(3,2)
C      Z(2,2) = X(2,1)*Y(1,2) + X(2,2)*Y(2,2) + X(2,3)*Y(3,2)
C      Z(3,2) = X(3,1)*Y(1,2) + X(3,2)*Y(2,2) + X(3,3)*Y(3,2)
C      Z(1,3) = X(1,1)*Y(1,3) + X(1,2)*Y(2,3) + X(1,3)*Y(3,3)
C      Z(2,3) = X(2,1)*Y(1,3) + X(2,2)*Y(2,3) + X(2,3)*Y(3,3)
C      Z(3,3) = X(3,1)*Y(1,3) + X(3,2)*Y(2,3) + X(3,3)*Y(3,3)
C
C      RETURN
C      END

```

## B.2.27 Unnavig.for

```

C-----
C      SUBROUTINE NAVIG(T,MASS,DELPHI,DELTHT,DELPSI,DELU,DELV,DELW,GR,
C          .              QS1,CIE,SP,SQ,SR,SUD,SVD,SWD,VMIR,RMIR,TI2M,
C          .              SPHI,STHT,SPSI,SU,SV,SW,AT,VMI,RMI)
C-----
C
C      SUBROUTINE NAME :      NAVIG
C
C      AUTHOR(S) :          B. HILL
C
C      FUNCTION :           COMPUTES THE QUATERNIONS AND TRANSFORMATION
C                          MATRICES USING DELTA ANGLES SENSED BY THE
C                          GYRO. COMPUTES THE POSITION AND VELOCITY IN
C                          INERTIAL AND EARTH-CENTERED FRAMES.
C                          COMPUTES SENSED BODY RATES, EULER ANGLES AND
C                          THE GRAVITY-COMPENSATED ACCELERATION.
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             T,MASS,DELPHI,DELTHT,DELPSI,DELU,DELV,DELW,
C                          GR,CIE
C
C      OUTPUTS :            QS1,TI2M,SPHI,STHT,SPSI,SU,SV,SW,AT,VMI,RMI
C
C      BOTH :               SP,SQ,SR,SUD,SVD,SWD,VMIR,RMIR
C
C      UPDATES :            T. THORNTON - CR # 016
C                          B. HILL      - CR # 019
C                          B. HILL      - CR # 022
C                          B. HILL      - CR # 030
C                          T. THORNTON - CR # 033
C                          T. THORNTON - CR # 037
C                          D. SMITH     - CR # 059
C                          B. HILL      - CR # 062
C                          D. SISSOM    - CR # 069
C                          D. SMITH     - CR # 070
C                          D. SMITH     - CR # 075
C                          D. SMITH     - CR # 076
C                          B. HILL /    - CR # 081
C                          R. RHYNE
C                          R. RHYNE     - CR # 087
C                          B. HILL      - CR # 089
C                          D. SMITH     - CR # 092
C                          B. HILL      - CR # 093
C-----
C
C      IMPLICIT DOUBLE PRECISION      (A-H)
C      IMPLICIT DOUBLE PRECISION      (O-Z)
C
C      DOUBLE PRECISION  VMIR(3)      ,  RMIR(3)      ,  VMI(3)
C      DOUBLE PRECISION  RMI(3)       ,  TI2M(9)
C      DOUBLE PRECISION  GR(3)        ,  CIE(9)       ,  AT(3)
C      DOUBLE PRECISION  QS1(4)       ,  MASS         ,  GRAVG(3)
C      DOUBLE PRECISION  GRLAST(3)
C
C      LOCAL COMMON USED FOR LOCAL VARIABLES AND
C      INITIALIZATION FLAG

```

```

SAVE                INAVIG

C    COMMON "RNAVIG" USED FOR MIDFLIGHT CAPABILITIES ONLY

COMMON / RNAVIG / GRLAST , TONAV , MNAV , DTX0 , DTY0 ,
      DTZ0

DATA INAVIG / 1 /

IF ( INAVIG.EQ.1 ) THEN

    INAVIG = 0

    QS1M = DSQRT(QS1(1)**2 + QS1(2)**2 + QS1(3)**2 + QS1(4)**2)
    IF ( QS1M .EQ. 0. ) THEN

C        COMPUTE QUATERNION COMPONENTS

        SITH0 = DSIN(STHT/2.0D0)
        COTH0 = DCOS(STHT/2.0D0)
        SIPS0 = DSIN(SPSI/2.0D0)
        COPS0 = DCOS(SPSI/2.0D0)
        SIPH0 = DSIN(SPHI/2.0D0)
        COPH0 = DCOS(SPHI/2.0D0)

C        CALCULATE QUATERNIONS

        QS1(4) = COPS0*COTH0*COPH0 + SIPS0*SITH0*SIPH0
        QS1(1) = COPS0*COTH0*SIPH0 - SIPS0*SITH0*COPH0
        QS1(2) = COPS0*SITH0*COPH0 + SIPS0*COTH0*SIPH0
        QS1(3) = -COPS0*SITH0*SIPH0 + SIPS0*COTH0*COPH0

C        COMPUTE TRANSFORMATION MATRICES

C        D1 = QS1(4) * QS1(4)
C        D2 = QS1(1) * QS1(1)
C        D3 = QS1(2) * QS1(2)
C        D4 = QS1(3) * QS1(3)
C        D5 = QS1(1) * QS1(2)
C        D6 = QS1(1) * QS1(3)
C        D7 = QS1(1) * QS1(4)
C        D8 = QS1(2) * QS1(3)
C        D9 = QS1(2) * QS1(4)
C        D10 = QS1(3) * QS1(4)
C        TI2M(1) = D1 + D2 - D3 - D4
C        TI2M(2) = 2.0D0*(D5 - D10)
C        TI2M(3) = 2.0D0*(D6 + D9)
C        TI2M(4) = 2.0D0*(D5 + D10)
C        TI2M(5) = D1 - D2 + D3 - D4
C        TI2M(6) = 2.0D0*(D8 - D7)
C        TI2M(7) = 2.0D0*(D6 - D9)
C        TI2M(8) = 2.0D0*(D8 + D7)
C        TI2M(9) = D1 - D2 - D3 + D4

    ENDIF
ENDIF

DTDEL = T - TONAV
TONAV = T

C    COMPUTE CORRECTED INTEGRAL ANGLES

DTX    = 0.5D0*DELPHI
DTY    = 0.5D0*DELTHI

```

```

DTZ      = 0.5D0*DELPSI

C      INTERMEDIATE COMPUTATIONS

PP0      = DTX**2 + DTY**2 + DTZ**2
PP1      = ( PP0*DTX + DTY*DTZ0 - DTZ*DTY0 ) / 6.0D0
PP2      = ( PP0*DTY + DTZ*DTX0 - DTX*DTZ0 ) / 6.0D0
PP3      = ( PP0*DTZ + DTX*DTY0 - DTY*DTX0 ) / 6.0D0

C      SET PAST VALUES OF CORRECTED INCREMENTAL ANGLES TO PRESENT

DTX0     = DTX
DTY0     = DTY
DTZ0     = DTZ

C      UPDATE CURRENT VALUES OF CORRECTED INCREMENTAL ANGLE

DTX      = DTX - PP1
DTY      = DTY - PP2
DTZ      = DTZ - PP3

C      CALCULATE DELTA QUATERNIONS

DUM      = -0.5D0*PP0
PQ0      = DUM*QS1(4) - DTX*QS1(1) - DTY*QS1(2) - DTZ*QS1(3)
PQ1      = DTX*QS1(4) + DUM*QS1(1) + DTZ*QS1(2) - DTY*QS1(3)
PQ2      = DTY*QS1(4) - DTZ*QS1(1) + DUM*QS1(2) + DTX*QS1(3)
PQ3      = DTZ*QS1(4) + DTY*QS1(1) - DTX*QS1(2) + DUM*QS1(3)

C      UPDATE QUATERNIONS

QS1(4)   = QS1(4) + PQ0
QS1(1)   = QS1(1) + PQ1
QS1(2)   = QS1(2) + PQ2
QS1(3)   = QS1(3) + PQ3

C      NORMALIZE QUATERNIONS

DQ       = 1.0D0 + 0.5D0*(1.0D0-QS1(4)**2-QS1(1)**2
1         -QS1(2)**2-QS1(3)**2)
QS1(1)   = QS1(1)*(DQ)
QS1(2)   = QS1(2)*(DQ)
QS1(3)   = QS1(3)*(DQ)
QS1(4)   = QS1(4)*(DQ)

C      COMPUTE TRANSFORMATION MATRICES

D1 = QS1(4) * QS1(4)
D2 = QS1(1) * QS1(1)
D3 = QS1(2) * QS1(2)
D4 = QS1(3) * QS1(3)
D5 = QS1(1) * QS1(2)
D6 = QS1(1) * QS1(3)
D7 = QS1(1) * QS1(4)
D8 = QS1(2) * QS1(3)
D9 = QS1(2) * QS1(4)
D10= QS1(3) * QS1(4)
TI2M(1) = D1 ÷ D2 - D3 - D4
TI2M(2) = 2.0D0*(D5 - D10)
TI2M(3) = 2.0D0*(D6 + D9)
TI2M(4) = 2.0D0*(D5 + D10)
TI2M(5) = D1 - D2 + D3 - D4
TI2M(6) = 2.0D0*(D8 - D7)
TI2M(7) = 2.0D0*(D6 - D9)

```

```

TI2M(8) = 2.0D0*(D8 + D7)
TI2M(9) = D1 - D2 - D3 + D4

C   COMPUTE SENSED EULER ANGLES

SPHI   = DATAN2(TI2M(8),TI2M(9))
STHT   = -DASIN (TI2M(7))
SPSI   = DATAN2(TI2M(4),TI2M(1))

C   CALCULATE SENSED ANGULAR RATES AND ACCELERATIONS IN BODY FRAME

IF ( DTDEL.GT.0.0D0 ) THEN
    SP    = DELPHI/DTDEL
    SQ    = DELTHT/DTDEL
    SR    = DELPSI/DTDEL
    SUD   = DELU/DTDEL
    SVD   = DELV/DTDEL
    SWD   = DELW/DTDEL
ENDIF

C   TRANSFORM THE SENSED BODY ACCELERATIONS TO THE INERTIAL FRAME (
DOES
C   NOT INCLUDE GRAVITY )
C   NOTE AT = (SUD,SVD,SWD) * TRANSPOSE[TM2I]

AT(1) = TI2M(1)*SUD + TI2M(2)*SVD + TI2M(3)*SWD
AT(2) = TI2M(4)*SUD + TI2M(5)*SVD + TI2M(6)*SWD
AT(3) = TI2M(7)*SUD + TI2M(8)*SVD + TI2M(9)*SWD

C   TRANSFORM THE SENSED DELTA VELOCITIES INTO INERTIAL COORDINATES

DELXD  = TI2M(1)*DELU + TI2M(2)*DELV + TI2M(3)*DELW
DELYD  = TI2M(4)*DELU + TI2M(5)*DELV + TI2M(6)*DELW
DELZD  = TI2M(7)*DELU + TI2M(8)*DELV + TI2M(9)*DELW

C   DETERMINE AVERAGE GRAVITY VECTOR OVER PREVIOUS INTERVAL

IF ( DTDEL.NE.0.0D0 ) THEN
    GRAVG(1) = 0.5D0*( GRLAST(1) + GR(1) )
    GRAVG(2) = 0.5D0*( GRLAST(2) + GR(2) )
    GRAVG(3) = 0.5D0*( GRLAST(3) + GR(3) )
ELSE
    GRAVG(1) = GR(1)
    GRAVG(2) = GR(2)
    GRAVG(3) = GR(3)
ENDIF

C   SAVE GRAVITY VECTOR FOR USE ON NEXT PASS

GRLAST(1) = GR(1)
GRLAST(2) = GR(2)
GRLAST(3) = GR(3)

C   GRAVITY COMPENSATE THE SENSED DELTA VELOCITY COMPONENTS

DELXD  = DELXD + DTDEL*GRAVG(1)
DELYD  = DELYD + DTDEL*GRAVG(2)
DELZD  = DELZD + DTDEL*GRAVG(3)

C   COMPUTE SENSED MISSILE POSITION AND VELOCITY IN INERTIAL FRAME

RMIR(1) = RMIR(1) + DTDEL*(VMIR(1) + 0.5D0*DELXD)
RMIR(2) = RMIR(2) + DTDEL*(VMIR(2) + 0.5D0*DELYD)
RMIR(3) = RMIR(3) + DTDEL*(VMIR(3) + 0.5D0*DELZD)

```

```
VMIR(1) = VMIR(1) + DELXD
VMIR(2) = VMIR(2) + DELYD
VMIR(3) = VMIR(3) + DELZD

C    TRANSFORM SENSED INERTIAL VELOCITIES INTO BODY FRAME

SU    = TI2M(1)*VMIR(1) + TI2M(4)*VMIR(2) + TI2M(7)*VMIR(3)
SV    = TI2M(2)*VMIR(1) + TI2M(5)*VMIR(2) + TI2M(8)*VMIR(3)
SW    = TI2M(3)*VMIR(1) + TI2M(6)*VMIR(2) + TI2M(9)*VMIR(3)

C    TRANSFORM THE SENSED INERTIAL STATES INTO EARTH COORDINATE FRAME

RMI(1) = CIE(1)*RMIR(1) + CIE(4)*RMIR(2) + CIE(7)*RMIR(3)
RMI(2) = CIE(2)*RMIR(1) + CIE(5)*RMIR(2) + CIE(8)*RMIR(3)
RMI(3) = CIE(3)*RMIR(1) + CIE(6)*RMIR(2) + CIE(9)*RMIR(3)

VMI(1) = CIE(1)*VMIR(1) + CIE(4)*VMIR(2) + CIE(7)*VMIR(3)
VMI(2) = CIE(2)*VMIR(1) + CIE(5)*VMIR(2) + CIE(8)*VMIR(3)
VMI(3) = CIE(3)*VMIR(1) + CIE(6)*VMIR(2) + CIE(9)*VMIR(3)

RETURN
END
```

## B.2.28 Unnorm.for

```

C-----
C      SUBROUTINE spNORM(SD,MN,ISEED,RDN)
C-----
C
C      SUBROUTINE NAME :      NORM
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           GENERATES NORMALLY DISTRIBUTED RANDOM
C                           NUMBERS USING THE BOX-MULLER TRANSFORMATION
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  RAN0
C
C      INPUTS :              SD,MN
C
C      OUTPUTS :             RDN
C
C      BOTH :                ISEED
C
C      UPDATES :             D. SMITH      - CR # 082
C                           R. RHYNE      - CR # 087
C-----
C
C      IMPLICIT REAL          (A-H)
C      IMPLICIT REAL          (O-Z)
C
C      INTEGER*4 ISEED
C
C      REAL  MN
C
C      COMMON / NORCOM / GSET , ISET
C
C      DATA ONE   / 1.0e0 /
C      DATA TWO   / 2.0e0 /
C
C      IF A SPARE RANDOM NUMBER IS NOT AVAILABLE FROM THE PREVIOUS PASS
C      GENERATE TWO NEW ONES
C
C      IF ( ISET.EQ.0 ) THEN
C
C          GET TWO UNIFORM RANDOM NUMBERS WITHIN THE SQUARE EXTENDING
C          FROM -1 TO 1 IN EACH DIRECTION
C
C          1  V1      = TWO*spRAN0(ISEED) - ONE
C             V2      = TWO*spRAN0(ISEED) - ONE
C
C          SEE IF THEY ARE WITHIN THE UNIT CIRCLE . IF NOT , TRY AGAIN .
C
C          R          = V1*V1 + V2*V2
C          IF ( R.GE.ONE ) GO TO 1
C
C          PERFORM BOX-MULLER TRANSFORMATION TO GENERATE TWO GAUSSIAN
C          RANDOM NUMBERS . RETURN ONE AND SAVE THE OTHER FOR THE NEXT
C          PASS .
C
C          FAC        = SQRT ( -TWO*ALOG(R)/R )
C          GSET        = FAC*V1

```

```
      RDN      = MN + SD*FAC*V2
      ISET      = 1

C      USE GAUSSIAN RANDOM NUMBER CARRIED OVER FROM PREVIOUS PASS .

      ELSE IF ( ISET.EQ.1 ) THEN
          RDN      = MN + SD*GSET
          ISET      = 0
      ENDIF

      RETURN
      END
```

## B.4.29 Uuobtarg.for

```

C-----
C      SUBROUTINE OBTARG(T,GRTEST,RTEST,VTEST)
C-----
C
C      SUBROUTINE NAME :      OBTARG
C
C      AUTHOR(S) :          D. SISSOM
C
C      FUNCTION :           COMPUTES THE ONBOARD TARGET ESTIMATES
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             T,GRTEST
C
C      BOTH :               RTEST,VTEST
C
C      UPDATES :            B. FILL      - CR # 030
C                          T. THORNTON - CR # 045
C                          B. HILL      - CR # 055
C                          D. SMITH     - CR # 059
C                          B. HILL      - CR # 062
C                          D. SISSOM    - CR # 069
C                          D. SMITH     - CR # 070
C                          B. HILL /    - CR # 081
C                          R. RHYNE
C                          R. RHYNE     - CR # 087
C                          D. SISSOM    - CR # 091
C                          B. HILL      - CR # 093
C-----
C
C      IMPLICIT DOUBLE PRECISION      (A-H)
C      IMPLICIT DOUBLE PRECISION      (O-Z)
C
C      DOUBLE PRECISION RTEST(3)
C      DOUBLE PRECISION GRTEST(3), VTEST(3)
C      DOUBLE PRECISION GRTPST(3)      , GRTOB(3)
C      DOUBLE PRECISION TARPOS(3)      , TARVEL(3)
C
C      INTEGER          FIRST2
C
C      COMMON "ROBTRG" USED FOR MIDFLIGHT CAPABILITIES ONLY
C
C      COMMON / ROBTRG / FIRST2, TL2, GRTPST
C
C      * DATA INITIALIZATION
C      $INCLUDE('^/INCLUDE/SSDATA65.DAT')
C
C      IF ( FIRST2 .EQ. 1 ) THEN
C          FIRST2 = 0
C          TL2 = T
C
C      INITIALIZE ESTIMATED TARGET STATES
C
C      DO 45 IAXIS = 1, 3
C          RTEST(IAXIS) = TARPOS(IAXIS)
C          VTEST(IAXIS) = TARVEL(IAXIS)
45  CONTINUE
C      ELSE

```

C        INTEGRATE TARGET ACCELERATION AND VELOCITY USING AVERAGE  
C        GRAVITY VECTOR OVER LAST INTERVAL

```
      TDELT = T - TL2
      TL2   = T
      DO 2 I = 1,3
        GRTAOB(I) = 0.5D0 * ( GRTEST(I) + GRTPST(I) )
        RTEST(I)  = RTEST(I) + VTEST(I)*TDELT +
                     0.5D0*GRTAOB(I)*TDELT*TDELT
        VTEST(I)  = VTEST(I) + GRTAOB(I)*TDELT
2      CONTINUE
      ENDIF
```

C        SAVE GRAVITY VECTOR FOR USE ON NEXT PASS

```
      DO 3 I = 1,3
        GRTPST(I) = GRTEST(I)
3      CONTINUE

      RETURN
      END
```

## B.2.30 Uuran.for

```

C-----
C      REAL FUNCTION RAN(ISEED)
C-----
C
C      SUBROUTINE NAME :      RAN
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           GENERATES A UNIFORMLY DISTRIBUTED RANDOM
C                           NUMBER
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             NONE
C
C      OUTPUTS :            RAN
C
C      BOTH :               ISEED
C
C      UPDATES :            NONE
C-----
C
C      integer*4 iseed
C
C      iseed = 69069*iseed + 1
C      ran = abs(float(iseed)/2147483647.0)
C      RETURN
C      END

```

## B.2.31 Uuran0.for

```

C-----
C      DOUBLE PRECISION FUNCTION RAN0 (ISEED)
C-----
C
C      SUBROUTINE NAME :      RAN0
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           GENERATES A UNIFORMLY DISTRIBUTED RANDOM
C                           NUMBER BETWEEN 0 AND 1 USING THE SYSTEM
C                           ROUTINE RAN (ISEED) . THE BUFFER IN COMMON
C                           BLOCK RANCOM IS INITIALIZED BY CALLING
C                           ROUTINE RANIT .
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  RAN
C
C      INPUTS :             NONE
C
C      OUTPUTS :            RAN0
C
C      EOTH :              ISEED
C
C      UPDATES :            NONE
C-----
C
C      NOTE : IMPLICIT DOUBLE PRECISION IS NOT NEEDED SINCE THE OUTPUT
C             OF RAN IS SINGLE PRECISION
C
C      integer*4  iseed
C      COMMON / RANCOM /          RANSEQ(97),      RANLST
C
C      USE PREVIOUSLY SAVED RANDOM NUMBER AS BUFFER INDEX AND MAKE
C      SURE ARRAY BOUNDS ARE NOT EXCEEDED .
C
C      J          = 1 + INT ( 97.0*RANLST )
C      IF ( J.LT.1 .OR. J.GT.97 ) THEN
C          CALL OUTMES(' RANDOM NUMBER OUT OF BOUNDS IN RAN0')
C      END IF
C
C      RETRIEVE RANDOM NUMBER FROM BUFFER FOR OUTPUT AND SAVE IT FOR
C      USE AS AN INDEX ON THE NEXT PASS .
C
C      RANLST = RANSEQ(J)
C      RAN0   = DBLE ( RANLST )
C
C      LOAD A NEW RANDOM NUMBER IN THE SLOT JUST VACATED .
C
C      RANSEQ(J) = RAN ( ISEED )
C
C      RETURN
C      END

```

## B.2.32 Uurelat.for

```

C-----
      SUBROUTINE RELAT (RTIC, VTIC, X, Y, Z, XD, YD, ZD, Q, R, CIM, CMS, RRELTR,
      :                MAGRTR, VRELTR, MGRDTR, MAGLOS, LAMTRU, LAMDXX,
      :                LAMDTR, LAMSEK, LAMDSK, TGOTR, RREL, VREL)
C-----
C
C      SUBROUTINE NAME :      RELAT
C
C      AUTHOR(S) :          T. THORNTON
C
C      FUNCTION :           COMPUTES RELATIVE RANGE, RANGE RATE,
C                          TIME-TO-GO, LOS ANGLES AND RATES
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             RTIC, VTIC, X, Y, Z, XD, YD, ZD, Q, R, CIM, CMS
C
C      OUTPUTS :            RRELTR, MAGRTR, VRELTR, MGRDTR, MAGLOS, LAMTRU,
C                          LAMDXX, LAMDTR, LAMSEK, LAMDSK, TGOTR, RREL,
C                          VREL, CAZ, CEL
C
C      UPDATES :            T. THORNTON - CR # 037
C                          B. HILL      - CR # 038
C                          T. THORNTON - CR # 048
C                          D. SMITH     - CR # 059
C                          B. HILL /    - CR # 081
C                          R. RHYNE
C                          D. SISSOM    - CR # 091
C                          B. HILL      - CR # 093
C-----
      IMPLICIT DOUBLE PRECISION (A-H)
      IMPLICIT DOUBLE PRECISION (O-Z)

      REAL    CIM(9)          , XD, YD, ZD, Q, R
      DOUBLE PRECISION CMS(9) , MAGLOS
      DOUBLE PRECISION RTIC(5,3) , RRELTR(3) , URRELTR(3)
      DOUBLE PRECISION MAGRTR , VTIC(5,3) , VRELTR(3)
      DOUBLE PRECISION MAGVTR , MGRDTR , RREL(3)
      DOUBLE PRECISION VREL(3) , LAMTRU(2) , LAMDXX(2)
      DOUBLE PRECISION LAMDTR(2) , RRELS(3) , VRELS(3)
      DOUBLE PRECISION LAMSEK(2) , LAMDSK(2)

      INTEGER      SEKTYP

      * DATA INITIALIZATION
      $INCLUDE('~/INCLUDE/SSDATA50.DAT')
      $INCLUDE('~/INCLUDE/SSDATA66.DAT')
      $INCLUDE('~/INCLUDE/SSDATA21.DAT')

C      COMPUTE RELATIVE RANGE, RANGE RATE, AND TIME-TO-GO

      RRELTR(1) - RTIC(1,1) - X
      RRELTR(2) = RTIC(1,2) - Y
      RRELTR(3) = RTIC(1,3) - Z

      MAGRTR = DSQRT(RRELTR(1)**2 + RRELTR(2)**2 + RRELTR(3)**2)
      URRELTR(1) = RRELTR(1)/MAGRTR

```

```
URRELT(2) = RRELTR(2)/MAGRTR
URRELT(3) = RRELTR(3)/MAGRTR
```

```
VRELTR(1) = VTIC(1,1) - XD
VRELTR(2) = VTIC(1,2) - YD
VRELTR(3) = VTIC(1,3) - ZD
```

```
MAGVTR = DSQRT(VRELTR(1)**2 + VRELTR(2)**2 + VRELTR(3)**2)
```

```
MGRDTR = VRELTR(1)*URRELT(1) + VRELTR(2)*URRELT(2) +
          VRELTR(3)*URRELT(3)
```

```
VRDRRT = VRELTR(1)*RRELTR(1) + VRELTR(2)*RRELTR(2) +
          VRELTR(3)*RRELTR(3)
```

```
TGOTR = -VRDRRT/(MAGVTR**2)
```

# C COMPUTE LOS ANGLES AND RATES IN BODY FRAME

```
RRELM(1) = RRELTR(1)*CIM(1) + RRELTR(2)*CIM(4) + RRELTR(3)*CIM(7)
RRFLM(2) = RRELTR(1)*CIM(2) + RRELTR(2)*CIM(5) + RRELTR(3)*CIM(8)
RRELM(3) = RRELTR(1)*CIM(3) + RRELTR(2)*CIM(6) + RRELTR(3)*CIM(9)
```

```
VRELM(1) = VRELTR(1)*CIM(1) + VRELTR(2)*CIM(4) + VRELTR(3)*CIM(7)
VRELM(2) = VRELTR(1)*CIM(2) + VRELTR(2)*CIM(5) + VRELTR(3)*CIM(8)
VRELM(3) = VRELTR(1)*CIM(3) + VRELTR(2)*CIM(6) + VRELTR(3)*CIM(9)
```

```
LAMTRU(1) = DATAN2(-RRELM(3),RRELM(1))
```

```
LAMTRU(2) = DATAN2(RRELM(2),RRELM(1))
```

```
LAMDXX(1) = (RRELM(3)*VRELM(1) - RRELM(1)*VRELM(3)) /
            (RRELM(1)**2 + RRELM(3)**2)
```

```
LAMDXX(2) = (RRELM(1)*VRELM(2) - RRELM(2)*VRELM(1)) /
            (RRELM(1)**2 + RRELM(2)**2)
```

```
LAMDTR(1) = LAMDXX(1) - Q
```

```
LAMDTR(2) = LAMDXX(2) - R
```

# C COMPUTE LOS ANGLES AND RATES IN SEEKER FRAME

```
RRELS(1) = RRELM(1)*CMS(1) + RRELM(2)*CMS(4) + RRELM(3)*CMS(7)
RRELS(2) = RRELM(1)*CMS(2) + RRELM(2)*CMS(5) + RRELM(3)*CMS(8)
RRELS(3) = RRELM(1)*CMS(3) + RRELM(2)*CMS(6) + RRELM(3)*CMS(9)
```

```
VRELS(1) = VRELM(1)*CMS(1) + VRELM(2)*CMS(4) + VRELM(3)*CMS(7)
VRELS(2) = VRELM(1)*CMS(2) + VRELM(2)*CMS(5) + VRELM(3)*CMS(8)
VRELS(3) = VRELM(1)*CMS(3) + VRELM(2)*CMS(6) + VRELM(3)*CMS(9)
```

```
LAMSEK(1) = DATAN2(-RRELS(3),RRELS(1))
```

```
LAMSEK(2) = DATAN2(RRELS(2),RRELS(1))
```

```
MAGLOS = DABS(DATAN2(DSQRT(RRELS(2)**2 + RRELS(3)**2),
                     RRELS(1)))/DTR
```

```
LAMDSK(1) = (RRELS(3)*VRELS(1) - RRELS(1)*VRELS(3)) /
            (RRELS(1)**2 + RRELS(3)**2)
```

```
LAMDSK(2) = (RRELS(1)*VRELS(2) - RRELS(2)*VRELS(1)) /
            (RRELS(1)**2 + RRELS(2)**2)
```

```
RETURN
```

```
END
```



```

IF ( ZD.EQ.ONE ) THEN
  A      = WD
  TMP1   = DEXP ( - A*DT )
  TMP2   = DEXP ( - TWO*A*DT )
  TMP3   = TWO + A*DT
  TMP4   = - TWO + A*DT
  CI     = TMP1*TMP3 + TMP4
  CIL    = TWO*( ONE - TWO*A*DT*TMP1 - TMP2 )
  CILL   = TMP1*TMP4 + TMP2*TMP3
  CO     = A*DT
  COL    = - CO*TWO*TMP1
  COLL   = CO*TMP2
END IF

```

C overdamped filter

```

IF ( ZD.GT.ONE ) THEN
  TMP5   = DSQRT ( ZD**2 - ONE )
  A      = WD*TMP5
  B      = WD/TMP5
  ASQ    = A*A
  BSQ    = B*B
  EXPA   = DEXP ( - A*DT )
  EXPB   = DEXP ( - B*DT )
  TMP1   = A*DT + EXPA - ONE
  TMP2   = B*DT + EXPB - ONE
  TMP3   = ONE + A*DT
  TMP4   = ONE + B*DT
  CI     = ASQ*TMP2 - BSQ*TMP1
  CIL    = ASQ*( ONE - EXPA*TMP2 - EXPB*TMP4 )
  .      - BSQ*( ONE - EXPB*TMP1 - EXPA*TMP3 )
  CILL   = ASQ*EXPA*( EXPB*TMP4 - ONE )
  .      - BSQ*EXPB*( EXPA*TMP3 - ONE )
  CO     = A*B*DT*( A - B )
  COL    = - CO*( EXPA + EXPB )
  COLL   = CO*EXPA*EXPB
END IF

RETURN
END

```

## B.2.34 Uuresthr.for

```

C-----
C      SUBROUTINE RESTHR(T, IDIST, ANVP, DTSAMP, TOFLTM, TRATON, TPATON, TYATON,
C      DTACSA, DTACSB)
C-----
C
C      SUBROUTINE NAME :      RESTHR
C
C      AUTHOR(S) :          T. THORNTON
C
C      FUNCTION :           ATTITUDE CONTROL SYSTEM THRUSTER
C                          CROSS COUPLING LOGIC
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             T, IDIST, ANVP, DTSAMP, TOFLTM
C
C      OUTPUTS :            DTACSA, DTACSB
C
C      BOTH :               TRATON, TPATON, TYATON
C
C      UPDATES :            B. HILL      - CR # 038
C                          T. THORNTON - CR # 043
C                          T. THORNTON - CR # 044
C                          B. HILL      - CR # 051
C                          D. SMITH     - CR # 059
C                          B. HILL /    - CR # 081
C                          R. RHYNE
C                          R. RHYNE     - CR # 084
C                          B. HILL      - CR # 086
C                          B. HILL      - CR # 093
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C
C      REAL DTACSA(4)      , DTACSB(4)
C
C      * DATA INITIALIZATION
C      $INCLUDE('~/INCLUDE/SSDATA67.DAT')
C      $INCLUDE('~/INCLUDE/SSDATA03.DAT')
C      $INCLUDE('~/INCLUDE/SSDATA08.DAT')
C
C      IN DISTURBANCE MODE TURN OFF ACS THRUSTERS WITH DIVERT THRUSTERS
C
C      IF( IDIST .EQ. 1 ) THEN
C        TMP1 = TOFLTM - T
C        IF( TMP1 .LE. 0. ) THEN
C          TMP2 = 0.
C        ELSEIF( TMP1 .LT. TSMPPH ) THEN
C          TMP2 = TMP1/TSMPPH
C        ELSE
C          TMP2 = 1.
C        ENDIF
C        TPATON = TPATON*TMP2
C        TYATON = TYATON*TMP2
C        TRATON = TRATON*TMP2
C      ENDIF

```

```

C   TEST SIGNS OF PITCH, YAW, ROLL AND ATTITUDE THRUSTER PULSEWIDTHS

C   PITCH SIGN TEST

      IF( TPATON .GE. 0.0 ) THEN
        TPATP = TPATON
        TPATN = 0.0
      ELSE
        TPATP = 0.0
        TPATN = -TPATON
      ENDIF

C   YAW SIGN TEST

      IF( TYATON .GE. 0.0 ) THEN
        TYATP = TYATON
        TYATN = 0.0
      ELSE
        TYATP = 0.0
        TYATN = -TYATON
      ENDIF

C   ROLL SIGN TEST

      IF( TRATON .GE. 0.0 ) THEN
        TRATP = TRATON
        TRATN = 0.0
      ELSE
        TRATP = 0.0
        TRATN = -TRATON
      ENDIF

C   RESOLVE PITCH, YAW, AND ROLL THRUSTER PULSEWIDTHS INTO
C   INDIVIDUAL THRUSTER PULSEWIDTHS

      IF( ANVP .LT. 1.5 ) THEN
        DTACSA(1) = TPATP + TRATP
        DTACSB(1) = TPATN + TRATN
        DTACSA(2) = TYATP
        DTACSB(2) = TYATN
        DTACSA(3) = TPATN + TRATP
        DTACSB(3) = TPATP + TRATN
        DTACSA(4) = TYATN
        DTACSB(4) = TYATP
      ELSE
        DTACSA(1) = TPATP + TRATP
        DTACSB(1) = TPATN + TRATN
        DTACSA(2) = TYATP + TRATP
        DTACSB(2) = TYATN + TRATN
        DTACSA(3) = TPATN + TRATP
        DTACSB(3) = TPATP + TRATN
        DTACSA(4) = TYATN + TRATP
        DTACSB(4) = TYATP + TRATN
      ENDIF

      DO 50 I=1,4

C   ENFORCE THRUSTER PAIR DEADBANDS

      IF( ABS( DTACSA(I) - DTACSB(I) ) .LT. ACSDB ) THEN
        DTACSA(I) = 0.0
        DTACSB(I) = 0.0
      ENDIF

```

```

C      ENFORCE MINIMUM COMMAND ON TIME

      IF((DTACSA(I) .LT. TCMINA .AND. DTACSA(I) .GT. 0.) .OR.
        (DTACSB(I) .LT. TCMINA .AND. DTACSB(I) .GT. 0.)) THEN
        DTACSA(I) = DTACSA(I) + TCMINA
        DTACSB(I) = DTACSB(I) + TCMINA
      ENDIF
      IF( DTACSA(I) .GT. DTSAMP ) DTACSA(I) = DTSAMP
      IF( DTACSB(I) .GT. DTSAMP ) DTACSB(I) = DTSAMP

50  CONTINUE

      RETURN
      END

```

## B.2.35 Uurotmx.for

```

C-----
C      SUBROUTINE ROTMX(X,I,XM)
C-----
C
C      SUBROUTINE NAME :      ROTMX
C
C      AUTHOR(S) :          J. SHEEHAN
C
C      FUNCTION :           GENERATES A DIRECTION COSINE MATRIX
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             X,I
C
C      OUTPUTS :            XM
C
C      UPDATES :            D. SMITH - CR # 59
C-----
C
C      IMPLICIT DOUBLE PRECISION (A-H)
C      IMPLICIT DOUBLE PRECISION (O-Z)
C      DOUBLE PRECISION XM(3,3)
C
C      INTEGER IIT(3), IIIT(3)
C      DATA IIT / 2 , 3 , 1 /
C      . , IIIT/ 3 , 1 , 2 /
C
C      SX = DSIN(X)
C      CX = DCOS(X)
C      II = IIT(I)
C      III = IIIT(I)
C
C      XM(I,I) = 1.0
C      XM(I,II) = 0.0
C      XM(I,III) = 0.0
C      XM(II,I) = 0.0
C      XM(III,I) = 0.0
C
C      XM(II,II) = CX
C      XM(III,III) = CX
C      XM(II,III) = SX
C      XM(III,II) = -SX
C
C      RETURN
C      END

```

## B.2.36 Uuseeker.for

```

C-----
      SUBROUTINE SEEKER(T,ACQD,LAMSEK,MAGRTR,SKSEED,FRMRAT,FRMCNT,
      .               SAMRAT,TRACK,TERM,SNR,LAMM)
C-----
C
C      SUBROUTINE NAME :      SEEKER
C
C      AUTHOR(S) :          M. K. DOUBLEDAY, D. C. FOREMAN
C
C      FUNCTION :           SEEKER MODEL
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NORM, TABLE
C
C      INPUTS :             T,ACQD,LAMSEK,MAGRTR
C
C      OUTPUTS :            SAMRAT,TRACK,TERM,SNR,LAMM
C
C      BOTH :               SKSEED,FRMRAT,FRMCNT
C
C      UPDATES :            T. THORNTON - CR # 014
C                          B. HILL      - CR # 020
C                          D. SMITH     - CR # 027
C                          B. HILL      - CR # 030
C                          B. HILL      - CR # 038
C                          T. THORNTON - CR # 043
C                          T. THORNTON - CR # 044
C                          T. THORNTON - CR # 048
C                          D. SISSOM    - CR # 053
C                          D. SMITH     - CR # 059
C                          D. SMITH     - CR # 064
C                          D. SISSOM    - CR # 069
C                          D. SMITH     - CR # 074
C                          D. SMITH     - CR # 080
C                          B. HILL /    - CR # 081
C                          R. RHYNE     - CR # 082
C                          D. SMITH     - CR # 084
C                          R. RHYNE     - CR # 087
C                          R. RHYNE     - CR # 088
C                          B. HILL      - CR # 093
C-----
C
      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)
C
      CHARACTER*128 MESSAGE
      REAL  ACQRNG(4,4)  , LAMB(2)      , LAMFOV
      REAL  LAMM(2)      , LAMNEA(2)    , LAMSEK(2)
      REAL  LAMSK(2)     , MAGRTR
      REAL  NEA          , RATE(6)      , SEKNOS(24)
      REAL  SEKTIM(24)   , TRGSIG(4)
C
      INTEGER ACQD      , BCKGRD      , FRMCNT
      INTEGER SEKTYP
      INTEGER*4 SKSEED
      INTEGER  TERM      , TRACK
C
      LOCAL COMMON USED FOR CONSTANTS AND INITIALIZATION FLAG

```

```

      SAVE                ISEKR,   IFOV

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSDATA47.DAT')
$INCLUDE('~/INCLUDE/SSDATA48.DAT')
$INCLUDE('~/INCLUDE/SSDATA50.DAT')
$INCLUDE('~/INCLUDE/SSDATA55.DAT')
$INCLUDE('~/INCLUDE/SSDATA61.DAT')
$INCLUDE('~/INCLUDE/SSDATA68.DAT')
$INCLUDE('~/INCLUDE/SSDATA10.DAT')
$INCLUDE('~/INCLUDE/SSDATA11.DAT')

      DATA ISEKR / 1 /
      DATA IT / 1 /

      IF (ISEKR.EQ.1) THEN

        ISEKR = 0

        IF ( SEKTYP.EQ.2 ) THEN
          TSIG   = TRGSIG(ITRGSG)
          TSGACQ = TSIG
          RAQREF = ACQRNG(BCKGRD,ITRGSG)
          RNGAQ  = SQRT((TSGACQ/TSIG)*(6.0/SNRACQ)*
                        (SQRT(1./FRMRAT)))*RAQREF
        ENDIF

      ENDIF

C    TEST FOR FIELD-OF-VIEW LIMIT

      IF ( SEKTYP.EQ.2 .AND. SNR.GE.SNRACQ ) THEN
        FOVCHK = FOVLIM
      ELSE IF ( ACQD.EQ.1 .AND. SEKTYP.NE.2 ) THEN
        FOVCHK = FOV
      ELSE
        FOVCHK = 1000.
      ENDIF
      LAMFOV = AMAX1( LAMSEK(1) , LAMSEK(2) )
      IF ( LAMFOV.GE.FOVCHK .AND. IFOV.EQ.0 ) THEN
        CALL OUTMES(' TRUE LOS ANGLE EXCEEDS FIELD-OF-VIEW LIMIT')
        IFOV = 1
      ELSE IF ( LAMFOV.LE.FOVCHK .AND. IFOV.EQ.1 ) THEN
        CALL OUTMES(' TARGET REACQUIRED')
        IFOV = 0
      ENDIF

C    DETERMINE SEEKER SAMPLE RATE FOR SEEKER TYPES 0 AND 1

      IF ( SEKTYP.EQ.0 .OR. SEKTYP.EQ.1 ) THEN
        IF ( MAGRTR .LE. RNGTRM ) THEN
          SAMRAT = SAMTRM
          IF (TERM.EQ.0) TERM = 1
        ELSE IF ( MAGRTR .LE. RNGTRK ) THEN
          SAMRAT = SAMTRK
          IF (TRACK.EQ.0) TRACK = 1
        ELSE
          SAMRAT = SAMACQ
        ENDIF
      ENDIF

C    PERFECT SEEKER MODEL

```

```

IF ( SEKTYP.EQ.0 ) THEN
  LAMM(1) = LAMSEK(1)
  LAMM(2) = LAMSEK(2)
  FRMRAT = 1.0/SAMRAT
ENDIF

C   SEEKER MODEL DEPENDENT ON RANGE, FRAME, AND ENVIRONMENT

IF ( SEKTYP.EQ.2 ) THEN

C   DETERMINE THE SIGNAL-TO-NOISE RATIO

  IF ( MAGRTR.LE.RFINAL ) THEN
    SNR = (RAQREF**2/RFINAL**2) * (TSGACQ/TSIG) *
      (SQRT(1.0/FRMRAT)) * SNRACQ
  ELSE
    SNR = (RAQREF**2/MAGRTR**2) * (TSGACQ/TSIG) *
      (SQRT(1.0/FRMRAT)) * SNRACQ
  ENDIF

C   CALCULATE THE NOISE EQUIVALENT ANGLE (RADIAN) FROM THE
C   EFFECTIVE SNR

  NEA = (32.56*SNR**(-0.29912)) * 1.0E-6

C   MULTIPLY NOISE EQUIVALENT ANGLE BY NORMALLY DISTRIBUTED RANDOM
C   VARIABLE WITH A MEAN OF ZERO AND A STANDARD DEVIATION OF ONE

  CALL spNORM(1.0e0,0.0,SKSEED,RANA)
  CALL spNORM(1.0e0,0.0,SKSEED,RANB)

  LAMNEA(1) = NEA*RANA
  LAMNEA(2) = NEA*RANB

C   DETERMINE MEASURED LOS ANGLE (RADIAN)

  LAMB(1) = LAMSEK(1) + LAMNEA(1)
  LAMB(2) = LAMSEK(2) + LAMNEA(2)

C   QUANTIZE THE MEASURED LOS ANGLE (RADIAN)

  IF ( QNTZP.GT.0.0 ) THEN
    LAMM(1) = (AINT(LAMB(1)/QNTZP + 0.5)) * QNTZP
    LAMM(2) = (AINT(LAMB(2)/QNTZP + 0.5)) * QNTZP
  ELSE
    LAMM(1) = LAMB(1)
    LAMM(2) = LAMB(2)
  ENDIF

C   DETERMINE IF A FRAME RATE SWITCH IS REQUIRED

  IF ( MAGRTR.LE.RFINAL ) THEN
    FRMR = ((6.0/SNRMIN) * (TSGACQ/TSIG) * (RAQREF**2/RFINAL**2)) **2
  ELSE
    FRMR = ((6.0/SNRMIN) * (TSGACQ/TSIG) * (RAQREF**2/MAGRTR**2)) **2
  ENDIF

  IF ( FRMR.GE.RATE(FRMCNT) .AND. FRMCNT.LT.7 ) THEN
    FRMRAT = RATE(FRMCNT)
    FRMCNT = FRMCNT + 1
    WRITE(MESSAGE,101) T,FRMRAT
    CALL OUTMES(MESSAGE)
101  FORMAT(1X,E16.9,' FRAME RATE CHANGE:  FRMRAT = ',E16.9)
  ENDIF

```

ENDIF

RETURN

END

## B.2.37 Uuspint.for

```

C-----
C      SUBROUTINE spINTEG ( X , XDOT , T , I )
C-----
C
C      SUBROUTINE NAME :      INTEG
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           Perform simple trapezoidal integration of
C                          XDOT to yield X. DTD is the time since
C                          the last integration and I is the array
C                          index where X is stored
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             XDOT,T,I
C
C      OUTPUTS :            X
C
C      UPDATES :            D. SISSOM   - CR # 58
C                          D. SMITH     - CR # 59
C-----
C
C      COMMON/STORAG/      XINT,          TINT,          XDOTL
C      REAL  XINT(50),      TINT(50),      XDOTL(50)
C      REAL  DT,            DTMP,          X
C      REAL  XDOT,          T
C
C      DT      = T - TINT(I)
C
C      XINT(I) = XINT(I) + 0.5*DT*(XDOT+XDOTL(I))
C      X      = XINT(I)
C
C      TINT(I) = T
C      XDOTL(I) = XDOT
C
C      TEMPORARY CODE TO NORMALIZE QUATERNION AFTER 4TH COMPONENT IS
C      REVISED
C
C      IF ( I.EQ.18 ) THEN
C          DTMP = SQRT ( XINT(15)**2 + XINT(16)**2 + XINT(17)**2 +
C                      XINT(18)**2 )
C          XINT(15) = XINT(15) / DTMP
C          XINT(16) = XINT(16) / DTMP
C          XINT(17) = XINT(17) / DTMP
C          XINT(18) = XINT(18) / DTMP
C      END IF
C
C      RETURN
C      END

```

## B.2.38 Uuspinti.for

```

C-----
C      SUBROUTINE spINTEGI ( X , XDOT , T , I )
C-----
C
C      SUBROUTINE NAME :      INTEG1
C
C      AUTHOR(S) :           D. F. SMITH
C
C      FUNCTION :            Initialize integral of X which is stored
C                           in position I of the integral array
C
C      CALLED FROM :         MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :              X,XDOT,T,I
C
C      OUTPUTS :             NONE
C
C      UPDATES :             D. SISSOM   - CR # 58
C                           D. SMITH    - CR # 59
C-----
C
COMMON/STORAG/      XINT,      TINT,      XDOTL
REAL  XINT(50),      TINT(50),      XDOTL(50)
REAL  X,              T,              XDOT

XINT(I)  = X
XDOTL(I) = XDOT
TINT(I)  = T

RETURN
END

```

## B.2.39 Uuspmmk.for

```

C-----
C      SUBROUTINE SPMMK (A, NA, B, NB, C, NC, RM)
C-----
C
C      SUBROUTINE NAME :      MMK
C
C      AUTHOR(S) :          J. SHEEHAN
C
C      FUNCTION :           GENERATES A DIRECTION COSINE MATRIX
C                          BY ROTATING IN ORDER:
C                          1) ANGLE C ABOUT THE NC AXIS
C                          2) ANGLE B ABOUT THE NB AXIS
C                          3) ANGLE A ABOUT THE NA AXIS
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  ROTMX, MMLXY
C
C      INPUTS :             A, NA, B, NB, C, NC
C
C      OUTPUTS :            RM
C
C      UPDATES :            D. SMITH      - CR # 59
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C
C      DIMENSION AM(3,3), BM(3,3), CM(3,3), RM(3,3), T(9)
C
C      CALL SPROTMX(A, NA, AM)
C      CALL SPROTMX(B, NB, BM)
C      CALL SPROTMX(C, NC, CM)
C
C      CALL SPMMLXY(BM, CM, T)
C      CALL SPMMLXY(AM, T, RM)
C
C      RETURN
C      END

```

## B.2.40 Uuspmmlx.for

```

C-----
C      SUBROUTINE SPMLXY(X,Y,Z)
C-----
C
C      SUBROUTINE NAME :      MMLXY
C
C      AUTHOR(S) :          J. SHEEHAN
C
C      FUNCTION :           MULTIPLY TWO 3X3 MATRICES
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             X, Y
C
C      OUTPUTS :            Z
C
C      UPDATES :            D. SMITH    - CR # 59
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C
C      DIMENSION X(3,3), Y(3,3), Z(3,3)
C
C      Z(I,J) = X(I,1)*Y(1,J) + X(I,2)*Y(2,J) + X(I,3)*Y(3,J)
C
C      Z(1,1) = X(1,1)*Y(1,1) + X(1,2)*Y(2,1) + X(1,3)*Y(3,1)
C      Z(2,1) = X(2,1)*Y(1,1) + X(2,2)*Y(2,1) + X(2,3)*Y(3,1)
C      Z(3,1) = X(3,1)*Y(1,1) + X(3,2)*Y(2,1) + X(3,3)*Y(3,1)
C      Z(1,2) = X(1,1)*Y(1,2) + X(1,2)*Y(2,2) + X(1,3)*Y(3,2)
C      Z(2,2) = X(2,1)*Y(1,2) + X(2,2)*Y(2,2) + X(2,3)*Y(3,2)
C      Z(3,2) = X(3,1)*Y(1,2) + X(3,2)*Y(2,2) + X(3,3)*Y(3,2)
C      Z(1,3) = X(1,1)*Y(1,3) + X(1,2)*Y(2,3) + X(1,3)*Y(3,3)
C      Z(2,3) = X(2,1)*Y(1,3) + X(2,2)*Y(2,3) + X(2,3)*Y(3,3)
C      Z(3,3) = X(3,1)*Y(1,3) + X(3,2)*Y(2,3) + X(3,3)*Y(3,3)
C
C      RETURN
C      END

```

## B.2.41 Uuspran.for

```

C-----
C      REAL FUNCTION RAN(ISEED)
C-----
C
C      SUBROUTINE NAME :      RAN
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           GENERATES A UNIFORMLY DISTRIBUTED RANDOM
C                           NUMBER
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             NONE
C
C      OUTPUTS :            RAN
C
C      BOTH :               ISEED
C
C      UPDATES :            NONE
C-----
C
C      integer*4 iseed
C
C      iseed = 69069*iseed + 1
C      ran = abs(float(iseed)/2147483647.0)
C      RETURN
C      END

```

## B.2.42 Uuspran0.for

```

C-----
C      REAL FUNCTION spran0(iseed)
C-----
C
C      SUBROUTINE NAME :      ran0
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           GENERATES A UNIFORMLY DISTRIBUTED RANDOM
C                           NUMBER BETWEEN 0 AND 1 USING THE SYSTEM
C                           ROUTINE ran(iseed) . THE BUFFER IN COMMON
C                           BLOCK rancom IS INITIALIZED BY CALLING
C                           ROUTINE ranit .
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  ran
C
C      INPUTS :             none
C
C      OUTPUTS :            ran0
C
C      BOTH :              iseed
C
C      UPDATES :           none
C-----
C
C      integer*4 iseed
C      COMMON / rancom /      ranseq(97),      ranlst
C
C      USE PREVIOUSLY SAVED RANDOM NUMBER AS BUFFER INDEX AND MAKE
C      SURE ARRAY BOUNDS ARE NOT EXCEEDED .
C
C      J      = 1 + INT ( 97.0*ranlst )
C      IF ( J.LT.1 .OR. J.GT.97 ) THEN
C          CALL outmes(' RANDOM NUMBER OUT OF BOUNDS IN ran0')
C      END IF
C
C      RETRIEVE RANDOM NUMBER FROM BUFFER FOR OUTPUT AND SAVE IT FOR
C      USE AS AN INDEX ON THE NEXT PASS .
C
C      ranlst = ranseq(J)
C      spran0 = ranlst
C
C      LOAD A NEW RANDOM NUMBER IN THE SLOT JUST VACATED .
C
C      ranseq(J, = ran ( iseed )
C
C      RETURN
C      END

```

## B.2.43 Uusprotm.for

```

C-----
C      SUBROUTINE SPROTMX(X,I,XM)
C-----
C
C      SUBROUTINE NAME :      ROTMX
C
C      AUTHOR(S) :          J. SHEEHAN
C
C      FUNCTION :           GENERATES A DIRECTION COSINE MATRIX
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             X,I
C
C      OUTPUTS :            XM
C
C      UPDATES :            D. SMITH - CR # 59
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C      REAL XM(3,3)
C
C      INTEGER IIT(3), IIIT(3)
C      DATA IIT / 2 , 3 , 1 /
C      . , IIIT/ 3 , 1 , 2 /
C
C      SX = SIN(X)
C      CX = COS(X)
C      II = IIT(I)
C      III = IIIT(I)
C
C      XM(I,I) = 1.0
C      XM(I,II) = 0.0
C      XM(I,III) = 0.0
C      XM(II,I) = 0.0
C      XM(III,I) = 0.0
C
C      XM(II,II) = CX
C      XM(III,III) = CX
C      XM(II,III) = SX
C      XM(III,II) = -SX
C
C      RETURN
C      END

```

## B.2.44 Uutable.for

```

C-----
C      SUBROUTINE spTABLE(XTAB,YTAB,X,Y,N,I)
C-----
C
C      SUBROUTINE NAME :      TABLE
C
C      AUTHOR(S) :          D. SMITH
C
C      FUNCTION :           PERFORMS TABLE LOOKUP VIA EITHER INDEXED
C                           SEARCH OR BINARY SEARCH AND LINEARLY
C                           INTERPOLATES
C
C      CALLED FROM :        UTILITY SUBROUTINE
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :              XTAB,YTAB,X,N
C
C      OUTPUTS :             Y
C
C      BOTH :                I
C
C      UPDATES :             D. SMITH      - CR # 27
C                           B. HILL       - CR # 38
C                           B. HILL       - CR # 46
C                           D. SMITH      - CR # 59
C-----
C
C      IMPLICIT real (A-H)
C      IMPLICIT real (O-Z)
C      INTEGER N,I
C      real XTAB(N),YTAB(N)
C
C      IF ( I.GE.1 .AND. I.LE.N ) THEN
C        IF ( X.LE.XTAB(1) ) THEN
C          Y      = YTAB(1)
C          I      = 1
C        ELSE IF ( X.GE.XTAB(N) ) THEN
C          Y      = YTAB(N)
C          I      = N
C        ELSE IF ( X.GE.XTAB(I) ) THEN
C          DO 10 K = I , N-1
C            IF ( X.LT.XTAB(K+1) ) GO TO 20
C10       CONTINUE
C20       FRACT = ( X - XTAB(K) ) / ( XTAB(K+1) - XTAB(K) )
C          Y      = YTAB(K) + FRACT * ( YTAB(K+1) - YTAB(K) )
C          I      = K
C        ELSE IF ( X.LT.XTAB(I) ) THEN
C          DO 30 K = I-1 , 1 , -1
C            IF ( X.GE.XTAB(K) ) GO TO 40
C30       CONTINUE
C40       FRACT = ( X - XTAB(K) ) / ( XTAB(K+1) - XTAB(K) )
C          Y      = YTAB(K) + FRACT * ( YTAB(K+1) - YTAB(K) )
C          I      = K
C        END IF
C
C      PERFORM BINARY SEARCH IF POINTER IS ZERO OR OUT OF BOUNDS
C
C      ELSE IF ( I.LT.1 .OR. I.GT.N ) THEN
C        IF ( X.GT.XTAB(1) .AND. X.LT.XTAB(N) ) THEN

```

```

      K      = 1
      L      = N
      DO 50 I = K , L
        IF ( L.EQ.K+1 ) GO TO 60
        M      = ( K + L ) / 2
        IF ( X.LT.XTAB(M) ) THEN
          L      = M
        ELSE
          K      = M
        END IF
50      CONTINUE
60      FRACT  = ( X - XTAB(K) ) / ( XTAB(L) - XTAB(K) )
      Y      = YTAB(K) + FRACT * ( YTAB(L) - YTAB(K) )
      I      = K
      ELSE IF ( X.LE.XTAB(1) ) THEN
        Y      = YTAB(1)
        I      = 1
      ELSE IF ( X.GE.XTAB(N) ) THEN
        Y      = YTAB(N)
        I      = N
      END IF
END IF
C
RETURN
END

```

## B.2.45 Uuvcth2.for

```

C-----
C      SUBROUTINE VCSTHR2 (T,FLTC,FLTCP,FLTCY,TBURNM,TOFFLT,
C                          TIMONV,IVTAB)
C-----
C
C      SUBROUTINE NAME :      VCSTHR2
C
C      AUTHOR(S) :          B. HILL
C
C      FUNCTION :           RESOLVES THE VCS THRUSTER BURN TIMES INTO
C                          THEIR APPROPRIATE FORCES AND MOMENTS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      INPUTS :             T,TOFFLT,TIMONV
C
C      OUTPUTS :            FLCPT,FLTCY
C
C      BOTH :               TBURNM,FLTC,IVTAB
C
C      UPDATES :            D. SISSOM - CR # 017
C                          B. HILL - CR # 030
C                          D. SISSOM - CR # 032
C                          B. HILL - CR # 038
C                          T. THORNTON - CR # 043
C                          B. HILL - CR # 051
C                          B. HILL - CR # 057
C                          D. SMITH - CR # 059
C                          D. SISSOM - CR # 069
C                          D. SMITH - CR # 074
C                          D. SMITH - CR # 076
C                          D. SMITH - CR # 080
C                          B. HILL / - CR # 081
C                          R. RHYNE
C                          D. SMITH - CR # 082
C                          R. RHYNE - CR # 084
C                          B. HILL - CR # 086
C                          R. RHYNE - CR # 087
C                          B. HILL - CR # 089
C                          B. HILL - CR # 093
C-----
C
C      IMPLICIT REAL          (A-H)
C      IMPLICIT REAL          (O-Z)
C
C      REAL FLTC(4)           , TOFFLT(4)
C
C      * DATA INITIALIZATION
C      $INCLUDE('^/INCLUDE/SSDATA09.DAT')
C
C      IF (IVTAB.EQ. 1) THEN
C
C          ivtab = 0
C
C          IF ( TBURNM .GE. TCMINV ) THEN
C
C              TBURNM = 0.0
C
C          ENDIF
C      ENDIF

```

```

DO 30 I=1,4
  IF ( (TOFFLT(I)-T).LE.0.0 )  FLTC(I) = 0.0
30 CONTINUE

  IF ( FLTC(1).EQ.0.0 .AND. FLTC(3).EQ.0.0 .AND.
.    (TIMONV).LE.T ) FLTCY = 0.0
  IF ( FLTC(2).EQ.0.0 .AND. FLTC(4).EQ.0.0 .AND.
.    (TIMONV).LE.T ) FLTCP = 0.0

END

```

## B.2.46 Uuvcsthr.for

```

C-----
C      SUBROUTINE VCSTHR(T,CG,TBURNM,IVCS,TOFFLT,
C      .                TIMONV,DTOFFV,TVTAB,FOFF1,FOFF2,IVTAB,TBRK,
C      .                FXVCS,FYVCS,FZVCS,MXVCS,MYVCS,MZVCS,MDOTV)
C-----
C
C      SUBROUTINE NAME :      VCSTHR
C
C      AUTHOR(S) :          B. HILL
C
C      FUNCTION :           RESOLVES THE VCS THRUSTER BURN TIMES INTO
C                           THEIR APPROPRIATE FORCES AND MOMENTS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  TABLE
C
C      INPUTS :             T,CG,TBURNM,IVCS,
C                           TOFFLT,TIMONV,DTOFFV,TVTAB,FOFF1,FOFF2,IVTAB
C
C      OUTPUTS :            FXVCS,FYVCS,FZVCS,MXVCS,MYVCS,
C                           MZVCS,MDOTV
C
C      BOTH :              TBRK
C
C      UPDATES :            D. SISSOM - CR # 017
C                           B. HILL - CR # 030
C                           D. SISSOM - CR # 032
C                           B. HILL - CR # 038
C                           T. THORNTON - CR # 043
C                           B. HILL - CR # 051
C                           B. HILL - CR # 057
C                           D. SMITH - CR # 059
C                           D. SISSOM - CR # 069
C                           D. SMITH - CR # 074
C                           D. SMITH - CR # 076
C                           D. SMITH - CR # 080
C                           B. HILL / - CR # 081
C                           R. RHYNE
C                           D. SMITH - CR # 082
C                           R. RHYNE - CR # 084
C                           B. HILL - CR # 086
C                           R. RHYNE - CR # 087
C                           B. HILL - CR # 089
C                           B. HILL - CR # 093
C-----

```

```

IMPLICIT REAL      (A-H)
IMPLICIT REAL      (O-Z)

REAL  ATHRV(4)      , CG(3)          , DTOFFV(4)
REAL  F(3)          , F0(3)
REAL  FOFF1(4)      , FOFF2(4)      , ISPVCS
REAL  M(3)          , MDOTV         , MXVCS
REAL  MYVCS         , MZVCS         , THVCS(6,4)
REAL  TMVCS(6,4)    , TOFFLT(4)
REAL  VCSDIR(3,4)   , VCSLOC(3,4)   , VCSMA(9,4)
REAL  VOFF1(4)      , VOFF2(4)      , XMOT(3)

INTEGER          INDX(4)

```

```

      INTEGER          LENVCS(4)

C      LOCAL COMMON USED FOR CONSTANTS, LOCAL VARIABLES AND
      ^      INITIALIZATION FLAG

      SAVE            IVCSTH , VCSMA

C      COMMON "RVCSTR" USED FOR MIDFLIGHT CAPABILITIES ONLY

      COMMON / RVCSTR / TREFLV , TLSTV , TMVCS , THVCS , LENVCS

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSDATA70.DAT')
$INCLUDE('~/INCLUDE/SSDATA09.DAT')

      DATA IVCSTH / 1 /

      IF (IVCSTH.EQ.1) THEN

          IVCSTH = 0

C          VCS MISALIGNMENT DIRECTIONS
C          VOFF1 = CONE ANGLE OFF NORMAL
C          VOFF2 = POLAR ANGLE

          DO 10 I = 1,4
              VOFF1(I) = FVOFF1(I)
              VOFF2(I) = FVOFF2(I)
10          CONTINUE

C          VCS THRUSTER MISALIGNMENT MATRIX

          DO 200 I = 1 , 4
              CVOFF1 = COS(VOFF1(I))
              SVOFF1 = SIN(VOFF1(I))
              CVOFF2 = COS(VOFF2(I))
              SVOFF2 = SIN(VOFF2(I))
              VCSMA(1,I) = CVOFF1
              VCSMA(2,I) = SVOFF1*CVOFF2
              VCSMA(3,I) = SVOFF1*SVOFF2
              VCSMA(4,I) = SVOFF1*SVOFF2
              VCSMA(5,I) = CVOFF1
              VCSMA(6,I) = SVOFF1*CVOFF2
              VCSMA(7,I) = SVOFF1*CVOFF2
              VCSMA(8,I) = SVOFF1*SVOFF2
              VCSMA(9,I) = CVOFF1
200          CONTINUE
          ENDDIF

C      RESET THE FORCE AND MOMENT TO ZERO

      FXVCS = 0.0
      FYVCS = 0.0
      FZVCS = 0.0
      MXVCS = 0.0
      MYVCS = 0.0
      MZVCS = 0.0
      MDOTV = 0.0

      IF (IVTAB .EQ. 1) THEN

          IF ( TBURNM .GE. TCMINV ) THEN

C              DEFINE VCS THRUST PROFILE

```

```

      TMVCS(1,IVCS) = TVTAB
      THVCS(1,IVCS) = 0.0
      TMVCS(2,IVCS) = TIMONV
      THVCS(2,IVCS) = 0.0
      TMVCS(3,IVCS) = TIMONV + TRUPV
      THVCS(3,IVCS) = FLATM
      TMVCS(4,IVCS) = TIMONV + TBURNM
      THVCS(4,IVCS) = FLATM
      TMVCS(5,IVCS) = TMVCS(4,IVCS) + TRDNV
      THVCS(5,IVCS) = 0.0
      TMVCS(6,IVCS) = 999.0
      THVCS(6,IVCS) = 0.0
      LENVCS(IVCS) = 6

ENDIF

C      GENERATE THRUSTER RESPONSE CURVE

DO 15 I=1,4
  IF ( DTOFFV(I).GT.0.0 ) THEN
    TMVCS(1,I) = TVTAB
    THVCS(1,I) = 0.0
    TMVCS(2,I) = TVTAB + TLAGV
    THVCS(2,I) = 0.0
    TMVCS(3,I) = TMVCS(2,I) + TRUPV
    THVCS(3,I) = FLATM
    TMVCS(4,I) = TOFFLT(I)
    THVCS(4,I) = FLATM
    TMVCS(5,I) = TMVCS(4,I) + TRDNV
    THVCS(5,I) = 0.0
    TMVCS(6,I) = 999.0
    THVCS(6,I) = 0.0
    LENVCS(I) = 6
  ENDIF
15 CONTINUE
ENDIF

C      SET TABLE LOOKUP REFERENCE TIME

TREF = T

DO 20 I=1,4

C      COMPUTE INSTANTANEOUS THRUST LEVEL VIA TABLE LOOKUP IF VCS
C      CYCLE IS SCHEDULED FOR THIS THRUSTER . ALSO EXTRAPOLATE TIME
C      OF NEXT VCS TABLE LOOKUP INDEX TRANSITION .

  IF ( TMVCS(1,I).GT.0.0 ) THEN
    CALL spTABLE(TMVCS(1,I),THVCS(1,I),TREF,ATHRV(I),
                LENVCS(I),INDX(I))
  ELSE
    ATHRV(I) = 0.0
    INDX(I) = 0
  ENDIF

C      CALCULATE FORCES AND MOMENTS DUE TO THE VCS THRUSTERS :
C      F(I) IS THE FORCE ALONG THE Ith AXIS.
C      XMOT(I) IS THE EFFECTIVE MOMENT ARM.
C      FORCES ARE ADJUSTED FOR MISALIGNMENT EFFECTS.
C      THE MOMENT GENERATED IS ( F x XMOT).

DO 25 J=1,3
  F0(J) = VCSDIR(J,I)*ATHRV(I)

```

```

      XMOT(J) = CG(J) - VCSLOC(J,I)
25  CONTINUE

      F(1)  = VCSMA(1,I)*F0(1) +VCSMA(4,I)*F0(2) +VCSMA(7,I)*F0(3)
      F(2)  = VCSMA(2,I)*F0(1) +VCSMA(5,I)*F0(2) +VCSMA(8,I)*F0(3)
      F(3)  = VCSMA(3,I)*F0(1) +VCSMA(6,I)*F0(2) +VCSMA(9,I)*F0(3)

      M(1)  = F(2)*XMOT(3) - F(3)*XMOT(2)
      M(2)  = F(3)*XMOT(1) - F(1)*XMOT(3)
      M(3)  = F(1)*XMOT(2) - F(2)*XMOT(1)

      FXVCS = FXVCS + F(1)
      FYVCS = FYVCS + F(2)
      FZVCS = FZVCS + F(3)
      MXVCS = MXVCS + M(1)
      MYVCS = MYVCS + M(2)
      MZVCS = MZVCS + M(3)

      MDOTV = MDOTV + ATRV(I)/ISPVCS
20  CONTINUE

      END

```

**B.3 Mainlines (C)****B.3.1 Uup00.c**

```

/* uup00.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lF77 -lI77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    real xint[50], tint[50], xdot1[50];
} storag_;

#define storag_1 storag_

struct {
    real tlstm, mass1;
} rmass_;

#define rmass_1 rmass_

struct {
    real xyzlch[3];
} rmissl_;

#define rmissl_1 rmissl_

/* Table of constant values */

static shortint cs__1 = 1;
static shortint cs__5 = 5;
static shortint cs__12 = 12;
static shortint cs__13 = 13;
static shortint cs__14 = 14;
static shortint cs__15 = 15;
static shortint cs__16 = 16;
static shortint cs__17 = 17;
static shortint cs__18 = 18;
static integer c__1 = 1;
static real c_b14 = (float)0.;
static shortint cs__2 = 2;
static shortint cs__3 = 3;

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    /* Format strings */
    static char fmt_155[] = "(1x,e16.9,\002 DROP NOSE FAIRING AND BOOST
ADAP\
TER\002)";

    /* System generated locals */

```

```

real r_1;

/* Builtin functions */
integer s_wsfi(), do_fio(), e_wsfi();

/* Local variables */
static real deltt;
static shortint igit;
static real eisp, mass, mdot, quat[4];
extern /* Subroutine */ int spintegi_();
static real tmsudriv, tmsustep, p, q, r, t, u, v, w, wbanf, mdota,
mxacs,
    myacs, mzacs, quatd[4];
static shortint imass, idrop;
static real dteps;
static shortint iexit;
static real tdrop, mdotv, xmtof, tstep, mxvcs, myvcs, mvcs, tlmsu;
extern /* Subroutine */ int receive_real_32bit_();
static real wprop, pd, qd, rd, xd, yd, zd;
static shortint nclear;
static real mx, my, mz, tfinal, weight;
extern /* Subroutine */ int send_real_32bit_(), massspr_();
static real impuls, wdotti;
extern /* Subroutine */ int outmes_();
static real wdotkv, wdottp;
extern /* Subroutine */ int missil2_();
static real cim[9], phi;
extern /* Subroutine */ int cw87_();
static real psi, tht;
static char message[128];
static real wkv, ix, iyy, izz;
extern /* Subroutine */ int spinteg_(), receive_signed_16bit_();

/* Fortran I/O blocks */
static icilist io_55 = { 0, message, 0, fmt_155, 128, 1 };

/*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
*/

/*      OUTPUTS */
/*      NAMELIST INPUTS */
/* DATA INITIALIZATION */
/* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */

```

```

/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE('SSp00.DAT') */
/* INITIALIZE 80x87 */
  cw87_();
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*           Execution of all events is performed
C
*/
/*           within this loop
C
*/
/*
C
*/
/*
-----C */
/* call initialize_timing() */
L1000:
/* call start_timing(0) */
/* WRITE(*,*)-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- MISSILE STATE UPDATE MODULE
-----C */
/*
-----C */
/*           Integrate missile states to current time
C
*/
/*
C
*/
/*
-----C */
/* 1001   format(1x,f7.4,3(a,1p13.6)) */
/* 1002   format(1x,3(a,1p13.6)) */
/* write(message,1001)'p ',p,' q ',q,' r ',r */
/* call outmes(message) */
/* write(message,1002)'           pd ',pd,' qd ',qd,' rd ',rd */
/* call outmes(message) */
/* write(message,1002)'   cim(1) ',cim(1),' (2) ',cim(2), */
/* *           ' (3) ',cim(3) */
/* call outmes(message) */
/* write(message,1002)'           (4) ',cim(4),' (5) ',cim(5), */
/* *           ' (6) ',cim(6) */
/* call outmes(message) */
/* write(message,1002)'           (7) ',cim(7),' (8) ',cim(8), */
/* *           ' (9) ',cim(9) */
/* call outmes(message) */
  if (tstep >= tmsudriv) {
    tmsudriv += tmsustep;
  }
/*
-----

```

```

-----C */
/* -----C */
-----C */
/*
-----C */
/*
C */
/*
C */
/*
-----C */
masspr_(&t, &mdota, &mdotv, &mass, &eisp, &imass, &mdot, &weight,
&
      wdottp, &wdotkv, &wdottti, &ixx, &iyy, &izz);
/*
-----C */
/* -----C */
-----C */
/*
-----C */
/*
C */
/*
C */
/*
-----C */
missil2_(&t, quat, cim, &p, &q, &r, &ixx, &iyy, &izz, &mxacs,
&mxvcs,
      &myacs, &myvcs, &mzacs, &mzvcs, &xd, &y, &zd, &nclear, &pd,
&
      qd, &rd, &mx, &my, &mz, &u, &v, &w, quatd, &phi, &ttht,
&psi);
/*
-----C */
/* -----C */
-----C */
/*
C */
/*
-----C */
/*
derivatives C */
/*
not C */
/*
C */
/*
last C */
/*
integration C */
/*
coincide C */
/*
lookup C */
/*
C */
/*
C */
/*

```

Update mass flow rate, cg and inertia

VEHICLE STATES MODULE

Compute missile state derivatives

MISSILE STATE INTEGRATION MODULE

Revise missile states using just computed . Missile states must be integrated if a table lookup index transition has occurred since the integration step . The next step should be rescheduled to with the earliest detected table index transition instead . Otherwise schedule the next integration step to occur at the default step size .

```

C */
/*
C */
/*
-----C */
/*      TRAPEZOIDAL INTEGRATION FOR SIMPLICITY */
    spinteg_(&mass, &mdot, &t, &cs__1);
    spinteg_(&wkv, &wdotkv, &t, &cs__5);
    spinteg_(&p, &pd, &t, &cs__12);
    spinteg_(&q, &qd, &t, &cs__13);
    spinteg_(&r, &rd, &t, &cs__14);
    spinteg_(quat, quate, &t, &cs__15);
    spinteg_(quat[1], &quate[1], &t, &cs__16);
    spinteg_(quat[2], &quate[2], &t, &cs__17);
    spinteg_(quat[3], &quate[3], &t, &cs__18);
/*      SAVE TIME OF LAST MISSILE STATE UPDATE */
    tlmsu = t;
}
/*
-----C */
/*      ----- SEPARATION MODULE
-----C */
/*
-----C */
/*      Models discontinuities occurring during
C
*/
/*      stage separation
C
*/
C
*/
C
*/
C
*/
-----C */
/*      NOSE FAIRING / BOOST ADAPTER SEPARATION */
    if (idrop == 1 || (r_1 = t - tdrop, dabs(r_1)) <= dsteps && igit ==
1) {
        wkv -= wbanf;
        mass = wkv / xmtof;
        s_wsfi(&io__55);
        do_fio(&c__1, (char *)&t, (ftnlen)sizeof(real));
        e_wsfi();
        outmes_(message, 128L);
/*      REINITIALIZE PERTINENT INTEGRALS */
        spintegi_(&mass, &c_b14, &t, &cs__1);
        spintegi_(&wprop, &c_b14, &t, &cs__2);
        spintegi_(&impuls, &c_b14, &t, &cs__3);
        spintegi_(&wkv, &c_b14, &t, &cs__5);
    }
/*
-----C */
/*      ----- Processor communication
-----C */
/*
-----C */
/*      call switch_timing() */
/*      ----- Communicate with p01 -----C */

```

```

    send_real_32bit__(&ixx);
    send_real_32bit__(&iyy);
    send_real_32bit__(&izz);
    send_real_32bit__(&mass);
/* ----- Communicate with p03 -----C
*/
    send_real_32bit__(&p);
    send_real_32bit__(&q);
    send_real_32bit__(&r);
/*    CALL RECEIVE_REAL_64BIT( d_XD ) */
/*    XD = d_XD */
/*    CALL RECEIVE_REAL_64BIT( d_YD ) */
/*    YD = d_YD */
/*    CALL RECEIVE_REAL_64BIT( d_ZD ) */
/*    ZD = d_ZD */
    receive_real_32bit__(&xd);
    receive_real_32bit__(&yd);
    receive_real_32bit__(&zd);
    send_real_32bit__(cim);
    send_real_32bit__(&cim[1]);
    send_real_32bit__(&cim[2]);
    send_real_32bit__(&cim[3]);
    send_real_32bit__(&cim[4]);
    send_real_32bit__(&cim[5]);
    send_real_32bit__(&cim[6]);
    send_real_32bit__(&cim[7]);
    send_real_32bit__(&cim[8]);
/* ----- Communicate with p01 -----C */
    receive_signed_16bit__(&idrop);
/* ----- Receive from ACSTHR and VCSTHR -----C */
    receive_real_32bit__(&mdotv);
    receive_real_32bit__(&mdota);
    receive_real_32bit__(&mxvcs);
    receive_real_32bit__(&myvcs);
    receive_real_32bit__(&mzvcs);
    receive_real_32bit__(&mxacs);
    receive_real_32bit__(&myacs);
    receive_real_32bit__(&mzacs);
    send_real_32bit__(&pd);
    send_real_32bit__(&qd);
    send_real_32bit__(&rd);
/*    call switch_timing() */
/*
-----C */
/* ----- OUTPUT MODULE -----C */
/*
-----C */
/*    Creates print and plot output data
C
*/
/*    files
C
*/
C
*/
/*
-----C */
/*    call stop_timing() */
/*    if ( mod(int(tstep),int(dtpert)).eq.0 ) then */

```

```

/*      call output_timing() */
/*      call initialize_timing() */
/*      ENDIF */
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/*      Defines the simulation termination
C
*/
/*      conditions
C
*/
/*
C
*/
/*
-----C */
/*      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
iexit = 0;
/*      ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
*/

/*      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED */
if (t >= tfinal) {
    iexit = 1;
}
/*      increment time */
tstep += (float)1.;
t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

    if (iexit == 0) {
        goto L1000;
    }
    outmes_("ERROR: Exit from P00", 20L);
} /* MAIN__ */

```

## B.3.2 Uup01.c

```

/* uup01.f -- translated by f2c (version of 3 February 1990 3:36:42).
   You must link the resulting object file with the libraries:
      -lF77 -lI77 -lm -lc (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    shortint first2;
    doublereal tl2, grtpst[3];
} robtrg_;

#define robtrg_1 robtrg_

struct {
    doublereal grlast[3], t0nav;
    shortint mnav;
    doublereal dtx0, dty0, dtz0;
} rnavig_;

#define rnavig_1 rnavig_

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    /* System generated locals */
    real r_1;
    doublereal d_1, d_2;

    /* Local variables */
    static doublereal delt, delu;
    static real s_gr__[3];
    static doublereal delv, delw;
    static real s_xd__, s_yd__, s_zd__;
    static doublereal rtic[15] /* was [5][3] */ , mass, rrel[3],
vtic[15] /* was [5][3] */ , sphi, vrel[3], rmir[3], vmir[3], spsi, stht;
    static real s_pulsea__[3];
    static doublereal xyze[3];
    static real s_pulseg__[3];
    static doublereal timudriv, tgpudriv, timustep, tgpustep, t, x, y,
z;
    extern /* Subroutine */ int navig_();
    static doublereal dteps;
    static shortint iexit;
    static doublereal tstep, rtest[3], xyzed[3], vtest[3];
    extern /* Subroutine */ int receive_real_32bit__(),
receive_real_64bit__();
    static doublereal tuplk1, tuplk2, at[3], gr[3], xd, yd, zd, sp, sq,
sr,
    delphi, su, sv, sw, tfinal;
    extern /* Subroutine */ int obtarg_();

```

```

static doublereal delpsi, deltn;
static real s_mass;
static doublereal pulsea[3];
extern /* Subroutine */ int send_real_32bit__();
static doublereal pulseg[3], grtest[3];
extern /* Subroutine */ int imupro();
static doublereal qsl[4];
extern /* Subroutine */ int estre12();
static doublereal cie[9], rmi[3];
extern /* Subroutine */ int cw87();
static doublereal vmi[3], grt[15] /* was [5][3] */, sud, svd,
swd, ti2m[
9];

/*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
*/

/*      OUTPUTS */
/*      NAMELIST INPUTS */
/* DATA INITIALIZATION */
/* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE('SSp01.DAT') */
/* INITIALIZE 80x87 */
cw87_();
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/* Execution of all events is performed
C
*/
/* within this loop
C
*/
C
*/
/*
-----C */

```

```

-----C */
L1000:
/* WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- COMMUNICATION WITH P00
-----C */
    receive_real_32bit__(s_gr__);
    receive_real_32bit__(&s_gr__[1]);
    receive_real_32bit__(&s_gr__[2]);
    gr[0] = s_gr__[0];
    gr[1] = s_gr__[1];
    gr[2] = s_gr__[2];
    receive_real_32bit__(&s_mass__);
    mass = s_mass__;
    receive_real_32bit__(s_pulsea__);
    pulsea[0] = s_pulsea__[0];
    receive_real_32bit__(&s_pulsea__[1]);
    pulsea[1] = s_pulsea__[1];
    receive_real_32bit__(&s_pulsea__[2]);
    pulsea[2] = s_pulsea__[2];
    receive_real_32bit__(s_pulseg__);
    pulseg[0] = s_pulseg__[0];
    receive_real_32bit__(&s_pulseg__[1]);
    pulseg[1] = s_pulseg__[1];
    receive_real_32bit__(&s_pulseg__[2]);
    pulseg[2] = s_pulseg__[2];
    receive_real_64bit__(xyze);
    receive_real_64bit__(&xyze[1]);
    receive_real_64bit__(&xyze[2]);
    receive_real_64bit__(xyzed);
    receive_real_64bit__(&xyzed[1]);
    receive_real_64bit__(&xyzed[2]);
/* ----- COMMUNICATION WITH SEEKER
-----C */
    receive_real_64bit__(&x);
    receive_real_64bit__(&y);
    receive_real_64bit__(&z);
    receive_real_32bit__(&s_xd__);
    receive_real_32bit__(&s_yd__);
    receive_real_32bit__(&s_zd__);
    xd = s_xd__;
    yd = s_yd__;
    zd = s_zd__;
/* ----- COMMUNICATION WITH CORVEL
-----C */
    r_1 = rmir[0];
    send_real_32bit__(&r_1);
    r_1 = rmir[1];
    send_real_32bit__(&r_1);
    r_1 = rmir[2];
    send_real_32bit__(&r_1);
    r_1 = vmir[0];
    send_real_32bit__(&r_1);
    r_1 = vmir[1];
    send_real_32bit__(&r_1);
    r_1 = vmir[2];
    send_real_32bit__(&r_1);

```

```

receive_real_64bit__(grt);
receive_real_64bit__(&grt[5]);
receive_real_64bit__(&grt[10]);
receive_real_64bit__(rtic);
receive_real_64bit__(&rtic[5]);
receive_real_64bit__(&rtic[10]);
receive_real_64bit__(vtic);
receive_real_64bit__(&vtic[5]);
receive_real_64bit__(&vtic[10]);
/* ----- COMMUNICATE WITH CORVEL -----C */
    r_1 = at[0];
    send_real_32bit__(&r_1);
    r_1 = a[1];
    send_real_32bit__(&r_1);
    r_1 = at[2];
    send_real_32bit__(&r_1);
/* ----- DAISY CHAIN -----C */
    r_1 = ti2m[0];
    send_real_32bit__(&r_1);
    r_1 = ti2m[1];
    send_real_32bit__(&r_1);
    r_1 = ti2m[2];
    send_real_32bit__(&r_1);
    r_1 = ti2m[3];
    send_real_32bit__(&r_1);
    r_1 = ti2m[4];
    send_real_32bit__(&r_1);
    r_1 = ti2m[5];
    send_real_32bit__(&r_1);
    r_1 = ti2m[6];
    send_real_32bit__(&r_1);
    r_1 = ti2m[7];
    send_real_32bit__(&r_1);
    r_1 = ti2m[8];
    send_real_32bit__(&r_1);
    r_1 = vrel[0];
    send_real_32bit__(&r_1);
    r_1 = vrel[1];
    send_real_32bit__(&r_1);
    r_1 = vrel[2];
    send_real_32bit__(&r_1);
    r_1 = rrel[0];
    send_real_32bit__(&r_1);
    r_1 = rrel[1];
    send_real_32bit__(&r_1);
    r_1 = rrel[2];
    send_real_32bit__(&r_1);
    r_1 = sp;
    send_real_32bit__(&r_1);
    r_1 = sq;
    send_real_32bit__(&r_1);
    r_1 = sr;
    send_real_32bit__(&r_1);
/*
-----C */
/* ----- INERTIAL MEASUREMENT UPDATE -----C */
/*
-----C */
/*
C                                     Get inertial measurement data needed
C
*/

```

```

/*                                     for guidance calculations .
C
*/
/*
C
*/
/*
-----C */
    if (tstep >= timudriv) {
        timudriv += timustep;
    }
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/*                                     Convert gyro and accelerometer
outputs C */
/*                                     to delta angle and delta velocity
C */
/*
C */
/*
-----C */
    imupro_(&t, pulseg, pulsea, &delphi, &deltht, &delpsi, &delu,
&delv, &
        delw);
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/*                                     This module calculates the
quaternions C */
/*                                     and transformation matrices using
delta C */
/*                                     angles sensed by the gyro and
calculatesC */
/*                                     the interceptor velocity and position
C */
/*                                     using delta velocity sensed by the
C */
/*                                     accelerometer
C */
/*
-----C */
    navig_(&t, &mass, &delphi, &deltht, &delpsi, &delu, &delv, &delw,
gr,
        qsl, cie, &sp, &sq, &sr, &sud, &svd, &swd, vmir, rmir, ti2m,
&
        sphl, &stht, &spsi, &su, &sv, &sw, at, vmi, rmi);
/*
-----

```

```

-----C */
/* ----- MIDCOURSE CORRECTION
-----C */
/*
-----C */
/*
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
if ((d_1 = t - tuplk1, abs(d_1)) <= dsteps || (d_2 = t - tuplk2,
abs(d_2))
    <= dsteps) {
/*      REVISE ESTIMATED MISSILE STATES */
    vmi[0] = xyzed[0];
    vmi[1] = xyzed[1];
    vmi[2] = xyzed[2];
    rmi[0] = xyze[0];
    rmi[1] = xyze[1];
    rmi[2] = xyze[2];
    vmir[0] = xd;
    vmir[1] = yd;
    vmir[2] = zd;
    rmir[0] = x;
    rmir[1] = y;
    rmir[2] = z;
    rnavig_1.t0nav = t;
}
/*
-----C */
/* ----- MIDCOURSE CORRECTION
-----C */
/*
-----C */
/*
C
*/
/*
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
if ((d_1 = t - tuplk1, abs(d_1)) <= dsteps || (d_2 = t - tuplk2,
abs(d_2))
    <= dsteps) {
/*      REVISE ESTIMATED TARGET STATES */
    rtest[0] = rtic[0];
    rtest[1] = rtic[5];
    rtest[2] = rtic[10];
    vtest[0] = vtic[0];

```

```

        vtest[1] = vtic[5];
        vtest[2] = vtic[10];
        grtest[0] = grt[0];
        grtest[1] = grt[5];
        grtest[2] = grt[10];
        robtrg_1.tl2 = t;
    }
/*
-----C */
/*
C
*/
/*
-----C */
/*
C
*/
/*
C
*/
/*
-----C */
    if (tstep >= tgpudriv) {
        tgpudriv += tgpustep;
    }
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/*
C */
/*
C */
/*
C */
/*
-----C */
/*
    GRTEST TEMPORARILY EQUAL TO GRT */
    grtest[0] = grt[0];
    grtest[1] = grt[5];
    grtest[2] = grt[10];
    obtarg_(&t, grtest, rtest, vtest);
    estrel2_(rtest, vtest, rmir, vmir, rrel, vrel);
}
/*
-----C */
/* -----C */
/* -----C */
/*
-----C */
/*
C
*/
/*
C
*/
C

```

ON BOARD GUIDANCE PROCESSING

Determine guidance commands

Estimate target position based on predicted intercept conditions

TERMINATION LOGIC

Defines the simulation termination conditions

```

*/
/*
C
*/
/*
-----C */
/*   INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
   iexit = 0;
/*   ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
*/

/*   EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED */
   if (t >= tfinal) {
       iexit = 1;
   }
/*   increment time */
   tstep += 1.;
   t = tstep * delt;
/*   CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

   if (iexit == 0) {
       goto L1000;
   }
} /* MAIN__ */

```

## B.3.3 Uup02.c

```

/* uup02.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
   -lf77 -li77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    real trefl, tlstv, tmvcs[24] /* was [6][4] */, thvcs[24] /*
        was [6][4] */;
    shortint lenvcs[4];
} rvcstr_;

#define rvcstr_1 rvcstr_

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    static real delt, tbrk;
    static shortint ivcs;
    static real foff1[4], foff2[4], tmsudriv, tmsustep, t;
    static shortint ivtab;
    static real tvtab;
    static shortint iexit;
    static real mdotv, fxvcs, fyvcs, fzvcs, tstep, mxvcs, myvcs, mzvcs,
tkvon;

    extern /* Subroutine */ int receive_real_32bit__();
    static real cg[3], dtoffv[4];
    extern /* Subroutine */ int send_real_32bit__();
    static real tofflt[4], tburnm;
    extern /* Subroutine */ int vcsthr__();
    static real timonv;
    extern /* Subroutine */ int cw87__(), receive_signed_16bit__();

/*      THE FOLLOWING COMMON BLOCK USED FOR MIDFLIGHT CAPABILITIES ONLY
*/
/* DATA INITIALIZATION */
/* $INCLUDE('~/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA21.DAT') */

```

```

/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE('SSp02.DAT') */
/* INITIALIZE 80x87 */
    cw87_();
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*                               Execution of all events is performed
C
*/
/*                               within this loop
C
*/
/*
C
*/
/*
-----C */
L1000:
/*   WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- MISSILE STATE UPDATE MODULE
-----C */
/*
-----C */
/*                               Integrate missile states to current time
C
*/
/*
C
*/
/*
-----C */
/* ----- receive from masspr (P00) -----C */
    receive_real_32bit__(cg);
    receive_real_32bit__(&cg[1]);
    receive_real_32bit__(&cg[2]);
/* ----- Send variables to masspr and missil (p00)
-----C */
    send_real_32bit__(&mdotv);
    send_real_32bit__(&fxvcs);
    send_real_32bit__(&fyvcs);
    send_real_32bit__(&fzvcs);
    send_real_32bit__(&mxvcs);
    send_real_32bit__(&myvcs);
    send_real_32bit__(&mzvcs);
/* ----- Communication with p01 -----C */
    receive_real_32bit__(dtoffv);

```

```

receive_real_32bit__(&dtoffv[1]);
receive_real_32bit__(&dtoffv[2]);
receive_real_32bit__(&dtoffv[3]);
receive_signed_16bit__(&ivcs);
receive_signed_16bit__(&ivtab);
receive_real_32bit__(&tburnm);
receive_real_32bit__(&timonv);
receive_real_32bit__(tofflt);
receive_real_32bit__(&tofflt[1]);
receive_real_32bit__(&tofflt[2]);
receive_real_32bit__(&tofflt[3]);
receive_real_32bit__(&vtvtab);
if (tstep >= tmsudriv) {
    tmsudriv += tmsustep;
}

/*
-----C */
/* ----- VCS THRUSTER RESPONSE MODULE
-----C */
/*
-----C */
/*                                     Determines the forces and moments
   C */
/*                                     imparted by the VCS thrusters
   C */
/*
   C */
/*
-----C */
    if (t >= tkvon) {
        vcsthr_(&t, cg, &tburnm, &ivcs, tofflt, &timonv, dtoffv,
&stvtab,
                foff1, foff2, &ivtab, &tbrk, &fxvcs, &fyvcs, &fzvcs, &
mxvcs, &myvcs, &mzvcs, &mdotv);
    }
}

/*
-----C */
/* ----- TERMINATION LOGIC
-----C */
/*
-----C */
/*                                     Defines the simulation termination
   C
   */
/*                                     conditions
   C
   */
/*
   C
   */
/*
-----C */
/*     INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
iexit = 0;
/*     increment time */
tstep += (float)1.;
t = tstep * delt;
```

```

/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

    if (iexit == 0) {
        goto Li000;
    }
} /* MAIN__ */

/* uup02.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
   -lF77 -li77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    real treflv, tlstv, tmvcs[24] /* was [6][4] */ , thvcs[24] /*
        was [6][4] */;
    shortint lenvcs[4];
} rvcstr_;

#define rvcstr_1 rvcstr_

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    static real deltt, tbrk;
    static shortint ivcs;
    static real foff1[4], foff2[4], tmsudriv, tmsustep, t;
    static shortint ivtab;
    static real tvtab;
    static shortint iexit;
    static real mdotv, fxvcs, fyvcs, fzvcs, tstep, mxvcs, myvcs, mzvcs,
tkvon;

    extern /* Subroutine */ int receive_real_32bit__();
    static real cg[3], dtoffv[4];
    extern /* Subroutine */ int send_real_32bit__();
    static real tofflt[4], tburnm;
    extern /* Subroutine */ int vcsthr__();
    static real timonv;
    extern /* Subroutine */ int cw87__(), receive_signed_16bit__();

/*      THE FOLLOWING COMMON BLOCK USED FOR MIDFLIGHT CAPABILITIES ONLY
*/
/* DATA INITIALIZATION */
/* $INCLUDE('~/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('~/INCLUDE/SSDATA49.DAT') */

```

```

/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE('SSp02.DAT') */
/* INITIALIZE 80x87 */
    cw87_();
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*                               Execution of all events is performed
C
*/
/*                               within this loop
C
*/
/*
C
*/
/*
-----C */
L1000:
/* WRITE(*,*)'-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- MISSILE STATE UPDATE MODULE
-----C */
/*
-----C */
/*                               Integrate missile states to current time
C
*/
/*
C
*/
/*
-----C */
/* ----- recieve from masspr (P00) -----C */
    receive_real_32bit__(cg);
    receive_real_32bit__(&cg[1]);
    receive_real_32bit__(&cg[2]);
/* ----- Send variables to masspr and missil (p00)
-----C */
    send_real_32bit__(&mdotv);
    send_real_32bit__(&fxvcs);
    send_real_32bit__(&fyvcs);
    send_real_32bit__(&fzvcs);

```

```

    send_real_32bit__(&mxvcs);
    send_real_32bit__(&myvcs);
    send_real_32bit__(&mzvcs);
/* ----- Communication with p01 -----C */
    receive_real_32bit__(dtoffv);
    receive_real_32bit__(&dtoffv[1]);
    receive_real_32bit__(&dtoffv[2]);
    receive_real_32bit__(&dtoffv[3]);
    receive_signed_16bit__(&ivcs);
    receive_signed_16bit__(&ivtab);
    receive_real_32bit__(&tburnm);
    receive_real_32bit__(&timonv);
    receive_real_32bit__(tofflt);
    receive_real_32bit__(&tofflt[1]);
    receive_real_32bit__(&tofflt[2]);
    receive_real_32bit__(&tofflt[3]);
    receive_real_32bit__(&tvtab);
    if (tstep >= tmsudriv) {
        tmsudriv += tmsustep;
/*
-----C */
/* ----- VCS THRUSTER RESPONSE MODULE -----C */
/*
-----C */
/*
        Determines the forces and moments
    C */
/*
        imparted by the VCS thrusters
    C */
/*
    C */
/*
-----C */
    if (t >= tkvon) {
        vcsthr_(t, cg, &tburnm, &ivcs, tofflt, &timonv, dtoffv,
&tvtab,
        foff1, foff2, &ivtab, &tbrk, &fxvcs, &fyvcs, &fzvcs, &
mxvcs, &myvcs, &mzvcs, &mdotv);
    }
/*
-----C */
/* ----- TERMINATION LOGIC -----C */
/*
-----C */
/*
        Defines the simulation termination
    C
    */
/*
        conditions
    C
    */
/*
    C
    */
/*
-----C */
/*
    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */

```

```
    iexit = 0;
/*    increment time */
    tstep += (float)1.;
    t = tstep * delt;
/*    CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

    if (iexit == 0) {
        goto L1000;
    }
} /* MAIN__ */
```

## B.3.4 Up03.c

```

/* uup03.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lf77 -li77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    doublereal t11, grt1st[15]      /* was [5][3] */;
    shortint first1;
} rtarg_;

#define rtarg_1 rtarg_

/* Table of constant values */

static real c_b2 = (float)0.;
static shortint cs__1 = 1;
static shortint cs__2 = 2;
static shortint cs__3 = 3;
static integer c__1 = 1;

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{

    /* Format strings */
    static char fmt_889[] = "(1x,e16.9,\002 MISS = \002,e16.9)";

    /* System generated locals */
    real r_1;
    doublereal d_1, d_2, d_3;

    /* Builtin functions */
    double sqrt();
    integer s_wsfi(), do_fio(), e_wsfi();

    /* Local variables */
    static doublereal delt, rtic[15]      /* was [5][3] */;
    static real tphi;
    static doublereal latt, rtar[3], miss;
    static real tpsi;
    static doublereal rte[3], vtar[3];
    static real ttht;
    static doublereal vtic[15]      /* was [5][3] */;
    extern /* Subroutine */ int send_signed_16bit__();
    static doublereal tmsudriv, trsudriv, ttsudriv, tmsustep, trsustep,
        ttsustep;
    static real q, r;
    static doublereal t, x, y, z;
    static real tphid, ptarg, qtarg, rtarg, tpsid;
    static doublereal longt;
    static real tthtd;

```

```

        static doublereal rrelm[3], vrelm[3], tstep;
        extern /* Subroutine */ int receive_real_32bit__(),
receive_real_64bit__(),
        , spmmk__(), relat__();
        static doublereal tgotr, xmiss, ymiss, zmiss;
        static shortint iexit;
        static doublereal tgomn;
        static real xd, yd, zd;
        static doublereal lamsek[2], lamdsk[2], maglos, lamdtr[2], magrtr,
mgrdtr,
        lamdxx[2], lamtru[2], rreltr[3], vreltr[3], rj[5];
        extern /* Subroutine */ int send_real_64bit__();
        static shortint ireslv;
        extern /* Subroutine */ int send_real_32bit__();
        static doublereal cie[9], onegae;
        extern /* Subroutine */ int target__(), oucmes__();
        static real cim[9], cer[9], cit[9], cti[9];
        static doublereal cms[9], grt[15] /* was [5][3] */;
        static char message[128];
        extern /* Subroutine */ int cw87__();

        /* Fortran I/O blocks */
        static icilist io__68 = { 0, message, 0, fmt_889, 128, 1 };

/*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
*/

/*      OUTPUTS */
/*      NAMELIST INPUTS */
/* DATA INITIALIZATION */
/* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA0.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE('^/INCLUDE/SSMAS_cg.DAT') */
/* INITIALIZE 80x87 */
        cw87__();
/* $INCLUDE('SSp03.DAT') */
/*
-----C */
/* -----C */
/* -----C */
/* -----C */

```

```

-----C */
/*                               Execution of all events is performed
C
*/
/*                               within this loop
C
*/
/*
C
*/
/*
-----C */
L1000:
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- Communicate with p00 -----C
*/
    receive_real_32bit__(&q);
    receive_real_32bit__(&r);
    receive_real_64bit__(&x);
    receive_real_64bit__(&y);
    receive_real_64bit__(&z);
    receive_real_32bit__(&xd);
    receive_real_32bit__(&yd);
    receive_real_32bit__(&zd);
    receive_real_32bit__(cim);
    receive_real_32bit__(&cim[1]);
    receive_real_32bit__(&cim[2]);
    receive_real_32bit__(&cim[3]);
    receive_real_32bit__(&cim[4]);
    receive_real_32bit__(&cim[5]);
    receive_real_32bit__(&cim[6]);
    receive_real_32bit__(&cim[7]);
    receive_real_32bit__(&cim[8]);
/* ----- Communicate with p01 -----
C
*/
    send_real_64bit__(grt);
    send_real_64bit__(&grt[5]);
    send_real_64bit__(&grt[10]);
    send_signed_16bit__(&ireslv);
    r_1 = lamdxx[0];
    send_real_32bit__(&r_1);
    r_1 = lamdxx[1];
    send_real_32bit__(&r_1);
    r_1 = lamsek[0];
    send_real_32bit__(&r_1);
    r_1 = lamsek[1];
    send_real_32bit__(&r_1);
    r_1 = magrtr;
    send_real_32bit__(&r_1);
    send_real_64bit__(rtic);
    send_real_64bit__(&rtic[5]);
    send_real_64bit__(&rtic[10]);
    send_real_64bit__(vtic);
    send_real_64bit__(&vtic[5]);
    send_real_64bit__(&vtic[10]);

```

```

/*      WRITE(*,*) '-----BEGINNING OF LOOP-----' */
if (tstep >= tmsudriv) {
    tmsudriv += tmsustep;
/*      ROTATING EARTH MODEL */
    r_1 = omegae * t;
    spmmk_(&c_b2, &cs__1, &c_b2, &cs__2, &r_1, &cs__3, cer);
}
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/* Calculate relative range, range rate,
C
*/
/* time-to-go, LOS angles and rates
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
if (tstep >= trsudriv) {
    trsudriv += trsustep;
    relat_(rtic, vtic, &x, &y, &z, &xd, &y, &z, &q, &r, cim, cms,
        rreltr, &magrtr, vreltr, &mgrdrtr, &maglos, lamtru, lamd:,
        lamdtr, lamsek, lamsk, &tgotr, rrelm, vrelm);
/*      EXTRAPOLATE POINT OF CLOSEST APPROACH */
    xmiss = rreltr[0] + tgotr * vreltr[0];
    ymiss = rreltr[1] + tgotr * vreltr[1];
    zmiss = rreltr[2] + tgotr * vreltr[2];
/* Computing 2nd power */
    d_1 = xmiss;
/* Computing 2nd power */
    d_2 = ymiss;
/* Computing 2nd power */
    d_3 = zmiss;
    miss = sqrt(d_1 * d_1 + d_2 * d_2 + d_3 * d_3);
}
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/* This module calculates the true exo-
C
*/
/* atmospheric trajectory data for
C
*/
/* the target
C
*/
/*
C
*/

```

```

/*-----C */
    if (tstep >= ttsudriv) {
        ttsudriv += ttsustep;
        target_(&t, &magrtr, cer, cie, &ptarg, &qtarg, &rtarg, &tphi,
&ttht, &
            tpsi, grt, &tphid, &tthtd, &tpsid, cit, rtic, vtic, rtar,
            rter, &ireslv, rj, cti, vtar, &latt, &longt);
    }
/*-----C */
/* ----- TERMINATION LOGIC -----C */
/*-----C */
/* Defines the simulation termination conditions
C
*/
C
*/
C
*/
C
*/
/*-----C */
/* INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
iexit = 0;
/* ENABLE EXIT IF INTERCEPT HAS OCCURRED AND ALL EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED */
if (tgotr <= tgomn) {
    iexit = 1;
}
/* increment time */
tstep += 1.;
t = tstep * delt;
/* CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET */
if (iexit == 0) {
    goto L1000;
}
/*-----C */
/* ----- POINT OF CLOSEST APPROACH CALCULATION -----C */
/*-----C */
/* Determines the miss distance at the point of closest approach
C
*/
C
*/
C
*/
C

```

```

*/
/*
-----C */
/* Computing 2nd power */
    d_1 = rreltr[0] + vreltr[0] * tgotr;
/* Computing 2nd power */
    d_2 = rreltr[1] + vreltr[1] * tgotr;
/* Computing 2nd power */
    d_3 = rreltr[2] + vreltr[2] * tgotr;
    miss = sqrt(d_1 * d_1 + d_2 * d_2 + d_3 * d_3);
    s_wsfi(&io_68);
    do_fio(&c__1, (char *)&t, (ftnlen)sizeof(doublereal));
    do_fio(&c__1, (char *)&miss, (ftnlen)sizeof(doublereal));
    e_wsfi();
    outmes_(message, 128L);
} /* MAIN__ */

```

## B.3.5 Uup04.c

```

/* uup04.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lF77 -lI77 -lm -lc   (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    doublereal gset;
    shortint iset;
} norcom_;

#define norcom_1 norcom_

struct {
    real ranseq[97], ranlst;
} rancom_;

#define rancom_1 rancom_

struct {
    doublereal psig, thtg, phig, thxzg, thxyg, thyzg, thyxg, thzyg,
    thzxcg,
        sflg[3], sf2g[3], dcg[3], t0gyro, cimo[9], wbi2[3], wbi1[3],
    wbo2[
        3], wbo1[3], drsigg;
} rgyro_;

#define rgyro_1 rgyro_

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    /* System generated locals */
    real r_1;

    /* Local variables */
    static doublereal delt;
    extern /* Subroutine */ int gyro_();
    static doublereal timudriv, timustep, p, q, r, t;
    static real s_cim_[9];
    static shortint iexit;
    static doublereal tstep;
    extern /* Subroutine */ int receive_real_32bit__();
    static doublereal qfracg[3];
    static integer gyseed;
    extern /* Subroutine */ int send_real_32bit__();
    static doublereal pulseg[3], cim[9];
    static real s_p_, s_q_, s_r_;
    extern /* Subroutine */ int cw87_();

    /*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
    */

```

```

/* DATA INITIALIZATION */
/* $include ('^/include/ssdata35.dat') */
/* $include ('^/include/ssdata38.dat') */
/* $include ('^/include/ssdata39.dat') */
/* $include ('^/include/ssdata42.dat') */
/* $include ('^/include/ssdata44.dat') */
/* $include ('^/include/ssdata45.dat') */
/* $include ('^/include/ssdata46.dat') */
/* $include ('^/include/ssdata47.dat') */
/* $include ('^/include/ssdata48.dat') */
/* $include ('^/include/ssdata49.dat') */
/* $include ('^/include/ssdata50.dat') */
/* $include ('^/include/ssdata01.dat') */
/* $include ('^/include/ssdata17.dat') */
/* $include ('^/include/ssdata18.dat') */
/* $include ('^/include/ssdata21.dat') */
/* $include ('^/include/ssdata22.dat') */
/* $include ('^/include/ssdata23.dat') */
/* $include ('^/include/ssdata28.dat') */
/* $include ('^/include/ssdata29.dat') */
/* $include ('^/include/ssdata30.dat') */
/* $include ('^/include/ssdata71.dat') */
/* $include ('^/include/sstiming.dat') */
    cw87_();
/* $include ('ssp04.dat') */
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*                               Execution of all events is performed
C
*/
/*                               within this loop
C
*/
/*
C
*/
/*
-----C */
L1000:
/*    WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- Communicate with p01
-----C */
    r_1 = pulseg[0];
    send_real_32bit__(&r_1);
    r_1 = pulseg[1];
    send_real_32bit__(&r_1);
    r_1 = pulseg[2];
    send_real_32bit__(&r_1);

```

```

receive_real_32bit_(&s_p_);
receive_real_32bit_(&s_q_);
receive_real_32bit_(&s_r_);
p = (double) s_p_;
q = (double) s_q_;
r = (double) s_r_;
receive_real_32bit_(s_cim_);
receive_real_32bit_(&s_cim_[1]);
receive_real_32bit_(&s_cim_[2]);
receive_real_32bit_(&s_cim_[3]);
receive_real_32bit_(&s_cim_[4]);
receive_real_32bit_(&s_cim_[5]);
receive_real_32bit_(&s_cim_[6]);
receive_real_32bit_(&s_cim_[7]);
receive_real_32bit_(&s_cim_[8]);
cim[0] = (double) s_cim_[0];
cim[1] = (double) s_cim_[1];
cim[2] = (double) s_cim_[2];
cim[3] = (double) s_cim_[3];
cim[4] = (double) s_cim_[4];
cim[5] = (double) s_cim_[5];
cim[6] = (double) s_cim_[6];
cim[7] = (double) s_cim_[7];
cim[8] = (double) s_cim_[8];
/*
-----C */
/* ----- INERTIAL MEASUREMENT UPDATE
-----C */
/*
-----C */
/*                               Get inertial measurement data needed
C
*/
/*                               for guidance calculations .
C
*/
C
*/
C
*/
C
*/
-----C */
    if (tstep >= timudriv) {
        timudriv += timustep;
    }
/*
-----C */
/* ----- GYRO MODULE
-----C */
/*
-----C */
/*                               Determine sensed body rates .
C */
/*
C */
/*
C */
/*
-----C */
    gyro_(&t, &p, &q, &r, cim, &gyseed, qfracg, pulseseg);
}
/*

```

```

-----C */
/* ----- TERMINATION LOGIC
-----C */
/*
-----C */
/*          Defines the simulation termination
C
*/
/*          conditions
C
*/
/*
C
*/
/*
-----C */
/*      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
    iexit = 0;
/*      increment time */
    tstep += 1.;
    t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

        if (iexit == 0) {
            goto L1000;
        }
} /* MAIN */

```

## B.3.6 Uup05.c

```

/* uup05.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lF77 -li77 -lm -lc   (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    real gset;
    shortint iset;
} norcom_;

#define norcom_1 norcom_

struct {
    real ranseq[97], ranlst;
} rancom_;

#define rancom_1 rancom_

struct {
    real trefla, tlstc, acsf, aoff1[4], aoff2[4], tmacsa[32] /* was
[8][4]
        */, thacsa[32] /* was [8][4] */;
    shortint lena[4];
    real tmacsb[32] /* was [8][4] */, thacsb[32] /* was [8][4] */;
    shortint lenb[4];
} racstr_;

#define racstr_1 racstr_

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    static real delt, tbrk;
    extern /* Subroutine */ int send_signed_16bit__();
    static real tmsudriv, tmsustep, t, tatab, mdota, fxacs, fyacs,
fzacs,
        mxacs, myacs, mzacs;
    static shortint iexit;
    static real tstep, tkvon;
    extern /* Subroutine */ int receive_real_32bit__();
    static real cg[3], dtacsa[4], dtacsb[4];
    static shortint iacson;
    static real acslev;
    static integer toseed;
    extern /* Subroutine */ int acsthr__();
    static real timona;
    extern /* Subroutine */ int send_real_32bit__();
    static shortint ithres;
    extern /* Subroutine */ int cw87__(), receive_signed_16bit__();

```

```

/*      THE FOLLOWING COMMON BLOCK USED FOR MIDFLIGHT CAPABILITIES ONLY
*/
/* DATA INITIALIZATION */
/* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* INITIALIZE 80x87 */
    cw87_();
/*      DETERMINE IF MIDFLIGHT RESTART */
/* $INCLUDE('SSp05.DAT') */
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*      Execution of all events is performed
C
*/
/*      within this loop
C
*/
C
*/
C
*/
C
*/
C
*/
-----C */
L1000:
/*      WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- MISSILE STATE UPDATE MODULE
-----C */
/*
-----C */
/*      Integrate missile states to current time
C
*/
/*
C
*/

```

```

*/
/*
-----C */
/* ----- recieve from masspr (P00) -----C */
    receive_real_32bit__(cg);
    receive_real_32bit__(&cg[1]);
    receive_real_32bit__(&cg[2]);
/* ----- Send variables to masspr and missil (p00)
-----C */
    send_real_32bit__(&mdota);
    send_real_32bit__(&fxacs);
    send_real_32bit__(&fyacs);
    send_real_32bit__(&fzacs);
    send_real_32bit__(&mxacs);
    send_real_32bit__(&myacs);
    send_real_32bit__(&mzacs);
/* ----- Communication with p01 -----C */
    receive_real_32bit__(&acslev);
    receive_real_32bit__(dtacsa);
    receive_real_32bit__(&dtacsa[1]);
    receive_real_32bit__(&dtacsa[2]);
    receive_real_32bit__(&dtacsa[3]);
    receive_real_32bit__(dtacsb);
    receive_real_32bit__(&dtacsb[1]);
    receive_real_32bit__(&dtacsb[2]);
    receive_real_32bit__(&dtacsb[3]);
    receive_signed_16bit__(&ithres);
    receive_real_32bit__(&tatab);
    send_signed_16bit__(&iacson);
/*
-----C */
    if (tstep >= tmsudriv) {
        tmsudriv += tmsustep;
        if (t >= tkvon) {
/*
-----C */
/* ----- ACS THRUSTER RESPONSE MODULE
-----C */
/*
-----C */
/* ----- Determines the forces and moments
C */
/* ----- imparted by the ACS thrusters
C */
/* ----- C */
/*
-----C */
    acsthr_(&t, cg, &acslev, dtacsa, dtacsb, &tatab, &toseed,
&tbrk, &
        ithres, &fxacs, &fyacs, &fzacs, &mxacs, &myacs, &mzacs,
&
        mdota, &iacson, &timona);
    }
/*
-----C */
/* ----- TERMINATION LOGIC

```

```

-----C */
/*
-----C */
/*                                     Defines the simulation termination
C
*/
/*                                     conditions
C
*/
/*
C
*/
/*
-----C */
/*      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
      iexit = 0;
/*      increment time */
      tstep += (float)1.;
      t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

      if (iexit == 0) {
          goto L1000;
      }
} /* MAIN__ */

```

## B.3.7 Uup06.c

```

/* uup06.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lF77 -lI77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Table of constant values */

static shortint cs__20 = 20;

/*      PROGRAM EXOSIM */
/* -----
-C */
/* --- ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    static real delt, mass, tmsudriv, tmsustep, t;
    static shortint iexit;
    static real tstep;
    extern /* Subroutine */ int receive_real_32bit__();
    static real masst1[20], cg[3], dt;
    extern /* Subroutine */ int send_real_32bit__();
    static shortint icg;
    static real cgx[20], cgy[20], cgz[20];
    extern /* Subroutine */ int cw87_(), sptable_();

/* DATA INITIALIZATION */
/* $INCLUDE('~/INCLUDE/SSMAS_cg.DAT') */
/* INITIALIZE 80x87 */
    cw87_();
/*      RESTARTING FROM MIDFLIGHT DATA FILE */
/* $INCLUDE('SSp06.DAT') */
/*
-----C */
/* ----- MAIN EXECUTION LOOP -----C */
/*
-----C */
/*      Execution of all events is performed
C
*/
/*      within this loop
C
*/
C
*/
/*
C
*/
/*
-----C */
L1000:
/*      WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */

```

```

/* ----- MISSILE STATE UPDATE MODULE
-----C */
/*
-----C */
/*
Integrate missile states to current time
C
*/
/*
C
*/
/*
-----C */
    if (tstep >= tmsudriv) {
        tmsudriv += tmsustep;
        dt = tmsustep * delt;
    }
/*
-----C */
/* ----- MASS PROPERTIES MODULE
-----C */
/*
-----C */
/*
Update cg
C */
/*
C */
/*
-----C */
/*
CALCULATE MISSILE CENTER OF GRAVITY COMPONENTS */
sptable_(masst1, cgx, &mass, cg, &cs_20, &icg);
sptable_(masst1, cgy, &mass, &cg[1], &cs_20, &icg);
sptable_(masst1, cgz, &mass, &cg[2], &cs_20, &icg);
}
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- communication with missil model */
receive_real_32bit__(&mass);
/* ----- send to ACSTHR and VCSTHR and ACCEL */
send_real_32bit__(cg);
send_real_32bit__(&cg[1]);
send_real_32bit__(&cg[2]);
/*
-----C */
/* ----- OUTPUT MODULE
-----C */
/*
-----C */
/*
Creates print and plot output data
C
*/
/*
files
C

```

```

*/
/*
C
*/
/*
-----C */
/*      if ( mod(idnint(tstep),idnint(dtpert)).eq.0 ) then */
/*      ENDIF */
/*
-----C */
/* ----- TERMINATION LOGIC
-----C */
/*
-----C */
/*                      Defines the simulation termination
C
/*                      conditions
C
/*
/*
C
/*
/*
-----C */
/*      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
/*      iexit = 0;
/*      increment time */
/*      tstep += (float)1.;
/*      t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

      if (iexit == 0) {
        goto L1000;
      }
} /* MAIN__ */

```

## B.3.8 Uup07.c

/\* uup07.f -- translated by f2c (version of 3 February 1990 3:36:42).  
 You must link the resulting object file with the libraries:  
     -lF77 -lI77 -lm -lc   (in that order)

```

*/

#include "f2c.h"

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    /* System generated locals */
    real r_1, r_2;

    /* Builtin functions */
    double r_nint();

    /* Local variables */
    static real tffe, delt, ttfe, rmir[3], vmir[3], vttp[3], timudriv,
        tgpudriv, dtmpl, timustep, tgpustep, t, x, y, z, dteps;
    static shortint iexit;
    static real dtcvu, tcorv, tstep;
    extern /* Subroutine */ int receive_real_32bit__();
    static real tuplk1, tuplk2, at[3], dt, vc[3], xd, vg[3], yd, zd,
vs[3],
        tfinal;
    extern /* Subroutine */ int corvel_(), send_real_32bit__();
    static real dtspvc, dlv[3];
    extern /* Subroutine */ int cw87_();
    static real ttf, mvr, mvs, vtt[3], uvs[3];

    /* DATA INITIALIZATION */
    /* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
    /* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
    /* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */

```

```

/* INITIALIZE 80x87 */
  cw87_();
/* $INCLUDE('SSp07.DAT') */
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*                               Execution of all events is performed
C
*/
/*                               within this loop
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
/*      CALL INITIALIZE_TIMING() */
L1000:
/*      CALL START_TIMING(0) */
/*      WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/*      CALL SWITCH_TIMING() */
/* ----- COMMUNICATION WITH P00
-----C */
  receive_real_32bit_(&x);
  receive_real_32bit_(&y);
  receive_real_32bit_(&z);
  receive_real_32bit_(&xd);
  receive_real_32bit_(&yd);
  receive_real_32bit_(&zd);
  receive_real_32bit_(rmir);
  receive_real_32bit_(&rmir[1]);
  receive_real_32bit_(&rmir[2]);
  receive_real_32bit_(vmir);
  receive_real_32bit_(&vmir[1]);
  receive_real_32bit_(&vmir[2]);
/* ----- COMMUNICATION WITH P01
-----C */
  receive_real_32bit_(at);
  receive_real_32bit_(&at[1]);
  receive_real_32bit_(&at[2]);
  send_real_32bit_(vg);
  send_real_32bit_(&vg[1]);
  send_real_32bit_(&vg[2]);
/*      CALL SWITCH_TIMING() */
/*
-----C */
/* ----- INERTIAL MEASUREMENT UPDATE
-----C */

```

```

/*
-----C */
/*
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
    if (tstep >= timudriv) {
        timudriv += timustep;
/*
        TIME SINCE LAST INERTIAL MEASUREMENT UPDATE */
        dt = timustep * delt;
/*
        INTEGRATE GRAVITY COMPENSATED ACCELERATION */
        vtt[0] += dt * at[0];
        vtt[1] += dt * at[1];
        vtt[2] += dt * at[2];
    }
/*
-----C */
/*
-----C */
/*
-----C */
/*
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
    if ((r_1 = t - tuplk1, dabs(r_1)) <= dsteps || (r_2 = t - tuplk2,
dabs(r_2)
    ) <= dsteps) {
/*
        REVISE ESTIMATED MISSILE STATES */
        vmir[0] = xd;
        vmir[1] = yd;
        vmir[2] = zd;
        rmir[0] = x;
        rmir[1] = y;
        rmir[2] = z;
    }
/*
-----C */
/*
C
*/
/*
-----C */

```

```

/*                                     Determine guidance commands
C
*/
/*
C
*/
/*
-----C */
      if (tstep >= tgpudriv) {
          tgpudriv += tgpustep;
/*
-----C */
/* -----CORRELATED VELOCITY MODULE
-----C */
/*
-----C */
/*                                     This section calculates the
correlated C */
/*                                     velocity vector (VC) through an iter-
/*                                     ative process. From VC, the steering
/*                                     velocity vector is produced by sub-
/*                                     tracting a bias velocity (VD0) from
the C */
/*                                     velocity to be gained (VG).
/*
C */
/*
C */
/*
-----C */
      if (t >= tcorv && t <= f - dtspvc) {
          corvel_(&t, &mvr, vcc, rmir, vmir, vttp, vg, vs, &mvs, uvs,
vc,
          dl, &stffe, &sttfe);
          r_1 = (t + dtcvu) / dtcvu;
          dtmpl = dtcvu * r_nint(&r_1);
          tcorv = dtmpl;
      }
/*
-----C */
/* -----OUTPUT MODULE
-----C */
/*
-----C */
/*      call stop_timing() */
/*      if ( mod(int(tstep),int(dtprt)).eq.0 ) then */
/*          call outpl_timing() */
/*          call INITIALIZE_TIMING() */
/*      ENDIF */
/*
-----C */
/* -----TERMINATION LOGIC
-----C */
/*

```

```

-----C */
/*                               Defines the simulation termination
C
*/
/*                               conditions
C
*/
/*
C
*/
/*
C
*/
/*
C
*/
-----C */
/*      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
iexit = 0;
/*      ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
*/

/*      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED */
if (t >= tfinal) {
    iexit = 1;
}
/*      increment time */
tstep += 1.;
t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

    if (iexit == 0) {
        goto L1000;
    }
} /* MAIN__ */

```

## B.3.9 Uup08.c

```

/* uup08.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
   -lF77 -lI77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    doublereal xint[50], tint[50], xdot1[50];
} storag_;

#define storag_1 storag_

struct {
    doublereal xyzlch[3];
} rmissl_;

#define rmissl_1 rmissl_

/* Table of constant values */

static shortint cs__6 = 6;
static shortint cs__7 = 7;
static shortint cs__8 = 8;
static shortint cs__9 = 9;
static shortint cs__10 = 10;
static shortint cs__11 = 11;
static doublereal c_b8 = 0.;
static shortint cs__1 = 1;
static shortint cs__2 = 2;
static shortint cs__3 = 3;
static integer c__1 = 1;

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{

    /* Format strings */
    static char fmt_202[] = "(1x,f8.4,4e14.7)";

    /* System generated locals */
    real r_1;
    doublereal d_1, d_2, d_3;

    /* Builtin functions */
    double sqrt(), atan2();
    integer s_wsfi(), do_fio(), e_wsfi();

    /* Local variables */
    static doublereal rade, delt, long_, mass, xyze[3], xyzr[3],
tmsudriv,

```

```

        tmsustep, t, x, y;
        static real s_cim__[9];
        static doublereal z, xyzed[3], tstep, fxacs, fxvcs, fyacs, fyvcs,
        fzacs,
        fzvcs;
        extern /* Subroutine */ int integ_();
        static doublereal tmsu;
        extern /* Subroutine */ int receive_real_32bit__();
        static doublereal dtprt;
        static shortint iexit;
        static doublereal gb[3], gr[3];
        static real s_mass__;
        static doublereal mxyzdd, xyzedd[3];
        extern /* Subroutine */ int missil_();
        static shortint nclear;
        static doublereal ud, vd, wd, fx, cie[9], fy, fz, xd, yd, zd,
        cim[9], cer[
        9], cir[9], lat, mgr, cri[9];
        static char message[128];
        static real s_fxacs__, s_fyacs__, s_fzacs__, s_fxvcs__, s_fyvcs__,
        s_fzvcs__;
        extern /* Subroutine */ int cw87_();
        static dcoublereal xdd, ydd, zdd;
        extern /* Subroutine */ int mmk_();
        static doublereal omegae, dtr, alt;
        extern /* Subroutine */ int send_real_32bit__(),
        send_real_64bit__();
        static shortint iprint;
        extern /* Subroutine */ int outmes_();
        static doublereal tfinal;

        /* Fortran I/O blocks */
        static icilist io__63 = { 0, message, 0, fmt_202, 128, 1 };

/*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
*/

/*      OUTPUTS */
/*      NAMELIST INPUTS */
/* DATA INITIALIZATION */
/* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE('^/INCLUDE/SSMAS_cg.DAT') */

```

```

/* INITIALIZE 80x87 */
  cw87_();
/* $INCLUDE('SSp08.DAT') */
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*           Execution of all events is performed
C
*/
/*           within this loop
C
*/
/*
C
*/
/*
-----C */
L1000:
/*   WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- MISSILE STATE UPDATE MODULE
-----C */
/*
-----C */
/*           Integrate missile states to current time
C
*/
/*
C
*/
/*
-----C */
  if (tstep >= tmsudriv) {
    tmsudriv += tmsustep;
  }
/*
-----C */
/* ----- VEHICLE STATES MODULE
-----C */
/*
-----C */
/*           Compute missile state derivatives
  C */
/*
  C */
/*
-----C */
  missil_(&t, &cim, &mass, &fxacs, &fxvcs, &fyacs, &fyvcs, &fzacs, &
    fzvcs, &x, &y, &z, &nclear, &ud, &vd, &wd, &gb, &gr, &mgr,
    &fx,
    &fy, &fz, &xdd, &ydd, &zdd, &mxyzdd);
/*

```

```

-----C */
/*                                MISSILE STATE INTEGRATION MODULE
C */
/*
-----C */
/*                                Revise missile states using
derivatives C */                                just computed . Missile states must
/*                                be integrated if a table lookup index
not C */                                transition has occurred since the
/*                                integration step . The next
C */                                step should be rescheduled to
/*                                with the earliest detected table
last C */                                index transition instead . Otherwise
/*                                schedule the next integration step to
integration C */                                occur at the default step size .
/*
coincide C */
/*
lookup C */
/*
C */
/*
C */
/*
C */
/*
C */
/*
-----C */
/*                                TRAPEZOIDAL INTEGRATION FOR SIMPLICITY */
integ_(&xd, &xdd, &t, &cs_6);
integ_(&y, &ydd, &t, &cs_7);
integ_(&z, &zdd, &t, &cs_8);
integ_(&x, &xd, &t, &cs_9);
integ_(&y, &ydd, &t, &cs_10);
integ_(&z, &zdd, &t, &cs_11);
/*                                TRANSFORM INERTIAL POSITION AND VELOCITY TO EARTH FRAME */
xyze[0] = cie[0] * x + cie[3] * y + cie[6] * z;
xyze[1] = cie[1] * x + cie[4] * y + cie[7] * z;
xyze[2] = cie[2] * x + cie[5] * y + cie[8] * z;
xyzed[0] = cie[0] * xd + cie[3] * yd + cie[6] * zd;
xyzed[1] = cie[1] * xd + cie[4] * yd + cie[7] * zd;
xyzed[2] = cie[2] * xd + cie[5] * yd + cie[8] * zd;
xyzedd[0] = cie[0] * xdd + cie[3] * ydd + cie[6] * zdd;
xyzedd[1] = cie[1] * xdd + cie[4] * ydd + cie[7] * zdd;
xyzedd[2] = cie[2] * xdd + cie[5] * ydd + cie[8] * zdd;
/*                                ROTATING EARTH MODEL */
d_1 = omegae * t;
mmk_(&c_b8, &cs_1, &c_b8, &cs_2, &d_1, &cs_3, cer);
xyzr[0] = cer[0] * xyze[0] + cer[3] * xyze[1] + cer[6] * xyze[2];
xyzr[1] = cer[1] * xyze[0] + cer[4] * xyze[1] + cer[7] * xyze[2];
xyzr[2] = cer[2] * xyze[0] + cer[5] * xyze[1] + cer[8] * xyze[2];
cir[0] = cer[0] * cie[0] + cer[3] * cie[1] + cer[6] * cie[2];
cir[1] = cer[1] * cie[0] + cer[4] * cie[1] + cer[7] * cie[2];
cir[2] = cer[2] * cie[0] + cer[5] * cie[1] + cer[8] * cie[2];
cir[3] = cer[0] * cie[3] + cer[3] * cie[4] + cer[6] * cie[5];
cir[4] = cer[1] * cie[3] + cer[4] * cie[4] + cer[7] * cie[5];
cir[5] = cer[2] * cie[3] + cer[5] * cie[4] + cer[8] * cie[5];
cir[6] = cer[0] * cie[6] + cer[3] * cie[7] + cer[6] * cie[8];
cir[7] = cer[1] * cie[6] + cer[4] * cie[7] + cer[7] * cie[8];
cir[8] = cer[2] * cie[6] + cer[5] * cie[7] + cer[8] * cie[8];

```

```

        cri[0] = cir[0];
        cri[1] = cir[3];
        cri[2] = cir[6];
        cri[3] = cir[1];
        cri[4] = cir[4];
        cri[5] = cir[7];
        cri[6] = cir[2];
        cri[7] = cir[5];
        cri[8] = cir[8];
/*      CALCULATE CURRENT LATITUDE AND LONGITUDE */
/* Computing 2nd power */
    d_1 = xyzr[0];
/* Computing 2nd power */
    d_2 = xyzr[1];
    lat = atan2(xyzr[2], (sqrt(d_1 * d_1 + d_2 * d_2))) / dtr;
    long_ = atan2(xyzr[1], xyzr[0]) / dtr;
/*      CALCULATE CURRENT MISSILE ALTITUDE */
/* Computing 2nd power */
    d_1 = x;
/* Computing 2nd power */
    d_2 = y;
/* Computing 2nd power */
    d_3 = z;
    alt = sqrt(d_1 * d_1 + d_2 * d_2 + d_3 * d_3) - rade;
/*      SAVE TIME OF LAST MISSILE STATE UPDATE */
    tlmsu = t;
}
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- Communicate with p01 -----C */
    r_1 = gr[0];
    send_real_32bit__(&r_1);
    r_1 = gr[1];
    send_real_32bit__(&r_1);
    r_1 = gr[2];
    send_real_32bit__(&r_1);
    receive_real_32bit__(&s_mass__);
    mass = s_mass__;
    send_real_64bit__(xyze);
    send_real_64bit__(&xyze[1]);
    send_real_64bit__(&xyze[2]);
    send_real_64bit__(xyzed);
    send_real_64bit__(&xyzed[1]);
    send_real_64bit__(&xyzed[2]);
/* ----- Communicate with p03 -----C
*/
    send_real_64bit__(&x);
    send_real_64bit__(&y);
    send_real_64bit__(&z);
    r_1 = x;
    send_real_32bit__(&r_1);
    r_1 = y;
    send_real_32bit__(&r_1);
    r_1 = z;
    send_real_32bit__(&r_1);
    r_1 = xd;
    send_real_32bit__(&r_1);
    r_1 = yd;

```

```

send_real_32bit__(&r_1);
r_1 = zd;
send_real_32bit__(&r_1);
receive_real_32bit__(s_cim__);
receive_real_32bit__(&s_cim__[1]);
receive_real_32bit__(&s_cim__[2]);
receive_real_32bit__(&s_cim__[3]);
receive_real_32bit__(&s_cim__[4]);
receive_real_32bit__(&s_cim__[5]);
receive_real_32bit__(&s_cim__[6]);
receive_real_32bit__(&s_cim__[7]);
receive_real_32bit__(&s_cim__[8]);
cim[0] = s_cim__[0];
cim[1] = s_cim__[1];
cim[2] = s_cim__[2];
cim[3] = s_cim__[3];
cim[4] = s_cim__[4];
cim[5] = s_cim__[5];
cim[6] = s_cim__[6];
cim[7] = s_cim__[7];
cim[8] = s_cim__[8];
/* ----- Receive from ACSTHR and VCSTHR -----C */
receive_real_32bit__(&s_fxvcs__);
receive_real_32bit__(&s_fyvcs__);
receive_real_32bit__(&s_fzvcs__);
receive_real_32bit__(&s_fxacs__);
receive_real_32bit__(&s_fyacs__);
receive_real_32bit__(&s_fzacs__);
fxvcs = s_fxvcs__;
fyvcs = s_fyvcs__;
fzvcs = s_fzvcs__;
fxacs = s_fxacs__;
fyacs = s_fyacs__;
fzacs = s_fzacs__;
r_1 = ud;
send_real_32bit__(&r_1);
r_1 = vd;
send_real_32bit__(&r_1);
r_1 = wd;
send_real_32bit__(&r_1);
/*
-----C */
/* ----- OUTPUT MODULE -----C */
/*
-----C */
/*                                     Creates print and plot output data
C
*/
/*                                     files
C
*/
/*
C
*/
/*
-----C */
++iprint;
if (iprint == (shortint) dtprt) {
    s_wsfi(&io_63);
    do_fio(&c_1, (char *)&t, (ftnlen)sizeof(doublereal));

```

```

do_fio(&c__1, (char *)&alt, (ftnlen)sizeof(doublereal));
do_fio(&c__1, (char *)&x, (ftnlen)sizeof(doublereal));
do_fio(&c__1, (char *)&y, (ftnlen)sizeof(doublereal));
do_fio(&c__1, (char *)&z, (ftnlen)sizeof(doublereal));
e_wsfi();
outmes_(message, 128L);
iprint = 0;
}
/*
-----C */
/* -----C */
/* -----C */
/*
-----C */
/*
C
*/
/*
C
*/
C
*/
C
*/
-----C */
/* INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
iexit = 0;
/* ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
*/
/* EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED */
if (t >= tfinal) {
    iexit = 1;
}
/* ENABLE EXIT IF MISSILE HAS IMPACTED AND ALL EVENTS SCHEDULED FOR
*/
/* THIS TIME HAVE BEEN EXECUTED */
if (alt < 0.) {
    iexit = 1;
}
/* increment time */
tstep += 1.;
t = tstep * delt;
/* CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/
    if (iexit == 0) {
        goto L1000;
    }
    outmes_("ERROR: Exit from P08", 20L);
} /* MAIN__ */

```

## B.3.10 Uup09.c

```

/* uup09.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lF77 -lI77 -lm -lc   (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    real gset;
    shortint iset;
} norcom_;

#define norcom_1 norcom_

struct {
    real ranseq[97], ranlst;
} rancom_

#define rancom_1 rancom_

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    static shortint acqd;
    static real lamm[2], delt;
    static shortint term;
    static real tspudriv, t;
    static shortint track, iexit;
    static real tstep;
    extern /* Subroutine */ int receive_real_32bit__();
    static real lamsek[2];
    static integer skseed;
    extern /* Subroutine */ int seeker_();
    static real samrat;
    static shortint frmcnt,
    static real frmrat, magrtr;
    extern /* Subroutine */ int send_real_32bit__(), cw87_();
    static real snr;

/* $INCLUDE('pfp:INCLUDE/target.for') */
/* INITIALIZE 80x87 */
    cw87_();
/* $INCLUDE('ssp09.dat') */
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/*      Execution of all events is performed
C

```

```

*/
/*                                within this loop
C
*/
/*
C
*/
/*
-----C */
/* CALL INITIALIZE_TIMING() */
L1000:
/* call reset_timer() */
/* timer = read_timer() */
/* CALL START_TIMING(0) */
/* WRITE(*,*)'-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* CALL SWITCH_TIMING() */
/* ----- COMMUNICATION WITH KALMAN
-----C */
send_real_32bit__(lamm);
send_real_32bit__(&lamm[1]);
send_real_32bit__(&snr);
send_real_32bit__(&frmrat);
/* ----- COMMUNICATION WITH RELAT
-----C */
receive_real_32bit__(lamsek);
receive_real_32bit__(&lamsek[1]);
receive_real_32bit__(&magrtr);
/* CALL SWITCH_TIMING() */
/*
-----C */
/* ----- SEEKER MODULE
-----C */
/*
-----C */
/*                                Calculates LOS angles measured by the
C
*/
/*                                seeker
C
*/
/*
C
*/
/*
-----C */
if (tstep >= tspudriv) {
/* TSPUDRIV = TSPUDRIV + TSPUSTEP */
seeker_(&t, &acqd, lamsek, &magrtr, &skseed, &frmrat, &frmcnt, &
samrat, &track, &term, &snr, lamm);
tspudriv += (shortint) ((float)le3 / frmrat);
}
/* delt_time = (timer -(read_timer() + 18))/1.229e6 */

```

```

/*      CALL output_message( %VAL(real_32bit), delt_time, */
/*      & %VAL(int2(1)) ) */
/*      call output_n1 */
/*
-----C */
/* -----C */ OUTPUT MODULE
/* -----C */
/*
-----C */
/*      call stop_timing() */
/*      if ( mod(idnint(tstep),idnint(dtpert)).eq.0 ) then */
/*      call output_timing() */
/*      call INITIALIZE_TIMING() */
/*      ENDIF */
/*
-----C */
/* -----C */ TERMINATION LOGIC
/* -----C */
/*
-----C */
/*      Defines the simulation termination
C
*/
/*      conditions
C
*/
/*
C
*/
/*
-----C */
/*      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
/*      iexit = 0;
/*      increment time */
/*      tstep += (float)1.;
/*      t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

      if (iexit == 0) {
        goto L1000;
      }
} /* MAIN__ */

```

## B.3.11 Uup10.c

```

/* uup10.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lF77 -lI77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    doublereal gset;
    shortint iset;
} norcom_;

#define norcom_1 norcom_

struct {
    real ranseq[97], ranlst;
} rancom_;

#define rancom_1 rancom_

struct {
    doublereal drsiga, psia, thta, phia, thxza, thxya, thyza, thyxa,
    thzya,
        thzxa, sfla[3], sf2a[3], dca[3], t0acce, grlst[3], xyzdp[3],
    abi2[
        3], abil[3], abo2[3], abol[3];
} raccel_;

#define raccel_1 raccel_

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    /* System generated locals */
    real r_1;

    /* Local variables */
    static real s_cg__[3], s_pd__, s_qd__, s_rd__;
    static doublereal delt;
    static real s_ud__, s_vd__, s_wd__, s_xd__, s_yd__, s_zd__,
    s_gr__[3];
    static doublereal timudriv, timustep;
    extern /* Subroutine */ int accel_();
    static doublereal p, q, r, t;
    static real s_cim__[9];
    static shortint iexit;
    static doublereal tstep;
    extern /* Subroutine */ int receive_real_32bit__();
    static doublereal cg[3], pd, qd, rd, gr[3], ud, vd, xd, qfrac[3],
    yd, zd,
        wd;
    static integer gyseed;
    static doublereal pulsea[3];

```

```

extern /* Subroutine */ int send_real_32bit__();
static doublereal cim[9];
static real s_p__, s_q__, s_r__;
extern /* Subroutine */ int cw87_();

/*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
*/

/* DATA INITIALIZATION */
/* $include ('^/include/ssdata35.dat') */
/* $include ('^/include/ssdata38.dat') */
/* $include ('^/include/ssdata39.dat') */
/* $include ('^/include/ssdata42.dat') */
/* $include ('^/include/ssdata44.dat') */
/* $include ('^/include/ssdata45.dat') */
/* $include ('^/include/ssdata46.dat') */
/* $include ('^/include/ssdata47.dat') */
/* $include ('^/include/ssdata48.dat') */
/* $include ('^/include/ssdata49.dat') */
/* $include ('^/include/ssdata50.dat') */
/* $include ('^/include/ssdata01.dat') */
/* $include ('^/include/ssdata17.dat') */
/* $include ('^/include/ssdata18.dat') */
/* $include ('^/include/ssdata21.dat') */
/* $include ('^/include/ssdata22.dat') */
/* $include ('^/include/ssdata23.dat') */
/* $include ('^/include/ssdata28.dat') */
/* $include ('^/include/ssdata29.dat') */
/* $include ('^/include/ssdata30.dat') */
/* $include ('^/include/ssdata71.dat') */
/* $include ('^/include/sstiming.dat') */
    cw87_();
/* $include ('ssp10.dat') */
/*
-----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----C */
/* Execution of all events is performed
C
*/
/* within this loop
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
L1000:
/* WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- Communicate with p01

```

```

-----C */
receive_real_32bit__(s_gr__);
receive_real_32bit__(&s_gr__[1]);
receive_real_32bit__(&s_gr__[2]);
gr[0] = (double) s_gr__[0];
gr[1] = (double) s_gr__[1];
gr[2] = (double) s_gr__[2];
r_1 = pulsea[0];
send_real_32bit__(&r_1);
r_1 = pulsea[1];
send_real_32bit__(&r_1);
r_1 = pulsea[2];
send_real_32bit__(&r_1);
receive_real_32bit__(s_cg__);
receive_real_32bit__(&s_cg__[1]);
receive_real_32bit__(&s_cg__[2]);
cg[0] = (double) s_cg__[0];
cg[1] = (double) s_cg__[1];
cg[2] = (double) s_cg__[2];
receive_real_32bit__(s_p__);
receive_real_32bit__(s_q__);
receive_real_32bit__(s_r__);
p = (double) s_p__;
q = (double) s_q__;
r = (double) s_r__;
receive_real_32bit__(s_xd__);
receive_real_32bit__(s_yd__);
receive_real_32bit__(s_zd__);
xd = (double) s_xd__;
yd = (double) s_yd__;
zd = (double) s_zd__;
receive_real_32bit__(s_cim__);
receive_real_32bit__(&s_cim__[1]);
receive_real_32bit__(&s_cim__[2]);
receive_real_32bit__(&s_cim__[3]);
receive_real_32bit__(&s_cim__[4]);
receive_real_32bit__(&s_cim__[5]);
receive_real_32bit__(&s_cim__[6]);
receive_real_32bit__(&s_cim__[7]);
receive_real_32bit__(&s_cim__[8]);
cim[0] = (double) s_cim__[0];
cim[1] = (double) s_cim__[1];
cim[2] = (double) s_cim__[2];
cim[3] = (double) s_cim__[3];
cim[4] = (double) s_cim__[4];
cim[5] = (double) s_cim__[5];
cim[6] = (double) s_cim__[6];
cim[7] = (double) s_cim__[7];
cim[8] = (double) s_cim__[8];
receive_real_32bit__(s_pd__);
receive_real_32bit__(s_qd__);
receive_real_32bit__(s_rd__);
receive_real_32bit__(s_ud__);
receive_real_32bit__(s_vd__);
receive_real_32bit__(s_wd__);
pd = (double) s_pd__;
qd = (double) s_qd__;
rd = (double) s_rd__;
ud = (double) s_ud__;
vd = (double) s_vd__;
wd = (double) s_wd__;
/*
-----C */

```

```

/* ----- INERTIAL MEASUREMENT UPDATE
-----C */
/*
-----C */
/*          Get inertial measurement data needed
C
*/
/*          for guidance calculations .
C
*/
/*
C
*/
/*
-----C */
    if (tstep >= timudriv) {
        timudriv += timustep;
/*
-----C */
/* ----- ACCELEROMETER MODULE
-----C */
/*
-----C */
/*          Determine sensed accelerations
    C */
/*
    C */
/*
-----C */
    accel_(&t, &ud, &vd, &wd, &p, &q, &r, &pd, &qd, &rd, cg, cim, &xd,
&
        yd, &z, gr, &gyseed, qfrac, pulsea);
/*
-----C */
/* ----- TERMINATION LOGIC
-----C */
/*
-----C */
/*          Defines the simulation termination
C
*/
/*          conditions
C
*/
/*
C
*/
/*
-----C */
/*          INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
    iexit = 0;
/*          increment time */
    tstep += 1.;
    t = tstep * delt;

```

```
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/
      if (iexit == 0) {
        goto L1000;
      }
} /* MAIN__ */
```

## B.3.12 Uup11.c

```

/* uup11.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
       -lF77 -lI77 -lm -lc   (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    shortint iseq[4];
    real tvcomp, omega0[3];
    shortint imidb2;
    real tmidb2;
    shortint isk3on;
} rmguid_;

#define rmguid_1 rmguid_

struct {
    real angac1[120] /* was [3][4][10] */;
    shortint imcpas[12] /* was [3][4] */;
    real tp2end, tp3end;
    shortint ip2end;
    real tcoast;
    shortint icoast;
    real trdone;
    shortint irate, iacsb1, iacsb2, icnt, ivpfl, ivpfln;
    real thurn2, omega1[3], tlstma, aaccel[12] /* was [3][4] */;
} rmauto_;

#define rmauto_1 rmauto_

struct {
    real sw17, sw18, sw18p, sw18y, sw19, sw19p, sw19y;
    shortint iroll;
    real tpton2, tyton2, tnextp, tnexty, fltcpl, fltcyl;
} rkvault_;

#define rkvault_1 rkvault_

/* Table of constant values */

static shortint cs__0 = 0;
static shortint cs__1 = 1;

/*      PROGRAM EXOSIM */
/* -----
-C */
/* ----- Declare and initialize variables -----
-C */
/* -----
-C */
/* Main program */ MAIN__()
{
    /* System generated locals */
    real r_1;

    /* Local variables */
    static shortint acqd, icmd;
    static real lamd[2], magr, delt, fltc[4];

```

```

static shortint flip;
static real magv;
static shortint igit;
static real tsah, dtacsa_s__[4], dtacsb_s__[4], mass, rrel[3],
vrel[3],
    tsal, anvp, acslev_s__;
static shortint ivcs;
static real tgi1, dtoffv_s__[4], tofflt_s__[4];
static shortint ithres_s__;
extern /* Subroutine */ int send_signed_16bit__();
static real tburnm_s__, tapudriv, timonv_s__, tgpudriv, tapustep,
tgi1p,
    tgi2p, tgi3p, tgpustep, tgi1y, tgi2y, tgi3y, tstg2, t, tatab;
static shortint ivtab;
static real fltcp;
static shortint idist, idrop;
static real dteps, tvtab, piter, tlaps, fltcy;
static shortint iexit;
static real yawer, tdrop, urrel[3], tstep;
extern /* Subroutine */ int receive_real_32bit__();
static real tkvon, tnext, tge2a1;
static shortint iburn1, iburn2, iburn3;
static real vg[3], dtacsa[4], dtacsb[4], sp, sq, sr;
static shortint idmeas, incend;
extern /* Subroutine */ int mcguid__();
static shortint midbrn, iacson;
static real acslev, tfinal;
static shortint estate, iburnd, ivcs_s__;
static real adistt[12] /* was [4][3] */ , dtoffv[4], mgrdot,
dtvcsp[3]
    , tofflt[4];
extern /* Subroutine */ int mcauto__();
static real roller, dtvcsp[3];
static shortint iburnm, ipassm;
static real traton, tpaton, tyaton, dtsamp;
static shortint ithres;
static real tmauto, tburnm, timonv;
extern /* Subroutine */ int send_real_32bit__();
static real trmtgo;
extern /* Subroutine */ int estrel__();
static real tmguid;
static shortint idpass;
extern /* Subroutine */ int kvauto__();
static real tcwait;
extern /* Subroutine */ int vcslog__();
static real tofltm, tburnp, tburny;
extern /* Subroutine */ int acsthr2__();
static real tgoflm;
extern /* Subroutine */ int resthr__(), vcsthr2__();
static real cms[9];
extern /* Subroutine */ int cw87__();
static real vgm[3], tgo, tatab_s__, sw80;
static shortint ivtab_s__;
static real ixx, iyy, izz;
static shortint idrop_s__;
static real tvtab_s__, tgel, tge2, ti2m[9];
extern /* Subroutine */ int receive_signed_16bit__();

/*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
*/

/*      OUTPUTS */
/*      NAMELIST INPUTS */
/*      DATA INITIALIZATION */

```

```

/* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE(':pfp:INCLUDE/target.for') */
/* INITIALIZE 80x87 */
    cw87_();
/* $INCLUDE('SSpl1.DAT') */
    mcauto_(&t, &ixx, &iyy, &izz, &sp, &sq, &sr, &roller, &piter,
&yawer, &
        idist, &iacson, &iburnd, &iburnm, &idmeas, &ipassm, &icmd, &
        traton, &tpaton, &tyaton, &dt Samp, &tsal, &tsah, &tlaps,
&ithres,
        &anvp, &acslev, &tmauto, &cs__0);
    idrop_s__ = idrop;
    acslev_s__ = acslev;
    dtacsa_s__[0] = dtacsa[0];
    dtacsa_s__[1] = dtacsa[1];
    dtacsa_s__[2] = dtacsa[2];
    dtacsa_s__[3] = dtacsa[3];
    dtacsb_s__[0] = dtacsb[0];
    dtacsb_s__[1] = dtacsb[1];
    dtacsb_s__[2] = dtacsb[2];
    dtacsb_s__[3] = dtacsb[3];
    dtoffv_s__[0] = dtoffv[0];
    dtoffv_s__[1] = dtoffv[1];
    dtoffv_s__[2] = dtoffv[2];
    dtoffv_s__[3] = dtoffv[3];
    ithres_s__ = ithres;
    ivcs_s__ = 'vcs;
    ivtab_s__ = ivtab;
    tatab_s__ = tatab;
    tburnm_s__ = tburnm;
    timonv_s__ = timonv;
    tofflt_s__[0] = tofflt[0];
    tofflt_s__[1] = tofflt[1];
    tofflt_s__[2] = tofflt[2];
    tofflt_s__[3] = tofflt[3];
    tvtab_s__ = tvtab;
/*
-----
----C */
/* ----- MAIN EXECUTION LOOP
-----C */
/*
-----

```

```

-----C */
/*                               Execution of all events is performed
C
*/
/*                               within this loop
C
*/
/*
C
*/
/*
-----C */
L1000:
/*   WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- COMMUNICATION WITH P00
-----C */
    receive_real_32bit__(&ixx);
    receive_real_32bit__(&iyy);
    receive_real_32bit__(&izz);
    receive_real_32bit__(&mass);
/* ----- COMMUNICATION WITH P00
-----C */
    send_signed_16bit__(&idrop_s__);
/* ----- COMMUNICATION WITH P02
-----C */
    send_real_32bit__(&acslev_s__);
    send_real_32bit__(&dtacsa_s__);
    send_real_32bit__(&dtacsa_s__[1]);
    send_real_32bit__(&dtacsa_s__[2]);
    send_real_32bit__(&dtacsa_s__[3]);
    send_real_32bit__(&dtacsb_s__);
    send_real_32bit__(&dtacsb_s__[1]);
    send_real_32bit__(&dtacsb_s__[2]);
    send_real_32bit__(&dtacsb_s__[3]);
    send_real_32bit__(&dtoffv_s__);
    send_real_32bit__(&dtoffv_s__[1]);
    send_real_32bit__(&dtoffv_s__[2]);
    send_real_32bit__(&dtoffv_s__[3]);
    send_signed_16bit__(&ithres_s__);
    send_signed_16bit__(&ivcs_s__);
    send_signed_16bit__(&ivtab_s__);
    send_real_32bit__(&tatab_s__);
    send_real_32bit__(&tburnm_s__);
    send_real_32bit__(&timonv_s__);
    send_real_32bit__(&tofflt_s__);
    send_real_32bit__(&tofflt_s__[1]);
    send_real_32bit__(&tofflt_s__[2]);
    send_real_32bit__(&tofflt_s__[3]);
    send_real_32bit__(&tvtab_s__);
/* ----- COMMUNICATION WITH P02
-----C */
    receive_signed_16bit__(&iacson);
/* ----- COMMUNICATE WITH CORVEL -----C */
    receive_real_32bit__(vg);
    receive_real_32bit__(vg[1]);

```

```

    receive_real_32bit__(&vg[2]);
/* ----- DAISY CHAIN WITH IMUPRO AND NAVIG
-----C */
    receive_real_32bit__ (ti2m);
    receive_real_32bit__ (&ti2m[1]);
    receive_real_32bit__ (&ti2m[2]);
    receive_real_32bit__ (&ti2m[3]);
    receive_real_32bit__ (&ti2m[4]);
    receive_real_32bit__ (&ti2m[5]);
    receive_real_32bit__ (&ti2m[6]);
    receive_real_32bit__ (&ti2m[7]);
    receive_real_32bit__ (&ti2m[8]);
    receive_real_32bit__ (vrel);
    receive_real_32bit__ (&vrel[1]);
    receive_real_32bit__ (&vrel[2]);
    receive_real_32bit__ (rrel);
    receive_real_32bit__ (&rrel[1]);
    receive_real_32bit__ (&rrel[2]);
    receive_real_32bit__ (&sp);
    receive_real_32bit__ (&sq);
    receive_real_32bit__ (&sr);
    send_real_32bit__ (&magr);
    send_real_32bit__ (&magv);
    send_real_32bit__ (&tgo);
    send_real_32bit__ (&piter);
    send_real_32bit__ (&roller);
    send_real_32bit__ (&yawer);
    send_signed_16bit__ (&iburn1);
    send_real_32bit__ (lamd);
    send_real_32bit__ (&lamd[1]);
    send_signed_16bit__ (&acqd);
    receive_signed_16bit__ (&estate);
    receive_real_32bit__ (&piter);
    receive_real_32bit__ (&roller);
    receive_real_32bit__ (&yawer);
    receive_signed_16bit__ (&iburn1);
    receive_real_32bit__ (lamd);
    receive_real_32bit__ (&lamd[1]);
    receive_signed_16bit__ (&acqd);
    receive_real_32bit__ (&tge1);
    receive_real_32bit__ (&tge2a1);
    receive_real_32bit__ (&trmtgo);
/*
-----
-----C */
/*                                     ON BOARD GUIDANCE PROCESSING
C
*/
/*
-----
-----C */
/*                                     Determine guidance commands
C
*/
/*
C
*/
/*
-----
-----C */
    if (tstep >= tgpudriv) {
/*         TGPUDRIV = TGPUDRIV + TGPUSTEP */
/*
-----

```

```

-----C */
/* ----- ESTIMATED RELATIVE STATES MODULE
-----C */
/*
-----C */
/*
time-to- C */
/*
target C */
/*
C */
/*
C */
/*
-----C */
    estrel_(ti2m, cms, &estate, rrel, vrel, &magr, &magv, urrel,
&mgrdot,
        &tgo, &piter, &yawer, lamd);
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
/* ----- COMMUNICATION WITH P00
-----C */
    receive_real_32bit_(&ixx);
    receive_real_32bit_(&iyy);
    receive_real_32bit_(&izz);
    receive_real_32bit_(&mass);
/* ----- COMMUNICATION WITH P00
-----C */
    send_signed_16bit_(&idrop_s_);
/* ----- COMMUNICATION WITH P02
-----C */
    send_real_32bit_(&acslev_s_);
    send_real_32bit_(&dtacsa_s_);
    send_real_32bit_(&dtacsa_s_[1]);
    send_real_32bit_(&dtacsa_s_[2]);
    send_real_32bit_(&dtacsa_s_[3]);
    send_real_32bit_(&dtacsb_s_);
    send_real_32bit_(&dtacsb_s_[1]);
    send_real_32bit_(&dtacsb_s_[2]);
    send_real_32bit_(&dtacsb_s_[3]);
    send_real_32bit_(&dtoffv_s_);
    send_real_32bit_(&dtoffv_s_[1]);
    send_real_32bit_(&dtoffv_s_[2]);
    send_real_32bit_(&dtoffv_s_[3]);
    send_signed_16bit_(&ihres_s_);
    send_signed_16bit_(&ivcs_s_);
    send_signed_16bit_(&ivtab_s_);
    send_real_32bit_(&tatab_s_);
    send_real_32bit_(&tburnm_s_);
    send_real_32bit_(&timonv_s_);
    send_real_32bit_(&tofflt_s_);
    send_real_32bit_(&tofflt_s_[1]);
    send_real_32bit_(&tofflt_s_[2]);
    send_real_32bit_(&tofflt_s_[3]);
    send_real_32bit_(&tvtab_s_);

```

```

/* ----- COMMUNICATION WITH P02
-----C */
    receive_real_16bit_(&iacsen);
/* ----- COMMUNICATE WITH CORVEL -----C */
    receive_real_32bit_(vg);
    receive_real_32bit_(&vg[1]);
    receive_real_32bit_(&vg[2]);
/* ----- DAISY CHAIN WITH IMUPRO AND NAVIG
-----C */
    receive_real_32bit_(ti2m);
    receive_real_32bit_(&ti2m[1]);
    receive_real_32bit_(&ti2m[2]);
    receive_real_32bit_(&ti2m[3]);
    receive_real_32bit_(&ti2m[4]);
    receive_real_32bit_(&ti2m[5]);
    receive_real_32bit_(&ti2m[6]);
    receive_real_32bit_(&ti2m[7]);
    receive_real_32bit_(&ti2m[8]);
    receive_real_32bit_(vrel);
    receive_real_32bit_(&vrel[1]);
    receive_real_32bit_(&vrel[2]);
    receive_real_32bit_(rrel);
    receive_real_32bit_(&rrel[1]);
    receive_real_32bit_(&rrel[2]);
    receive_real_32bit_(&sp);
    receive_real_32bit_(&sq);
    receive_real_32bit_(&sr);
/*
-----C */
/* ----- MISSILE STATE UPDATE MODULE
-----C */
/*
-----C */
/*                                     Integrate missile states to current time
C
*/
/*
C
*/
/*
-----C */
/*
-----C */
/* ----- VCS THRUSTER RESPONSE MODULE
-----C */
/*
-----C */
/*                                     Determines the forces and moments
C
*/
/*                                     imparted by the VCS thrusters
C
*/
/*
C
*/
/*
-----C */

```



```

/*
-----C */
    if (t > tstg2 && t >= tmguid && acqd == 0) {
        mcguid_(t, ti2m, vg, urrel, &mass, &idist, &midbrn, &magr,
&magv,
            &sp, &sq, &sr, &piter, &yawer, &flip, &ivcs, &icmd, &
idmeas, &idpass, &idrop, &imcend, &iburnd, &iburnm, vgm,
            adistt, &roller, &tmguid);
    }
}
/*
-----C */
/* -----KALMAN FILTER MODULE
-----C */
/*
-----C */
    send_real_32bit__(&magr);
    send_real_32bit__(&magv);
    send_real_32bit__(&tgo);
    send_real_32bit__(&piter);
    send_real_32bit__(&roller);
    send_real_32bit__(&yawer);
    send_signed_16bit__(&iburni),
    send_real_32bit__(&lamd);
    send_real_32bit__(&lamd[1]);
    send_signed_16bit__(&acqd);
    receive_signed_16bit__(&estate);
    receive_real_32bit__(&piter);
    receive_real_32bit__(&roller);
    receive_real_32bit__(&yawer);
    receive_signed_16bit__(&iburn1);
    receive_real_32bit__(&lamd);
    receive_real_32bit__(&lamd[1]);
    receive_signed_16bit__(&acqd);
    receive_real_32bit__(&tge1);
    receive_real_32bit__(&tge2a1);
    receive_real_32bit__(&trmtgo);
/*
-----C */
/* -----Processor communication
-----C */
/*
-----C */
/* -----COMMUNICATION WITH P00
-----C */
    receive_real_32bit__(&ixx);
    receive_real_32bit__(&iyy);
    receive_real_32bit__(&izz);
    receive_real_32bit__(&mass);
/* -----COMMUNICATION WITH P00
-----C */
    send_signed_16bit__(&idrop_s__);
/* -----COMMUNICATION WITH P02
-----C */
    send_real_32bit__(&acslev_s__);
    send_real_32bit__(&dtacs_s__);
    send_real_32bit__(&dtacs_s__[1]);
    send_real_32bit__(&dtacs_s__[2]);
    send_real_32bit__(&dtacs_s__[3]);

```

```

send_real_32bit__(dtacsb_s__);
send_real_32bit__(&dtacsb_s__[1]);
send_real_32bit__(&dtacsb_s__[2]);
send_real_32bit__(&dtacsb_s__[3]);
send_real_32bit__(dtoffv_s__);
send_real_32bit__(&dtoffv_s__[1]);
send_real_32bit__(&dtoffv_s__[2]);
send_real_32bit__(&dtoffv_s__[3]);
send_signed_16bit__(&ithres_s__);
send_signed_16bit__(&ivcs_s__);
send_signed_16bit__(&ivtab_s__);
send_real_32bit__(&tatab_s__);
send_real_32bit__(&tburnm_s__);
send_real_32bit__(&timonv_s__);
send_real_32bit__(tofflt_s__);
send_real_32bit__(&tofflt_s__[1]);
send_real_32bit__(&tofflt_s__[2]);
send_real_32bit__(&tofflt_s__[3]);
send_real_32bit__(&vtvtab_s__);
/* ----- COMMUNICATION WITH P02
-----C */
    receive_signed_16bit__(&iacson);
/* ----- COMMUNICATE WITH CORVEL -----C */
    receive_real_32bit__(vg);
    receive_real_32bit__(&vg[1]);
    receive_real_32bit__(&vg[2]);
/* ----- DAISY CHAIN WITH IMUPRO AND NAVIG
-----C */
    receive_real_32bit__(ti2m);
    receive_real_32bit__(&ti2m[1]);
    receive_real_32bit__(&ti2m[2]);
    receive_real_32bit__(&ti2m[3]);
    receive_real_32bit__(&ti2m[4]);
    receive_real_32bit__(&ti2m[5]);
    receive_real_32bit__(&ti2m[6]);
    receive_real_32bit__(&ti2m[7]);
    receive_real_32bit__(&ti2m[8]);
    receive_real_32bit__(vrel);
    receive_real_32bit__(&vrel[1]);
    receive_real_32bit__(&vrel[2]);
    receive_real_32bit__(rrel);
    receive_real_32bit__(&rrel[1]);
    receive_real_32bit__(&rrel[2]);
    receive_real_32bit__(&sp);
    receive_real_32bit__(&sq);
    receive_real_32bit__(&sr);
    send_real_32bit__(&magr);
    send_real_32bit__(&magv);
    send_real_32bit__(&tgo);
    send_real_32bit__(&piter);
    send_real_32bit__(&roller);
    send_real_32bit__(&yawer);
    send_signed_16bit__(&iburn1);
    send_real_32bit__(lamd);
    send_real_32bit__(&lamd[1]);
    send_signed_16bit__(&acqd);
    receive_signed_16bit__(&estate);
    receive_real_32bit__(&piter);
    receive_real_32bit__(&roller);
    receive_real_32bit__(&yawer);
    receive_signed_16bit__(&iburn1);
    receive_real_32bit__(lamd);
    receive_real_32bit__(&lamd[1]);
    receive_signed_16bit__(&acqd);

```

```

receive_real_32bit__(&tge1);
receive_real_32bit__(&tge2a1);
receive_real_32bit__(&trmtgo);
/*
-----C */
/* ----- AUTOPILOTS
-----C */
/*
-----C */
/*
C
*/
/*
-----C */
    if (tstep >= tapudriv) {
/*
-----C */
/* ----- MIDCOURSE AUTOPILOT MODULE
-----C */
/*
-----C */
/*                                     Performs large angle reorients and
rate C */
/*                                     control during midcourse
    C */
/*
-----C */
    if (t >= tkvon) {
        if (t > tstg2 && t >= tmauto && (icmd != 0 || acqd == 0)) {
            mcauto_(&t, &ixx, &iyy, &izz, &sp, &sq, &sr, &roller,
&piter,
                &yawer, &idist, &iacson, &iburnd, &iburnm, &idmeas, &
                ipassm, &icmd, &straton, &tpaton, &tyaton, &dtsamp, &
                tsal, &tsah, &tlaps, &ithres, &anvp, &acslev, &tmauto,
                &cs_1);
        }
    }
/*
-----C */
/* ----- Processor communication
-----C */
/*
-----C */
    idrop_s__ = idrop;
/* ----- COMMUNICATION WITH P00
-----C */
    receive_real_32bit__(&ixx);
    receive_real_32bit__(&iyy);
    receive_real_32bit__(&izz);
    receive_real_32bit__(&mass);
/* ----- COMMUNICATION WITH P00
-----C */
    send_signed_16bit__(&idrop_s__);
/* ----- COMMUNICATION WITH P02
-----C */

```

```

send_real_32bit__(&acslev_s__);
send_real_32bit__(&dtacsa_s__);
send_real_32bit__(&dtacsa_s__[1]);
send_real_32bit__(&dtacsa_s__[2]);
send_real_32bit__(&dtacsa_s__[3]);
send_real_32bit__(&dtacsb_s__);
send_real_32bit__(&dtacsb_s__[1]);
send_real_32bit__(&dtacsb_s__[2]);
send_real_32bit__(&dtacsb_s__[3]);
send_real_32bit__(&dtoffv_s__);
send_real_32bit__(&dtoffv_s__[1]);
send_real_32bit__(&dtoffv_s__[2]);
send_real_32bit__(&dtoffv_s__[3]);
send_signed_16bit__(&ithres_s__);
send_signed_16bit__(&ivcs_s__);
send_signed_16bit__(&ivtab_s__);
send_real_32bit__(&tatab_s__);
send_real_32bit__(&tburnm_s__);
send_real_32bit__(&timonv_s__);
send_real_32bit__(&tofflt_s__);
send_real_32bit__(&tofflt_s__[1]);
send_real_32bit__(&tofflt_s__[2]);
send_real_32bit__(&tofflt_s__[3]);
send_real_32bit__(&tvtab_s__);
/* ----- COMMUNICATION WITH P02
-----C */
    receive_signed_16bit__(&iacson);
/* ----- COMMUNICATE WITH CORVEL -----C */
    receive_real_32bit__(&vg);
    receive_real_32bit__(&vg[1]);
    receive_real_32bit__(&vg[2]);
/* ----- DAISY CHAIN WITH IMUPRO AND NAVIG
-----C */
    receive_real_32bit__(&ti2m);
    receive_real_32bit__(&ti2m[1]);
    receive_real_32bit__(&ti2m[2]);
    receive_real_32bit__(&ti2m[3]);
    receive_real_32bit__(&ti2m[4]);
    receive_real_32bit__(&ti2m[5]);
    receive_real_32bit__(&ti2m[6]);
    receive_real_32bit__(&ti2m[7]);
    receive_real_32bit__(&ti2m[8]);
    receive_real_32bit__(&vrel);
    receive_real_32bit__(&vrel[1]);
    receive_real_32bit__(&vrel[2]);
    receive_real_32bit__(&rrel);
    receive_real_32bit__(&rrel[1]);
    receive_real_32bit__(&rrel[2]);
    receive_real_32bit__(&sp);
    receive_real_32bit__(&sq);
    receive_real_32bit__(&sr);
    send_real_32bit__(&magr);
    send_real_32bit__(&magv);
    send_real_32bit__(&tgo);
    send_real_32bit__(&piter);
    send_real_32bit__(&roller);
    send_real_32bit__(&yawer);
    send_signed_16bit__(&iburn1);
    send_real_32bit__(&lamd);
    send_real_32bit__(&lamd[1]);
    send_signed_16bit__(&acqd);
    receive_signed_16bit__(&estate);
    receive_real_32bit__(&piter);
    receive_real_32bit__(&roller);

```

```

receive_real_32bit__(&yawer);
receive_signed_16bit__(&iburn1);
receive_real_32bit__(lamd);
receive_real_32bit__(&lamd[1]);
receive_signed_16bit__(&acqd);
receive_real_32bit__(tgel);
receive_real_32bit__(tge2al);
receive_real_32bit__(trmtgo);

/*
-----C */
/* ----- AUTOPILOTS
-----C */
/*
-----C */
/*
C
*/
/*
-----C */
    if (tstep >= tapudriv) {
        if (t >= tkvon) {
/*
-----C */
/* ----- KV AUTOPILOT MODULE
-----C */
/*
-----C */
/*
C */
/*
C */
/*
flight.    C */
/*
C */
/*
and at    C */
/*
C */
/*
C */
/*
-----C */
    kvauto_(&t, &sp, &sq, &sr, &fltcp, &fltcy, &ixx, &iyy, &izz,
&adistt, &roller, &spiter, &yawer, &tcwait, &idist, &sw80,
&tsal, &tsah, &tnext, &tlaps, &anvp, &dtsamp, &sacslev, &
traton, &tpaton, &tyaton, &ithres);
}
}
/* ----- COMMUNICATION WITH P00
-----C */
    receive_real_32bit__(ixx);
    receive_real_32bit__(iyy);
    receive_real_32bit__(izz);
    receive_real_32bit__(mass);
/* ----- COMMUNICATION WITH P00
-----C */

```

```

    send_signed_16bit__(&idrop_s__);
/* ----- COMMUNICATION WITH P02
-----C */
    send_real_32bit__(&acslev_s__);
    send_real_32bit__(&dtacsa_s__);
    send_real_32bit__(&dtacsa_s__[1]);
    send_real_32bit__(&dtacsa_s__[2]);
    send_real_32bit__(&dtacsa_s__[3]);
    send_real_32bit__(&dtacsb_s__);
    send_real_32bit__(&dtacsb_s__[1]);
    send_real_32bit__(&dtacsb_s__[2]);
    send_real_32bit__(&dtacsb_s__[3]);
    send_real_32bit__(&dtoffv_s__);
    send_real_32bit__(&dtoffv_s__[1]);
    send_real_32bit__(&dtoffv_s__[2]);
    send_real_32bit__(&dtoffv_s__[3]);
    send_signed_16bit__(&ithres_s__);
    send_signed_16bit__(&ivcs_s__);
    send_signed_16bit__(&ivtab_s__);
    send_real_32bit__(&tatab_s__);
    send_real_32bit__(&tburnm_s__);
    send_real_32bit__(&timonv_s__);
    send_real_32bit__(&tofflt_s__);
    send_real_32bit__(&tofflt_s__[1]);
    send_real_32bit__(&tofflt_s__[2]);
    send_real_32bit__(&tofflt_s__[3]);
    send_real_32bit__(&tvtab_s__);
/* ----- COMMUNICATION WITH P02
-----C */
    receive_signed_16bit__(&iacson);
/* ----- COMMUNICATE WITH CORVEL -----C */
    receive_real_32bit__(vg);
    receive_real_32bit__(&vg[1]);
    receive_real_32bit__(&vg[2]);
/* ----- DAISY CHAIN WITH IMUPRO AND NAVIG
-----C */
    receive_real_32bit__(ti2m);
    receive_real_32bit__(&ti2m[1]);
    receive_real_32bit__(&ti2m[2]);
    receive_real_32bit__(&ti2m[3]);
    receive_real_32bit__(&ti2m[4]);
    receive_real_32bit__(&ti2m[5]);
    receive_real_32bit__(&ti2m[6]);
    receive_real_32bit__(&ti2m[7]);
    receive_real_32bit__(&ti2m[8]);
    receive_real_32bit__(vrel);
    receive_real_32bit__(&vrel[1]);
    receive_real_32bit__(&vrel[2]);
    receive_real_32bit__(rrel);
    receive_real_32bit__(&rrel[1]);
    receive_real_32bit__(&rrel[2]);
    receive_real_32bit__(sp);
    receive_real_32bit__(sq);
    receive_real_32bit__(sr);
    send_real_32bit__(magr);
    send_real_32bit__(magv);
    send_real_32bit__(tgo);
    send_real_32bit__(piter);
    send_real_32bit__(rroller);
    send_real_32bit__(yawer);
    send_signed_16bit__(&iburn1);
    send_real_32bit__(lamd);
    send_real_32bit__(&lamd[1]);
    send_signed_16bit__(&acqd);

```

```

receive_signed_16bit__(&estate);
receive_real_32bit__(&piter);
receive_real_32bit__(&roller);
receive_real_32bit__(&yawer);
receive_signed_16bit__(&iburn1);
receive_real_32bit__(lamd);
receive_real_32bit__(lamd[1]);
receive_signed_16bit__(&acqd);
receive_real_32bit__(&tge1);
receive_real_32bit__(&tge2al);
receive_real_32bit__(&trmtgo);

/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/* C
*/
/* -----C */
if (tstep >= tapudriv) {
    tapudriv += tapustep;
    if (t >= tkvon) {
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/* Controls the kill vehicle velocity
by      C */
/* determining the appropriate VCS
thruster C */
/* on and off times.
/* C */
/* C */
/* -----C */
vcslog_(&t, &mass, lamd, &tgo, &magv, &tgil, &trmtgo, &tge2al,
&
    tgel, vgm, &ivcs, &idmeas, &iburnm, &midbrn, &iburn1, &
    iburn2, &iburn3, &idist, fltc, &fltcp, &fltcy, &tsal, &
    tsah, tofflt, &tofltm, &tburnp, &tburny, &tge2, &tgilp,
&
    tgi2p, &tgi3p, &tgily, &tgi2y, &tgi3y, &timonv, &tgoflm,
&
    tcwait, dtvcsp, dtvcsy, dtoffv, &tburnm);
/* SET FLAG TO COMPUTE VCS THRUSTER RESPONSE TABLE */
    ivtab = 1;
    tvtab - t;
/*
-----C */
/* -----C */
/* -----C */

```

```

/*
-----C */
/*
    C */
    if (ithres == 1) {
        resthr_(&t, &idist, &anvp, &dt Samp, &tofltm, &traton,
&tpaton,
        &tyaton, dtacsa, dtacsb);
/*
        BEGINNING TIME OF ACS THRUSTER RESPONSE TABLE
*/
        tatab = t;
    }
}
}
ithres_s__ = ithres;
acslev_s__ = acslev;
dtacsa_s__[0] = dtacsa[0];
dtacsa_s__[1] = dtacsa[1];
dtacsa_s__[2] = dtacsa[2];
dtacsa_s__[3] = dtacsa[3];
dtacsb_s__[0] = dtacsb[0];
dtacsb_s__[1] = dtacsb[1];
dtacsb_s__[2] = dtacsb[2];
dtacsb_s__[3] = dtacsb[3];
dtoffv_s__[0] = dtoffv[0];
dtoffv_s__[1] = dtoffv[1];
dtoffv_s__[2] = dtoffv[2];
dtoffv_s__[3] = dtoffv[3];
ivcs_s__ = ivcs;
ivtab_s__ = ivtab;
tatab_s__ = tatab;
tburnm_s__ = tburnm;
timonv_s__ = timonv;
tofflt_s__[0] = tofflt[0];
tofflt_s__[1] = tofflt[1];
tofflt_s__[2] = tofflt[2];
tofflt_s__[3] = tofflt[3];
tvtab_s__ = tvtab;
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/*
    Defines the simulation termination
    conditions
*/
/*
    INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
iexit = 0;
/*
    ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
*/

```

```
/*      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED */
  if (t >= tfinal) {
    iexit = 1;
  }
/*      increment time */
  tstep += (float)5.;
  t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

  if (iexit == 0) {
    goto L1000;
  }
} /* MAIN__ */
```

## B.3.13 Uup12.c

```

/* uup12.f -- translated by f2c (version of 3 February 1990  3:36:42).
   You must link the resulting object file with the libraries:
      -lF77 -lI77 -lm -lc      (in that order)
*/

#include "f2c.h"

/* Common Block Declarations */

struct {
    real tkf;
    shortint idrtok;
    real pp11, pp12, pp22, py11, py12, py22, plmdfp, ylmdfp, plamh,
    ylamh,
        plamdh, ylamdh, plamdf, ylamdf, tgil;
    shortint kfmode, ifpas;
} rkalmn_;

#define rkalmn_1 rkalmn_

/* Table of constant values */

static doublereal c_b2 = -.29912;

/* ----- PROGRAM EXOSIM ----- */
/* ----- Declare and initialize variables ----- */
/* ----- MAIN ----- */
/* Main program */ MAIN__()
{
    /* System generated locals */
    doublereal d_1;

    /* Builtin functions */
    double pow_dd();

    /* Local variables */
    static shortint acqd;
    static real lamd[2];
    static shortint macq;
    static real asig, magr, lamm[2], delt, magv, racq, dtacsa_s__[4],
        dtacsb_s__[4], rrel[3], vrel[3];
    static shortint term, ivcs, mcso;
    static real acslev_s__, dtoffv_s__[4], tofflt_s__[4];
    static shortint ithres_s__;
    extern /* Subroutine */ int send_signed_16bit__();
    static real tburnm_s__, tkfudriv, timonv_s__, t, tatab;
    static shortint track, ivcab, idrop;
    static real tvtab;
    static shortint iexit;
    static real piter;
    static shortint mterm;
    static real wfilt, yawer, zfilt, tstep;
    extern /* Subroutine */ int receive_real_32bit__();
    static real tge2al;
    static shortint iburnl;
    static real dtacsa[4], dtacsb[4];
    extern /* Subroutine */ int kalman_();

```

```

static real lamsek[2], acslev, tfinal;
static shortint estate, ivcs_s__;
static real snracq, dtoffv[4], magrtr, lamdxx[2], tofflt[4];
static shortint ithres;
static real frmrat, tburnm;
static shortint ireslv;
static real roller;
extern /* Subroutine */ int send_real_32bit__();
static real timonv, trmtgo;
static shortint sektyp;
static real lam[2], cms[9];
extern /* Subroutine */ int cw87__();
static real tatab_s__, tgo, snr;
static shortint ivtab_s__, idrop_s__;
static real tvtab_s__, tgel, ti2m[9];
extern /* Subroutine */ int receive_signed_16bit__();

/*      THE FOLLOWING COMMON BLOCKS USED FOR MIDFLIGHT CAPABILITIES ONLY
*/

/*      OUTPUTS */
/*      NAMELIST INPUTS */
/* DATA INITIALIZATION */
/* $INCLUDE('^/INCLUDE/SSDATA35.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA38.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA39.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA42.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA44.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA45.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA46.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA47.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA48.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA49.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA50.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA01.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA17.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA18.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA21.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA22.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA23.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA28.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA29.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA30.DAT') */
/* $INCLUDE('^/INCLUDE/SSDATA71.DAT') */
/* $INCLUDE('^/INCLUDE/SSTIMING.DAT') */
/* $INCLUDE(':pfp:INCLUDE/target.for') */
/* INITIALIZE 80x87 */
  cw87__();
/* $INCLUDE('SSp12.DAT') */
  idrop_s__ = idrop;
  acslev_s__ = acslev;
  dtacsa_s__[0] = dtacsa[0];
  dtacsa_s__[1] = dtacsa[1];
  dtacsa_s__[2] = dtacsa[2];
  dtacsa_s__[3] = dtacsa[3];
  dtacsb_s__[0] = dtacsb[0];
  dtacsb_s__[1] = dtacsb[1];
  dtacsb_s__[2] = dtacsb[2];
  dtacsb_s__[3] = dtacsb[3];
  dtoffv_s__[0] = dtoffv[0];
  dtoffv_s__[1] = dtoffv[1];
  dtoffv_s__[2] = dtoffv[2];
  dtoffv_s__[3] = dtoffv[3];
  ithres_s__ = ithres;

```

```

ivcs_s__ = ivcs;
ivtab_s__ = ivtab;
tatab_s__ = tatab;
tburnm_s__ = tburnm;
timonv_s__ = timonv;
tofflt_s__[0] = tofflt[0];
tofflt_s__[1] = tofflt[1];
tofflt_s__[2] = tofflt[2];
tofflt_s__[3] = tofflt[3];
tvtab_s__ = tvtab;
/*
-----C */
/* -----C */
/* -----C */
/* -----C */
/* Execution of all events is performed
C
*/
/* within this loop
C
*/
C
*/
C
*/
/*
-----C */
L1000:
/* WRITE(*,*) '-----BEGINNING OF LOOP-----' */
/*
-----C */
/* -----C */ Processor communication
/* -----C */
/*
-----C */
/* -----C */ COMMUNICATION WITH SEEKER
/* -----C */
receive_real_32bit__(lammm);
receive_real_32bit__(&lammm[1]);
receive_real_32bit__(&snr);
receive_real_32bit__(&frmrat);
/* -----C */ COMMUNICATION WITH P03
/* -----C */
receive_signed_16bit__(&ireslv);
receive_real_32bit__(lamdxx);
receive_real_32bit__(&lamdxx[1]);
receive_real_32bit__(lamsek);
receive_real_32bit__(&lamsek[1]);
receive_real_32bit__(&magrtr);
/* -----C */ DAISY CHAIN WITH IMUPRO AND NAVIG
/* -----C */
receive_real_32bit__(ti2m);
receive_real_32bit__(&ti2m[1]);
receive_real_32bit__(&ti2m[2]);
receive_real_32bit__(&ti2m[3]);
receive_real_32bit__(&ti2m[4]);
receive_real_32bit__(&ti2m[5]);
receive_real_32bit__(&ti2m[6]);
receive_real_32bit__(&ti2m[7]);

```

```

receive_real_32bit__(&ti2m[8]);
receive_real_32bit__(vrel);
receive_real_32bit__(&vrel[1]);
receive_real_32bit__(&vrel[2]);
receive_real_32bit__(rrel);
receive_real_32bit__(&rrel[1]);
receive_real_32bit__(&rrel[2]);
/*
-----C */
/* ----- KALMAN FILTER MODULE
-----C */
/*
-----C */
/*                               Filter LOS angles
C
*/
/*
C
*/
/*
-----C */
receive_real_32bit__(&magr);
receive_real_32bit__(&magv);
receive_real_32bit__(&tgo);
receive_real_32bit__(&piter);
receive_real_32bit__(&roller);
receive_real_32bit__(&yawer);
receive_signed_16bit__(&iburn1);
receive_real_32bit__(lamd);
receive_real_32bit__(&lamd[1]);
receive_signed_16bit__(&acqd);
if (tstep >= tkfudriv) {
/*      TKFUDRIV = TKFUDRIV + TKFUSTEP */
      tkfudriv += (shortint) ((float)1e3 / frmrat);
/*      write(message,103)t */
/* 103      format(' kalman',f10.4) */
/*      call outmes(message) */
/*      CALL FILTER IF SNR IS SUFFICIENT */
      if (snr >= snracq || sektyp != 2) {
        if (sektyp == 1 || sektyp == 2) {
          d_1 = (double) snr;
          asig = pow_dd(&d_1, &c_b2) * (float)32.56 * (float)1e-6;
        }
        kalman_(&t, ti2m, lamm, &asig, &snr, &tgo, rrel, vrel, ti2m, &
          racq, &magrtr, &magr, &magv, lamsek, lamdxx, &frmrat,
cms,
          &macq, &mcs0, &mterm, &ireslv, &track, &term, &trmtgo,
&
          tge1, &tge2a1, &wfilt, &zfilt, lam, lamd, &iburn1,
&acqa,
          &estate, &piter, &yawer, &roller);
      }
    }
    send_signed_16bit__(&estate);
    send_real_32bit__(&piter);
    send_real_32bit__(&roller);
    send_real_32bit__(&yawer);
    send_signed_16bit__(&iburn1);
    send_real_32bit__(lamd);
    send_real_32bit__(&lamd[1]);
    send_signed_16bit__(&acqd);

```

```

send_real_32bit__(&tge1);
send_real_32bit__(&tge2a1);
send_real_32bit__(&trmtgo);
/*
-----C */
/* ----- TERMINATION LOGIC
-----C */
/*
-----C */
/*                                     Defines the simulation termination
C
*/
/*                                     conditions
C
*/
/*
C
*/
/*
C
*/
/*
-----C */
/*      INITIALIZE SIMULATION EXIT FLAG TO ZERO ( PREVENTS EXIT ) */
iexit = 0;
/*      ENABLE EXIT IF MAXIMUM SIMULATION TIME HAS BEEN EXECUTED AND ALL
*/
/*      EVENTS SCHEDULED FOR THIS TIME HAVE BEEN EXECUTED */
if (t >= tfinal) {
    iexit = 1;
}
/*      increment time */
tstep += (float)1.;
t = tstep * delt;
/*      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
*/

    if (iexit == 0) {
        goto L1000;
    }
} /* MAIN */

```

A.4 Crossbar Code

```

#define relay y3
ssp00 is      ssp00.fpp      on x1
ssp01 is      ssp01.fpx      on x15
ssp02 is      ssp02.fpp      on x2
ssp03 is      ssp03.fpx      on x14
ssp04 is      ssp04.fpx      on x13
ssp05 is      ssp05.fpp      on x5
ssp06 is      ssp06.fpp      on x6
ssp07 is      ssp07.fpp      on x7
ssp08 is      ssp08.fpx      on y13
ssp09 is      ssp09.fpx      on y14
ssp10 is      ssp10.fpx      on x11
ssp11 is      ssp11.386      on x10
ssp12 is      ssp12.fpp      on x12

timer is      fpptimer.fpp    on y15
print is      print.386       on p23

loop

cycle
  ssp01,ssp10 := ssp08.2; [REAL GR(01) ]
  [ relay,ssp01,ssp10 := ssp08.2; REAL GR(01) ]

cycle
  ssp01,ssp10 := ssp08.2; [ REAL GR(02) ]
  print := ssp00.2; [REAL PHI]

cycle
  ssp01,ssp10 := ssp08.2; [ REAL GR(03) ]
  print := ssp00.2; [REAL THT]

cycle
  ssp11 := ssp00.2; [ REAL*8 IXX ]
  print := ssp01.2; [ T ]

cycle
  ssp11 := ssp00.2; [ REAL*8 IYY ]
  print := ssp08.2; [REAL ALT]

cycle
  ssp11 := ssp00.2; [ REAL*8 IZZ ]

cycle
  ssp01,ssp11,ssp06,ssp08 := ssp00.2; [ REAL*8 MASS ]

cycle
  ssp01 := ssp10.2; [ REAL*8 PULSEA(01) ]
  print := ssp00.2; [REAL PSI]

cycle
  ssp01 := ssp10.2; [ REAL*8 PULSEA(02) ]

cycle
  ssp01 := ssp10.2; [ REAL*8 PULSEA(03) ]

cycle
  ssp02,ssp10,ssp05 := ssp06.2; [ cg(1) ]
  ssp12 := ssp09.2; [ lamm(1) ]

cycle

```

```

ssp01 := ssp04.2; [ REAL*8 PULSESEG(01) ]
ssp02,ssp10,ssp05 := ssp06.2; [ cg(2) ]
ssp12 := ssp09.2; [ lamm(2) ]

cycle
  ssp01 := ssp04.2; [ REAL*8 PULSESEG(02) ]
  ssp02,ssp10,ssp05 := ssp06.2; [ cg(3) ]
  ssp12 := ssp09.2; [ snr ]

cycle
  ssp01 := ssp04.2; [ REAL*8 PULSESEG(03) ]
  ssp12 := ssp09.2; [ frmrat ]

cycle
  ssp01 := ssp08.4; [ REAL*8 XYZE(01) ]
  ssp04,ssp10 := ssp00.2; [ REAL*8 P ]

cycle
  ssp01 := ssp08.4; [ REAL*8 XYZE(02) ]
  ssp03,ssp04,ssp10 := ssp00.2; [ REAL*8 Q ]

cycle
  ssp01 := ssp08.4; [ REAL*8 XYZE(03) ]
  ssp03,ssp04,ssp10 := ssp00.2; [ REAL*8 R ]

cycle
  ssp01 := ssp08.4; [ REAL*8 XYZED(01) ]

cycle
  ssp01 := ssp08.4; [ REAL*8 XYZED(02) ]

cycle
  ssp01 := ssp08.4; [ REAL*8 XYZED(03) ]

cycle
  ssp01,ssp03 := ssp08.4; [ REAL*8 X ]

cycle
  ssp01,ssp03 := ssp08.4; [ REAL*8 Y ]

cycle
  ssp01,ssp03 := ssp08.4; [ REAL*8 Z ]

cycle
  ssp07,print := ssp08.2; [ REAL X ]

cycle
  ssp07,print := ssp08.2; [ REAL Y ]

cycle
  ssp07,print := ssp08.2; [ REAL Z ]

cycle
  ssp00,ssp01,ssp03,ssp10,ssp07,print := ssp08.2; [ REAL XD ]

cycle
  ssp00,ssp01,ssp03,ssp10,ssp07,print := ssp08.2; [ REAL YD ]

cycle
  ssp00,ssp01,ssp03,ssp10,ssp07,print := ssp08.2; [ REAL ZD ]

cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(1) ]

```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(2) ]
```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(3) ]
```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(4) ]
```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(5) ]
```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(6) ]
```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(7) ]
```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(8) ]
```

```
cycle
  ssp03,ssp04,ssp10,ssp08 := ssp00.2; [ REAL*8 CIM(9) ]
```

```
cycle
  ssp00 := ssp11.1; [ INTEGER IDROP ]
```

```
cycle
  ssp00 := ssp02.2; [ mdotV ]
```

```
cycle
  ssp00 := ssp05.2; [ mdota ]
  ssp08 := ssp02.2; [ fxvcs ]
```

```
cycle
  ssp08 := ssp02.2; [ fyvcs ]
  ssp07 := ssp01.2; [ REAL*8 RMIR(1) ]
```

```
cycle
  ssp08 := ssp02.2; [ fzvcs ]
  ssp07 := ssp01.2; [ REAL*8 RMIR(2) ]
```

```
cycle
  ssp00 := ssp02.2; [ mxvcs ]
  ssp07 := ssp01.2; [ REAL*8 RMIR(3) ]
```

```
cycle
  ssp00 := ssp02.2; [ myvcs ]
  ssp07 := ssp01.2; [ REAL*8 VMIR(1) ]
```

```
cycle
  ssp00 := ssp02.2; [ mzvcs ]
  ssp07 := ssp01.2; [ REAL*8 VMIR(2) ]
```

```
cycle
  ssp08 := ssp05.2; [ fxacs ]
  ssp07 := ssp01.2; [ REAL*8 VMIR(3) ]
```

```
cycle
  ssp08 := ssp05.2; [ fyacs ]
  ssp01 := ssp03.4; [ REAL*8 GRT(01,01) ]
```

```
cycle
  ssp08 := ssp05.2; [ fzacs ]
```

```

ssp01 := ssp03.4; [ REAL*8 GRT(01,02) ]

cycle
  ssp00 := ssp05.2; [ mxacs ]
  ssp01 := ssp03.4; [ REAL*8 GRT(01,03) ]

cycle
  ssp00 := ssp05.2; [ myacs ]
  ssp12 := ssp03.1; [ INTEGER IRESLV ]

cycle
  ssp00 := ssp05.2; [ mzacs ]
  ssp12 := ssp03.2; [ REAL*8 LAMDXX(01) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 ACSLEV ]
  ssp12 := ssp03.2; [ REAL*8 LAMDXX(02) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSA(01) ]
  ssp12,ssp09 := ssp03.2; [ REAL*8 LAMSEK(01) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSA(02) ]
  ssp12,ssp09 := ssp03.2; [ REAL*8 LAMSEK(02) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSA(03) ]
  ssp12,ssp09,print := ssp03.2; [ REAL*8 MAGRTR ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSA(04) ]
  ssp01,print := ssp03.4; [ REAL*8 RTIC(01,01) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSB(01) ]
  ssp01,print := ssp03.4; [ REAL*8 RTIC(01,02) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSB(02) ]
  ssp01,print := ssp03.4; [ REAL*8 RTIC(01,03) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSB(03) ]
  ssp01 := ssp03.4; [ REAL*8 VTIC(01,01) ]

cycle
  ssp05 := ssp11.2; [ REAL*8 DTACSB(04) ]
  ssp01 := ssp03.4; [ REAL*8 VTIC(01,02) ]

cycle
  ssp02 := ssp11.2; [ REAL*8 DTOFFV(01) ]
  ssp01 := ssp03.4; [ REAL*8 VTIC(01,03) ]

cycle
  ssp02 := ssp11.2; [ REAL*8 DTOFFV(02) ]
  ssp10 := ssp00.2; [ REAL*8 PD ]
  ssp07 := ssp01.2; [ REAL*8 AT(1) ]

cycle
  ssp02 := ssp11.2; [ REAL*8 DTOFFV(03) ]
  ssp10 := ssp00.2; [ REAL*8 QD ]
  ssp07 := ssp01.2; [ REAL*8 AT(2) ]

```

```

cycle
  ssp02 := ssp11.2; [ REAL*8 DTOFFV(04) ]
  ssp10 := ssp00.2; [ REAL*8 RD ]
  ssp07 := ssp01.2; [ REAL*8 AT(3) ]

cycle
  ssp05 := ssp11.1; [ INTEGER ITHRES ]

cycle
  ssp02 := ssp11.1; [ INTEGER IVCS ]

cycle
  ssp02 := ssp11.1; [ INTEGER IVTAB ]

cycle
  ssp05 := ssp11.2; [ REAL*8 TATAB ]

cycle
  ssp02 := ssp11.2; [ REAL*8 TBURNM ]

cycle
  ssp02 := ssp11.2; [ REAL*8 TIMONV ]

cycle
  ssp02 := ssp11.2; [ REAL*8 TOFFLT(01) ]

cycle
  ssp02 := ssp11.2; [ REAL*8 TOFFLT(02) ]

cycle
  ssp02 := ssp11.2; [ REAL*8 TOFFLT(03) ]

cycle
  ssp02 := ssp11.2; [ REAL*8 TOFFLT(04) ]

cycle
  ssp02 := ssp11.2; [ REAL*8 TVTAB ]

cycle
  ssp11 := ssp05.1; [ INTEGER IACSON ]

cycle
  ssp10 := ssp08.2; [ REAL*8 UD ]
  ssp11 := ssp07.2; [ REAL*8 VG(1) ]

cycle
  ssp10 := ssp08.2; [ REAL*8 VD ]
  ssp11 := ssp07.2; [ REAL*8 VG(2) ]

cycle
  ssp10 := ssp08.2; [ REAL*8 WD ]
  ssp11 := ssp07.2; [ REAL*8 VG(3) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(1) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(2) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(3) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(4) ]

```

```

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(5) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(6) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(7) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(8) ]

cycle
  ssp11,ssp12 := ssp01.2; [ ti2m(9) ]

cycle
  ssp11,ssp12 := ssp01.2; [ VREL(1) ]

cycle
  ssp11,ssp12 := ssp01.2; [ VREL(2) ]

cycle
  ssp11,ssp12 := ssp01.2; [ VREL(3) ]

cycle
  ssp11,ssp12 := ssp01.2; [ RREL(1) ]

cycle
  ssp11,ssp12 := ssp01.2; [ RREL(2) ]

cycle
  ssp11,ssp12 := ssp01.2; [ RREL(3) ]

cycle
  ssp11 := ssp01.2; [ sp ]

cycle
  ssp11 := ssp01.2; [ sq ]

cycle
  ssp11 := ssp01.2; [ sr ]

cycle
  ssp12 := ssp11.2; [ magr ]

cycle
  ssp12 := ssp11.2; [ magv ]

cycle
  ssp12,print := ssp11.2; [ tgo ]

cycle
  ssp12 := ssp11.2; [ piter ]

cycle
  ssp12 := ssp11.2; [ roller ]

cycle
  ssp12 := ssp11.2; [ yawer ]

cycle
  ssp12 := ssp11.1; [ iburn1 ]

```

```
cycle
  ssp12 := ssp11.2; [ lamd(1) ]

cycle
  ssp12 := ssp11.2; [ lamd(2) ]

cycle
  ssp12 := ssp11.1; [ acqd ]

cycle
  ssp11 := ssp12.1; [ estate ]

cycle
  ssp11 := ssp12.2; [ piter ]

cycle
  ssp11 := ssp12.2; [ roller ]

cycle
  ssp11 := ssp12.2; [ yawer ]

cycle
  ssp11 := ssp12.1; [ iburn1 ]

cycle
  ssp11 := ssp12.2; [ lamd(1) ]

cycle
  ssp11 := ssp12.2; [ lamd(2) ]

cycle
  ssp11 := ssp12.1; [ acqd ]

cycle
  ssp11 := ssp12.2; [ tge1 ]

cycle
  ssp11 := ssp12.2; [ tge2a1 ]

cycle
  ssp11 := ssp12.2; [ trmtgo ]
```