

2

AD-A241 693



ANNUAL REPORT

VOLUME 1

PART 3

TASK 1: DIGITAL EMULATION TECHNOLOGY LABORATORY

REPORT NO. AR-0142-91-001

September 27, 1991

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

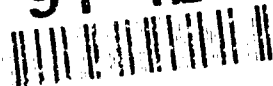
Atlanta, Georgia 30332 - 0540

Contract Data Requirements List Item A005

Period Covered: FY 91

Type Report: Annual

91-12566



DISCLAIMER STATEMENT - 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.

- (1) **DISTRIBUTION STATEMENT** - Approved for public release; distribution is unlimited.
- (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
N. 62841 N
D. 701
G. 701
Justification

By
D. 701
G. 701
Justification

A-1

ANNUAL REPORT

VOLUME 1

PART 3

TASK 1: DIGITAL EMULATION TECHNOLOGY LABORATORY

September 27, 1991

Authors

Thomas R. Collins and Stephen R. Wachtel

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

PART 1

1. Introduction	1
1.1. Objectives	1
1.1.1. GN&C Test and Evaluation -- EXOSIM.....	3
1.1.2. Education and Technical Support	4
1.2. Schedules and milestones	5
2. Hardware and Facilities.....	9
2.1. Parallel Function Processor (PFP)	9
2.1.1. Physical Description.....	10
2.1.2. Intel 310 Host.....	12
2.1.3. Sun 386i Host	12
2.2. Seeker Scene Emulator (SSE).....	12
2.3. Other computer systems	13
2.4. Secure laboratory	14
3. FPP/FPX Development Tools	16
3.1. Introduction.....	16
3.2. FPP/FPX object module loader	16
3.3. FPP/FPX program downloader.....	17
4. Software Development Tools	18
4.1. Introduction.....	18
4.2. Sequential programming tools	18
4.2.1. INITIAL program.....	18
4.2.2. DECLARE program	21
4.2.3. STRUCTURE program	25
4.2.4. CTIMER program	27

4.3. Parallel programming tools	31
4.3.1. NETWORK program.....	31
4.3.2. USAGE program.....	37
4.3.3. ETIMER program	41
4.4. Special purpose tools.....	45
4.4.1. NAMELIST program	45
4.4.2. EQUIVALENCE program.....	46
4.4.3. COMMON program.....	49
4.4.4. PROLOG utility.....	51
5. Application Software.....	63
5.1. EXOSIM.....	63
5.1.1. EXOSIM 1.0.....	65
5.1.2. EXOSIM 2.0.....	66
5.1.2.1. SSV19.3.....	70
5.1.2.2. SSV19.5.....	71
5.1.2.3. SSV19.6.....	74
5.1.2.4. SSV19.7 and SSV19.8	76
5.1.2.5. SSV20.8.....	77
5.1.2.6. SSV20.9.....	78
5.1.2.7. SSV20.10.....	80
5.1.2.8. SSV20.11.....	81
5.1.2.9. SSV20.12.....	85
5.1.2.10. SSV20.13.....	88
5.1.2.11. SSV20.14.....	91
5.1.2.12. SSV20.15.....	94
5.1.2.13. SSV20.16.....	98
5.1.2.14. SSV21.16.....	102
5.1.2.15. SSV22.16.....	103
5.1.2.16. SSV22.19.....	108
5.2. LEAP.....	114
6. Appendix A: Environment file format	117
7. Appendix B: vield program source.....	119
8. Appendix C: loadfpp program source.....	135

PART 2

9. Appendix D: common program source	1
10. Appendix E: ctimer program source	43
11. Appendix F: declare program source	109
12. Appendix G: equivalence program source	169
13. Appendix H: etimer program source.....	213
14. Appendix I: initial program source	285
15. Appendix J: namelist program source	328
16. Appendix K: network program source.....	343
17. Appendix L: structure program source.....	393
18. Appendix M: usage program source	426

PART 3

19. Appendix N: EXOSIM 2.0 (End-to-end).....	1
--	---

19. Appendix N: EXOSIM 2.0 (End-to-end)

FILE: uuv22.19g/debug/makefile

```
FORFLAGS = code large optimize(3) storage(integer*2)
DUTILITY = ^/DUTILITY/UTILITY.LIB
SUTILITY = ^/SUTILITY/UTILITY.LIB
```

```
PROGRAM = \
  SSBLK00.BL \
  SSBLK01.BL \
  SSBLK02.BL \
  SSBLK03.BL \
  SSBLK04.BL \
  SSBLK05.BL \
  SSBLK06.BL \
  SSBLK07.BL \
  SSBLK08.BL \
  SSBLK09.BL \
  SSBLK10.BL \
  SSBLK11.BL \
  SSBLK12.BL \
  SSBLK13.BL \
  SSBLK14.BL \
  SSBLK15.BL \
  SSBLK16.BL \
  SSBLK17.BL \
  SSBLK18.BL \
  CROSSBAR.BL \
  SEQUENCER.BL
```

```
default: $(PROGRAM)
```

```
SSBLK00.BL: UUBLK00.OBJ $(DUTILITY)
  submit :PFP:csd/forbldnew( SSBLK00, 'UUBLK00.OBJ,$(DUTILITY)', notype )
```

```
SSBLK01.BL: UUBLK01.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK01, 'UUBLK01.OBJ,$(SUTILITY)', notype )
```

```
SSBLK02.BL: UUBLK02.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK02, 'UUBLK02.OBJ,$(SUTILITY)', notype )
```

```
SSBLK03.BL: UUBLK03.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK03, 'UUBLK03.OBJ,$(SUTILITY)', notype )
```

```
SSBLK04.BL: UUBLK04.OBJ $(DUTILITY)
  submit :PFP:csd/forbldnew( SSBLK04, 'UUBLK04.OBJ,$(DUTILITY)', notype )
```

```
SSBLK05.BL: UUBLK05.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK05, 'UUBLK05.OBJ,$(SUTILITY)', notype )
```

```
SSBLK06.BL: UUBLK06.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK06, 'UUBLK06.OBJ,$(SUTILITY)', notype )
```

```
SSBLK07.BL: UUBLK07.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK07, 'UUBLK07.OBJ,$(SUTILITY)', notype )
```

```
SSBLK08.BL: UUBLK08.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK08, 'UUBLK08.OBJ,$(SUTILITY)', notype )
```

```
SSBLK09.BL: UUBLK09.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK09, 'UUBLK09.OBJ,$(SUTILITY)', notype )
```

```
SSBLK10.BL: UUBLK10.OBJ $(SUTILITY)
  submit :PFP:csd/forbldnew( SSBLK10, 'UUBLK10.OBJ,$(SUTILITY)', notype )
```

```

SSBLK11.BL:UUBLK11.OBJ $(DUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK11, 'UUBLK11.OBJ,$(DUTILITY)', notype )

SSBLK12.BL:UUBLK12.OBJ $(DUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK12, 'UUBLK12.OBJ,$(DUTILITY)', notype )

SSBLK13.BL:UUBLK13.OBJ $(SUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK13, 'UUBLK13.OBJ,$(SUTILITY)', notype )

SSBLK14.BL:UUBLK14.OBJ $(SUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK14, 'UUBLK14.OBJ,$(SUTILITY)', notype )

SSBLK15.BL:UUBLK15.OBJ $(DUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK15, 'UUBLK15.OBJ,$(DUTILITY)', notype )

SSBLK16.BL:UUBLK16.OBJ $(SUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK16, 'UUBLK16.OBJ,$(SUTILITY)', notype )

SSBLK17.BL:UUBLK17.OBJ $(SUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK17, 'UUBLK17.OBJ,$(SUTILITY)', notype )

SSBLK18.BL:UUBLK18.OBJ $(SUTILITY)
    submit :PFP:csd/forbldlnew( SSBLK18, 'UUBLK18.OBJ,$(SUTILITY)', notype )

CROSSBAR.BL SEQUENCER.BL: NETWORK.TXT
    submit :PFP:csd/xbc( network.txt )

.for.obj:
    ftn286.new $< $(forflags)

clean:
    delete *.obj,*.lst,*.mpl,*.mp2,*.bl

run:
    reset
    download process.txt
    start process.txt
    ioserve process.txt 3600 -debug

FILE: uuv22.19g/debug/network.txt

LOOP

CYCLE [ 1 ]
    p05, p13 := p00.4; [ REAL*8 XD 1000 ]

CYCLE [ 2 ]
    p08 := p00.2; [ REAL*4 XD_ 1001 ]

CYCLE [ 3 ]
    p05, p13 := p00.4; [ REAL*8 YD 1002 ]

CYCLE [ 4 ]
    p08 := p00.2; [ REAL*4 YD_ 1003 ]

CYCLE [ 5 ]
    p05, p13 := p00.4; [ REAL*8 ZD 1004 ]

CYCLE [ 6 ]
    p08 := p00.2; [ REAL*4 ZD_ 1005 ]

CYCLE [ 7 ]
    p05, p13 := p00.4; [ REAL*8 X 1006 ]

CYCLE [ 8 ]
    p02, p08, p10 := p00.2; [ REAL*4 X_ 1007 ]

```

```

CYCLE [ 9 ]
  p05, p13 := p00.4; [ REAL*8 Y 1008 ]

CYCLE [ 10 ]
  p02, p08, p10 := p00.2; [ REAL*4 Y_ 1009 ]

CYCLE [ 11 ]
  p05, p13 := p00.4; [ REAL*8 Z 1010 ]

CYCLE [ 12 ]
  p02, p08, p10 := p00.2; [ REAL*4 Z_ 1011 ]

CYCLE [ 13 ]
  p04, p13 := p10.2; [ REAL*4 P 1012 ]

CYCLE [ 14 ]
  p04, p05, p13 := p10.2; [ REAL*4 Q 1013 ]

CYCLE [ 15 ]
  p04, p05, p13 := p10.2; [ REAL*4 R 1014 ]

CYCLE [ 16 ]
  p00 := p10.2; [ REAL*4 QUAT(1) 1015 ]

CYCLE [ 17 ]
  p00 := p10.2; [ REAL*4 QUAT(2) 1015 ]

CYCLE [ 18 ]
  p00 := p10.2; [ REAL*4 QUAT(3) 1015 ]

CYCLE [ 19 ]
  p00 := p10.2; [ REAL*4 QUAT(4) 1015 ]

CYCLE [ 20 ]
  p19 := p00.4; [ REAL*8 MASS 1016 ]

CYCLE [ 21 ]
  p06, p10, p21 := p00.2; [ REAL*4 MASS_ 1017 ]

CYCLE [ 22 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(1) 1018 ]

CYCLE [ 23 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(2) 1018 ]

CYCLE [ 24 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(3) 1018 ]

CYCLE [ 25 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(4) 1018 ]

CYCLE [ 26 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(5) 1018 ]

CYCLE [ 27 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(6) 1018 ]

CYCLE [ 28 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(7) 1018 ]

CYCLE [ 29 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(8) 1018 ]

CYCLE [ 30 ]
  p04, p05, p08, p10, p13 := p00.2; [ REAL*4 CIM(9) 1018 ]

CYCLE [ 31 ]
  p13 := p00.4; [ REAL*8 UD 1019 ]

CYCLE [ 32 ]
  p13 := p00.4; [ REAL*8 VD 1022 ]

CYCLE [ 33 ]
  p13 := p00.4; [ REAL*8 WD 1024 ]

CYCLE [ 34 ]
  p13 := p10.2; [ REAL*4 PD 1020 ]

CYCLE [ 35 ]
  p13 := p10.2; [ REAL*4 QD 1021 ]

```

```

CYCLE [ 36 ]
  p13 := p10.2; [ REAL*4 RD 1023 ]

CYCLE [ 37 ]
  p13 := p00.4; [ REAL*8 GR(1) 1025 ]

CYCLE [ 38 ]
  p13 := p00.4; [ REAL*8 GR(2) 1025 ]

CYCLE [ 39 ]
  p13 := p00.4; [ REAL*8 GR(3) 1025 ]

CYCLE [ 40 ]
  p13 := p00.4; [ REAL*8 XYZE(1) 2000 ]

CYCLE [ 41 ]
  p13 := p00.4; [ REAL*8 XYZE(2) 2000 ]

CYCLE [ 42 ]
  p13 := p00.4; [ REAL*8 XYZE(3) 2000 ]

CYCLE [ 43 ]
  p08 := p00.2; [ REAL*4 XYZE_(1) 2001 ]

CYCLE [ 44 ]
  p08 := p00.2; [ REAL*4 XYZE_(2) 2001 ]

CYCLE [ 45 ]
  p08 := p00.2; [ REAL*4 XYZE_(3) 2001 ]

CYCLE [ 46 ]
  p13 := p00.4; [ REAL*8 XYZED(1) 3000 ]

CYCLE [ 47 ]
  p13 := p00.4; [ REAL*8 XYZED(2) 3000 ]

CYCLE [ 48 ]
  p13 := p00.4; [ REAL*8 XYZED(3) 3000 ]

CYCLE [ 49 ]
  p09 := p00.2; [ REAL*4 ZISP 4000 ]

CYCLE [ 50 ]
  p06, p09, p26, p13, p15, p25, p23 := p00.2; [ REAL*4 CG(1) 4001 ]

CYCLE [ 51 ]
  p06, p09, p26, p13, p15, p25, p23 := p00.2; [ REAL*4 CG(2) 4001 ]

CYCLE [ 52 ]
  p06, p09, p26, p13, p15, p25, p23 := p00.2; [ REAL*4 CG(3) 4001 ]

CYCLE [ 53 ]
  p10, p19 := p00.2; [ REAL*4 IXX 4002 ]

CYCLE [ 54 ]
  p06, p10, p19 := p00.2; [ REAL*4 IYY 4003 ]

CYCLE [ 55 ]
  p06, p10, p19 := p00.2; [ REAL*4 IZZ 4004 ]

CYCLE [ 56 ]
  p00, p10 := p09.2; [ REAL*4 FXT 4006 ]

CYCLE [ 57 ]
  p00 := p09.2; [ REAL*4 FYT 4008 ]

CYCLE [ 58 ]
  p00 := p09.2; [ REAL*4 FZT 4010 ]

CYCLE [ 59 ]
  p10 := p09.2; [ REAL*4 MXT 4012 ]

CYCLE [ 60 ]
  p10 := p09.2; [ REAL*4 MYT 4014 ]

CYCLE [ 61 ]
  p10 := p09.2; [ REAL*4 MZT 4016 ]

CYCLE [ 62 ]
  p00 := p09.2; [ REAL*4 MDOTT 4018 ]

```

```

CYCLE [ 63 ]
  p00, p10 := p23.2; [ REAL*4 FRCX 4005 ]
CYCLE [ 64 ]
  p00 := p23.2; [ REAL*4 FRCY 4007 ]
CYCLE [ 65 ]
  p00 := p23.2; [ REAL*4 FRCZ 4009 ]
CYCLE [ 66 ]
  p10 := p23.2; [ REAL*4 MRCX 4011 ]
CYCLE [ 67 ]
  p10 := p23.2; [ REAL*4 MRCY 4013 ]
CYCLE [ 68 ]
  p10 := p23.2; [ REAL*4 MRCZ 4015 ]
CYCLE [ 69 ]
  p00 := p23.2; [ REAL*4 MDOTF 4017 ]
CYCLE [ 70 ]
  p09 := p23.2; [ REAL*4 FOFF1(1) 5000 ]
CYCLE [ 71 ]
  p09 := p23.2; [ REAL*4 FOFF1(2) 5000 ]
CYCLE [ 72 ]
  p09 := p23.2; [ REAL*4 FOFF1(3) 5000 ]
CYCLE [ 73 ]
  p09 := p23.2; [ REAL*4 FOFF1(4) 5000 ]
CYCLE [ 74 ]
  p09 := p23.2; [ REAL*4 FOFF2(1) 5001 ]
CYCLE [ 75 ]
  p09 := p23.2; [ REAL*4 FOFF2(2) 5001 ]
CYCLE [ 76 ]
  p09 := p23.2; [ REAL*4 FOFF2(3) 5001 ]
CYCLE [ 77 ]
  p09 := p23.2; [ REAL*4 FOFF2(4) 5001 ]
CYCLE [ 78 ]
  p05 := p08.2; [ REAL*4 CER(1) 6000 ]
CYCLE [ 79 ]
  p05 := p08.2; [ REAL*4 CER(2) 6000 ]
CYCLE [ 80 ]
  p05 := p08.2; [ REAL*4 CER(3) 6000 ]
CYCLE [ 81 ]
  p05 := p08.2; [ REAL*4 CER(4) 6000 ]
CYCLE [ 82 ]
  p05 := p08.2; [ REAL*4 CER(5) 6000 ]
CYCLE [ 83 ]
  p05 := p08.2; [ REAL*4 CER(6) 6000 ]
CYCLE [ 84 ]
  p05 := p08.2; [ REAL*4 CER(7) 6000 ]
CYCLE [ 85 ]
  p05 := p08.2; [ REAL*4 CER(8) 6000 ]
CYCLE [ 86 ]
  p05 := p08.2; [ REAL*4 CER(9) 6000 ]
CYCLE [ 87 ]
  p02, p12 := p08.2; [ REAL*4 ALT 7000 ]
CYCLE [ 88 ]
  p09 := p12.2; [ REAL*4 PRESS 8001 ]
  p14 := p05.4; [ REAL*8 GRT(1,1) 8002 ]
CYCLE [ 89 ]

```

```

p14 := p05.4; [ REAL*8 GRT(1,2) 8002 ]
p15 := p12.2; [ REAL*4 RHO 8003 ]

CYCLE [ 90 ]
p14 := p05.4; [ REAL*8 GRT(1,3) 8002 ]
p15 := p12.2; [ REAL*4 VSND 8004 ]

CYCLE [ 91 ]
p23 := p15.2; [ REAL*4 MACH 8006 ]
p14 := p05.4; [ REAL*8 VTIC(1,1) 8010 ]

CYCLE [ 92 ]
p23 := p15.2; [ REAL*4 QA 8007 ]
p14 := p05.4; [ REAL*8 VTIC(1,2) 8010 ]

CYCLE [ 93 ]
p00, p10 := p15.2; [ REAL*4 FXA 8008 ]
p14 := p05.4; [ REAL*8 VTIC(1,3) 8010 ]

CYCLE [ 94 ]
p00 := p15.2; [ REAL*4 FYA 8009 ]
p14 := p05.4; [ REAL*8 RTIC(1,1) 8015 ]

CYCLE [ 95 ]
p00 := p15.2; [ REAL*4 FZA 8011 ]
p14 := p05.4; [ REAL*8 RTIC(1,2) 8015 ]

CYCLE [ 96 ]
p10 := p15.2; [ REAL*4 MXA 8012 ]
p14 := p05.4; [ REAL*8 RTIC(1,3) 8015 ]

CYCLE [ 97 ]
p10 := p15.2; [ REAL*4 MYA 8013 ]

CYCLE [ 98 ]
p10 := p15.2; [ REAL*4 MZA 8014 ]

CYCLE [ 99 ]
p15 := p08.2; [ REAL*4 VRWM(1) 8000 ]

CYCLE [ 100 ]
p15 := p08.2; [ REAL*4 VRWM(2) 8000 ]

CYCLE [ 101 ]
p15 := p08.2; [ REAL*4 VRWM(3) 8000 ]

CYCLE [ 102 ]
p15 := p08.2; [ REAL*4 MVRWM 8005 ]

CYCLE [ 103 ]
p14, p20 := p05.2; [ REAL*4 MAGRTR 9000 ]

CYCLE [ 104 ]
p14 := p05.4; [ REAL*8 LAMDX(1) 9001 ]

CYCLE [ 105 ]
p14 := p05.4; [ REAL*8 LAMDX(2) 9001 ]

CYCLE [ 106 ]
p20 := p05.2; [ REAL*4 LAMSEK(1) 9002 ]

CYCLE [ 107 ]
p20 := p05.2; [ REAL*4 LAMSEK(2) 9002 ]

CYCLE [ 108 ]
p13 := p04.2; [ REAL*4 PULSE(1) 10000 ]

CYCLE [ 109 ]
p13 := p04.2; [ REAL*4 PULSE(2) 10000 ]

CYCLE [ 110 ]
p13 := p04.2; [ REAL*4 PULSE(3) 10000 ]

CYCLE [ 111 ]
p02 := p05.2; [ REAL*4 RRELTR(1) 11000 ]
p26 := p25.2; [ REAL*4 BFXACS 11003 ]

CYCLE [ 112 ]
p02 := p05.2; [ REAL*4 RRELTR(2) 11000 ]
p26 := p25.2; [ REAL*4 BFXACS 11004 ]

```

```

CYCLE [ 113 ]
  p02 := p05.2; [ REAL*4 RRELTR(3) 11000 ]
  p26 := p25.2; [ REAL*4 BFZACS 11005 ]

CYCLE [ 114 ]
  p02 := p05.2; [ REAL*4 VRELTR(1) 11001 ]
  p26 := p25.2; [ REAL*4 BMXACS 11006 ]

CYCLE [ 115 ]
  p02 := p05.2; [ REAL*4 VRELTR(2) 11001 ]
  p26 := p25.2; [ REAL*4 BMYACS 11007 ]

CYCLE [ 116 ]
  p02 := p05.2; [ REAL*4 VRELTR(3) 11001 ]
  p26 := p25.2; [ REAL*4 BMZACS 11008 ]

CYCLE [ 117 ]
  p02 := p05.2; [ REAL*4 TGOTR 11002 ]
  p26 := p25.2; [ REAL*4 BMDOTA 11009 ]

CYCLE [ 118 ]
  p19 := p26.1; [ INTEGER*2 IACSONA 11011 ]

CYCLE [ 119 ]
  p19 := p25.1; [ INTEGER*2 IACSONB 11010 ]
  p00, p10 := p26.2; [ REAL*4 FXACS 11012 ]

CYCLE [ 120 ]
  p00 := p26.2; [ REAL*4 FYACS 11013 ]

CYCLE [ 121 ]
  p00 := p26.2; [ REAL*4 FZACS 11014 ]

CYCLE [ 122 ]
  p10 := p26.2; [ REAL*4 MXACS 11015 ]

CYCLE [ 123 ]
  p10 := p26.2; [ REAL*4 MYACS 11016 ]

CYCLE [ 124 ]
  p10 := p26.2; [ REAL*4 MZACS 11017 ]

CYCLE [ 125 ]
  p00 := p26.2; [ REAL*4 MDOTA 11018 ]

CYCLE [ 126 ]
  p00, p10 := p09.2; [ REAL*4 FXVCS 11019 ]

CYCLE [ 127 ]
  p00 := p09.2; [ REAL*4 FYVCS 11020 ]

CYCLE [ 128 ]
  p00 := p09.2; [ REAL*4 FZVCS 11021 ]

CYCLE [ 129 ]
  p10 := p09.2; [ REAL*4 MXVCS 11022 ]

CYCLE [ 130 ]
  p10 := p09.2; [ REAL*4 MYVCS 11023 ]

CYCLE [ 131 ]
  p10 := p09.2; [ REAL*4 MZVCS 11024 ]

CYCLE [ 132 ]
  p00 := p09.2; [ REAL*4 MDO TV 11025 ]

CYCLE [ 133 ]
  p01 := p13.2; [ REAL*4 AT(1) 12000 ]

CYCLE [ 134 ]
  p01 := p13.2; [ REAL*4 AT(2) 12000 ]

CYCLE [ 135 ]
  p01 := p13.2; [ REAL*4 AT(3) 12000 ]

CYCLE [ 136 ]
  p14 := p13.4; [ REAL*8 RMIR(1) 12001 ]

CYCLE [ 137 ]
  p14 := p13.4; [ REAL*8 RMIR(2) 12001 ]

```

```

CYCLE [ 138 ]
  p14 := p13.4; [ REAL*8 RMIR(3) 12001 ]

CYCLE [ 139 ]
  p01, p06, p21 := p13.2; [ REAL*4 RMIR_(1) 12002 ]

CYCLE [ 140 ]
  p01, p06, p21 := p13.2; [ REAL*4 RMIR_(2) 12002 ]

CYCLE [ 141 ]
  p01, p06, p21 := p13.2; [ REAL*4 RMIR_(3) 12002 ]

CYCLE [ 142 ]
  p14 := p13.4; [ REAL*8 VMIR(1) 12003 ]

CYCLE [ 143 ]
  p14 := p13.4; [ REAL*8 VMIR(2) 12003 ]

CYCLE [ 144 ]
  p14 := p13.4; [ REAL*8 VMIR(3) 12003 ]

CYCLE [ 145 ]
  p01, p06, p21 := p13.2; [ REAL*4 VMIR_(1) 12004 ]

CYCLE [ 146 ]
  p01, p06, p21 := p13.2; [ REAL*4 VMIR_(2) 12004 ]

CYCLE [ 147 ]
  p01, p06, p21 := p13.2; [ REAL*4 VMIR_(3) 12004 ]

CYCLE [ 148 ]
  p19, p21 := p13.2; [ REAL*4 SP 12005 ]

CYCLE [ 149 ]
  p06, p19, p21 := p13.2; [ REAL*4 SQ 12006 ]

CYCLE [ 150 ]
  p06, p19, p21 := p13.2; [ REAL*4 SR 12007 ]

CYCLE [ 151 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(1) 12008 ]

CYCLE [ 152 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(2) 12008 ]

CYCLE [ 153 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(3) 12008 ]

CYCLE [ 154 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(4) 12008 ]

CYCLE [ 155 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(5) 12008 ]

CYCLE [ 156 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(6) 12008 ]

CYCLE [ 157 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(7) 12008 ]

CYCLE [ 158 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(8) 12008 ]

CYCLE [ 159 ]
  p01, p06, p14, p20, p21 := p13.2; [ REAL*4 TI2M(9) 12008 ]

CYCLE [ 160 ]
  p01, p21 := p13.2; [ REAL*4 MVR 12009 ]

CYCLE [ 161 ]
  p21 := p13.2; [ REAL*4 VTT(1) 12010 ]

CYCLE [ 162 ]
  p21 := p13.2; [ REAL*4 VTT(2) 12010 ]

CYCLE [ 163 ]
  p21 := p13.2; [ REAL*4 VTT(3) 12010 ]

CYCLE [ 164 ]
  p21 := p14.2; [ REAL*4 URREL(1) 13000 ]

```

```

CYCLE [ 165 ]
  p21 := p14.2; [ REAL*4 URREL(2) 13000 ]

CYCLE [ 166 ]
  p21 := p14.2; [ REAL*4 URREL(3) 13000 ]

CYCLE [ 167 ]
  p20 := p14.2; [ REAL*4 RREL(1) 13001 ]

CYCLE [ 168 ]
  p20 := p14.2; [ REAL*4 RREL(2) 13001 ]

CYCLE [ 169 ]
  p20 := p14.2; [ REAL*4 RREL(3) 13001 ]

CYCLE [ 170 ]
  p20 := p14.2; [ REAL*4 VREL(1) 13002 ]

CYCLE [ 171 ]
  p20 := p14.2; [ REAL*4 VREL(2) 13002 ]

CYCLE [ 172 ]
  p20 := p14.2; [ REAL*4 VREL(3) 13002 ]

CYCLE [ 173 ]
  p19 := p14.2; [ REAL*4 TGO 13003 ]

CYCLE [ 174 ]
  p21 := p14.2; [ REAL*4 MAGR 13004 ]

CYCLE [ 175 ]
  p19, p21 := p14.2; [ REAL*4 MAGV 13005 ]

CYCLE [ 176 ]
  p21 := p14.2; [ REAL*4 PITERO 13006 ]

CYCLE [ 177 ]
  p21 := p14.2; [ REAL*4 YAWERO 13007 ]

CYCLE [ 178 ]
  p19, p20, p21 := p14.1; [ INTEGER*2 ACQD 13008 ]

CYCLE [ 179 ]
  p06 := p01.2; [ REAL*4 THTER 14000 ]

CYCLE [ 180 ]
  p06 := p01.2; [ REAL*4 PSIER 14001 ]

CYCLE [ 181 ]
  p00, p02, p19 := p21.1; [ INTEGER*2 IDROP 15000 ]
  p14 := p20.2; [ REAL*4 FRMRAT 15005 ]

CYCLE [ 182 ]
  p19 := p21.1; [ INTEGER*2 IBURND 15001 ]
  p14 := p20.2; [ REAL*4 LAMMO(1) 15006 ]

CYCLE [ 183 ]
  p19 := p21.1; [ INTEGER*2 IBURNM 15002 ]
  p14 := p20.2; [ REAL*4 LAMMO(2) 15006 ]

CYCLE [ 184 ]
  p19 := p21.1; [ INTEGER*2 IDMEAS 15003 ]
  p14 := p20.2; [ REAL*4 RRELO(1) 15008 ]

CYCLE [ 185 ]
  p19 := p21.2; [ REAL*4 ADISTT(1,1) 15004 ]
  p14 := p20.2; [ REAL*4 RRELO(2) 15008 ]

CYCLE [ 186 ]
  p19 := p21.2; [ REAL*4 ADISTT(1,2) 15004 ]
  p14 := p20.2; [ REAL*4 RRELO(3) 15008 ]

CYCLE [ 187 ]
  p19 := p21.2; [ REAL*4 ADISTT(1,3) 15004 ]

CYCLE [ 188 ]
  p19 := p21.2; [ REAL*4 ADISTT(2,1) 15004 ]

CYCLE [ 189 ]
  p19 := p21.2; [ REAL*4 ADISTT(2,2) 15004 ]

```

```

CYCLE [ 190 ]
  p19 := p21.2; [ REAL*4 ADISTT(2,3) 15004 ]

CYCLE [ 191 ]
  p19 := p21.2; [ REAL*4 ADISTT(3,1) 15004 ]

CYCLE [ 192 ]
  p19 := p21.2; [ REAL*4 ADISTT(3,2) 15004 ]

CYCLE [ 193 ]
  p19 := p21.2; [ REAL*4 ADISTT(3,3) 15004 ]

CYCLE [ 194 ]
  p19 := p21.2; [ REAL*4 ADISTT(4,1) 15004 ]

CYCLE [ 195 ]
  p19 := p21.2; [ REAL*4 ADISTT(4,2) 15004 ]

CYCLE [ 196 ]
  p19 := p21.2; [ REAL*4 ADISTT(4,3) 15004 ]

CYCLE [ 197 ]
  p19 := p21.2; [ REAL*4 VGM(1) 15007 ]

CYCLE [ 198 ]
  p19 := p21.2; [ REAL*4 VGM(2) 15007 ]

CYCLE [ 199 ]
  p19 := p21.2; [ REAL*4 VGM(3) 15007 ]

CYCLE [ 200 ]
  p14, p21 := p20.2; [ REAL*4 SNRO 15009 ]

CYCLE [ 201 ]
  p14 := p20.2; [ REAL*4 TI2MO(1) 15010 ]
  p09, p19 := p21.1; [ INTEGER*2 IVCS 15011 ]

CYCLE [ 202 ]
  p14 := p20.2; [ REAL*4 TI2MO(2) 15010 ]
  p09 := p06.2; [ REAL*4 CMMD(1) 15012 ]
  p01 := p21.2; [ REAL*4 UVS(1) 15016 ]

CYCLE [ 203 ]
  p14 := p20.2; [ REAL*4 TI2MO(3) 15010 ]
  p09 := p06.2; [ REAL*4 CMMD(2) 15012 ]
  p01 := p21.2; [ REAL*4 UVS(2) 15016 ]

CYCLE [ 204 ]
  p14 := p20.2; [ REAL*4 TI2MO(4) 15010 ]
  p23 := p06.2; [ REAL*4 VCMD(1) 15013 ]
  p01 := p21.2; [ REAL*4 UVS(3) 15016 ]

CYCLE [ 205 ]
  p14 := p20.2; [ REAL*4 TI2MO(5) 15010 ]
  p23 := p06.2; [ REAL*4 VCMD(2) 15013 ]
  p01 := p21.2; [ REAL*4 MVS 15017 ]

CYCLE [ 206 ]
  p14 := p20.2; [ REAL*4 TI2MO(6) 15010 ]
  p23 := p06.2; [ REAL*4 VCMD(3) 15013 ]

CYCLE [ 207 ]
  p14 := p20.2; [ REAL*4 TI2MO(7) 15010 ]
  p23 := p06.2; [ REAL*4 VCMD(4) 15013 ]

CYCLE [ 208 ]
  p14 := p20.2; [ REAL*4 TI2MO(8) 15010 ]
  p23 := p06.1; [ INTEGER*2 IFTAB 15014 ]

CYCLE [ 209 ]
  p14 := p20.2; [ REAL*4 TI2MO(9) 15010 ]
  p23 := p06.2; [ REAL*4 TFTAB 15015 ]

CYCLE [ 210 ]
  p14 := p20.2; [ REAL*4 VRELO(1) 15018 ]

CYCLE [ 211 ]
  p14 := p20.2; [ REAL*4 VRELO(2) 15018 ]

CYCLE [ 212 ]
  p14 := p20.2; [ REAL*4 VRELO(3) 15018 ]

```

```
CYCLE [ 213 ]  
  p19 := p14.2; [ REAL*4 TGIL 16000 ]  
  
CYCLE [ 214 ]  
  p19 := p14.2; [ REAL*4 PITER 16001 ]  
  
CYCLE [ 215 ]  
  p19 := p14.2; [ REAL*4 YAWER 16002 ]  
  
CYCLE [ 216 ]  
  p19 := p14.4; [ REAL*8 LAMD(1) 16003 ]  
  
CYCLE [ 217 ]  
  p19 := p14.4; [ REAL*8 LAMD(2) 16003 ]  
  
CYCLE [ 218 ]  
  p19 := p14.2; [ REAL*4 TRMTGO 16004 ]  
  
CYCLE [ 219 ]  
  p19 := p14.2; [ REAL*4 TGE1 16005 ]  
  
CYCLE [ 220 ]  
  p19 := p14.2; [ REAL*4 TGE2AL 16006 ]  
  
CYCLE [ 221 ]  
  p19 := p14.1; [ INTEGER*2 IBURN1 16007 ]  
  
CYCLE [ 222 ]  
  p21 := p14.1; [ INTEGER*2 ESTATE 16008 ]  
  
CYCLE [ 223 ]  
  p19 := p21.2; [ REAL*4 ROLLER 16009 ]  
  
CYCLE [ 224 ]  
  p26, p25 := p19.2; [ REAL*4 ACSLEV 17000 ]  
  
CYCLE [ 225 ]  
  p26, p25 := p19.1; [ INTEGER*2 ITHRES 17001 ]  
  
CYCLE [ 226 ]  
  p09 := p19.2; [ REAL*4 DTOFFV(1) 17002 ]  
  
CYCLE [ 227 ]  
  p09 := p19.2; [ REAL*4 DTOFFV(2) 17002 ]  
  
CYCLE [ 228 ]  
  p09 := p19.2; [ REAL*4 DTOFFV(3) 17002 ]  
  
CYCLE [ 229 ]  
  p09 := p19.2; [ REAL*4 DTOFFV(4) 17002 ]  
  
CYCLE [ 230 ]  
  p09 := p19.1; [ INTEGER*2 IVTAB 17003 ]  
  
CYCLE [ 231 ]  
  p09 := p19.2; [ REAL*4 TBURNM 17004 ]  
  
CYCLE [ 232 ]  
  p09 := p19.2; [ REAL*4 TIMONV 17005 ]  
  
CYCLE [ 233 ]  
  p09 := p19.2; [ REAL*4 TOFFLT(1) 17006 ]  
  
CYCLE [ 234 ]  
  p09 := p19.2; [ REAL*4 TOFFLT(2) 17006 ]  
  
CYCLE [ 235 ]  
  p09 := p19.2; [ REAL*4 TOFFLT(3) 17006 ]  
  
CYCLE [ 236 ]  
  p09 := p19.2; [ REAL*4 TOFFLT(4) 17006 ]  
  
CYCLE [ 237 ]  
  p09 := p19.2; [ REAL*4 IVTAB 17007 ]  
  
CYCLE [ 238 ]  
  p26 := p19.2; [ REAL*4 DTACSA(1) 17008 ]  
  
CYCLE [ 239 ]  
  p26 := p19.2; [ REAL*4 DTACSA(2) 17008 ]
```

```

CYCLE [ 240 ]
  p26 := p19.2; [ REAL*4 DTACSA(3) 17008 ]

CYCLE [ 241 ]
  p26 := p19.2; [ REAL*4 DTACSA(4) 17008 ]

CYCLE [ 242 ]
  p25 := p19.2; [ REAL*4 DTACSB(1) 17009 ]

CYCLE [ 243 ]
  p25 := p19.2; [ REAL*4 DTACSB(2) 17009 ]

CYCLE [ 244 ]
  p25 := p19.2; [ REAL*4 DTACSB(3) 17009 ]

CYCLE [ 245 ]
  p25 := p19.2; [ REAL*4 DTACSB(4) 17009 ]

CYCLE [ 246 ]
  p26, p25 := p19.2; [ REAL*4 TATAB 17010 ]

CYCLE [ 247 ]
  p21 := p19.1; [ INTEGER*2 MIDBRN 17011 ]

CYCLE [ 248 ]
  p21 := p19.1; [ INTEGER*2 ICMD 17012 ]

CYCLE [ 249 ]
  p21 := p19.1; [ INTEGER*2 IDIST 17013 ]

CYCLE [ 250 ]
  p00, p01, p04, p05, p06, p08, p09, p10, p26, p12, p13, p14, p15, p25, p19, p20, p21, p23
:= p02.1; [ INTEGER*2 IEXIT 17014 ]

[ p00 = uublk00.for, S = 45, R = 25, 70 ]
[ p01 = uublk01.for, S = 2, R = 24, 26 ]
[ p02 = uublk02.for, S = 1, R = 12, 13 ]
[ p04 = uublk03.for, S = 3, R = 13, 16 ]
[ p05 = uublk04.for, S = 21, R = 27, 48 ]
[ p06 = uublk05.for, S = 8, R = 26, 34 ]
[ p08 = uublk06.for, S = 14, R = 19, 33 ]
[ p09 = uublk07.for, S = 14, R = 29, 43 ]
[ p10 = uublk08.for, S = 10, R = 37, 47 ]
[ p26 = uublk09.for, S = 8, R = 18, 26 ]
[ p12 = uublk10.for, S = 3, R = 2, 5 ]
[ p13 = uublk11.for, S = 31, R = 40, 71 ]
[ p14 = uublk12.for, S = 25, R = 47, 72 ]
[ p15 = uublk13.for, S = 8, R = 10, 18 ]
[ p25 = uublk14.for, S = 8, R = 11, 19 ]
[ p19 = uublk15.for, S = 26, R = 43, 69 ]
[ p20 = uublk16.for, S = 19, R = 20, 39 ]
[ p21 = uublk17.for, S = 25, R = 37, 62 ]
[ p23 = uublk18.for, S = 15, R = 12, 27 ]

```

FILE: uuv22.19g/debug/priority.txt

```

XD
YD
ZD
X
Y
Z
P
Q
R
QUAT
MA~S
CIM
UD
PD
VD
QD
WD
RD
GR
#
XYZE
XYZED

```

```
#
CER
EISP
CG
IXX
IYY
IZZ
BFXACS
BFYACS
BFZACS
BMXACS
BMYACS
BMZACS
BMDOTA
IACSONB
IACSONA
FXACS
FYACS
FZACS
MXACS
MYACS
MZACS
MDOTA
#
ALT
GRT
VTIC
RTIC
FXT
FYT
FZT
MXT
MYT
MZT
MDOTT
#
PULSEG
PRESS
RRELTR
RHO
VSND
FRCX
MAGRTR
FRCY
VRELTR
FRCZ
MRCX
MRCY
LAMDX
MRCZ
MDOTF
LAMSEK
TGOTR
FXVCS
FYVCS
FZVCS
MXVCS
MYVCS
MZVCS
FXA
MDOTV
FYA
FZA
MXA
MYA
MZA
MACH
QA
#
VRWM
MVRWM
#
AT
RMIR
VMIR
ST
SQ
SR
TI2M
#
MVR
```

VTT
 THTER
 PSIER
 #
 IDROP
 IBURN0
 IBURNM
 IDMEAS
 ADISTT
 IEXIT
 TGO
 MAGV
 VGM
 #
 TGIL
 IVCS
 ROLLER
 PITER
 YAWER
 LAMD
 TRMTGO
 TGE1
 TGE2AL
 IBURN1
 ACQD
 UVS
 MVS
 #
 CMMD
 VCMD
 IFTAB
 TFTAB
 ACSLEV
 ITHRES
 DTOFFV
 IVTAB
 TBURNM
 TIMONV
 TOFFLT
 TVTAB
 DTACSA
 DTACSB
 TATAB
 MIDBR
 ICMD
 IDIST

FILE: uuv22.19g/debug/process.txt

p00 ssblk00.bl <null> ssblk00.out
 p01 ssblk01.bl <null> ssblk01.out
 p02 ssblk02.bl uuexosim.txt ssblk02.out
 p04 ssblk03.bl <null> ssblk03.out
 p05 ssblk04.bl <null> ssblk04.out
 p06 ssblk05.bl <null> ssblk05.out
 p08 ssblk06.bl <null> ssblk06.out
 p09 ssblk07.bl <null> ssblk07.out
 p10 ssblk08.bl <null> ssblk08.out
 p26 ssblk09.bl <null> ssblk09.out
 p12 ssblk10.bl <null> ssblk10.out
 p13 ssblk11.bl <null> ssblk11.out
 p14 ssblk12.bl <null> ssblk12.out
 p15 ssblk13.bl <null> ssblk13.out
 p25 ssblk14.bl <null> ssblk14.out
 p19 ssblk15.bl <null> ssblk15.out
 p20 ssblk16.bl <null> ssblk16.out
 p21 ssblk17.bl <null> ssblk17.out
 p23 ssblk18.bl <null> ssblk18.out
 sequencer sequencer.bl <null> <null>
 crossbar crossbar.bl <null> <null>

FILE: uuv22.19g/debug/ssblk00.for

PROGRAM BLK00

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

```
REAL CEI(9)
REAL CG(3)
DOUBLE PRECISION CIE(9)
REAL CIM(9)
DOUBLE PRECISION DELT
DOUBLE PRECISION DTEPS
DOUBLE PRECISION DTR
REAL EISP
REAL FRCX
REAL FRCY
REAL FRCZ
DOUBLE PRECISION FX
REAL FXA
REAL FXACS
REAL FXT
REAL FXVCS
DOUBLE PRECISION FY
REAL FYA
REAL FYACS
REAL FYT
REAL FYVCS
DOUBLE PRECISION FZ
REAL FZA
REAL FZACS
REAL FZT
REAL FZVCS
DOUBLE PRECISION GB(3)
DOUBLE PRECISION GR(3)
INTEGER IDROP
INTEGER IEXIT
INTEGER IMASS
DOUBLE PRECISION IMPLSO
DOUBLE PRECISION IMPULS
REAL IXX
REAL IYY
REAL IZZ
DOUBLE PRECISION LATLP
DOUBLE PRECISION LONGLP
DOUBLE PRECISION MASS
DOUBLE PRECISION MASSO
REAL MASS
DOUBLE PRECISION MDOT
REAL MDOTA
REAL MDOTF
REAL MDOTT
REAL MDOTV
DOUBLE PRECISION MGR
DOUBLE PRECISION MSSTG2
DOUBLE PRECISION MXYZDD
DOUBLE PRECISION PHI
DOUBLE PRECISION PHIICD
DOUBLE PRECISION PSI
DOUBLE PRECISION PSIICD
REAL QUAT(4)
DOUBLE PRECISION T
DOUBLE PRECISION TBRK
DOUBLE PRECISION TDROP
DOUBLE PRECISION TEMPMASS
DOUBLE PRECISION THT
DOUBLE PRECISION THTICD
DOUBLE PRECISION TSTEP
DOUBLE PRECISION TSTG1
DOUBLE PRECISION TSTG2
DOUBLE PRECISION U
DOUBLE PRECISION UD
DOUBLE PRECISION V
DOUBLE PRECISION VD
DOUBLE PRECISION W
DOUBLE PRECISION WBANF
DOUBLE PRECISION WD
DOUBLE PRECISION WDOTFR
DOUBLE PRECISION WDOTKV
DOUBLE PRECISION WDOTTI
DOUBLE PRECISION WDOTTP
DOUBLE PRECISION WEIGHT
DOUBLE PRECISION WKV
DOUBLE PRECISION WKVO
DOUBLE PRECISION WPFRAC
DOUBLE PRECISION WPFRCO
DOUBLE PRECISION WPROP
```

```

DOUBLE PRECISION WPROP1
DOUBLE PRECISION WPROP2
DOUBLE PRECISION X
DOUBLE PRECISION XD
DOUBLE PRECISION XDD
REAL XD_
DOUBLE PRECISION XMTOF
DOUBLE PRECISION XYZE(3)
DOUBLE PRECISION XYZED(3)
DOUBLE PRECISION XYZEDD(3)
REAL XYZE_(3)
REAL X_
DOUBLE PRECISION Y
DOUBLE PRECISION YD
DOUBLE PRECISION YDD
REAL YD_
REAL Y_
DOUBLE PRECISION Z
DOUBLE PRECISION ZD
DOUBLE PRECISION ZDD
REAL ZD_
REAL Z_

```

```
$INCLUDE ('~/INCLUDE/SSBLK00.DAT')
```

```
*LOOP* PROLOGUE
```

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

```
* initialization of variables not computed until end of blk00
  UD      = 0.0
  VD      = 0.0
  WD      = 0.0
  GR(1)   = 0.0
  GR(2)   = 0.0
  GR(3)   = 0.0

```

```

C-----C
C----- MISSILE STATE INITIALIZATION MODULE -----C
C-----C
C              Initialize integrated missile states      C
C-----C

```

```
C      MISSILE MASS PROPERTIES
```

```

MASS      = MASS0
IMPULS    = IMPLS0
WPFRCO    = WPFRCO
WKV       = WKV0
WPROP     = WPROP1

```

```
C      COORDINATE TRANSFORMATION MATRICES
```

```

CALL MMK(SNGL(-90.0*DTR),1,SNGL(LATLP*DTR),2,
  SNGL(LONGLP*DTR),3,CEI)

```

```

CIE(1) = CEI(1)
CIE(2) = CEI(4)
CIE(3) = CEI(7)
CIE(4) = CEI(2)
CIE(5) = CEI(5)
CIE(6) = CEI(8)
CIE(7) = CEI(3)
CIE(8) = CEI(6)
CIE(9) = CEI(9)

```

```
C      COMPUTE MISSILE STATES ~N INERTIAL FRAME
```

```

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

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

XDD = XYZEDD(1)*CEI(1) + XYZEDD(2)*CEI(4) + XYZEDD(3)*CEI(7)
YDD = XYZEDD(1)*CEI(2) + XYZEDD(2)*CEI(5) + XYZEDD(3)*CEI(8)

```

```

ZDD = XYZEDD(1)*CEI(3) + XYZEDD(2)*CEI(6) + XYZEDD(3)*CEI(9)

C      INITIAL MISSILE EULER ANGLES IN RADIANs

      PHI = PHIICD*DTR
      THT = THTICD*DTR
      PSI = PSIICD*DTR

C      COMPUTE INERTIAL TO MISSILE TRANSFORMATION MATRIX

      CALL MMK(SNGL(PHI),1,SNGL(THT),2,SNGL(PSI),3,CIM)

C      INITIALIZE MISSILE TRUTH STATES

      CALL INTEG1 ( MASS      , MDOT      , T , 1 )
      CALL INTEG1 ( WPROP     , WDOTTP    , T , 2 )
      CALL INTEG1 ( IMPULS    , WDOTTI    , T , 3 )
      CALL INTEG1 ( WPFPRAC   , WDOTFR    , T , 4 )
      CALL INTEG1 ( WKV       , WDOTKV    , T , 5 )
      CALL INTEG1 ( XD        , XDD       , T , 6 )
      CALL INTEG1 ( YD        , YDD       , T , 7 )
      CALL INTEG1 ( ZD        , ZDD       , T , 8 )
      CALL INTEG1 ( X         , XD        , T , 9 )
      CALL INTEG1 ( Y         , YD        , T , 10 )
      CALL INTEG1 ( Z         , ZD        , T , 11 )

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

      1000 CONTINUE
*LOOP* START

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

* tmsudriv is no longer needed -- IF/ENDIF and assignment deleted

* The extrapolated states have been deleted. There should be no need
* to look into the future.
* Note that the states which follow have all been initialized, and each
* is integrated at the end of the timestep.
      XD_ = SNGL(XD)
      YD_ = SNGL(YD)
      ZD_ = SNGL(ZD)
      X_  = SNGL(X)
      Y_  = SNGL(Y)
      Z_  = SNGL(Z)
      MASS_ = SNGL(MASS)

      CALL SEND_REAL_64BIT( XD )
      CALL SEND_REAL_32BIT( XD_ )
      CALL SEND_REAL_64BIT( YD )
      CALL SEND_REAL_32BIT( YD_ )
      CALL SEND_REAL_64BIT( ZD )
      CALL SEND_REAL_32BIT( ZD_ )
      CALL SEND_REAL_64BIT( X )
      CALL SEND_REAL_32BIT( X_ )
      CALL SEND_REAL_64BIT( Y )
      CALL SEND_REAL_32BIT( Y_ )
      CALL SEND_REAL_64BIT( Z )
      CALL SEND_REAL_32BIT( Z_ )
      CALL RECEIVE_REAL_32BIT( QUAT(1) )
      CALL RECEIVE_REAL_32BIT( QUAT(2) )
      CALL RECEIVE_REAL_32BIT( QUAT(3) )
      CALL RECEIVE_REAL_32BIT( QUAT(4) )

* MASS is much like the other state variables above in that it should
* have a very close value at this point in the code. Other partitions
* will be notified one timestep later, however, about staging.
      CALL SEND_REAL_64BIT( MASS )
      CALL SEND_REAL_32BIT( MASS_ )

```

* initialization of these variables was added so that they could be
 * sent early.

```
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) )
CALL SEND_REAL_64BIT( UD )
CALL SEND_REAL_64BIT( VD )
CALL SEND_REAL_64BIT( WD )
CALL SEND_REAL_64BIT( GR(1) )
CALL SEND_REAL_64BIT( GR(2) )
CALL SEND_REAL_64BIT( GR(3) )
```

C TRANSFORM INERTIAL POSITION, VELOCITY AND ACCELERATION
 C 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
```

```
XYZE_(1) = SNGL(XYZE(1))
XYZE_(2) = SNGL(XYZE(2))
XYZE_(3) = SNGL(XYZE(3))
```

```
CALL SEND_REAL_64BIT( XYZE(1) )
CALL SEND_REAL_64BIT( XYZE(2) )
CALL SEND_REAL_64BIT( XYZE(3) )
CALL SEND_REAL_32BIT( XYZE_(1) )
CALL SEND_REAL_32BIT( XYZE_(2) )
CALL SEND_REAL_32BIT( XYZE_(3) )
```

```
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
```

```
CALL SEND_REAL_64BIT( XYZED(1) )
CALL SEND_REAL_64BIT( XYZED(2) )
CALL SEND_REAL_64BIT( XYZED(3) )
```

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

```
CALL MASSPR(T,MDOTT,MDOTF,MDOTA,MDOTV,MASS,EISP,TBRK,IMASS,
. MDOT,WEIGHT,WDOTTP,WDOTFR,WDOTKV,WDOTTI,CG,IXX,
. IYY,IZZ)
```

```
CALL SEND_REAL_32BIT( EISP )
CALL SEND_REAL_32BIT( CG(1) )
CALL SEND_REAL_32BIT( CG(2) )
CALL SEND_REAL_32BIT( CG(3) )
CALL SEND_REAL_32BIT( IXX )
CALL SEND_REAL_32BIT( IYY )
CALL SEND_REAL_32BIT( IZZ )
```

* moved up here, since MISSIL doesn't generate these derivs (it needs
 * the old MASS value, which is saved)

```
TEMPMASS = MASS
CALL INTEG ( MASS , MDOT , T , 1 )
CALL INTEG ( WPROP , WDOTTP , T , 2 )
CALL INTEG ( IMPULS , WDOTTI , T , 3 )
CALL INTEG ( WPFRACTION , WDOTFR , T , 4 )
CALL INTEG ( WKV , WDOTKV , T , 5 )
```

C from BTHRST

```
CALL RECEIVE_REAL_32BIT( FXT )
CALL RECEIVE_REAL_32BIT( FYT )
CALL RECEIVE_REAL_32BIT( FZT )
CALL RECEIVE_REAL_32BIT( MDOTT )
```

C from FRCTHR

```

      CALL RECEIVE_REAL_32BIT( FRCX )
      CALL RECEIVE_REAL_32BIT( FRCY )
      CALL RECEIVE_REAL_32BIT( FRCZ )
      CALL RECEIVE_REAL_32BIT( MDOTF )
C   from AERO
      CALL RECEIVE_REAL_32BIT( FXA )
      CALL RECEIVE_REAL_32BIT( FYA )
      CALL RECEIVE_REAL_32BIT( FZA )
C   from ACSTHR
      CALL RECEIVE_REAL_32BIT( FXACS )
      CALL RECEIVE_REAL_32BIT( FYACS )
      CALL RECEIVE_REAL_32BIT( FZACS )
      CALL RECEIVE_REAL_32BIT( MDOTA )
C   from VCSTHR
      CALL RECEIVE_REAL_32BIT( FXVCS )
      CALL RECEIVE_REAL_32BIT( FYVCS )
      CALL RECEIVE_REAL_32BIT( FZVCS )
      CALL RECEIVE_REAL_32BIT( MDOTV )

```

```

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

```

```

      CALL MISSLT(T,QUAT,CIM,TEMPMASS,FXA,FXT,
      .           FRCX,FXACS,FXVCS,FYA,FYT,FRCY,FYACS,FYVCS,FZA,
      .           FZT,FRCZ,FZACS,FZVCS,
      .           X,Y,Z,XD,YD,ZD,UD,VD,WD,
      .           GB,GR,MGR,FX,FY,FZ,XDL,YDD,ZDD,MXYZDD,
      .           U,V,W,PHI,THT,PSI)

```

```

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

```

```

C   FIRST STAGE SEPARATION

      IF ( DABS(T-TSTG1).LE.DTEPS ) THEN
        MASS      = MSSTG2
        WPROP     = WPROP2
        IMASS     = 1

```

REINITIALIZE PERTINENT INTEGRALS

```

      CALL INTEG1 ( MASS      , 0.0D0 , T , 1 )
      CALL INTEG1 ( WPROP    , 0.0D0 , T , 2 )
      CALL INTEG1 ( IMPULS   , 0.0D0 , T , 3 )
      CALL INTEG1 ( WKV      , 0.0D0 , T , 5 )
    ENDIF

C      SECOND STAGE SEPARATION

      IF ( DABS(T-TSTG2).LE.DTEPS ) THEN
        MASS = WKV / XMTOF
        WPROP = 0.0
        IMPULS = 0.0
        IMASS = 1

C      REINITIALIZE PERTINENT INTEGRALS

        CALL INTEG1 ( MASS      , 0.0D0 , T , 1 )
        CALL INTEG1 ( WPROP    , 0.0D0 , T , 2 )
        CALL INTEG1 ( IMPULS   , 0.0D0 , T , 3 )
        CALL INTEG1 ( WKV      , 0.0D0 , T , 5 )
      ENDIF

C      NOSE FAIRING / BOOST ADAPTER SEPARATION

      IF ( IDROP.EQ.1 .OR. (DABS(T-TDROP).LE.DTEPS) ) THEN
        WKV = WKV - WBANF
        MASS = WKV/XMTOF

C      REINITIALIZE PERTINENT INTEGRALS

        CALL INTEG1 ( MASS      , 0.0D0 , T , 1 )
        CALL INTEG1 ( WPROP    , 0.0D0 , T , 2 )
        CALL INTEG1 ( IMPULS   , 0.0D0 , T , 3 )
        CALL INTEG1 ( WKV      , 0.0D0 , T , 5 )
      ENDIF

      CALL RECEIVE_SIGNED_16BIT( IDROP )

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

C      increment time

      TSTEP = TSTEP + 1.0D0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

```

FILE: uuv22.19g/debug/uublk01.for

PROGFAM BLK01

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

REAL AC(3)
REAL AT(3)
REAL DELT
REAL DT
REAL DTBGU

```

REAL DTMP1
REAL DTR
REAL GMU
REAL GREST(3)
INTEGER IEXIT
INTEGER IMINSF
REAL KA
REAL KV
REAL MVR
REAL MVS
REAL PG(3)
REAL PG0(3)
REAL PGD(3)
REAL PM(3)
REAL PSI
REAL PSIER
REAL PSIICD
REAL PSIPG
REAL RADE
REAL RMIR_(3)
REAL SPSI_
REAL STHT
REAL T
REAL T5
REAL TFRCS
REAL TGCALL
REAL TGPUDRIV
REAL TGPUSTEP
REAL THT
REAL THTER
REAL THTICD
REAL THTPG
REAL TI2M(9)
REAL TIMTMP
REAL TLGPU
REAL TMP1
REAL TMP2
REAL TMP3
REAL TMP4
REAL TMP5
REAL TSTCAL
REAL TSTEP
REAL TSTG2
REAL US(3)
REAL US0(3)
REAL US0D
REAL USD(3)
REAL USF(3)
REAL USFD
REAL USI(3)
REAL UVS(3)
REAL VELW0
REAL VELWD
REAL VELWST
REAL VFRCS
REAL VGEMS
REAL VMIR_(3)
REAL VRATIO
REAL VW(3)
REAL VWD(3)
REAL VWIC(3)
REAL WASTAN
REAL WC(3)

```

```
$INCLUDE('~/INCLUDE/SSBLK01.DAT')
```

```
*LOOP* PROLOGUE
```

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

```

C-----C
C----- MISSILE STATE INITIALIZATION MODULE -----C
C-----C
C          Initialize integrated missile states          C
C-----C

```

```
C          INITIAL MISSILE EULER ANGLES IN RADIANS
```

```

THT = THTICD*DTR
PSI = PSIICD*DTR

C      ESTIMATED MISSILE EULER ANGLES AND BODY RATES

STHT = THT
SPSI = PSI

C      INITIALIZE NAVIGATION INTEGRATED PARAMETERS

VELWST = VELWO
VW(1) = VWIC(1)
VW(2) = VWIC(2)
VW(3) = VWIC(3)

C      INITIAL UNIT STEERING VECTOR

USO(1) = COS(SPSI)*COS(USOD*DTR)
USO(2) = SIN(SPSI)*COS(USOD*DTR)
USO(3) = -SIN(USOD*DTR)

C      FINAL UNIT STEERING VECTOR

USF(1) = COS(SPSI)*COS(USFD*DTR)
USF(2) = SIN(SPSI)*COS(USFD*DTR)
USF(3) = -SIN(USFD*DTR)

C      INTERMEDIATE UNIT STEERING VECTOR ( AT FRACS INITIATION )
C      ESTIMATE DELTA FLIGHT PATH ANGLE DURING MINS PORTION OF
C      FRACS DUE TO GRAVITY

TIMTMP = T5
USI(1) = USF(1)
USI(2) = USF(2)
USI(3) = USF(3)
GREST(1) = - GMU/RADE**2
GREST(2) = 0.0
GREST(3) = 0.0
5  CONTINUE
    TMP1 = GREST(2)*USI(3) - GREST(3)*USI(2)
    TMP2 = GREST(3)*USI(1) - GREST(1)*USI(3)
    TMP3 = GREST(1)*USI(2) - GREST(2)*USI(1)
    TMP4 = ( TIMTMP - TFRCS )/( T5 - TFRCS )
    TMP5 = VFRCS + TMP4*( VGEMS - VFRCS )
    USD(1) = ( USI(2)*TMP3 - USI(3)*TMP2 )/TMP5
    USD(2) = ( USI(3)*TMP1 - USI(1)*TMP3 )/TMP5
    USD(3) = ( USI(1)*TMP2 - USI(2)*TMP1 )/TMP5
    TIMTMP = TIMTMP - DTBGU
    USI(1) = USI(1) - DTBGU*USD(1)
    USI(2) = USI(2) - DTBGU*USD(2)
    USI(3) = USI(3) - DTBGU*USD(3)
    TMP1 = SQRT ( USI(1)**2 + USI(2)**2 + USI(3)**2 )
    USI(1) = USI(1)/TMP1
    USI(2) = USI(2)/TMP1
    USI(3) = USI(3)/TMP1
    IF ( TIMTMP.GT.TFRCS ) GO TO 5

C      INITIALIZE GUIDANCE INTEGRATED PARAMETERS

PGO(1) = COS(SPSI)*COS(STHT)
PGO(2) = SIN(SPSI)*COS(STHT)
PGO(3) = -SIN(STHT)

US(1) = USO(1)
US(2) = USO(2)
US(3) = USO(3)
PG(1) = PGO(1)
PG(2) = PGO(2)
PG(3) = PGO(3)

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

```

```
1000 CONTINUE
*LOOP* START
```

```
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      GET TIME SINCE LAST GUIDANCE PROCESSOR UPDATE
C      DT      = T - TLGPU
      TLGPU    = T
      DT      = TGPUSTEP * DELT
C      INTEGRATE GUIDANCE STATES FROM LAST PASS THROUGH
      US(1)   = US(1) + DT*USD(1)
      US(2)   = US(2) + DT*USD(2)
      US(3)   = US(3) + DT*USD(3)
      VELWST  = VELWST + DT*VELWD
      PG(1)   = PG(1) + DT*PGD(1)
      PG(2)   = PG(2) + DT*PGD(2)
      PG(3)   = PG(3) + DT*PGD(3)
      VW(1)   = VW(1) + DT*VWD(1)
      VW(2)   = VW(2) + DT*VWD(2)
      VW(3)   = VW(3) + DT*VWD(3)
C      NORMALIZE UNIT STEERING VECTOR
      DTMP1   = SQRT ( US(1)**2 + US(2)**2 + US(3)**2 )
      US(1)   = US(1) / DTMP1
      US(2)   = US(2) / DTMP1
      US(3)   = US(3) / DTMP1
C      NORMALIZE UNIT POINTING VECTOR
      DTMP1   = SQRT ( PG(1)**2 + PG(2)**2 + PG(3)**2 )
      PG(1)   = PG(1) / DTMP1
      PG(2)   = PG(2) / DTMP1
      PG(3)   = PG(3) / DTMP1
C      DETERMINE COMMANDED BODY ANGLES FOR OUTPUT COMPARISON
      THTPG   = - ASIN ( PG(3) )
      PSIPG   = ATAN2 ( PG(2) , PG(1) )
```

```
ENDIF
```

```
CALL RECEIVE_REAL_32BIT( AT(1) )
CALL RECEIVE_REAL_32BIT( AT(2) )
CALL RECEIVE_REAL_32BIT( AT(3) )
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) )
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( MVR )
```

```
C-----C
C----- BOOST STEERING MODULE -----C
```

```

C-----C
C                                         Calculates the unit steering and
C                                         acceleration direction vector for boost
C                                         phase steering
C-----C

      IF ( TSTEP .GE. TGPUDRIV ) THEN

        TGPUDRIV = TGPUDRIV + TGPUSTEP

        IF ( T.GE.TSTCAL .AND. T.LT.TSTG2 ) THEN

          CALL BSTEER(T,USI,USF,UVS,MVS,MVR,AT,RMIR_,VMIR_,US,USD,
            AC,WASTAN,VRATIO,VELWD)

C-----C
C-----C BOOST GUIDANCE MODULE -----C
C-----C
C                                         This code calculates the error between
C                                         the commanded pointing vector and the
C                                         actual direction in which the intercep-
C                                         tor is pointing. This error signal is
C                                         then sent to the autopilot.
C-----C

          CALL BGUID(T,AT,AC,TI2M,PG,IMINSF,VW,PGD,VWD,WC,PSIER,
            THTER,PM,KA,KV)

C      SCHEDULE TIME FOR NEXT BOOST STEERING/GUIDANCE CALL

          DTMP1 = DTBGU * ANINT ( (T+DTBGU) / DTBGU )
          TSTCAL = DTMP1
          TGCALL = DTMP1

        ENDIF

C      ZERO BOOST STEERING/GUIDANCE DERIVATIVES AFTER SECOND STAGE
C      SEPARATION

        IF ( T.GE.TSTG2 ) THEN
          USD(1) = 0.0
          USD(2) = 0.0
          USD(3) = 0.0
          PGD(1) = 0.0
          PGD(2) = 0.0
          PGD(3) = 0.0
          VWD(1) = 0.0
          VWD(2) = 0.0
          VWD(3) = 0.0
        ENDIF

      ENDIF

      CALL SEND_REAL_32BIT( THTER )
      CALL SEND_REAL_32BIT( PSIER )
      CALL RECEIVE_REAL_32BIT( UVS(1) )
      CALL RECEIVE_REAL_32BIT( UVS(2) )
      CALL RECEIVE_REAL_32BIT( UVS(3) )
      CALL RECEIVE_REAL_32BIT( MVS )

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

C      increment time

      TSTEP = TSTEP + 1.0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )
      *LOOP* STOP

```

```

      IF ( IEXIT.EQ.0 ) GO TO 1000
*LOOP* EPILOGUE
      END

FILE: uuv22.19g/debug/uublk02.for

      PROGRAM BLK02

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

$INCLUDE(' :PFP:INCLUDE/TARGET.FOR')

      REAL ALT
      REAL DELT
      REAL DTEPS
      REAL DTPRT
      INTEGER I
      INTEGER IDROP
      INTEGER IEXIT
      INTEGER MESSAGE_SIZE
      INTEGER MESSAGE_TYPE
      REAL MISS
      INTEGER NUMBER_OUTPUT
      REAL OUTPUT(5,0:149)
      REAL RRELTR(3)
      REAL T
      REAL TDROP
      REAL TFINAL
      REAL TGOMN
      REAL TGOTr
      REAL TSTEP
      REAL TSTG1
      REAL TSTG2
      REAL VRELTR(3)
      REAL X_
      REAL Y_
      REAL Z_

$INCLUDE(' ^/INCLUDE/SSBLK02.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87 .
  CALL CW87

  CALL INPUT_MESSAGE( MESSAGE_TYPE, TFINAL, MESSAGE_SIZE )

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

1000 CONTINUE
*LOOP* START
  CALL RECEIVE_REAL_32BIT( X_ )
  CALL RECEIVE_REAL_32BIT( Y_ )
  CALL RECEIVE_REAL_32BIT( Z_ )
  CALL RECEIVE_REAL_32BIT( ALT )
  CALL RECEIVE_REAL_32BIT( RRELTR(1) )
  CALL RECEIVE_REAL_32BIT( RRELTR(2) )
  CALL RECEIVE_REAL_32BIT( RRELTR(3) )
  CALL RECEIVE_REAL_32BIT( VRELTR(1) )
  CALL RECEIVE_REAL_32BIT( VRELTR(2) )
  CALL RECEIVE_REAL_32BIT( VRELTR(3) )
  CALL RECEIVE_REAL_32BIT( TGOTr )
  CALL RECEIVE_SIGNED_16BIT( IDROP )

C-----C
C-----C SEPARATION MODULE -----C

```

```

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

C      FIRST STAGE SEPARATION

      IF ( ABS(T-TSTG1).LE.DTEPS ) THEN
        CALL OUTMES(0101,T,0.0)
      ENDIF

C      SECOND STAGE SEPARATION

      IF ( ABS(T-TSTG2).LE.DTEPS ) THEN
        CALL OUTMES(0102,T,0.0)
      ENDIF

C      NOSE FAIRING / BOOST ADAPTER SEPARATION

      IF ( IDROP.EQ.1 .OR. (ABS(T-TDROP).LE.DTEPS) ) THEN
        CALL OUTMES(0103,T,0.0)
      ENDIF

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

      if ( nint(mod(tstep,dtprt)).eq.0 ) then
        CALL OUTMES(0104,T,ALT)

        OUTPUT(1,NUMBER_OUTPUT) = T
        OUTPUT(2,NUMBER_OUTPUT) = ALT
        OUTPUT(3,NUMBER_OUTPUT) = X
        OUTPUT(4,NUMBER_OUTPUT) = Y
        OUTPUT(5,NUMBER_OUTPUT) = Z
        NUMBER_OUTPUT = NUMBER_OUTPUT + 1
      ENDIF

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

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

      IF ( (TGOTR.LE.TGOMN) .AND. (T.GT.1.0) ) THEN
        IEXIT = 1
      ENDIF

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 + 1.0
      T = TSTEP * DELT

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

      CALL SEND_SIGNED_16BIT( IEXIT )

```

```

*LOOP* STOP
      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 = SQRT ( (RRELTR(1) + VRELTR(1)*TGOTR)**2
      .          + (RRELTR(2) + VRELTR(2)*TGOTR)**2
      .          + (RRELTR(3) + VRELTR(3)*TGOTR)**2 )

      CALL OUTMES(0105,T,MISS)

      OUTPUT(1,NUMBER_OUTPUT) = T
      OUTPUT(2,NUMBER_OUTPUT) = MISS
      OUTPUT(3,NUMBER_OUTPUT) = X
      OUTPUT(4,NUMBER_OUTPUT) = Y
      OUTPUT(5,NUMBER_OUTPUT) = Z

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

      DO 500 I = 0, NUMBER_OUTPUT
        CALL OUTPUT_MESSAGE(%VAL(REAL_32BIT),OUTPUT(1,I),%VAL(INT2(5)))
        CALL OUTPUT_NL
500  CONTINUE

*LOOP* EPILOGUE

      END

```

FILE: uuv22.19g/debug/uublk03.for

```

PROGRAM BLK03

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

      REAL CIM(9)
      REAL DELT
      INTEGER*4 GYSEED
      INTEGER IEXIT
      REAL P
      REAL PULSESEG(3)
      REAL Q
      REAL QFRACG(3)
      REAL R
      REAL T
      REAL TIMUDRIV
      REAL TIMUSTEP
      INTEGER*4 TOSEED
      REAL TSTEP

$INCLUDE('~/INCLUDE/SSBLK03.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
      CALL CW87

C      INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
      CALL RANIT ( TOSEED )

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
LOOP START

```
CALL RECEIVE_REAL_32BIT( P )
CALL RECEIVE_REAL_32BIT( Q )
CALL RECEIVE_REAL_32BIT( R )
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-----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----- 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
```

```
CALL SEND_REAL_32BIT( PULSEG(1) )
CALL SEND_REAL_32BIT( PULSEG(2) )
CALL SEND_REAL_32BIT( PULSEG(3) )
```

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

C increment time

```
TSTEP = TSTEP + 1.0
T = TSTEP * DELT
```

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

```
CALL RECEIVE_SIGNED_16BIT( IEXIT )
```

```
*LOOP* STOP
IF ( IEXIT.EQ.0 ) GO TO 1000
```

LOOP EPILOGUE

```
END
```

FILE: uuv22.19g/debug/uublk04.for

PROGRAM BLK04

```

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

DOUBLE PRECISION AZSUB(100)
DOUBLE PRECISION CAZ(100)
REAL CEI(9)
DOUBLE PRECISION CEL(100)
REAL CER(9)
DOUBLE PRECISION CIE(9)
REAL CIM(9)
REAL CIT(9)
DOUBLE PRECISION CMS(9)
DOUBLE PRECISION CSK1
DOUBLE PRECISION CSK2
DOUBLE PRECISION CTI(9)
DOUBLE PRECISION DELT
DOUBLE PRECISION DTR
DOUBLE PRECISION ELSUB(100)
DOUBLE PRECISION GRT(5, 3)
INTEGER IEXIT
INTEGER IRESLV
DOUBLE PRECISION LAMDSK(2)
DOUBLE PRECISION LAMDTR(2)
DOUBLE PRECISION LAMDXX(2)
REAL LAMSEK(2)
DOUBLE PRECISION LAMTRU(2)
DOUBLE PRECISION LATLP
DOUBLE PRECISION LATT
DOUBLE PRECISION LONGLP
DOUBLE PRECISION LONGT
DOUBLE PRECISION MAGLOS
REAL MAGRTR
DOUBLE PRECISION MGRDTR
INTEGER NSUB
DOUBLE PRECISION PI
DOUBLE PRECISION PTARG
DOUBLE PRECISION PTRGIC
REAL Q
DOUBLE PRECISION QTARG
DOUBLE PRECISION QTRGIC
REAL R
DOUBLE PRECISION RJ(5)
DOUBLE PRECISION RJSUB(100)
DOUBLE PRECISION RRELM(3)
REAL RRELTR(3)
DOUBLE PRECISION RTAR(3)
DOUBLE PRECISION RTARG
DOUBLE PRECISION RTER(3)
DOUBLE PRECISION RTIC(5, 3)
DOUBLE PRECISION RTRGIC
DOUBLE PRECISION SKOFF1
DOUBLE PRECISION SKOFF2
DOUBLE PRECISION SSK1
DOUBLE PRECISION SSK2
DOUBLE PRECISION T
REAL TGOTR
INTEGER*4 TOSEED
DOUBLE PRECISION TPHI
DOUBLE PRECISION TPHID
DOUBLE PRECISION TPHIIC
DOUBLE PRECISION TPSI
DOUBLE PRECISION TPSID
DOUBLE PRECISION TPSIIC
DOUBLE PRECISION TRSUDRIV
DOUBLE PRECISION TRSUSTEP
DOUBLE PRECISION TSTEP
DOUBLE PRECISION TTHT
DOUBLE PRECISION TTHTD
DOUBLE PRECISION TTHTIC
DOUBLE PRECISION TTSUDRIV
DOUBLE PRECISION TTSUSTEP
DOUBLE PRECISION VRELM(3)
REAL VRELTR(3)
DOUBLE PRECISION VTAR(3)
DOUBLE PRECISION VTIC(5, 3)
DOUBLE PRECISION X
DOUBLE PRECISION XD
DOUBLE PRECISION Y
DOUBLE PRECISION YD
DOUBLE PRECISION Z

```

```

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

```

C-----C

1000 CONTINUE

LOOP START

```
CALL RECEIVE_REAL_64BIT( XD )
CALL RECEIVE_REAL_64BIT( YD )
CALL RECEIVE_REAL_64BIT( ZD )
CALL RECEIVE_REAL_64BIT( X )
CALL RECEIVE_REAL_64BIT( Y )
CALL RECEIVE_REAL_64BIT( Z )
CALL RECEIVE_REAL_32BIT( Q )
CALL RECEIVE_REAL_32BIT( R )
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) )
CALL RECEIVE_REAL_32BIT( CER(1) )
CALL RECEIVE_REAL_32BIT( CER(2) )
CALL RECEIVE_REAL_32BIT( CER(3) )
CALL RECEIVE_REAL_32BIT( CER(4) )
CALL RECEIVE_REAL_32BIT( CER(5) )
CALL RECEIVE_REAL_32BIT( CER(6) )
CALL RECEIVE_REAL_32BIT( CER(7) )
CALL RECEIVE_REAL_32BIT( CER(8) )
CALL RECEIVE_REAL_32BIT( CER(9) )
```

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

IF (TSTEP .GE. TTSUDRIV) THEN

TTSUDRIV = TTSUDRIV + TTSUSTEP

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

ENDIF

```
CALL SEND_REAL_64BIT( GRT(1,1) )
CALL SEND_REAL_64BIT( GRT(1,2) )
CALL SEND_REAL_64BIT( GRT(1,3) )
* CALL SEND_REAL_64BIT( GRT(2,1) )
* CALL SEND_REAL_64BIT( GRT(2,2) )
* CALL SEND_REAL_64BIT( GRT(2,3) )
* CALL SEND_REAL_64BIT( GRT(3,1) )
* CALL SEND_REAL_64BIT( GRT(3,2) )
* CALL SEND_REAL_64BIT( GRT(3,3) )
* CALL SEND_REAL_64BIT( GRT(4,1) )
* CALL SEND_REAL_64BIT( GRT(4,2) )
* CALL SEND_REAL_64BIT( GRT(4,3) )
* CALL SEND_REAL_64BIT( GRT(5,1) )
* CALL SEND_REAL_64BIT( GRT(5,2) )
* CALL SEND_REAL_64BIT( GRT(5,3) )
CALL SEND_REAL_64BIT( VTIC(1,1) )
CALL SEND_REAL_64BIT( VTIC(1,2) )
CALL SEND_REAL_64BIT( VTIC(1,3) )
* CALL SEND_REAL_64BIT( VTIC(2,1) )
* CALL SEND_REAL_64BIT( VTIC(2,2) )
* CALL SEND_REAL_64BIT( VTIC(2,3) )
* CALL SEND_REAL_64BIT( VTIC(3,1) )
* CALL SEND_REAL_64BIT( VTIC(3,2) )
* CALL SEND_REAL_64BIT( VTIC(3,3) )
* CALL SEND_REAL_64BIT( VTIC(4,1) )
* CALL SEND_REAL_64BIT( VTIC(4,2) )
```

```

* CALL SEND_REAL_64BIT( VTIC(4,3) )
* CALL SEND_REAL_64BIT( VTIC(5,1) )
* CALL SEND_REAL_64BIT( VTIC(5,2) )
* CALL SEND_REAL_64BIT( VTIC(5,3) )
CALL SEND_REAL_64BIT( RTIC(1,1) )
CALL SEND_REAL_64BIT( RTIC(1,2) )
CALL SEND_REAL_64BIT( RTIC(1,3) )
* CALL SEND_REAL_64BIT( RTIC(2,1) )
* CALL SEND_REAL_64BIT( RTIC(2,2) )
* CALL SEND_REAL_64BIT( RTIC(2,3) )
* CALL SEND_REAL_64BIT( RTIC(3,1) )
* CALL SEND_REAL_64BIT( RTIC(3,2) )
* CALL SEND_REAL_64BIT( RTIC(3,3) )
* CALL SEND_REAL_64BIT( RTIC(4,1) )
* CALL SEND_REAL_64BIT( RTIC(4,2) )
* CALL SEND_REAL_64BIT( RTIC(4,3) )
* CALL SEND_REAL_64BIT( RTIC(5,1) )
* CALL SEND_REAL_64BIT( RTIC(5,2) )
* CALL SEND_REAL_64BIT( RTIC(5,3) )

```

```

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

```

```
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,CAZ,CEL)

```

```
ENDIF
```

```
CALL SEND_REAL_32BIT( MAGRTR )
CALL SEND_REAL_64BIT( LAMDXX(1) )
CALL SEND_REAL_64BIT( LAMDXX(2) )
CALL SEND_REAL_32BIT( LAMSEK(1) )
CALL SEND_REAL_32BIT( LAMSEK(2) )

```

```
CALL SEND_REAL_32BIT( RRELTR(1) )
CALL SEND_REAL_32BIT( RRELTR(2) )
CALL SEND_REAL_32BIT( RRELTR(3) )
CALL SEND_REAL_32BIT( VRELTR(1) )
CALL SEND_REAL_32BIT( VRELTR(2) )
CALL SEND_REAL_32BIT( VRELTR(3) )
CALL SEND_REAL_32BIT( TGOTR )

```

```

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

```

```
C increment time
```

```
    TSTEP = TSTEP + 1.0D0
    T = TSTEP * DELT
```

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

```
    CALL RECEIVE_SIGNED_16BIT( IEXIT )
```

```
*LOOP* STOP
```

```
    I ( IEXIT.EQ.0 ) GO TO 1000
```

```
*LOOP* EPILOGUE
```

```
END
```

```
FILE: uuv22.19g/debug/uublk05.for
```

```

PROGRAM BLK05

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

REAL CG(3)
REAL CGEST(3)
REAL CMMD(2)
REAL CNALP
REAL DELT
REAL DLPC
REAL DLYC
REAL DTEPS
REAL DTFRU
REAL DTMP1
REAL EPSL
INTEGER IBAUTO
INTEGER IEXIT
INTEGER IFTAB
REAL IYY
REAL IZZ
REAL KME
REAL KNE
REAL KTHT
REAL KTHTD
REAL LFRACS
REAL MALPHA
REAL MASS
REAL MDELTA
REAL PSIER
REAL RMIR_(3)
REAL SQ
REAL SR
REAL T
REAL TAPUDRIV
REAL TAPUSTEP
REAL TFRAC
REAL TFRCS
REAL TFTAB
REAL THTER
REAL TI2M(9)
REAL TSTEP
REAL TSTG1
REAL TSTG2
REAL VCMD(4)
INTEGER VLVCMS
REAL VMIR_(3)
REAL XCPCG
REAL XDEL

$INCLUDE('~/INCLUDE/SSBLK05.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
  CALL CW87

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

1000 CONTINUE
*LOOP* START

  CALL RECEIVE_REAL_32BIT( MASS )
  CALL RECEIVE_REAL_32BIT( CG(1) )
  CALL RECEIVE_REAL_32BIT( CG(2) )
  CALL RECEIVE_REAL_32BIT( CG(3) )
  CALL RECEIVE_REAL_32BIT( IYY )
  CALL RECEIVE_REAL_32BIT( IZZ )
  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) )
CALL RECEIVE_REAL_32BIT( SQ )
CALL RECEIVE_REAL_32BIT( SR )
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( THTER )
CALL RECEIVE_REAL_32BIT( PSIER )

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

      IF ( TSTEP .GE. TAPUDRIV ) THEN

*      TAPUDRIV = TAPUDRIV + TAPUSTEP

      IF ( T.LT.TSTG2 ) THEN

C      CGEST TEMPORARILY EQUAL TO CG

      CGEST(1) = CG(1)
      CGEST(2) = CG(2)
      CGEST(3) = CG(3)

C-----C
C----- BOOST AUTOPILOT MODULE -----C
C-----C
C      Computes commands to the steering devicesC
C-----C

C      FIRST STAGE SEPARATION

      IF ( ABS(T-TSTG1).LE.DTEPS ) THEN
        IBAUTO = 1
      ENDIF

      CALL BAUTO(T,THTER,PSIER,SQ,SR,MASS ,IYY,IZZ,CGEST,TI2M,
        , RMIR, VMIR, IBAUTO, CMMD, DLPC, DLYC, KTHT, KTHTD, XDEL, XCPCG,
        , LFRACS, CNAIP, MDELTA, KNE, KME, MALPHA)

      ENDIF

      ENDIF

* bauto
      CALL SEND_REAL_32BIT( CMMD(1) )
      CALL SEND_REAL_32BIT( CMMD(2) )

C-----C
C----- FRACS LOGIC MODULE -----C
C-----C
C      Models FRACS hysteresis logicC
C-----C

      IF ( TSTEP .GE. TAPUDRIV ) THEN

      TAPUDRIV = TAPUDRIV + TAPUSTEP

      IF ( T.LT.TSTG2 ) THEN

      IF ( T.GE.TFRCS .AND. T.GE.TFRAC ) THEN

      CALL FRACS(T,DLPC,DLYC,VCMD,VLCMS)

C      SET FLAG TO COMPUTE FRACS THRUSTER RESPONSE TABLE

```

```

        IFTAB = 1
        TFTAB = T

C      SCHEDULE NEXT FRACS CALCULATION

        DTMP1 = DTFRU * ANINT ( (T+DTFRU) / DTFRU )
        TFRAC = DTMP1 - EPSL

        ENDIF

    ENDIF

ENDIF

* fracs
  CALL SEND_REAL_32BIT( VCMD(1) )
  CALL SEND_REAL_32BIT( VCMD(2) )
  CALL SEND_REAL_32BIT( VCMD(3) )
  CALL SEND_REAL_32BIT( VCMD(4) )
  CALL SEND_SIGNED_16BIT( IFTAB )
  CALL SEND_REAL_32BIT( TFTAB )

* The IFTAB assignment was moved from the partition with FRCTHR
  IFTAB = 0

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

C      increment time

        TSTEP = TSTEP + 1.0D0
        T = TSTEP * DELT

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

        CALL RECEIVE_SIGNED_16BIT( IEXIT )
*LOOP* STOP
        IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

        END

```

FILE: uuv22.19g/debug/uublk06.for

```

PROGRAM BLK06

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

REAL ALT
REAL CEI(9)
REAL CER(9)
REAL CIE(9)
REAL CIM(9)
REAL CIR(9)
REAL CRI(9)
REAL DELT
REAL DTR
INTEGER IEXIT
REAL LAT
REAL LATLP
REAL LONG
REAL LONGLP
REAL MVRWM
REAL OMEGAE
REAL RADE
REAL SHEAR
REAL T
REAL TSTEP

```

```

REAL TSTG2
REAL VRWM(3)
REAL VWIND
REAL XD_
REAL XYZE_(3)
REAL XYZR(3)
REAL X_
REAL YD_
REAL Y_
REAL ZD_
REAL Z_

```

```
$INCLUDE('^/INCLUDE/SSBLK06.DAT')
```

```
*LOOP* PROLOGUE
```

```

* INITIALIZE 80x87
  CALL CW87

```

```

C-----C
C-----C  MISSILE STATE INITIALIZATION MODULE  ---C
C-----C
C          Initialize integrated missile states      C
C                                                    C
C-----C

```

```
C      COORDINATE TRANSFORMATION MATRICES
```

```
CALL MMK(-90.0*DTR,1,LATLP*DTR,2,LOGLP*DTR,3,CEI)
```

```

CIE(1) = CEI(1)
CIE(2) = CEI(4)
CIE(3) = CEI(7)
CIE(4) = CEI(2)
CIE(5) = CEI(5)
CIE(6) = CEI(8)
CIE(7) = CEI(3)
CIE(8) = CEI(6)
CIE(9) = CEI(9)

```

```

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
*LOOP* START

```

```

CALL RECEIVE_REAL_32BIT( XD_ )
CALL RECEIVE_REAL_32BIT( YD_ )
CALL RECEIVE_REAL_32BIT( ZD_ )
CALL RECEIVE_REAL_32BIT( X_ )
CALL RECEIVE_REAL_32BIT( Y_ )
CALL RECEIVE_REAL_32BIT( Z_ )
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) )
CALL RECEIVE_REAL_32BIT( XYZE_(1) )
CALL RECEIVE_REAL_32BIT( XYZE_(2) )
CALL RECEIVE_REAL_32BIT( XYZE_(3) )

```

```
C      ROTATING EARTH MODEL
```

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

```

* CER used to be recalculated later, along with other values
* associated with the rotating earth model. We now use only

```

```

* these values derived from the first-order estimates
CALL SEND_REAL_32BIT( CER(1) )
CALL SEND_REAL_32BIT( CER(2) )
CALL SEND_REAL_32BIT( CER(3) )
CALL SEND_REAL_32BIT( CER(4) )
CALL SEND_REAL_32BIT( CER(5) )
CALL SEND_REAL_32BIT( CER(6) )
CALL SEND_REAL_32BIT( CER(7) )
CALL SEND_REAL_32BIT( CER(8) )
CALL SEND_REAL_32BIT( CER(9) )

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    = ATAN2(XYZR(3),SQRT(XYZR(1)**2+XYZR(2)**2))/DTR
LONG   = ATAN2(XYZR(2),XYZR(1))/DTR

C      CALCULATE CURRENT MISSILE ALTITUDE

ALT    = SQRT ( X_**2 + Y_**2 + Z_**2 ) - RADE
CALL SEND_REAL_32BIT( ALT )

C-----C
C----- ATMOSPHERE MODULE -----C
C-----C
C      Computes the atmospheric properties      C
C-----C

      IF ( T.LT.TSTG2 ) THEN
        CALL ATMOS2(T,ALT,XD ,YD ,ZD ,CIM,CRI,LAT,LONG,
          VWIND,SHEAR,VRWM,MVRWM)
      ENDIF

      CALL SEND_REAL_32BIT( VRWM(1) )
      CALL SEND_REAL_32BIT( VRWM(2) )
      CALL SEND_REAL_32BIT( VRWM(3) )
      CALL SEND_REAL_32BIT( MVRWM )

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

C      increment time

      TSTEP = TSTEP + 1.0
      T = TSTEP * DELT

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

```

```

      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

```

FILE: uu22.19g/debug/uublk07.for

```

PROGRAM BLK07

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

REAL CG(3)
REAL CMMD(2)
REAL DELT
REAL DLP
REAL DLPD
REAL DLPIC
REAL DLY
REAL DLYD
REAL DLYIC
REAL DTEPS
REAL DTOFFV(4)
REAL EISP
REAL FOFF1(4)
REAL FOFF2(4)
REAL FXT
REAL FXVCS
REAL FYT
REAL FYVCS
REAL FZT
REAL FZVCS
INTEGER IBTHR
INTEGER IEXIT
INTEGER IVCS
INTEGER IVTAB
REAL MDOTT
REAL MDOTV
REAL MXT
REAL MXVCS
REAL MYT
REAL MYVCS
REAL MZT
REAL MZVCS
REAL PMAX
REAL PRESS
REAL T
REAL TBRK
REAL TBURNM
REAL THR
REAL THRV
REAL TIMONV
REAL TINHIB
REAL TKVON
REAL TOFFLT(4)
INTEGER*4 TOSEED
REAL TOTDEL
REAL TSTEP
REAL TSTG1
REAL TSTG2
REAL TVTAB

$INCLUDE('~/INCLUDE/SSBLK07.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
  CALL CW87

C      INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
      CALL RANIT ( TOSEED )

* initialization for purpose of delaying receipt of actual values
  PRESS = 2116.25

```

```

C-----C
C----- MISSILE STATE INITIALIZATION MODULE -----C
C-----C
C           Initialize integrated missile states      C
C-----C

C           INITIAL TVC NOZZLE POSITION

          DLP = DLPIC
          DLY = DLYIC
          DLPD = 0.0
          DLYD = 0.0

          CALL INTEG1 ( DLP      , DLPD      , T , 19 )
          CALL INTEG1 ( DLY      , DLYD      , T , 20 )

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
*LOOP* START

C from MASSPR
  CALL RECEIVE_REAL_32BIT( EISP )
  CALL RECEIVE_REAL_32BIT( CG(1) )
  CALL RECEIVE_REAL_32BIT( CG(2) )
  CALL RECEIVE_REAL_32BIT( CG(3) )

C-----C
C----- BOOSTERS MODULE -----C
C-----C
C-----C
C-----C

      IF ( T.LE.TSTG2 ) THEN
        CALL BTHRST(T,CG,EISP,PRESS,DLP,DLY,TOSEED,TBRK,IBTHR,
          FXT,FYT,FZT,MXT,MYT,MZT,MDOTT,THRV,THR)
      ENDIF

      IF ( ABS(T-TSTG1).LE.DTEPS ) THEN
        IBTHR = 1
      ENDIF

      CALL SEND_REAL_32BIT( FXT )
      CALL SEND_REAL_32BIT( FYT )
      CALL SEND_REAL_32BIT( FZT )
      CALL SEND_REAL_32BIT( MXT )
      CALL SEND_REAL_32BIT( MYT )
      CALL SEND_REAL_32BIT( MZT )
      CALL SEND_REAL_32BIT( MDOTT )

C-----C
C----- NOZZLE CONTROL UNIT MODULE -----C
C-----C
C           Models the response of the nozzle        C
C           control unit during first stage          C
C-----C

      IF ( T.LE.TSTG1 .AND. T.GT.TINHIB ) THEN
        CALL NCU(DLP,DLY,CMMD,DLPD,DLYD)
      ENDIF

* from frcthr
  CALL RECEIVE_REAL_32BIT( FOFF1(1) )
  CALL RECEIVE_REAL_32BIT( FOFF1(2) )
  CALL RECEIVE_REAL_32BIT( FOFF1(3) )
  CALL RECEIVE_REAL_32BIT( FOFF1(4) )
  CALL RECEIVE_REAL_32BIT( FOFF2(1) )

```

```

CALL RECEIVE_REAL_32BIT( FOFF2(2) )
CALL RECEIVE_REAL_32BIT( FOFF2(3) )
CALL RECEIVE_REAL_32BIT( FOFF2(4) )

```

```

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 VCSTH1( T, CG, TBURNM, IVCS, TOFFLT, TIMONV, DTOFFV,
    .           TVTAB, FOFF1, FOFF2, IVTAB, FXVCS, FYVCS, FZVCS,
    .           MXVCS, MYVCS, MZVCS, MDOTV )

```

```

ENDIF

```

```

* from ATMOS (delayed)

```

```

    CALL RECEIVE_REAL_32BIT( PRESS )

```

```

    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 )
    CALL SEND_REAL_32BIT( MDOTV )

```

```

    IF ( T.LE.TSTG1 ) THEN

```

```

        CALL INTEG ( DLP , DLPD , T , 19 )
        CALL INTEG ( DLY , DLYD , T , 20 )
        TOTDEL = SQRT ( DLP**2 + DLY**2 )
        IF ( TOTDEL.GT.PMAX ) THEN
            DLP = DLP*PMAX/TOTDEL
            DLY = DLY*PMAX/TOTDEL

```

```

        ENDIF

```

```

    ENDIF

```

```

C FIRST STAGE SEPARATION

```

```

    IF ( ABS(T-TSTG1).LE.DTEPS ) THEN
        DLP = 0.0
        DLY = 0.0
    ENDIF

```

```

* guidance

```

```

    CALL RECEIVE_SIGNED_16BIT( IVCS )

```

```

* bauto

```

```

    CALL RECEIVE_REAL_32BIT( CMMD(1) )
    CALL RECEIVE_REAL_32BIT( CMMD(2) )

```

```

* vcslog

```

```

    CALL RECEIVE_REAL_32BIT( DTOFFV(1) )
    CALL RECEIVE_REAL_32BIT( DTOFFV(2) )
    CALL RECEIVE_REAL_32BIT( DTOFFV(3) )
    CALL RECEIVE_REAL_32BIT( DTOFFV(4) )
    CALL RECEIVE_SIGNED_16BIT( IVTAB )
    CALL RECEIVE_REAL_32BIT( TBURNM )
    CALL RECEIVE_REAL_32BIT( TIMONV )
    CALL RECEIVE_REAL_32BIT( TOFFLT(1) )
    CALL RECEIVE_REAL_32BIT( TOFFLT(2) )
    CALL RECEIVE_REAL_32BIT( TOFFLT(3) )
    CALL RECEIVE_REAL_32BIT( TOFFLT(4) )
    CALL RECEIVE_REAL_32BIT( TVTAB )

```

```

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

```

```

C increment time

```

```

    TSTEP = TSTEP + 1.0
    T = TSTEP * DELT

```

```

C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

```

FILE: uuv22.19g/debug/uublk08.for

```

PROGRAM BLK08

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

REAL CIM(9)
REAL CMI(9)
REAL DELT
REAL DTR
REAL FRCX
REAL FXA
REAL FXACS
REAL FXT
REAL FXVCS
INTEGER IEXIT
REAL IXX
REAL IYY
REAL IZZ
REAL MASS_
REAL MRCX_
REAL MRCY
REAL MRCZ
REAL MX
REAL MXA
REAL MXACS
REAL MXT
REAL MXVCS
REAL MY
REAL MYA
REAL MYACS
REAL MYT
REAL MYVCS
REAL MZ
REAL MZA
REAL MZACS
REAL MZT
REAL MZVCS
REAL P
REAL PD
REAL PHI
REAL PHIICD
REAL PQR(3)
REAL PSI
REAL PSIICD
REAL Q
REAL QD
REAL QUAT(4)
REAL QUATD(4)
REAL QUATIC(4)
REAL QUATM
REAL R
REAL RD
REAL T
REAL THT
REAL THTICD
REAL TMP1
REAL TSTEP
REAL X_
REAL Y_
REAL Z_

$INCLUDE('~/INCLUDE/SSBLK08.DAT')

*LOOP* PROLOGUE

```

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

```
* initialization of variables not computed until end of blk00
  PD      = 0.0
  QD      = 0.0
  RD      = 0.0
```

```
C-----C
C----- MISSILE STATE INITIALIZATION MODULE -----C
C-----C
C              Initialize integrated missile states      C
C-----C
```

```
C      INITIAL MISSILE EULER ANGLES IN RADIANS
```

```
  PHI = PHIICD*DTR
  THT = THTICD*DTR
  PSI = PSIICD*DTR
```

```
C      COMPUTE INERTIAL TO MISSILE TRANSFORMATION MATRIX
```

```
  CALL MMK(PHI,1,THT,2,PSI,3,CIM)
```

```
  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)
```

```
  CALL BXI2FV(QUATM,CMI,QUATIC)
```

```
  PQR(1) = P
  PQR(2) = Q
  PQR(3) = R
```

```
  CALL FVDOT(PQR,TMP1,QUATIC,QUATD)
```

```
  QUAT(1) = QUATIC(1)
  QUAT(2) = QUATIC(2)
  QUAT(3) = QUATIC(3)
  QUAT(4) = QUATIC(4)
```

```
C      INITIALIZE MISSILE TRUTH STATES
```

```
  CALL INTEG1 ( P      , PD      , T , 12 )
  CALL INTEG1 ( Q      , QD      , T , 13 )
  CALL INTEG1 ( R      , RD      , T , 14 )
  CALL INTEG1 ( QUAT(1) , QUATD(1) , T , 15 )
  CALL INTEG1 ( QUAT(2) , QUATD(2) , T , 16 )
  CALL INTEG1 ( QUAT(3) , QUATD(3) , T , 17 )
  CALL INTEG1 ( QUAT(4) , QUATD(4) , T , 18 )
```

```
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
*LOOP* START
```

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

```
* tmsudriv is no longer needed -- IF/ENDIF and assignment deleted
```

- * The extrapolated states have been deleted. There should be no need
- * to look into the future.
- * Note that the states which follow have all been initialized, and each
- * is integrated at the end of the timestep.

```
CALL RECEIVE_REAL_32BIT( X_ )
CALL RECEIVE_REAL_32BIT( Y_ )
CALL RECEIVE_REAL_32BIT( Z_ )
CALL SEND_REAL_32BIT( P )
CALL SEND_REAL_32BIT( Q )
CALL SEND_REAL_32BIT( R )
CALL SEND_REAL_32BIT( QUAT(1) )
CALL SEND_REAL_32BIT( QUAT(2) )
CALL SEND_REAL_32BIT( QUAT(3) )
CALL SEND_REAL_32BIT( QUAT(4) )
```

```
CALL RECEIVE_REAL_32BIT( MASS )
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) )
```

- * initialization of these variables was added

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

```
CALL RECEIVE_REAL_32BIT( IXX )
CALL RECEIVE_REAL_32BIT( IYY )
CALL RECEIVE_REAL_32BIT( IZZ )
```

- C from BTHRST

```
CALL RECEIVE_REAL_32BIT( FXT )
CALL RECEIVE_REAL_32BIT( MXT )
CALL RECEIVE_REAL_32BIT( MYT )
CALL RECEIVE_REAL_32BIT( MZT )
```

- C from FRCTHR

```
CALL RECEIVE_REAL_32BIT( FRCX )
CALL RECEIVE_REAL_32BIT( MRCX )
CALL RECEIVE_REAL_32BIT( MRCY )
CALL RECEIVE_REAL_32BIT( MRCZ )
```

- C from AERO

```
CALL RECEIVE_REAL_32BIT( FXA )
CALL RECEIVE_REAL_32BIT( MXA )
CALL RECEIVE_REAL_32BIT( MYA )
CALL RECEIVE_REAL_32BIT( MZA )
```

- C from ACSTHR

```
CALL RECEIVE_REAL_32BIT( FXACS )
CALL RECEIVE_REAL_32BIT( MXACS )
CALL RECEIVE_REAL_32BIT( MYACS )
CALL RECEIVE_REAL_32BIT( MZACS )
```

- C from VCSTHR

```
CALL RECEIVE_REAL_32BIT( FXVCS )
CALL RECEIVE_REAL_32BIT( MXVCS )
CALL RECEIVE_REAL_32BIT( MYVCS )
CALL RECEIVE_REAL_32BIT( MZVCS )
```

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

```
CALL MISSLR(T,QUAT,CIM,P,Q,R,IXX,IYY,IZZ,MASS_,FXA,FXT,
.      FRCX,FXACS,FXVCS,
.      MXA,MXT,MRCX,MXACS,MXVCS,
.      MYA,MYT,MRCY,MYACS,MYVCS,MZA,MZT,MRCZ,MZACS,
.      MZVCS,X_,Y_,Z_,PD,QD,RD,
.      MX,MY,MZ,
.      QUATD)
```

```
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 ( P      , PD      , T , 12 )
CALL INTEG ( Q      , QD      , T , 13 )
CALL INTEG ( R      , RD      , T , 14 )
CALL INTEG ( QUAT(1) , QUATD(1) , T , 15 )
CALL INTEG ( QUAT(2) , QUATD(2) , T , 16 )
CALL INTEG ( QUAT(3) , QUATD(3) , T , 17 )
CALL INTEG ( QUAT(4) , QUATD(4) , T , 18 )

```

```

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

```

C increment time

```

TSTEP = TSTEP + 1.0
T = TSTEP * DELT

```

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

```

CALL RECEIVE_SIGNED_16BIT( IEXIT )

```

```

*LOOP* STOP
IF ( IEXIT.EQ.0 ) GO TO 1000

```

LOOP EPILOGUE

```

END

```

FILE: uu22.19g/debug/uublk09.for

PROGRAM BLK09

```

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

```

```

REAL ACSLEV
REAL AFXACS
REAL AFYACS
REAL AFZACS
REAL AMDOTA
REAL AMXACS
REAL AMYACS
REAL AMZACS
REAL BFXACS
REAL BFYACS
REAL BFZACS
REAL BMDOTA
REAL BMXACS
REAL BMYACS
REAL BMZACS
REAL CG(3)
REAL DELT
REAL DTACSA(4)
REAL FXACS
REAL FYACS
REAL FZACS
INTEGER IACSONA

```

```

INTEGER IEXIT
INTEGER ITHRES
REAL MDOTA
REAL MXACS
REAL MYACS
REAL MZACS
REAL T
REAL TATAB
REAL TKVON
INTEGER*4 TOSEED
REAL TSTEP

$INCLUDE ('^/INCLUDE/SSBLK09.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
CALL CW87

C   INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
CALL RANIT ( TOSEED )

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
*LOOP* START

C from MASSPR
CALL RECEIVE_REAL_32BIT( CG(1) )
CALL RECEIVE_REAL_32BIT( CG(2) )
CALL RECEIVE_REAL_32BIT( CG(3) )

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

      IF ( T.GE.TKVON ) THEN
        CALL ACSTHA(T,CG,ACSLEV,DTACSA,TATAB,TOSEED,
          . ITHRES,AFXACS,AFYACS,AFZACS,AMXACS,AMYACS,AMZACS,
          . AMDOTA,IACSONA)
      ENDIF

CALL RECEIVE_REAL_32BIT( BFXACS )
CALL RECEIVE_REAL_32BIT( BFYACS )
CALL RECEIVE_REAL_32BIT( BFZACS )
CALL RECEIVE_REAL_32BIT( BMXACS )
CALL RECEIVE_REAL_32BIT( BMYACS )
CALL RECEIVE_REAL_32BIT( BMZACS )
CALL RECEIVE_REAL_32BIT( BMDOTA )

      IF ( T.GE.TKVON ) THEN
        FXACS = BFXACS + AFXACS
        FYACS = BFYACS + AFYACS
        FZACS = BFZACS + AFZACS
        MXACS = BMXACS + AMXACS
        MYACS = BMYACS + AMYACS
        MZACS = BMZACS + AMZACS
        MDOTA = BMDOTA + AMDOTA
      ENDIF

CALL SEND_SIGNED_16BIT( IACSONA )

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 )
CALL SEND_REAL_32BIT( MDOA )

* kvauto
CALL RECEIVE_REAL_32BIT( ACSLEV )
CALL RECEIVE_SIGNED_16BIT( ITHRES )
* resthr
CALL RECEIVE_REAL_32BIT( DTACSA(1) )
CALL RECEIVE_REAL_32BIT( DTACSA(2) )
CALL RECEIVE_REAL_32BIT( DTACSA(3) )
CALL RECEIVE_REAL_32BIT( DTACSA(4) )
CALL RECEIVE_REAL_32BIT( TATAB )
C-----C
C----- TERMINATION LOGIC -----C
C-----C
C          Defines the simulation termination C
C          conditions C
C-----C

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

C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE
      END

FILE: uuv22.19g/debug/uublk10.for

PROGRAM BLK10
      IMPLICIT REAL      (A-H)
      IMPLICIT REAL      (O-Z)

      REAL ALT
      REAL DELT
      INTEGER IEXIT
      REAL PRESS
      REAL RHO
      REAL T
      REAL TSTEP
      REAL TSTG2
      REAL VSND

$INCLUDE ('~/INCLUDE/SSBLK10.DAT')

*LOOP* PROLOGUE
*      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
*LOOP* START
      CALL RECEIVE_REAL_32BIT( ALT )

```

```

C-----C
C----- ATMOSPHERE MODULE -----C
C-----C
C          Computes the atmospheric properties      C
C-----C

```

```

      IF ( T.LT.TSTG2 ) THEN
        CALL ATMOS1(T,ALT,RHO,PRESS,VSND)
      ENDIF

```

```

      CALL SEND_REAL_32BIT( PRESS )
      CALL SEND_REAL_32BIT( RHO )
      CALL SEND_REAL_32BIT( VSND )

```

```

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

```

```

C      increment time

      TSTEP = TSTEP + 1.0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

```

FILE: uuv22.19g/debug/uublk11.for

PROGRAM BLK11

```

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

```

```

      REAL AT(3)
      REAL CEI(9)
      REAL CG(3)
      DOUBLE PRECISION CIE(9)
      REAL CIM(9)
      DOUBLE PRECISION DELPHI
      DOUBLE PRECISION DELPSI
      DOUBLE PRECISION DELT
      DOUBLE PRECISION DELTHT
      DOUBLE PRECISION DELU
      DOUBLE PRECISION DELV
      DOUBLE PRECISION DELW
      DOUBLE PRECISION DT
      DOUBLE PRECISION DTEPS
      DOUBLE PRECISION DTR
      DOUBLE PRECISION GR(3)
      INTEGER*4 GYSEED
      INTEGER IEXIT
      DOUBLE PRECISION LATLP
      DOUBLE PRECISION LONGLP
      REAL MVR
      DOUBLE PRECISION MVRDOT
      REAL P
      REAL PD
      DOUBLE PRECISION PHI
      DOUBLE PRECISION PHIICD
      DOUBLE PRECISION PSI
      DOUBLE PRECISION PSIICD
      DOUBLE PRECISION PULSEA(3)
      REAL PULSEG(3)
      REAL Q

```

```

REAL QD
DOUBLE PRECISION QFRACA(3)
DOUBLE PRECISION QSI(4)
REAL R
REAL RD
DOUBLE PRECISION RMI(3)
DOUBLE PRECISION RMIR(3)
REAL RMIR_(3)
REAL SP
DOUBLE PRECISION SPHI
DOUBLE PRECISION SPSI
REAL SQ
REAL SR
DOUBLE PRECISION STHT
DOUBLE PRECISION SU
DOUBLE PRECISION SUD
DOUBLE PRECISION SV
DOUBLE PRECISION SVD
DOUBLE PRECISION SW
DOUBLE PRECISION SWD
DOUBLE PRECISION T
DOUBLE PRECISION TONAV
DOUBLE PRECISION TACCEL
DOUBLE PRECISION THT
DOUBLE PRECISION THTICD
REAL TI2M(9)
DOUBLE PRECISION TIMUDRIV
DOUBLE PRECISION TIMUPR
DOUBLE PRECISION TIMUSTEP
DOUBLE PRECISION TLIMU
DOUBLE PRECISION TNAV
INTEGER*4 TOSEED
DOUBLE PRECISION TST2ON
DOUBLE PRECISION TSTEP
DOUBLE PRECISION TSTG2
DOUBLE PRECISION TUPLK1
DOUBLE PRECISION TUPLK2
DOUBLE PRECISION UD
DOUBLE PRECISION VD
DOUBLE PRECISION VMI(3)
DOUBLE PRECISION VMIR(3)
REAL VMIR_(3)
DOUBLE PRECISION VP1
REAL VTT(3)
DOUBLE PRECISION WD
DOUBLE PRECISION X
DOUBLE PRECISION XD
DOUBLE PRECISION XYZE(3)
DOUBLE PRECISION XYZED(3)
DOUBLE PRECISION Y
DOUBLE PRECISION YD
DOUBLE PRECISION Z
DOUBLE PRECISION ZD

```

```
$INCLUDE ('^/INCLUDE/SSBLK11.DAT')
```

```
*LOOP* PROLOGUE
```

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

```
C   INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
    CALL RANIT ( TCSEED )
```

```

C-----C
C----- MISSILE STATE INITIALIZATION MODULE -----C
C-----C
C           Initialize integrated missile states      C
C                                                    C
C-----C

```

```
C   COORDINATE TRANSFORMATION MATRICES
```

```
  CALL MMK(SNGL(-90.0*DTR),1,SNGL(LATLP*DTR),2,
    SNGL(LONGLP*DTR),3,CEI)
```

```

  CIE(1) = CEI(1)
  CIE(2) = CEI(4)
  CIE(3) = CEI(7)
  CIE(4) = CEI(2)

```

```

CIE(5) = CEI(5)
CIE(6) = CEI(8)
CIE(7) = CEI(3)
CIE(8) = CEI(6)
CIE(9) = CEI(9)

C      COMPUTE MISSILE STATES IN INERTIAL FRAME

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

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

C      INITIAL MISSILE EULER ANGLES IN RADIANS.

PHI = PHIICD*DTR
THT = THTICD*DTR
PSI = PSIICD*DTR

C      ESTIMATED MISSILE POSITION AND VELOCITY

RMIR(1) = X
RMIR(2) = Y
RMIR(3) = Z
VMIR(1) = XD
VMIR(2) = YD
VMIR(3) = ZD

C      ESTIMATED MISSILE EULER ANGLES AND BODY RATES

SPHI = PHI
STHT = THT
SPSI = PSI

MVR = VPI

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

1000 CONTINUE
*LOOP* START

CALL RECEIVE_REAL_64BIT( XD )
CALL RECEIVE_REAL_64BIT( YD )
CALL RECEIVE_REAL_64BIT( ZD )
CALL RECEIVE_REAL_64BIT( X )
CALL RECEIVE_REAL_64BIT( Y )
CALL RECEIVE_REAL_64BIT( Z )
CALL RECEIVE_REAL_32BIT( P )
CALL RECEIVE_REAL_32BIT( Q )
CALL RECEIVE_REAL_32BIT( R )
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) )

CALL RECEIVE_REAL_64BIT( UD )
CALL RECEIVE_REAL_64BIT( VD )
CALL RECEIVE_REAL_64BIT( WD )
CALL RECEIVE_REAL_32BIT( PD )
CALL RECEIVE_REAL_32BIT( QD )
CALL RECEIVE_REAL_32BIT( RD )
CALL RECEIVE_REAL_64BIT( GR(1) )
CALL RECEIVE_REAL_64BIT( GR(2) )
CALL RECEIVE_REAL_64BIT( GR(3) )
CALL RECEIVE_REAL_64BIT( XYZE(1) )
CALL RECEIVE_REAL_64BIT( XYZE(2) )

```

```

CALL RECEIVE_REAL_64BIT( XYZE(3) )
CALL RECEIVE_REAL_64BIT( XYZED(1) )
CALL RECEIVE_REAL_64BIT( XYZED(2) )
CALL RECEIVE_REAL_64BIT( XYZED(3) )

CALL RECEIVE_REAL_32BIT( CG(1) )
CALL RECEIVE_REAL_32BIT( CG(2) )
CALL RECEIVE_REAL_32BIT( CG(3) )

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

IF ( TSTEP .GE. TIMUDRIV ) THEN
*
  TIMUDRIV = TIMUDRIV + TIMUSTEP
  CALL ACCEL(T,UD,VD,WD,P,Q,R,PD,QD,RD,CG,CIM,XD,YD,ZD,
    GR,GYSEED,QFRACA,PULSEA)
ENDIF

CALL RECEIVE_REAL_32BIT( PULSESEG(1) )
CALL RECEIVE_REAL_32BIT( PULSESEG(2) )
CALL RECEIVE_REAL_32BIT( PULSESEG(3) )

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

IF ( TSTEP .GE. TIMUDRIV ) THEN
  TIMUDRIV = TIMUDRIV + TIMUSTEP
  CALL IMUPRO(T,PULSESEG,PULSFA,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,DELPHI,DELTHT,DELPSI,DELU,DELV,DELW,GR,
  QS1,C1E,SP,SQ,SR,SUD,SVD,SWD,VMIR,RMIR,TI2M,SPHI,STHT,
  SPSI,SU,SV,SW,AT,VMI,RMI,TONAV)

C
C  TIME SINCE LAST INERTIAL MEASUREMENT UPDATE
C
C  DT      = T - TLIMU
C  TLIMU   = T
C  DT      = TIMUSTEP * DELT
C
C  INTEGRATE PERFORMANCE VELOCITY REMAINING USING NAVIGATION
C  OUTPUT

```

```

      IF ( T.LT.TST2ON .OR. T.GE.TSTG2 ) THEN
        MVRDOT = 0.0
      ELSE
        MVRDOT = -DBLE(SQRT(AT(1)**2 + AT(2)**2 + AT(3)**2))
      ENDIF

      MVR = MVR + DT*MVRDOT
      IF ( MVR.LT.0.0 ) MVR = 0.0

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)

      TACCEL = TIMUDRIV * DELT
      TIMUPR = TIMUDRIV * DELT
      TNAV = TIMUDRIV * DELT

    ENDIF

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

      RMIR_(1) = RMIR(1)
      RMIR_(2) = RMIR(2)
      RMIR_(3) = RMIR(3)
      VMIR_(1) = VMIR(1)
      VMIR_(2) = VMIR(2)
      VMIR_(3) = VMIR(3)

      CALL SEND_REAL_32BIT( AT(1) )
      CALL SEND_REAL_32BIT( AT(2) )
      CALL SEND_REAL_32BIT( AT(3) )
      CALL SEND_REAL_64BIT( RMIR(1) )
      CALL SEND_REAL_64BIT( RMIR(2) )
      CALL SEND_REAL_64BIT( RMIR(3) )
      CALL SEND_REAL_32BIT( RMIR_(1) )
      CALL SEND_REAL_32BIT( RMIR_(2) )
      CALL SEND_REAL_32BIT( RMIR_(3) )
      CALL SEND_REAL_64BIT( VMIR(1) )
      CALL SEND_REAL_64BIT( VMIR(2) )
      CALL SEND_REAL_64BIT( VMIR(3) )
      CALL SEND_REAL_32BIT( VMIR_(1) )
      CALL SEND_REAL_32BIT( VMIR_(2) )
      CALL SEND_REAL_32BIT( VMIR_(3) )
      CALL SEND_REAL_32BIT( SP )
      CALL SEND_REAL_32BIT( SQ )
      CALL SEND_REAL_32BIT( SR )

```

```

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

```

```

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

```

```

C      increment time
      TSTEP = TSTEP + 1.0D0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )
*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

```

FILE: uuuv22.19g/debug/uublk12.for

PROGRAM BLK12

```

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

```

```

INTEGER ACQD
DOUBLE PRECISION ACQRM
DOUBLE PRECISION AQRERR
DOUBLE PRECISION AQRU
DOUBLE PRECISION ASIG
DOUBLE PRECISION CMS(9)
DOUBLE PRECISION CSK1
DOUBLE PRECISION CSK2
DOUBLE PRECISION DELT
DOUBLE PRECISION DTEPS
INTEGER ESTATE
REAL FRMRAT
DOUBLE PRECISION GRT(5, 3)
DOUBLE PRECISION GRTEST(3)
INTEGER IBURN1
INTEGER IEXIT
DOUBLE PRECISION LAM(2)
DOUBLE PRECISION LAMD(2)
DOUBLE PRECISION LAMDOXX(2)
REAL LAMMO(2)
DOUBLE PRECISION LAMSEK(2)
INTEGER MACQ
REAL MAGR
REAL MAGRTR
REAL MAGV
INTEGER MCSO
DOUBLE PRECISION MGRDOT
INTEGER MTERM
DOUBLE PRECISION PI
REAL PITER
REAL PITERO
DOUBLE PRECISION RACQ

```

```

DOUBLE PRECISION RMIR(3)
REAL RREL(3)
REAL RRELO(3)
DOUBLE PRECISION RTEST(3)
DOUBLE PRECISION RTIC(5, 3)
INTEGER SEKTY
DOUBLE PRECISION SKOFF1
DOUBLE PRECISION SKOFF2
INTEGER*4 SKSEED
DOUBLE PRECISION SNRACQ
REAL SNRO
DOUBLE PRECISION SSK1
DOUBLE PRECISION SSK2
DOUBLE PRECISION T
DOUBLE PRECISION TAPUDRIV
DOUBLE PRECISION TAPUSTEP
INTEGER TERM
REAL TGE1
REAL TGE2AL
REAL TGIL
REAL TGO
DOUBLE PRECISION TGPUDRIV
DOUBLE PRECISION TGPUSTEP
REAL TI2M(9)
REAL TI2MO(9)
DOUBLE PRECISION TKFUDRIV
DOUBLE PRECISION TKFUSTEP
DOUBLE PRECISION TKVON
DOUBLE PRECISION TL2
INTEGER*4 TOSEED
INTEGER TRACK
REAL TRMTGO
DOUBLE PRECISION TSTEP
DOUBLE PRECISION TUPLK1
DOUBLE PRECISION TUPLK2
REAL URREL(3)
DOUBLE PRECISION VMIR(3)
REAL VREL(3)
REAL VRELO(3)
DOUBLE PRECISION VTEST(3)
DOUBLE PRECISION VTIC(5, 3)
DOUBLE PRECISION WFILT
REAL YAWER
REAL YAWERO
DOUBLE PRECISION ZFILT

$INCLUDE('^/INCLUDE/SSBLK12.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
  CALL CW87

C   INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
  CALL RANIT ( TOSEED )

C-----C
C----- MISSILE STATE INITIALIZATION MODULE -----C
C-----C
C           Initialize integrated missile states      C
C-----C

C   MISSILE TO SEEKER MATRIX ( INCLUDES MISALIGNMENT )
C   SEEKER MISALIGNMENT DIRECTIONS :
C       SKOFF1 = CONE ANGLE OFF NORMAL ( CURRENTLY UNDEFINED )
C       SKOFF2 = POLAR ANGLE

C   NOTE:  TRANSFORMATION INCLUDES 180 DEGREE ROTATION ABOUT Y-AXIS

SKOFF1 = 0.0
SKOFF2 = 2.0*PI*RAN0(TOSEED)

CSK1   = DCOS(SKOFF1)
SSK1   = DSIN(SKOFF1)
CSK2   = DCOS(SKOFF2)
SSK2   = DSIN(SKOFF2)
CMS(1) = -CSK1

```

```

CMS(2) = SSK1*CSK2
CMS(3) = SSK1*SSK2
CMS(4) = SSK1*SSK2
CMS(5) = CSK1
CMS(6) = SSK1*CSK2
CMS(7) = SSK1*CSK2
CMS(8) = SSK1*SSK2
CMS(9) = -CSK1

```

C INITIALIZE SEEKER PARAMETERS

```

CALL NORM(AQRU,0.0D0,SKSEED,AQRERR)
RACQ = ACQRM + AQRERR

```

```

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

```

1000 CONTINUE
LOOP START

```

CALL RECEIVE_REAL_64BIT( GRT(1,1) )
CALL RECEIVE_REAL_64BIT( GRT(1,2) )
CALL RECEIVE_REAL_64BIT( GRT(1,3) )
* CALL RECEIVE_REAL_64BIT( GRT(2,1) )
* CALL RECEIVE_REAL_64BIT( GRT(2,2) )
* CALL RECEIVE_REAL_64BIT( GRT(2,3) )
* CALL RECEIVE_REAL_64BIT( GRT(3,1) )
* CALL RECEIVE_REAL_64BIT( GRT(3,2) )
* CALL RECEIVE_REAL_64BIT( GRT(3,3) )
* CALL RECEIVE_REAL_64BIT( GRT(4,1) )
* CALL RECEIVE_REAL_64BIT( GRT(4,2) )
* CALL RECEIVE_REAL_64BIT( GRT(4,3) )
* CALL RECEIVE_REAL_64BIT( GRT(5,1) )
* CALL RECEIVE_REAL_64BIT( GRT(5,2) )
* CALL RECEIVE_REAL_64BIT( GRT(5,3) )
CALL RECEIVE_REAL_64BIT( VTIC(1,1) )
CALL RECEIVE_REAL_64BIT( VTIC(1,2) )
CALL RECEIVE_REAL_64BIT( VTIC(1,3) )
* CALL RECEIVE_REAL_64BIT( VTIC(2,1) )
* CALL RECEIVE_REAL_64BIT( VTIC(2,2) )
* CALL RECEIVE_REAL_64BIT( VTIC(2,3) )
* CALL RECEIVE_REAL_64BIT( VTIC(3,1) )
* CALL RECEIVE_REAL_64BIT( VTIC(3,2) )
* CALL RECEIVE_REAL_64BIT( VTIC(3,3) )
* CALL RECEIVE_REAL_64BIT( VTIC(4,1) )
* CALL RECEIVE_REAL_64BIT( VTIC(4,2) )
* CALL RECEIVE_REAL_64BIT( VTIC(4,3) )
* CALL RECEIVE_REAL_64BIT( VTIC(5,1) )
* CALL RECEIVE_REAL_64BIT( VTIC(5,2) )
* CALL RECEIVE_REAL_64BIT( VTIC(5,3) )
CALL RECEIVE_REAL_64BIT( RTIC(1,1) )
CALL RECEIVE_REAL_64BIT( RTIC(1,2) )
CALL RECEIVE_REAL_64BIT( RTIC(1,3) )
* CALL RECEIVE_REAL_64BIT( RTIC(2,1) )
* CALL RECEIVE_REAL_64BIT( RTIC(2,2) )
* CALL RECEIVE_REAL_64BIT( RTIC(2,3) )
* CALL RECEIVE_REAL_64BIT( RTIC(3,1) )
* CALL RECEIVE_REAL_64BIT( RTIC(3,2) )
* CALL RECEIVE_REAL_64BIT( RTIC(3,3) )
* CALL RECEIVE_REAL_64BIT( RTIC(4,1) )
* CALL RECEIVE_REAL_64BIT( RTIC(4,2) )
* CALL RECEIVE_REAL_64BIT( RTIC(4,3) )
* CALL RECEIVE_REAL_64BIT( RTIC(5,1) )
* CALL RECEIVE_REAL_64BIT( RTIC(5,2) )
* CALL RECEIVE_REAL_64BIT( RTIC(5,3) )

```

```

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

      IF ( TSTEP .GE. TGPUDRIV ) THEN

*      TGPUDRIV = TGPUDRIV + TGPUSTEP

C      GRTEST TEMPORARILY EQUAL TO GRT

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

      CALL OBRTARG(T,GRTEST,RTEST,VTEST,TL2)

      ENDIF

      CALL RECEIVE_REAL_32BIT( MAGRTR )
      CALL RECEIVE_REAL_64BIT( LAMDXX(1) )
      CALL RECEIVE_REAL_64BIT( LAMDXX(2) )

      CALL RECEIVE_REAL_64BIT( RMIR(1) )
      CALL RECEIVE_REAL_64BIT( RMIR(2) )
      CALL RECEIVE_REAL_64BIT( RMIR(3) )
      CALL RECEIVE_REAL_64BIT( VMIR(1) )
      CALL RECEIVE_REAL_64BIT( VMIR(2) )
      CALL RECEIVE_REAL_64BIT( VMIR(3) )
      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) )

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
C      model estimates                          C
C-----C

      IF ( TSTEP .GE. TGPUDRIV ) THEN

      TGPUDRIV = TGPUDRIV + TGPUSTEP

      CALL ESTREL(RTEST,VTEST,RMIR,VMIR,TI2M,CMS,ESTATE,RREL,VREL,
      .      MAGR,MAGV,URREL,MGRDOT,TGO,PITER,YAWER,LAMD)

      ENDIF

```

PITER0 = PITER
YAWER0 = YAWER

CALL SEND_REAL_32BIT(URREL(1))
CALL SEND_REAL_32BIT(URREL(2))
CALL SEND_REAL_32BIT(URREL(3))
CALL SEND_REAL_32BIT(RREL(1))
CALL SEND_REAL_32BIT(RREL(2))
CALL SEND_REAL_32BIT(RREL(3))
CALL SEND_REAL_32BIT(VREL(1))
CALL SEND_REAL_32BIT(VREL(2))
CALL SEND_REAL_32BIT(VREL(3))
CALL SEND_REAL_32BIT(TGO)
CALL SEND_REAL_32BIT(MAGR)
CALL SEND_REAL_32BIT(MAGV)
CALL SEND_REAL_32BIT(PITER0)
CALL SEND_REAL_32BIT(YAWER0)

* seeker

CALL SEND_SIGNED_16BIT(ACQD)
CALL RECEIVE_REAL_32BIT(FRMRAT)

CALL RECEIVE_REAL_32BIT(LAMMO(1))
CALL RECEIVE_REAL_32BIT(LAMMO(2))
CALL RECEIVE_REAL_32BIT(RRELO(1))
CALL RECEIVE_REAL_32BIT(RRELO(2))
CALL RECEIVE_REAL_32BIT(RRELO(3))
CALL RECEIVE_REAL_32BIT(SNRO)
CALL RECEIVE_REAL_32BIT(TI2MO(1))
CALL RECEIVE_REAL_32BIT(TI2MO(2))
CALL RECEIVE_REAL_32BIT(TI2MO(3))
CALL RECEIVE_REAL_32BIT(TI2MO(4))
CALL RECEIVE_REAL_32BIT(TI2MO(5))
CALL RECEIVE_REAL_32BIT(TI2MO(6))
CALL RECEIVE_REAL_32BIT(TI2MO(7))
CALL RECEIVE_REAL_32BIT(TI2MO(8))
CALL RECEIVE_REAL_32BIT(TI2MO(9))
CALL RECEIVE_REAL_32BIT(VRELO(1))
CALL RECEIVE_REAL_32BIT(VRELO(2))
CALL RECEIVE_REAL_32BIT(VRELO(3))

```

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

```

IF (TSTEP .GE. TKFUDRIV) THEN

TKFUDRIV = TKFUDRIV + TKFUSTEP

C CALL FILTER IF SNR IS SUFFICIENT

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

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

CALL KALMAN(TI2M, LAMMO, ASIG, SNRO, TGO, RRELO, VRELO,
. TI2MO, RACQ, MAGRTR, MAGR, MAGV, LAMSEK, LAMDXX, FRMRAT, CMS,
. MACQ, MCSO, MTERM, TRACK, TERM, TRMTGO, TGE1,
. TGE2AL, WFILT, ZFILT, LAM, LAMD, IBURN1, ACQD, ESTATE,
. PITER, YAWER, TGIL)

ENDIF
ENDIF

CALL SEND_REAL_32BIT(TGIL)
CALL SEND_REAL_32BIT(PITER)
CALL SEND_REAL_32BIT(YAWER)
CALL SEND_REAL_64BIT(LAMD(1))
CALL SEND_REAL_64BIT(LAMD(2))
CALL SEND_REAL_32BIT(TRMTGO)
CALL SEND_REAL_32BIT(TGE1)
CALL SEND_REAL_32BIT(TGE2AL)
CALL SEND_SIGNED_16BIT(IBURN1)
CALL SEND_SIGNED_16BIT(ESTATE)

```

      IF ( TSTEP.GE.TAPUDRIV ) THEN
        TAPUDRIV = TAPUDRIV + TAPUSTEP

        IF ( T.GE.TKVON ) THEN

          IF ( TGO.LE.TGE1 .AND. IBURN1.EQ.0 ) THEN
            * The IBURN1 assignment was moved from the partition with VCSLOG
              IBURN1 = 1
            ENDIF
          ENDIF
        ENDIF
      ENDIF

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

C      increment time

      TSTEP = TSTEP + 1.0D0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )
*LOOP* STOP

      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

```

FILE: uuv22.19g/debug/uublk13.for

```

PROGRAM BLK13

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

REAL ALFAP
REAL ALFAT
REAL ALFAY
REAL CA
REAL CG(3)
REAL CN
REAL DELT
REAL DTEPS
REAL FXA
REAL FYA
REAL FZA
INTEGER IAERO
INTEGER IEXIT
REAL MACH
REAL MVRWM
REAL MXA
REAL MYA
REAL MZA
REAL QA
REAL RHO
REAL T
REAL TBRK
INTEGER*4 TOSEED
REAL TSTEP
REAL TSTG1
REAL TSTG2
REAL VRWM(3)
REAL VSND
REAL XCP

$INCLUDE('~/INCLUDE/SSBLK13.DAT')

*LOOP* PROLOGUE

```

```

* INITIALIZE 80x87
  CALL CW87

C   INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
  CALL RANIT ( TOSEED )

* initialization for purpose of delaying receipt of actual values
  RHO      = 0.23769E-02
  VSND     = 1116.45
  VRWM(1)  = 0.0
  VRWM(2)  = 0.0
  VRWM(3)  = 0.0
  MVRWM    = 0.0

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
*LOOP* START

  CALL RECEIVE_REAL_32BIT( CG(1) )
  CALL RECEIVE_REAL_32BIT( CG(2) )
  CALL RECEIVE_REAL_32BIT( CG(3) )

C-----C
C----- AERODYNAMICS MODULE -----C
C-----C
C           Computes the aerodynamic forces and   C
C           moments                               C
C-----C

  IF ( T.LE.(TSTG2+DTEPS) ) THEN
    CALL AERO(T,VRWM,CG,MVRWM,RHO,VSND,IAERO,TBRK,QA,MACH,
      .   alfat,ALFAP,ALFAY,CA,CN,XCP,FXA,FYA,FZA,MXA,MYA,MZA)
  ENDIF

  IF ( ABS(T-TSTG1).LE.DTEPS ) THEN
    IAERO = 1
  ENDIF

* These values are delayed so that AERO can run sooner
  CALL RECEIVE_REAL_32BIT( RHO )
  CALL RECEIVE_REAL_32BIT( VSND )

  CALL SEND_REAL_32BIT( MACH )
  CALL SEND_REAL_32BIT( QA )
  CALL SEND_REAL_32BIT( FXA )
  CALL SEND_REAL_32BIT( FYA )
  CALL SEND_REAL_32BIT( FZA )
  CALL SEND_REAL_32BIT( MXA )
  CALL SEND_REAL_32BIT( MYA )
  CALL SEND_REAL_32BIT( MZA )

* These values are delayed so that AERO can run sooner
  CALL RECEIVE_REAL_32BIT( VRWM(1) )
  CALL RECEIVE_REAL_32BIT( VRWM(2) )
  CALL RECEIVE_REAL_32BIT( VRWM(3) )
  CALL RECEIVE_REAL_32BIT( MVRWM )

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

C   increment time

  TSTEP = TSTEP + 1.0
  T = TSTEP * DELT

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

```

```

      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

FILE: uuv22.19g/debug/uublk14.for

PROGRAM BLK14

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

      REAL ACSLEV
      REAL BFXACS
      REAL BFYACS
      REAL BFZACS
      REAL BMDOTA
      REAL BMXACS
      REAL BMYACS
      REAL BMZACS
      REAL CG(3)
      REAL DELT
      REAL DTACSB(4)
      INTEGER IACSONB
      INTEGER IEXIT
      INTEGER ITHRES
      REAL T
      REAL TATAB
      REAL TKVON
      INTEGER*4 TOSEED
      REAL TSTEP

$INCLUDE('~/INCLUDE/SSBLK14.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
      CALL CW87

C      INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
      CALL RANIT ( TOSEED )

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

      1000 CONTINUE
*LOOP* START

C from MASSPR
      CALL RECEIVE_REAL_32BIT( CG(1) )
      CALL RECEIVE_REAL_32BIT( CG(2) )
      CALL RECEIVE_REAL_32BIT( CG(3) )

C-----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

      IF ( T.GE.TKVON ) THEN

          CALL ACSTHB(T,CG,ACSLEV,DTACSB,TATAB,TOSEED,
.              ITHRES,BFXACS,BFYACS,BFZACS,BMXACS,BMYACS,BMZACS,
.              BMDOTA,IACSONB)

```

```

ENDIF

CALL SEND_REAL_32BIT( BFXACS )
CALL SEND_REAL_32BIT( BFYACS )
CALL SEND_REAL_32BIT( BFZACS )
CALL SEND_REAL_32BIT( BMXACS )
CALL SEND_REAL_32BIT( BMYACS )
CALL SEND_REAL_32BIT( BMZACS )
CALL SEND_REAL_32BIT( BMDOTA )
CALL SEND_SIGNED_16BIT( IACSONB )

* kvauto
CALL RECEIVE_REAL_32BIT( ACSLEV )
CALL RECEIVE_SIGNED_16BIT( ITHRES )
* resthr
CALL RECEIVE_REAL_32BIT( DTACSB(1) )
CALL RECEIVE_REAL_32BIT( DTACSB(2) )
CALL RECEIVE_REAL_32BIT( DTACSB(3) )
CALL RECEIVE_REAL_32BIT( DTACSB(4) )
CALL RECEIVE_REAL_32BIT( TATAB )
C-----C
C----- TERMINATION LOGIC -----C
C-----C
C                                     C
C                                     C
C                                     C
C-----C

C      increment time

      TSTEP = TSTEP + 1.0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

FILE: uuv22.19g/debug/uublk15.for

PROGRAM BLK15

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

INTEGER ACQD
REAL ACSLEV
REAL ADISTT(4, 3)
DOUBLE PRECISION ANVP
DOUBLE PRECISION DELT
REAL DTACSA(4)
REAL DTACSB(4)
DOUBLE PRECISION DTEPS
REAL DTOFFV(4)
DOUBLE PRECISION DTCSAMP
DOUBLE PRECISION DTVCSY(3)
DOUBLE PRECISION DTVCSY(3)
DOUBLE PRECISION FLTC(4)
DOUBLE PRECISION FLTCP
DOUBLE PRECISION FLTCY
INTEGER IACSON
INTEGER IACSONA
INTEGER IACSONB
INTEGER IBURN1
INTEGER IBURN2
INTEGER IBURN3
INTEGER IBURN4
INTEGER IBURNM
INTEGER ICMD
INTEGER IDIST
INTEGER IDMEAS
INTEGER IDROP

```

```

INTEGER IEXIT
INTEGER IPASSM
INTEGER ITHRES
INTEGER IVCS
INTEGER IVTAB
REAL IXX
REAL IYY
REAL IZZ
DOUBLE PRECISION LAMD(2)
REAL MAGV
DOUBLE PRECISION MASS
INTEGER MIDBRN
REAL PITER
REAL ROLLER
REAL SP
REAL SQ
REAL SR
DOUBLE PRECISION SW80
DOUBLE PRECISION T
DOUBLE PRECISION TAPUDRIV
DOUBLE PRECISION TAPUSTEP
REAL TATAB
REAL TBURNM
DOUBLE PRECISION TBURNP
DOUBLE PRECISION TBURNY
DOUBLE PRECISION TCMINV
DOUBLE PRECISION TCWAIT
DOUBLE PRECISION TDROP
REAL TGE1
DOUBLE PRECISION TGE2
REAL TGE2AL
DOUBLE PRECISION TGI1P
DOUBLE PRECISION TGI1Y
DOUBLE PRECISION TGI2P
DOUBLE PRECISION TGI2Y
DOUBLE PRECISION TGI3P
DOUBLE PRECISION TGI3Y
REAL TGIL
REAL TGO
DOUBLE PRECISION TGOFLM
REAL TIMONV
DOUBLE PRECISION TKVON
DOUBLE PRECISION TLAPS
DOUBLE PRECISION TMAUTO
DOUBLE PRECISION TNEXT
REAL TOFFLT(4)
DOUBLE PRECISION TOFLTM
DOUBLE PRECISION TPATON
DOUBLE PRECISION TRATON
REAL TRMTGO
DOUBLE PRECISION TSAH
DOUBLE PRECISION TSAL
DOUBLE PRECISION TSTEP
DOUBLE PRECISION TSTG2
REAL TVTAB
DOUBLE PRECISION TYATON
REAL VGM(3)
REAL YAWER

$INCLUDE ('~/INCLUDE/SSBLK15.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
  CALL CW87

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

1000 CONTINUE
*LOOP* START

  CALL RECEIVE_REAL_64BIT( MASS )
  CALL RECEIVE_REAL_32BIT( IXX )

```

```

CALL RECEIVE_REAL_32BIT( IYY )
CALL RECEIVE_REAL_32BIT( IZZ )
CALL RECEIVE_SIGNED_16BIT( IACSONA )
CALL RECEIVE_SIGNED_16BIT( IACSONB )
CALL RECEIVE_REAL_32BIT( SP )
CALL RECEIVE_REAL_32BIT( SQ )
CALL RECEIVE_REAL_32BIT( SR )
CALL RECEIVE_REAL_32BIT( TGO )
CALL RECEIVE_REAL_32BIT( MAGV )
CALL RECEIVE_SIGNED_16BIT( ACQD )
CALL RECEIVE_SIGNED_16BIT( IDROP )
CALL RECEIVE_SIGNED_16BIT( IBURND )
CALL RECEIVE_SIGNED_16BIT( IBURNM )
CALL RECEIVE_SIGNED_16BIT( IDMEAS )
CALL RECEIVE_REAL_32BIT( ADISTT(1,1) )
CALL RECEIVE_REAL_32BIT( ADISTT(1,2) )
CALL RECEIVE_REAL_32BIT( ADISTT(1,3) )
CALL RECEIVE_REAL_32BIT( ADISTT(2,1) )
CALL RECEIVE_REAL_32BIT( ADISTT(2,2) )
CALL RECEIVE_REAL_32BIT( ADISTT(2,3) )
CALL RECEIVE_REAL_32BIT( ADISTT(3,1) )
CALL RECEIVE_REAL_32BIT( ADISTT(3,2) )
CALL RECEIVE_REAL_32BIT( ADISTT(3,3) )
CALL RECEIVE_REAL_32BIT( ADISTT(4,1) )
CALL RECEIVE_REAL_32BIT( ADISTT(4,2) )
CALL RECEIVE_REAL_32BIT( ADISTT(4,3) )
CALL RECEIVE_REAL_32BIT( VGM(1) )
CALL RECEIVE_REAL_32BIT( VGM(2) )
CALL RECEIVE_REAL_32BIT( VGM(3) )
CALL RECEIVE_SIGNED_16BIT( IVCS )
CALL RECEIVE_REAL_32BIT( TGIL )
CALL RECEIVE_REAL_32BIT( PITER )
CALL RECEIVE_REAL_32BIT( YAWER )
CALL RECEIVE_REAL_64BIT( LAMD(1) )
CALL RECEIVE_REAL_64BIT( LAMD(2) )
CALL RECEIVE_REAL_32BIT( TRMTGO )
CALL RECEIVE_REAL_32BIT( TGE1 )
CALL RECEIVE_REAL_32BIT( TGE2AL )
CALL RECEIVE_SIGNED_16BIT( IBURN1 )
CALL RECEIVE_REAL_32BIT( ROLLER )

```

```

IF ( TSTEP .GE. TAPUDRIV ) THEN

```

```

*   TAPUDRIV = TAPUDRIV + TAPUSTEP

```

```

    IF ( T.GE.TKVON ) THEN

```

```

        CALL VCSTH2(T,FLTC,FLTCP,FLTCY,TOFFLT,TIMONV)

```

```

    ENDIF

```

```

ENDIF

```

```

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

```

```

IF ( TSTEP .GE. TAPUDRIV ) THEN

```

```

    TAPUDRIV = TAPUDRIV + TAPUSTEP

```

```

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

```

```

    IF ( T.GE.TKVON ) THEN

```

```

* (above) CALL VCSTH2(T,FLTC,FLTCP,FLTCY,TOFFLT,TIMONV)

```

```

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

```

```

C   NOSE FAIRING / BOOST ADAPTER SEPARATION.

```

```

      IF ( IDROP.EQ.1 .OR. (DABS(T-TDROP).LE.DTEPS) ) THEN
        IPASSM = 0
      ENDIF

      IF ( ( IACSONA .EQ. 1 ) .OR. ( IACSONB .EQ. 1 ) ) THEN
        IACSON = 1
      ELSE
        IACSON = 0
      ENDIF

      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)

      ENDIF

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)

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, TSAH, TOFFLT,
        .      TOFLTM, TBURNP, TBURNV, TGE2, TGI1P, TGI2P, TGI3P,
        .      TGI1Y, TGI2Y, TGI3Y, TIMONV, TGOFLM, TCWAIT, DTVCS,
        .      DTVCSP, 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
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

```

```

* kvauto
  CALL SEND_REAL_32BIT( ACSLEV )
  CALL SEND_SIGNED_16BIT( ITHRES )
* vcslog
  CALL SEND_REAL_32BIT( DTOFFV(1) )
  CALL SEND_REAL_32BIT( DTOFFV(2) )
  CALL SEND_REAL_32BIT( DTOFFV(3) )
  CALL SEND_REAL_32BIT( DTOFFV(4) )
  CALL SEND_SIGNED_16BIT( IVTAB )
  CALL SEND_REAL_32BIT( TPURNM )
  CALL SEND_REAL_32BIT( TMONV )
  CALL SEND_REAL_32BIT( TOFFLT(1) )
  CALL SEND_REAL_32BIT( TOFFLT(2) )
  CALL SEND_REAL_32BIT( TOFFLT(3) )
  CALL SEND_REAL_32BIT( TOFFLT(4) )
  CALL SEND_REAL_32BIT( TVTAB )
* resthr
  CALL SEND_REAL_32BIT( DTACSA(1) )
  CALL SEND_REAL_32BIT( DTACSA(2) )
  CALL SEND_REAL_32BIT( DTACSA(3) )
  CALL SEND_REAL_32BIT( DTACSA(4) )
  CALL SEND_REAL_32BIT( DTACSB(1) )
  CALL SEND_REAL_32BIT( DTACSB(2) )
  CALL SEND_REAL_32BIT( DTACSB(3) )
  CALL SEND_REAL_32BIT( DTACSB(4) )
  CALL SEND_REAL_32BIT( TATAB )
* needed by blk01 only
  CALL SEND_SIGNED_16BIT( MIDBRN )
  CALL SEND_SIGNED_16BIT( ICMD )
  CALL SEND_SIGNED_16BIT( IDIST )

  IF ( T.GE.TKVON ) THEN

* The ITHRES assignment was moved from the partition with ACSTHR
  ITHRES=0

  IF (IVTAB.EQ.1) THEN
* The IVTAB assignment was moved from the partition with VCSTHR
  IVTAB = 0

  IF (TBURNM.GE.TCMINV) TBURNM=0.0
  ENDIF

  ENDIF

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

C      increment time

      TSTEP = TSTEP + 1.0D0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )
*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

FILE: uuv22.19g/debug/uublk16.for

PROGRAM BLK16

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

```

```

INTEGER ACQD
REAL ACQRM
REAL AQRERR
REAL AQRU
REAL DELT
INTEGER FRMCNT
REAL FRMRAT
INTEGER IEXIT
INTEGER KFSF
REAL LAMM(2)
REAL LAMMO(2)
REAL LAMSEK(2)
INTEGER LATCH
REAL MAGRTR
REAL RACQ
REAL RREL(3)
REAL RRELO(3)
REAL SAMACQ
REAL SAMRAT
INTEGER*4 SKSEED
REAL SNR
REAL SNRO
REAL T
INTEGER TERM
REAL TI2M(9)
REAL TI2MO(9)
REAL TKFU
REAL TKFUDRIV
REAL TKFUSTEP
INTEGER*4 TOSEED
INTEGER TRACK
REAL TSPUDRIV
REAL TSPUSTEP
REAL TSTEP
REAL VREL(3)
REAL VRELO(3)

$INCLUDE('~/INCLUDE/SSBLK16.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
  CALL CW87

C   INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
  CALL RANIT ( TOSEED )

C-----C
C-----C  MISSILE STATE INITIALIZATION MODULE  -----C
C-----C
C           Initialize integrated missile states      C
C                                                    C
C-----C

C   INITIALIZE SEEKER PARAMETERS

  SAMRAT = SAMACQ
  CALL NORM(AQRU,0.0,SKSEED,AQRERR)
  RACQ   = ACQRM + AQRERR

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
*LOOP* START

  CALL RECEIVE_REAL_32BIT( MAGRTR )
  CALL RECEIVE_REAL_32BIT( LAMSEK(1) )
  CALL RECEIVE_REAL_32BIT( LAMSEK(2) )

  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) )
* may want to look at reordering these
CALL RECEIVE_REAL_32BIT( RREL(1) )
CALL RECEIVE_REAL_32BIT( RREL(2) )
CALL RECEIVE_REAL_32BIT( RREL(3) )
CALL RECEIVE_REAL_32BIT( VREL(1) )
CALL RECEIVE_REAL_32BIT( VREL(2) )
CALL RECEIVE_REAL_32BIT( VREL(3) )

CALL RECEIVE_SIGNED_16BIT( ACQD )

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

IF ( TSTEP .GE. TSPUDRIV ) THEN

*      TSPUDRIV = TSPUDRIV + TSPUSTEP

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

ENDIF

CALL SEND_REAL_32BIT( FRMRAT )

IF ( TSTEP .GE. TSPUDRIV ) THEN
  TSPUDRIV = TSPUDRIV + TSPUSTEP
C      EMULATE SIGNAL PROCESSING LAG

  LATCH = 1
  CALL SSPLAG(T,LAMM,RREL,VREL,TI2M,SNR,LATCH,KFSF,TKFU,
    LAMMO,RRELO,VRELO,TI2MO,SNRO)

ENDIF

IF ( TSTEP .GE. TKFUDRIV ) THEN
  TKFUDRIV = TKFUDRIV + TKFUSTEP
C      GET FILTER INPUTS APPROPRIATE FOR THIS PASS

  LATCH = -1
  CALL SSPLAG(T,LAMM,RREL,VREL,TI2M,SNR,LATCH,KFSF,TKFU,
    LAMMO,RRELO,VRELO,TI2MO,SNRO)

ENDIF

CALL SEND_REAL_32BIT( LAMMO(1) )
CALL SEND_REAL_32BIT( LAMMO(2) )
CALL SEND_REAL_32BIT( RRELO(1) )
CALL SEND_REAL_32BIT( RRELO(2) )
CALL SEND_REAL_32BIT( RRELO(3) )
CALL SEND_REAL_32BIT( SNRO )
CALL SEND_REAL_32BIT( TI2MO(1) )
CALL SEND_REAL_32BIT( TI2MO(2) )
CALL SEND_REAL_32BIT( TI2MO(3) )
CALL SEND_REAL_32BIT( TI2MO(4) )
CALL SEND_REAL_32BIT( TI2MO(5) )
CALL SEND_REAL_32BIT( TI2MO(6) )
CALL SEND_REAL_32BIT( TI2MO(7) )
CALL SEND_REAL_32BIT( TI2MO(8) )
CALL SEND_REAL_32BIT( TI2MO(9) )
CALL SEND_REAL_32BIT( VRELO(1) )
CALL SEND_REAL_32BIT( VRELO(2) )
CALL SEND_REAL_32BIT( VRELO(3) )

C-----C
C-----TERMINATION LOGIC -----C
C-----C

```

```

C                      Defines the simulation termination      C
C                      conditions                               C
C                      -----C
C
C      increment time
C
C      TSTEP = TSTEP + 1.0
C      T = TSTEP * DELT
C
C      CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
C
C      CALL RECEIVE_SIGNED_16BIT( IEXIT )
C*LOOP* STOP
C
C      IF ( IEXIT.EQ.0 ) GO TO 1000
C
C*LOOP* EPILOGUE
C
C      END

```

FILE: uuv22.19g/debug/uublk17.for

```

PROGRAM BLK17
IMPLICIT REAL      (A-H)
IMPLICIT REAL      (O-Z)

INTEGER ACQD
REAL ADISTT(4, 3)
REAL DELT
REAL DLV(3)
REAL DTCVU
REAL DTEPS
REAL DTMP1
REAL DTSPVC
INTEGER ESTATE
INTEGER FLIP
INTEGER IBURND
INTEGER IBURNM
INTEGER ICMU
INTEGER IDIST
INTEGER IDMEAS
INTEGER IDPASS
INTEGER IDROP
INTEGER IEXIT
INTEGER IMCEND
INTEGER IVCS
REAL MAGR
REAL MAGV
REAL MASS
INTEGER MIDBRN
REAL MVR
REAL MVS
REAL PITERO
REAL RMIR (3)
REAL ROLLER
INTEGER SEKTYP
REAL SNRACQ
REAL SNRO
REAL SP
REAL SQ
REAL SR
REAL T
REAL TAPUDRIV
REAL TAPUSTEP
REAL TCORV
REAL TDROP
REAL TFFE
REAL TGPUDRIV
REAL TGPUSTEP
REAL TI2M(9)
REAL TKFUDRIV
REAL TKFUSTEP
REAL TKVON
REAL TMGUID
INTEGER*4 TOSEED
REAL TSTEP

```

```

REAL TSTG2
REAL TTF
REAL TTFE
REAL URREL(3)
REAL UVS(3)
REAL VC(3)
REAL VG(3)
REAL VGM(3)
REAL VMIR(3)
REAL VS(3)
REAL VTT(3)
REAL VTTIC(3)
REAL VTTP(3)
REAL YAWERO

```

```
$INCLUDE ('^/INCLUDE/SSBLK17.DAT')
```

```
*LOOP* PROLOGUE
```

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

```
C   INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
  CALL RANIT ( TOSEED )
```

```
C   INITIALIZE
  VTTP(1) = VTTIC(1)
  VTTP(2) = VTTIC(2)
  VTTP(3) = VTTIC(3)
```

```

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

```

```
1000 CONTINUE
*LOOP* START
```

```

  CALL RECEIVE_REAL_32BIT( MASS_ )

  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) )
  CALL RECEIVE_REAL_32BIT( SP )
  CALL RECEIVE_REAL_32BIT( SQ )
  CALL RECEIVE_REAL_32BIT( SR )
  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( MVR )
  CALL RECEIVE_REAL_32BIT( VTT(1) )
  CALL RECEIVE_REAL_32BIT( VTT(2) )
  CALL RECEIVE_REAL_32BIT( VTT(3) )

```

```

C-----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 ( TSTEP .GE. TGPUDRIV ) THEN
```

```

*      TGPUDRIV = TGPUDRIV + TGPUSTEP

      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

CALL RECEIVE_REAL_32BIT( URREL(1) )
CALL RECEIVE_REAL_32BIT( URREL(2) )
CALL RECEIVE_REAL_32BIT( URREL(3) )
CALL RECEIVE_REAL_32BIT( MAGR )
CALL RECEIVE_REAL_32BIT( MAGV )
CALL RECEIVE_REAL_32BIT( PITERO )
CALL RECEIVE_REAL_32BIT( YAWERO )

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 ( TSTEP .GE. TGPUDRIV ) THEN

        TGPUDRIV = TGPUDRIV + TGPUSTEP

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

C      NOSE FAIRING / BOOST ADAPTER SEPARATION

          IF ( IDROP.EQ.1 .OR. (ABS(T-TDROP).LE.DTEPS) ) THEN
            IDROP = 2
          ENDIF

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

          ENDIF

        ENDIF

      ENDIF

* seeker
CALL RECEIVE_SIGNED_16BIT( ACQD )

CALL SEND_SIGNED_16BIT( IDROP )
CALL SEND_SIGNED_16BIT( IBURND )
CALL SEND_SIGNED_16BIT( IBURNM )
CALL SEND_SIGNED_16BIT( IDMEAS )
CALL SEND_REAL_32BIT( ADISTT(1,1) )
CALL SEND_REAL_32BIT( ADISTT(1,2) )
CALL SEND_REAL_32BIT( ADISTT(1,3) )
CALL SEND_REAL_32BIT( ADISTT(2,1) )
CALL SEND_REAL_32BIT( ADISTT(2,2) )
CALL SEND_REAL_32BIT( ADISTT(2,3) )
CALL SEND_REAL_32BIT( ADISTT(3,1) )
CALL SEND_REAL_32BIT( ADISTT(3,2) )
CALL SEND_REAL_32BIT( ADISTT(3,3) )
CALL SEND_REAL_32BIT( ADISTT(4,1) )
CALL SEND_REAL_32BIT( ADISTT(4,2) )
CALL SEND_REAL_32BIT( ADISTT(4,3) )
CALL SEND_REAL_32BIT( VGM(1) )
CALL SEND_REAL_32BIT( VGM(2) )
CALL SEND_REAL_32BIT( VGM(3) )

CALL RECEIVE_REAL_32BIT( SNRO )
CALL SEND_SIGNED_16BIT( IVCS )
CALL SEND_REAL_32BIT( UVS(1) )
CALL SEND_REAL_32BIT( UVS(2) )
CALL SEND_REAL_32BIT( UVS(3) )
CALL SEND_REAL_32BIT( MVS )

```

```

CALL RECEIVE_SIGNED_16BIT( ESTATE )
IF ( TSTEP .GE. TKFUDRIV ) THEN
    TKFUDRIV = TKFUDRIV + TKFUSTEP
C    CALL FILTER IF SNR IS SUFFICIENT
    IF ( SNRO.GE.SNRACQ .OR. SEKTYP.NE.2 ) THEN
        IF ( ESTATE.EQ.0 ) THEN
            ROLLER = 0.0
        ENDIF
    ENDIF
    ENDIF
    CALL SEND_REAL_32BIT( ROLLER )
    CALL RECEIVE_SIGNED_16BIT( MIDBRN )
    CALL RECEIVE_SIGNED_16BIT( ICMD )
    CALL RECEIVE_SIGNED_16BIT( IDIST )
    IF ( TSTEP.GE.TAPUDRIV ) THEN
        TAPUDRIV = TAPUDRIV + TAPUSTEP
        IF ( T.GE.TKVON ) THEN
            IF ( IBURNM .EQ. 0 ) THEN
                * The IBURNM assignment was moved from the partition with VCSLOG
                  IBURNM = 1
            ENDIF
        ENDIF
    ENDIF
    ENDIF
    ENDIF
C-----C
C-----C      TERMINATION LOGIC      C-----C
C-----C      Defines the simulation termination C-----C
C      conditions C-----C
C-----C
C    increment time
    TSTEP = TSTEP + 1.0
    T = TSTEP * DELT
C    CONTINUE LOOPING UNTIL ONE OR MORE EXIT CONDITIONS HAVE BEEN MET
    CALL RECEIVE_SIGNED_16BIT( IEXIT )
*LOOP* STOP
    IF ( IEXIT.EQ.0 ) GO TO 1000
*LOOP* EPILOGUE
    END

```

FILE: uu22.19g/debug/uublk18.for

PROGRAM BLK18

```

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

```

```

REAL ATHRF(4)
REAL CG(3)
REAL DELT
REAL DTOFF(4)
REAL FOFF1(4)
REAL FOFF2(4)
REAL FRCX
REAL FRCY
REAL FRCZ
INTEGER IEXIT
INTEGER IFTAB
REAL KM

```

```

REAL KN
INTEGER LENF(4)
REAL MACH
REAL MDOTF
REAL MRCX
REAL MRCY
REAL MRCZ
REAL QA
REAL T
REAL TBRK
REAL TFRCS
REAL TFTAB
REAL THF(8, 4)
REAL TMF(8, 4)
INTEGER*4 TOSEED
REAL TSTEP
REAL TSTG2
REAL VCMD(4)
REAL VCMDL(4)

$INCLUDE ('~/INCLUDE/SSBLK18.DAT')

*LOOP* PROLOGUE

* INITIALIZE 80x87
CALL CW87

C    INITIALIZE UNIFORM RANDOM NUMBER GENERATOR
CALL RANIT ( TOSEED )

* initialization for purpose of delaying receipt of actual values
QA      = 0.0
MACH    = 0.0

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
*LOOP* START

C from MASSPR
CALL RECEIVE_REAL_32BIT( CG(1) )
CALL RECEIVE_REAL_32BIT( CG(2) )
CALL RECEIVE_REAL_32BIT( CG(3) )

C-----C
C----- FRACS THRUSTER RESPONSE MODULE -----C
C-----C
C      Models forces and moments generated by C
C      the forward reaction control system C
C-----C

IF ( T.GE.TFRCS .AND. T.LE.TSTG2 ) THEN
  CALL FRCTHR(T,CG,MACH,QA,VCMD,VCMDL,DTOFF,TFTAB,IFTAB,
    .      TOSEED,TBRK,TMF,THF,LENF,FRCX,FRCY,FRCZ,MRCX,
    .      MRCY,MRCZ,MDOTF,ATHRF,KN,KM,FOFF1,FOFF2)
  ENDIF

CALL SEND_REAL_32BIT( FRCX )
CALL SEND_REAL_32BIT( FRCY )
CALL SEND_REAL_32BIT( FRCZ )
CALL SEND_REAL_32BIT( MRCX )
CALL SEND_REAL_32BIT( MRCY )
CALL SEND_REAL_32BIT( MRCZ )
CALL SEND_REAL_32BIT( MDOTF )
CALL SEND_REAL_32BIT( FOFF1(1) )
CALL SEND_REAL_32BIT( FOFF1(2) )
CALL SEND_REAL_32BIT( FOFF1(3) )
CALL SEND_REAL_32BIT( FOFF1(4) )
CALL SEND_REAL_32BIT( FOFF2(1) )
CALL SEND_REAL_32BIT( FOFF2(2) )
CALL SEND_REAL_32BIT( FOFF2(3) )

```

```

CALL SEND_REAL_32BIT( FOFF2(4) )

* from AERO (delayed)
CALL RECEIVE_REAL_32BIT( MACH )
CALL RECEIVE_REAL_32BIT( QA )

* fracs
CALL RECEIVE_REAL_32BIT( VCMD(1) )
CALL RECEIVE_REAL_32BIT( VCMD(2) )
CALL RECEIVE_REAL_32BIT( VCMD(3) )
CALL RECEIVE_REAL_32BIT( VCMD(4) )
CALL RECEIVE_SIGNED_16BIT( IFTAB )
CALL RECEIVE_REAL_32BIT( TFTAB )

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

C      increment time

      TSTEP = TSTEP + 1.0
      T = TSTEP * DELT

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

      CALL RECEIVE_SIGNED_16BIT( IEXIT )

*LOOP* STOP
      IF ( IEXIT.EQ.0 ) GO TO 1000

*LOOP* EPILOGUE

      END

```

FILE: uuv22.19g/debug/uuexosim.txt

```

# tfinal
real_32bit
1
145.0

```

FILE: uuv22.19g/dutility/makefile

FORFLAGS = code large optimize(3) storage(integer*2)

```

OBJECTS = \
SSKVAUTO.OBJ \
SSVCSLOG.OBJ \
UUACCEL.OBJ \
UUCW87.OBJ \
UUESTREL.OBJ \
UUFV2BXI.OBJ \
UUIMUPRO.OBJ \
UUIINTEG.OBJ \
UUIINTEGI.OBJ \
UUKALMAN.OBJ \
UUMASSPR.OBJ \
UUMCAUTO.OBJ \
UUMISSLT.OBJ \
UUMMK.OBJ \
UUMMLXY.OBJ \
UUNAVIG.OBJ \
UUNORM.OBJ \
UUOBTARG.OBJ \
UUOUTMES.OBJ \
UURAN.OBJ \
UURANO.OBJ \
UURANIT.OBJ \
UURELAT.OBJ \
UURES2R.OBJ \

```

```

UURESTHR.OBJ \
UUR0TMX.OBJ \
UUTABLE.OBJ \
UUTARGET.OBJ \
UUVCSH2.OBJ \
UUTIMER.OBJ

```

```
LIBRARY = UTILITY.LIB
```

```
$(LIBRARY):$(OBJECTS)
```

```

.for.obj:
    ftn286.new $< $(forflags)
    bnd286 $*.obj name($*) object($*.lnk) noload noprint
    rename $*.lnk over $*.obj
    submit :PFP:csd/lib( $(LIBRARY), $* )

```

```

clean:
    delete *.obj,*.lst,$(LIBRARY)

```

```
FILE: uuv22.19g/dutility/uuaccel.for
```

```

C-----
      SUBROUTINE ACCEL(T,UD,VD,WD,P,Q,R,PD,QD,RD,CG,CIM,XD,YD,ZD,GR,
      .
      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)
      REAL              CG(7)        , CIM(9)

```

```

DOUBLE PRECISION DCA(3)
DOUBLE PRECISION DUM1(3) , DUM2(3) , DUM3(3)
DOUBLE PRECISION DVEL(3) , GRAVG(3)
DOUBLE PRECISION GR(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)

REAL P, Q, R, PD, QD, RD

INTEGER*4 GYSEED

C LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

SAVE IACCEL

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSACCEL.DAT')
$INCLUDE('~/INCLUDE/SSCON15.DAT')
$INCLUDE('~/INCLUDE/SSCON16.DAT')

DATA IACCEL / 1 /

IF (IACCEL.EQ. 1) THEN

    IACCEL = 0

C INITIALIZE ACCELEROMETER PARAMETERS

    IF ( T.EQ. 0.0 ) THEN
        DRSGA = 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

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.0.000 ) 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 * ABO0(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

```

FILE: uuv22.19g/dutility/uucw87.for

```

subroutine cw87
integer*2 icw87
call stcw87(icw87)
icw87 = icw87 .and. #ff7ah
call ldcw87(icw87)
end

```

FILE: uuv22.19g/dutility/uuestrel.for

```

C-----
SUBROUTINE ESIREL(RTEST,VTEST,RMIR,VMIR,TI2M,CMS,ESTATE,RREL,
.             'REL,MAGR,MACV,URREL,MGRDOT,TGO,PITER,YAWER,
.             L,MD)
C-----
C
C      SUBROUTINE NAME :      ESTREL
C
C      AUTHOR(S) :          T. THOMPSON
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,PMIR,VMIR,TI2M,CMS,ESTATE
C
C      OUTPUTS :          RREL,VREL,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                        E. HILL /     - CR # 081
C                        R. RHYNE
C                        R. RHYNE      - CR # 088
C                        R. RHYNE      - CR # 093

```

```

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

```

```

      DOUBLE PRECISION  CMS(9)      , LAMD(2)      , LAMSKE(2)
      REAL              MAGR        , MAGV        , VRDRR
      REAL              PITER       , YAWER        , TGO
      DOUBLE PRECISION  MGRDOT
      DOUBLE PRECISION  RELM(3)     , RELS(3)      , RMIR(3)
      DOUBLE PRECISION  RTEST(3)
      REAL              TI2M(9)
      REAL              URREL(3)    , VREL(3)      , RREI(3)
      DOUBLE PRECISION  VELM(3)     , VEL(3)
      DOUBLE PRECISION  VMIR(3)    , VTEST(3)

```

```

      INTEGER          ESTATE

```

```

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

```

```

      RREL(1) = RTEST(1) - RMIR(1)
      RREL(2) = RTEST(2) - RMIR(2)
      RREL(3) = RTEST(3) - RMIR(3)

```

```

      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

```

```

      VREL(1) = VTEST(1) - VMIR(1)
      VREL(2) = VTEST(2) - VMIR(2)
      VREL(3) = VTEST(3) - VMIR(3)

```

```

      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/(MAGR**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)

```

```

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

```

```

C      COMPUTE ESTIMATED LINE OF SIGHT ERRORS

      LAMSKE(1) = DATAN2(-RELS(3),RELS(1))
      LAMSKE(2) = DATAN2( 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

```

FILE: uuv22.19g/dutility/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

```

```

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

      DIMENSION FV ( 4 ),  B ( 9 )

      PARAMETER (R1 = 1.0, R2 = 2.0)

      FV1SQ = FV(1)*FV(1)
      FV2SQ = FV(2)*FV(2)
      FV3SQ = FV(3)*FV(3)
      FV4SQ = FV(4)*FV(4)

      FVSQ = FV1SQ + FV2SQ + FV3SQ + FV4SQ

      IF( FVSQ .GT. 0.0 ) THEN
* FTN286 X415 OPTIMIZE(3)
99999 CONTINUE
        T1 = R2/FVSQ

        T2 = FV(3)*FV(4)
        T3 = FV(1)*FV(2)
        B(2) = T1*( T3 + T2 )
        B(4) = T1*( T3 - T2 )

        T2 = FV(2)*FV(4)
        T3 = FV(1)*FV(3)
        B(7) = T1*( T3 + T2 )
        B(3) = T1*( T3 - T2 )

        T2 = FV(1)*FV(4)
        T3 = FV(2)*FV(3)
        B(6) = T1*( T3 + T2 )
        B(8) = T1*( T3 - T2 )

```

```

      T2 = T1*FV4SQ - R1
      B(1) = T1*FV1SQ + T2
      B(5) = T1*FV2SQ + T2
      B(9) = T1*FV3SQ + T2
    ENDIF

```

```

    RETURN
  END

```

FILE: uu22.19g/dutility/uuimupro.for

```

C-----
      SUBROUTINE IMUPRO(T,PULSESEG,PULSESEA,DELPHI,DELTHI,DELPSI,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,PULSESEG,PULSESEA
C
C      OUTPUTS :            DELPHI,DELTHI,DELPSI,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
C                          B. HILL - CR # 093
C-----
C
C      IMPLICIT DOUBLE PRECISION      (A-H)
C      IMPLICIT DOUBLE PRECISION      (O-Z)
C
C      DOUBLE PRECISION  PULSESEA(3)
C      REAL              PULSESEG(3)
C
C      * DATA INITIALIZATION
C      $INCLUDE('~/INCLUDE/SSCON54.DAT')
C
C      GYRO OUTPUT COMPENSATION
C
C      CALCULATE DELTA ANGLES
C
C      IF ( PERPG.GT.0.0 ) THEN
C        DELPHS = PULSESEG(1)*PERPG
C        DELTHS = PULSESEG(2)*PERPG
C        DELPSS = PULSESEG(3)*PERPG
C      ELSE
C        DELPHS = PULSESEG(1)
C        DELTHS = PULSESEG(2)
C        DELPSS = PULSESEG(3)
C      END IF
C
C      COMPENSATE SENSED DELTA ANGLES FOR SCALE FACTOR ERRORS
C
C      DELPH = DELPHS*SFCGX
C      DELTH = DELTHS*SFCGY
C      DELPS = DELPSS*SFCGZ
C
C      COMPENSATE SENSED DELTA ANGLES FOR GYRO MISALIGNMENTS
C
C      DELPHI = DELPH - DELTH*PSIGP + DELPS*THIGP
C      DELTHI = DELPH*PSIGP + DELTH - DELPS*PHIGP
C      DELPSI = -DELPH*THIGP + 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*SFCAY

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*DELUM )
        DELW = DELWM - 0.5 * ( DELTHT*DELUM - DELPHI*DELVM )
      END IF

      RETURN
      END

```

FILE: uu22.19g/dutility/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) + 0.500*DT*(XDOT+XDOTL(I))
C      X = XINT(I)

```

```

TINT(I) = T
XDOTL(I) = XDOT
C   TEMPORARY CODE TO NORMALIZE QUATERNION AFTER 4TH COMPONENT IS REVISED
      IF ( I.EQ.18 ) THEN
        DTMP = DSQRT ( XINT(15)**2 + XINT(16)**2 + XINT(17)**2 +
          .           XINT(18)**2 )
        XINT(15) = XINT(15) / DTMP
        XINT(16) = XINT(16) / DTMP
        XINT(17) = XINT(17) / DTMP
        XINT(18) = XINT(18) / DTMP
      END IF

      RETURN
      END

```

FILE: uuv22.19g/dutility/uuintegi.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 :           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

```

FILE: uuv22.19g/dutility/uukalman.for

```

C-----
C   SUBROUTINE KALMAN(T, TI2M, LAMMO, ASIG, SNRO, TGO, RRELO, VRELO, TI2MO,
C   .               RACQ, MAGRTR, MAGR, MAGV, LAMSEK, LAMDX, FRMRAT, CMS,
C   .               MACQ, MCSO, MTERM, TRACK, TERM, TRMTGO, TGE1,
C   .               TGE2AL, WFILT, ZFILT, LAM, LAMD, IBURN1, ACQD, ESTATE,
C   .               PITER, YAWER, TGIL)
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, LAMDX,

```

```

C          FRMRAT,CMS,MACQ,MCSO,MTERM
C
C          OUTPUTS :          TRMTGO,TGE1,TGE2AL,WFILT,ZFILT,LAM,
C                               LAMD,IBURN1,ACQD,PITER,YAWER
C
C          BOTH :            ESTATE,TRACK,TERM,TGIL
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 DOUBLE PRECISION      (A-H)
IMPLICIT DOUBLE PRECISION      (O-Z)

REAL          SNRO              , FRMRAT

DOUBLE PRECISION CSSHFT(3)      , TMSHFT(3)      , TKSHFT(3)
DOUBLE PRECISION LAMSEK(2)      , LAMDXX(2)      , MAGRSQ
DOUBLE PRECISION LAM(2)         , LAMD(2)         , MAGRO
DOUBLE PRECISION RATE(6)
REAL          VRELO(3)
REAL          RRELO(3)          , LAMMO(2)          , TI2MO(9)
REAL          TI2M(9)
REAL          MAGR              , MAGV              , MAGRTR
REAL          PITER             , YAWER             , TGE1
REAL          TGE2AL            , TGIL              , TGO
REAL          TRMTGO
DOUBLE PRECISION CMS(9)

INTEGER       SEKTYP            , ACQD
INTEGER       ESTATE            , TRACK            , TERM

```

```

C          LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

```

```

          SAVE          IKALMN

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSKALMAN.DAT')
$INCLUDE('~/INCLUDE/SSCON11.DAT')
$INCLUDE('~/INCLUDE/SSCON12.DAT')
$INCLUDE('~/INCLUDE/SSCON48.DAT')
$INCLUDE('~/INCLUDE/SSCON50.DAT')
$INCLUDE('~/INCLUDE/SSCON55.DAT')
$INCLUDE('~/INCLUDE/SSCON56.DAT')
$INCLUDE('~/INCLUDE/SSCON57.DAT')

```

```

          DATA IKALM: / 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)
      PLMDH = PLMDH1
      YLMDH1 = (RRELO(1)*VRELO(2) - RRELO(2)*VRELO(1))/
      (RRELO(1)**2 + RRELO(2)**2)
      YLMDH = 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 ACQUISITION MODE

          IF ( ACQD.EQ.0 .AND. MAGRTR.LE.RACQ ) THEN
              ESTATE = 0
              ACQD = 1
              TRMTGO = TGO - (MAGR - RNGTRM)/MAGV
              TGIL = TRMTGO + TBWAIT
              TGE2AL = TGIL + DTVCS2
              CALL OUTMES(0601,T,0.0D0)
          ENDIF

C          COMPUTE THE SEEKER DATA RATE

          IF ( TRACK.EQ.1 ) THEN
              TRACK = 2
              TGE1 = TGO - (RNHITS + ILAG)/FRMRAT
              IBURN1 = 0
              CALL OUTMES(0602,T,0.0D0)
          ELSEIF ( TERM.EQ.1 ) THEN
              TERM = 2
              CALL OUTMES(0603,T,0.0D0)
          ENDIF

          ENDIF

C          USE TRUE LOS ANGLES AND RATES WITH PERFECT SEEKER MODEL

          * cannot allow this partition to set angular errors, even if perfect
          * seeker is used sometime in future

          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 = VP7LO(1)

```

```

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

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

C      RECONSTRUCT MEASURED LOS VECTOR IN SEEKER FRAME

TANPCH =   DBLE ( TAN ( LAMMO(1) ) )
TANYAW =   DBLE ( TAN ( LAMMO(2) ) )

XLOSS  =   1.0D0 / DSQRT ( 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  =   DATAN2 ( -ZLOSI , XLOSI )
YLAMM  =   DATAN2 ( 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  = 0.0D0
      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.FRMROT.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
          CALL OUTMES(0604,T,MAGRO)
          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

          CALL OUTMES(0605,T,MAGRO)
          KFMODE = 3

```

```

MAGRSQ = MAGRO**2
TGOSQ  = TGO**2
PP11   = PP11 + TKSHFT(3)**2/MAGRSQ
PY11   = PY11 + TKSHFT(2)**2/MAGRSQ
PP22   = PP22 + TKSHFT(3)**2/(MAGRSQ*TGOSQ)
PY22   = PY22 + TKSHFT(2)**2/(MAGRSQ*TGOSQ)
ENDIF

IF ( KFMODE.GE.3 .AND. IFPAS.GE.IDNINT(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
  CALL OUTMES(0606,T,MAGRO)
  KFMODE = 4
  MAGRSQ = MAGRO**2
  TGOSQ  = TGO**2
  PP11   = PP11 + CSSHFT(3)**2/MAGRSQ
  PY11   = PY11 + CSSHFT(2)**2/MAGRSQ
  PP22   = PP22 + CSSHFT(3)**2/(MAGRSQ*TGOSQ)
  PY22   = PY22 + CSSHFT(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
  CALL OUTMES(0607,T,MAGRO)
  KFMODE = 5
  TGE2AL = TGO - RNHITS/FRMRAT
  TRMTGO = TGO - RNHITS/FRMRAT
  MAGRSQ = MAGRO**2
  TGOSQ  = TGO**2
  PP11   = PP11 + TMSHFT(3)**2/MAGRSQ
  PY11   = PY11 + TMSHFT(2)**2/MAGRSQ
  PP22   = PP22 + TMSHFT(3)**2/(MAGRSQ*TGOSQ)
  PY22   = PY22 + TMSHFT(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 / 4.0D0
Q12      = RW * DTKF / 2.0D0
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.0.0D0 .AND. DTKF.GT.0.0D0 ) THEN
    WFILT = DSQRT ( AKP21 / DTKF )
    ZFILT = AKP11 * WFILT / ( 2.0D0 * 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.0.0D0 ) THEN
    DLPLMD = ( PLMDF - PLMDFP )
    DLYLMD = ( YLMDF - YLMDFP )
ELSE
    DLPLMD = 0.0D0
    DLYLMD = 0.0D0
ENDIF

PLMDFP = PLMDF
YLMDFP = YLMDF

C   EXTRAPOLATE FILTERED INERTIAL FRAME STATES TO CURRENT TIME

PLAMH1 = PLAMH + DTKF * ( PLAMDH + 0.5D0*DTKF*DLPLMD )
YLAMH1 = YLAMH + DTKF * ( YLAMDH + 0.5D0*DTKF*DLYLMD )

PLMDH1 = PLAMDH + DLPLMD
YLMDH1 = 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.0.0D0 ) THEN
    DLPLMD = DLPLMD * SPLAG / DTKF
    DLYLMD = DLYLMD * SPLAG / DTKF
ELSE
    DLPLMD = 0.0D0
    DLYLMD = 0.0D0
ENDIF

PLAMF = PLAMH + SPLAG * ( PLAMDH + 0.5D0*SPLAG*DLPLMD )
YLAMF = YLAMH + SPLAG * ( YLAMDH + 0.5D0*SPLAG*DLYLMD )

PLAMDF = PLAMDH + DLPLMD
YLAMDF = YLAMDH + DLYLMD

C   RECONSTRUCT FILTERED LOS VECTOR IN INERTIAL FRAME

TANPCH = DTAN ( PLAMF )

```

```

TANYAW = DTAN ( YLAMF )
COSPSQ = DCOS ( PLAMF ) **2
COSYSQ = DCOS ( YLAMF ) **2

XLOSI = 1.000 / DSQRT ( 1.000 + 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

XLOSDS = CMS(1)*XLOSDM + CMS(4)*YLOSDM + CMS(7)*ZLOSDM
YLOSDS = CMS(2)*XLOSDM + CMS(5)*YLOSDM + CMS(8)*ZLOSDM
ZLOSDS = CMS(3)*XLOSDM + CMS(6)*YLOSDM + CMS(9)*ZLOSDM

C   DETERMINE LOS ANGLES IN SEEKER FRAME

LAM(1) = DATAN2 ( -ZLOSS , XLOSS )
LAM(2) = DATAN2 ( YLOSS , XLOSS )

C   DETERMINE LOS ANGULAR RATES IN SEEKER FRAME

TANPCH = DTAN ( LAM(1) )
TANYAW = DTAN ( LAM(2) )
COSPSQ = DCOS ( LAM(1) ) **2
COSYSQ = DCOS ( LAM(2) ) **2

LAMD(1) = ( - ZLOSDS - XLOSDS*TANPCH ) * COSPSQ / XLOSS
LAMD(2) = ( YLOSDS - XLOSDS*TANYAW ) * COSYSQ / XLOSS

C   DETERMINE ATTITUDE ERRORS

IF ( ESTATE .EQ. 0 ) THEN
  PITER = LAM(1)
  YAWER = -LAM(2)
* following line moved to partition with MCGUID
*   ROLLER = 0.0
ENDIF

RETURN
END

```

FILE: uuv22.19g/dutility/uumasspr.for

```

C-----
      SUBROUTINE MASSPR(T,MDOTT,MDOTF,MDOTA,MDOTV,MASS,EISP,TBRK,IMASS,
      .                MDOT,WEIGHT,WDOTTP,WDOTFR,WDOTKV,WDOTTI,CG,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   - CR # 087
C                    B. HILL   - CR # 089
C                    B. HILL   - CR # 093
C
C -----

```

```

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

REAL      CG(3)      , EISP
REAL      CGX(20)    , CGY(20)
REAL      CGZ(20)    , INERXX(20) , INERYY(20)
REAL      INERZZ(20)
REAL      IXX      , IYY      , IZZ
DOUBLE PRECISION MASS
REAL      SNGLMASS
DOUBLE PRECISION MDOT
REAL      MASST1(20) , MASST2(20)
REAL      MDOTA      , MDOTF      , MDOTT
REAL      MDOTV

```

```

C LOCAL DATA USED TO HOLD CONSTANTS, VARIABLES AND INITIALIZATION FLAG

```

```

SAVE          IDATIN , BISP

```

```

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSMASSPR.DAT')
$INCLUDE('^/INCLUDE/SSCON22.DAT')
$INCLUDE('^/INCLUDE/SSCON23.DAT')
$INCLUDE('^/INCLUDE/SSCON45.DAT')
$INCLUDE('^/INCLUDE/SSCON58.DAT')

```

```

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

```

```

IF (IMASS .EQ. 1) THEN

```

```

    IMASS = 0

```

```

    IF (IDATIN .EQ. 1) THEN

```

```

        IDATIN = 0

```

```

        IF (T .LE. TSTG1) THEN

```

```

            BISP = BISP1

```

```

            EISP = BISP*WPROP1/(WPROP1 + WINS1)

```

```

        ELSEIF (T .LE. TSTG2) THEN

```

```

            BISP = BISP2

```

```

            EISP = BISP*WPROP2/(WPROP2 + WINS2)

```

```

        ELSE

```

```

            BISP = 0.0

```

```

            EISP = 0.0

```

```

        ENDIF

```

```

    ELSEIF (T .LE. TSTG2) THEN

```

```

C      CALCULATE BOOSTER SPECIFIC IMPULSE AT FIRST STAGE
C      SEPARATION

```

```

        BISP = BISP2

```

```

        EISP = BISP*WPROP2/(WPROP2 + WINS2)

```

```

    ELSE

```

```

C      ZERO BOOSTER SPECIFIC IMPULSE AFTER SECOND STAGE
      BISP = 0.0
      EISP = 0.0
      ENDIF
    ENDIF

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

C      CONVERT MASS TO WEIGHT
      WEIGHT = MASS*XMTOF

C      CALCULATE WEIGHT EXPULSION RATES
      IF ( T.LE.TSTG2 ) THEN
        WDOTTP = -MDOTT*XMTOF*EISP/BISP
        WDOTTI = MDOTT*XMTOF*EISP
      ELSE
        WDOTTP = 0.0
        WDOTTI = 0.0
      ENDIF

      WDOTFR = MDOTF*XMTOF
      WDOTKV = (-MDOTF - MDOTA - MDOTV)*XMTOF

C      CALCULATE MISSILE CENTER OF GRAVITY COMPONENTS
      SNGLMASS = SNGL(MASS)
      CALL TABLE(MASST1,CGX,SNGLMASS,CG(1),20,ICG)
      CALL TABLE(MASST1,CGY,SNGLMASS,CG(2),20,ICG)
      CALL TABLE(MASST1,CGZ,SNGLMASS,CG(3),20,ICG)

C      CALCULATE MISSILE MOMENT OF INERTIA
      CALL TABLE(MASST2,INERXX,SNGLMASS,IXX,20,III)
      CALL TABLE(MASST2,INERY,SNGLMASS,IYY,20,III)
      CALL TABLE(MASST2,INERZZ,SNGLMASS,IZZ,20,III)

      RETURN
      END

FILE: uuv22.19g/dutility/uumcauto.for

C-----
SUBROUTINE MCAUTO(T,IXX,IYY,IZZ,SP,SQ,SR,ROLLER,PITER,YAWER,IDIST,
.             IACSON,IBURN,IBURNM,IDMEAS,IPASSM,ICMD,TRATON,
.             TPATON,TYATON,DTSAMP,TSAL,TAH,TLAPS,ITHRES,
.             ANVP,ACSLEV,TMAUTO)
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,IBURN,IBURNM,IDMEAS
C
C      OUTPUTS     :      ICMD,TRATON,TPATON,TYATON,DTSAMP,TSAL,TAH,
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
C
C
CD. SMITH - CR # 092
B. HILL - CR # 093

```

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

DOUBLE PRECISION  II(3)        , ANGACL(3,4,10), OMEGAI(3)
DOUBLE PRECISION  OMEGA(3)      , TBURNA(3)    , MOMARM(3)
DOUBLE PRECISION  AERROR(3)     , OMEGAD       , AACCEL(3,4)
REAL              IXX           , IYY          , IZZ
REAL              SP            , SQ           , SR
REAL              ROLLER        , PITER        , YAWER
REAL              ACSLEV

INTEGER           IMCPAS(3,4)

```

```

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSMCAUTO.DAT')
$INCLUDE('^/INCLUDE/SSCON59.DAT')
$INCLUDE('^/INCLUDE/SSCON60.DAT')
$INCLUDE('^/INCLUDE/SSCON01.DAT')
$INCLUDE('^/INCLUDE/SSCON02.DAT')
$INCLUDE('^/INCLUDE/SSCON05.DAT')
$INCLUDE('^/INCLUDE/SSCON07.DAT')
$INCLUDE('^/INCLUDE/SSCON08.DAT')
$INCLUDE('^/INCLUDE/SSCON19.DAT')

```

```

IF ( IPASSM.EQ.0 ) THEN

```

```

C      INITIALIZE ACCELERATION TABLE, PULSE FLAGS, AND PULSE TIMES

```

```

MOMARM(1) = RIARM
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

```

```

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
  RLLERO = ROLLER + SP**2/(2.*OMEGAD)
ELSE
  RLLERO = ROLLER
ENDIF
TBACS = DSQRT(DABS(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
        PITERO = PITER + SQ**2/(2.*OMEGAD)
      ELSE
        PITERO = PITER
      ENDIF
      TBACS = DSQRT(DABS(PITERO)/(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
          YAWERO = YAWER + SR**2/(2.*OMEGAD)
        ELSE
          YAWERO = YAWER
        ENDIF
        TBACS = DSQRT(DABS(YAWERO)/(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)=DABS(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)
            AACCEL(ICMD,IVPFL) = AACCEL(ICMD,IVPFL)+ANGACL(ICMD,IVPFL,I)
20      CONTINUE
        AACCEL(ICMD,IVPFL) = AACCEL(ICMD,IVPFL)/
                                DBLE(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 = DABS(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
        ENDIF
    ENDIF

```

```

ELSEIF ( IACSB1.EQ.1 ) THEN
C      ENABLE FIRST ATTITUDE CONTROL PULSE

      TBURNA(ICMD) = AERROR(ICMD)*TBACS/DABS(AERROR(ICMD))
      DTSAMP = DABS(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.EC.2 ) THEN
        IVPFLN = 3
      ELSE
        IVPFLN = 1
      ENDIF
      TBURN2=(OMEGA(ICMD)-DIRECT*AACCEL(ICMD,IVPFLN)*TBURN3)
        /AACCEL(ICMD,IVPFLN)

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,IVPFLN)*TBURN2
      THT2DD = -DIRECT*AACCEL(ICMD,IVPFLN)
      THT1DD = -DIRECT*AACCEL(ICMD,IVPFLN)
      DELANG = 0.5*(THET2D**2 - OMEGA(ICMD)**2)/THT1DD +
        2.*THET2D*TBURN3 - 0.5*THET2D**2/THT2DD
      DELNXT = AERROR(ICMD) - OMEGA(ICMD)*DTMCU
      IF ( DABS(DELANG).GE.DABS(DELNXT) ) THEN
        IACSB2 = 0
        ICNT = 0
        TBURNA(ICMD) = -TBURN2
        DTSAMP = DABS(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 = (DABS(AERROR(ICMD)) - DELANG)/DABS(OMEGA(ICMD))
      IF ( DELANG.GE.DABS(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 = DABS(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
      RETURN
      END

```

FILE: uuv22.19g/dutility/uumisslt.for

```

C-----
      SUBROUTINE MISSLT(T,QUAT,CIM,MASS,FXA,FXT,
      .                FRCX,FXACS,FXVCS,FYA,FYT,FRCY,FYACS,FYVCS,FZA,
      .                FZT,FRCZ,FZACS,FZVCS,
      .                X,Y,Z,XD,YD,ZD,UD,VD,WD,
      .                GB,GR,MGR,FX,FY,FZ,XDD,YDD,ZDD,MXYZDD,
      .                U,V,W,PHI,THT,PSI)
C-----
C   SUBROUTINE NAME :      MISSLT
C   AUTHOR(S) :          D. C. FOREMAN, A. P. BUKLEY
C   FUNCTION :           COMPUTES THE TRANSLATIONAL
C                       MISSILE ACCELERATIONS
C   CALLED FROM :        FORTRAN MAIN
C   SUBROUTINES CALLED :  FV2BXI
C   INPUTS :             T,QUAT,CIM,MASS,FXA,
C                       FXT,FRCX,FXACS,FXVCS,FYA,FYT,FRCY,FYACS,
C                       FYVCS,FZA,FZT,FRCZ,FZACS,FZVCS,
C                       X,Y,Z,XD,YD,ZD
C   OUTPUTS :            UD,VD,WD,PD,QD,RD,GB,GR,MGR,FX,FY,
C                       FZ,XDD,YDD,ZDD,MXYZDD,U,V,W,QUATD,PHI,THT,
C                       PSI
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)

      REAL          FRCX, FRCY, FRCZ, FXA, FYA, FZA
      REAL          FXACS, FYACS, FZACS, FXT, FYT, FZT
      REAL          FXVCS, FYVCS, FZVCS

      REAL          CIM(9)           , CMI(9)           , TMP1
      DOUBLE PRECISION GB(3)
      DOUBLE PRECISION GR(3)
      DOUBLE PRECISION IXX           , IYY
      DOUBLE PRECISION IZZ           , MASS             , MGR
      DOUBLE PRECISION MXYZ
      DOUBLE PRECISION MXYZDD
      DOUBLE PRECISION PQR(3)
      REAL          QUAT(4)
      DOUBLE PRECISION QUATD(4)      , UXYZ(3)
      DOUBLE PRECISION UXYZDD(3)     , XYZLCH(3)

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

      SAVE          IMISL

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSMISSIL.DAT')
$INCLUDE('~/INCLUDE/SSCON28.DAT')
$INCLUDE('~/INCLUDE/SSCON39.DAT')
$INCLUDE('~/INCLUDE/SSCON63.DAT')

      DATA IMISL / 1 /
      DATA NCLEAR / 0 /

      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)

          IF (T .EQ. 0.0) THEN
              XYZLCH(1) = XLNCH*CMI(1) + RADE
              XYZLCH(2) = XLNCH*CMI(2)
              XYZLCH(3) = XLNCH*CMI(3)
          ENDIF

      ENDIF

C      DETERMINE LOCAL GRAVITY VECTOR

      MXYZ = DSQRT ( X**2 + Y**2 + Z**2 )
      MGR = GMU / MXYZ**2

      IF ( MXYZ.GT.0.0D0 ) THEN
* FTN286 X415 OPTIMIZE(3)
99999 CONTINUE
          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

```
FX = FXT + FXA + FRCX + FXACS + FXVCS
FY = FYT + FYA + FRCY + FYACS + FYVCS
FZ = FZT + FZA + FRCZ + 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)
```

C MISSILE STILL ON GROUND

```
ELSE IF ( FX/MASS.LE.DABS(GB(1)) ) THEN
  GB(1) = 0.0
  GB(2) = 0.0
  GB(3) = 0.0
  GR(1) = 0.0
  GR(2) = 0.0
  GR(3) = 0.0
  UD = 0.0
  VD = 0.0
  WD = 0.0
```

C MISSILE OFF GROUND BUT NOT CLEAR OF THE LAUNCHER

```
ELSE IF ( X.LE.XYZLCH(1) .AND. Y.LE.XYZLCH(2) .AND.
  Z.LE.XYZLCH(3) ) THEN
  GB(2) = 0.0
  GB(3) = 0.0
  GR(1) = CMI(1)*GB(1)
  GR(2) = CMI(2)*GB(1)
  GR(3) = CMI(3)*GB(1)
  UD = FX/MASS + GB(1)
  VD = 0.0
  WD = 0.0
```

C MISSILE JUST NOW CLEARING LAUNCHER

```
ELSE
  NCLEAR = 1
  CALL OUTMES(0901,T,0.0D0)
  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
```

```
* FTN286 X415 OPTIMIZE(3)
```

```
99998 CONTINUE
```

```
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
```

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   = DBLE(ATAN2(CIM(8),CIM(9)))
THT   = -DBLE(ASIN (CIM(7)))
PSI   = DBLE(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
```

```
FILE: uuv22.19g/dutility/uummk.for
```

```
-----
C SUBROUTINE MMK(A,NA,B,NB,C,NC,RM)
C-----
C SUBROUTINE NAME : MMK
C AUTHOR(S) : J. SHEEHAN
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 CALLED FROM : UTILITY SUBROUTINE
C SUBROUTINES CALLED : ROTMX, MMLXY
C INPUTS : A,NA,B,NB,C,NC
C OUTPUTS : RM
C UPDATES : D. SMITH - CR # 59
C-----
C IMPLICIT REAL (A-H)
C IMPLICIT REAL (O-Z)
C DIMENSION AM(3,3), BM(3,3), CM(3,3), RM(3,3), T(9)
C CALL ROTMX(A,NA,AM)
C CALL ROTMX(B,NB,BM)
C CALL ROTMX(C,NC,CM)
C CALL MMLXY(BM,CM,T)
C CALL MMLXY(AM,T,RM)
C RETURN
C END
```

```
FILE: uuv22.19g/dutility/uummlxy.for
```

```
-----
C SUBROUTINE MMLXY(X,Y,Z)
C-----
C SUBROUTINE NAME : MMLXY
```

```

C
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
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

```

FILE: uuv22.19g/duility/uunavig.for

```

C-----
C  SUBROUTINE NAVIG(T,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,TONAV)
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,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,TONAV
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                     C. SMITH   - CR # 076
C
C

```

```

C          B. HILL /      - CR # 081
C          R. RHYNE      - CR # 087
C          R. RHYNE      - CR # 089
C          P. HILL       - CR # 092
C          D. SMITH      - CR # 093
C          B. HILL       - CR # 093
C
-----
      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      REAL      SP      , SQ      , SR
      DOUBLE PRECISION VMIR(3)      , RMIR(3)      , VMI(3)
      DOUBLE PRECISION RMI(3)
      REAL      TI2M(9)
      DOUBLE PRECISION GR(3)
      DOUBLE PRECISION CIE(9)
      REAL      AT(3)
      DOUBLE PRECISION QS1(4)      , GRAVG(3)
      DOUBLE PRECISION GRLAST(3)

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

      SAVE      INAVIG

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

      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)
              SIPSO = DSIN(SPSI/2.0D0)
              COPS0 = DCOS(SPSI/2.0D0)
              SITH0 = DSIN(SPHI/2.0D0)
              COPHO = DCOS(SPHI/2.0D0)

C              CALCULATE QUATERNIONS

              QS1(4) = COPS0*COTH0*COPHO + SIPSO*SITH0*SIPHO
              QS1(1) = COPS0*COTH0*SIPHO - SIPSO*SITH0*COPHO
              QS1(2) = COPS0*SITH0*COPHO + SIPSO*COTH0*SIPHO
              QS1(3) = -COPS0*SITH0*SIPHO + SIPSO*COTH0*COPHO

C              COMPUTE TRANSFORMATION MATRICES

              TI2M(1) = QS1(4)**2 + QS1(1)**2 - QS1(2)**2 - QS1(3)**2
              TI2M(2) = 2.0D0*(QS1(1)*QS1(2) - QS1(4)*QS1(3))
              TI2M(3) = 2.0D0*(QS1(1)*QS1(3) + QS1(4)*QS1(2))
              TI2M(4) = 2.0D0*(QS1(1)*QS1(2) + QS1(4)*QS1(3))
              TI2M(5) = QS1(4)**2 - QS1(1)**2 + QS1(2)**2 - QS1(3)**2
              TI2M(6) = 2.0D0*(QS1(2)*QS1(3) - QS1(4)*QS1(1))
              TI2M(7) = 2.0D0*(QS1(1)*QS1(3) - QS1(4)*QS1(2))
              TI2M(8) = 2.0D0*(QS1(2)*QS1(3) + QS1(4)*QS1(1))
              TI2M(9) = QS1(4)**2 - QS1(1)**2 - QS1(2)**2 + QS1(3)**2

          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

      PPO = DTX**2 + DTY**2 + DTZ**2

```

```

PP1   = ( PP0*DTX + DTY*DTZ0 - DTZ*DTY0 ) / 6.000
PP2   = ( PP0*DTY + DTZ*DTX0 - DTX*DTZ0 ) / 6.000
PP3   = ( PP0*DTZ + DTX*DTY0 - DTY*DTX0 ) / 6.000

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   = -0.500*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    = 0.500*(1.000-QS1(4)**2-QS1(1)**2-QS1(2)**2-QS1(3)**2)
QS1(1) = QS1(1)*(1.000 + DQ)
QS1(2) = QS1(2)*(1.000 + DQ)
QS1(3) = QS1(3)*(1.000 + DQ)
QS1(4) = QS1(4)*(1.000 + DQ)

C     COMPUTE TRANSFORMATION MATRICES

TI2M(1) = QS1(4)**2 + QS1(1)**2 - QS1(2)**2 - QS1(3)**2
TI2M(2) = 2.000*(QS1(1)*QS1(2) - QS1(4)*QS1(3))
TI2M(3) = 2.000*(QS1(1)*QS1(3) + QS1(4)*QS1(2))
TI2M(4) = 2.000*(QS1(1)*QS1(2) + QS1(4)*QS1(3))
TI2M(5) = QS1(4)**2 - QS1(1)**2 + QS1(2)**2 - QS1(3)**2
TI2M(6) = 2.000*(QS1(2)*QS1(3) - QS1(4)*QS1(1))
TI2M(7) = 2.000*(QS1(1)*QS1(3) - QS1(4)*QS1(2))
TI2M(8) = 2.000*(QS1(2)*QS1(3) + QS1(4)*QS1(1))
TI2M(9) = QS1(4)**2 - QS1(1)**2 - QS1(2)**2 + QS1(3)**2

C     COMPUTE SENSED EULER ANGLES

SPHI  = DBLE(ATAN2(TI2M(8),TI2M(9)))
STHT  = -DBLE(ASIN (TI2M(7)))
SPSI  = DBLE(ATAN2(TI2M(4),TI2M(1)))

C     CALCULATE SENSED ANGULAR RATES AND ACCELERATIONS IN BODY FRAME

IF ( DTDEL.GT.0.000 ) 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 GRAVITY INTERVAL
      IF ( DTDEL.NE.0.500 ) THEN
        GRAVG(1) = 0.500*( GRLAST(1) + GR(1) )
        GRAVG(2) = 0.500*( GRLAST(2) + GR(2) )
        GRAVG(3) = 0.500*( 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) + 0.500*DELXD)
      RMIR(2) = RMIR(2) + DTDEL*(VMIR(2) + 0.500*DELYD)
      RMIR(3) = RMIR(3) + DTDEL*(VMIR(3) + 0.500*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 = T12M(1)*VMIR(1) + T12M(4)*VMIR(2) + T12M(7)*VMIR(3)
      SV = T12M(2)*VMIR(1) + T12M(5)*VMIR(2) + T12M(8)*VMIR(3)
      SW = T12M(3)*VMIR(1) + T12M(6)*VMIR(2) + T12M(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

```

FILE: dev22.19quantity/convert.for

```

-----
C   SUBROUTINE NORM(SD,MN,ISEED,RNN)
-----
C   SUBROUTINE NAME :      NORM
C   AUTHOR(S) :          D. F. SMITH
C   FUNCTION :           GENERATES NORMALLY DISTRIBUTED RANDOM
C                        NUMBERS USING THE BOX-MULLER TRANSFORMATION
C   CALLED FROM :        UTILITY SUBROUTINE
C   SUBROUTINES CALLED :  RAND
C   INPUTS :             SD,MN
C   OUTPUTS :            RNN
C   BOTH :              ISEED
C   UPDATES :            D. SMITH   - CR # 382
C                       R. RHYNE   - CR # 387

```

```

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

DOUBLE PRECISION  MN
INTEGER*4         ISEED

SAVE GSET, ISET
DATA GSET/0., ISET/0/

DATA ONE  / 1.000 /
DATA TWO  / 2.000 /

C IF A SPARE RANDOM NUMBER IS NOT AVAILABLE FROM THE PREVIOUS PASS
C GENERATE TWO NEW ONES

IF ( ISET.EQ.0 ) THEN

C   GET TWO UNIFORM RANDOM NUMBERS WITHIN THE SQUARE EXTENDING
C   FROM -1 TO 1 IN EACH DIRECTION
1   V1      = TWO*RAN0(ISEED) - ONE
   V2      = TWO*RAN0(ISEED) - ONE

C   SEE IF THEY ARE WITHIN THE UNIT CIRCLE . IF NOT , TRY AGAIN .

   R      = V1*V1 + V2*V2
   IF ( R.GE.ONE ) GO TO 1

C   PERFORM BOX-MULLER TRANSFORMATION TO GENERATE TWO GAUSSIAN
C   RANDOM NUMBERS . RETURN ONE AND SAVE THE OTHER FOR THE NEXT
C   PASS .

   FAC    = DSQRT ( -TWO*DLOG(R)/R )
   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

```

FILE: uuov22.19q/utility/uuobtgarg.for

```

C -----
C   SUBROUTINE  OBTARG(T,GRTEST,RTEST,VTEST,TL2)
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,TL2
C
C   UPDATES :
C       B. HILL           - 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
      IMPLICIT DOUBLE PRECISION      (A-H)
      IMPLICIT DOUBLE PRECISION      (O-Z)

      DOUBLE PRECISION RTEST(3)      , VTEST(3)
      DOUBLE PRECISION GRTEST(3)     , GRTPST(3) , GRTAOB(3)
      DOUBLE PRECISION TARPOS(3)     , TARVEL(3)

      INTEGER FIRST2

* DATA INITIALIZATION
SINCLUDE('~/INCLUDE/SSOBTARG.DAT')
SINCLUDE('~/INCLUDE/SSCON65.DAT')

      IF ( FIRST2 .EQ. 1 ) THEN
        FIRST2 = 0
        TL2 = T

C      INITIALIZE ESTIMATED TARGET STATES

        DO 45 IAXIS = 1, 3
          RTEST(IAxis) = TARPOS(IAxis)
          VTEST(IAxis) = TARVEL(IAxis)
45      CONTINUE
        ELSE

C      INTEGRATE TARGET ACCELERATION AND VELOCITY USING AVERAGE
C      GRAVITY VECTOR OVER LAST INTERVAL

          TDELTA = T - TL2
          TL2 = T
          DO 2 I = 1,3
            GRTAOB(I) = 0.5D0 * ( GRTEST(I) + GRTPST(I) )
            RTEST(I) = RTEST(I) + VTEST(I)*TDELTA +
              0.5D0*GRTAOB(I)*TDELTA*TDELTA
            VTEST(I) = VTEST(I) + GRTAOB(I)*TDELTA
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

```

FILE: uuv22.19g/dutility/uuoutmes.for

```

      SUBROUTINE OUTMES(N,T,ARG)
      INTEGER N
      DOUBLE PRECISION T,ARG
SINCLUDE('~/INCLUDE/TARGET.FOR')
      CHARACTER*80 MESSAGE

C
C      PROGRAM: MAIN (0101...0200)
C
      IF ( N.EQ.0101 ) THEN
        WRITE(MESSAGE,0101) T
0101      FORMAT(1X,E16.9,' 1ST STAGE SEPARATION')
        GO TO 99999
      END IF

      IF ( N.EQ.0102 ) THEN
        WRITE(MESSAGE,0102) T
0102      FORMAT(1X,E16.9,' 2ND STAGE SEPARATION')
        GO TO 99999
      END IF

      IF ( N.EQ.0103 ) THEN
        WRITE(MESSAGE,0103) T
0103      FORMAT(1X,E16.9,' DROP NOSE FAIRING AND BOOST ADAPTER')

```

```

      GO TO 99999
      END IF

      IF ( N.EQ.0104 ) THEN
        WRITE(MESSAGE,0104) T,ARG
0104      FORMAT(1X,E16.9,1X,E16.9)
        GO TO 99999
      END IF

      IF ( N.EQ.0105 ) THEN
        WRITE(MESSAGE,0105) T,ARG
0105      FORMAT(1X,E16.9,' MISS = ',E16.9)
        GO TO 99999
      END IF

C
C      SUBROUTINE: CMPINV (0201...0300)
C
      IF ( N.EQ.0201 ) THEN
        WRITE(MESSAGE,0201)
0201      FORMAT(' MATRIX SIZE TOO LARGE IN CMPINV')
        GO TO 99999
      END IF

C
C      SUBROUTINE: DISCRT (0301...0400)
C
      IF ( N.EQ.0301 ) THEN
        WRITE(MESSAGE,0301)
0301      FORMAT(' SYSTEM ORDER TOO LARGE IN DISCRT')
        GO TO 99999
      END IF

      IF ( N.EQ.0302 ) THEN
        WRITE(MESSAGE,0302)
0302      FORMAT(' SUITABLE CONVERGENCE WAS NOT REACHED IN DISCRT')
        GO TO 99999
      END IF

C
C      SUBROUTINE: EIGVEC (0401...0500)
C
      IF ( N.EQ.0401 ) THEN
        WRITE(MESSAGE,0401)
0401      FORMAT(' MATRIX SIZE TOO LARGE IN EIGVEC')
        GO TO 99999
      END IF

C
C      SUBROUTINE: HQR (0501...0600)
C
      IF ( N.EQ.0501 ) THEN
        WRITE(MESSAGE,0501)
0501      FORMAT(' TOO MANY ITERATIONS IN HQR')
        GO TO 99999
      END IF

C
C      SUBROUTINE: KALMAN (0601...0700)
C
      IF ( N.EQ.0601 ) THEN
        WRITE(MESSAGE,0601) T
0601      FORMAT(1X,E16.9,' INITIATE ACQUISITION PHASE')
        GO TO 99999
      END IF

      IF ( N.EQ.0602 ) THEN
        WRITE(MESSAGE,0602) T
0602      FORMAT(1X,E16.9,' INITIATE TRACK PHASE')
        GO TO 99999
      END IF

      IF ( N.EQ.0603 ) THEN
        WRITE(MESSAGE,0603) T
0603      FORMAT(1X,E16.9,' INITIATE TERMINAL PHASE')
        GO TO 99999

```

```

END IF

IF ( N.EQ.0604 ) THEN
  WRITE(MESSAGE,0604) T,ARG
0604  FORMAT(1X,E16.9,' ACQUISITION MODE ENABLED:  MAGRO = ',E16.9)
  GO TO 99999
END IF

IF ( N.EQ.0605 ) THEN
  WRITE(MESSAGE,0605) T,ARG
0605  FORMAT(1X,E16.9,' TRACK MODE ENABLED:  MAGRO = ',E16.9)
  GO TO 99999
END IF

IF ( N.EQ.0606 ) THEN
  WRITE(MESSAGE,0606) T,ARG
0606  FORMAT(1X,E16.9,' CSO MODE ENABLED:  MAGRO = ',E16.9)
  GO TO 99999
END IF

IF ( N.EQ.0607 ) THEN
  WRITE(MESSAGE,0607) T,ARG
0607  FORMAT(1X,E16.9,' TERMINAL MODE ENABLED:  MAGRO = ',E16.9)
  GO TO 99999
END IF

C
C  SUBROUTINE: MATINV (0701...0800)
C
  IF ( N.EQ.0701 ) THEN
    WRITE(MESSAGE,0701)
0701  FORMAT(' MATRIX SIZE TOO LARGE IN MATINV')
    GO TO 99999
  END IF

C
C  SUBROUTINE: MCGUID (0801...0900)
C
  IF ( N.EQ.0801 ) THEN
    WRITE(MESSAGE,0801) T
0801  FORMAT(1X,E16.9,' KV PITCHOVER COMPLETE',
    &      ' - BEGIN DISTURBANCE MEASUREMENT')
    GO TO 99999
  END IF

  IF ( N.EQ.0802 ) THEN
    WRITE(MESSAGE,0802) T
0802  FORMAT(1X,E16.9,' DISTURBANCE MEASUREMENT COMPLETE',
    &      ' - ORIENT KV TO LOS')
    GO TO 99999
  END IF

  IF ( N.EQ.0803 ) THEN
    WRITE(MESSAGE,0803) T
0803  FORMAT(1X,E16.9,' KV ORIENTATION COMPLETE')
    GO TO 99999
  END IF

C
C  SUBROUTINE: MISSIL (0901...1000)
C
  IF ( N.EQ.0901 ) THEN
    WRITE(MESSAGE,0901) T
0901  FORMAT(1X,E16.9,' MISSILE HAS CLEARED THE LAUNCHER')
    GO TO 99999
  END IF

C
C  SUBROUTINE: OPTSSC (1001...1100)
C
  IF ( N.EQ.1001 ) THEN
    WRITE(MESSAGE,1001)
1001  FORMAT(' MAXIMUM NUMBER OF STATES EXCEEDED IN OPTSSC')
    GO TO 99999
  END IF

```

```

C
C SUBROUTINE: RANO (1101...1200)
C
  IF ( N.EQ.1101 ) THEN
    WRITE(MESSAGE,1101)
1101  FORMAT(' RANDOM NUMBER OUT OF BOUNDS IN RANO')
    GO TO 99999
  END IF

C
C SUBROUTINE: SEEKER (1201...1300)
C
  IF ( N.EQ.1201 ) THEN
    WRITE(MESSAGE,1201) T
1201  FORMAT(1X,E16.9,' TRUE LOS ANGLE EXCEEDS FIELD-OF-VIEW LIMIT')
    GO TO 99999
  END IF

  IF ( N.EQ.1202 ) THEN
    WRITE(MESSAGE,1202) T
1202  FORMAT(1X,E16.9,' TARGET REACQUIRED')
  END IF

  IF ( N.EQ.1203 ) THEN
    WRITE(MESSAGE,1203) T,ARG
1203  FORMAT(1X,E16.9,' FRAME RATE CHANGE: FRMRAT = ',E16.9)
    GO TO 99999
  END IF

C
C SUBROUTINE: SSPLAG (1301...1400)
C
  IF ( N.EQ.1301 ) THEN
    WRITE(MESSAGE,1301)
1301  FORMAT(' BUFFER SIZE INSUFFICIENT IN SSPLAG')
    GO TO 99999
  END IF

C
C SUBROUTINE: TARGET (1401...1500)
C
  IF ( N.EQ.1401 ) THEN
    WRITE(MESSAGE,1401) T,ARG
1401  FORMAT(1X,E16.9,' TARGET RESOLVED: RANGE = ',E16.9)
    GO TO 99999
  END IF

C
C SUBROUTINE: VCSLOG (1501...1600)
C
  IF ( N.EQ.1501 ) THEN
    WRITE(MESSAGE,1501) T,ARG
1501  FORMAT(1X,E16.9,' ISSUE MIDCOURSE DISTURBANCE BURN',
    & ' - VCS THRUSTER ',F2.0)
    GO TO 99999
  END IF

  IF ( N.EQ.1502 ) THEN
    WRITE(MESSAGE,1502) T,ARG
1502  FORMAT(1X,E16.9,' ISSUE MIDCOURSE BURN ',F2.0)
    GO TO 99999
  END IF

  IF ( N.EQ.1503 ) THEN
    WRITE(MESSAGE,1503) T,ARG
1503  FORMAT(1X,E16.9,' ISSUE MIDCOURSE BURN ',F2.0,
    & ' - BURN TIME BELOW THRESHOLD')
    GO TO 99999
  END IF

  IF ( N.EQ.1504 ) THEN
    WRITE(MESSAGE,1504) T
1504  FORMAT(1X,E16.9,' ISSUE FIRST BURN')
    GO TO 99999
  END IF

  IF ( N.EQ.1505 ) THEN

```

```

      WRITE(MESSAGE,1505) T
1505   FORMAT(1X,E16.9,' ISSUE FIRST BURN',
      &   ' - BURN TIME BELOW THRESHOLD')
      GO TO 99999
      END IF

      IF ( N.EQ.1506 ) THEN
      WRITE(MESSAGE,1506) T
1506   FORMAT(1X,E16.9,' ISSUE SECOND BURN')
      GO TO 99999
      END IF

      IF ( N.EQ.1507 ) THEN
      WRITE(MESSAGE,1507) T
1507   FORMAT(1X,E16.9,' ISSUE SECOND BURN',
      &   ' - BURN TIME BELOW THRESHOLD')
      GO TO 99999
      END IF

      IF ( N.EQ.1508 ) THEN
      WRITE(MESSAGE,1508) T
1508   FORMAT(1X,E16.9,' ISSUE THIRD BURN')
      GO TO 99999
      END IF

      IF ( N.EQ.1509 ) THEN
      WRITE(MESSAGE,1509) T
1509   FORMAT(1X,E16.9,' ISSUE THIRD BURN',
      &   ' - BURN TIME BELOW THRESHOLD')
      GO TO 99999
      END IF

      WRITE(MESSAGE,0001) N
0001   FORMAT(' ERROR: MESSAGE NUMBER = ',I4)

99999 CONTINUE
      CALL OUTPUT_MESSAGE( %VAL(CHARACTER_08BIT), MESSAGE )
      CALL OUTPUT_NL

      RETURN
      END

```

FILE: uuv22.19g/utility/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

```

FILE: uuv22.19g/dutility/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      BOTH :               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
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(1100,0.0D0,0.0D0)
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

```

FILE: uuv22.19g/dutility/uuranit.for

```

C-----
C      SUBROUTINE RANIT ( ISEED )
C-----
C
C      SUBROUTINE NAME :      RANIT
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           INITIALIZES A TABLE OF RANDOM NUMBERS FOR
C                           USE BY THE UNIFORM RANDOM GENERATOR RAN0
C
C      CALLED FROM :        EXECUTIVE ROUTINE
C
C      SUBROUTINES CALLED :  RAN
C
C      INPUTS :             NONE
C
C      OUTPUTS :            NONE

```

```

C
C      BOTH :                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  RANIT
C
C      COMMON / RANCOM /          RANSEQ(97),      RANLST
C
C      EXERCISE SYSTEM ROUTINE
C
C      DO 10 I = 1 , 97
C          DUMMY = RAN ( ISEED )
10 CONTINUE
C
C      STORE 97 RANDOM NUMBERS IN BUFFER ( 97 IS NOT SPECIAL )
C
C      DO 20 I = 1 , 97
C          RANSEQ(I) = RAN ( ISEED )
20 CONTINUE
C
C      SAVE ANOTHER RANDOM NUMBER TO USE FOR INDEXING BUFFER
C
C      RANLST = RAN ( ISEED )
C
C      RETURN
C      END

```

FILE: uuv22.19g/dutility/uurelat.for

```

C-----
C      SUBROUTINE RELAT(RTIC,VTIC,X,Y,Z,XD,YD,ZD,Q,R,CIM,CMS,RRELTR,
C      .               MAGRTR,VRELTR,MGRDTR,MAGLOS,LAMTRU,LAMDXX,
C      .               LAMDTR,LAMSEK,LAMDSK,TGOTR,RRELM,VRELM,CAZ,CEL)
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,RRELM,
C                          VRELM,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)
      DOUBLE PRECISION CMS(9)      , MAGLOS
      DOUBLE PRECISION RTIC(5,3)
      REAL          RRELTR(3)      , URRELTR(3)
      REAL          MAGRTR
      DOUBLE PRECISION VTIC(5,3)
      REAL          VRELTR(3)      , MAGVTR      , VRDRRT
      DOUBLE PRECISION MGRDTR      , RRELM(3)

```

```

DOUBLE PRECISION VRELM(3)      , LAMTRU(2)      , LAMDXX(2)
DOUBLE PRECISION LAMDTR(2)     , RRELS(3)       , VRELS(3)
REAL               LAMSEK(2)    , Q             , R
REAL              TGOTR
DOUBLE PRECISION  LAMDSK(2)     , CAZ(100)
DOUBLE PRECISION  CEL(100)

INTEGER           SEKTYP

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSCON50.DAT')
$INCLUDE('~/INCLUDE/SSCON66.DAT')
$INCLUDE('~/INCLUDE/SSCON21.DAT')

IF ( SEKTYP .EQ. 3 ) THEN
  DO 65 IOBJ = 1, NOBJ

C      CALCULATE TEMPORARY RELATIVE RANGE FOR EACH OBJECT
C      IN INERTIAL FRAME

      TMP1 = RTIC(IOBJ,1) - X
      TMP2 = RTIC(IOBJ,2) - Y
      TMP3 = RTIC(IOBJ,3) - Z

C      TRANSFORM TO MISSILE FRAME

      TMP4 = TMP1*CIM(1) + TMP2*CIM(4) + TMP3*CIM(7)
      TMP5 = TMP1*CIM(2) + TMP2*CIM(5) + TMP3*CIM(8)
      TMP6 = TMP1*CIM(3) + TMP2*CIM(6) + TMP3*CIM(9)

C      TRANSFORM TO SEEKER FRAME

      TMP7 = TMP4*CMS(1) + TMP5*CMS(4) + TMP6*CMS(7)
      TMP8 = TMP4*CMS(2) + TMP5*CMS(5) + TMP6*CMS(8)
      TMP9 = TMP4*CMS(3) + TMP5*CMS(6) + TMP6*CMS(9)

C      DETERMINE ELEVATION AND AZIMUTH FOR EACH OBJECT

      CEL(IOBJ) = DATAN2(-TMP9,TMP7)
      CAZ(IOBJ) = DATAN2( TMP8,TMP7)
65  CONTINUE
  ENDIF

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 = SQRT(RRELTR(1)**2 + RRELTR(2)**2 + RRELTR(3)**2)
      URRELT(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 = SQRT(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)
      RRELM(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) = SNGL ( DATAN2(-RRELS(3),RRELS(1)) )
LAMSEK(2) = SNGL ( 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

```

FILE: uuv22.19g/utility/uuresp2r.for

```

C-----
C      SUBROUTINE RESP2R ( DT,WD,ZD,CILL,CIL,CI,COLL,COL,CO )
C-----
C      SUBROUTINE NAME :      RESP2R
C      AUTHOR(S) :          D. F. SMITH
C      FUNCTION :           Given a second order continuous filter of
C                           the form
C
C                           
$$G(s) = \frac{WD**2}{s**2 + 2.0*ZD*WD*s + WD**2}$$

C                           compute a digital filter which yields the
C                           same ramp response . The digital filter has
C                           the transfer function
C
C                           
$$G(z) = \frac{CI*z**2 + CIL*z + CILL}{CO*z**2 + COL*z + COLL}$$

C
C      CALLED FROM :        UTILITY ROUTINE
C      SUBROUTINES CALLED :  NONE
C      INPUTS :             DT,WD,ZD
C      OUTPUTS :            CILL,CIL,CI,COLL,COL,CO
C      UPDATES :            NONE
C-----

```

```

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

```

```

DATA      ONE   / 1.000 /
DATA      TWO   / 2.000 /

```

```

C      Underdamped filter

IF ( ZD.LT.ONE ) THEN
  A = WD*ZD
  B = WD*DSQRT ( ONE - ZD**2 )
  TMP1 = DEXP ( - A*DT )
  TMP2 = DEXP ( - TWO*A*DT )
  TMP3 = DCOS ( B*DT )
  TMP4 = DSIN ( B*DT )

```

```

TMP5 = A*A + B*B
TMP6 = TMP1*TMP4*( A*A - B*B )/B
CI = TMP5*DT - TWO*A + TWO*A*TMP1*TMP3 + TMP6
CIL = TWO*( A - DT*TMP1*TMP3*TMP5 - TMP6 - A*TMP2 )
CILL = TMP6 - TWO*A*TMP1*TMP3 + TMP2*( TWO*A + TMP5*DT )
CO = TMP5*DT
COL = - TWO*TMP1*TMP3*CO
COLL = TMP2*CO
END IF

```

C Critically damped filter

```

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 )
  CILL = - BSQ*( ONE - EXPB*TMP1 - EXPA*TMP3 )
  CO = ASQ*EXPA*( EXPB*TMP4 - ONE )
  COL = - BSQ*EXPB*( EXPA*TMP3 - ONE )
  COLL = CO*EXPA*EXPB
END IF

RETURN
END

```

FILE: uuv22.19g/dutility/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

```

```

C          T. THORNTON - CR # 044
C          B. HILL    - CR # 051
C          D. SMITH   - CR # 059
C          B. HILL /   - CR # 081
C          R. RHYNE   - CR # 084
C          B. HILL    - CR # 086
C          B. HILL    - CR # 093
C

```

```

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

```

```

      REAL          DTACSA(4)      , DTACSB(4)

```

```

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSCON67.DAT')
$INCLUDE('^/INCLUDE/SSCON03.DAT')
$INCLUDE('^/INCLUDE/SSCON08.DAT')

```

```

C      IN DISTURBANCE MODE TURN OFF ACS THRUSTERS WITH DIVERT THRUSTERS

```

```

      IF( IDIST.EQ. 1 ) THEN
        TMP1 = TOFLTM - T
        IF( TMP1.LE. 0. ) THEN
          TMP2 = 0.
        ELSEIF( TMP1.LT. TSMPPH ) THEN
          TMP2 = TMP1/TSMPPH
        ELSE
          TMP2 = 1.
        ENDIF
        TPATON = TPATON*TMP2
        TYATON = TYATON*TMP2
        TRATON = TRATON*TMP2
      ENDIF

```

```

C      TEST SIGNS OF PITCH, YAW, ROLL AND ATTITUDE THRUSTER PULSEWIDTHS

```

```

C      PITCH SIGN TEST

```

```

      IF( TPATON.GE. 0.0 ) THEN
* FTN286 X415 OPTIMIZE(3)
99999 CONTINUE
        TPATP = TPATON
        TPATN = 0.0
      ELSE
        TPATP = 0.0
        TPATN = -TPATON
      ENDIF

```

```

C      YAW SIGN TEST

```

```

      IF( TYATON.GE. 0.0 , THEN
* FTN286 X415 OPTIMIZE(3)
99998 CONTINUE
        TYATP = TYATON
        TYATN = 0.0
      ELSE
        TYATP = 0.0
        TYATN = -TYATON
      ENDIF

```

```

C      ROLL SIGN TEST

```

```

      IF( TRATON.GE. 0.0 ) THEN
* FTN286 X415 OPTIMIZE(3)
99997 CONTINUE
        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
*   FTN286 X415 OPTIMIZE(3)
99996 CONTINUE
        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

```

FILE: uu22.19g/utility/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

```

```

      IMPLICIT REAL (/ H)
      IMPLICIT REAL (O-Z)
      REAL XM(3,3)

```

```

      SX = SIN(X)
      CX = COS(X)

```

```

      IF ( I.EQ.1 ) THEN
        XM(1,1) = 1.0
        XM(1,2) = 0.0
        XM(1,3) = 0.0

```

```

      XM(2,1) = 0.0
      XM(2,2) = CX
      XM(2,3) = SX

      XM(3,1) = 0.0
      XM(3,2) = -SX
      XM(3,3) = CX
    END IF

    IF ( I.EQ.2 ) THEN
      XM(1,1) = CX
      XM(1,2) = 0.0
      XM(1,3) = -SX

      XM(2,1) = 0.0
      XM(2,2) = 1.0
      XM(2,3) = 0.0

      XM(3,1) = SX
      XM(3,2) = 0.0
      XM(3,3) = CX
    END IF

    IF ( I.EQ.3 , THEN
      XM(1,1) = CX
      XM(1,2) = SX
      XM(1,3) = 0.0

      XM(2,1) = -SX
      XM(2,2) = CX
      XM(2,3) = 0.0

      XM(3,1) = 0.0
      XM(3,2) = 0.0
      XM(3,3) = 1.0
    END IF

    RETURN
  END

```

FILE: uuv22.19g/utility/utable.for

```

C-----
C      SUBROUTINE TABLE(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 :               :
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)

```

FILE: uvv22.19g/utility/uutarget.for

```

SUBROUTINE TARGET( T,MAGRTR,CAZ,CEL,CER,CIE,PTARG,QTARG,RTARG,
TPHI,THT,TPSI,GRT,TPHID,THTD,TPSID,CIT,
RTIC,VTIC,RTAR,RTER,NSUB,IRESLV,RJ,CTI,
VTAR,LATT,LONGT,AZSUB,ELSUB,RJSUB )
-----
SUBROUTINE NAME :      TARGET
AUTHOR(S) :          D. SISSOM
FUNCTION :            COMPUTES THE ROTATIONAL AND TRANSLATIONAL
                      STATES FOR EACH OBJECT
CALLED FROM :        FORTRAN MAIN
SUBROUTINES CALLED :  MMK
INPUTS :              T,MAGRTR,CAZ,CEL,CER,CIE,PTARG,QTARG,RTARG
BOTH :                TPHI,THT,TPSI,GRT,TPHID,THTD,TPSID,CIT,
                      RTIC,VTIC,RTAR,RTER,NSUB,IRESLV
OUTPUTS :              RJ,CTI,VTAR,LATT,LONGT,AZSUB,ELSUB,RJSUB
UPDATES :              B. HILL      - CR # 036,
                      T. THORNTON - CR # 045

```

```

C          T. THORNTON - CR # 047
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   - CR # 087
C          D. SISSOM  - CR # 091
C          B. HILL    - CR # 093
C          D. SISSOM  - CR # 094
C

```

```

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

CHARACTER*128 MESSAGE
DOUBLE PRECISION  AZSUB(100)   , CAZ(100)       , CEL(100)
REAL              CER(9)
DOUBLE PRECISION  CIE(9)
REAL              CIT(9)
DOUBLE PRECISION  CLTPOS(3)     , CLTVEL(3)      , CSOPOS(3)
DOUBLE PRECISION  CSOVEL(3)     , CTI(9)        , ELSUB(100)
DOUBLE PRECISION  GRT(5,3)
DOUBLE PRECISION  GRTAVG(5,3)   , GRTLST(5,3)
DOUBLE PRECISION  LATT          , LONGT
REAL              MAGRTR
DOUBLE PRECISION  MGRT          , MRTIC         , RHOPOS(3)
DOUBLE PRECISION  RHOVEL(3)     , RJ(5)         , RJSUB(100)
DOUBLE PRECISION  RTAR(3)       , RTER(3)       , RTIC(5,3)
DOUBLE PRECISION  TARPOS(3)     , TARVEL(3)     , TNKPOS(3)
DOUBLE PRECISION  TNKVEL(3)     , URTIC(3)
DOUBLE PRECISION  VTAR(3)       , VTIC(5,3)

```

```

INTEGER          FIRST1
INTEGER          SEKTYP

```

```

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSTARGET.DAT')
$INCLUDE('^/INCLUDE/SSCON21.DAT')
$INCLUDE('^/INCLUDE/SSCON39.DAT')
$INCLUDE('^/INCLUDE/SSCON50.DAT')
$INCLUDE('^/INCLUDE/SSCON65.DAT')
$INCLUDE('^/INCLUDE/SSCON66.DAT')
$INCLUDE('^/INCLUDE/SSCON69.DAT')

```

```

IF ( FIRST1 .EQ. 1 ) THEN
  FIRST1 = 0
  TLI = T

```

```

C      INITIALIZE STATES FOR EACH OBJECT

```

```

      DO 45 IAXIS = 1, 3
        RTIC(1, IAXIS) = TARPOS(IAXIS)
        RTIC(2, IAXIS) = TARPOS(IAXIS) + CSOPOS(IAXIS)
        RTIC(3, IAXIS) = TARPOS(IAXIS) + TNKPOS(IAXIS)
        RTIC(4, IAXIS) = TARPOS(IAXIS) + RHOPOS(IAXIS)
        RTIC(5, IAXIS) = TARPOS(IAXIS) + CLTPOS(IAXIS)
        VTIC(1, IAXIS) = TARVEL(IAXIS)
        VTIC(2, IAXIS) = TARVEL(IAXIS) + CSOVEL(IAXIS)
        VTIC(3, IAXIS) = TARVEL(IAXIS) + TNKVEL(IAXIS)
        VTIC(4, IAXIS) = TARVEL(IAXIS) + RHOVEL(IAXIS)
        VTIC(5, IAXIS) = TARVEL(IAXIS) + CLTVEL(IAXIS)
45    CONTINUE
      RJ(1) = TARRI
      RJ(2) = CSORI
      RJ(3) = TNKRI
      RJ(4) = RHORI
      RJ(5) = CLTRI
    ENDIF
    IF ( SEKTYP .EQ. 3 ) THEN

```

```

C      DETERMINE IF TARGET IS LARGER THAN A USER-INPUT
C      MULTIPLE OF A PIXEL FIELD OF VIEW

```

```

      IF ( MAGRTR .GT. 0.0 .AND. NTARRS .EQ. 1 ) THEN
* FTN286 X415 OPTIMIZE(3)
99999 CONTINUE
      IF ( DMAX1(TARWID, TARLEN)/MAGRTR .GE.

```

```

      RMULT*(WIDTH/FOCLEN) ) THEN
      IF ( IRESLV .EQ. 0 ) THEN
        IRESLV = 1
        DELTL = (1.0/12.0)*TARLEN
        DELTW = (1.0/11.0)*TARWID
        RJNEW = (1.0/73.0)*TARRI
        CALL OUTMES(1401,T,DBLE(MAGRTR))
      ENDIF

C      GENERATE RESOLVABLE TARGET (ON FIRST PASS ONLY) -
C      MODEL THIS TARGET AS SUPERPOSITION OF MANY OBJECTS
C      WITH CENTROID AT ILEN=0, IWID=0

      NSUB = 0
      DO 5 ILEN = -8, 4
        IF ( ILEN .EQ. -8 .OR. ILEN .EQ. -7
          .OR. ILEN .EQ. -6 ) ILOW = 0
        IF ( ILEN .EQ. -5 .OR. ILEN .EQ. -4 ) ILOW = -1
        IF ( ILEN .EQ. -3 .OR. ILEN .EQ. -2 ) ILOW = -2
        IF ( ILEN .EQ. -1 .OR. ILEN .EQ. 0 ) ILOW = -3
        IF ( ILEN .EQ. 1 .OR. ILEN .EQ. 2 ) ILOW = -4
        IF ( ILEN .EQ. 3 .OR. ILEN .EQ. 4 ) ILOW = -5
        IHIGH = IABS(ILOW)
        DO 6 IWID = ILOW, IHIGH
          NSUB = NSUB + 1
          AZSUB(NSUB) = CAZ(1) + (FLOAT(ILEN)*DELT)/MAGRTR
          ELSUB(NSUB) = CEL(1) + (FLOAT(IWID)*DELT)/MAGRTR
          RJSUB(NSUB) = RJNEW
6          CONTINUE
5          CONTINUE
        ENDIF
      ENDIF
      ELSE
        NOBJ = 1
      ENDIF

C      TARGET GRAVITY MODEL

      DO 10 IOBJ = 1, NOBJ
        MRTIC = DSQRT(RTIC(IOBJ,1)**2 + RTIC(IOBJ,2)**2 +
          RTIC(IOBJ,3)**2)
        URTIC(1) = RTIC(IOBJ,1)/MRTIC
        URTIC(2) = RTIC(IOBJ,2)/MRTIC
        URTIC(3) = RTIC(IOBJ,3)/MRTIC

        MGRT = GMU/MRTIC**2
        GRT(IOBJ,1) = -MGRT*URTIC(1)
        GRT(IOBJ,2) = -MGRT*URTIC(2)
        GRT(IOBJ,3) = -MGRT*URTIC(3)

C      INTEGRATE TARGET ACCELERATION AND VELOCITY USING AVERAGE
C      GRAVITY OVER INTERVAL

      TDELT = T - TL1
      DO 2 I = 1,3
        GRTAVG(IOBJ,I) = 0.5D0*(GRT(IOBJ,I) + GRTLST(IOBJ,I))
        RTIC(IOBJ,I) = RTIC(IOBJ,I) + VTIC(IOBJ,I)*TDELT +
          0.5D0*GRTAVG(IOBJ,I)*TDELT*TDELT
        VTIC(IOBJ,I) = VTIC(IOBJ,I) + GRTAVG(IOBJ,I)*TDELT
2      CONTINUE

C      SAVE GRAVITY VECTOR FOR USE ON NEXT PASS

      DO 3 I = 1,3
        GRTLST(IOBJ,I) = GRT(IOBJ,I)
3      CONTINUE
10 CONTINUE
      TL1 = T

C      TRANSFORM INERTIAL POSITION AND VELOCITY TO EARTH FRAME

      RTAR(1) = RTIC(1,1)*CIE(1) + RTIC(1,2)*CIE(4) + RTIC(1,3)*CIE(7)
      RTAR(1) = RTIC(1,1)*CIE(2) + RTIC(1,2)*CIE(5) + RTIC(1,3)*CIE(8)
      RTAR(1) = RTIC(1,1)*CIE(3) + RTIC(1,2)*CIE(6) + RTIC(1,3)*CIE(9)

      VTAR(1) = VTIC(1,1)*CIE(1) + VTIC(1,2)*CIE(4) + VTIC(1,3)*CIE(7)
      VTAR(2) = VTIC(1,1)*CIE(2) + VTIC(1,2)*CIE(5) + VTIC(1,3)*CIE(8)
      VTAR(3) = VTIC(1,1)*CIE(3) + VTIC(1,2)*CIE(6) + VTIC(1,3)*CIE(9)

C      TRANSFORM BODY RATES TO EULER RATES

```

```

TPHID = PTARG + QTARG*DSIN(TPHI)*DTAN(TTHT) +
      RTARG*DCOS(TPHI)*DTAN(TTHT)
TTHTD = QTARG*DCOS(TPHI) - RTARG*DSIN(TPHI)
TPSID = QTARG*DSIN(TPHI)/DCOS(TTHT) + RTARG*DCOS(TPHI)/DCOS(TTHT)

C   INTEGRATE EULER RATES TO OBTAIN TARGET ATTITUDE

TPHI = TPHI + TPHID*TDELTA
TTHT = TTHT + TTHTD*TDELTA
TPSI = TPSI + TPSID*TDELTA

C   COMPUTE TARGET BODY-TO-INERTIAL TRANSFORMATION MATRIX

SNGLTPHI = SNGL(TPHI)
SNGLTTHT = SNGL(TTHT)
SNGLTPSI = SNGL(TPSI)
CALL MMK(SNGLTPHI,1,SNGLTTHT,2,SNGLTPSI,3,CIT)

CTI(1) = CIT(1)
CTI(2) = CIT(4)
CTI(3) = CIT(7)
CTI(4) = CIT(2)
CTI(5) = CIT(5)
CTI(6) = CIT(8)
CTI(7) = CIT(3)
CTI(8) = CIT(6)
CTI(9) = CIT(9)

C   TRANSFORM TARGET EARTH FRAME POSITION TO ROTATING EARTH

RTER(1) = RTAR(1)*CER(1) + RTAR(2)*CER(4) + RTAR(3)*CER(7)
RTER(2) = RTAR(1)*CER(2) + RTAR(2)*CER(5) + RTAR(3)*CER(8)
RTER(3) = RTAR(1)*CER(3) + RTAR(2)*CER(6) + RTAR(3)*CER(9)

C   CALCULATE LATITUDE AND LONGITUDE OF TARGET

LATT = DATAN2(RTER(3),DSQRT(RTER(1)**2 + RTER(2)**2))/DTR
LONGT = DATAN2(RTER(2),RTER(1))/DTR

RETURN
END

```

FILE: uuv22.19g/dutility/uutimer.for

```

      SUBROUTINE INITIALIZE_TIMER()
$INCLUDE('PFP:INCLUDE/TARGET.FOR')
$INCLUDE('^/INCLUDE/UUTIMER.COM')
$INCLUDE('^/INCLUDE/SSCON22.DAT')
$INCLUDE('^/INCLUDE/SSCON23.DAT')
      INTEGER BN, TN

      DO 20 BN=1,4
        DO 10 TN=1,500
          NUMBER_TIMER(BN,TN) = 0
          NUMBER_TICKS(BN,TN) = 0.000
10        CONTINUE
20      CONTINUE

      STAGE1 = INT4( TSTG1 * 1000.0 )
      STAGE2 = INT4( TSTG2 * 1000.0 )
      CALL RESET_TIMER()
      END

      SUBROUTINE START_TIMER( TN )
$INCLUDE('PFP:INCLUDE/TARGET.FOR')
$INCLUDE('^/INCLUDE/UUTIMER.COM')
      INTEGER TN

      TIMER(TN) = READ_TIMER()
      END

      SUBROUTINE STOP_TIMER( TN )
$INCLUDE('PFP:INCLUDE/TARGET.FOR')
$INCLUDE('^/INCLUDE/UUTIMER.COM')
      INTEGER TN

      TIMER(TN) = TIMER(TN) - READ_TIMER()

      NUMBER_TIMER(4,TN) = NUMBER_TIMER(4,TN) + 1

```

```

NUMBER_TICKS(4,TN) = NUMBER_TICKS(4,TN) + DBLE(TIMER(TN))

IF ( NUMBER_TIMER(4,TN) .LT. STAGE1 ) THEN
  NUMBER_TIMER(1,TN) = NUMBER_TIMER(1,TN) + 1
  NUMBER_TICKS(1,TN) = NUMBER_TICKS(1,TN) + DBLE(TIMER(TN))
ELSEIF ( NUMBER_TIMER(4,TN) .LT. STAGE2 ) THEN
  NUMBER_TIMER(2,TN) = NUMBER_TIMER(2,TN) + 1
  NUMBER_TICKS(2,TN) = NUMBER_TICKS(2,TN) + DBLE(TIMER(TN))
ELSE
  NUMBER_TIMER(3,TN) = NUMBER_TIMER(3,TN) + 1
  NUMBER_TICKS(3,TN) = NUMBER_TICKS(3,TN) + DBLE(TIMER(TN))
ENDIF
END

SUBROUTINE OUTPUT_TIMER()
$INCLUDE('PFP:INCLUDE/TARGET.FOR')
$INCLUDE('^/INCLUDE/UUTIMER.COM')
  INTEGER BN, TN
  INTEGER*4 AVERAGE

  DO 20 TN=1,500
    IF ( NUMBER_TIMER(4,TN) .NE. 0 ) THEN
      CALL OUTPUT_MESSAGE(%VAL(SIGNED_16BIT),TN,%VAL(INT2(1)))
      CALL OUTPUT_MESSAGE(%VAL(CHARACTER_08BIT), 'TIMER ' )

      DO 10 BN=1,4
        IF ( NUMBER_TIMER(BN,TN) .NE. 0 ) THEN
          AVERAGE = INT4(NUMBER_TICKS(BN,TN) /
& DBLE(NUMBER_TIMER(BN,TN)))
        ELSE
          AVERAGE = 0
        ENDIF
        CALL OUTPUT_MESSAGE(%VAL(SIGNED_32BIT),AVERAGE,
& %VAL(INT2(1)))
10      CONTINUE

        CALL OUTPUT_NL
      END IF
20    CONTINUE
  END

```

FILE: uuv22.19g/dutility/uuvvsth2.for

```

C-----
C  SUBROUTINE VCSTH2(T,FLTC,FLTCP,FLTCY,TOFFLT,TIMONV)
C-----
C
C  SUBROUTINE NAME :      VCSTH2
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,FLTC,TOFFLT,TIMONV
C
C  OUTPUTS :            FLCY,FLTCP
C
C  BOTH :               NONE
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

```

```

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

      DOUBLE PRECISION  FLTC(4)
      REAL              TIMONV      , TOFFLT(4)

      DO 10 I=1,4
        IF ( (TOFFLT(I)-T).LE.0.0 ) FLTC(I) = 0.0
10      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 ) FLTCY = 0.0

      END

```

FILE: uuv22.19g/include/uutimer.com

```

      COMMON /TIMER_COMMON/ STAGE1, STAGE2, TIMER, NUMBER_TIMER, NUMBER_TICKS
      INTEGER*4 STAGE1, STAGE2, TIMER(500), NUMBER_TIMER(4,500)
      REAL*8 NUMBER_TICKS(4,500)

```

FILE: uuv22.19g/sutility/makefile

FORFLAGS = code large optimize(3) storage(integer*2)

```

OBJECTS = \
  UUACSTHA.OBJ \
  UUACSTHB.OBJ \
  UUAERO.OBJ \
  UUATMOS1.OBJ \
  UUATMOS2.OBJ \
  UUBAUTO.OBJ \
  UUBGUID.OBJ \
  UUBRTAVG.OBJ \
  UUBSTEER.OBJ \
  UUBTHRST.OBJ \
  UUBXI2FV.OBJ \
  UUCORVEL.OBJ \
  UUCW87.OBJ \
  UUFRACS.OBJ \
  UUFRCTHR.OBJ \
  UUFVDOT.OBJ \
  UUGYRO.OBJ \
  UUINTEG.OBJ \
  UUINTEGI.OBJ \
  UUM3X3I.OBJ \
  UUMCGUID.OBJ \
  UUMISLR.OBJ \
  UUMMK.OBJ \
  UUMMLXY.OBJ \
  UUNCU.OBJ \
  UUNORM.OBJ \
  UUOUTMES.OBJ \
  UURAN.OBJ \
  UURANO.OBJ \
  UURANIT.OBJ \
  UURES2R.OBJ \
  UURTMX.OBJ \
  UUSEEKE.OBJ \
  USSPLAG.OBJ \
  UUTABLE.OBJ \
  UUTLU2EI.OBJ \
  UUVCSH1.OBJ \
  UUTIMER.OBJ

```

LIBRARY = UTILITY.LIB

\$(LIBRARY):\$(OBJECTS)

```
.for.obj:
  ftn286.new $< $(forflags)
  bnd286 $*.obj name($*) object($*.lnk) noload noprint
  rename $*.lnk over $*.obj
  submit :PFP:csd/lib( $(LIBRARY), $* )
```

```
clean:
  delete *.obj,*.lst,$(LIBRARY)
```

FILE: uuv22.19g/sutility/uuacstha.for

```
-----
C      SUBROUTINE ACSTHA(T,CG,ACSLEV,DTACSA,TATAB,TOSEED,
C      .                ITHRES,FXACS,FYACS,FZACS,MXACS,MYACS,MZACS,
C      .                MDOTA,IACSON)
C-----
C
C      SUBROUTINE NAME :      ACSTHA
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 :  none
C
C      INPUTS :             T,CG,ACSLEV,DTACSA,TATAB
C
C      OUTPUTS :            FXACS,FYACS,FZACS,MXACS,MYACS,MZACS,MDOTA,
C                           IACSON
C
C      BOTH :               TOSEED,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  F(3) , FO(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) , INDXB(4)
INTEGER  LENA(4) , LENB(4)
INTEGER*4  TOSEED
```

C LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

SAVE IACSTH , ACSMA

* DATA INITIALIZATION

```
$INCLUDE('^/INCLUDE/SSACSTHR.DAT')
$INCLUDE('^/INCLUDE/SSCON01.DAT')
$INCLUDE('^/INCLUDE/SSCON02.DAT')
$INCLUDE('^/INCLUDE/SSCON03.DAT')
$INCLUDE('^/INCLUDE/SSCON17.DAT')
$INCLUDE('^/INCLUDE/SSCON18.DAT')
$INCLUDE('^/INCLUDE/SSCON19.DAT')
$INCLUDE('^/INCLUDE/SSCON20.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 NORM(AOFFSD,0.0,TOSEED,AOFF1(1))
CALL NORM(AOFFSD,0.0,TOSEED,AOFF1(2))
CALL NORM(AOFFSD,0.0,TOSEED,AOFF1(3))
CALL NORM(AOFFSD,0.0,TOSEED,AOFF1(4))
```

```
AOFF2(1) = 2.0*PI*RAN0(TOSEED)
AOFF2(2) = 2.0*PI*RAN0(TOSEED)
AOFF2(3) = 2.0*PI*RAN0(TOSEED)
AOFF2(4) = 2.0*PI*RAN0(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

```
* The ITHRES assignment was moved to the partition with MCAUTO, KVAUTO
* ITHRES = 0
```

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

```
TIMJNA = 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 TABLE(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 TABLE(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,
            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 TABLE(TMACSA(1,I),THACSA(1,I),TUPA,THA3,
              LENA(I),INDXA(I))
            IF (THA3 .GE. (ACSF-EPSL)) 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
              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
      ENDIF
      TMACSA(IPTR,I) = TIMONA + DTACSA(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. EPSL) 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 TABLE(TMACSA(1,I),THACSA(1,I),TUPA,THA3,LENA(I),
                     INDXA(I))
          IF (THA3 .LT. EPSL) 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
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.0.0 ) THEN
              CALL TABLE(TMACSA(1,I),THACSA(1,I),TREF,ATHRA(I),
                         LENA(I),INDXA(I))
              IF ( ATHRA(I) .GE. ACSF-EPSL ) IACSON = 1
          ELSE
              ATHRA(I) = 0.0
              INDXA(I) = 0
          ENDIF
C      CALCULATE THE FORCES AND MOMENTS PRODUCED BY THE "A"
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 10 J=1,3
        F0(J) = ACSDIR(J,I)*ATHRA(I)
        XMOM(J) = CG(J) - ACSLOC(J,I)
10    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 + ATHRA(I)/ISPACS
20  CONTINUE

      RETURN
      END

```

FILE: uuv22.19g/sutility/uuacsthb.for

```

C-----
C          SUBROUTINE ACSTHB(T,CG,ACSLEV,DTACSB,TATAB,TOSEED,
C          .                ITHRES,FXACS,FYACS,FZACS,MXACS,MYACS,MZACS,
C          .                MDOTA,IACSON)
C-----
C
C          SUBROUTINE NAME :      ACSTHB
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 : none
C
C          INPUTS :            T,CG,ACSLEV,DTACSB,TATAB
C
C          OUTPUTS :           FXACS,FYACS,FZACS,MXACS,MYACS,MZACS,MDOTA,
C                               IACSON
C
C          BOTH :              TOSEED,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)
REAL  DTACSB(4)        , F(3)             , FO(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)      , INDXB(4)
INTEGER      LENA(4)       , LENB(4)
INTEGER*4    TOSEED

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

      SAVE                IACSTH , ACSMA

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSACSTHR.DAT')
$INCLUDE('^/INCLUDE/SSCON01.DAT')
$INCLUDE('^/INCLUDE/SSCON02.DAT')
$INCLUDE('^/INCLUDE/SSCON03.DAT')
$INCLUDE('^/INCLUDE/SSCON17.DAT')
$INCLUDE('^/INCLUDE/SSCON18.DAT')
$INCLUDE('^/INCLUDE/SSCON19.DAT')
$INCLUDE('^/INCLUDE/SSCON20.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 NORM(AOFFSD,0.0,TOSEED,AOFF1(1))
          CALL NORM(AOFFSD,0.0,TOSEED,AOFF1(2))
          CALL NORM(AOFFSD,0.0,TOSEED,AOFF1(3))
          CALL NORM(AOFFSD,0.0,TOSEED,AOFF1(4))

          AOFF2(1) = 2.0*PI*RANO(TOSEED)
          AOFF2(2) = 2.0*PI*RANO(TOSEED)
          AOFF2(3) = 2.0*PI*RANO(TOSEED)
          AOFF2(4) = 2.0*PI*RANO(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
* The ITHRES assignment was moved to the partition with MCAUTO, KVAUTO
*   ITHRES = 0

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 "B" THRUSTERS

      CALL TABLE(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 TABLE(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 TABLE(TMCSB(1,I),THACSB(1,I),TUPA,THB3,
            LENB(I),INDXB(I))
          IF (THB3 .GE. (ACSF-EPSL)) THEN

C           PREVIOUS VALVE STATE WAS HIGH

          TMCSB(1,I) = TATAB
          THACSB(1,I) = ACSF
          IPTR = 2
        ELSE

C           PREVIOUS VALVE STATE WAS DELAY, AND A
C           CROSS-OVER CONDITION HAS OCCURED

          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) + TIMONA)

```

```

      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 TABLE(TMACSB(1,I),THACSB(1,I),TUPA,THR?,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
  ENDIF
101  CONTINUE

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

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.0.0 ) THEN
        CALL TABLE(TMCSB(1,I),THACSB(1,I),TREF,ATHRB(I),
                   LENB(I),INDXB(I))
        IF ( ATRHB(I) .GE. ACSF-EPSL ) IACSON = 1
      ELSE
        ATRHB(I) = 0.0
        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 + ATRHB(I)/ISPACS
40 CONTINUE

      RETURN
      END

```

FILE: uuv22.19g/sutility/uuaero.for

```

C-----
C      SUBROUTINE AERO(T,VRWM,CG,MVRWM,RHO,VSND,IAERO,TBRK,QA,MACH,ALFAT,
C      .              ALFAP,ALFAY,CA,CN,XCP,FXA,FYA,FZA,MXA,MYA,MZA)
C-----
C
C      SUBROUTINE NAME :      AERO
C
C      AUTHOR(S) :          B. HILL
C
C      FUNCTION :           COMPUTE AERODYNAMIC COEFFICIENTS VIA TABLE
C                          LOOK UP AS A FUNCTION OF MACH NUMBER AND
C                          ANGLE OF ATTACK . ALSO CALCULATE THE AERO
C                          FORCES AND MOMENTS .
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  TLU2EI
C
C      INPUTS :             T,VRWM,CG,MVRWM,RHO,VSND
C
C      OUTPUTS :            QA,MACH,ALFAT,ALFAP,ALFAY,CA,CN,XCP,FXA,FYA,
C                          FZA,MXA,MYA,MZA
C
C      BOTH :              IAERO,TBRK
C
C      UPDATES :            B. HILL      - CR # 019
C                          B. HILL      - CR # 022
C                          D. SMITH     - CR # 027
C                          B. HILL      - CR # 030
C                          T. THORNTON  - CR # 037
C                          T. THORNTON  - CR # 043
C                          D. SMITH     - CR # 059
C

```

```

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          B. HILL - CR # 095
C

```

```

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

REAL  CA1M(205)    , CA2M(205)    , CATAB(205)
REAL  CG(3)        , CNA1(205)    , CNA2(205)
REAL  CNTAB(205)   , CPTAB(205)   , MACH
REAL  MACHL        , MVRWM        , MXA
REAL  MYA          , MZA
REAL  VRWM(3)      , XCPL1(205)   , XCPL2(205)

```

```

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

```

```

      SAVE  CATAB , CNTAB , CPTAB ,
      .      ICAM , ICAA , ICNM ,
      .      ICNA , ICPM , ICPA

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSAERO.DAT')
$INCLUDE('~/INCLUDE/SSCON21.DAT')
$INCLUDE('~/INCLUDE/SSCON22.DAT')
$INCLUDE('~/INCLUDE/SSCON23.DAT')
$INCLUDE('~/INCLUDE/SSCON24.DAT')
$INCLUDE('~/INCLUDE/SSCON25.DAT')
$INCLUDE('~/INCLUDE/SSCON71.DAT')

```

```

      IF (IAERO .EQ. 1) THEN

```

```

        IAERO = 0

```

```

        IF (T .LT. TSTG1) THEN

```

```

          SUR = SREF1
          DO 10 I=1,205
            CATAB(I) = CA1M(I)
            CNTAB(I) = CNA1(I)
            CPTAB(I) = XCPL1(I)

```

```

10        CONTINUE

```

```

        ELSE

```

```

          SUR = SREF2
          DO 20 I=1,205
            CATAB(I) = CA2M(I)
            CNTAB(I) = CNA2(I)
            CPTAB(I) = XCPL2(I)

```

```

20        CONTINUE

```

```

      ENDIF

```

```

      ICAM      = 1
      ICAA      = 1
      ICNM      = 1
      ICNA      = 1
      ICPM      = 1
      ICPA      = 1

```

```

      ENDIF

```

```

C      CALCULATE DYNAMIC PRESSURE AND MACH NUMBER

```

```

      QA      = (MVRWM**2)*RHO/2.0
      MACH     = MVRWM/VSND

```

```

C      ZERO AERO FORCES AND MOMENTS WHEN MISSILE VELOCITY IS ZERO

```

```

      IF ( MVRWM.LE.0.0 .OR. (ABS(T-TSTG2).LE.DTEPS) ) THEN
        FXA    = 0.0
        FYA    = 0.0
        FZA    = 0.0
        MXA    = 0.0
        MYA    = 0.0

```

```

      MZA      = 0.0
    ELSE
C      COMPUTE TOTAL, PITCH, AND YAW ANGLES OF ATTACK

      TMP1     = SQRT ( VRWM(2)**2 + VRWM(3)**2 )
      ALFAT    = ATAN2 ( TMP1 , ABS(VRWM(1)) ) / DTR
      ALFAP    = ATAN2 ( VRWM(3) , VRWM(1) ) / DTR
      ALFAY    = ATAN2 ( VRWM(2) , SQRT ( VRWM(1)**2
                                   + VRWM(3)**2 ) ) / DTR

      IF ( ABS(TMP1).GT.1.0E-6 ) THEN
        CPHIA  = VRWM(3) / TMP1
        SPHIA  = VRWM(2) / TMP1
      ELSE
        CPHIA  = 1.0
        SPHIA  = 0.0
      ENDIF

C      AXIAL FORCE COEFFICIENT - F(M,A)

      CALL TLU2EI ( MACH, ALFAT, CATAB, ICAM, ICAA, CA )

C      NORMAL FORCE COEFFICIENT - F(M,A)

      CALL TLU2EI ( MACH, ALFAT, CNTAB, ICNM, ICNA, CN )

C      CENTER-OF-PRESSURE FOR PITCH AND YAW FORCE - F(M,A)

      CALL TLU2EI ( MACH, ALFAT, CPTAB, ICPM, ICPA, XCP )

C      COMPUTE AERODYNAMIC FORCES

      QS      = QA*SUR
      FXA     = QS*CA
      FYA     = -QS*CN*SPHIA
      FZA     = -QS*CN*CPHIA

C      COMPUTE AERODYNAMIC MOMENTS

      MXA     = FYA*CG(3) - FZA*CG(2)
      MYA     = -FXA*CG(3) + FZA*( CG(1) - XCP )
      MZA     = FXA*CG(2) - FYA*( CG(1) - XCP )

    ENDIF

    RETURN
  END

```

FILE: uuv22.19g/sutility/uuatmos1.for

```

C-----
C      SUBROUTINE ATMOS1(T,ALT,RHO,PRESS,VSND)
C-----
C
C      SUBROUTINE NAME :      ATMOS1
C
C      AUTHOR(S) :          DAVID C. FOREMAN
C
C      FUNCTION :           COMPUTES ATMOSPHERIC PROPERTIES AS A
C                           FUNCTION OF ALTITUDE
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  TABLE
C
C      INPUTS :             T,ALT
C
C      OUTPUTS :            RHO,PRESS,VSND
C
C      UPDATES :            T. THORNTON - CR # 003
C                           T. THORNTON - CR # 016
C                           D. SMITH   - CR # 027
C                           B. HILL    - CR # 030
C                           B. HILL    - CR # 036
C                           T. THORNTON - CR # 037
C                           T. THORNTON - CR # 042
C                           D. SMITH   - CR # 059
C                           D. SISSOM  - CR # 069
C                           D. SMITH   - CR # 076
C

```

```

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
C-----
C
C          IMPLICIT REAL      (A-H)
C          IMPLICIT REAL      (O-Z)
C
C          REAL  ALTT(59)      , CIM(9)      , CRI(9)
C          REAL  CRW(9)        , CWR(9)      , LAT
C          REAL  LONG          ,              PRESST(59)
C          REAL  RHOT(59)      , SHEART(59)   , UVRWM(3)
C          REAL          ,              VSNDT(59)
C          REAL  VWINDT(59)    ,              WINDRT(59)

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSATMOS.DAT')
$INCLUDE('~/INCLUDE/SSCON21.DAT')
$INCLUDE('~/INCLUDE/SSCON26.DAT')
$INCLUDE('~/INCLUDE/SSCON27.DAT')

```

```
DATA IALT/ 1 /
```

```
C DETERMINE ATMOSPHERIC DENSITY
```

```
CALL TABLE (ALTT,RHOT,ALT,RHO,59,IALT)
```

```
C DETERMINE ATMOSPHERIC PRESSURE
```

```
CALL TABLE (ALTT,PRESST,ALT,PRESS,59,IALT)
```

```
C DETERMINE SPEED OF SOUND
```

```
CALL TABLE (ALTT,VSNDT,ALT,VSND,59,IALT)
```

```
RETURN
END
```

```
FILE: uuv22.19g/sutility/uuatmos2.for
```

```

C-----
C          SUBROUTINE ATMOS2(T,ALT,XD,YD,ZD,CIM,CRI,LAT, LONG,
C          .              VWIND,SHEAR,VRWM,MVRWM)
C-----
C
C          SUBROUTINE NAME :      ATMOS
C
C          AUTHOR(S) :          DAVID C. FOREMAN
C
C          FUNCTION :           COMPUTES ATMOSPHERIC PROPERTIES AS A
C                               FUNCTION OF ALTITUDE
C
C          CALLED FROM :        FORTRAN MAIN
C
C          SUBROUTINES CALLED :  TABLE , MMK
C
C          INPUTS :              T,ALT,XD,YD,ZD,CIM,CRI,LAT, LONG
C
C          OUTPUTS :             VWIND,SHEAR,VRWM,MVRWM
C
C          UPDATES :             T. THORNTON - CR # 003
C                               T. THORNTON - CR # 016
C                               D. SMITH - CR # 027
C                               B. HILL - CR # 030
C                               B. HILL - CR # 036
C                               T. THORNTON - CR # 037
C                               T. THORNTON - CR # 042
C                               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

```

C B. HILL - CR # 093

```

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

      REAL  ALTT(59)      , CIM(9)      , CRI(9)
      REAL  CRW(9)        , CWR(9)      , LAT
      REAL  LONG          , MVRWM       , PRESST(59)
      REAL  RHOT(59)      , SHEART(59)  , UVRWM(3)
      REAL  VIWIND(3)     , VRWI(3)     , VSRWM(3)
      REAL  VRWIND(3)     , VRWM(3)     , VSNDT(59)
      REAL  VWINDT(59)    , VVWIND(3)   , WINDRT(59)

```

* DATA INITIALIZATION

```

$INCLUDE ('^/INCLUDE/SSATMOS.DAT')
$INCLUDE ('^/INCLUDE/SSCON21.DAT')
$INCLUDE ('^/INCLUDE/SSCON26.DAT')
$INCLUDE ('^/INCLUDE/SSCON27.DAT')

```

DATA IALT/ 1 /

C DETERMINE LOCAL WIND VELOCITY

CALL TABLE (ALTT,VWINDT,ALT,VWIND,59,IALT)

C DETERMINE HORIZONTAL WIND DIRECTION

```

CALL TABLE (ALTT,WINDRT,ALT,WINDIR,59,IALT)
SWDIR = SIN(WINDIR*DTR)
CWDIR = COS(WINDIR*DTR)

```

C DETERMINE VERTICAL WIND COMPONENT

CALL TABLE (ALTT,SHEART,ALT,SHEAR,59,IALT)

C COMPUTE THE TRANSFORMATION FROM THE WIND FRAME TO INERTIAL AND VICE VERSA

CALL MMK(0.0,1,-LAT*DTR,2, LONG*DTR,3,CRW)

```

CWR(1) = CRW(1)
CWR(2) = CRW(4)
CWR(3) = CRW(7)
CWR(4) = CRW(2)
CWR(5) = CRW(5)
CWR(6) = CRW(8)
CWR(7) = CRW(3)
CWR(8) = CRW(6)
CWR(9) = CRW(9)

```

C CALCULATE THE WIND VELOCITY IN WIND FRAME

```

VWVIND(1) = SHEAR
VWVIND(2) = CWDIR*VWIND
VWVIND(3) = SWDIR*VWIND

```

C COMPUTE WIND VELOCITY IN THE INERTIAL FRAME

```

VRWIND(1) = CWR(1)*VWVIND(1) + CWR(4)*VWVIND(2) + CWR(7)*VWVIND(3)
VRWIND(2) = CWR(2)*VWVIND(1) + CWR(5)*VWVIND(2) + CWR(8)*VWVIND(3)
VRWIND(3) = CWR(3)*VWVIND(1) + CWR(6)*VWVIND(2) + CWR(9)*VWVIND(3)

VIWIND(1) = CRI(1)*VRWIND(1) + CRI(4)*VRWIND(2) + CRI(7)*VRWIND(3)
VIWIND(2) = CRI(2)*VRWIND(1) + CRI(5)*VRWIND(2) + CRI(8)*VRWIND(3)
VIWIND(3) = CRI(3)*VRWIND(1) + CRI(6)*VRWIND(2) + CRI(9)*VRWIND(3)

```

C COMPUTE WIND RELATIVE MISSILE VELOCITY

```

VRWI(1) = XD - VIWIND(1)
VRWI(2) = YD - VIWIND(2)
VRWI(3) = ZD - VIWIND(3)

```

```

VRWM(1) = CIM(1)*VRWI(1) + CIM(4)*VRWI(2) + CIM(7)*VRWI(3)
VRWM(2) = CIM(2)*VRWI(1) + CIM(5)*VRWI(2) + CIM(8)*VRWI(3)
VRWM(3) = CIM(3)*VRWI(1) + CIM(6)*VRWI(2) + CIM(9)*VRWI(3)

```

```

MVRWM = SQRT ( VRWM(1)**2 + VRWM(2)**2 + VRWM(3)**2 )
IF ( MVRWM.GT.0.0 ) THEN

```

* FTN286 X415 OPTIMIZE(3)

```

99999  CONTINUE
      UVRWM(1) = VRWM(1) / MVRWM
      UVRWM(2) = VRWM(2) / MVRWM
      UVRWM(3) = VRWM(3) / MVRWM
    ELSE
      UVRWM(1) = 0.0
      UVRWM(2) = 0.0
      UVRWM(3) = 0.0
    ENDIF

    RETURN
  END

```

FILE: uuv22.19g/sutility/uubauto.for

```

C-----
      SUBROUTINE BAUTO(T,THTER,PSIER,SQ,SR,MASS,IYY,IZZ,CGEST,TI2M,RMIR,
      .               VMIR,IBAUTO,CMD,DLPC,DLYC,KTHT,KHTD,XDEL,
      .               XCPCG,LFRACS,CNALP,MDELTA,KNE,KME,MALPHA)
C-----

```

```

C
C  SUBROUTINE NAME :      BAUTO
C
C  AUTHOR(S) :          L. C. HECK, D. C. FOREMAN
C
C  FUNCTION :           PROVIDES CONTROL OF THE MISSILE ABOUT THREE
C                        AXES THROUGHOUT THE BOOST PHASE OF FLIGHT
C
C  CALLED FROM :        FORTRAN MAIN
C
C  SUBROUTINES CALLED :  TABLE , TLU2EI , OPTSCC
C
C  INPUTS :             T,THTER,PSIER,SQ,SR,MASS,IYY,IZZ,CGEST,TI2M,
C                        RMIR,VMIR,IBAUTO
C
C  OUTPUTS :            CMD,DLPC,DLYC,KTHT,KHTD,XDEL,XCPCG,
C                        LFRACS,CNALP,MDELTA,KNE,KME,MALPHA
C
C  UPDATES :            T. THORNTON - CR # 025
C                        D. SMITH   - CR # 027
C                        T. THORNTON - CR # 037
C                        B. HILL    - CR # 038
C                        D. SMITH   - CR # 039
C                        T. THORNTON - CR # 042
C                        T. THORNTON - CR # 046
C                        T. THORNTON - CR # 048
C                        B. HILL    - CR # 056
C                        D. SMITH   - CR # 059
C                        D. SISSOM  - CR # 069
C                        D. SMITH   - CR # 072
C                        B. HILL /  - CR # 081
C                        R. RHYNE   - CR # 087
C                        B. HILL    - CR # 089
C                        D. SMITH   - CR # 092
C                        B. HILL    - CR # 093
C
C-----

```

```

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

      REAL CGEST(3)      , CMD(2)      , TI2M(9)
      REAL KTHT          , KPSI       , KHTD
      REAL KPSID         , TIMTE1(26) , TIMTE2(29)
      REAL THRTE1(26)    , THRTE2(29) , ALTT(59)
      REAL CNA1E(205)    , CNA2E(205) , XCPL1E(205)
      REAL XCPL2E(205)   , RHOT(59)   , VMIR(3)
      REAL LD            , KNE        , KME
      REAL LFRACS        , FRCLOC(3,4) , MALPHA
      REAL MDELTA        , VSNDT(59)  , PRESST(59)
      REAL MCHLIM        , CA1ME(205) , CA2ME(205)
      REAL IYY           , IZZ
      REAL AFR(3,3)      , BFR(3,1)   , KFR(1,3)
      REAL Q1FR(3,3)     , Q2FR(1,1)  , MASS
      REAL RMIR(3)       , VMRWE(3)
      REAL CNTABE(205)   , CPTABE(205) , CATABE(205)

```

C LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

```

      SAVE          ICNME, ICNAE, ICPME, ICPAE,
                    ICAME, ICAAE, IALTE, ITH1E, ITH2E

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSCON33.DAT')
$INCLUDE('~/INCLUDE/SSCON34.DAT')
$INCLUDE('~/INCLUDE/SSCON35.DAT')
$INCLUDE('~/INCLUDE/SSCON21.DAT')
$INCLUDE('~/INCLUDE/SSCON22.DAT')
$INCLUDE('~/INCLUDE/SSCON24.DAT')
$INCLUDE('~/INCLUDE/SSCON26.DAT')
$INCLUDE('~/INCLUDE/SSCON28.DAT')
$INCLUDE('~/INCLUDE/SSCON29.DAT')
$INCLUDE('~/INCLUDE/SSCON30.DAT')
$INCLUDE('~/INCLUDE/SSCON31.DAT')
$INCLUDE('~/INCLUDE/SSCON32.DAT')
$INCLUDE('~/INCLUDE/SSOPTSSC.FUN')

      IF (IBAUTO .EQ. 1) THEN
        IBAUTO = 0

        IF (T .LT. TSTG1) THEN
          DO 10 I=1,205
            CNTABE(I) = CNA1E(I)
            CPTABE(I) = XCPL1E(I)
            CATABE(I) = CA1ME(I)
10          CONTINUE
        ELSE
          DO 20 I=1,205
            CNTABE(I) = CNA2E(I)
            CPTABE(I) = XCPL2E(I)
            CATABE(I) = CA2ME(I)
20          CONTINUE
        ENDIF

        ICNME = 1
        ICNAE = 1
        ICPME = 1
        ICPAE = 1
        ICAME = 1
        ICAAE = 1
        IALTE = 1
        ITH1E = 1
        ITH2E = 1
      ENDIF

C      COMPUTE ESTIMATED ALTITUDE

      ESTALT = SQRT ( RMIR(1)**2 + RMIR(2)**2 + RMIR(3)**2 ) - RADE

C      COMPUTE ESTIMATED ATMOSPHERIC PROPERTIES

      CALL TABLE(ALT, RHOT , ESTALT, ESTRHO, 59, IALTE)
      CALL TABLE(ALT, PRESST, ESTALT, ESTPRE, 59, IALTE)
      CALL TABLE(ALT, VSNDT , ESTALT, ESTVSD, 59, IALTE)

C      COMPUTE ESTIMATED WIND RELATIVE VELOCITY COMPONENTS

      VMRWE(1) = VMIR(1)*TI2M(1) + VMIR(2)*TI2M(4) + VMIR(3)*TI2M(7)
      VMRWE(2) = VMIR(1)*TI2M(2) + VMIR(2)*TI2M(5) + VMIR(3)*TI2M(8)
      VMRWE(3) = VMIR(1)*TI2M(3) + VMIR(2)*TI2M(6) + VMIR(3)*TI2M(9)

C      COMPUTE ESTIMATED MACH NUMBER AND DYNAMIC PRESSURE

      ESTVEL = SQRT ( VMRWE(1)**2 + VMRWE(2)**2 + VMRWE(3)**2 )
      ESTMCH = ESTVEL/ESTVSD
      ESTQA = ESTRHO*ESTVEL**2/2.0

C      CALCULATE ESTIMATED VACUUM THRUST

      IF ( T.GE.TSTG1 ) THEN
        TO = T - TST2ON
        SREFE = SREF2
        XNOZE = XNOZ2
        AEXITE = AEXIT2
        CALL TABLE(TIMTE2, THRTE2, TO, THRVE, 29, ITH2E)
      ELSE
        TO = T - TIGN
        SREFE = SREF1

```

```

      XNOZE = XNOZ1
      AEXITE = AEXIT1
      CALL TABLE(TIMTE1, THRTE1, TO, THRVE, 26, ITH1E)
    ENDIF

C   COMPUTE ESTIMATED DELIVERED THRUST
      THRE = THRVE - AEXITE*ESTPRE
      IF ( THRE.LT.0.0 ) THRE = 0.0

C   COMPUTE ESTIMATED TOTAL ANGLE OF ATTACK IN DEGREES AND PLANAR
C   ANGLES OF ATTACK IN RADIAN .
      IF ( ESTVEL.GT.0.0 ) THEN
        ALFATE = ATAN2 ( SQRT(VMRWE(2)**2 + VMRWE(3)**2),
          ABS(VMRWE(1)) )/DTR
        ALFAPE = ATAN2 ( VMRWE(3) , VMRWE(1) )
        ALFAYE = ATAN2 ( - VMRWE(2) , VMRWE(1) )
      ELSE
        ALFATE = 0.0
        ALFAPE = 0.0
        ALFAYE = 0.0
      ENDIF

      CALL TLU2EI ( ESTMCH, 4.0 , CNTABE , ICNME, ICNAE, CNE )
      CALL TLU2EI ( ESTMCH, ALFATE, CPTABE , ICPME, ICPAE, XCPE )
      CALL TLU2EI ( ESTMCH, ALFATE, CATABE , ICAME, ICAAE, CAE )

C   CALCULATE CNALFA (PER RADIAN)
      CNALP = CNE/(4.0*DTR)

C   ESTIMATE DRAG FORCE
      DRAGE = CAE*ESTQA*SREFE

C   COMPUTE AERODYNAMIC MOMENT ARM
      XCPCG = XCPE - CGEST(1)

C   TVC AUTOPILOT
      IF ( T.LT.TFRCS .AND. T.GE.TINHIB ) THEN
        XDEL = CGEST(1) - XNOZE
        MALPHA = ABS(CNALP*XCPCG*SREFE*ESTQA/IYY)
        KTHT = - (IYY*WMTVC**2 + CNALP*SREFE*ESTQA*XCPCG)
          / (THRE*XDEL)
        KPSI = - (IZZ*WMTVC**2 + CNALP*SREFE*ESTQA*XCPCG)
          / (THRE*XDEL)
        KTHTD = - 2.0*ZETTVC*WMTVC*IYY/(THRE*XDEL)
        KPSID = - 2.0*ZETTVC*WMTVC*IZZ/(THRE*XDEL)

C   AUTOPILOT PITCH AND YAW CONTROL FOR THRUST VECTOR CONTROL
        CMMD(1) = THTER*KTHT - SQ*KTHTD
        CMMD(2) = PSIER*KPSI - SR*KPSID

C   COMPUTE BUCKET LIMIT ON NOZZLE COMMANDS
        TOTCMD = SQRT(CMMD(1)**2 + CMMD(2)**2)
        IF ( TOTCMD.GT.BCKLMT ) THEN
          CMMD(1) = CMMD(1)*BCKLMT/TOTCMD
          CMMD(2) = CMMD(2)*BCKLMT/TOTCMD
        ENDIF
      ELSE
        KTHT = 4.0
        KPSI = 4.0
        KTHTD = 4.0
        KPSID = 4.0
        CMMD(1) = 0.0
        CMMD(2) = 0.0
      ENDIF

C   FORWARD REACTION CONTROL SYSTEM AUTOPILOT
      IF ( T.GE.TFRCS ) THEN

```

```

C      COMPUTE FORCE AND MOMENT MULTIPLIERS

      LD      = ( XJET - XNOZE )/DJET
      CT      = THJET/(ESTQA*SJET)
      TMP1    = SQRT ( CT )
      IF ( ESTMCH.LE.MCHLIM ) THEN
         KNE   = 0.0118 + (0.1358*(1.0485*SQRT(LD))/TMP1)
                + 0.0946*ESTMCH + 0.004317/LD
      ELSE
         KNE   = 1.0 + EXP(1.1 - 0.2116*(ALOG(CT)+8.5)**1.4)
      ENDIF
      KME     = 0.5582 - 0.1884/TMP1 - 1.9659/LD

C      DETERMINE AIRFRAME COEFFICIENTS FOR PLANT MODEL
C      NOTE : AN ALTERNATE CALCULATION FOR MDELTA IS
C      MDELTA = (-KME*DJET + KNE*LFRACS)/IYY

      TMP1    = ESTQA*SREFE*CNALP
      TMP2    = MASS*ESTVEL
      LFRACS  = FRCLOC(1,1) - CGEST(1)
      MALPHA  = TMP1*XCPCG/IYY
      MDELTA  = - KNE*LFRACS/IYY
      ZALPHA  = ( THRE + DRAGE + TMP1 )/TMP2
      ZDELTA  = - KNE/TMP2

C      ESTIMATE MAXIMUM ANGLE OF ATTACK

      ALFAMX  = ABS ( THJET*MDELTA/MALPHA )

C      SET PLANT WEIGHTING MATRIX

      Q1FR(1,1) = 1.0/THERMX**2
      Q1FR(2,2) = 1.0/THDTMX**2
      Q1FR(3,3) = 1.0/ALFAMX**2

C      SET INPUT WEIGHTING MATRIX

      Q2FR(1,1) = 1.0/(KNE*THJET)**2

C      INITIALIZE ANALOG PLANT MODEL

      AFR(1,1) = 0.0
      AFR(1,2) = 1.0
      AFR(1,3) = 0.0
      AFR(2,1) = 0.0
      AFR(2,2) = 0.0
      AFR(2,3) = MALPHA
      AFR(3,1) = 0.0
      AFR(3,2) = 1.0
      AFR(3,3) = - ZALPHA

      BFR(1,1) = 0.0
      BFR(2,1) = MDELTA
      BFR(3,1) = - ZDELTA

C      COMPUTE STEADY STATE OPTIMAL CONTROL GAINS
*      CALL OPTSSC(AFR,BFR,3,1,DTAPU,Q1FR,Q2FR,KFR)
      KFR(1,1) = OPTSSC1(ESTALT)
      KFR(1,2) = OPTSSC2(ESTALT)
      KFR(1,3) = OPTSSC3(ESTALT)

C      COMPUTE DESIRED PLANAR CONTROL FORCES

      FCMDP   = KFR(1,1)*THTER - KFR(1,2)*SQ - KFR(1,3)*ALFAPE
      FCMDY   = - KFR(1,1)*PSIER + KFR(1,2)*SR + KFR(1,3)*ALFAYE

C      COMPUTE DURATION OF NEEDED VALVE OPEN PULSES

      DLPC    = FCMDP/(KNE*THJET)
      DLYC    = FCMDY/(KNE*THJET)

      ELSEIF ( T.LT.TFRCS ) THEN
         DLPC  = 0.0
         DLYC  = 0.0
      ENDIF

      RETURN
      END

```

FILE: uuv22.19g/sutlity/uubguid.for

```

C-----
C      SUBROUTINE BGUID(T,AT,AC,TI2M,PG,IMINSF,VW,PGD,VWD,WC,PSIER,
C      .               TH,ER,PM,KA,KV)
C-----
C      SUBROUTINE NAME :      BGUID
C
C      AUTHOR(S) :           L. C. HECK, D. C. FOREMAN
C
C      FUNCTION :            TO CALCULATE THE ERROR BETWEEN THE COMMANDED
C                           POINTING VECTOR AND THE ACTUAL DIRECTION THE
C                           MISSILE IS POINTED DURING BOOST
C
C      CALLED FROM :         FORTRAN MAIN
C
C      SUBROUTINES CALLED :  TABLE, intrinsics
C
C      INPUTS :              T,AT,AC,TI2M
C
C      OUTPUTS :             PGD,VWD,WC,PSIER,THTER,PM,KA,KV
C
C      BOTH :               PG,IMINSF,VW
C
C      UPDATES :            T. THORNTON - CR # 006
C                           T. THORNTON - CR # 016
C                           T. THORNTON - CR # 025
C                           B. HILL    - CR # 030
C                           T. THORNTON - CR # 037
C                           T. THORNTON - CR # 042
C                           T. THORNTON - CR # 046
C                           D. SMITH   - CR # 059
C                           D. SMITH   - CR # 072
C                           B. HILL /  - CR # 081
C                           R. RHYNE   - CR # 092
C                           D. SMITH   - CR # 092
C                           B. HILL    - CR # 093
C-----

```

```

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

      REAL  VWD(3)           , VW(3)           , WC(3)
      REAL  PGD(3)           , PG(3)           , PM(3)
      REAL  AT(3)            , AC(3)           , TI2M(9)
      REAL  KA                , KA1            , KA2
      REAL  KA3              , KA4            , KA5
      REAL  KV                , KV1            , KV2
      REAL  KV3              , KV4            , KV5
      REAL  ATTLT(5)         , ATTLMT(5)

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSCON36.DAT')
$INCLUDE('~/INCLUDE/SSCON37.DAT')
$INCLUDE('~/INCLUDE/SSCON38.DAT')
$INCLUDE('~/INCLUDE/SSCON22.DAT')
$INCLUDE('~/INCLUDE/SSCON30.DAT')

```

```
DATA IATLMT / 1 /
```

```
C      COMPUTE POINTING VECTOR COMMAND GAINS
```

```

      IF ( T.LE.TS ) THEN
        KA = KA1
        KV = KV1
      ELSEIF ( T.LE.TSTG1 ) THEN
        KA = KA2
        KV = KV2
      ELSEIF ( T.LE.T5 ) THEN
        KA = KA3
        KV = KV3
      ELSEIF ( T.LE.T2S ) THEN
        KA = KA4
        KV = KV4
      ELSE
        KA = KA5
        KV = KV5
      ENDIF

```

```

C      COMPUTE COMMAND ANGULAR VELOCITY INTEGRAL (VW) DERIVATIVE
      VWD(1) = AT(2)*AC(3) - AT(3)*AC(2)
      VWD(2) = AT(3)*AC(1) - AT(1)*AC(3)
      VWD(3) = AT(1)*AC(2) - AT(2)*AC(1)

C      LAUNCH STEERING MODE
      IF ( T.LE.TC ) THEN

C          COMPUTE THE COMMAND ANGULAR VELOCITY VECTOR (WC)
          WC(1) = 0.0
          WC(2) = 0.0
          WC(3) = 0.0

C          COMPUTE POINTING VECTOR (PG) DERIVATIVE
          PGD(1) = 0.0
          PGD(2) = 0.0
          PGD(3) = 0.0

C      MINIMUM IMPULSE STEERING MODE
      ELSEIF ( T.LE.T5 ) THEN

C          RESET POINTING ON FIRST PASS THROUGH MINS LOGIC
          IF ( IMINSF.EQ.0 ) THEN
              PG(1) = AC(1)
              PG(2) = AC(2)
              PG(3) = AC(3)
              IMINSF = 1
          ENDIF

C          COMPUTE THE COMMAND ANGULAR VELOCITY VECTOR (WC)
          WC(1) = KA*VWD(1) + KV*VW(1)
          WC(2) = KA*VWD(2) + KV*VW(2)
          WC(3) = KA*VWD(3) + KV*VW(3)
          WCMAX = AMAX1(ABS(WC(1)),ABS(WC(2)),ABS(WC(3)))
          IF ( WCMAX.GT.WLIM ) THEN
              SCALE = WLIM/WCMAX
              WC(1) = SCALE*WC(1)
              WC(2) = SCALE*WC(2)
              WC(3) = SCALE*WC(3)
          ENDIF

C          COMPUTE POINTING VECTOR (PG) DERIVATIVE
          PGD(1) = WC(2)*PG(3) - WC(3)*PG(2)
          PGD(2) = WC(3)*PG(1) - WC(1)*PG(3)
          PGD(3) = WC(1)*PG(2) - WC(2)*PG(1)

C      SET POINTING VECTOR COINCIDENT WITH STEERING VECTOR DURING
C      FRACS
      IF ( T.GE.TFRCS ) THEN
          PG(1) = AC(1)
          PG(2) = AC(2)
          PG(3) = AC(3)
      ENDIF

C      GENERALIZED ENERGY MANAGEMENT STEERING MODE
      ELSEIF ( T.LE.TCD ) THEN

C          COMPUTE COMMAND ANGULAR VELOCITY INTEGRAL (VW)
          VWMAX = AMAX1(ABS(VW(1)),ABS(VW(2)),ABS(VW(3)))
          IF ( VWMAX.GT.VWLIM ) THEN
              SCALE = VWLIM/VWMAX
              VW(1) = SCALE*VW(1)
              VW(2) = SCALE*VW(2)
              VW(3) = SCALE*VW(3)
          ENDIF

C          COMPUTE THE COMMAND ANGULAR VELOCITY VECTOR (WC)
          WC(1) = KA*VWD(1) + KV*VW(1)

```

```

WC(2) = KA*VWD(2) + KV*VW(2)
WC(3) = KA*VWD(3) + KV*VW(3)
WCMAX = AMAX1(ABS(WC(1)),ABS(WC(2)),ABS(WC(3)))
IF ( WCMAX.GT.WLIM ) THEN
  SCALE = WLIM/WCMAX
  WC(1) = SCALE*WC(1)
  WC(2) = SCALE*WC(2)
  WC(3) = SCALE*WC(3)
ENDIF

C      COMPUTE POINTING VECTOR (PG) DERIVATIVE

PGD(1) = WC(2)*PG(3) - WC(3)*PG(2)
PGD(2) = WC(3)*PG(1) - WC(1)*PG(3)
PGD(3) = WC(1)*PG(2) - WC(2)*PG(1)

C      COUNTDOWN STEERING MODE
ELSE

C      COMPUTE COMMAND ANGULAR VELOCITY INTEGRAL (VW)

VWMAX = AMAX1(ABS(VW(1)),ABS(VW(2)),ABS(VW(3)))
IF ( VWMAX.GT.VWLIM ) THEN
  SCALE = VWLIM/VWMAX
  VW(1) = SCALE*VW(1)
  VW(2) = SCALE*VW(2)
  VW(3) = SCALE*VW(3)
ENDIF

C      COMPUTE THE COMMAND ANGULAR VELOCITY VECTOR (WC)

WC(1) = 0.0
WC(2) = 0.0
WC(3) = 0.0

C      COMPUTE POINTING VECTOR (PG) DERIVATIVE

PGD(1) = 0.0
PGD(2) = 0.0
PGD(3) = 0.0
ENDIF

C      TRANSFORM THE POINTING VECTOR FROM THE INERTIAL GUIDANCE FRAME
C      INTO THE MISSILE BODY FRAME

PM(1) = PG(1)*TI2M(1) + PG(2)*TI2M(4) + PG(3)*TI2M(7)
PM(2) = PG(1)*TI2M(2) + PG(2)*TI2M(5) + PG(3)*TI2M(8)
PM(3) = PG(1)*TI2M(3) + PG(2)*TI2M(6) + PG(3)*TI2M(9)

C      COMPUTE THE ERROR SIGNAL SENT TO THE AUTOPILOT

PSIER = PM(2)
THTER = -PM(3)

C      LIMIT ATTITUDE ERRORS SENT TO THE AUTOPILOT

CALL TABLE(ATTLTT,ATLMT,T,ATLTM,5,IATLTM)
TOTERR = SQRT ( PSIER**2 + THTER**2 )
IF ( TOTERR.GT.ATLTM ) THEN
  PSIER = PSIER*ATLTM/TOTERR
  THTER = THTER*ATLTM/TOTERR
ENDIF

RETURN
END

```

FILE: uuv22.19g/sutility/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-----

```

```

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

```

```

REAL  TN(9),      TA(9),      W(3)
REAL  TD(9),      TI(9),      TE(9)

```

```

C   COMPUTE INVERSE OF PREVIOUS TRANSFORMATION MATRIX

```

```

CALL M3X3I ( TA , TI )

```

```

C   COMPUTE DELTA ROTATION MATRIX FROM PREVIOUS MISSILE ATTITUDE TO CURRENT
C   MISSILE ATTITUDE

```

```

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

```

```

C   DETERMINE DELTA EULER ANGLES FROM PREVIOUS ORIENTATION ( EULER ROTATION
C   SEQUENCE IS PSI-THETA-PHI )

```

```

DLPSI = ATAN2 ( TD(4) , TD(1) )
DLTHE = ASIN ( -TD(7) )
DLPHI = ATAN2 ( TD(8) , TD(9) )

```

```

CDLPSI = COS ( DLPSI )
SDLPSI = SIN ( DLPSI )
CDLTHE = COS ( DLTHE )
SDLTHE = SIN ( DLTHE )
CDLPHI = COS ( DLPHI )
SDLPHI = SIN ( DLPHI )

```

```

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

```

```

TE(1) = 1.0
TE(2) = 0.0
TE(3) = 0.0
TE(4) = 0.0
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) + 1.0
TD(2) = TE(2)
TD(3) = TE(3)
TD(4) = TE(4)
TD(5) = TE(5) + 1.0
TD(6) = TE(6)
TD(7) = TE(7)
TD(8) = TE(8)
TD(9) = TE(9) + 1.0

```

```

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

RETURN
END

```

FILE: uuv22.19g/sutility/uubsteer.for

```

C-----
C      SUBROUTINE BSTEER(T,USI,USF,UVS,MVS,MVR,AT,RMIR,VMIR,US,USD,AC,
C      .                WASTAN,VRATIO,VELWD)
C-----
C
C      SUBROUTINE NAME :      BSTEER
C
C      AUTHOR(S) :           L. C. HECK, D. C. FOREMAN
C
C      FUNCTION :            CALCULATES THE STEERING COMMANDS FOR THE
C                           BOOST PHASE OF FLIGHT
C
C      CALLED FROM :         FORTRAN MAIN
C
C      SUBROUTINES CALLED :   NONE
C
C      INPUTS :              T,USI,USF,UVS,MVS,MVR,AT,RMIR,VMIR
C
C      OUTPUTS :             USD,AC,WASTAN,VRATIO,VELWD
C
C      BOTH :               US
C
C      UPDATES :             T. THORNTON - CR # 005
C                           T. THORNTON - CR # 016
C                           T. THORNTON - CR # 025
C                           B. HILL    - CR # 030
C                           T. THORNTON - CR # 037
C                           T. THORNTON - CR # 042
C                           T. THORNTON - CR # 046
C                           D. SMITH   - CR # 059
C                           D. SMITH   - CR # 072
C                           D. SMITH   - CR # 073
C                           B. HILL /  - CR # 081
C                           R. RHYNE   - CR # 092
C                           D. SMITH   - CR # 093
C                           B. HILL    - CR # 093
C-----
C
C      IMPLICIT REAL          (A-H)
C      IMPLICIT REAL          (O-Z)
C
C      REAL  USF(3)           , UVS(3)           , DBAR(3)
C      REAL  USD(3)           , AT(3)            , US(3)
C      REAL  AC(3)            , BBAR(3)          , BIGAC(3)
C      REAL  BIGB(3)          , MVS              , MVR
C      REAL  KS1              , KB               , MBIGAC
C      REAL  MBIGB            , VMIR(3)          , USI(3)
C      REAL  RMIR(3)          , GREST(3)
C
C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG
C
C      SAVE                  IBSTR
C
C      * DATA INITIALIZATION
C      $INCLUDE('^/INCLUDE/SSCON36.DAT')
C      $INCLUDE('^/INCLUDE/SSCON38.DAT')
C      $INCLUDE('^/INCLUDE/SSCON39.DAT')
C      $INCLUDE('^/INCLUDE/SSCON40.DAT')
C      $INCLUDE('^/INCLUDE/SSCON30.DAT')

```

```
DATA IBSTR / 1 /
```

```
IF ( IBSTR.EQ.1 ) THEN
```

```
    IBSTR = 0
```

```
C      INITIALIZE FLAG WHICH ENABLES RESET OF STEERING VECTOR AT
C      FRACS INITIATION
```

```
    IF ( T.GT.TFRCS ) THEN
```

```
        ISETUS = 1
```

```
    ELSEIF ( T.LE.TFRCS ) THEN
```

```
        ISETUS = 0
```

```
    ENDIF
```

```
ENDIF
```

```
C      LAUNCH STEERING LOGIC
```

```
IF ( T.LE.TC ) THEN
```

```
    USD(1) = 0.0
```

```
    USD(2) = 0.0
```

```
    USD(3) = 0.0
```

```
    AC(1) = US(1)
```

```
    AC(2) = US(2)
```

```
    AC(3) = US(3)
```

```
C      MINIMUM IMPULSE STEERING (MINS) LOGIC
```

```
ELSEIF ( T.LE.T5 ) THEN
```

```
C      RESET UNIT STEERING VECTOR AT FRACS INITIATION
```

```
IF ( T.GE.TFRCS .AND. ISETUS.EQ.0 ) THEN
```

```
    TMP1 = SQRT ( VMIR(1)**2 + VMIR(2)**2 + VMIR(3)**2 )
```

```
    US(1) = VMIR(1)/TMP1
```

```
    US(2) = VMIR(2)/TMP1
```

```
    US(3) = VMIR(3)/TMP1
```

```
    ISETUS = 1
```

```
ENDIF
```

```
C      CALCULATE STEERING VECTOR DERIVATIVE
```

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

```
C      ESTIMATE GRAVITY VECTOR
```

```
    TMP1 = SQRT ( RMIR(1)**2 + RMIR(2)**2 + RMIR(3)**2 )
```

```
    TMP3 = TMP1**3
```

```
    GREST(1) = - GMU*RMIR(1)/TMP3
```

```
    GREST(2) = - GMU*RMIR(2)/TMP3
```

```
    GREST(3) = - GMU*RMIR(3)/TMP3
```

```
C      ESTIMATE TURNING RATE DUE TO GRAVITY
```

```
    TMP4 = SQRT ( VMIR(1)**2 + VMIR(2)**2 + VMIR(3)**2 )
```

```
    TMP1 = ( GREST(2)*US(3) - GREST(3)*US(2) )/TMP4
```

```
    TMP2 = ( GREST(3)*US(1) - GREST(1)*US(3) )/TMP4
```

```
    TMP3 = ( GREST(1)*US(2) - GREST(2)*US(1) )/TMP4
```

```
    USD(1) = US(2)*TMP3 - US(3)*TMP2
```

```
    USD(2) = US(3)*TMP1 - US(1)*TMP3
```

```
    USD(3) = US(1)*TMP2 - US(2)*TMP1
```

```
ELSE
```

```
    USDOT = US(1)*USI(1) + US(2)*USI(2) + US(3)*USI(3)
```

```
    USD(1) = KS1*(USI(1) - USDOT*US(1))
```

```
    USD(2) = KS1*(USI(2) - USDOT*US(2))
```

```
    USD(3) = KS1*(USI(3) - USDOT*US(3))
```

```
ENDIF
```

```
    AC(1) = US(1)
```

```
    AC(2) = US(2)
```

```
    AC(3) = US(3)
```

```
C      GENERAL ENERGY MANAGEMENT (GEMS) STEERING LOGIC
```

```
ELSEIF ( T.LE.TCD ) THEN
```

```
    USD(1) = 0.0
```

```
    USD(2) = 0.0
```

```
    USD(3) = 0.0
```

```
    US(1) = UVS(1)
```

```
    US(2) = UVS(2)
```

```

US(3)   = UVS(3)

BIGB(1) = DBAR(2)*US(3) - DBAR(3)*US(2)
BIGB(2) = DBAR(3)*US(1) - DBAR(1)*US(3)
BIGB(3) = DBAR(1)*US(2) - DBAR(2)*US(1)

MBIGB   = SQRT(BIGB(1)**2 + BIGB(2)**2 + BIGB(3)**2)
BBAR(1) = BIGB(1)/MBIGB
BBAR(2) = BIGB(2)/MBIGB
BBAR(3) = BIGB(3)/MBIGB

IF ( MVR.NE.0.0 ) THEN
  VRATIO = MVS/MVR
ENDIF

IF ( MVS.LE.MVR ) THEN
  WASTAN = KB*(1.0 - VRATIO)**0.5
ELSE
  WASTAN = 0.0
ENDIF

SINWAN   = VRATIO*WASTAN
COSWAN   = 1.0 - WASTAN**2/2.0
BIGAC(1) = US(1)*COSWAN - BBAR(1)*SINWAN
BIGAC(2) = US(2)*COSWAN - BBAR(2)*SINWAN
BIGAC(3) = US(3)*COSWAN - BBAR(3)*SINWAN

MBIGAC   = SQRT(BIGAC(1)**2 + BIGAC(2)**2 + BIGAC(3)**2)
AC(1)    = BIGAC(1)/MBIGAC
AC(2)    = BIGAC(2)/MBIGAC
AC(3)    = BIGAC(3)/MBIGAC

ATDTUS   = AT(1)*US(1) + AT(2)*US(2) + AT(3)*US(3)
VELWD    = SQRT(AT(1)**2 + AT(2)**2 + AT(3)**2) - ATDTUS

C      COUNTDOWN STEERING LOGIC

ELSE
  AC(1)   = US(1)
  AC(2)   = US(2)
  AC(3)   = US(3)
  VELWD   = 0.0
ENDIF

RETURN
END

```

FILE: uu22.19g/sutlity/uubthrst.for

```

C-----
C      SUBROUTINE BTHRST(T,CG,EISP,PRESS,DLP,DLY,TOSEED,TBRK,IBTHR,FXT,
C      .               FYT,FZT,MXT,MYT,MZT,MDOTT,THRV,THR)
C-----
C
C      SUBROUTINE NAME :      BTHRST
C
C      AUTHOR(S) :          L. D. HUEBNER, D. C. FOREMAN
C
C      FUNCTION :           COMPUTES MISSILE THRUST VECTOR AND MOMENTS
C                           DUE TO FIRST AND SECOND STAGE BOOSTERS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  TABLE
C
C      INPUTS :             T,CG,EISP,PRESS,DLP,DLY,IBTHR
C
C      OUTPUTS :            FXT,FYT,FZT,MXT,MYT,MZT,MDOTT,THRV,THR
C
C      BOTH :              TOSEED,TBRK
C
C      UPDATES :            T. THORNTON - CR # 002
C                           T. THORNTON - CR # 016
C                           D. SMITH   - CR # 027
C                           T. THORNTON - CR # 037
C                           B. HILL    - CR # 038
C                           D. SMITH   - CR # 039
C                           T. THORNTON - CR # 043
C                           T. THORNTON - CR # 046
C

```

```

C          D. SMITH   - CR # 059
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          D. SMITH   - CR # 092
C          B. HILL   - CR # 093
C

```

```

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

```

```

REAL  TIMTH1(26) , THRTH1(26) , TIMTH2(29)
REAL  THRTH2(29) , CG(3)      , THRMA(9)
REAL  MXT      , MYT      , MZT
REAL  MDOTT    , BOFF1(2)   , BOFF2(2)

```

```

INTEGER*4          TOSEED

```

```

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

```

```

      SAVE  ITH1 , ITH2 , AEXIT , XNOZ , THRMA

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSBTHRST.DAT')
$INCLUDE('~/INCLUDE/SSCON17.DAT')
$INCLUDE('~/INCLUDE/SSCON22.DAT')
$INCLUDE('~/INCLUDE/SSCON23.DAT')
$INCLUDE('~/INCLUDE/SSCON29.DAT')
$INCLUDE('~/INCLUDE/SSCON31.DAT')
$INCLUDE('~/INCLUDE/SSCON32.DAT')
$INCLUDE('~/INCLUDE/SSCON41.DAT')

```

```

      DATA ITH1, ITH2 / 2*1 /

```

```

      IF (IBTHR .EQ. 1) THEN

```

```

        IBTHR = 0

```

```

        IF (T .LT. TSTG1) THEN

```

```

          AEXIT = AEXIT1

```

```

          XNOZ  = XNOZ1

```

```

          IF (T .EQ. 0.0) THEN

```

```

            BOFF2(1) = 2.0*PI*RAN0(TOSEED)

```

```

            BOFF2(2) = 2.0*PI*RAN0(TOSEED)

```

```

          ENDIF

```

```

          OFF1      = BOFF1(1)

```

```

          OFF2      = BOFF2(1)

```

```

        ELSE

```

```

          AEXIT = AEXIT2

```

```

          XNOZ  = XNOZ2

```

```

          OFF1  = BOFF1(2)

```

```

          OFF2  = BOFF2(2)

```

```

        ENDIF

```

```

        COFF1   = COS(OFF1)

```

```

        SOFF1   = SIN(OFF1)

```

```

        COFF2   = COS(OFF2)

```

```

        SOFF2   = SIN(OFF2)

```

```

C      CALCULATE DIRECTION OF BOOSTER NOZZLE MISALIGNMENT

```

```

      THRMA(1) = COFF1

```

```

      THRMA(2) = SOFF1*COFF2

```

```

      THRMA(3) = SOFF1*SOFF2

```

```

      THRMA(4) = SOFF1*SOFF2

```

```

      THRMA(5) = COFF1

```

```

      THRMA(6) = SOFF1*COFF2

```

```

      THRMA(7) = SOFF1*COFF2

```

```

      THRMA(8) = SOFF1*SOFF2

```

```

      THRMA(9) = COFF1

```

```

      ENDIF

```

```

      IF (T .LT. TSTG2) THEN

```

```

C      SFCOND STAGE BOOST

```

```

      IF ( T.GE.TSTG1 ) THEN

```

```

        T0 = T - TST2ON

```

```

      CALL TABLE(TIMTH2,THRTH2,T0,THRV,29,ITH2)
C     FIRST STAGE BOOST
      ELSE
        TO = T - TIGN
        CALL TABLE(TIMTH1,THRTH1,T0,THRV,26,ITH1)
      ENDIF
C     COMPUTE DELIVERED THRUST
      THR = AMAX1 ( 0.0 , THRV - AEXIT*PRESS )
C     RESOLVE DELIVERED THRUST FROM GIMBAL TO BODY COORDINATES
      FX = THR*COS(DLP)*COS(DLY)
      FY = THR*SIN(DLY)
      FZ = - THR*SIN(DLP)*COS(DLY)
C     INCORPORATE THRUSTER MISALIGNMENTS
      FXT = FX*THRMA(1) + FY*THRMA(4) + FZ*THRMA(7)
      FYT = FX*THRMA(2) + FY*THRMA(5) + FZ*THRMA(8)
      FZT = FX*THRMA(3) + FY*THRMA(6) + FZ*THRMA(9)
C     CALCULATE THE MOMENTS DUE TO THRUST
      MXT = FYT*CG(3) - FZT*CG(2)
      MYT = - FXT*CG(3) + FZT*(CG(1) - XNOZ)
      MZT = FXT*CG(2) - FYT*(CG(1) - XNOZ)
C     CALCULATE MASS EXPULSION RATE
      MDOTT = THRV/EISP
      ELSE
        FXT = 0.0
        FYT = 0.0
        FZT = 0.0
        MXT = 0.0
        MYT = 0.0
        MZT = 0.0
        MDOTT = 0.0
        THR = 0.0
        THRV = 0.0
      ENDIF
      RETURN
      END

```

FILE: uuv22.19g/sutility/uubxi2fv.for

```

C-----
      SUBROUTINE BXI2FV ( FVM, B, FV )
C-----
C
C     SUBROUTINE NAME :      BXI2FV
C
C     AUTHOR(S) :          W. E. EXELY
C
C     FUNCTION :           COMPUTE QUATERNION (FV) ATTITUDE PARAMETERS
C                          FROM A BODY TO INERTIAL TRANSFORMATION
C                          MATRIX (B) AND SET THE SQUARE OF MAGNITUDE
C                          OF QUATERNION TO (FVM)
C
C     CALLED FROM :        FORTRAN MAIN
C
C     SUBROUTINES CALLED :  NONE
C
C     INPUTS :              FVM,B
C
C     OUTPUTS :             FV
C
C     UPDATES :             D. SMITH   - CR # 59
C-----
C

```

```

      IMPLICIT REAL (A-H)
      IMPLICIT REAL (O-Z)
C
      DIMENSION  B ( 9 ),  FV ( 4 )
C
      EQUIVALENCE ( T3 , Q ), ( B1 , AA )
C
      DATA  F4, F2, P25, P0001 /  4., 2., 0.25, 0.0001  /
      DATA  F1, F0           /  1., 0.                /
C
      T3 = P25
C
      A1 = B(6) - B(8)
      A2 = B(7) - B(3)
      A3 = B(2) - B(4)
C
      TRA = B(1) + B(5) + B(9) + F1
C
      IF ( TRA .LT. P0001 ) GO TO 100
C
      FV(4) = SQRT(T3*TRA)
      T3 = T3/FV(4)
      FV(1) = T3*A1
      FV(2) = T3*A2
      FV(3) = T3*A3
      GO TO 200
C
100  CONTINUE
C
      SETUP FOR ILL-CONDITION ... TRA = 0 (LT .0001)
C
      IFLAG = 0
C
      TRA = F2 - TRA
      B1 = T3*( B(1) + B(1) + TRA )
      IF ( B1 .LT. F0 ) B1 = F0
      FV(1) = SQRT( B1 )
C
      IF( FV(1) .NE. F0 ) IFLAG = 1
C
      B1 = T3*( B(5) + B(5) + TRA )
      IF ( B1 .LT. F0 ) B1 = F0
      FV(2) = SQRT( B1 )
C
      IF( IFLAG .EQ. 1 ) FV(2) = SIGN ( FV(2), B(2)+B(4) )
C
      B1 = T3*( B(9) + B(9) + TRA )
      IF ( B1 .LT. F0 ) B1 = F0
      FV(3) = SQRT( B1 )
C
      IF( IFLAG .EQ. 1 ) FV(3) = SIGN ( FV(3), B(3)+B(7) )
C
      IF( IFLAG .EQ. 1 ) GO TO 110
      IF( FV(2) .NE. F0 ) FV(3) = SIGN ( FV(3), B(6)+B(8) )
110  CONTINUE
C
      AA  = F0
      FV(4) = F0
      Q   = F0
C
      IF( FV(1) .EQ. F0 ) GO TO 120
      Q = F4
      AA = AA + A1/FV(1)
120  CONTINUE
C
      IF( FV(2) .EQ. F0 ) GO TO 140
      Q = Q + F4
      AA = AA + A2/FV(2)
140  CONTINUE
C
      IF( FV(3) .EQ. F0 ) GO TO 160
      Q = Q + F4
      AA = AA + A3/FV(3)
160  CONTINUE
C
      IF( Q .NE. F0 ) FV(4) = AA/Q
C
200  CONTINUE
C
      RE-NORMALIZE QUATERNION
C

```

```

      Q = SQRT ( FV(1)**2 + FV(2)**2 + FV(3)**2 + FV(4)**2 )
      IF( Q .EQ. F0 ) GO TO 500
      Q = FVM/Q
C
      FV(1) = Q*FV(1)
      FV(2) = Q*FV(2)
      FV(3) = Q*FV(3)
      FV(4) = Q*FV(4)
C
      500 CONTINUE
C
      RETURN
      END

```

FILE: uuv22.19g/sutility/uucorvel.for

```

C-----
      SUBROUTINE CORVEL(T,MVR,VTT,RMIR,VMIR,VTTTP,VG,VS,MVS,UVS,VC,DLV,
      .               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,TTFE,TTFE
C
C      BOTH :               VTTTP,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 # 081
C                          B. HILL    - CR # 093
C-----

```

```

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

      REAL DLV(3)            , MDVT            , MRB
      REAL MRT               , MTMPV           , MVCE
      REAL MVR               , MVS             , MVSE
      REAL RMIR(3)
      REAL RB(3)             , RTPRED(3)
      REAL TMPV(3)           , URB(3)          , URT(3)
      REAL UTHP(3)           , UTMPV(3)       , UVS(3)
      REAL VC(3)             , VCE(3)
      REAL VDO(3)            , VG(3)          , VGE(3)
      REAL VMIR(3)           , VPHI(3)        , VS(3)
      REAL VSE(3)            , VTT(3)         , VTTTP(3)

```

```

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

      SAVE                  ICORV

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSCON39.DAT')
$INCLUDE('~/INCLUDE/SSCON42.DAT')
$INCLUDE('~/INCLUDE/SSCON43.DAT')
$INCLUDE('~/INCLUDE/SSCON23.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) - VD0(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 )
COSPFI = 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      = 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
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*COSPFI - (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)*COSPFI )

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

```

FILE: uuv22.19g/sutility/uucw87.for

```

subroutine cw87
integer*2 icw87
call stcw87(icw87)
icw87 = icw87 .and. #ff7ah
call ldcw87(icw87)
end

```

FILE: uuv22.19g/sutility/uufracs.for

```

C-----
C      SUBROUTINE FRACS(T,DLPC,DLYC,VCMD,VLVCM5)
C-----
C
C      SUBROUTINE NAME :      FRACS
C
C      AUTHOR(S) :          DAVID F. SMITH
C
C      FUNCTION :           GENERATE THRUSTER VALVE COMMANDS
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  TABLE
C
C      INPUTS :             T,DLPC,DLYC
C
C      OUTPUTS :            VCMD,VLVCM5
C
C      UPDATES :            B. HILL      - CR # 008
C                          B. HILL      - CR # 022
C                          T. THORNTON - CR # 026
C                          B. HILL      - CR # 030
C                          T. THORNTON - CR # 037
C                          B. HILL      - CR # 038
C                          B. HILL      - CR # 046
C                          B. HILL      - CR # 054
C                          D. SMITH     - CR # 059
C                          D. SISSOM    - CR # 069
C                          D. SMITH     - CR # 072
C                          B. HILL /    - CR # 081
C                          R. RHYNE
C                          B. HILL      - CR # 086
C                          D. SMITH     - CR # 092
C                          B. HILL      - CR # 093
C-----

```

```

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

```

```

REAL  VCMD(4)

```

```

INTEGER          VLVCM5

```

* DATA INITIALIZATION

```

$INCLUDE ('^/INCLUDE/SSCON49.DAT')
$INCLUDE ('^/INCLUDE/SSCON51.DAT')

```

C DETERMINE PITCH PLANE VALVE COMMANDS

```

IF ( DLPC.GE.DELON ) THEN
  VCMD(2) = 0.0
  VCMD(4) = DTFRU*DLPC
ELSEIF ( DLPC.LT.DELON .AND. DLPC.GT.-DELON ) THEN
  VCMD(2) = 0.0
  VCMD(4) = 0.0
ELSEIF ( DLPC.LE.-DELON ) THEN
  VCMD(2) = DTFRU*ABS(DLPC)
  VCMD(4) = 0.0

```

```

ENDIF
C   DETERMINE YAW PLANE VALVE COMMANDS
      IF ( DLYC.GE.DELON ) THEN
        VCMD(1) = 0.0
        VCMD(3) = DTFRU*DLYC
      ELSEIF ( DLYC.LT.DELCN .AND. DLYC.GT.-DELON ) THEN
        VCMD(1) = 0.0
        VCMD(3) = 0.0
      ELSEIF ( DLYC.LE.-DELON ) THEN
        VCMD(1) = DTFRU*ABS(DLYC)
        VCMD(3) = 0.0
      ENDIF
C   UPDATE TOTAL NUMBER OF CYCLES THAT THE VALVES ARE ON
      DO 10 I = 1, 4
        IF ( VCMD(I).NE.0.0 ) THEN
          VLVCM5 = VLVCM5 + 1
        ENDIF
10  CONTINUE
      RETURN
      END

```

FILE: uuv22.19g/sutlity/uufrcthr.for

```

C-----
C   SUBROUTINE FRCTHR(T,CG,MACH,QA,VCMD,VCMDL,DTOFF,TFTAB,IFTAB,
C   .               TOSEED,TBRK,TMF,THF,LENF,FRCX,FRCY,FRCZ,MRCX,
C   .               MRCY,MRCZ,MDOTF,ATHRF,KN,KM,FOFF1,FOFF2)
C-----
C
C   SUBROUTINE NAME :      FRCTHR
C
C   AUTHOR(S) .          K. S. BOGAN, D. C. FOREMAN
C
C   FUNCTION :            COMPUTES FORCES AND MOMENTS RESULTING FROM
C   .                     THE FORWARD REACTION CONTROL THRUSTERS
C
C   CALLED FROM :         FORTRAN MAIN
C
C   SUBROUTINES CALLED :  NONE
C
C   INPUTS :              T,CG,MACH,QA,VCMD,VCMDL,DTOFF,TFTAB
C
C   OUTPUTS :             FRCX,FRCY,FRCZ,MRCX,MRCY,MRCZ,MDOTF,ATHRF,
C   .                     KN,KM,FOFF1,FOFF2
C
C   BOTH :               IFTAB,TOSEED,TBRK,TMF,THF,LENF
C
C   UPDATES :             B. HILL      - CR # 008
C   .                     B. HILL      - CR # 022
C   .                     T. THORNTON - CR # 026
C   .                     B. HILL      - CR # 030
C   .                     B. HILL      - CR # 038
C   .                     B. HILL      - CR # 046
C   .                     D. SMITH     - CR # 059
C   .                     D. SISSOM    - CR # 061
C   .                     D. SISSOM    - CR # 069
C   .                     D. SMITH     - CR # 072
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   .                     D. SMITH     - CR # 092
C   .                     B. HILL      - CR # 093
C-----
C
C   IMPLICIT REAL          (A-H)
C   IMPLICIT REAL          (O-Z)

```

```

REAL  FRCDIR(3,4)    , FRCLOC(3,4)    , FRCMA(9,4)
REAL  CG(3)          , F0(3)          , VCMDL(4)
REAL  F(3)           , XMOM(3)        , M(3)
REAL  MRCX           , MRCY           , MRCZ
REAL  MACH           , MCHLIM         , KN
REAL  KM             , LD             , MDOTF
REAL  FISP           , TMF(8,4)       , THF(8,4)
REAL  ATHRF(4)       , KNFAC          , DTOFF(4)
REAL  KMFAC          , FOFF1(4)       , FOFF2(4)
REAL  VCMD(4)

```

```

INTEGER      INDX(4)      , LENF(4)
INTEGER*4    TOSEED
INTEGER      VCOD(4)

```

C LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

```

SAVE          IFRCTH

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSFRCTHR.DAT')
$INCLUDE('~/INCLUDE/SSCON01.DAT')
$INCLUDE('~/INCLUDE/SSCON17.DAT')
$INCLUDE('~/INCLUDE/SSCON22.DAT')
$INCLUDE('~/INCLUDE/SSCON23.DAT')
$INCLUDE('~/INCLUDE/SSCON30.DAT')
$INCLUDE('~/INCLUDE/SSCON32.DAT')
$INCLUDE('~/INCLUDE/SSCON33.DAT')
$INCLUDE('~/INCLUDE/SSCON49.DAT')
$INCLUDE('~/INCLUDE/SSCON51.DAT')
$INCLUDE('~/INCLUDE/SSCON52.DAT')

```

```

DATA IFRCTH / 1 /

```

```

IF ( IFRCTH.EQ.1 ) THEN

```

```

    IFRCTH = 0

```

```

    IF ( T .LT. TFRCS+EPSL ) THEN

```

```

C      FRACS MISALIGNMENT DIRECTIONS
C      FOFF1 = CONE ANGLE OFF NORMAL
C      FOFF2 = POLAR ANGLE

```

```

    CALL NORM(SDTHR,0.0,TOSEED,FOFF1(1))
    CALL NORM(SDTHR,0.0,TOSEED,FOFF1(2))
    CALL NORM(SDTHR,0.0,TOSEED,FOFF1(3))
    CALL NORM(SDTHR,0.0,TOSEED,FOFF1(4))

```

```

    FOFF2(1) = 2.0*PI*RAN0(TOSEED)
    FOFF2(2) = 2.0*PI*RAN0(TOSEED)
    FOFF2(3) = 2.0*PI*RAN0(TOSEED)
    FOFF2(4) = 2.0*PI*RAN0(TOSEED)

```

```

    ENDIF

```

C FRACS THRUSTER MISALIGNMENT MATRIX

```

DO 10 I = 1 , 4
    CFOFF1 = COS(FOFF1(I))
    SFOFF1 = SIN(FOFF1(I))
    CFOFF2 = COS(FOFF2(I))
    SFOFF2 = SIN(FOFF2(I))
    FRCMA(1,I) = CFOFF1
    FRCMA(2,I) = SFOFF1*CFOFF2
    FRCMA(3,I) = SFOFF1*SFOFF2
    FRCMA(4,I) = SFOFF1*SFOFF2
    FRCMA(5,I) = CFOFF1
    FRCMA(6,I) = SFOFF1*CFOFF2
    FRCMA(7,I) = SFOFF1*CFOFF2
    FRCMA(8,I) = SFOFF1*SFOFF2
    FRCMA(9,I) = CFOFF1

```

```

10  CONTINUE

```

```

    ENDIF

```

C REINITIALIZE FORCES AND MOMENTS EACH PASS THROUGH

```

FRCX = 0.0
FRCY = 0.0
FRCZ = 0.0

```

```

MRCX  = 0.0
MRCY  = 0.0
MRCZ  = 0.0
MDOTF = 0.0

IF ( T .LT. TSTG2) THEN

C      CALCULATE JET INTERACTION AMPLIFICATION FACTORS

      IF ( T.LE.TSTG1 ) THEN
        LD      = ( XJET - XNOZ1 )/DJET
      ELSE
        LD      = ( XJET - XNOZ2 )/DJET
      ENDIF

      CT      = THJET/( QA*SJET )

C      FORCE COEFFICIENT

      IF ( MACH.LE.MCHLIM ) THEN
        KN      = 0.6118 + (0.1358*(1.-0.485*SQRT(LD))/SQRT(CT))
                  + 0.0946*MACH + 0.004317/LD
      ELSE
        KN      = 1.0 + EXP(1.1 - 0.2116*(ALOG(CT)+8.5)**1.4)
      ENDIF

C      MOMENT COEFFICIENT

      KM      = 0.5582 - 0.1884/SQRT(CT) - 1.9659/LD

      IF ( IFTAB.EQ.1 ) THEN
* The IFTAB assignment was moved to the partition with FRACS
* IFTAB = 0

C      LOOP ON EACH VALVE

      DO 20 I = 1 , 4

C          INITIALIZE TEMPORARY TABLE LOOKUP POINTER TO ZERO

          ITMP  = 0

C          VALVE COMMAND IS OFF

          IF ( VCMD(I).EQ.0.0 ) THEN

C              VALVE WAS SHUT DURING CYCLE JUST ENDED

              IF ( LENF(I).EQ.0 ) THEN

C                  VALVE WAS OPEN DURING CYCLE JUST ENDED

                  ELSEIF ( LENF(I).GT.0 ) THEN

C                      VALVE IS CURRENTLY SHUT

                      IF ( TFTAB.GE.TMF(LENF(I),I) ) THEN
                        LENF(I) = 0
                      ENDIF

C                      VALVE IS CURRENTLY OPEN

                      ELSEIF ( TFTAB.LT.TMF(LENF(I),I) ) THEN

C                          VALVE IS RAMPING SHUT

                          IF ( TFTAB.GE.TMF(LENF(I)-1,I) ) THEN
                            CALL TABLE(TMF(1,I),THF(1,I),TFTAB,TMP1,
                                           LENF(I),ITMP)
                            TMF(1,I) = TFTAB
                            TMF(2,I) = TMF(LENF(I),I)
                            THF(1,I) = TMP1
                            THF(2,I) = 0.0
                            LENF(I)  = 2
                          ENDIF

C                          VALVE IS WIDE OPEN

                          ELSEIF ( TFTAB.LT.TMF(LENF(I)-1,I) ) THEN
                            TMF(1,I) = TFTAB
                            TMF(2,I) = TMF(LENF(I)-1,I)
                          ENDIF

```

```

        TMF(3,I) = TMF(LENF(I),I)
        THF(1,I) = 1.0
        THF(2,I) = 1.0
        THF(3,I) = 0.0
        LENF(I) = 3
    ENDIF

ENDIF

ENDIF

C      VALVE IS COMMANDED OPEN

      ELSEIF ( VCMD(I).GT.0.0 ) THEN
* FTN286 X415 OPTIMIZE(3)
99999 CONTINUE
C      VALVE WAS SHUT DURING CYCLE JUST ENDED

      IF ( LENF(I).EQ.0 ) THEN
        VCMD(I) = 0.001*ANINT(VCMD(I)/0.001)
        TMF(1,I) = TFTAB
        TMF(2,I) = TMF(1,I) + TLAGFR
        TMF(3,I) = TMF(2,I) + TRUPFR
        TMF(4,I) = TMF(2,I) + VCMD(I)
        TMF(5,I) = TMF(4,I) + TRDNFR
        THF(1,I) = 0.0
        THF(2,I) = 0.0
        THF(3,I) = 1.0
        THF(4,I) = 1.0
        THF(5,I) = 0.0
        LENF(I) = 5
      C      VALVE WAS OPEN DURING CYCLE JUST ENDED

      ELSEIF ( LENF(I).GT.0 ) THEN

      C      VALVE IS CURRENTLY SHUT

        IF ( TFTAB.GE.TMF(LENF(I),I) ) THEN
          VCMD(I) = 0.001*ANINT(VCMD(I)/0.001)
          TMF(1,I) = TFTAB
          TMF(2,I) = TMF(1,I) + TLAGFR
          TMF(3,I) = TMF(2,I) + TRUPFR
          TMF(4,I) = TMF(2,I) + VCMD(I)
          TMF(5,I) = TMF(4,I) + TRDNFR
          THF(1,I) = 0.0
          THF(2,I) = 0.0
          THF(3,I) = 1.0
          THF(4,I) = 1.0
          THF(5,I) = 0.0
          LENF(I) = 5
        C      VALVE IS CURRENTLY OPEN

        ELSEIF ( TFTAB.LT.TMF(LENF(I),I) ) THEN

      C      VALVE WILL BE SHUT AT REISSUE TIME

        IF ( TFTAB+TLAGFR.GE.TMF(LENF(I),I) ) THEN

      C      VALVE IS RAMPING SHUT NOW

        IF ( TFTAB.GT.TMF(LENF(I)-1,I) ) THEN
          TMP1 = TMF(LENF(I),I) - TFTAB
          VLVRES = 0.5*TMP1**2/TRDNFR
          VCMD(I) = VCMD(I) - VLVRES
          CALL TABLE(TMF(1,I),THF(1,I),TFTAB,TMP1,
                     LENF(I),ITMP)

      C      ISSUE A NEW COMMAND IF THRESHOLD IS
      C      REACHED

        IF ( VCMD(I).GE.DELON*DTFRU ) THEN
          VCMD(I) = AMIN1 ( DTFRU , VCMD(I) )
          VCMD(I) = 0.001*ANINT(VCMD(I)/
                                0.001)
          TMF(1,I) = TFTAB
          TMF(2,I) = TMF(LENF(I),I)
          TMF(3,I) = TMF(1,I) + TLAGFR
          TMF(4,I) = TMF(3,I) + TRUPFR
          TMF(5,I) = TMF(3,I) + VCMD(I)

```

```

      TMF(6,I) = TMF(5,I) + TRDNFR
      THF(1,I) = TMP1
      THF(2,I) = 0.0
      THF(3,I) = 0.0
      THF(4,I) = 1.0
      THF(5,I) = 1.0
      THF(6,I) = 0.0
      LENF(I) = 6

C      NO NEW COMMAND IS ISSUED IF THRESHOLD IS
C      NOT REACHED

      ELSEIF ( VCMD(I).LT.DELON*DTRU ) THEN
        TMF(1,I) = TFTAB
        TMF(2,I) = TMF(LENF(I),I)
        THF(1,I) = TMP1
        THF(2,I) = 0.0
        LENF(I) = 2
      ENDIF

C      VALVE IS WIDE OPEN NOW

      ELSEIF ( TFTAB.LE.TMF(LENF(I)-1,I) ) THEN
        VLVRES = TMF(LENF(I)-1,I) - TFTAB
        + 0.5*TRDNFR
        VCMD(I) = VCMD(I) - VLVRES

C      ISSUE A NEW COMMAND IF THRESHOLD IS
C      REACHED

      IF ( VCMD(I).GE.DELON*DTRU ) THEN
        VCMD(I) = AMIN1 ( DTRU , VCMD(I) )
        VCMD(I) = 0.001*ANINT(VCMD(I)/
          0.001)
        TMF(1,I) = TFTAB
        TMF(2,I) = TMF(LENF(I)-1,I)
        TMF(3,I) = TMF(LENF(I),I)
        TMF(4,I) = TMF(1,I) + TLAGFR
        TMF(5,I) = TMF(4,I) + TRUPFR
        TMF(6,I) = TMF(4,I) + VCMD(I)
        TMF(7,I) = TMF(6,I) + TRDNFR
        THF(1,I) = 1.0
        THF(2,I) = 1.0
        THF(3,I) = 0.0
        THF(4,I) = 0.0
        THF(5,I) = 1.0
        THF(6,I) = 1.0
        THF(7,I) = 0.0
        LENF(I) = 7

C      NO NEW COMMAND IS ISSUED IF THRESHOLD IS
C      NOT REACHED

      ELSEIF ( VCMD(I).LT.DELON*DTRU ) THEN
        TMF(1,I) = TFTAB
        TMF(2,I) = TMF(LENF(I)-1,I)
        TMF(3,I) = TMF(LENF(I),I)
        THF(1,I) = 1.0
        THF(2,I) = 1.0
        THF(3,I) = 0.0
        LENF(I) = 3
      ENDIF

      ENDIF

C      VALVE WILL BE OPEN AT REISSUE TIME

      ELSEIF ( TFTAB+TLAGFR.LT.TMF(LENF(I),I) ) THEN

C      VALVE WILL BE RAMPING SHUT AT REISSUE TIME

      IF ( TFTAB+TLAGFR.GT.TMF(LENF(I)-1,I) ) THEN
        VLVRES = TMF(LENF(I)-1,I) - TFTAB
        + 0.5*TRDNFR
        VCMD(I) = VCMD(I) - VLVRES

C      ISSUE COMMAND ONLY IF DESIRED DURATION
C      EXCEEDS RAMP INTERVAL

      IF ( VCMD(I).GE.TRUPFR ) THEN
        CALL TABLE(TMFI(1,I),THFI(1,I),TFTAB,

```

```

      TMPL, LENF(I), ITMP)
VCMD(I) = AMIN1 ( DTFRU , VCMD(I) )
VCMD(I) = 0.001*ANINT(VCMD(I) /
0.001)
TMF(1,I) = TFTAB
TMF(2,I) = TMF(LENF(I)-1,I)
TMF(3,I) = TMF(1,I) + TLAGFR
TMF(4,I) = TMF(3,I) + TMF(3,I) -
      TMF(2,I)
TMF(5,I) = TMF(3,I) + VCMD(I)
TMF(6,I) = TMF(5,I) + TRDNFR
THF(1,I) = 1.0
THF(2,I) = 1.0
THF(3,I) = TMPL
THF(4,I) = 1.0
THF(5,I) = 1.0
THF(6,I) = 0.0
LENF(I) = 6

C      NO NEW COMMAND IS ISSUED IF THRESHOLD IS
C      NOT REACHED

      ELSEIF ( VCMD(I).LT.TRUPFR ) THEN
        TMF(1,I) = TFTAB
        TMF(2,I) = TMF(LENF(I)-1,I)
        TMF(3,I) = TMF(LENF(I),I)
        THF(1,I) = 1.0
        THF(2,I) = 1.0
        THF(3,I) = 0.0
        LENF(I) = 3
      ENDIF

C      VALVE WILL BE WIDE OPEN AT REISSUE TIME

      ELSEIF (TFTAB+TLAGFR.LE.TMF(LENF(I)-1,I)) THEN

C      COMPARE REMAINING NORMALIZED IMPULSE WITH
C      REQUESTED NORMALIZED IMPULSE TO SEE IF NEW
C      COMMAND SHOULD BE ISSUED

      VLVRES = TMF(LENF(I)-1,I) - TFTAB
      + 0.5*TRDNFR
      VCMD(I) = VCMD(I) - VLVRES

C      REVISE VALVE SHUT TIME IF ADDITIONAL
C      IMPULSE IS REQUIRED

      IF ( VCMD(I).GT.0.0 ) THEN

* FTN286 X415 OPTIMIZE(3)
99998 CONTINUE

      VCMD(I) = AMIN1 ( DTFRU , VCMD(I) )
      VCMD(I) = 0.001*ANINT(VCMD(I) /
0.001)
      TMF(1,I) = TFTAB
      TMF(2,I) = TMF(LENF(I)-1,I) + VCMD(I)
      TMF(3,I) = TMF(LENF(I),I) + VCMD(I)
      THF(1,I) = 1.0
      THF(2,I) = 1.0
      THF(3,I) = 0.0
      LENF(I) = 3

C      NO NEW COMMAND IS ISSUED IF ADDITIONAL
C      IMPULSE IS NOT REQUIRED

      ELSEIF ( VCMD(I).LE.0.0 ) THEN
        TMF(1,I) = TFTAB
        TMF(2,I) = TMF(LENF(I)-1,I)
        TMF(3,I) = TMF(LENF(I),I)
        THF(1,I) = 1.0
        THF(2,I) = 1.0
        THF(3,I) = 0.0
        LENF(I) = 3
      ENDIF

      ENDIF

      ENDIF

      ENDIF

      ENDIF

```

```

                ENDIF
20      CONTINUE
      ENDIF
C      DETERMINE TABLE LOOKUP REFERENCE TIME
      TREF = T
C      CALCULATE CURRENT THRUST LEVELS FOR EACH VALVE
      DO 40 I = 1 , 4
C          COMPUTE INSTANTANEOUS NORMALIZED THRUST LEVEL VIA TABLE
C          LOOKUP IF FRACS CYCLE IS SCHEDULED FOR THIS THRUSTER.  ALSO
C          EXTRAPOLATE TIME OF NEXT FRACS TABLE LOOKUP INDEX TRANSITION.
          IF ( LENF(I).GT.0 ) THEN
              CALL TABLE(TMF(1,I),THF(1,I),TREF,ATHRF(I),LENF(I),
                          INDX(I),
          ELSE
              ATHRF(I) = 0.0
              INDX(I) = 0
          ENDIF
C          CALCULATE THE FORCES AND MOMENTS PRODUCED BY THE FRACS
C          THRUSTERS :
C          F(I) IS THE FORCE ALONG THE Ith AXIS , ADJUSTED
C          FOR MISALIGNMENT .
C          XMOM(I) IS THE EFFECTIVE MOMENT ARM.
C          THE MOMENT GENERATED IS ( F x XMOM ).
          DO 30 J = 1 , 3
              F0(J) = FRCDIR(J,I)*KN*KNFAC*THJET*ATHRF(I)
              XMOM(J) = CG(J) - FRCLOC(J,I)
30          CONTINUE
C          THRUSTER MISALIGNMENT EFFECTS
          F(1) = F0(1)*FRCMA(1,I) + F0(2)*FRCMA(4,I) +
              F0(3)*FRCMA(7,I)
          F(2) = F0(1)*FRCMA(2,I) + F0(2)*FRCMA(5,I) +
              F0(3)*FRCMA(8,I)
          F(3) = F0(1)*FRCMA(3,I) + F0(2)*FRCMA(6,I) +
              F0(3)*FRCMA(9,I)
          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)
C          SUM FORCES AND MOMENTS OF ALL THRUSTERS
          FRCX = FRCX + F(1)
          FRCY = FRCY + F(2)
          FRCZ = FRCZ + F(3)
          MRCX = MRCX + M(1)
          MRCY = MRCY + M(2)
          MRCZ = MRCZ + M(3)
          IF ( I.EQ.1 .OR. I.EQ.3 ) THEN
              MRCY = MRCY + FRCDIR(3,I)*THJET*KN*KNFAC*DJET*
                  ATHRF(I)
          ELSE
              MRCZ = MRCZ - FRCDIR(2,I)*THJET*KN*KNFAC*DJET*
                  ATHRF(I)
          ENDIF
          MDOTF = MDOTF + THJET*ATHRF(I)/FISP
40      CONTINUE
      ENDIF
      RETURN
      END

```

FILE: uuv22.19g/utility/uufvdot.for

```

C-----
C      SUBROUTINE FVDOT ( W, WD, F, FD )
C-----
C      SUBROUTINE NAME :      FVDOT
C      AUTHOR(S) :          W. E. EXELY
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      SUBROUTINES CALLED :  NONE
C      INPUTS :             W,WD,F
C      OUTPUTS :            FD
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

```

FILE: uu22.19g/sutility/uugyro.for

```

C-----
C      SUBROUTINE GYRO(T,P,Q,R,CIM,GYSEED,QFRACG,PULSEG)
C-----
C      SUBROUTINE NAME :      GYRO
C      AUTHOR(S) :          A. P. BUKLEY, M. K. DOUBLEDAY
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      SUBROUTINES CALLED :  NORM,BRTAVG,RESP2P
C      INPUTS :             T,P,Q,R,CIM
C      OUTPUTS :            NONE
C      BOTH :               GYSEED,QFRACG,PULSEG
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-----

```

```

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

      REAL  CIM(9)        , CIMO(9)        , DCG(3)
      REAL  DTHET(3)      , PULSEG(3)      , PQRAVG(3)
      REAL  QFRACG(3)     , SF1G(3)        , SF2G(3)
      REAL  SFEG(3)       , WBI0(3)        ,
      REAL  WBI1(3)       , WBI2(3)        , WBO0(3)
      REAL  WBO1(3)       , WBO2(3)        , WDRG(3)

      INTEGER*4          GYSEED

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

      SAVE              IGYRO

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSGYRO.DAT')
$INCLUDE('~/INCLUDE/SSCON16.DAT')
$INCLUDE('~/INCLUDE/SSCON21.DAT')
$INCLUDE('~/INCLUDE/SSCON53.DAT')

      DATA IGYRO / 1 /

      IF (IGYRO .EQ. 1) THEN

        IGYRO = 0

C      INITIALIZE GYRO PARAMETERS

        IF (T .EQ. 0.0) THEN
          DRSIGG = DRSGGI/(60.0*SQRT(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) = 0.0
            WBI1(I) = 0.0
            WBO2(I) = 0.0
            WBO1(I) = 0.0
10        CONTINUE
          ENDIF

C      COMPUTE SECOND ORDER RESPONSE DIFFERENCE EQUATION COEFFICIENTS

          IF ( IGRTP.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 - TOGYRO
TOGYRO = T

C      DETERMINE AVERAGE BODY RATE OVER LAST INTERVAL

IF ( DTDEL.NE.0.0 ) 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.0.0 ) THEN
    CALL NORM(DRSIGG,DRMENG,GYSEED,DRG)
ENDIF

WDRG(I) = DRG + DCG(I)

C      COMPUTE INPUT TO GYRO RESPONSE MODEL

WBIO(I) = SFEG(I) + WDRG(I)

C      SECOND ORDER RESPONSE MODEL

IF ( IGRTYP.EQ.2 ) THEN
    WBOO(I) = ( CWBIO*WBIO(I) + CWBI1*WBII(I)
    + CWBI2*WBII(I) - CWBO1*WBOI(I)
    - CWBO2*WBOI(I) ) / CWBIO
    WBI2(I) = WBII(I)
    WBI1(I) = WBIO(I)
    WBO2(I) = WBOI(I)
    WBO1(I) = WBOO(I)
ENDIF

C      INSTANTANEOUS RESPONSE MODEL

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

C      COMPUTE DELTA THETA

DTHET(I) = DTDEL * WBOO(I)

IF ( SPPG.GT.0.0 ) THEN

```

```

C      UNQUANTIZED OUTPUT IN COUNTS
      QFRACG(I) = QFRACG(I) - PULSEG(I) + DTHET(1)/SPPG
C      QUANTIZED OUTPUT IN COUNTS
      PULSEG(I) = AINT(QFRACG(I))
      ELSE
      PULSEG(I) = DTHET(I)
      ENDIF
30 CONTINUE
      RETURN
      END

```

FILE: uuv22.19g/sutility/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      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 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

```

FILE: uuv22.19g/sutility/uuintegi.for

```

C-----
C      SUBROUTINE INTEGI ( X , XDOT , T , I )
C-----

```

```

C
C SUBROUTINE NAME :      INTEG
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-----

```

```

COMMON/STORAG/      XINT,      TINT,      XDOTL      XDOTL
REAL  XINT(50),      TINT(50),      XDOTL(50)
REAL  X,              T,              XDOT

```

```

XINT(I) = X
XDOTL(I) = XDOT
TINT(I) = T

```

```

RETURN
END

```

FILE: uuv22.19g/sutility/uum3x3i.for

```

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-----

```

```

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

REAL  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.0.0 ) 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) = 0.0
  B(2,1) = 0.0
  B(3,1) = 0.0

```

```

      B(1,2) = 0.0
      B(2,2) = 0.0
      B(3,2) = 0.0
      B(1,3) = 0.0
      B(2,3) = 0.0
      B(3,3) = 0.0
END IF

```

```

RETURN
END

```

FILE: uuv22.19g/sutlity/uumcguid.for

```

C-----
SUBROUTINE 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-----
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-----

```

```

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

```

```

REAL TI2M(9)      , VG(3)      , URREL(3)
REAL MASS      , MAGR      , MAGV
REAL VGM(3)      , ADISTT(4,3) , OMEGA0(3)
REAL VGP(3)      , VGPM(3)    , ACQRNG(4,4)
REAL RATE(6)     , TRGSIG(4)
INTEGER ISEQ(4)    , FLIP      , SEKTYP
INTEGER BCKGRD

```

C LOCAL DATA USED FOR CONSTANTS AND INITIALIZATION FLAG

```

SAVE          IMGUID

```

```

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSMCGUID.DAT')
$INCLUDE('~/INCLUDE/SSCON46.DAT')
$INCLUDE('~/INCLUDE/SSCON48.DAT')
$INCLUDE('~/INCLUDE/SSCON50.DAT')
$INCLUDE('~/INCLUDE/SSCON55.DAT')
$INCLUDE('~/INCLUDE/SSCON60.DAT')
$INCLUDE('~/INCLUDE/SSCON61.DAT')
$INCLUDE('~/INCLUDE/SSCON62.DAT')
$INCLUDE('~/INCLUDE/SSCON01.DAT')
$INCLUDE('~/INCLUDE/SSCON04.DAT')
$INCLUDE('~/INCLUDE/SSCON05.DAT')
$INCLUDE('~/INCLUDE/SSCON09.DAT')
$INCLUDE('~/INCLUDE/SSCON12.DAT')
$INCLUDE('~/INCLUDE/SSCON13.DAT')

```

```
$INCLUDE ('^/INCLUDE/SSCON17.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 = RRGACQ
  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. .. RERR.GT.PI/4. ) THEN
        ROLLER = RERR - PI/2.
        IVCS = 1
      ELSE
        ROLLER = RERR + PI/2.
        IVCS = 3
      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

        CALL OUTMES(0801,T,0.0)
        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
        ENDIF
      ENDIF

```

```

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
          CALL OUTMES(0802,T,0.0)
        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
      CALL OUTMES(0803,T,0.0)
    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 - RGAQ )/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

```

```

ENDIF
C    COMPUTE TIME OF NEXT CALL
      TMGUID = T + DTMCU - EPSL
      RETURN
      END

```

FILE: uuv22.19g/sutility/uumisslr.for

```

C-----
      SUBROUTINE MISSLR(T,QUAT,CIM,P,Q,R,IXX,IYY,IZZ,MASS,FXA,FXT,
      .                FRCX,FXACS,FXVCS,
      .                MXA,MXT,MRCX,MXACS,MXVCS,
      .                MYA,MYT,MRCY,MYACS,MYVCS,MZA,MZT,MRCZ,MZACS,
      .                MZVCS,X,Y,Z,PD,QD,RD,
      .                MX,MY,MZ,
      .                QUATD)
C-----
C
C    SUBROUTINE NAME :      MISSLR
C
C    AUTHOR(S) :          D. C. FOREMAN, A. P. BUKLEY
C
C    FUNCTION :           COMPUTES THE ROTATIONAL MISSILE ACCELERATIONS
C
C    CALLED FROM :        FORTRAN MAIN
C
C    SUBROUTINES CALLED :  FVDOT
C
C    INPUTS :             T,QUAT,CIM,P,Q,R,IXX,IYY,IZZ,MASS,FXA,
C                          FXT,FRCX,FXACS,FXVCS,
C                          MXA,MXT,
C                          MRCX,MXACS,MXVCS,MYA,MYT,MRCY,MYACS,MYVCS,
C                          MZA,MZT,MRCZ,MZACS,MZVCS,X,Y,Z,
C
C    OUTPUTS :            PD,QD,RD,MX,MY,MZ,
C                          QUATD
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    - CR # 087
C                          R. RHYNE    - CR # 087
C                          B. HILL     - CR # 093
C-----

```

```

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

      REAL  CJM(9)        , CMI(9)        , GB(3)
      REAL  GR(3)         , IXX           , IYY
      REAL  IZZ           , MASS          , MGR
      REAL  MRCX          , MRCY          , MRCZ
      REAL  MX            , MXA           , MXACS
      REAL  MXT           , MXVCS        , MXYZ
      REAL  MXYZDD        , MY           , MYA
      REAL  MYACS         , MYT           , MYVCS
      REAL  MZ            , MZA          , MZACS
      REAL  MZT           , MZVCS        , PQR(3)
      REAL  QUAT(4)       , QUATD(4)     , UXYZ(3)
      REAL  UXYZDD(3)     , XYZLCH(3)

```

C LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

SAVE IMISL

```
* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSMISSIL.DAT')
$INCLUDE('~/INCLUDE/SSCON28.DAT')
$INCLUDE('~/INCLUDE/SSCON39.DAT')
$INCLUDE('~/INCLUDE/SSCON63.DAT')
```

```
DATA IMISL / 1 /
DATA NCLEAR / 0 /
```

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)

IF (T .EQ. 0.0) THEN
  XYZLCH(1) = XLNCH*CMI(1) + RADE
  XYZLCH(2) = XLNCH*CMI(2)
  XYZLCH(3) = XLNCH*CMI(3)
ENDIF
```

C DETERMINE initial GRAVITY VECTOR, just for seeing if we are still
* on the ground later

```
MXYZ = SQRT ( X**2 + Y**2 + Z**2 )
MGR = GMU / MXYZ**2
```

IF (MXYZ.GT.0.0) THEN

* FTN286 X415 OPTIMIZE(3)

99999 CONTINUE

```
UXYZ(1) = X / MXYZ
UXYZ(2) = Y / MXYZ
UXYZ(3) = Z / MXYZ
ELSE
  UXYZ(1) = 0.0
  UXYZ(2) = 0.0
  UXYZ(3) = 0.0
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)
```

ENDIF

C CALCULATE TOTAL X FORCE, just to see if still on ground later

FX = FXT + FXA + FRCX + FXACS + FXVCS

C CALCULATE TOTAL MOMENTS

```
MX = MXA + MXT + MRCX + MXACS + MXVCS
MY = MYA + MYT + MRCY + MYACS + MYVCS
MZ = MZA + MZT + MRCZ + 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)
```

```

C      MISSILE STILL ON GROUND
      ELSE IF ( FX/MASS.LE.ABS(GB(1)) ) THEN
        PD      = 0.0
        QD      = 0.0
        RD      = 0.0

C      MISSILE OFF GROUND BUT NOT CLEAR OF THE LAUNCHER
      ELSE IF ( X.LE.XYZLCH(1) .AND. Y.LE.XYZLCH(2) .AND.
        Z.LE.XYZLCH(3) ) THEN
        PD      = MX/IXX + Q*R*((IYY-IZZ)/IXX)
        QD      = 0.0
        RD      = 0.0

C      MISSILE JUST NOW CLEARING LAUNCHER
      ELSE
        NCLEAR = 1
        CALL OUTMES(0901,T,0.0)
        PD      = MX/IXX + Q*R*((IYY-IZZ)/IXX)
        QD      = MY/IYY + R*P*((IZZ-IXX)/IYY)
        RD      = IZ/IZZ + P*Q*((IXX-IYY)/IZZ)
      ENDTF

C      COMPUTE QUATERNION DERIVATIVES

      PQR(1) = P
      PQR(2) = Q
      PQR(3) = R

      TMP1    = 0.0
      CALL FVDOT(PQR,TMP1,QUAT,QUATD)

      RETURN
      END

```

FILE: uuv22.19g/sutility/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 REAL (A-H)
C      IMPLICIT REAL (C ?)
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

```

RETURN
END

FILE: uuv22.19g/sutility/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 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

```

FILE: uuv22.19g/sutility/uuncu.for

```

C-----
C      SUBROUTINE NCU(DLP,DLY,CMMD,DLPD,DLYD)
C-----
C
C      SUBROUTINE NAME :      NCU
C
C      AUTHOR(S) :          T. THORNTON
C
C      FUNCTION :           MODELS THE RESPONSE OF THE NOZZLE
C                           CONTROL UNIT
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             DLP,DLY,CMMD
C
C      OUTPUTS :            DLPD,DLYD
C
C      UPDATES :            D. SMITH      - CR # 040
C                           D. SMITH      - CR # 059
C                           B. HILL /     - CR # 081
C                           R. RHYNE
C                           B. HILL      - CR # 093
C-----
C
C      IMPLICIT REAL (A-H)

```

```

      IMPLICIT REAL (O-Z)

      REAL CMMD(2)      , KNCU

      * DATA INITIALIZATION
      $INCLUDE('~/INCLUDE/SSCON64.DAT')

      C      PITCH NOZZLE GIMBAL RESPONSE

      DLPD  = (CMMD(1) - KNCU*DLP)*OMEGAT

      C      YAW NOZZLE GIMBAL RESPONSE

      DLYD  = (CMMD(2) - KNCU*DLY)*OMEGAT

      C      LIMIT GIMBAL RATES

      TOTRAT = SQRT ( DLPD**2 + DLYD**2 )
      IF ( TOTRAT.GT.RMAX ) THEN
        DLPD = DLPD * RMAX / TOTRAT
        DLYD = DLYD * RMAX / TOTRAT
      END IF

      RETURN
      END

```

FILE: uuv22.19g/sutility/uunorm.for

```

C-----
C      SUBROUTINE NORM(SD,MN,ISEED,RDN)
C-----
C      SUBROUTINE NAME :      NORM
C      AUTHOR(S) :          D. F. SMITH
C      FUNCTION :           GENERATES NORMALLY DISTRIBUTED RANDOM
C                           NUMBERS USING THE BOX-MULLER TRANSFORMATION
C      CALLED FROM :        UTILITY SUBROUTINE
C      SUBROUTINES CALLED :  RANO
C      INPUTS :              SD,MN
C      OUTPUTS :             RDN
C      BOTH :                ISEED
C      UPDATES :             D. SMITH   - CR # 082
C                           R. RHYNE   - CR # 087
C-----

```

```

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

      REAL MN
      INTEGER*4      ISEED

      SAVE GSET      , ISET
      DATA GSET/0., ISET/0/

      DATA ONE      / 1.0 /
      DATA TWO      / 2.0 /

      C      IF A SPARE RANDOM NUMBER IS NOT AVAILABLE FROM THE PREVIOUS PASS
      C      GENERATE TWO NEW ONES

      IF ( ISET.EQ.0 ) THEN

      C      GET TWO UNIFORM RANDOM NUMBERS WITHIN THE SQUARE EXTENDING
      C      FROM -1 TO 1 IN EACH DIRECTION

      V1      = TWO*RAN0( ISEED ) - ONE
      V2      = TWO*RAN0( ISEED ) - ONE

      C      SEE IF THEY ARE WITHIN THE UNIT CIRCLE . IF NOT , TRY AGAIN .

```

```

      R      = V1*V1 + V2*V2
      IF ( R.GE.ONE ) GO TO 1

C      PERFORM BOX-MULLER TRANSFORMATION TO GENERATE TWO GAUSSIAN
C      RANDOM NUMBERS . RETURN ONE AND SAVE THE OTHER FOR THE NEXT
C      PASS .

      FAC    = SQRT ( -TWO*ALOG(R)/R )
      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

```

FILE: uu22.19g/sutility/uuoutmes.for

```

      SUBROUTINE OUTMES(N,T,ARG)
      INTEGER N
      REAL T,ARG
$INCLUDE('PFP:INCLUDE/TARGET.FOR')
      CHARACTER*80 MESSAGE

C
C      PROGRAM: MAIN (0101...0200)
C
      IF ( N.EQ.0101 ) THEN
        WRITE(MESSAGE,0101) T
0101    FORMAT(1X,E16.9,' 1ST STAGE SEPARATION')
        GO TO 99999
      END IF

      IF ( N.EQ.0102 ) THEN
        WRITE(MESSAGE,0102) T
0102    FORMAT(1X,E16.9,' 2ND STAGE SEPARATION')
        GO TO 99999
      END IF

      IF ( N.EQ.0103 ) THEN
        WRITE(MESSAGE,0103) T
0103    FORMAT(1X,E16.9,' DROP NOSE FAIRING AND BOOST ADAPTER')
        GO TO 99999
      END IF

      IF ( N.EQ.0104 ) THEN
        WRITE(MESSAGE,0104) T,ARG
0104    FORMAT(1X,E16.9,1X,E16.9)
        GO TO 99999
      END IF

      IF ( N.EQ.0105 ) THEN
        WRITE(MESSAGE,0105) T,ARG
0105    FORMAT(1X,E16.9,' MISS = ',E16.9)
        GO TO 99999
      END IF

C
C      SUBROUTINE: CMPINV (0201...0300)
C
      IF ( N.EQ.0201 ) THEN
        WRITE(MESSAGE,0201)
0201    FORMAT(' MATRIX SIZE TOO LARGE IN CMPINV')
        GO TO 99999
      END IF

C
C      SUBROUTINE: DISCRT (0301...0400)
C
      IF ( N.EQ.0301 ) THEN
        WRITE(MESSAGE,0301)

```

```

0301  FORMAT(' SYSTEM ORDER TOO LARGE IN DISCRT')
      GO TO 99999
      END IF

      IF ( N.EQ.0302 ) THEN
        WRITE(MESSAGE,0302)
0302  FORMAT(' SUITABLE CONVERGENCE WAS NOT REACHED IN DISCRT')
      GO TO 99999
      END IF

C
C  SUBROUTINE: EIGVEC (0401...0500)
C
      IF ( N.EQ.0401 ) THEN
        WRITE(MESSAGE,0401)
0401  FORMAT(' MATRIX SIZE TOO LARGE IN EIGVEC')
      GO TO 99999
      END IF

C
C  SUBROUTINE: HQR (0501...0600)
C
      IF ( N.EQ.0501 ) THEN
        WRITE(MESSAGE,0501)
0501  FORMAT(' TOO MANY ITERATIONS IN HQR')
      GO TO 99999
      END IF

C
C  SUBROUTINE: KALMAN (0601...0700)
C
      IF ( N.EQ.0601 ) THEN
        WRITE(MESSAGE,0601) T
0601  FORMAT(1X,E16.9,' INITIATE ACQUISITION PHASE')
      GO TO 99999
      END IF

      IF ( N.EQ.0602 ) THEN
        WRITE(MESSAGE,0602) T
0602  FORMAT(1X,E16.9,' INITIATE TRACK PHASE')
      GO TO 99999
      END IF

      IF ( N.EQ.0603 ) THEN
        WRITE(MESSAGE,0603) T
0603  FORMAT(1X,E16.9,' INITIATE TERMINAL PHASE')
      GO TO 99999
      END IF

      IF ( N.EQ.0604 ) THEN
        WRITE(MESSAGE,0604) T,ARG
0604  FORMAT(1X,E16.9,' ACQUISITION MODE ENABLED:  MAGRO = ',E16.9)
      GO TO 99999
      END IF

      IF ( N.EQ.0605 ) THEN
        WRITE(MESSAGE,0605) T,ARG
0605  FORMAT(1X,E16.9,' TRACK MODE ENABLED:  MAGRO = ',E16.9)
      GO TO 99999
      END IF

      IF ( N.EQ.0606 ) THEN
        WRITE(MESSAGE,0606) T,ARG
0606  FORMAT(1X,E16.9,' CSO MODE ENABLED:  MAGRO = ',E16.9)
      GO TO 99999
      END IF

      IF ( N.EQ.0607 ) THEN
        WRITE(MESSAGE,0607) T,ARG
0607  FORMAT(1X,E16.9,' TERMINAL MODE ENABLED:  MAGRO = ',E16.9)
      GO TO 99999
      END IF

C
C  SUBROUTINE: MATINV (0701...0800)
C
      IF ( N.EQ.0701 ) THEN

```

```

      WRITE(MESSAGE,0701)
0701  FORMAT(' MATRIX SIZE TOO LARGE IN MATINV')
      GO TO 99999
    END IF

```

```

C
C  SUBROUTINE: MCGUID (0801...0900)
C
      IF ( N.EQ.0801 ) THEN
        WRITE(MESSAGE,0801) T
0801  FORMAT(1X,E16.9,' KV PITCHOVER COMPLETE',
        &      ' - BEGIN DISTURBANCE MEASUREMENT')
        GO TO 99999
      END IF

      IF ( N.EQ.0802 ) THEN
        WRITE(MESSAGE,0802) T
0802  FORMAT(1X,E16.9,' DISTURBANCE MEASUREMENT COMPLETE',
        &      ' - ORIENT KV TO LOS')
        GO TO 99999
      END IF

      IF ( N.EQ.0803 ) THEN
        WRITE(MESSAGE,0803) T
0803  FORMAT(1X,E16.9,' KV ORIENTATION COMPLETE')
        GO TO 99999
      END IF

```

```

C
C  SUBROUTINE: MISSIL (0901...1000)
C
      IF ( N.EQ.0901 ) THEN
        WRITE(MESSAGE,0901) T
0901  FORMAT(1X,E16.9,' MISSILE HAS CLEARED THE LAUNCHER')
        GO TO 99999
      END IF

```

```

C
C  SUBROUTINE: OPTSSC (1001...1100)
C
      IF ( N.EQ.1001 ) THEN
        WRITE(MESSAGE,1001)
1001  FORMAT(' MAXIMUM NUMBER OF STATES EXCEEDED IN OPTSSC')
        GO TO 99999
      END IF

```

```

C
C  SUBROUTINE: RANO (1101...1200)
C
      IF ( N.EQ.1101 ) THEN
        WRITE(MESSAGE,1101)
1101  FORMAT(' RANDOM NUMBER OUT OF BOUNDS IN RANO')
        GO TO 99999
      END IF

```

```

C
C  SUBROUTINE: SEEKER (1201...1300)
C
      IF ( N.EQ.1201 ) THEN
        WRITE(MESSAGE,1201) T
1201  FORMAT(1X,E16.9,' TRUE LOS ANGLE EXCEEDS FIELD-OF-VIEW LIMIT')
        GO TO 99999
      END IF

      IF ( N.EQ.1202 ) THEN
        WRITE(MESSAGE,1202) T
1202  FORMAT(1X,E16.9,' TARGET REACQUIRED')
      END IF

      IF ( N.EQ.1203 ) THEN
        WRITE(MESSAGE,1203) T,ARG
1203  FORMAT(1X,E16.9,' FRAME RATE CHANGE: FRMRAT = ',E16.9)
        GO TO 99999
      END IF

```

```

C
C  SUBROUTINE: SSPLAG (1301...1400)
C
  IF ( N.EQ.1301 ) THEN
    WRITE(MESSAGE,1301)
    1301  FORMAT(' BUFFER SIZE INSUFFICIENT IN SSPLAG')
    GO TO 99999
  END IF

C
C  SUBROUTINE: TARGET (1401...1500)
C
  IF ( N.EQ.1401 ) THEN
    WRITE(MESSAGE,1401) T,ARG
    1401  FORMAT(1X,E16.9,' TARGET RESOLVED:  RANGE = ',E16.9)
    GO TO 99999
  END IF

C
C  SUBROUTINE: VCSLOG (1501...1600)
C
  IF ( N.EQ.1501 ) THEN
    WRITE(MESSAGE,1501) T,ARG
    1501  FORMAT(1X,E16.9,' ISSUE MIDCOURSE DISTURBANCE BURN',
    &      ' - VCS THRUSTER ',F2.0)
    GO TO 99999
  END IF

  IF ( N.EQ.1502 ) THEN
    WRITE(MESSAGE,1502) T,ARG
    1502  FORMAT(1X,E16.9,' ISSUE MIDCOURSE BURN ',F2.0)
    GO TO 99999
  END IF

  IF ( N.EQ.1503 ) THEN
    WRITE(MESSAGE,1503) T,ARG
    1503  FORMAT(1X,E16.9,' ISSUE MIDCOURSE BURN ',F2.0,
    &      ' - BURN TIME BELOW THRESHOLD')
    GO TO 99999
  END IF

  IF ( N.EQ.1504 ) THEN
    WRITE(MESSAGE,1504) T
    1504  FORMAT(1X,E16.9,' ISSUE FIRST BURN')
    GO TO 99999
  END IF

  IF ( N.EQ.1505 ) THEN
    WRITE(MESSAGE,1505) T
    1505  FORMAT(1X,E16.9,' ISSUE FIRST BURN',
    &      ' - BURN TIME BELOW THRESHOLD')
    GO TO 99999
  END IF

  IF ( N.EQ.1506 ) THEN
    WRITE(MESSAGE,1506) T
    1506  FORMAT(1X,E16.9,' ISSUE SECOND BURN')
    GO TO 99999
  END IF

  IF ( N.EQ.1507 ) THEN
    WRITE(MESSAGE,1507) T
    1507  FORMAT(1X,E16.9,' ISSUE SECOND BURN',
    &      ' - BURN TIME BELOW THRESHOLD')
    GO TO 99999
  END IF

  IF ( N.EQ.1508 ) THEN
    WRITE(MESSAGE,1508) T
    1508  FORMAT(1X,E16.9,' ISSUE THIRD BURN')
    GO TO 99999
  END IF

  IF ( N.EQ.1509 ) THEN
    WRITE(MESSAGE,1509) T
    1509  FORMAT(1X,E16.9,' ISSUE THIRD BURN',
    &      ' - BURN TIME BELOW THRESHOLD')
    GO TO 99999
  END IF

```

```

      WRITE(MESSAGE,0001) N
0001  FORMAT(' ERROR: MESSAGE NUMBER = ',I4)

```

```

99999 CONTINUE
      CALL OUTPUT_MESSAGE( %VAL(Character_08BIT), MESSAGE )
      CALL OUTPUT_NL

      RETURN
      END

```

FILE: uu22.19g/sutility/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

```

FILE: uu22.19g/sutility/uuran0.for

```

C-----
C      REAL 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      BOTH :               ISEED
C
C      UPDATES :            NONE
C-----
C

```

```

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

      INTEGER*4  ISEED

      COMMON / RANCOM /          RANSEQ(97),      RANLST

C     USE PREVIOUSLY SAVED RANDOM NUMBER AS BUFFER INDEX AND MAKE
C     SURE ARRAY BOUNDS ARE NOT EXCEEDED .

      J      = 1 + INT ( 97.0*RANLST )
      IF ( J.LT.1 .OR. J.GT.97 ) THEN
        CALL OUTMES(1100,0.0,0.0)
      END IF

C     RETRIEVE RANDOM NUMBER FROM BUFFER FOR OUTPUT AND SAVE IT FOR
C     USE AS AN INDEX ON THE NEXT PASS .

      RANLST = RANSEQ(J)
      *      RANO  = DBLE ( RANLST )
      RANO  = RANLST

C     LOAD A NEW RANDOM NUMBER IN THE SLOT JUST VACATED .

      RANSEQ(J) = RAN ( ISEED )

      RETURN
      END

```

FILE: uuv22.19g/sutility/uuranit.for

```

C-----
      SUBROUTINE RANIT ( ISEED )
C-----
C
C     SUBROUTINE NAME :      RANIT
C
C     AUTHOR(S) :          D. F. SMITH
C
C     FUNCTION :            INITIALIZES A TABLE OF RANDOM NUMBERS FOR
C                           USE BY THE UNIFORM RANDOM GENERATOR RANO
C
C     CALLED FROM :         EXECUTIVE ROUTINE
C
C     SUBROUTINES CALLED :  RAN
C
C     INPUTS :              NONE
C
C     OUTPUTS :             NONE
C
C     BOTH :                ISEED
C
C     UPDATES :             NONE
C-----

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

      INTEGER*4  RANIT

      COMMON / RANCOM /          RANSEQ(97),      RANLST

C     EXERCISE SYSTEM ROUTINE

      DO 10 I = 1 , 97
        DUMMY = RAN ( ISEED )
10    CONTINUE

C     STORE 97 RANDOM NUMBERS IN BUFFER ( 97 IS NOT SPECIAL )

      DO 20 I = 1 , 97
        RANSEQ(I) = RAN ( ISEED )
20    CONTINUE

C     SAVE ANOTHER RANDOM NUMBER TO USE FOR INDEXING BUFFER

      RANLST = RAN ( ISEED )

```



```

      COLL = CO*TMP2
END IF

C   Overdamped filter
      IF ( ZD.GT.ONE ) THEN
        TMP5 = SQRT ( ZD**2 - ONE )
        A = WD*TMP5
        B = WD/TMP5
        ASQ = A*A
        BSQ = B*B
        EXPA = EXP ( - A*DT )
        EXPB = EXP ( - 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 )
        CILL = ASQ*EXPA*( EXPB*TMP4 - ONE )
        CO = A*B*DT*( A - B )
        COL = - CO*( EXPA + EXPB )
        COLL = CO*EXPA*EXPB
      END IF

      RETURN
      END

```

FILE: uuv22.19g/sutility/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 REAL (A-H)
C   IMPLICIT REAL (O-Z)
C   REAL XM(3,3)
C
C   SX = SIN(X)
C   CX = COS(X)
C
C   IF ( I.EQ.1 ) THEN
C     XM(1,1) = 1.0
C     XM(1,2) = 0.0
C     XM(1,3) = 0.0
C
C     XM(2,1) = 0.0
C     XM(2,2) = CX
C     XM(2,3) = SX
C
C     XM(3,1) = 0.0
C     XM(3,2) = -SX
C     XM(3,3) = CX
C   END IF
C
C   IF ( I.EQ.2 ) THEN
C     XM(1,1) = CX
C     XM(1,2) = 0.0
C     XM(1,3) = -SX

```

```

      XM(2,1) = 0.0
      XM(2,2) = 1.0
      XM(2,3) = 0.0

      XM(3,1) = SX
      XM(3,2) = 0.0
      XM(3,3) = CX
      END IF

      IF ( I.EQ.3 ) THEN
        XM(1,1) = CX
        XM(1,2) = SX
        XM(1,3) = 0.0

        XM(2,1) = -SX
        XM(2,2) = CX
        XM(2,3) = 0.0

        XM(3,1) = 0.0
        XM(3,2) = 0.0
        XM(3,3) = 1.0
      END IF

      RETURN
      END

```

FILE: uuv22.19g/sutility/uuseeker.for

```

-----
      SUBROUTINE SEEKER(T,ACQD,LAMSEK,MAGRTR,SKSEED,FRMRAT,FRMCNT,
      .               SAMRAT,TRACK,TERM,SNR,LAMM)
-----
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
C                          D. SMITH - CR # 082
C                          R. RHYNE - CR # 084
C                          R. RHYNE - CR # 087
C                          R. RHYNE - CR # 088
C                          B. HILL - CR # 093
C
-----

```

```

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

      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)

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

C      LOCAL DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

      SAVE          ISEKR,   IFOV

* DATA INITIALIZATION
$INCLUDE('~/INCLUDE/SSCON47.DAT')
$INCLUDE('~/INCLUDE/SSCON48.DAT')
$INCLUDE('~/INCLUDE/SSCON50.DAT')
$INCLUDE('~/INCLUDE/SSCON55.DAT')
$INCLUDE('~/INCLUDE/SSCON61.DAT')
$INCLUDE('~/INCLUDE/SSCON68.DAT')
$INCLUDE('~/INCLUDE/SSCON10.DAT')
$INCLUDE('~/INCLUDE/SSCON11.DAT')

      DATA ISEKR / 1 /
      DATA IFOV / 0 /
      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(1201,T,0.0)
        IFOV = 1
      ELSE IF ( LAMFOV.LE.FOVCHK .AND. IFOV.EQ.1 ) THEN
        CALL OUTMES(1202,T,0.0)
        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      SIMPLE SEEKER MODEL

*      IF ( SEKTYP.EQ.1 ) THEN
*      FRMRAT = 1.0/SAMRAT
*      CALL NORM(1.0,0.0,SKSEED,RANA)
*      CALL TABLE(SEKTIM,SEKNOS,T,SKNOSA,24,IT)
*      LAMSK(1) = LAMSEK(1) + 0.001*RANA*SKNOSA
*      CALL NORM(1.0,0.0,SKSEED,RANB)
*      CALL TABLE(SEKTIM,SEKNOS,T,SKNOSB,24,IT)
*      LAMSK(2) = LAMSEK(2) + 0.001*RANB*SKNOSB

C      ANGLE QUANTIZATION

*      IF ( QNTZP .GT. 0. ) THEN
*      LAMM(1) = (AINT(LAMSK(1))/QNTZP + 0.5)*QNTZP
*      LAMM(2) = (AINT(LAMSK(2))/QNTZP + 0.5)*QNTZP
*      ELSE
*      LAMM(1) = LAMSK(1)
*      LAMM(2) = LAMSK(2)
*      END IF
*      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 NORM(1.0,0.0,SKSEED,RANA)
      CALL NORM(1.0,0.0,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
        CALL OUTMES(1203,T,FRMRAT)
      ENDIF
    ENDIF

RETURN

```

END

FILE: uuv22.19g/sutility/uussplag.for

```

C-----
C      SUBROUTINE SSPLAG(T,LAMM,RREL,VREL,TI2M,SNR,LATCH,KFSF,TKFU,
C      .               LAMMO,RRELO,VRELO,TI2MO,SNRO)
C-----
C
C      SUBROUTINE NAME :      SSPLAG
C
C      AUTHOR(S) :          D. F. SMITH
C
C      FUNCTION :           Emulate the signal processing lag which
C                           occurs between the seeker signal processor
C                           input and output.
C
C      CALLED FROM :        FORTRAN MAIN
C
C      SUBROUTINES CALLED :  NONE
C
C      INPUTS :             T,LAMM,RREL,VREL,TI2M,SNR,LATCH
C
C      OUTPUTS :            KFSF,TKFU,LAMMO,RRELO,VRELO,TI2MO,SNRO
C
C      UPDATES :            D. SISSOM   - CR # 069
C                           B. HILL /   - CR # 081
C                           R. RHYNE
C                           R. RHYNE   - CR # 087
C-----

```

PARAMETER (NSAVMX=10)

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

REAL LAMM(2),	LAMMO(2),	LAMMSV(2,NSAVMX)
REAL RREL(3),	RRELO(3),	RRELSV(3,NSAVMX)
REAL VREL(3),	VRELO(3),	VRELSV(3,NSAVMX)
REAL TI2M(9),	TI2MO(9),	TI2MSV(9,NSAVMX)
REAL SNR,	SNRO,	SNRSV(NSAVMX)
REAL TLATCH(NSAVMX)		

* DATA INITIALIZATION

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

\$INCLUDE('~/INCLUDE/SSCON56.DAT')

C ENSURE THAT BUFFER BOUNDARIES ARE NOT VIOLATED

```

IF ( LATCH.GT.0 ) THEN
  NLATCH = NLATCH + 1
  IF ( NLATCH.GT.NSAVMX ) THEN
    CALL OUTMES(1301,0.0,0.0)
  ENDIF
ENDIF

```

C LATCH DATA INTO BUFFER AT MEASUREMENT TIME

```

IF ( LATCH.GT.0 ) THEN
  TLATCH(NLATCH) = T + SPLAG
  LAMMSV(1,NLATCH) = LAMM(1)
  LAMMSV(2,NLATCH) = LAMM(2)
  RRELSV(1,NLATCH) = RREL(1)
  RRELSV(2,NLATCH) = RREL(2)
  RRELSV(3,NLATCH) = RREL(3)
  VRELSV(1,NLATCH) = VREL(1)
  VRELSV(2,NLATCH) = VREL(2)
  VRELSV(3,NLATCH) = VREL(3)
  TI2MSV(1,NLATCH) = TI2M(1)
  TI2MSV(2,NLATCH) = TI2M(2)
  TI2MSV(3,NLATCH) = TI2M(3)
  TI2MSV(4,NLATCH) = TI2M(4)
  TI2MSV(5,NLATCH) = TI2M(5)
  TI2MSV(6,NLATCH) = TI2M(6)
  TI2MSV(7,NLATCH) = TI2M(7)
  TI2MSV(8,NLATCH) = TI2M(8)
  TI2MSV(9,NLATCH) = TI2M(9)
  SNRSV(NLATCH) = SNR

```

ENDIF

C UNLATCH DATA FROM BUFFER AT KALMAN FILTER PROCESSING TIME

```

IF ( LATCH.LT.0 ) THEN
  LAMMO(1) = LAMMSV(1,1)
  LAMMO(2) = LAMMSV(2,1)
  RRELO(1) = RRELSV(1,1)
  RRELO(2) = RRELSV(2,1)
  RRELO(3) = RRELSV(3,1)
  VRELO(1) = VRELSV(1,1)
  VRELO(2) = VRELSV(2,1)
  VRELO(3) = VRELSV(3,1)
  TI2MO(1) = TI2MSV(1,1)
  TI2MO(2) = TI2MSV(2,1)
  TI2MO(3) = TI2MSV(3,1)
  TI2MC(4) = TI2MSV(4,1)
  TI2MO(5) = TI2MSV(5,1)
  TI2MO(6) = TI2MSV(6,1)
  TI2MO(7) = TI2MSV(7,1)
  TI2MO(8) = TI2MSV(8,1)
  TI2MO(9) = TI2MSV(9,1)
  SNRO = SNRV(1)
ENDIF

```

C ALTER BUFFER CONTENTS WHEN DATA IS UNLATCHED

```

IF ( LATCH.LT.0 ) THEN
  DO 20 I = 1, NLATCH-1
    TLATCH(I) = TLATCH(I+1)
    LAMMSV(1,I) = LAMMSV(1,I+1)
    LAMMSV(2,I) = LAMMSV(2,I+1)
    RRELSV(1,I) = RRELSV(1,I+1)
    RRELSV(2,I) = RRELSV(2,I+1)
    RRELSV(3,I) = RRELSV(3,I+1)
    VRELSV(1,I) = VRELSV(1,I+1)
    VRELSV(2,I) = VRELSV(2,I+1)
    VRELSV(3,I) = VRELSV(3,I+1)
    TI2MSV(1,I) = TI2MSV(1,I+1)
    TI2MSV(2,I) = TI2MSV(2,I+1)
    TI2MSV(3,I) = TI2MSV(3,I+1)
    TI2MSV(4,I) = TI2MSV(4,I+1)
    TI2MSV(5,I) = TI2MSV(5,I+1)
    TI2MSV(6,I) = TI2MSV(6,I+1)
    TI2MSV(7,I) = TI2MSV(7,I+1)
    TI2MSV(8,I) = TI2MSV(8,I+1)
    TI2MSV(9,I) = TI2MSV(9,I+1)
    SNRSV(I) = SNRSV(I+1)
20  CONTINUE
    TLATCH(NLATCH) = 0.0
    LAMMSV(1,NLATCH) = 0.0
    LAMMSV(2,NLATCH) = 0.0
    RRELSV(1,NLATCH) = 0.0
    RRELSV(2,NLATCH) = 0.0
    RRELSV(3,NLATCH) = 0.0
    VRELSV(1,NLATCH) = 0.0
    VRELSV(2,NLATCH) = 0.0
    VRELSV(3,NLATCH) = 0.0
    TI2MSV(1,NLATCH) = 0.0
    TI2MSV(2,NLATCH) = 0.0
    TI2MSV(3,NLATCH) = 0.0
    TI2MSV(4,NLATCH) = 0.0
    TI2MSV(5,NLATCH) = 0.0
    TI2MSV(6,NLATCH) = 0.0
    TI2MSV(7,NLATCH) = 0.0
    TI2MSV(8,NLATCH) = 0.0
    TI2MSV(9,NLATCH) = 0.0
    SNRSV(NLATCH) = 0.0
    NLATCH = NLATCH - 1
ENDIF

```

C DETERMINE TIME OF NEXT KALMAN FILTER UPDATE AND ENABLE KALMAN
C FILTER SCHEDULING FLAG AS NEEDED

```

IF ( LATCH.GT.0 .AND. NLATCH.EQ.1 ) THEN
  TKFU = TLATCH(1)
  KFSF = 1
ELSE IF ( LATCH.LT.0 .AND. NLATCH.GT.0 ) THEN
  TKFU = TLATCH(1)
  KFSF = 1
ELSE

```

```

      KFSF = 0
ENDIF

RETURN
END

```

FILE: uuv22.19g/utility/uutable.for

```

-----
C      SUBROUTINE TABLE(XTAB,YTAB,X,Y,N,I)
-----
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
C            CONTINUE
C          10  FRACT = ( X - XTAB(K) ) / ( XTAB(K+1) - XTAB(K) )
C          20  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
C            CONTINUE
C          30  FRACT = ( X - XTAB(K) ) / ( XTAB(K+1) - XTAB(K) )
C          40  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
C          K = 1
C          L = N
C          DO 50 I = K , L
C            IF ( L.EQ.K+1 ) GO TO 60
C            M = ( K + L ) / 2
C            IF ( X.LT.XTAB(M) ) THEN
C              L = M
C            ELSE
C              K = M
C            END IF
C          50 CONTINUE
C          60 CONTINUE
C        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

```

FILE: uvv22.19g/sutility/uutimer.for

```

      SUBROUTINE INITIALIZE_TIMER()
$INCLUDE(' :PFP:INCLUDE/TARGET.FOR')
$INCLUDE(' ^/INCLUDE/UUTIMER.COM')
$INCLUDE(' ^/INCLUDE/SSCON22.DAT')
$INCLUDE(' ^/INCLUDE/SSCON23.DAT')
      INTEGER BN, TN

      DO 20 BN=1,4
        DO 10 TN=1,500
          NUMBER_TIMER(BN,TN) = 0
          NUMBER_TICKS(BN,TN) = 0.0
10        CONTINUE
20      CONTINUE

      STAGE1 = INT4( TSTG1 * 1000.0 )
      STAGE2 = INT4( TSTG2 * 1000.0 )
      CALL RESET_TIMER()
      END

      SUBROUTINE START_TIMER( TN )
$INCLUDE(' :PFP:INCLUDE/TARGET.FOR')
$INCLUDE(' ^/INCLUDE/UUTIMER.COM')
      INTEGER TN

      TIMER(TN) = READ_TIMER()
      END

      SUBROUTINE STOP_TIMER( TN )
$INCLUDE(' :PFP:INCLUDE/TARGET.FOR')
$INCLUDE(' ^/INCLUDE/UUTIMER.COM')
      INTEGER TN

      TIMER(TN) = TIMER(TN) - READ_TIMER()

      NUMBER_TIMER(4,TN) = NUMBER_TIMER(4,TN) + 1
      NUMBER_TICKS(4,TN) = NUMBER_TICKS(4,TN) + DBLE(TIMER(TN))

      IF ( NUMBER_TIMER(4,TN) .LT. STAGE1 ) THEN
        NUMBER_TIMER(1,TN) = NUMBER_TIMER(1,TN) + 1
        NUMBER_TICKS(1,TN) = NUMBER_TICKS(1,TN) + DBLE(TIMER(TN))
      ELSEIF ( NUMBER_TIMER(4,TN) .LT. STAGE2 ) THEN
        NUMBER_TIMER(2,TN) = NUMBER_TIMER(2,TN) + 1
        NUMBER_TICKS(2,TN) = NUMBER_TICKS(2,TN) + DBLE(TIMER(TN))
      ELSE
        NUMBER_TIMER(3,TN) = NUMBER_TIMER(3,TN) + 1
        NUMBER_TICKS(3,TN) = NUMBER_TICKS(3,TN) + DBLE(TIMER(TN))
      ENDIF
      END

      SUBROUTINE OUTPUT_TIMER()
$INCLUDE(' :PFP:INCLUDE/TARGET.FOR')
$INCLUDE(' ^/INCLUDE/UUTIMER.COM')
      INTEGER BN, TN
      INTEGER*4 AVERAGE

      DO 20 TN=1,500
        IF ( NUMBER_TIMER(4,TN) .NE. 0 ) THEN
          CALL OUTPUT_MESSAGE(%VAL(SIGNED 16BIT),TN,%VAL(INT2(1)))
          CALL OUTPUT_MESSAGE(%VAL(CHARACTER_08BIT), 'TIMER ' )
        DO 10 BN=1,4

```

```

      IF ( NUMBER_TIMER(BN,TN) .NE. 0 ) THEN
        AVERAGE = INT4(NUMBER_TICKS(BN,TN)/
& DBLE(NUMBER_TIMER(BN,TN)))
      ELSE
        AVERAGE = 0
      ENDIF
      CALL OUTPUT_MESSAGE(%VAL(SIGNED_32BIT),AVERAGE,
& %VAL(INT2(1)))
10    CONTINUE

      CALL OUTPUT_NL
      END IF
20    CONTINUE
      END

```

FILE: uuv22.19g/utility/outlu2ei.for

```

C-----
C      SUBROUTINE TLU2EI ( X, Y, F, I, J, TBVAL )
C-----
C
C      SUBROUTINE NAME :      AERO
C
C      AUTHOR(S) :          B. HILL
C
C      FUNCTION :           PERFORMS A LINEAR TABLE LOOK-UP OF A TABLE
C                           WITH TWO INDEPENDENT VARIABLES, AND RETURNS
C                           INDICES POINTING TO THE AREA OF THE TABLE
C                           IN USE
C
C      CALLED FROM :        AERO, BAUTO
C
C      SUBROUTINES CALLED :  ABS
C
C      INPUTS :             X,Y,F
C
C      OUTPUTS :            I,J,TBVAL
C
C      UPDATES :            D. SMITH - CR # 59
C-----
C
C      IMPLICIT REAL (A-H)
C      IMPLICIT REAL (O-Z)
C      REAL F( 3 )
C
C      EQUIVALENCE (N12, NYU), (N21, NXL), (N22, NXU), (N11, ISP)
C      EQUIVALENCE ( DX, XX), ( DY, YY)
C
C      COMPUTE UPPER AND LOWER BOUNDS ON INDICES FOR XX AND YY
C
C      NXU = ABS( F(1) ) + .1
C      MP1 = ABS( F(2) ) + 1.1
C      NYU = MP1 + 1
C      NXL = NYU + 1
C      NXU = NXU*MP1 + 2
C      JS = J
C      IS = I
C      XX = X
C      YY = Y
C      IF(( F(1) .GE. 0.0 ).AND.( F(2) .GE. 0.0 )) GO TO 5
C
C      SWAP THE INDEPENDENTS - MIRROR IMAGE TABLE WITH FIXED
C      PROGRAM CALLING SEQUENCE
C
C      XX = Y
C      YY = X
C5  CONTINUE
C
C      GET POINTERS WITHIN LIMITS
C
C      IF( IS .LT. NXL ) IS = NXL
C      IF( JS .LT. 3 ) JS = 3
C
C      FIND GREATEST LOWER BOUND ON INDEX FOR YY
C      (UNLESS YY IS OFF THE TABLE)
C
C10  CONTINUE
C      ISP = .5 + 1

```

```

      IF( YY .LE. F(JSP) ) GO TO 30
      IF( JSP .EQ. NYU ) GO TO 100
      JS = JSP
      GO TO 10
C
      20 CONTINUE
      IF( JS .EQ. 3 ) GO TO 100
      JS = JS - 1
C
      30 CONTINUE
      IF( YY .LT. F(JS) ) GO TO 20
C
C      FIND GREATEST LOWER BOUND ON INDEX FOR XX
C      (UNLESS XX IS OFF THE TABLE)
C
      100 CONTINUE
      ISP = IS + MP1
      IF( XX .LE. F(ISP) ) GO TO 300
      IF( ISP .EQ. NXU ) GO TO 400
      IS = ISP
      GO TO 100
C
      200 CONTINUE
      IF( IS .EQ. NXL ) GO TO 400
      IS = IS - MP1
C
      300 CONTINUE
      IF( XX .LT. F(IS) ) GO TO 200
C
      400 CONTINUE
C
      SET UP INDEXING+      F(JS)      YY      F(JS+1)
      (INTERPOLATING)      *****
C      *
C      F(IS) * F(N11)      F(N12) *
C      *
C      XX * XJ      DOUBLE XJP1 *
C      *
C      F(IS+MP1) * F(N21)      F(N22) *
C      *
C      *****
C
      N11 = IS + JS - 2
      N12 = N11 + 1
      N21 = N11 + MP1
      N22 = N21 + 1
C
      IPMP1 = IS + MP1
      DX = ( XX - F(IS) ) / ( F(IPMP1) - F(IS) )
      XJ = ( F(N21) - F(N11) ) * DX + F(N11)
      XJP1 = ( F(N22) - F(N12) ) * DX + F(N12)
      DY = ( YY - F(JS) ) / ( F(JS + 1) - F(JS) )
C
C      RESULTS
C
      J = JS
      I = IS
      TBVAL = ( XJP1 - XJ ) * DY + XJ
C
      RETURN
      END

```

FILE: uuv22.19g/sutlity/uuvvcssthl.for

```

-----
SUBROUTINE VCSTH1(T,CG,TBURNM,IVCS,TOFFLT,TIMONV,DTOFFV,TVTAB,
. FOFF1,FOFF2,IVTAB,FVCS,FVCS,FZVCS,MXVCS,MYVCS,MZVCS,MDOTV)
-----
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,TOFFLT,TIMONV,DTOFFV,
C TVTAB,FOFF1,FOFF2
C
C OUTPUTS : FXVCS,FYVCS,FZVCS,MXVCS,MYVCS,MZVCS,MDOTV
C
C BOTH : 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-----

```

```

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 DATA USED FOR CONSTANTS, VARIABLES AND INITIALIZATION FLAG

```

```

SAVE IVCSTH , VCSMA

```

```

* DATA INITIALIZATION
$INCLUDE('^/INCLUDE/SSVCSTHR.DAT')
$INCLUDE('^/INCLUDE/SSCON70.DAT')
$INCLUDE('^/INCLUDE/SSCON09.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) = FOFF1(I)
    VOFF2(I) = FOFF2(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
200 CONTINUE

```

```

      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
      ENDIF

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

* The IVTAB assignment was moved to the partition with VCSLOG
*   IVTAB = 0

      IF (TBURNM .GE. TCMINV) THEN

C    DEFINE VCS THRUST PROFILE

      TMVCS(1,IVCS) = IVTAB
      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

      TBURNM = 0.0

      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(I,I).GT.0.0) THEN
        CALL TABLE(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 + ATHRV(I)/ISPVCS
20    CONTINUE

    END

```