(NASA-CR-187248) RESEARCH AND DEVELOPMENT FOR ONBOARD NAVIGATION (ONAV) GROUND BASED EXPERT/TRAINER SYSTEM: ONAV ENTRY EXPERT SYSTEM CODE (LinCom Corp.) 181 p CSCL 14B

N91-12720

Unclas
G3/14 0308270

Research and Development for Onboard Navigation (ONAV) Ground Based Expert/Trainer System

ONAV Entry Expert System Code

Daniel C. Bochsler

LinCom Corporation

January 28, 1988

Cooperative Agreement NCC 9-16
Research Activity No. *AI.8*

Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information systems in 1986 to encourage NASA Johnson Space Center and local industry to actively support research in the computing and information sciences. As part of this endeavor, UH-Clear Lake proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a three-year cooperative agreement with UH-Clear Lake beginning in May, 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The mission of RICIS is to conduct, coordinate and disseminate research on computing and information systems among researchers, sponsors and users from UH-Clear Lake, NASA/JSC, and other research organizations. Within UH-Clear Lake, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business, Education, Human Sciences and Humanities, and Natural and Applied Sciences.

Other research organizations are involved via the "gateway" concept. UH-Clear Lake establishes relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research.

A major role of RICIS is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. Working jointly with NASA/JSC, RICIS advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research, and integrates technical results into the cooperative goals of UH-Clear Lake and NASA/JSC.

Research and Development for Onboard Navigation (ONAV) Ground Based Expert/Trainer System

ONAV Entry Expert System Code

Preface

This research was conducted under the auspices of the Research Institute for Computing and Information Systems by LinCom Corporation under the direction of Daniel C. Bocshsler. Terry Feagin, Professor of Computer Science at the University of Houston - Clear Lake, served as the technical representative for RICIS.

Funding has been provided by the Mission Planning and Analysis Division, NASA/JSC through Cooperative Agreement NCC 9-16 between NASA Johnson Space Center and the University of Houston - Clear Lake. The NASA Technical Monitor for this activity was Robert Savely, Head, Artificial Intelligence Section, Technology Development and Applications Branch, NASA/JSC.

The views and conclusions contained in this report are those of the author and should not be interpreted as representative of the official policies, either express or implied, of NASA or the United States Government.

Research and Development for Onboard Navigation (ONAV)

Ground Based Expert/Trainer System

ONAV ENTRY EXPERT SYSTEM CODE

(Deliverable C)

Prepared For:

Dr. Terry Feagin
Research Institute for Computing and Information Systems
University of Houston - Clear Lake

Prepared By:

Daniel C. Bochsler
LinCom Corporation
18100 Upper Bay Road, Suite 208
Houston, Texas 77058

Performed Under:

Project No. AI.8
Cooperation Agreement no. NCC9-16
Subcontract No. 005

January 28, 1988

EXPERT SYSTEM CODE FOR THE ONBOARD NAVIGATION (ONAV) CONSOLE EXPERT/TRAINER SYSTEM

ENTRY PHASE

January 1988

LinCom Corporation
Houston Texas

TABLE OF CONTENTS

Section Page

1 SUMMARY 1-1

2 INTRODUCTION 2-1
2.1 Purpose 2-1
2.2 Rule Organization Overview 2-1

3 SOURCE CODE LISTINGS 3-1
3.1 Initial Conditions 3-2
3.2 Telemetry Status 3-9
3.3 Runway Status 3-11
3.4 Inertial Measurement Units 3-16
3.5 State Vectors 3-55
3.6 Three-String State Vectors 3-66
3.7 Drag Altitude 3-72
3.8 Tactical Air Navigation 3-77
3.9 Baro Altitude 3-118
3.10 Microwave Scan Beam Landing System . 3-127
3.11 High Speed Trajectory Determinator . 3-145
3.12 Control Flow 3-150
3.13 Operator Input 3-153
3.14 Output Management 3-158
3.15 Data Tables 3-164

4 REFERENCES 4-1

Section 1

SUMMARY

This document provides the user with a listing of the expert rules for the ENTRY phase of the Onboard Navigation (ONAV) Console Expert/Trainer system. Included is an overview of each group of rules into which the program is divided.

[illegible]

Section 2

INTRODUCTION

Space shuttle

2.1 PURPOSE

The purpose of this document is to present a complete listing of the expert system rules for the Entry phase of the ONAV expert system. These source listings appear in the same format as utilized and required by the CLIPS (C Language Integrated Production System) expert system shell which is the basis for the ONAV entry system.

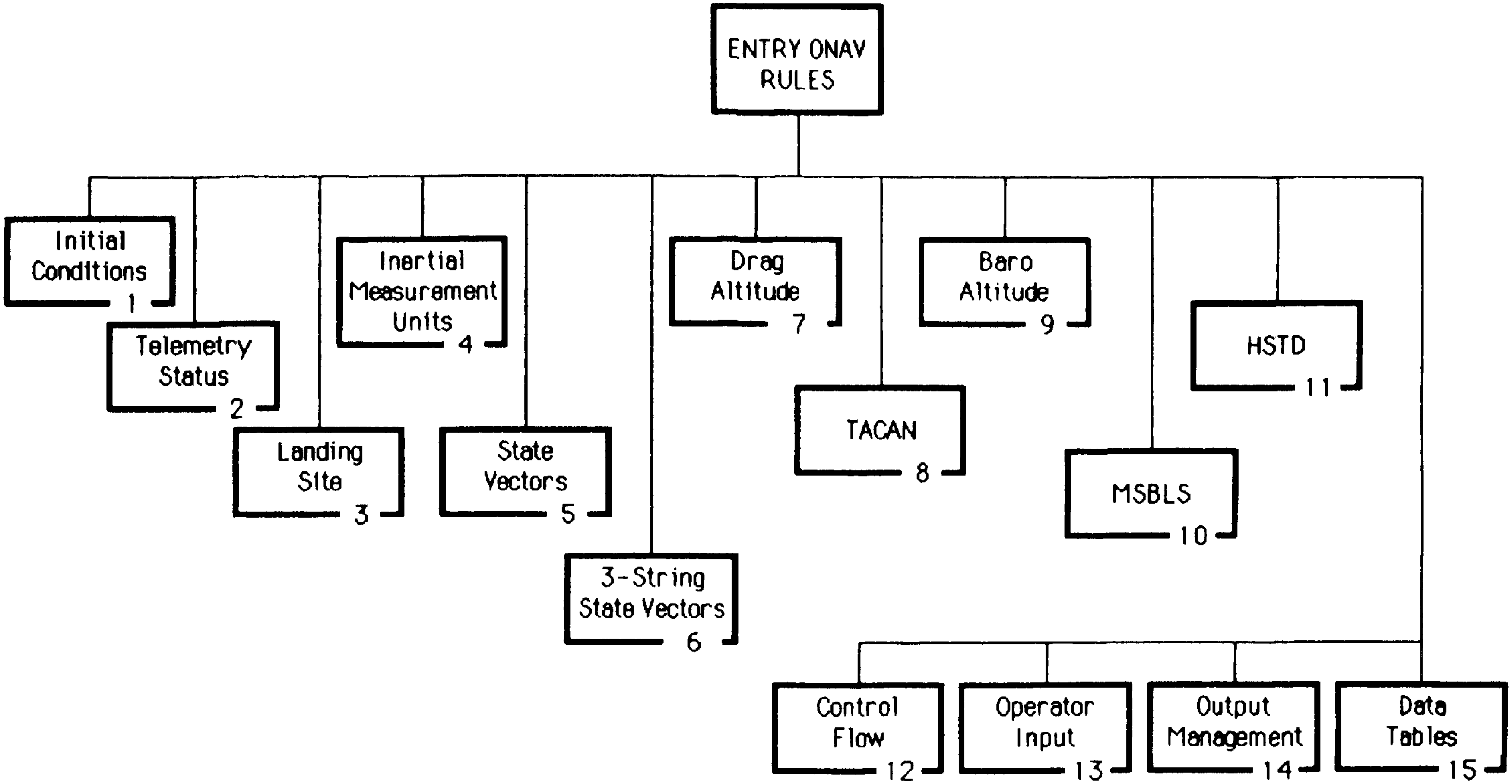
(CLIPS)

2.2 RULE ORGANIZATION OVERVIEW

Figure 2.2-1 gives a schematic overview of how the rules are organized. These groups result from a partitioning of the rules according to the overall function which a given set of rules performs. This partitioning was established and maintained according to that established in the knowledge specification document [1].

In addition, four other groups of rules are specified in this document. The four groups (control flow, operator inputs, output management, and data tables) perform functions that affect all the other functional rule groups. As the name implies; 1) control flow ensures that the rule groups are executed in the order required for proper operation; 2) operator input rules control the introduction into the CLIPS fact base of various kinds of data required by the expert system; 3) output management rules control the updating of the ONAV expert system user display screen during execution of the system; and 4) data tables are static information utilized by many different rule sets gathered in one convenient place.

Figure 2.2-1: ONAV Entry Expert System Rule Organization



Section 3

SOURCE CODE LISTINGS

The following sections provide lists of the Entry ONAV expert system source code in the CLIPS format.

3.1 Initial Conditions

```
;;*************************************************************************
;;
;;;      GROUP
;;          Initial Conditions (3.1)
;;
;;          This group handles some global types
;;          of info used by many rules sections
;;          (e.g., engaged system, system availability
;;          wrong atmosphere, wrong major mode, etc.).
;;
;;       CONTROL FACTS
;;          (sub-phase init ?)
;;
;;       CONTAINING GROUP
;;          Entry
;;
;;*************************************************************************

;;;  FACTS

(deffacts  monitoring-init-phases       ;These facts list the sequence of
                                        ;sub phases in the monitoring phase
                                        ;of the init rules
        (first-sub-phase  init  monitoring  status)
)                                       ;There is only 1 subphase

(deffacts  init-phase-facts             ;This fact indicates which
                                        ;system pass or bfs is the
                                        ;proper source of information
        (engaged-system none)
        (system-available none)
)                                       ;default is set to none


(deffacts  string-phases
        (first-sub-phase  string  monitoring  commfault)
        (first-sub-phase  string  analysis  clear)
)

(deffacts  initial-strings
        (prev-string-cf  pass  1  off)
        (prev-string-cf  pass  2  off)
        (prev-string-cf  pass  3  off)
        (prev-string-cf  pass  4  off)
        (prev-string-cf  bfs  1  off)
        (prev-string-cf  bfs  2  off)
        (prev-string-cf  bfs  3  off)
        (prev-string-cf  bfs  4  off)
)

;;------------------------------------------------------------------------

(defrule engaged-system-is-bfs

;;       IF
;;               BFS engage is on
;;       THEN
;;               BFS is the engaged system
;;       END
```

```
        (sub-phase init status)
        (bfs-engage on)
        ?x <- (engaged-system ~bfs)
        =>
        (retract ?x)
        (assert (engaged-system bfs)))

;;------------------------------------------------------------------------------

(defrule engaged-system-is-pass

;;      IF
;;              BFS engage is off
;;      THEN
;;              PASS is the engaged system
;;      END

        (sub-phase init status)
        (bfs-engage off)
        ?x <- (engaged-system ~pass)
        =>
        (retract ?x)
        (assert (engaged-system pass)))


;;------------------------------------------------------------------------------

;; Note: The following 3 availability rules were implemented
;;       with the assumption that CLIPS would ensure that
;;       two duplicate facts are not allowed to reside in
;;       the fact base.  These rules will cause duplicate
;;       facts to be generated; therefore, proper operation
;;       depends upon the above stated feature of CLIPS to
;;       be active.(i.e., check-facts function is assumed to
;;       be "on".)

(defrule system-availability-bfs-only

;;      IF
;;              the BFS is engaged
;;      THEN
;;              the BFS is the only system available

        (sub-phase init status)
        (engaged-system bfs)
        ?x <- (system-available ~bfs)
        =>
        (retract ?x)
        (assert (system-available bfs)))

;;------------------------------------------------------------------------------

(defrule system-availability-pass-only

;;      IF
;;              the BFS is not engaged
;;              the BFS is no go
;;      THEN
;;              the PASS is the only system available
```

```
        (sub-phase init status)
        (not (engaged-system bfs))
        (bfs-status no-go)
        ?x <- (system-available ~pass)
        =>
        (retract ?x)
        (assert (system-available pass)))

;;----------------------------------------------------------------------------

(defrule system-availability-both

;;      IF
;;              the BFS is not engaged
;;              the BFS is Go
;;      THEN
;;              both systems are available

        (sub-phase init status)
        (not (engaged-system bfs))
        (bfs-status go)
        =>
        (assert (system-available bfs))
        (assert (system-available pass)))

;;----------------------------------------------------------------------------

(defrule  wrong-atmosphere

;;      IF
;;              For the engaged system
;;              The ONAV operator desired atmosphere model
;;                    is not the same as the downlisted model
;;      THEN
;;              Notify operator that crew has incorrect atmosphere
;;                  selected
;;              Recommend call to crew to select the desired atmosphere
;;      END

        (sub-phase  init status)
        (engaged-system ?sys)
        (atmosphere  desired  ?model)
        (atmosphere  ?sys  ~?model)
        =>
        (assert  (status-light  drag  0  atmos))
        (if  (eq ?model nominal) then
                (bind ?item 37)
        else (if (eq ?model cold) then
                (bind ?item 38)
        else
                (bind ?item 39)))
        (assert  (recommend  drag atmos  off-nominal  alt
                "Need to select the " ?model " atmosphere by ITEM "
                ?item " on SPEC 51")))

;-----------------------------------------------------------------------------

(defrule  right-atmosphere

;;      IF
;;          The desired atmosphere is the same as the downlisted
;;             atmosphere
;;       THEN
;;          Notify operator that correct atmosphere is selected

         (sub-phase init status)
         (engaged-system ?sys)
         (atmosphere  desired  ?model)
         (atmosphere  ?sys  ?model)
         =>
         (assert  (status-light  drag  0  blank)))

;;------------------------------------------------------------

(defrule wrong-major-mode

;;       IF
;;               For the available systems,
;;               the major mode is not 304
;;       THEN
;;               Notify the operator that the (system) is in the wrong
;;               major mode.
;;               Recommend call to crew to select major mode 304 in
;;               the (system).

         (sub-phase  init  status)
         (system-available  ?sys)
         (major-mode  ?sys  ~304)
         =>
         (assert (recommend  three-state wrong-majormode  off-nominal alt
         " wrong" " major mode in "  ?sys
         " ; Recommend crew call to select mm304")))


;;**************************************************************************
;;
;;;  GROUP String Commfaults
;;
;;      This group notifies the operator when commfaults occur or clear up
;;      on entire strings.
;;
;;;  CONTROL FACTS
;       (sub-phase  string  ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;**************************************************************************

(defrule  commfault-string-pass

;;       IF
;;               A string is commfaulted in the PASS  AND
;;               The string was not previously commfaulted
;;       THEN
;;               Notify the operator that the string is commfaulted
;;       END

         (sub-phase  string  commfault)
```

```
        (string-commfault  pass  ?string  on)
        ?x <- (prev-string-cf  pass  ?string  off)
        =>
        (retract  ?x)
        (assert  (prev-string-cf  pass  ?string  on))
        (assert  (event  three-state  off-nominal  alt
                "Commfault string " ?string " in the PASS")))

;-----------------------------------------------------------------------------

(defrule  commfault-string-bfs

;;      IF
;;              A string is commfaulted in the BFS  AND
;;              The string was not previously commfaulted
;;      THEN
;;              Notify the operator that the string is commfaulted
;;      END

        (sub-phase  string  commfault)
        (string-commfault  bfs  ?string  on)
        ?x <- (prev-string-cf  bfs  ?string  off)
        =>
        (retract  ?x)
        (assert  (prev-string-cf  bfs  ?string  on))
        (assert  (event  three-state  off-nominal  alt
                "Commfault string " ?string " in the BFS")))

;-----------------------------------------------------------------------------

(defrule  clear-string-pass

;;      IF
;;              A string is not commfaulted in the PASS  AND
;;              The string was previously commfaulted
;;      THEN
;;              Notify the operator that the commfault is clear
;;      END

        (sub-phase  string  clear)
        (string-commfault  pass  ?string  off)
        ?x <- (prev-string-cf  pass  ?string  on)
        =>
        (retract  ?x)
        (assert  (prev-string-cf  pass  ?string  off))
        (assert  (event  three-state  off-nominal  alt
                "Commfault on string " ?string " has cleared in the PASS")))

;-----------------------------------------------------------------------------

(defrule  clear-string-bfs

;;      IF
;;              A string is not commfaulted in the BFS  AND
;;              The string was previously commfaulted
;;      THEN
;;              Notify the operator that the commfault is clear
;;      END

        (sub-phase  string  clear)
```

```
(string-commfault  bfs  ?string  off)
?x <- (prev-string-cf  bfs  ?string  on)
=>
(retract  ?x)
(assert  (prev-string-cf  bfs  ?string  off))
(assert  (event  three-state  off-nominal  alt
        "Commfault on string " ?string " has cleared in the BFS")))
```

3.2 Telemetry Status

```
;; ********************************************************************
;;
;;  Telemetry Status Rules (3.2)
;;
;;
;;     No rules specified at this time pending further details
;;
;;
;; *********************************************************************
```

3.3 Runway Status

```
;;****************************************************************************
;;
;;;  GROUP  Landing Site Checks (3.3)
;;
;;      This group determines whether or not the correct runway is selected
;;      in the onboard systems, and determines what action is needed when the
;;      wrong runway is selected.
;;
;;;  CONTROL FACTS
;       (sub-phase  runway  ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;****************************************************************************


(deffacts  monitoring-runway-phases     ; These facts define the runway
                                        ;  sub-phases in the monitoring phase
        (first-sub-phase  runway  monitoring  check)
                                        ; There is only 1 sub-phase: "check"
)


(deffacts  initial-runway-facts         ; These facts represent assumptions
                                        ;  about the runways before any data
                                        ;  is received.
        (runway-status  pass  unknown)  ; Don't know if right rw in the pass
        (runway-status  bfs  unknown)   ; Don't know if right rw in the bfs
        (runway-status  ground  unknown)
                                        ; Don't know if right rw in the ground
)

;-----------------------------------------------------------------------------

(defrule  desired-runway-from-operator

;;      IF
;;              The operator entered the desired runway slot
;;              number
;;      THEN
;;              Conclude the desired runway has that slot
;;              number
;;      END

        (sub-phase  runway  check)
        ?x <-(runway  desired  ?)
        ?y <-(operator-input runway ?slot)
        =>
        (retract  ?x ?y)
        (assert (runway desired ?slot)))
;-----------------------------------------------------------------------------

(defrule  onboard-runway-correct

;;      IF
;;              For the available system
;;              The selected runway in an onboard system is the same as
;;                      the desired runway  AND
;;              The runway status of that system was previously unknown or no-go
;;      THEN
;;              Conclude that the runway status of the onboard system is go
;;              Notify the operator
;;      END

        (sub-phase  runway  check)
        (system-available ?sys)
        (runway  desired  ?slot)
        (runway  ?sys ?slot)
        ?x <- (runway-status  ?sys  ~go)
        =>
        (retract  ?x)
        (assert  (runway-status  ?sys  go))
        (assert  (status-light  runway  ?sys  go))
        (assert  (event  site  nominal  alt
                "The " ?sys " has the correct runway selected")))

;---------------------------------------------------------------------------

(defrule  onboard-runway-incorrect

;;      IF
;;              For the available systems
;;              The system runway (slot) is not the same as the
;;              desired runway (slot)
;;      THEN
;;              Notify operator that the system has selected the
;;              wrong runway.
;;              Recommend call to crew to select proper runway.
;;      END

        (sub-phase  runway  check)
        (system-available ?sys)
        (runway  desired  ?desired-slot)
        (runway  ?sys  ?actual-slot&~?desired-slot)
        (same-area  ?desired-slot  ?actual-slot)
        ?x <- (runway-status  ?sys  ?status)
        =>
        (if (neq ?status no-go)
            then
                (retract  ?x)
                (assert  (runway-status  ?sys  no-go)))
        (if  (> ?actual-slot ?desired-slot)
            then
                (bind ?item 3)
                (bind ?name "primary")
            else
                (bind ?item 4)
                (bind ?name "secondary"))
        (assert  (status-light  runway  ?sys  no-go))
        (assert  (recommend  site  ?sys  off-nominal  alt
                "Need to select the " ?name " runway in the " ?sys
                " by ITEM " ?item " on SPEC 50")))

;---------------------------------------------------------------------------

(defrule  onboard-area-incorrect

;;      IF
;;              For the available systems
;;             The selected runway in an onboard system is different from
;;                   the desired runway  AND
;;             The selected runway is not in the same area as the
;;                   desired runway
;;       THEN
;;             Notify the operator that the correct area must be selected
;;       END

         (sub-phase  runway  check)
         (system-available ?sys)
         (runway  desired  ?desired-slot)
         (runway  ?sys ?actual-slot&~?desired-slot)
         (not  (same-area  ?desired-slot  ?actual-slot))
         (same-area  ?desired-slot  ?other-slot)
         ?x <- (runway-status  ?sys  ?status)
         =>
         (if (neq ?status no-go)
             then
                 (retract  ?x)
                 (assert  (runway-status  ?sys  no-go)))
         (assert  (status-light  runway  ?sys  no-go))
         (if  (> ?desired-slot ?other-slot)
             then
                 (bind  ?area  (/ ?desired-slot 2))
                 (assert  (recommend  site  ?sys  off-nominal  alt
                       "Need to select runway " =(lookup-rw-name ?desired-slot)
                       " in the " ?sys " by ITEM 41 +" ?area
                       " followed by ITEM 4 on SPEC 50"))
             else
                 (bind  ?area  (/ (+ ?desired-slot 1) 2))
                 (assert  (recommend  site  ?sys  off-nominal  alt
                       "Need to select runway " =(lookup-rw-name ?desired-slot)
                       " in the " ?sys " by ITEM 41 +" ?area
                       " on SPEC 50"))))

;-----------------------------------------------------------------------------

(defrule ground-runway-incorrect

;;       IF
;;             The GND runway (name) is not the same as the desired
;;             runway (name)
;;       THEN
;;             Notify operator that the selected GND runway is
;;             in error.
;;             Recommend call to GDO to have trajectory change the
;;             GND runway
;;       END

         (sub-phase  runway  check)
         (runway  desired  ?desired-slot&~unknown)
         (runway  ground  ?actual-slot&~?desired-slot)
         ?x <- (runway-status  ground  ?status)
         =>
         (if (neq ?status no-go)
             then
                 (retract  ?x)
                 (assert  (runway-status  ground  no-go)))
         (assert  (status-light  runway  ground  no-go))
         (assert  (recommend  site  ground  off-nominal  alt
"GDO needs to select runway "
=(lookup-rw-name ?desired-slot))))
```

3.4 Inertial Measurement Units

```
;;**************************************************************************
;;
;;;  GROUP
;;      Inertial Measurement Units (3.4)
;;
;;      This group watches the IMUs for failures and determines
;;      the cause of those failures.  This group also determines
;;      which IMUs should be used at any given time.
;;
;;;  CONTROL FACTS
;       (sub-phase  imu  ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;**************************************************************************


;;;  FACTS

(deffacts monitoring-imu-phases                 ; Defines the sequence of
                                                ; sub-phases in the monitoring
                                                ; phase of the IMU section.
        (first-sub-phase  imu  monitoring  pass-availability)
                                                ; The first sub-phase is
                                                ; PASS availability.
        (next-sub-phase  imu  pass-availability  bfs-availability)
                                                ; After PASS availability comes
                                                ; BFS availability.
        (next-sub-phase  imu  bfs-availability  error-detection)
                                                ; After BFS availability comes
                                                ; error detection.
        (next-sub-phase  imu  error-detection  error-isolation)
                                                ; After error detection comes
                                                ; error isolation.
        (next-sub-phase  imu  error-isolation  error-magnitude)
                                                ; After error isolation comes
                                                ; error magnitude determination.
        (next-sub-phase  imu  error-magnitude  failure-prediction)
                                                ; After error magnitude comes
                                                ; failure prediction.
)                                               ; Failure prediction is the last
                                                ; IMU sub-phase in monitoring
                                                ; phase.

(deffacts analysis-imu-phases                   ; Defines sequence of sub-phase
                                                ; in the analysis phase of the
                                                ; IMU section.
        (first-sub-phase  imu  analysis  pass-recommendation)
                                                ; The first sub-phase is
                                                ; PASS recommendations.
        (next-sub-phase  imu  pass-recommendation  bfs-recommendation)
                                                ; After PASS recommendations
                                                ; comes BFS recommendations.
)                                               ; BFS recommendations is the
                                                ; last IMU sub-phase in the
                                                ; analysis phase.


(deffacts  initial-imu-facts            ; These facts represent assumptions
                                        ; about the IMUs before any data is
                                        ; received
        (imu-avail-output pass 1 avail) ; IMU 1 is available in the PASS
        (imu-avail-output pass 2 avail) ; IMU 2 is available in the PASS
        (imu-avail-output pass 3 avail) ; IMU 3 is available in the PASS
        (imu-avail-output bfs 1 avail)  ; IMU 1 is available in the BFS
        (imu-avail-output bfs 2 avail)  ; IMU 2 is available in the BFS
        (imu-avail-output bfs 3 avail)  ; IMU 3 is available in the BFS
        (good-imus 3)                   ; There are three good IMUs
        (prev-bfs-imu 0)                ; The BFS has been mid-value selecting
        (is-imu-valid 1 vel valid)      ; IMU 1 is producing valid velocity data
        (is-imu-valid 2 vel valid)      ; IMU 2 is producing valid velocity data
        (is-imu-valid 3 vel valid)      ; IMU 3 is producing valid velocity data
        (is-imu-valid 1 att valid)      ; IMU 1 is producing valid attitude data
        (is-imu-valid 2 att valid)      ; IMU 2 is producing valid attitude data
        (is-imu-valid 3 att valid)      ; IMU 3 is producing valid attitude data
        (is-imu-valid 1 acc invalid)    ; IMU 1 is producing valid ACC data
        (is-imu-valid 2 acc invalid)    ; IMU 2 is producing valid ACC data
        (is-imu-valid 3 acc invalid)    ; IMU 3 is producing valid ACC data
        (imu-quality 1 good)            ; IMU 1 has no problems
        (imu-quality 2 good)            ; IMU 2 has no problems
        (imu-quality 3 good)            ; IMU 3 has no problems
        (imu-vel 1 under)               ; IMU 1 velocity compared to other IMUs
                                        ; is within limits
        (imu-vel 2 under)               ; IMU 2 velocity compared to other IMUs
                                        ; is within limits
        (imu-vel 3 under)               ; IMU 3 velocity compared to other IMUs
                                        ; is within limits
        (imu-att 1 under)               ; IMU 1 attitude compared to other IMUs
                                        ; is within limits
        (imu-att 2 under)               ; IMU 2 attitude compared to other IMUs
                                        ; is within limits
        (imu-att 3 under)               ; IMU 3 attitude compared to other IMUs
                                        ; is within limits
        (imu-acc 1 under)               ; IMU 1 ACC data compared to other IMUs
                                        ; is within limits
        (imu-acc 2 under)               ; IMU 2 ACC data compared to other IMUs
                                        ; is within limits
        (imu-acc 3 under)               ; IMU 3 ACC data compared to other IMUs
                                        ; is within limits
        (imu-rm-prediction none)        ; IMU RM is not predicted to fail any
                                        ; current candidates.
        (initial-misalignment 1 unknown); The initial misalignment for IMU 1
                                        ; is unknown
        (initial-misalignment 2 unknown); The initial misalignment for IMU 2
                                        ; is unknown
        (initial-misalignment 3 unknown); The initial misalignment for IMU 3
                                        ; is unknown
)

;;*****************************************************************************
;;
;;;  GROUP
;;      PASS IMU Availability (3.4.1.1)
;;
;;      This group determines which IMUs are available in the PASS, and why
;;      the unavailable ones are unavailable.
;;
;;;  CONTROL FACTS
```

```
;       (sub-phase  imu  pass-availability)
;;
;;;  CONTAINING GROUP
;;      Inertial Measurement Units
;;
;;*****************************************************************************

(defrule imu-commfault-pass

;;      IF
;;              The PASS is engaged
;;              An IMU was not previously commfaulted in the PASS  AND
;;              The commfault flag for that IMU is on in the PASS
;;      THEN
;;              Notify operator that an IMU is commfaulted (unless the whole
;;                      string is commfaulted).
;;              Conclude the IMU is unavailable to the PASS due to
;;                      a commfault.
;;              Conclude no IMU RM prediction
;;      END

        (sub-phase  imu  pass-availability)
        (engaged-system pass)
        ?x <- (imu-avail-output  pass  ?imu  ~commfault)
        (imu-flag  pass  commfault  ?imu  on)
        (string-commfault  pass  ?imu  ?string-flag)
        ?y <- (imu-rm-prediction  $?)
        =>
        (if  (eq  ?string-flag  off)
            then
                (assert  (event pass-imu off-nominal alt
                        "Commfault IMU "  ?imu  " in PASS")))
        (retract ?x)
        (assert  (imu-avail-output  pass  ?imu  commfault))
        (retract ?y)
        (assert  (imu-rm-prediction  none)))

;------------------------------------------------------------------------

(defrule imu-comf-clear-pass-1

;;      IF
;;              The PASS is engaged
;;              An IMU has been unavailable to the PASS due to commfault
;;              The commfault flag for that IMU is off in the PASS
;;              The fail flag or deselect flag for that IMU is on in the PASS
;;      THEN
;;              Notify operator that the commfault has cleared
;;                      (unless it was a string commfault)
;;              Conclude the IMU is unavailable to the PASS due to failure
;;                      or deselect, whichever flag is on
;;              Conclude no IMU RM prediction
;;      END

        (sub-phase  imu  pass-availability)
        (engaged-system pass)
        ?x <- (imu-avail-output  pass  ?imu  commfault)
        (imu-flag  pass  commfault  ?imu  off)
        (imu-flag  pass  ?flag&fail|deselect  ?imu  on)
        (prev-string-cf  pass  ?imu  ?string-flag)
        ?y <- (imu-rm-prediction  $?)
        =>
        (if  (eq  ?string-flag  off)
            then
                (assert  (event pass-imu off-nominal alt
                        "Commfault clear on IMU " ?imu " in PASS")))
        (retract ?x)
        (assert  (imu-avail-output  pass  ?imu  ?flag))
        (retract ?y)
        (assert  (imu-rm-prediction  none)))


;-----------------------------------------------------------------------

(defrule imu-comf-clear-pass-2

;;      IF
;;              The PASS is engaged
;;              An IMU has been unavailable to the PASS due to commfault
;;              The commfault flag for that IMU is off in the PASS
;;              The fail flag for that IMU is off in the PASS
;;              The deselect flag for that IMU is off in the PASS
;;      THEN
;;              Notify operator that the commfault has cleared
;;                      (unless it was a string commfault)
;;              Conclude the IMU is now available to the PASS
;;              Conclude no IMU RM prediction
;;      END

        (sub-phase  imu  pass-availability)
        (engaged-system pass)
        ?x <- (imu-avail-output  pass  ?imu  commfault)
        (imu-flag  pass  commfault ?imu  off)
        (imu-flag  pass  fail  ?imu  off)
        (imu-flag  pass  deselect  ?imu  off)
        (prev-string-cf  pass  ?imu  ?string-flag)
        ?y <- (imu-rm-prediction  $?)
        =>
        (if  (eq  ?string-flag  off)
            then
                (assert  (event pass-imu off-nominal alt
                        "Commfault clear on IMU " ?imu " in PASS")))
        (retract ?x)
        (assert  (imu-avail-output  pass  ?imu  avail))
        (retract ?y)
        (assert  (imu-rm-prediction  none)))


;-----------------------------------------------------------------------

(defrule imu-failed-pass

;;      IF
;;              The PASS is engaged
;;              An IMU has been available to the PASS
;;              The fail flag for that IMU is on in the PASS
;;      THEN
;;              Notify operator of IMU failure
;;              Conclude the IMU is unavailable to the PASS due to failure
;;              Conclude no IMU RM prediction
;;       END

         (sub-phase  imu  pass-availability)
         (engaged-system pass)
         ?x <- (imu-avail-output  pass  ?imu  avail)
         (imu-flag  pass  fail  ?imu  on)
         ?y <- (imu-rm-prediction  $?)
         =>
         (assert  (event pass-imu off-nominal alt "RM failed IMU " ?imu))
         (retract ?x)
         (assert  (imu-avail-output  pass  ?imu  fail))
         (retract ?y)
         (assert  (imu-rm-prediction  none)))


;-------------------------------------------------------------------------

(defrule imu-deselected-pass

;;       IF
;;               The PASS is engaged
;;               An IMU has been available to the PASS
;;               The deselect flag for that IMU is on in the PASS
;;       THEN
;;               Notify operator of crew deselection
;;               Conclude the IMU is unavailable to the PASS due to deselect
;;               Conclude no IMU RM prediction
;;       END

         (sub-phase  imu  pass-availability)
         (engaged-system pass)
         ?x <- (imu-avail-output  pass  ?imu  avail)
         (imu-flag  pass  deselect  ?imu  on)
         ?y <- (imu-rm-prediction  $?)
         =>
         (assert  (event pass-imu off-nominal alt "Crew deselected IMU " ?imu))
         (retract ?x)
         (assert  (imu-avail-output  pass  ?imu  deselect))
         (retract ?y)
         (assert  (imu-rm-prediction  none)))


;-------------------------------------------------------------------------

(defrule imu-reselected-pass

;;       IF
;;               The PASS is engaged
;;               An IMU has been unavailable to the PASS due to failure
;;                       or deselect
;;               The fail flag for that IMU is off in the PASS
;;               The deselect flag for that IMU is off in the PASS
;;       THEN
;;               Notify operator of crew reselection.
;;               Conclude the IMU is now available to the PASS.
;;               Conclude no IMU RM prediction
;;       END

         (sub-phase  imu  pass-availability)
         (engaged-system pass)
```

```
        ?x <- (imu-avail-output  pass  ?imu  fail|deselect)
        (imu-flag  pass  fail  ?imu  off)
        (imu-flag  pass  deselect  ?imu  off)
        ?y <- (imu-rm-prediction  $?)
        =>
        (assert  (event pass-imu off-nominal alt "Crew reselected IMU " ?imu))
        (retract ?x)
        (assert  (imu-avail-output  pass  ?imu  avail))
        (retract ?y)
        (assert  (imu-rm-prediction  none)))


;-----------------------------------------------------------------------------

(defrule three-good-imus

;;      IF
;;              The PASS is engaged
;;              All 3 IMUs are not commfaulted in the PASS
;;              All 3 IMUs are good
;;      THEN
;;              Conclude that there are 3 good IMUs in the PASS
;;      END

        (sub-phase  imu  pass-availability)
        (engaged-system pass)
        ?x <- (good-imus  ~3)
        (imu-avail-output  pass  1  ~commfault)
        (imu-avail-output  pass  2  ~commfault)
        (imu-avail-output  pass  3  ~commfault)
        (imu-quality  1  good)
        (imu-quality  2  good)
        (imu-quality  3  good)
        =>
        (retract ?x)
        (assert  (good-imus  3)))


;-----------------------------------------------------------------------------

(defrule two-good-imus

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted in the PASS
;;              IMU A is good
;;              IMU B is not commfaulted in the PASS
;;              IMU B is good
;;              IMU C is commfaulted in the PASS or suspect
;;      THEN
;;              Conclude we have 2 good IMUs in the PASS
;;      END

        (sub-phase  imu  pass-availability)
        (engaged-system pass)
        ?x <- (good-imus  ~2)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (imu-quality  ?imu-a  good)
        (imu-avail-output  pass  ?imu-b&~?imu-a  ~commfault)
        (imu-quality  ?imu-b  good)
        (or  (imu-avail-output  pass  ~?imu-a&~?imu-b  commfault)
             (imu-quality  ~?imu-a&~?imu-b  ~good))
        =>
        (retract ?x)
        (assert  (good-imus  2)))


;---------------------------------------------------------------------------

(defrule one-good-imu

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted in the PASS
;;              IMU A is good
;;              IMU B is commfaulted in the PASS or suspect
;;              IMU C is commfaulted in the PASS or suspect
;;      THEN
;;              Conclude we have 1 good IMU in the PASS
;;      END

        (sub-phase  imu  pass-availability)
        (engaged-system pass)
        ?x <- (good-imus  ~1)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (imu-quality  ?imu-a  good)
        (or  (imu-avail-output  pass  ?imu-b&~?imu-a  commfault)
             (imu-quality  ?imu-b&~?imu-a  ~good))
        (or  (imu-avail-output  pass  ~?imu-a&~?imu-b  commfault)
             (imu-quality  ~?imu-a&~?imu-b  ~good))
        =>
        (retract ?x)
        (assert  (good-imus  1)))


;---------------------------------------------------------------------------

(defrule no-good-imus

;;      IF
;;              The PASS is engaged
;;              All 3 IMUs are commfaulted in the PASS or suspect
;;      THEN
;;              Conclude we have no good IMUs in the PASS
;;              Notify operator of no good IMU's in the PASS
;;      END

        (sub-phase  imu  pass-availability)
        (engaged-system pass)
        ?x <- (good-imus  ~0)
        (or  (imu-avail-output  pass  1  commfault)
             (imu-quality  1  ~good))
        (or  (imu-avail-output  pass  2  commfault)
             (imu-quality  2  ~good))
        (or  (imu-avail-output  pass  3  commfault)
             (imu-quality  3  ~good))
        =>
        (retract ?x)
        (assert  (good-imus  0))
        (assert  (event pass-imu off-nominal alt
                    "WARNING -- WE HAVE NO GOOD IMUS IN THE PASS")))

;;****************************************************************************
;;
;;; GROUP
;;       BFS IMU Availability (3.4.1.2)
;;
;;       This group determines which IMUs are available in the BFS.  It also
;;       determines why the unavailable IMUs are unavailable.
;;
;;; CONTROL FACTS
;        (sub-phase  imu  bfs-availability)
;;
;;; CONTAINING GROUP
;;       Inertial Measurement Units
;;
;;****************************************************************************

(defrule imu-commfault-bfs

;;       IF
;;               The BFS is available
;;               An IMU was not previously commfaulted in the BFS
;;               The commfault flag for that IMU is on in the BFS
;;       THEN
;;               Notify operator of IMU commfault (unless the whole string
;;                       is commfaulted).
;;               Conclude the IMU is not available to the BFS due to commfault.
;;       END

         (sub-phase  imu  bfs-availability)
         (system-available bfs)
         ?x <- (imu-avail-output  bfs  ?imu  ~commfault)
         (imu-flag  bfs  commfault  ?imu  on)
         (string-commfault  bfs  ?imu  ?string-flag)
         =>
         (if  (eq  ?string-flag  off)
             then
                 (assert  (event bfs-imu off-nominal alt
                         "Commfault IMU " ?imu " in the BFS")))
         (retract ?x)
         (assert  (imu-avail-output  bfs  ?imu  commfault)))

;-------------------------------------------------------------------------

(defrule imu-comf-clear-bfs-not-engaged

;;       IF
;;               The BFS is available
;;               The BFS is engaged
;;               An IMU was unavailable to the BFS due to commfault
;;               The commfault flag for that IMU is off in the BFS
;;       THEN
;;               Notify operator that commfault has been cleared (unless the
;;                       whole string was commfaulted).
;;               Conclude the IMU is available to the BFS (if the fail flag is
;;                       off) or unavailable due to failure (if the fail flag
;;                       is on).
;;       END

         (sub-phase  imu  bfs-availability)
         (system-available bfs)
         (engaged-system ~bfs)
         ?x <- (imu-avail-output  bfs  ?imu  commfault)
         (imu-flag  bfs  commfault  ?imu  off)
         (imu-flag  bfs  fail  ?imu  ?fail-flag)
         (prev-string-cf  bfs  ?imu  ?string-flag)
         =>
         (if  (eq  ?string-flag  off)
             then
                 (assert  (event bfs-imu off-nominal alt
                         "Commfault on IMU " ?imu " cleared in BFS")))
         (retract ?x)
         (if  (eq  ?fail-flag  off)
             then
                 (assert  (imu-avail-output  bfs  ?imu  avail))
             else
                 (assert  (imu-avail-output  bfs  ?imu  fail))))


;-----------------------------------------------------------------------

(defrule imu-comf-clear-bfs-engaged-part1

;;       IF
;;               The BFS is engaged
;;               An IMU has been unavailable to the BFS due to commfault
;;               The commfault flag for that IMU is off in the BFS
;;               The fail flag or deselect flag for that IMU is
;;                       on in the BFS
;;       THEN
;;               Notify operator that the commfault has cleared
;;                       (unless it was a string commfault)
;;               Conclude the IMU is unavailable to the BFS due to
;;                       failure or deselect, whichever flag is on

         (sub-phase imu bfs-availability)
         (engaged-system bfs)
         ?x <- (imu-avail-output bfs ?imu commfault)
         (imu-flag bfs commfault ?imu off)
         (imu-flag bfs ?flag&fail|deselect ?imu on)
         (prev-string-cf bfs ?imu ?string-flag)
         =>
         (if (eq ?string-flag off)
           then
                 (assert (event bfs-imu off-nominal alt
                  "Commfault on IMU " ?imu " cleared in BFS")))
         (retract ?x)
         (if (eq ?flag fail)
           then (assert (imu-avail-output bfs ?imu fail))
           else (assert (imu-avail-output bfs ?imu deselect))))

;-----------------------------------------------------------------------

(defrule imu-comf-clear-bfs-engaged-part2

;;       IF
;;               The BFS is engaged
;;              An IMU has been unavailable to the BFS due to commfault
;;              The commfault flag for that IMU is off in the BFS
;;              The fail flag for that IMU is off in the BFS
;;              The deselect flag for that IMU is off in the BFS
;;      THEN
;;              Notify the operator that the commfault has cleared
;;                      (unless it was a string commfault)
;;              Conclude the IMU is now available to the BFS

        (sub-phase imu bfs availability)
        (engaged-system  bfs)
        ?x <- (imu-avail-output  bfs  ?imu  commfault)
        (imu-flag  bfs  commfault  ?imu  off)
        (imu-flag  bfs  fail  ?imu  off)
        (imu-flag  bfs  deselect  ?imu  off)
        (prev-string-cf  bfs  ?imu  ?string-flag)
        =>
        (if (eq  ?string-flag  off)
          then (assert (event  bfs-imu  off-nominal  alt
                "Commfault on IMU " ?imu " cleared in BFS")))
        (retract ?x)
        (assert (imu-avail-output bfs ?imu avail)))

;-----------------------------------------------------------------------------

(defrule imu-failed-bfs

;;      IF
;;              BFS is available
;;              An IMU was available to the BFS
;;              The fail flag for that IMU is on in the BFS
;;      THEN
;;              Notify operator of IMU failure in the BFS
;;              Conclude the IMU is unavailable to the BFS due to failure
;;      END

        (sub-phase  imu  bfs-availability)
        (system-available  bfs)
        ?x <- (imu-avail-output  bfs  ?imu  avail)
        (imu-flag  bfs  fail  ?imu  on)
        =>
        (assert  (event bfs-imu off-nominal alt "IMU " ?imu " failed in BFS"))
        (retract ?x)
        (assert  (imu-avail-output  bfs  ?imu  fail)))

;-----------------------------------------------------------------------------

(defrule imu-deselected-bfs-1-not-engaged

;;      IF
;;              The BFS is not engaged
;;              The BFS is available
;;              The BFS was mid-value-selecting IMUs
;;              All IMU commfault flags are off in the BFS
;;              All IMU fail flags are off in the BFS
;;              The BFS is prime selecting an IMU
;;      THEN
;;              Notify operator that BFS has changed IMU status due to
;;                      a crew action.
;;              Notify the operator that BFS is now prime selecting an
;;                      IMU
;;      END

        (sub-phase  imu  bfs-availability)
        (engaged-system  ~bfs)
        (system-available  bfs)
        ?x <- (prev-bfs-imu_  0)
        (bfs-imu  ?new-imu&~0)
        (imu-flag  bfs  commfault  1  off)
        (imu-flag  bfs  commfault  2  off)
        (imu-flag  bfs  commfault  3  off)
        (imu-flag  bfs  fail  1  off)
        (imu-flag  bfs  fail  2  off)
        (imu-flag  bfs  fail  3  off)
        =>
        (assert  (event bfs-imu off-nominal alt
                        "Crew deslected an IMU in the BFS"))
        (assert  (event bfs-imu off-nominal alt "BFS is now on IMU " ?new-imu))
        (retract ?x)
        (assert  (prev-bfs-imu  ?new-imu)))

;-------------------------------------------------------------------------

(defrule imu-deselected-bfs-2-not-engaged

;;      IF
;;              The BFS is available
;;              The BFS is not engaged
;;              The BFS was prime selecting an IMU
;;              The commfault flag for that IMU is off in the BFS
;;              The fail flag for that IMU is off in the BFS
;;              The BFS is now prime selecting a different IMU
;;      THEN
;;              Notify operator the formerly selected IMU has been deselected.
;;              Notify operator that BFS is now prime selecting a different
;;                      IMU
;;      END

        (sub-phase  imu  bfs-availability)
        (engaged-system  ~bfs)
        (system-available bfs)
        ?x <- (prev-bfs-imu_  ?imu&~0)
        (bfs-imu  ?new-imu&~?imu)
        (imu-flag  bfs  commfault  ?imu  off)
        (imu-flag  bfs  fail  ?imu  off)
        =>
        (assert  (event bfs-imu off-nominal alt
                "Crew deselected IMU " ?imu " in the BFS"))
        (assert  (event bfs-imu off-nominal alt "BFS is now on IMU " ?new-imu))
        (retract ?x)
        (assert  (prev-bfs-imu  ?new-imu)))


;-------------------------------------------------------------------------

(defrule imu-deselected-bfs-engaged

;;      IF
;;              The BFS is available
;;              The BFS is engaged
;;              An IMU has been available to the BFS
;;              The deselect flag for that IMU is on in the BFS
;;      THEN
;;              Notify operator of crew deselection in the BFS
;;              Conclude the IMU is unavailable to the BFS
;;                      due to deselection

        (sub-phase imu bfs-availability)
        (system-available  bfs)
        (engaged-system  bfs)
        ?x <- (imu-avail-output  bfs  ?imu  avail)
        (imu-flag  bfs  deselect  ?imu  on)
        =>
        (assert (event  bfs-imu  off-nominal  alt
                 "Crew deselected IMU " ?imu " in the BFS"))
        (retract ?x)
        (assert (imu-avail-output  bfs  ?imu  deselect)))

;------------------------------------------------------------------------------

(defrule imu-reselect-bfs-engaged

;;      IF
;;              The BFS is engaged
;;              An IMU has been unavailable to the BFS due to
;;                      failure or deselect
;;              The fail flag for that IMU is off in the BFS
;;              The deselect flag for that IMU is off in the BFS
;;      THEN
;;              Notify operator of crew reselection
;;              Conclude the IMU is now available to the BFS


        (sub-phase  imu  bfs-availability)
        (engaged-system  bfs)
        ?x <- (imu-avail-output  bfs  ?imu  fail|deselect)
        (imu-flag  bfs  fail  ?imu  off)
        (imu-flag  bfs  deselect  ?imu  off)
        =>
        (assert (event  bfs-imu  off-nominal  alt
                 "Crew reselected IMU " ?imu " in the BFS"))
        (retract ?x)
        (assert (imu-avail-output  bfs  ?imu  avail)))

;------------------------------------------------------------------------------

(defrule imu-change-bfs

;;      IF
;;              The BFS is available
;;              The fail flag or commfault flag for an IMU is on in the BFS
;;              That IMU was the prime selected IMU or the BFS was
;;                      mid-value selecting
;;      THEN
;;              Notify operator of a change in BFS IMU status due to
;;                      commfault or failure.
;;      END

        (sub-phase  imu  bfs-availability)
        (system-available bfs)
        ?x <- (prev-bfs-imu  ?imu-a)
        (bfs-imu  ?new-imu&~?imu-a)
        (imu-flag  bfs  commfault|fail  ?imu-b  on)
        (test (||  (=  ?imu-a  0)
                   (=  ?imu-a  ?imu-b)))
        =>
        (assert  (event bfs-imu off-nominal alt "BFS is now on IMU " ?new-imu))
        (retract ?x)
        (assert  (prev-bfs-imu  ?new-imu)))


;;*****************************************************************************
;;
;;;  GROUP
;;      Error Detection (3.4.2.1)
;;
;;      This group determines when an IMU error exists.
;;
;;;  CONTROL FACTS
;       (sub-phase  imu  error-detection)
;;
;;;  CONTAINING GROUP
;;      Inertial Measurement Units
;;
;;*****************************************************************************

(defrule valid-velocity

;;      IF
;;              The PASS is engaged
;;              An IMU is not commfaulted
;;              That IMU is good or is suspect due to drift
;;      THEN
;;              Conclude that velocity comparisons with that IMU are valid.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (is-imu-valid  ?imu  vel  ~invalid)
        (imu-avail-output  pass  ?imu  ~commfault)
        (imu-quality  ?imu  good|drift)
        =>
        (retract ?x)
        (assert  (is-imu-valid  ?imu  vel  valid)))


;---------------------------------------------------------------------------

(defrule invalid-velocity

;;      IF
;;              The PASS is engaged
;;              An IMU is commfaulted or is suspect due to anything but drift
;;      THEN
;;              Conclude that velocity comparisons with that IMU are invalid.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (is-imu-valid  ?imu  vel  valid)
        (or  (imu-avail-output  pass ?imu  commfault)
             (imu-quality  ?imu  ~good&~drift))
        =>
        (retract ?x)
        (assert  (is-imu-valid  ?imu  vel  invalid)))


;----------------------------------------------------------------------

(defrule valid-attitude

;;      IF
;;              The PASS is engaged
;;              An IMU is not commfaulted
;;              That IMU is good or is suspect due to accelerometer bias
;;      THEN
;;              Conclude that attitude comparisons with that IMU are valid.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (is-imu-valid  ?imu  att  ~invalid)
        (imu-avail-output  pass  ?imu  ~commfault)
        (imu-quality  ?imu  good|bias)
        =>
        (retract ?x)
        (assert  (is-imu-valid  ?imu  att  valid)))


;----------------------------------------------------------------------

(defrule invalid-attitude

;;      IF
;;              The PASS is engaged
;;              An IMU is commfaulted or is suspect due to anything but bias
;;      THEN
;;              Conclude that attitude comparisons with that IMU are invalid.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (is-imu-valid  ?imu  att  valid)
        (or  (imu-avail-output  pass ?imu  commfault)
             (imu-quality  ?imu  ~good&~bias))
        =>
        (retract ?x)
        (assert  (is-imu-valid  ?imu  att  invalid)))


;----------------------------------------------------------------------

(defrule valid-to-use-acc-comparison

;;      IF
;;              The PASS is engaged
;;              The ACC delta-t > 30 seconds
;;      THEN
;;                Valid to use ACC comparison
;;
        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        (acc-delta-time ?t&:(> ?t 30.0))
        ?x1 <- (is-imu-valid 1 acc invalid)
        ?x2 <- (is-imu-valid 2 acc invalid)
        ?x3 <- (is-imu-valid 3 acc invalid)
=>
        (retract ?x1 ?x2 ?x3)
        (assert (is-imu-valid 1 acc valid))
        (assert (is-imu-valid 2 acc valid))
        (assert (is-imu-valid 3 acc valid)))
;------------------------------------------------------------------------------

(defrule valid-acc

;;      IF
;;              The PASS is engaged
;;              An IMU is not commfaulted
;;              That IMU is good or is suspect due to resolver
;;      THEN
;;              Conclude that ACC comparisons with that IMU are valid.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (is-imu-valid  ?imu  acc  invalid)
        (imu-avail-output  pass  ?imu  ~commfault)
        (imu-quality  ?imu  good|resolver)
        =>
        (retract ?x)
        (assert  (is-imu-valid  ?imu  acc  valid)))


;------------------------------------------------------------------------------

(defrule invalid-acc

;;      IF
;;              The PASS is engaged
;;              An IMU is commfaulted or is suspect due to anything but resolver
;;      THEN
;;              Conclude that ACC comparisons with that IMU are invalid.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (is-imu-valid  ?imu  acc  valid)
        (or  (imu-avail-output  pass ?imu  commfault)
             (imu-quality  ?imu  ~good&~resolver))
        =>
        (retract ?x)
        (assert  (is-imu-valid  ?imu  acc  invalid)))

;;*****************************************************************************
;;
;;  ERROR DETECTION - Velocity Comparisons
;;
```

```
(defrule velocity-comparison-1

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted
;;              IMU B velocity is valid
;;              Velocity comparison A-B is different from IMU A's earlier
;;                      velocity comparison status
;;              IMU C velocity is invalid
;;      THEN
;;              Change IMU A's velocity comparison status to current A-B
;;                      comparison status.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (imu-vel  ?imu-a  ?status)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (lrus-in-pair  ?pair-1  ?imu-a  ?imu-b)
        (is-imu-valid  ?imu-b  vel  valid)
        (rel-imu-comp  ?pair-1  vel  ?status-1&~?status)
        (lrus-in-pair  ?pair-2&~?pair-1  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-c  vel  invalid)
        =>
        (retract ?x)
        (assert  (imu-vel  ?imu-a  ?status-1)))


;---------------------------------------------------------------------------

(defrule velocity-comparison-2

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted
;;              IMU B velocity is valid
;;              Velocity comparison A-B is some status (call it status-1)
;;              IMU C velocity is valid
;;              Velocity comparison A-C is some status (call it status-2)
;;              The smaller of status-1 and status-2 is different from
;;                      IMU A's earlier velocity comparison status
;;      THEN
;;              Change IMU A's velocity comparison status to the smaller
;;                      of status-1 and status-2.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (imu-vel  ?imu-a  ?status)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (lrus-in-pair  ?pair-1  ?imu-a  ?imu-b)
        (is-imu-valid  ?imu-b  vel  valid)
        (rel-imu-comp  ?pair-1  vel  ?status-1)
        (lrus-in-pair  ?pair-2&~?pair-1  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-c  vel  valid)
        (rel-imu-comp  ?pair-2  vel  ?status-2)
        (min-miscompare  ?status-1  ?status-2  ?new-status&~?status)
        =>
        (retract ?x)
        (assert  (imu-vel  ?imu-a  ?new-status)))

;;*************************************************************************
;;
;;  ERROR DETECTION - Attitude Comparisons
;;

(defrule attitude-comparison-1

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted
;;              IMU B attitude is valid
;;              Attitude comparison A-B is different from IMU A's earlier
;;                      attitude comparison status
;;              IMU C attitude is invalid
;;      THEN
;;              Change IMU A's attitude comparison status to current A-B
;;                      comparison status.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (imu-att  ?imu-a  ?status)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (lrus-in-pair  ?pair-1  ?imu-a  ?imu-b)
        (is-imu-valid  ?imu-b  att  valid)
        (rel-imu-comp  ?pair-1  att  ?status-1&~?status)
        (lrus-in-pair  ?pair-2&~?pair-1  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-c  att  invalid)
        =>
        (retract ?x)
        (assert  (imu-att  ?imu-a  ?status-1)))


;-------------------------------------------------------------------------

(defrule attitude-comparison-2

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted
;;              IMU B attitude is valid
;;              Attitude comparison A-B is some status (call it status-1)
;;              IMU C attitude is valid
;;              Attitude comparison A-C is some status (call it status-2)
;;              The smaller of status-1 and status-2 is different from
;;                      IMU A's earlier attitude comparison status
;;      THEN
;;              Change IMU A's attitude comparison status to the smaller of
;;                      status-1 and status-2
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (imu-att  ?imu-a  ?status)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (lrus-in-pair  ?pair-1  ?imu-a  ?imu-b)
        (is-imu-valid  ?imu-b  att  valid)
        (rel-imu-comp  ?pair-1  att  ?status-1)
        (lrus-in-pair  ?pair-2&~?pair-1  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-c  att  valid)
        (rel-imu-comp  ?pair-2  att  ?status-2)
        (min-miscompare  ?status-1  ?status-2  ?new-status&~?status)
        =>
        (retract ?x)
        (assert  (imu-att  ?imu-a  ?new-status)))


;;**************************************************************************
;;
;;  ERROR DETECTION - ACC Comparisons
;;

(defrule acc-comparison-1

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted
;;              IMU B ACC is valid
;;              Worst axis ACC comparison A-B is different from IMU A's
;;                      earlier ACC comparison status
;;              IMU C ACC is invalid
;;      THEN
;;              Change IMU A's ACC comparison status to current A-B
;;                      comparison status.
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (imu-acc  ?imu-a  ?status)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (lrus-in-pair  ?pair-1  ?imu-a  ?imu-b)
        (is-imu-valid  ?imu-b  acc  valid)
        (rel-imu-acc  ?pair-1  worst-axis  ?status-1&~?status)
        (lrus-in-pair  ?pair-2&~?pair-1  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-c  acc  invalid)
        =>
        (retract ?x)
        (assert  (imu-acc  ?imu-a  ?status-1)))


;--------------------------------------------------------------------------

(defrule acc-comparison-2

;;      IF
;;              The PASS is engaged
;;              IMU A is not commfaulted
;;              IMU B ACC is valid
;;              Worst axis ACC comparison A-B is some status (call it
;;                      status-1)
;;              IMU C ACC is valid
;;              Worst axis ACC comparison A-C is some status (call it
;;                      status-2)
;;              The smaller of status-1 and status-2 is different from
;;                      IMU A's earlier ACC comparison status
;;      THEN
;;              Change IMU A's ACC comparison status to the smaller of status-1
;;                      and status-2
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        ?x <- (imu-acc  ?imu-a  ?status)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (lrus-in-pair  ?pair-1  ?imu-a  ?imu-b)
        (is-imu-valid  ?imu-b  acc  valid)
        (rel-imu-acc  ?pair-1  worst-axis  ?status-1)
        (lrus-in-pair  ?pair-2&~?pair-1  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-c  acc  valid)
        (rel-imu-acc  ?pair-2  worst-axis  ?status-2)
        (min-miscompare  ?status-1  ?status-2  ?new-status&~?status)
        =>
        (retract ?x)
        (assert  (imu-acc  ?imu-a  ?new-status)))


;---------------------------------------------------------------------------

(defrule worst-comparison

;;      IF
;;              The PASS is engaged
;;              Exactly 2 good IMUs are available
;;              Those 2 IMUs disagree in any way
;;      THEN
;;              Conclude that 2-level isolation must be used to determine
;;                      which of the 2 IMUs has a problem
;;      END

        (sub-phase  imu  error-detection)
        (engaged-system  pass)
        (good-imus  2)
        (imu-avail-output  pass  ?imu-a  ~commfault)
        (imu-avail-output  pass  ?imu-b&~?imu-a  ~commfault)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (imu-quality  ?imu-a  good)
        (imu-quality  ?imu-b  good)
        (rel-imu-comp  ?pair  vel  ?s1)
        (rel-imu-comp  ?pair  att  ?s2)
        (max-miscompare  ?s1  ?s2  ?s3)
        (rel-imu-acc  ?pair  worst-axis  ?s4)
        (max-miscompare  ?s3  ?s4  ~under)
        =>
        (assert  (isolate  ?pair)))


;;****************************************************************************
;;
;;;  GROUP
;;      Error Isolation (3.4.2.2)
;;
;;      When an IMU error has been detected, this group determines which IMU
;;      has the problem, and what the problem is.
;;
;;;  CONTROL FACTS
;       (sub-phase  IMU  error-isolation)
;;
```

```
;;;  CONTAINING GROUP
;;      Inertial Measurement Units
;;
;;*****************************************************************************
;;**************************************************************************
;;
;;  ERROR ISOLATION - 3 level
;;

(defrule three-level-component-isolation

;;       IF
;;               The PASS is engaged
;;               There are 3 good IMUs
;;               An IMU disagrees with the other 2 IMUs
;;       THEN
;;               Use the fault matrix to determine the problem with the IMU
;;               Notify operator of an IMU problem
;;       END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (good-imus  3)
        (imu-vel  ?imu  ?vel)
        (imu-att  ?imu  ?att)
        (imu-acc  ?imu  ?acc)
        (fault-matrix  ?vel  ?att  ?acc  ?fault)
        ?x <- (imu-quality  ?imu  ?fault)
        =>
        (if  (eq  ?fault  suspect)
            then
                (assert  (event pass-imu off-nominal alt
                        "IMU " ?imu " has an undiagnosable problem"))
            else
                (if  (eq  ?fault  good)
                    then
                        (assert  (event pass-imu off-nominal alt
                                "IMU " ?imu " is good"))
                    else
                        (assert  (event pass-imu off-nominal alt
                                "IMU " ?imu " has a " ?fault " error"))))
        (retract ?x)
        (assert  (imu-quality  ?imu  ?fault)))


;;**************************************************************************
;;
;;  ERROR ISOLATION - 2 level
;;

(defrule two-level-gnd-comparison

;;       IF
;;               The PASS is engaged
;;               HSTD is good
;;               An error between IMUs A and B has been detected at the 2
;;                       level
;;               Worst axis GND-IMUA comparison is some status (call it
;;                       status-a)
;;               Worst axis GND-IMUB comparison is some status (call it
;;                      status-b)
;;              GND-IMU comparison has not yet voted
;;      THEN
;;              When status-a = status-b, vote 0 for both IMUs.
;;              Otherwise, vote 1 for the IMU with the larger difference, and
;;                      0 for the other IMU.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (hstd  good)
        (isolate  ?pair)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (gnd-imu  ?imu-a  worst-axis  ?status-a)
        (gnd-imu  ?imu-b  worst-axis  ?status-b)
        (not  (imu-vote  gnd  $?))
        =>
        (bind  ?vote-a  0)
        (bind  ?vote-b  0)
        (if  (neq  ?status-a  ?status-b)
            then
                (if  (neq  ?status-a  under)
                    then
                        (bind  ?vote-a  1)
                    else
                        (bind  ?vote-b  1)))
        (assert  (imu-vote  gnd  ?vote-a  ?imu-a))
        (assert  (imu-vote  gnd  ?vote-b  ?imu-b)))


;---------------------------------------------------------------------------

(defrule two-level-gnd-cant-vote

;;      IF
;;              The PASS is engaged
;;              An error between IMUs A and B has been detected at the 2
;;                      level
;;              The HSTD is not good
;;              GND-IMU comparison has not voted yet
;;      THEN
;;              Vote 0 for IMUs A and B
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (isolate  ?pair)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (hstd  ~good)
        (not  (imu-vote  gnd  $?))
        =>
        (assert  (imu-vote  gnd  0  ?imu-a))
        (assert  (imu-vote  gnd  0  ?imu-b)))


;---------------------------------------------------------------------------

(defrule two-level-state-comparison

;;      IF
;;             The PASS is engaged
;;             The HSTD is good
;;             3-state nav is active
;;             An error between IMUs A and B has been detected at the 2
;;                     level
;;             Worst axis GND-state-A comparison is some status
;;                     (call it status-a)
;;             Worst axis GND-state-B comparison is some status
;;                     (call it status-b)
;;             GND-state comparison has not voted yet
;;       THEN
;;             When status-a = status-b, vote 0 for both IMUs.
;;             Otherwise, vote 2 for the IMU with the larger difference, and
;;                     0 for the other IMU.
;;       END

         (sub-phase  imu  error-isolation)
         (engaged-system  pass)
         (hstd  good)
         (nav-3-state  on)
         (isolate  ?pair)
         (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
         (gnd-3state  ?imu-a  worst-axis  ?status-a)
         (gnd-3state  ?imu-b  worst-axis  ?status-b)
         (not  (imu-vote  state  $?))
         =>
         (bind  ?vote-a  0)
         (bind  ?vote-b  0)
         (if  (neq  ?status-a  ?status-b)
            then
               (if  (neq  ?status-a  under)
                  then
                     (bind  ?vote-a  2)
                  else
                     (bind  ?vote-b  2)))
         (assert  (imu-vote  state  ?vote-a  ?imu-a))
         (assert  (imu-vote  state  ?vote-b  ?imu-b)))


;-----------------------------------------------------------------------

(defrule two-level-state-cant-vote

;;       IF
;;             The PASS is engaged
;;             An error between IMUs A and B has been detected at the 2
;;                     level
;;             The HSTD is not good  OR  3-state nav is inactive
;;             GND-state comparison has not voted yet
;;       THEN
;;             Vote 0 for IMUs A and B
;;       END

         (sub-phase  imu  error-isolation)
         (engaged-system  pass)
         (isolate  ?pair)
         (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
         (or  (hstd  ~good)
              (nav-3-state  off))
         (not  (imu-vote  state  $?))
```

```
        =>
        (assert  (imu-vote  state  0  ?imu-a))
        (assert  (imu-vote  state  0  ?imu-b)))


;--------------------------------------------------------------------------

(defrule two-level-acc-comparison

;;      IF
;;              The PASS is engaged
;;              An error between IMUs A and B has been detected at the 2
;;                      level
;;              IMU A is the reference for ACC comparisons
;;              X-axis ACC comparions A-B is some status (call it status-x)  AND
;;              Y-axis ACC comparions A-B is some status (call it status-y)  AND
;;              Z-axis ACC comparions A-B is some status (call it status-z)  AND
;;              ACC comparison has not voted yet
;;      THEN
;;              If status-x, status-y, and status-z indicate the error lies
;;                      in the x-y plane or z-axis of IMU A, vote 1 for
;;                      IMU A; otherwise, vote 0 for IMU A.
;;              Vote 0 for IMU B.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (isolate  ?pair)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (ref-imu-acc  ?imu-a)
        (rel-imu-acc  ?pair  x  ?status-x)
        (rel-imu-acc  ?pair  y  ?status-y)
        (rel-imu-acc  ?pair  z  ?status-z)
        (not  (imu-vote  acc  $?))
        =>
        (bind  ?vote-a  0)
        (if  (neq (|| (neq  ?status-x  under) (neq  ?status-y  under))
                  (neq  ?status-z  under))
            then
                (bind  ?vote-a  1))
        (assert (imu-vote acc ?vote-a  ?imu-a))
        (assert (imu-vote acc 0 ?imu-b)))


;--------------------------------------------------------------------------

(defrule two-level-acc-cant-vote

;;      IF
;;              The PASS is engaged
;;              An error between IMUs A and B has been detected at the 2
;;                      level
;;              Neither A nor B is the ACC reference IMU
;;              Acc comparison has not voted yet
;;      THEN
;;              Vote 0 for both IMUs A and B.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (isolate  ?pair)
        (excluded-lru  ?pair  ?imu-c)
        (ref-imu-acc  ?imu-c)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (not  (imu-vote  acc  $?))
        =>
        (assert  (imu-vote  acc  0  ?imu-a))
        (assert  (imu-vote  acc  0  ?imu-b)))


;------------------------------------------------------------------------

(defrule partial-imu-velocity

;;      IF
;;              The PASS is engaged
;;              An error between IMUs A and B has been detected at the 2
;;                      level
;;              IMU C velocity is valid
;;              IMU A's velocity comparisons with IMUs B and C is some
;;                      status (call it status-a)
;;              IMU B's velocity comparisons with IMUs A and C is some
;;                      status (call it status-b)
;;              Partial IMU velocity comparison has not voted yet
;;      THEN
;;              When status-a = status-b, vote 0 for both IMUs A and B.
;;              Otherwise, vote 1 for the IMU with the larger difference, and
;;                      0 for the other IMU.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (isolate  ?pair)
        (excluded-lru  ?pair  ?imu-c)
        (is-imu-valid  ?imu-c  vel  valid)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (imu-vel  ?imu-a  ?status-a)
        (imu-vel  ?imu-b  ?status-b)
        (not  (imu-vote partial-imu-vel  $?))
        =>
        (bind  ?vote-a  0)
        (bind  ?vote-b  0)
        (if (neq  ?status-a  ?status-b)
            then
                (if (neq  ?status-a  under)
                    then
                        (bind  ?vote-a  1)
                    else
                        (bind  ?vote-b  1)))
        (assert  (imu-vote  partial-imu-vel  ?vote-a  ?imu-a))
        (assert  (imu-vote  partial-imu-vel  ?vote-b  ?imu-b)))


;------------------------------------------------------------------------

(defrule partial-imu-attitude

;;      IF
;;              The PASS is engaged
;;              An error between IMUs A and B has been detected at the 2
;;                     level
;;             IMU C attitude is valid
;;             IMU A's attitude comparisons with IMUs B and C is some
;;                     status (call it status-a)
;;             IMU B's attitude comparisons with IMUs A and C is some
;;                     status (call it status-b)
;;             Partial IMU attitude comparison has not voted yet
;;      THEN
;;             When status-a = status-b, vote 0 for both IMUs A and B.
;;             Otherwise, vote 1 for the IMU with the larger difference, and
;;                     0 for the other IMU.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (isolate  ?pair)
        (excluded-lru  ?pair  ?imu-c)
        (is-imu-valid  ?imu-c  att  valid)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (imu-att  ?imu-a  ?status-a)
        (imu-att  ?imu-b  ?status-b)
        (not  (imu-vote partial-imu-att  $?))
        =>
        (bind  ?vote-a  0)
        (bind  ?vote-b  0)
        (if  (neq  ?status-a  ?status-b)
            then
                (if (neq  ?status-a  under)
                    then
                        (bind  ?vote-a  1)
                    else
                        (bind  ?vote-b  1)))
        (assert  (imu-vote  partial-imu-att  ?vote-a  ?imu-a))
        (assert  (imu-vote  partial-imu-att  ?vote-b  ?imu-b)))


;-----------------------------------------------------------------------

(defrule partial-imu-acc

;;      IF
;;             The PASS is engaged
;;             An error between IMUs A and B has been detected at the 2
;;                     level
;;             IMU C ACC is valid
;;             IMU A's ACC comparisons with IMUs B and C is some
;;                     status (call it status-a)
;;             IMU B's ACC comparisons with IMUs A and C is some
;;                     status (call it status-b)
;;             Partial IMU acceleration comparison has not voted yet
;;      THEN
;;             When status-a = status-b, vote 0 for both IMUs.
;;             Otherwise, vote 1 for the IMU with the larger difference, and
;;                     0 for the other IMU.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (isolate  ?pair)
        (excluded-lru  ?pair  ?imu-c)
        (is-imu-valid  ?imu-c  acc  valid)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (imu-acc  ?imu-a  ?status-a)
        (imu-acc  ?imu-b  ?status-b)
        (not  (imu-vote partial-imu-acc  $?))
        =>
        (bind  ?vote-a  0)
        (bind  ?vote-b  0)
        (if  (neq  ?status-a  ?status-b)
            then
                (if  (neq  ?status-a  under)
                    then
                        (bind  ?vote-a  1)
                    else
                        (bind  ?vote-b  1)))
        (assert  (imu-vote  partial-imu-acc  ?vote-a  ?imu-a))
        (assert  (imu-vote  partial-imu-acc  ?vote-b  ?imu-b)))


;-------------------------------------------------------------------------

(defrule partial-imu-cant-vote

;;      IF
;;              The PASS is engaged
;;              An error between IMUs A and B has been detected at the 2
;;                      level
;;              IMU C is invalid in velocity, attitude, and ACC  AND
;;              Partial IMU comparison has not voted yet
;;      THEN
;;              Vote 0 for IMUs A and B.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (isolate  ?pair)
        (excluded-lru  ?pair  ?imu-c)
        (is-imu-valid  ?imu-c  vel  invalid)
        (is-imu-valid  ?imu-c  att  invalid)
        (is-imu-valid  ?imu-c  acc  invalid)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (not  (imu-vote  partial-imu  $?))
        =>
        (assert  (imu-vote  partial-imu  0  ?imu-a))
        (assert  (imu-vote  partial-imu  0  ?imu-b)))


;-------------------------------------------------------------------------

(defrule two-level-vote-count

;;      IF
;;              The PASS is engaged
;;              GND-IMU comparison rules have cast v1 votes for an IMU  AND
;;              GND-state comparison rules have cast v2 votes for that IMU  AND
;;              ACC comparison rules have cast v3 votes for that IMU  AND
;;              Partial IMU vel comparison rules have cast v4 votes for that IMU
;;              Partial IMU att comparison rules have cast v5 votes for that IMU
;;              Partial IMU acc comparison rules have cast v6 votes for that IMU
;;      THEN
;;              Compute vote total for the IMU as v1+v2+v3+v4+v5+v6.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (imu-vote  gnd  ?v1  ?imu)
        (imu-vote  state  ?v2  ?imu)
        (imu-vote  acc  ?v3  ?imu)
        (imu-vote  partial-imu-vel  ?v4  ?imu)
        (imu-vote  partial-imu-att  ?v5  ?imu)
        (imu-vote  partial-imu-acc  ?v6  ?imu)
        =>
        (bind  ?total  (+  ?v1  ?v2  ?v3  ?v4 ?v5 ?v6))
        (assert  (imu-vote  total  ?total  ?imu)))


;-----------------------------------------------------------------------

(defrule two-level-imu-isolation

;;      IF
;;              The PASS is engaged
;;              Votes for IMU A exceeded votes for IMU B by 2 or more
;;      THEN
;;              Conclude IMU A has an error.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (imu-vote  total  ?vote-a  ?imu-a)
        (imu-vote  total  ?vote-b  ?imu-b&~?imu-a)
        (test  (>=  (- ?vote-a ?vote-b) 2))
        ?x <- (imu-quality  ?imu-a  $?)
        =>
        (retract ?x)
        (assert  (imu-quality  ?imu-a  suspect)))


;-----------------------------------------------------------------------

(defrule two-level-component-isolation

;;      IF
;;              The PASS is engaged
;;              An error between IMUs A and B has been detected at the 2
;;                      level
;;              IMU A is the one with the problem
;;      THEN
;;              Use the fault matrix to determine the problem with IMU A.
;;              Notify operator of the problem.
;;              Clear the miscompare indications for IMU B.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        ?y <-  (isolate  ?pair)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        ?x <- (imu-quality  ?imu-a  suspect)
        (imu-vel  ?imu-a  ?vel)
        (imu-att  ?imu-a  ?att)
        (imu-acc  ?imu-a  ?acc)
        (fault-matrix  ?vel  ?att  ?acc  ?fault)
        ?f1 <- (imu-vel  ?imu-b  ?vel)
        ?f2 <- (imu-att  ?imu-b  ?att)
        ?f3 <- (imu-acc  ?imu-b  ?acc)
        =>
        (if  (eq  ?fault  suspect)
            then
                (assert  (event pass-imu off-nominal alt
                        "IMU " ?imu-a " has an undiagnosable problem"))
            else
                (if  (eq  ?fault  good)
                    then
                        (assert  (event pass-imu off-nominal alt
                                "IMU " ?imu-a " is good"))
                    else
                        (assert  (event pass-imu off-nominal alt
                                "IMU " ?imu-a " has a " ?fault " error"))))
        (retract ?x)
        (assert  (imu-quality  ?imu-a  ?fault))
        (retract ?f1)
        (assert  (imu-vel  ?imu-b  under))
        (retract ?f2)
        (assert  (imu-att  ?imu-b  under))
        (retract ?f3)
        (assert  (imu-acc  ?imu-b  under))
        (retract ?y))


;--------------------------------------------------------------------------

(defrule two-level-cant-isolate

;;      IF
;;              The PASS is engaged
;;              Votes for IMU A did not exceed votes for IMU B by 2 or more
;;              Votes for IMU B did not exceed votes for IMU A by 2 or more
;;      THEN
;;              Notify operator that the IMU error cannot be isolated.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        ?x <- (isolate  ?pair)
        (imu-vote  total  ?vote-a  ?imu-a)
        (imu-vote  total  ?vote-b  ?imu-b&~?imu-a)
        (test  (<  (- ?vote-a ?vote-b)  2))
        (test  (<  (- ?vote-b ?vote-a)  2))
        =>
        (assert  (event pass-imu off-nominal alt
                "Cannot isolate problem to IMU " ?imu-a " or " ?imu-b))
        (retract ?x))


;--------------------------------------------------------------------------

(defrule two-level-vote-cleanup
        (sub-phase  imu  error-isolation)
        (not  (isolate $?))
        ?x  <-  (imu-vote  $?)
        =>
        (retract  ?x))

;;-----------------------------------------------------------------

(defrule change-imu-quality

;;      IF
;;              The PASS is engaged
;;              An IMU was previously diagnosed as having a problem
;;              That IMU's comparisons now indicate a different diagnosis
;;              The new indicated diagnosis is a bias, resolver, or drift,
;;                      or no problem at all
;;      THEN
;;              Update the IMU's quality rating to reflect the new diagnosis.
;;              Notify the operator of the new diagnosis.
;;      END

        (sub-phase  imu  error-isolation)
        (engaged-system  pass)
        (good-imus  ~3)
        (not  (isolate $?))
        ?x <- (imu-quality  ?imu  ?quality)
        (imu-vel  ?imu  ?vel)
        (imu-att  ?imu  ?att)
        (imu-acc  ?imu  ?acc)
        (fault-matrix  ?vel  ?att  ?acc  ?fault&~?quality)
        (test  (||  (eq  ?fault  bias)
                    (eq  ?fault  resolver)
                    (eq  ?fault  drift)
                    (eq  ?fault  good)))
        =>
        (if  (eq  ?fault  good)
            then
                (assert  (event pass-imu nominal alt
                        "IMU " ?imu " is good"))
            else
                (assert  (event pass-imu off-nominal alt
                        "IMU " ?imu " has a " ?fault " error")))
        (retract ?x)
        (assert  (imu-quality  ?imu  ?fault)))

;-----------------------------------------------------------------------------

(defrule  imu-status-light
        (sub-phase  imu  error-isolation)
        (imu-avail-output  ?system  ?imu  ?availability)
        (imu-quality  ?imu  ?quality)
        =>
        (if  (eq  ?system  pass)
            then
                (bind  ?subsys  pass-imu)
            else
                (bind  ?subsys  bfs-imu))
        (if  (eq ?availability avail)
            then
                (bind  ?status  ?quality)
            else
                (bind  ?status  ?availability))
        (assert  (status-light  ?subsys  ?imu  ?status)))
```

```
;;****************************************************************************
;;
;;;  GROUP
;;      IMU Error Magnitude (3.4.2.3)
;;
;;      This group determines the magnitude of an error on an IMU; i.e., how
;;      much bias, how much drift, how big a resolver error.
;;
;;;  CONTROL FACTS
;       (sub-phase  imu  error-magnitude)
;;
;;;  CONTAINING GROUP
;;      Inertial Measurement Units
;;
;;****************************************************************************

(defrule bias-magnitude

;;      IF
;;              The PASS is engaged
;;              IMU A has an accelerometer bias
;;              IMU B velocity is valid
;;              IMU C velocity is invalid or IMU A-C compare has a smaller
;;              difference than the IMU A-B comparison
;;      THEN
;;              Compute the magnitude of the bias using the A-B
;;                      pairwise velocity comparison.
;;              Notify operator of the magnitude of the bias.
;;      END

        (sub-phase  imu  error-magnitude)
        (engaged-system  pass)
        (imu-quality  ?imu-a  bias)
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (lrus-in-pair  ?pair-ac&~?pair-ab  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-b  vel  valid)
        (or (is-imu-valid  ?imu-c  vel  ~valid)
            (test (< (vel-diff  ?pair-ac)
                     (vel-diff  ?pair-ab))) )
        =>
        (assert  (event pass-imu off-nominal alt
                "Bias on IMU " ?imu-a " is " =(bias (vel-diff ?pair-ab))
                " micro-gs")))


;---------------------------------------------------------------------------

(defrule resolver-magnitude

;;      IF
;;              The PASS is engaged
;;              IMU A has a resolver error
;;              IMU B attitude is valid
;;              IMU C attitude is invalid or IMU A-C compare has
;;                a smaller difference than the IMU A-B comparison
;;      THEN
;;              Compute the magnitude of the resolver error using the A-B
;;                      pairwise attitude comparison.
;;              Notify operator of the magnitude of the resolver error.
;;      END

        (sub-phase  imu  error-magnitude)
        (engaged-system  pass)
        (imu-quality  ?imu-a  resolver)
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (lrus-in-pair  ?pair-ac&~?pair-ab  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-b  att  valid)
        (or (is-imu-valid  ?imu-c  att  ~valid)
            (test (< (att-diff ?pair-ac)
                     (att-diff ?pair-ab))))
        =>
        (assert  (event pass-imu off-nominal alt
                "Resolver error on IMU " ?imu-a " is "
                   =(resolver (att-diff ?pair-ab)) " degrees")))


;-----------------------------------------------------------------------------

(defrule initial-misalignment

;;      IF
;;              The PASS is engaged
;;              The initial misalignment for IMU A is unknown
;;              IMU B attitude is valid
;;              IMU C attitude is invalid or IMU A-C has a lower difference
;;                than The IMU A-B comparison
;;      THEN
;;              Compute the misalignment of IMU A using the A-B
;;                      pairwise attitude comparison.
;;              Save the computed misalignment for later drift calculations.
;;      END

        (sub-phase  imu  error-magnitude)
        (engaged-system  pass)
        ?x <- (initial-misalignment  ?imu-a  unknown)
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (lrus-in-pair  ?pair-ac&~?pair-ab  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-b  att  valid)
        (or (is-imu-valid  ?imu-c  att  ~valid)
            (test (< (att-diff ?pair-ac)
                     (att-diff ?pair-ab))) )
        (current-time  ?time)
        =>
        (bind  ?resolver  (resolver  (att-diff  ?pair-ab)))
        (retract  ?x)
        (assert  (initial-misalignment  ?imu-a  ?resolver  ?time)))


;--------------------------------------------------------------------------

(defrule drift-magnitude

;;      IF
;;              The PASS is engaged
;;              IMU A has a drift
;;              The initial misalignment of IMU A is known
;;              IMU B attitude is valid
;;              IMU C attitude is invalid or IMU A-C compare has a
;;                smaller difference than IMU A-B compare
;;      THEN
;;              Compute the magnitude of the drift using the A-B
;;                      pairwise attitude comparison and
;;                      the initial misalignment of A.
;;              Notify operator of the magnitude of the drift.
;;      END

        (sub-phase  imu  error-magnitude)
        (engaged-system  pass)
        (imu-quality  ?imu-a  drift)
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (lrus-in-pair  ?pair-ac&~?pair-ab  ?imu-a  ?imu-c)
        (is-imu-valid  ?imu-b  att  valid)
        (or (is-imu-valid  ?imu-c  att  ~valid)
            (test (< (att-diff ?pair-ac)
                     (att-diff ?pair-ab))) )
        (current-time  ?time)
        (initial-misalignment  ?imu-a  ?resolver-0  ?time-0)
        =>
        (bind  ?resolver  (resolver  (att-diff ?pair-ab)))
        (bind  ?drift  (drift  ?resolver  ?resolver-0  ?time  ?time-0))
        (assert  (event pass-imu off-nominal alt
                "Drift on IMU " ?imu-a " is " ?drift " deg/hr")))


;;****************************************************************************
;;
;;;  GROUP
;;      IMU Failure Prediction (3.4.2.4)
;;
;;      This group tries to predict whether IMU RM will take any action on an
;;      IMU error
;;
;;;  CONTROL FACTS
;       (sub-phase  imu  failure-prediction)
;;
;;;  CONTAINING GROUP
;;      Inertial Measurement Units
;;
;;****************************************************************************

(defrule three-level-failure-prediction

;;      IF
;;              Onboard IMU RM is at the 3 level
;;              Exactly two pairwise differences exceed the fail threshold in
;;                      either velocity or attitude
;;              A failure has not yet been predicted
;;      THEN
;;              Predict RM will fail the IMU common to the two pairs that
;;                      exceed the threshold and notify the operator.
;;      END

        (sub-phase  imu  failure-prediction)
        (imu-sfc  111)
        (rel-imu-comp  ?pair-1  ?comp  over)
```

```
         (rel-imu-comp  ?pair-2&~?pair-1  ?comp  over)
         (rel-imu-comp  ?pair-3&~?pair-1&~?pair-2  ?comp  ~over)
         (common-lru  ?pair-1  ?pair-2  ?imu)
         ?x <- (imu-rm-prediction  ~fail)
         =>
         (assert  (event pass-imu off-nominal alt
                  "Predict RM will fail IMU " ?imu))
         (retract ?x)
         (assert  (imu-rm-prediction  fail)))



;-----------------------------------------------------------------------------

(defrule three-level-no-failure-prediction

;;       IF
;;                Onboard IMU RM is at the 3 level
;;                All 3 pairwise differences in velocity or attitude exceed the
;;                        fail threshold
;;                A failure has not yet been predicted
;;       THEN
;;                Predict IMU RM will not take any action.
;;       END

         (sub-phase  imu  failure-prediction)
         (imu-sfc  111)
         (rel-imu-comp  p-1-2  ?comp  over)
         (rel-imu-comp  p-1-3  ?comp  over)
         (rel-imu-comp  p-2-3  ?comp  over)
         ?x <- (imu-rm-prediction  none)
         =>
         (assert  (event pass-imu off-nominal alt
                  "RM will not fail any IMUs"))
         (retract ?x)
         (assert  (imu-rm-prediction  inaction)))



;-----------------------------------------------------------------------------

(defrule two-level-failure-prediction

;;       IF
;;                Onboard IMU RM is at the 2 level
;;                IMU A is available but not good
;;                IMU B is available and good
;;                IMUs A and B differ in velocity or attitude by more than
;;                        some threshold
;;                A failure has not yet been predicted
;;       THEN
;;                Predict an RM action, and indicate IMU A is the one that needs
;;                        to be failed.
;;       END

         (sub-phase  imu  failure-prediction)
         (imu-sfc  011|101|110)
         (imu-avail-output  pass  ?imu-a  avail)
         (imu-quality  ?imu-a  ~good)
         (imu-avail-output  pass  ?imu-b&~?imu-a  avail)
```

```
        (imu-quality  ?imu-b  good)
        (lrus-in-pair  ?pair  ?imu-a  ?imu-b)
        (rel-imu-comp  ?pair  ?comp  over)
        ?x <- (imu-rm-prediction   fail)
        =>
        (assert  (event pass-imu off-nominal alt
                "RM needs to fail IMU " ?imu-a))
        (retract ?x)
        (assert  (imu-rm-prediction  fail)))


;;*****************************************************************************
;;
;;;  GROUP
;;      PASS IMU Recommendations (3.4.3.1)
;;
;;      Given the current state of IMUs, this group determines what actions are
;;      required in the PASS.
;;
;;;  CONTROL FACTS
;       (sub-phase  imu  pass-recommendation)
;;
;;;  CONTAINING GROUP
;;      Inertial Measurement Units
;;
;;*****************************************************************************

(defrule reselect-imu-with-one-or-three-state-nav

;;      IF
;;              An IMU is unavailable to the PASS due to deselection
;;              That IMU is good
;;      THEN
;;              Recommend that IMU be reselected (after 0-delta-state if
;;                      3-state nav is still active).
;;      END

        (sub-phase  imu  pass-recommendation)
        (imu-avail-output  pass  ?imu  deselect)
        (imu-quality  ?imu  good)
        (nav-3-state  ?nav-flag)
        =>
        (if  (eq  ?nav-flag  on)
            then
                (assert (recommend pass-imu reselect-imu off-nominal alt
                        "After zero delta state, OK to reselect IMU " ?imu))
            else
                (assert (recommend pass-imu reselect-imu off-nominal alt
                        "OK to reselect IMU " ?imu))))

;---------------------------------------------------------------------------

(defrule help-imu-dilemma

;;      IF
;;              IMU RM is in dilemma
;;              IMU A is available to the PASS and good
;;              IMU B is available to the PASS and not good
;;        THEN
;;                Recommend deselecting IMU B.
;;        END

          (sub-phase  imu  pass-recommendation)
          (imu-dilemma  on)
          (imu-avail-output  pass  ?imu-a  avail)
          (imu-quality  ?imu-a  good)
          (imu-avail-output  pass  ?imu-b  avail)
          (imu-quality  ?imu-b  ~good)
          =>
          (assert  (recommend pass-imu help-imu-dilemma off-nominal alt
                  "Resolve IMU dilemma by deselecting IMU " ?imu-b)))


;---------------------------------------------------------------------------

(defrule cant-help-imu-dilemma

;;        IF
;;                IMU RM is in dilemma
;;                IMU A is available to the PASS
;;                IMU B is available to the PASS
;;                Either A and B are both good or A and B are both not good
;;        THEN
;;                Notify operator that dilemma cannot be resolved.
;;        END

          (sub-phase  imu  pass-recommendation)
          (imu-dilemma  on)
          (imu-avail-output  pass  ?imu-a  avail)
          (imu-avail-output  pass  ?imu-b&~?imu-a  avail)
          (or  (and
                  (imu-quality  ?imu-a  good)
                  (imu-quality  ?imu-b  good))
               (and
                  (imu-quality  ?imu-a  ~good)
                  (imu-quality  ?imu-b  ~good)))
          (not  (cant-help-imu-dilemma))
          =>
          (assert  (cant-help-imu-dilemma))
          (assert  (event pass-imu off-nominal alt
                  "IMU RM DILEMMA.  Don't know which IMU is best.")))

;------------------------------------------------------------------------------

(defrule  end-imu-dilemma
          (sub-phase  imu  pass-recommendation)
          ?x <- (cant-help-imu-dilemma)
          (imu-dilemma  off)
          =>
          (retract  ?x))

;---------------------------------------------------------------------------

(defrule incorrect-imu-failure

;;        IF
;;                IMU A is unavailable to the PASS due to failure
;;                IMU A is good
;;              IMU B is available to the PASS
;;              IMU B is not good
;;      THEN
;;              Notify operator of incorrect RM isolation and recommend
;;                      switching to IMU A.
;;      END

        (sub-phase  imu  pass-recommendation)
        (imu-avail-output  pass  ?imu-a  fail)
        (imu-quality  ?imu-a  good)
        (imu-avail-output  pass  ?imu-b  avail)
        (imu-quality  ?imu-b  ~good)
        =>
        (assert (recommend pass-imu incorrect-imu-failure off-nominal alt
                "RM failed the wrong IMU; Reselect IMU " ?imu-a
                " and deselect IMU " ?imu-b)))


;--------------------------------------------------------------------------

(defrule deselect-commfaulted-imu

;;      IF
;;              An IMU is unavailable to the PASS due to commfault
;;              That IMU has not been deselected
;;      THEN
;;              Recommend deselecting the IMU.
;;      END

        (sub-phase  imu  pass-recommendation)
        (imu-avail-output  pass  ?imu  commfault)
        (imu-flag  pass  deselect  ?imu  off)
        =>
        (assert  (recommend pass-imu deselect-commfaulted-imu  off-nominal alt
                "Need to deselect IMU " ?imu)))


;;*****************************************************************************
;;
;;;  GROUP
;;      BFS IMU Recommendations (3.4.3.2)
;;
;;      Given the current state of IMUs, this group determines what actions
;;      are required in the BFS.
;;
;;;  CONTROL FACTS
;       (sub-phase  imu  bfs-recommendation)
;;
;;;  CONTAINING GROUP
;;      Inertial Measurement Units
;;
;;*****************************************************************************

(defrule deselect-imu-in-bfs

;;      IF
;;              IMU A is not available to the PASS
;;              IMU A is available to the BFS
;;              IMU B is available to the BFS
;;              IMU B is good
;;      THEN
;;              Recommend deselecting IMU A in the BFS.
;;      END

        (sub-phase  imu  bfs-recommendation)
        (imu-avail-output  pass  ?imu  ~avail)
        ?x <- (imu-avail-output  bfs  ?imu  avail)
        (imu-avail-output  bfs  ?other-imu&~?imu  avail)
        (imu-quality  ?other-imu  good)
        =>
        (assert  (recommend  bfs-imu deselect-imu-in-bfs  off-nominal alt
                "Recommend deselecting IMU " ?imu " in the BFS")))



;-----------------------------------------------------------------------------

(defrule no-bfs-imus

;;      IF
;;              The BFS is on IMU A
;;              IMU A is unavailable to the PASS
;;              Neither IMUs B nor C is good and available to the BFS
;;      THEN
;;              Notify operator of IMU shortage in the BFS.
;;      END

        (sub-phase  imu  bfs-recommendation)
        (bfs-imu  ?imu-a)
        (imu-avail-output  pass  ?imu-a  ~avail)
        (lrus-in-pair  ?  ?imu-a  ?imu-b)
        (lrus-in-pair  ?  ?imu-a  ?imu-c&~?imu-b)
        (test  (<  ?imu-b  ?imu-c))
        (imu-avail-output  bfs  ?imu-b  ~avail)
        (imu-quality  ?imu-b  ~good)
        (imu-avail-output  bfs  ?imu-c  ~avail)
        (imu-quality  ?imu-c  ~good)
        =>
        (assert  (event bfs-imu off-nominal alt
                "The BFS is on IMU " ?imu-a
                " and has no other IMUs available")))



;-----------------------------------------------------------------------------

(defrule change-bfs-imu-1

;;      IF
;;              The BFS is on IMU A
;;              IMU A is not good
;;              IMU A is available to the PASS
;;              IMU B is available to the BFS
;;              IMU B is good
;;              Either IMU C is unavailable to the BFS or has a higher number
;;                      than IMU B
;;      THEN
;;              Recommend deselect/reselect IMU A to put the BFS on IMU B.
;;      END
```

```
        (sub-phase  imu  bfs-recommendation)
        (bfs-imu  ?imu-a)
        (imu-quality  ?imu-a  ~good)
        (imu-avail-output  pass  ?imu-a_ avail)
        (imu-avail-output  bfs  ?imu-b&~?imu-a  avail)
        (imu-quality  ?imu-b  good)
        (or  (imu-avail-output  bfs  ~?imu-a&~?imu-b  ~avail)
             (and  (imu-quality  ?imu-c&~?imu-b&~?imu-a  good)
                   (imu-avail-output  bfs  ?imu-c  avail)
                   (test  (<  ?imu-b  ?imu-c))))
        =>
        (assert  (recommend bfs-imu change-bfs-imu off-nominal alt
                "Recommend deselect-reselect IMU " ?imu-a
                " in the BFS to get it on IMU " ?imu-b)))


;-------------------------------------------------------------------------

(defrule change-bfs-imu-2

;;      IF
;;              The BFS is on IMU A
;;              IMU A is not good
;;              IMU B is available to the BFS and is good
;;              IMU C is available to the BFS but is not good

;;              IMU C has a lower number than IMU B
;;      THEN
;;              Recommend deselect/reselect IMUs A and C to put the BFS
;;                      on IMU B.
;;      END
        (sub-phase  imu  bfs-recommendation)
        (bfs-imu  ?imu-a)
        (imu-quality  ?imu-a  ~good)
        (imu-avail-output  bfs  ?imu-b&~?imu-a  avail)
        (imu-quality  ?imu-b  good)
        (imu-avail-output  bfs_  ?imu-c&~?imu-a&~?imu-b  avail)
        (imu-quality  ?imu-c  ~good)
        (test  (<  ?imu-c  ?imu-b))
        =>
        (assert (recommend bfs-imu change-bfs-imu off-nominal alt
                "Recommend deselect-reselect IMUs " ?imu-a " and "
                ?imu-c " in the BFS to get it on IMU " ?imu-b)))
```

3.5 State Vectors

```
;;*****************************************************************************
;;
;;;  GROUP (3.5)
;;      State Vector.
;;
;;      This group watches the PASS and BFS state vectors.
;;
;;;  CONTROL FACTS
;       (sub-phase  state  ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;*****************************************************************************


;;;  FACTS

(deffacts  monitoring-state-phases       ; These facts define the sequence of
                                         ; sub-phases in the monitoring phase
                                         ; of state vectors
        (first-sub-phase  state  monitoring  quality)
                                         ; The only sub-phase is quality checks
)

(deffacts  analysis-state-phases         ; These facts define the sequence of
                                         ; sub-phases in the analysis phase of
                                         ; state vectors
        (first-sub-phase  state  analysis  delta-state)
                                         ; The first sub-phase is delta-state
                                         ; recommendations
        (next-sub-phase  state  delta-state  bfs-transfer)
                                         ; The last sub-phase is BFS transfer
                                         ; recommendations
)


(deffacts  last-state-report                     ; Initializes facts which
                                                 ;  contain the times when the
                                                 ;  state errors were reported
                                                 ;  and the status that was
                                                 ;  reported. The initial
        (last-state-report-with-hstd pass unknown 0.0)
                                                 ;  status is set to "unknown"
                                                 ;  so the status will be
                                                 ;  reported as soon as it is
                                                 ;  known.
        (last-state-report-with-hstd bfs unknown 0.0)
        (last-state-report-no-hstd unknown 0.0)
        (previous-pass-bfs x unknown)
        (previous-pass-bfs y unknown)
        (previous-pass-bfs z unknown)
)


;;*****************************************************************************
;;
;;;  GROUP (3.5.1)
;;      State Error Status
```

```
;;
;;      This group reports the quality of the PASS and
;;      BFS state vectors
;;
;;;  CONTROL FACTS
;       (sub-phase  state  quality)
;;
;;;  CONTAINING GROUP
;;      State Vectors
;;
;;*****************************************************************************

(defrule  state-error-change

;;      IF
;;              For the available system
;;              The HSTD is good  AND
;;              The PASS or BFS worst axis error is different from what
;;                      it was on the previous cycle
;;      THEN
;;              Record the new worst axis status
;;      END

        (sub-phase  state  quality)
        (hstd  good)
        (system-available  ?system)
        (gnd-state  ?system worst-axis  ?status)
        ?x <- (last-state-report-with-hstd  ?system  ~?status  ?)
        =>
        (if  (eq ?status over)
            then
                (assert  (status-light state ?system no-go))
            else
                (assert  (status-light state ?system go)))
        (retract  ?x)
        (assert  (last-state-report-with-hstd  ?system  ?status  0.0)))

;-----------------------------------------------------------------------------

(defrule  state-report-state-error

;;      IF
;;              For the available system
;;              The HSTD is good  AND
;;              More than 60 seconds has elapsed since the last report
;;      THEN
;;              Report the error on every axis whose status is the same
;;                      as the worst axis
;;      END

        (sub-phase  state  quality)
        (hstd  good)
        (system-available  ?system)
        ?x <- (last-state-report-with-hstd  ?system  ?status  ?last-time)
        (gnd-state  ?system  u  ?u)
        (gnd-state  ?system  v  ?v)
        (gnd-state  ?system  w  ?w)
        (gnd-state  ?system  udot  ?udot)
        (gnd-state  ?system  vdot  ?vdot)
        (gnd-state  ?system  wdot  ?wdot)
        (current-time  ?time)
        (test  (>=  ?time  (+ ?last-time 60.0)))
        =>
        (if  (eq  ?status  under)
            then
                (assert  (event state nominal alt
                        "The " ?system " nav state is go"))
            else
                (if  (eq  ?u  ?status)
                    then
                        (bind  ?e  (state-error ?system u))
                        (assert  (event state nominal alt
                            "The " ?system " U error is " ?e " feet")))
                (if  (eq  ?v  ?status)
                    then
                        (bind  ?e  (state-error ?system v))
                        (assert  (event state nominal alt
                            "The " ?system " V error is " ?e " feet")))
                (if  (eq  ?w  ?status)
                    then
                        (bind  ?e  (state-error ?system w))
                        (assert  (event state nominal alt
                            "The " ?system " W error is " ?e " feet")))
                (if  (eq  ?udot  ?status)
                    then
                        (bind  ?e  (state-error ?system udot))
                        (assert  (event state nominal alt
                            "The " ?system " UDOT error is " ?e " feet/sec")))
                (if  (eq  ?vdot  ?status)
                    then
                        (bind  ?e  (state-error ?system vdot))
                        (assert  (event state nominal alt
                            "The " ?system " VDOT error is " ?e " feet/sec")))
                (if  (eq  ?wdot  ?status)
                    then
                        (bind  ?e  (state-error ?system wdot))
                        (assert  (event state nominal alt
                           "The " ?system " WDOT error is " ?e " feet/sec"))))
        (retract  ?x)
        (assert  (last-state-report-with-hstd  ?system  ?status  ?time)))

;-----------------------------------------------------------------------------

(defrule state-pass-bfs-timing-problem

;;      IF
;;              The HSTD is not good AND
;;              Both systems are available AND
;;              The delta time is greater than 0.003 seconds
;;      THEN
;;              Report that there is a timing problem between
;;              the PASS and BFS
;;      END

        (sub-phase  state  quality)
        (hstd  ~good)
        (system-available  pass)
        (system-available  bfs)
        (pass-bfs-delta-time  over)
        =>
        (assert (event  state  off-nominal  alt
           "There is a timing problem between PASS and BFS")))

;------------------------------------------------------------------------------

(defrule  state-pass-bfs-error-change

;;      IF
;;              Both systems are available AND
;;              There is no timing problem between the PASS and the BFS AND
;;              The HSTD is not good  AND
;;              The PASS-BFS worst axis error is different from what
;;                      it was on the previous cycle
;;      THEN
;;              Record the new worst axis status
;;      END

        (sub-phase  state  quality)
        (system-available  pass)
        (system-available  bfs)
        (pass-bfs-delta-time  under)
        (hstd  ~good)
        (pass-bfs  worst-axis  ?status)
        ?x <- (last-state-report-no-hstd  ~?status  ?)
        =>
        (retract  ?x)
        (assert  (last-state-report-no-hstd  ?status  0.0)))

;------------------------------------------------------------------------------

(defrule  state-report-pass-bfs-error

;;      IF
;;              Both systems are available AND
;;              There is no timing problem between the PASS and the BFS AND
;;              The HSTD is not good  AND
;;              More than 60 seconds has elapsed since the last report
;;                      of PASS-BFS errors
;;      THEN
;;              Report the error on every axis whose status is the same
;;                      as the worst axis
;;      END

        (sub-phase  state  quality)
        (hstd  ~good)
        (system-available  pass)
        (system-available bfs)
        (pass-bfs-delta-time  under)
        ?a <- (last-state-report-no-hstd  bfs  ?status  ?last-time)
        (pass-bfs  x  ?x)
        (pass-bfs  y  ?y)
        (pass-bfs  z  ?z)
        (pass-bfs  xdot  ?xdot)
        (pass-bfs  ydot  ?ydot)
        (pass-bfs  zdot  ?zdot)
        (current-time  ?time)
        (test  (>=  ?time  (+ ?last-time 60.0)))
        =>
        (if  (eq  ?status  under)
           then
```

```
                (assert  (event state nominal alt
                        "The" " PASS and BFS are tracking"))
            else
                (if  (eq  ?x  ?status)
                    then
                        (bind  ?e  (pass-bfs x))
                        (assert  (event state nominal alt
                            "PASS-BFS X is " ?e " feet")))
                (if  (eq  ?y  ?status)
                    then
                        (bind  ?e  (pass-bfs y))
                        (assert  (event state nominal alt
                            "PASS-BFS Y is " ?e " feet")))
                (if  (eq  ?z  ?status)
                    then
                        (bind  ?e  (pass-bfs z))
                        (assert  (event state nominal alt
                            "PASS-BFS Z is " ?e " feet")))
                (if  (eq  ?xdot  ?status)
                    then
                        (bind  ?e  (pass-bfs xdot))
                        (assert  (event state nominal alt
                            "PASS-BFS XDOT is " ?e " feet/sec")))
                (if  (eq  ?ydot  ?status)
                    then
                        (bind  ?e  (pass-bfs ydot))
                        (assert  (event state nominal alt
                            "PASS-BFS YDOT is " ?e " feet/sec")))
                (if  (eq  ?zdot  ?status)
                    then
                        (bind  ?e  (pass-bfs zdot))
                        (assert  (event state nominal alt
                            "PASS-BFS ZDOT is " ?e " feet/sec"))))
        (retract  ?a)
        (assert  (last-state-report-no-hstd  bfs  ?status  ?time)))


;;*****************************************************************************
;;
;;;  GROUP (3.5.2)
;;      Delta State Update
;;
;;      This group determines whether or not a delta state update is
;;      needed.
;;
;;;  CONTROL FACTS
;       (sub-phase  state  delta-state)
;;
;;;  CONTAINING GROUP
;;      State Vectors
;;
;;*****************************************************************************

(defrule  state-need-delta-state

;;      IF
;;              The HSTD is good  AND
;;              For the engaged system the
;;              GND-system shows the system is above the update limits
;;      THEN
;;              Request a delta-state update.
;;      END

        (sub-phase  state  delta-state)
        (hstd  good)
        (engaged-system ?system)
        (gnd-state  ?system  worst-axis  over)
        (gnd-state  ?system  worst-velocity  ?velocity)
        =>
        (if  (||  (eq  ?velocity  under) (eq  ?velocity  zero))
            then
                (bind  ?update-type  position-only)
            else
                (bind  ?update-type  position-and-velocity))
        (assert  (need-delta-state  ?update-type)))

;-----------------------------------------------------------------------------

(defrule  state-ok-for-delta-state

;;      IF
;;              The HSTD is good  AND
;;              A delta state is needed
;;              Ground and engaged system runway are the same
;;      THEN
;;              Recommend a delta state update
;;      END

        (sub-phase  state  delta-state)
        (hstd good)
        (need-delta-state  ?update-type)
        (engaged-system  ?system)
        (runway  ground  ?runway)
        (runway  ?system  ?runway)
        =>
        (assert  (recommend  state  update-xfer  off-nominal  alt
                "We need a " ?update-type
                " update to the " ?system)))


;-----------------------------------------------------------------------------

(defrule  state-not-ok-for-delta-state

;;      IF
;;              The HSTD is good  AND
;;              A delta state is needed
;;              Ground and engaged system runway are not the same
;;      THEN
;;              Notify the operator that a delta is needed but
;;              there is a runway mismatch.
;;      END

        (sub-phase  state  delta-state)
        (hstd  good)
        (need-delta-state  ?update-type)
        (engaged-system  ?system)
        (runway  ground  ?runwaya)
        (runway  ?system  ?runwayb&~?runwaya)
        =>
        (assert  (recommend  state  update-xfer  off-nominal  alt
                "We need a " ?update-type " update to the " ?system
                 " but there is a mismatch in runways ground = "
                 ?runwaya " " ?system " = " ?runwayb)))


;-----------------------------------------------------------------------------

(defrule  state-inhibit-filter-processing

;;      IF
;;              For the engaged system
;;              A position and/or velocity delta state is needed AND
;;              The drag, TACAN, and/or ADTA flags are not inhibited.
;;      THEN
;;              Notify the operator that (sensor) is not inhibited
;;              and need to be inhibited before the delta state.
;;              (include item entries)
;;              NOTE: item entries are as follows:
;;
;;              Specification number: BFS=50 PASS=51
;;                     TACAN inhibit   item 20
;;                     Drag  inhibit   item 23
;;                     ADTA  inhibit   item 26
;;      END

        (sub-phase  state  delta-state)
        (hstd  good)
        (need-delta-state  ?update-type)
        (engaged-system  ?system)
        (aif  ?system  tacan  ?status-tacan)
        (aif  ?system  baro  ?status-baro)
        (aif  ?system  drag  ?status-drag)
        =>
        (if   (eq ?system pass)
              then
                (bind ?spec 51)
              else
                (bind ?spec 50))
        (if   (neq  ?status-tacan  inhibit)
              then
                (assert (event state update-xfer off-nominal alt
                "need to inhibit tacan in the " ?system
                " to perform a " ?update-type "delta state by "
                "executing an item 20 of spec " ?spec)))
        (if   (neq  ?status-baro  inhibit)
              then
                (assert (event state update-xfer off-nominal alt
                "need to inhibit baro in the " ?system
                " to perform a " ?update-type "delta state by "
                "executing an item 26 of spec " ?spec)))
        (if   (neq  ?status-drag  inhibit)
              then
                (assert (event state update-xfer off-nominal alt
                "need to inhibit drag in the " ?system
                " to perform a " ?update-type "delta state by "
                "executing an item 23 of spec " ?spec))))

;-----------------------------------------------------------------------
```

```
(defrule  state-delta-state-is-in-bfs

;;      IF
;;              BFS is engaged AND
;;              Delta-state is in progress AND
;;              Ground-system errors previously not close to zero AND
;;              Ground-system errors are now close to zero
;;      THEN
;;              Report that state update is in
;;      END

        (sub-phase  state  delta-state)
        (engaged-system bfs)
        ?x <- (need-delta-state ?update-type)
        (gnd-state bfs worst-axis ?near-zero)
        (test (< ?near-zero 200))
        =>
        (assert (event state update-xfer nominal alt
                "delta state " ?update-type " occurred in the bfs"))
        (retract ?x))


;;**************************************************************************
;;
;;; GROUP (3.5.3)
;;      BFS Transfer
;;
;;      This group checks to see if a transfer to the BFS is needed.
;;
;;; CONTROL FACTS
;       (sub-phase  state  bfs-transfer)
;;
;;; CONTAINING GROUP
;;      State Vectors
;;
;;**************************************************************************

(defrule  state-need-transfer

;;      IF
;;              The HSTD is good  AND
;;              Both systems are available AND
;;              GND-BFS shows the BFS state is above the update limits AND
;;              Either the PASS state error is good OR
;;              The PASS state error status is suspect and the PASS-BFS
;;                 status is suspect or bad AND
;;              No timing error exist between the PASS-BFS
;;      THEN
;;              Recommend a transfer to the BFS
;;      END

        (sub-phase  state  bfs-transfer)
        (hstd  good)
        (system-available pass)
        (system-available bfs)
        (gnd-state  bfs  worst-axis  over)
        (gnd-state  pass worst-axis  ?status-a)
        (pass-bfs worst-axis ?status-b)
        (pass-bfs-delta-time under)
        (or   (or (test (eq ?status-a zero))
                  (test (eq ?status-a under)))
             (and (test (eq ?status-a suspect))
              (or (test (eq ?status-b suspect))
                  (test (eq ?status-b over)))))
        =>
        (assert  (recommend state  bfs-transfer  off-nominal  alt
                "We" " need a transfer to the BFS")))

;----------------------------------------------------------------------------

(defrule  state-transfer-in

;;      IF
;;              PASS-BFS position differences are now close to zero  AND
;;              PASS-BFS position differences were not close to zero previously
;;      THEN
;;              Report that the transfer is in
;;      END

        (sub-phase  state  bfs-transfer)
        (pass-bfs  x  zero)
        (pass-bfs  y  zero)
        (pass-bfs  z  zero)
        (previous-pass-bfs  x  ~zero&~unknown)
        (previous-pass-bfs  y  ~zero&~unknown)
        (previous-pass-bfs  z  ~zero&~unknown)
        (not  (transfer-occurred))
        =>
        (assert  (event  state  nominal  alt  "BFS" " transfer is in"))
        (assert  (transfer-occurred)))

;----------------------------------------------------------------------------

(defrule  state-previous-pass-bfs-error-update

;;      IF
;;              PASS-BFS position differences are different from what
;;                 it was on the previous cycle
;;      THEN
;;              Update the previous PASS-BFS error differences
;;      END

        (sub-phase  state  bfs-transfer)
        (pass-bfs  x  ?x-error)
        (pass-bfs  y  ?y-error)
        (pass-bfs  z  ?z-error)
        ?x <- (previous-pass-bfs  x  ~?x-error)
        ?y <- (previous-pass-bfs  y  ~?y-error)
        ?z <- (previous-pass-bfs  z  ~?z-error)
        =>
        (retract ?x ?y ?z)
        (assert (previous-pass-bfs  x  ?x-error))
        (assert (previous-pass-bfs  y  ?y-error))
        (assert (previous-pass-bfs  z  ?z-error)))

;----------------------------------------------------------------------------
(defrule  state-transfer-cleanup
        (sub-phase  state  bfs-transfer)
        ?x <- (transfer-occurred)
        (pass-bfs  x|y|z  ~zero)
        =>
        (retract ?x))

;-----------------------------------------------------------------------------
```

3.6 Three-String State Vectors

```
;;******************************************************************************
;;
;;;  GROUP
;;      Three State Nav (3.6)
;;
;;      This section performs checks on the 3-string state vectors, determining
;;      the quality of each vector.  It also detects delta-state updates.
;;
;;;  CONTROL FACTS
;       (sub-phase three-state three-state)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;******************************************************************************


;;;  FACTS

(deffacts  monitoring-3state-phases      ; These facts define the sequence of
                                         ; subphases within the monitoring phase
                                         ; of 3-state nav.
        (first-sub-phase  three-state  monitoring  three-state)
                                         ; There is only 1 subphase, called
                                         ; three-state.
)


(deffacts initial-3state-facts           ; These facts represent assumptions
                                         ; about 3-state nav before any data is
                                         ; received.
        (state-quality 1 unknown)        ; quality of state vector 1 is unknown.
        (state-quality 2 unknown)        ; quality of state vector 2 is unknown.
        (state-quality 3 unknown)        ; quality of state vector 3 is unknown.
        (nav-3-state on)                 ; 3-state nav is active
)

;-----------------------------------------------------------------------------

(defrule end-3-state-nav

;;      IF
;;              3-state nav is active
;;              A MSBLS measurement has been processed
;;      THEN
;;              Conclude 3-state nav is no longer active
;;      END

        (sub-phase  three-state  three-state)
        ?x <- (nav-3-state  on)
        (filter-flag  pass  mlsr|mlsa|mlse  process)
        =>
        (retract  ?x)
        (assert  (nav-3-state  off)))

;-----------------------------------------------------------------------------

(defrule gnd-to-state-comparison

;;      IF
;;              3-state nav is active
;;              The HSTD is good
;;              A state vector previously had a certain quality rating
;;              Comparison with the ground indicates a different quality
;;      THEN
;;              Change that state vector's rating to the quality indicated
;;                      by the ground comparison
;;      END

        (sub-phase  three-state  three-state)
        (nav-3-state  on)
        (hstd  good)
        (gnd-3state  ?id  worst-axis  ?status)
        (quality-table  ?status  ?quality)
        ?x <- (state-quality  ?id  ~?quality)
        =>
        (assert  (status-light  three-state  ?id  ?quality))
        (retract ?x)
        (assert  (state-quality  ?id  ?quality)))


;--------------------------------------------------------------------------------

(defrule state-to-state-comparison-1

;;      IF
;;              3-state nav is active
;;              all 3 IMU's are available
;;              The hstd is not good
;;              State A previously had a certain quality rating
;;              Comparison with states B and C indicates a different
;;                      quality
;;      THEN
;;              Change the quality rating of state A to that indicated by
;;                      comparisons with states B and C.
;;      END

        (sub-phase  three-state  three-state)
        (nav-3-state  on)
        (good-imus 3)
        (hstd  ~good)
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (state-state  ?pair-ab  worst-axis  ?status-ab)
        (lrus-in-pair  ?pair-ac  ?imu-a  ?imu-c&~?imu-b)
        (state-state  ?pair-ac  worst-axis  ?status-ac)
        (min-miscompare  ?status-ab  ?status-ac  ?status)
        (quality-table  ?status  ?quality)
        ?x <- (state-quality  ?imu-a  ~?quality)
        =>
        (assert  (status-light  three-state  ?imu-a  ?quality))
        (retract ?x)
        (assert  (state-quality  ?imu-a  ?quality)))


;--------------------------------------------------------------------------------

(defrule state-to-state-comparison-2

;;      IF
;;              3-state nav is active
;;              2 IMU's are not commfaulted
;;              The hstd is not good
;;              State A previously had same rating as State B
;;              IMU A previously had same rating as IMU B
;;              State A comparison with State B has a different
;;                  rating
;;      THEN
;;              Change the quality rating of both states A and B
;;              Notify the operator of inability to tell which
;;              state is going bad
;;      END

        (sub-phase  three-state  three-state)
        (nav-3-state  on)
        (good-imus 2)
        (hstd  ~good)
        ?x <- (state-quality  ?imu-a  ?quality)
        ?y <- (state-quality  ?imu-b&~?imu-a  ?quality)
        (imu-quality ?imu-a ?imu-quality)
        (imu-quality ?imu-b ?imu-quality)
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (state-state  ?pair-ab  worst-axis  ?status-ab)
        (quality-table  ?status-ab  ?new-quality&~?quality)
        =>
        (assert  (status-light  three-state  ?imu-a  ?new-quality))
        (assert  (status-light  three-state  ?imu-b  ?new-quality))
        (retract ?x ?y)
        (assert  (state-quality  ?imu-a  ?new-quality))
        (assert  (state-quality  ?imu-b  ?new-quality))
        (assert  (event three-state off-nominal alt
                  "Unable to isolate which state is going bad, "
                  "state " ?imu-a " or state " ?imu-b)))


;------------------------------------------------------------------------------


(defrule state-to-state-comparison-3

;;      IF
;;              3-state nav is active
;;              2 IMU's are not commfaulted
;;              The hstd is not good
;;              State A previously had same rating as State B
;;              IMU A previously had a better rating than IMU B
;;              State A comparison with State B has a different
;;                   rating
;;      THEN
;;              Change State B's quality rating to the new one
;;              Leave State A's quality rating as it was
;;      END

        (sub-phase  three-state  three-state)
        (nav-3-state  on)
        (good-imus 2)
        (hstd  ~good)
        (state-quality  ?imu-a  ?quality)
        ?x <- (state-quality  ?imu-b&~?imu-a  ?quality)
        (imu-quality ?imu-a ?quality-imua)
        (imu-quality ?imu-b ?quality-imub)
        (or (and  (test  (eq  ?quality-imua  good))
                  (test  (neq  ?quality-imub  good)))
            (and  (test  (eq  ?quality-imua  suspect))
                  (test  (||  (eq  ?quality-imub  bad)
                              (eq  ?quality-imub  unknown)))))
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (state-state  ?pair-ab  worst-axis  ?status-ab)
        (quality-table  ?status-ab  ?new-quality&~?quality)
        =>
        (assert  (status-light  three-state  ?imu-b  ?new-quality))
        (retract ?x)
        (assert  (state-quality  ?imu-b  ?new-quality)))

;----------------------------------------------------------------------------

(defrule state-to-state-comparison-4

;;      IF
;;              3-state nav is active
;;              2 IMU's are not commfaulted
;;              The hstd is not good
;;              State A previously had same rating as State B
;;              State A comparison with State B has a different
;;                   rating
;;      THEN
;;              Change State B's quality rating to the new one
;;              Leave State A's quality rating as it was
;;      END

        (sub-phase  three-state  three-state)
        (nav-3-state  on)
        (good-imus 2)
        (hstd  ~good)
        (state-quality  ?imu-a  ?quality)
        ?x <- (state-quality  ?imu-b&~?imu-a  ?quality)
        (lrus-in-pair  ?pair-ab  ?imu-a  ?imu-b)
        (state-state  ?pair-ab  worst-axis  ?status-ab)
        (quality-table  ?status-ab  ?new-quality&~?quality)
        =>
        (assert  (status-light  three-state  ?imu-b  ?new-quality))
        (retract ?x)
        (assert  (state-quality  ?imu-b  ?new-quality)))


;----------------------------------------------------------------------------

(defrule zero-delta-state-occurred

;;      IF
;;              3-state nav is active
;;              A non-zero delta state has not been recommended
;;              All three pairwise state differences go to zero
;;      THEN
;;              Notify operator that zero-delta-state occurred
;;      END

        (sub-phase  three-state  three-state)
        (nav-3-state  on)
        (not  (need-delta-state  $?))
        (not  (delta-state-occurred))
        (state-state  p-1-2  worst-axis  zero)
        (state-state  p-1-3  worst-axis  zero)
        (state-state  p-2-3  worst-axis  zero)
        =>
        (assert  (event three-state off-nominal alt
                "The " "crew did a zero-delta-state"))
        (assert  (delta-state-occurred)))


;-----------------------------------------------------------------------------

(defrule delta-state-occurred

;;      IF
;;              3-state nav is active
;;              A non-zero delta-state has been recommended
;;              All three pairwise state differences go to zero
;;      THEN
;;              Notify operator that delta state has been performed

        (sub-phase  three-state  three-state)
        (nav-3-state  on)
        ?x <- (need-delta-state  $?)
        (not  (delta-state-occurred))
        (state-state  p-1-2  worst-axis  zero)
        (state-state  p-1-3  worst-axis  zero)
        (state-state  p-2-3  worst-axis  zero)
        =>
        (assert  (event three-state nominal alt
                "Delta-state " "is in the PASS"))
        (assert  (delta-state-occurred))
        (retract  ?x))


;-----------------------------------------------------------------------------

(defrule delta-state-cleanup
        (sub-phase  three-state  three-state)
        ?x <- (delta-state-occurred)
        (state-state  ?  ?  ~zero)
        =>
        (retract  ?x))
```

3.7 Drag Altitude

```
;;******************************************************************************
;;
;;;  GROUP
;;      Drag Altitude (3.7)
;;
;;      This group monitors drag altitude and recommends (output)
;;      a setting for the drag AIF switch.
;;
;;;  CONTROL FACTS
;       (sub-phase  drag  ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;******************************************************************************

;;;  FACTS

(deffacts  monitoring-drag-phases       ; These facts define the sequence of
                                        ; sub-phases within the monitoring
                                        ; phase of drag
        (first-sub-phase  drag  monitoring  watch-flags)
                                        ; The first sub-phase watches for change
                                        ; in the value of flag parameters
)

(deffacts  analysis-drag-phases         ; These facts define the sequence of
                                        ; sub-phases within the analysis phase
                                        ; of drag
        (first-sub-phase  drag  analysis  recommendation)
                                        ; There is only one sub-phase: recom-
)

(deffacts  initial-drag-facts           ; These facts represent assumptions
                                        ; about drag before any data is received
        (prev-filter-flag  pass  drag  process)
                                        ; drag is being processed in the PASS
        (prev-filter-flag  bfs  drag  process)
                                        ; drag is being processed in the BFS
)

;;******************************************************************************
;;
;;;  GROUP
;;      Drag Flag Status (3.7.1)
;;
;;      This group watches for changes in the drag filter flag
;;
;;;  CONTROL FACTS
;       (sub-phase  drag  watch-flags)
;;
;;;  CONTAINING GROUP
;;      Drag Altitude
;;
;;******************************************************************************

(defrule  drag-filter-flag-changed
;;      IF
;;              For available systems
;;              The current value of the drag filter flag is anything but
;;                      off  AND
;;              The value of the flag is different from its previous value
;;      THEN
;;              Conclude that the value has changed
;;              Notify the operator if the new value is "process"
;;      END

        (sub-phase  drag  watch-flags)
        (system-available ?sys)
        (filter-flag  ?sys  drag  ?flag&~off)
        ?x <- (prev-filter-flag  ?sys  drag  ~?flag)
        =>
        (retract ?x)
        (assert  (prev-filter-flag  ?sys  drag  ?flag))
        (if  (eq  ?flag  process)
            then
                (assert  (event drag nominal alt "Processing" " drag"))))

;-----------------------------------------------------------------------------

(defrule  drag-end-drag-processing

;;      IF
;;              For available systems
;;              The current value of the drag filter flag is off  AND
;;              The previous value is not off  AND
;;              Either
;;                      The altitude is less than 85.2 kft  OR
;;                      Baro is being processed
;;      THEN
;;              Conclude drag processing has ended
;;      END

        (sub-phase  drag  watch-flags)
        (system-available  ?sys)
        (filter-flag  ?sys  drag  off)
        ?x <- (prev-filter-flag  ?sys  drag  ~off)
        (altitude  ?alt)
        (or  (test  (<  ?alt  85200))
             (filter-flag  ?sys  baro  process|edit))
        =>
        (retract  ?x)
        (assert  (prev-filter-flag  ?sys  drag  off))
        (assert (event drag nominal alt
                 "Processing" " of drag has stopped in " ?sys)))


;;**************************************************************************
;;
;;;  GROUP
;;      Drag Recommendations (3.7.2)
;;
;;      This group determines a recommended setting for the drag altitude
;;      AIF switch
;;
;;;  CONTROL FACTS
;       (sub-phase  drag  recommendation)
```

End of Document

```
;;              Drag is being forced
;;              The altitude is less than 85.2 kft
;;      THEN
;;              Recommend drag be inhibited
;;      END

        (sub-phase  drag  recommendation)
        (system-available ?sys)
        (aif  ?sys  drag  force)
        (altitude  ?alt)
        (test  (<= ?alt  85200))
        =>
        (assert  (recommend  drag  inhibit-drag  off-nominal  alt
          "We" " are below 85.2 kft; Recommend inhibiting drag in the " ?sys)))

;---------------------------------------------------------------------------
```

3.8 Tactical Air Navigation

```
;;*****************************************************************************
;;
;;;  GROUP
;;      TACAN (3.8)
;;
;;      This group watches the TACAN systems to determine whether
;;      TACAN data is useable, which LRUs are good, and which
;;      ground station should be used.
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;*****************************************************************************


;;;  FACTS

(deffacts monitoring-tacan-phases               ; These facts define the
                                                ; sequence of sub-phases in the
                                                ; monitoring phase of TACAN
        (first-sub-phase  tacan  monitoring  configuration)
                                                ; First is a check of the
                                                ; onboard configuration
        (next-sub-phase  tacan  configuration  availability)
                                                ; Then comes a check for LRU
                                                ; availability
        (next-sub-phase  tacan  availability  quality-rating)
                                                ; Then comes a check on quality
        (next-sub-phase  tacan  quality-rating quality)
        (next-sub-phase  tacan  quality  watch-flags)
                                                ; Last is a flag-status check
)

(deffacts analysis-tacan-phases                 ; These facts define the
                                                ; sequence of sub-phases in the
                                                ; analysis phase of TACAN
        (first-sub-phase  tacan  analysis  toggle)
                                                ; First is a check to see if
                                                ; a toggle is necessary
        (next-sub-phase  tacan  toggle  deselect)
                                                ; Next is a check to see which
                                                ; LRUs need to be deselected
        (next-sub-phase  tacan  deselect  clean-up)
                                                ; Next is a fact-base clean-up
        (next-sub-phase  tacan  clean-up  reselect)
                                                ; Next is a check to see which
                                                ; LRUs need to be reselected
        (next-sub-phase  tacan  reselect  aif-change)
                                                ; Last is a determination of
                                                ; the best AIF setting
)

(deffacts  initial-tacan-facts                  ; These facts represent
                                                ; assumptions about TACAN
                                                ; before any data is received
        (tacan-status  pass  1  range  avail)   ; LRU 1 range available in PASS
        (tacan-status  pass  1  bearing  avail) ; LRU 1 bear available in PASS
        (tacan-status  pass  2  range  avail)   ; LRU 2 range available in PASS
        (tacan-status  pass  2  bearing  avail) ; LRU 2 bear available in PASS
        (tacan-status  pass  3  range  avail)   ; LRU 3 range available in PASS
        (tacan-status  pass  3  bearing  avail) ; LRU 3 bear available in PASS
        (tacan-status  bfs  1  range  avail)    ; LRU 1 range available in BFS
        (tacan-status  bfs  1  bearing  avail)  ; LRU 1 bear available in BFS
        (tacan-status  bfs  2  range  avail)    ; LRU 2 range available in BFS
        (tacan-status  bfs  2  bearing  avail)  ; LRU 2 bear available in BFS
        (tacan-status  bfs  3  range  avail)    ; LRU 3 range available in BFS
        (tacan-status  bfs  3  bearing  avail)  ; LRU 3 bear available in BFS
        (tacan-lru-quality  1  range  none)     ; no rating yet on LRU 1 range
        (tacan-lru-quality  1  bearing  none)   ; no rating yet on LRU 1 bearing
        (tacan-lru-quality  2  range  none)     ; no rating yet on LRU 2 range
        (tacan-lru-quality  2  bearing  none)   ; no rating yet on LRU 2 bearing
        (tacan-lru-quality  3  range  none)     ; no rating yet on LRU 3 range
        (tacan-lru-quality  3  bearing  none)   ; no rating yet on LRU 3 bearing
        (prev-tacan-channel  1  -999)           ; LRU 1 channel number not known
        (prev-tacan-channel  2  -999)           ; LRU 2 channel number not known
        (prev-tacan-channel  3  -999)           ; LRU 3 channel number not known
        (prev-tacan-lock  range  off)           ; no range locked on yet
        (prev-tacan-lock  bearing  off)         ; no bearing locked on yet
        (prev-filter-flag  pass  tacr  off)     ; PASS is not processing range
        (prev-filter-flag  pass  tacb  off)     ; PASS is not processing bearing
        (prev-filter-flag  bfs  tacr  off)      ; BFS is not processing range
        (prev-filter-flag  bfs  tacb  off)      ; BFS is not processing bearing
        (prev-data-good  pass  tacr  off)       ; range data-good off in PASS
        (prev-data-good  pass  tacb  off)       ; bearing data-good off in PASS
        (prev-data-good  bfs  tacr  off)        ; range data-good off in BFS
        (prev-data-good  bfs  tacb  off)        ; bearing data-good off in BFS
        (last-tacan-quality  1  range  unknown) ; LRU 1 previous range quality
        (last-tacan-quality  1  bearing  unknown);LRU 1 previous bearing uality
        (last-tacan-quality  2  range  unknown) ; LRU 2 previous range quality
        (last-tacan-quality  2  bearing  unknown);LRU 2 previous bearing uality
        (last-tacan-quality  3  range  unknown) ; LRU 3 previous range quality
        (last-tacan-quality  3  bearing  unknown);LRU 3 previous bearing uality
        (selected-channel  0)                   ; Actual TACAN channel unknown
        (error-before-tacan  unknown)           ; Status of the state error
                                                ;  before TACAN processing is
                                                ;  unknown
        (selected-tacan  range  no-go)          ; Selected range is not yet good
        (selected-tacan  bearing  no-go)        ; Selected brng is not yet good
)


;;***************************************************************************
;;
;;;   GROUP  (3.8.1)
;;       TACAN Channel Configuration
;;
;;       This group makes sure all LRUs are tuned to the correct channel.
;;
;;;   CONTROL FACTS
;        (sub-phase  tacan  configuration)
;;
;;;   CONTAINING GROUP
;;       TACAN
;;
;;***************************************************************************
```

```
(defrule  tacan-skip-tacan

;;       IF
;;               The wrong runway is selected in the engaged system
;;       THEN
;;               Disable the rest of the TACAN checks
;;       END

         ?x <- (sub-phase  tacan  configuration)
         (runway  desired  ?slot)
         (engaged-system ?sys)
         (runway  ?sys  ~?slot)
         =>
         (retract  ?x))

;-----------------------------------------------------------------------------

(defrule  tacan-channel-changed

;;       IF
;;               All LRUs are tuned to a different channel than before
;;       THEN
;;               Notify operator of the change in selected channel
;;       END

         (sub-phase  tacan  configuration)
         (tacan-channel  1  ?channel)
         (tacan-channel  2  ?channel)
         (tacan-channel  3  ?channel)
         ?x <- (selected-channel  ~?channel)
         =>
         (assert  (tacan-status-changed))
         (retract  ?x)
         (assert  (selected-channel  ?channel))
         (assert  (event  tacan  nominal  alt
                  "TACAN is now on channel " ?channel)))

;-----------------------------------------------------------------------------

(defrule  tacan-toggle-tacan-due-to-wrong-channel

;;       IF
;;               For the engaged system
;;               The selected channel is not the desired channel
;;               The selected channel is in the correct area of the
;;                       site table
;;       THEN
;;               Recommend toggle TACAN to get to the desired channel
;;               Indicate that tacan is no-go for the engaged system
;;       END

         (sub-phase  tacan  configuration)
         (engaged-system ?sys)
         (selected-channel  ?channel&~0)
         (desired-channel  ?desired&~?channel)
         (desired-tacan  ?slot)
         (same-area  ?slot  ?other-slot)
         (test  (=  ?channel (lookup-tacan ?other-slot)))
         =>
         (assert  (status-light  tacan  ?sys  no-go))
```

```
        (assert  (recommend  tacan  toggle-tacan  off-nominal  alt
                "Need to toggle TACAN to get on channel " ?desired)))

;----------------------------------------------------------------------

(defrule  tacan-gpc-mode

;;      IF
;;              For the engaged system
;;              The selected channel is not the desired channel
;;              The selected channel it not in the correct area of the
;;                      site table
;;      THEN
;;              Recommend the TACANs be put in GPC mode
;;              Indicate that TACAN is no-go for the engaged system
;;      END

        (sub-phase  tacan  configuration)
        (engaged-system ?sys)
        (selected-channel  ?channel&~0)
        (desired-channel  ?desired&~?channel)
        (desired-tacan  ?slot)
        (same-area  ?slot  ?other-slot)
        (test  (!  (=  ?channel (lookup-tacan ?other-slot))))
        =>
        (assert  (status-light  tacan  ?sys  no-go))
        (assert  (recommend  tacan  gpc-mode  off-nominal  alt
                "Need" " to put the TACANs in GPC mode")))

;----------------------------------------------------------------------

(defrule  tacan-fix-lru-channel

;;      IF
;;              For the engaged system
;;              One LRU is not tuned to the desired channel
;;              At least one other LRU is tuned to the desired channel
;;      THEN
;;              Recommend the mis-tuned LRU be put in GPC mode
;;              Indicate that TACAN is no-go for the engaged system
;;      END

        (sub-phase  tacan  configuration)
        (engaged-system ?sys)
        (desired-channel  ?channel)
        (tacan-channel  ?lru-a  ~?channel)
        (tacan-channel  ?lru-b  ?channel)
        =>
        (assert  (status-light  tacan  ?sys  no-go))
        (assert  (recommend  tacan  gpc-mode  off-nominal  alt
                "Need to put TACAN " ?lru-a " in GPC mode")))

;----------------------------------------------------------------------

(defrule  tacan-config-is-good

;;      IF
;;              For the engaged system
;;              All three LRUs are tuned to the desired channel
;;      THEN
;;              The TACAN configuration is good
;;      END

        (sub-phase  tacan  configuration)
        (engaged-system ?sys)
        (desired-channel  ?channel)
        (tacan-channel  1  ?channel)
        (tacan-channel  2  ?channel)
        (tacan-channel  3  ?channel)
        =>
        (assert  (status-light  tacan  ?sys  go)))


;;***************************************************************************
;;
;;;  GROUP
;;      TACAN Availability (3.8.2)
;;
;;      This group determines which LRUs are available in the engaged system.
;;      It also determines why the unavailable LRUs are unavailable.
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  availability)
;;
;;;  CONTAINING GROUP
;;      TACAN
;;
;;***************************************************************************

(defrule  tacan-commfault

;;      IF
;;              For the engaged system
;;              A TACAN LRU was not previously commfaulted or powered down
;;              The commfault flag for that LRU is now on
;;      THEN
;;              Notify the operator that the LRU is commfaulted (unless the
;;                      whole string is down)
;;              Conclude that range and bearing from the LRU are no longer
;;                      available due to commfault
;;      END

        (sub-phase  tacan  availability)
        (engaged-system ?sys)
        ?x <- (tacan-status  ?sys  ?lru  range  ~commfault&~power-off)
        ?y <- (tacan-status  ?sys  ?lru  bearing  ~commfault&~power-off)
        (tacan-flag  ?sys  commfault  ?lru  on)
        (string-commfault  ?sys  ?lru  ?string-flag)
        =>
        (if  (eq  ?string-flag  off)
            then
                (assert  (event  tacan  off-nominal  alt
                        "Commfault TACAN " ?lru " in the " ?sys)))
        (assert  (status-light  tacr  ?lru  commfault))
        (assert  (status-light  tacb  ?lru  commfault))
        (assert  (tacan-status-changed))
        (retract  ?x)
        (retract  ?y)
        (assert  (tacan-status  ?sys  ?lru  range  commfault))
        (assert  (tacan-status  ?sys  ?lru  bearing  commfault)))

;---------------------------------------------------------------------------

(defrule  tacan-commfault-clear

;;      IF
;;              For the engaged system
;;              A TACAN LRU was previously commfaulted
;;              The commfault flag for that LRU is now off
;;      THEN
;;              Notify the operator that the commfault has cleared
;;                      (unless) the whole string was down)
;;              Conclude that the LRU has the status indicated by the
;;                      fail and deselect indicators
;;      END

        (sub-phase  tacan  availability)
        (engaged-system ?sys)
        ?x <- (tacan-status  ?sys  ?lru  range  commfault)
        ?y <- (tacan-status  ?sys  ?lru  bearing  commfault)
        (tacan-flag  ?sys  commfault  ?lru  off)
        (tacan-flag  ?sys  deselect  ?lru  ?desel-flag)
        (tacan-fail-flag  ?lru  range  ?range-fail)
        (tacan-fail-flag  ?lru  bearing  ?bearing-fail)
        (prev-string-cf  ?sys  ?lru  ?string-flag)
        (tacan-lru-quality  ?lru  range  ?range-status)
        (tacan-lru-quality  ?lru  bearing  ?bearing-status)
        =>
        (if  (eq  ?string-flag  off)
            then
                (assert  (event  tacan  off-nominal  alt
                    "Commfault clear on TACAN " ?lru " in the " ?sys)))
        (assert  (tacan-status-changed))
        (retract  ?x)
        (retract  ?y)
        (if  (eq  ?desel-flag  on)
            then
                (assert  (status-light  tacr  ?lru  deselect))
                (assert  (status-light  tacb  ?lru  deselect))
                (assert  (tacan-status  ?sys  ?lru  range  deselect))
                (assert  (tacan-status  ?sys  ?lru  bearing  deselect))
            else
                (if  (eq  ?range-fail  on)
                    then
                        (assert  (status-light  tacr  ?lru  fail))
                        (assert  (tacan-status  ?sys  ?lru  range  fail))
                    else
                        (assert  (status-light  tacr  ?lru  ?range-status))
                        (assert  (tacan-status  ?sys  ?lru  range  avail)))
                (if  (eq  ?bearing-fail  on)
                    then
                        (assert  (status-light  tacb  ?lru  fail))
                        (assert  (tacan-status  ?sys  ?lru  bearing  fail))
                    else
                        (assert  (status-light  tacb  ?lru  ?bearing-status))
                        (assert  (tacan-status  ?sys  ?lru  bearing  avail)))))

;---------------------------------------------------------------------------
```

```
(defrule  tacan-deselect

;;      IF
;;              For the engaged system
;;              A TACAN LRU has been available in either range
;;                      or bearing
;;              The deselect flag for that LRU is on
;;      THEN
;;              Notify the operator of crew deselection
;;              Conclude the LRU is unavailable in range and
;;                      bearing due to deselection
;;      END

        (sub-phase  tacan  availability)
        (engaged-system ?sys)
        ?x <- (tacan-status  ?sys  ?lru  range  ?range-status)
        ?y <- (tacan-status  ?sys  ?lru  bearing  ?bearing-status)
        (test  (||  (eq ?range-status avail) (eq ?bearing-status avail)))
        (tacan-flag  ?sys  deselect  ?lru  on)
        =>
        (assert  (event  tacan  off-nominal  alt
                "Crew deselected TACAN " ?lru " in the " ?sys))
        (assert  (status-light  tacr  ?lru  deselect))
        (assert  (status-light  tacb  ?lru  deselect))
        (assert  (tacan-status-changed))
        (retract  ?x)
        (retract  ?y)
        (assert  (tacan-status  ?sys  ?lru  range  deselect))
        (assert  (tacan-status  ?sys  ?lru  bearing  deselect)))

;-----------------------------------------------------------------------------

(defrule  tacan-power-off

;;      IF
;;              For the engaged system
;;              A TACAN LRU was previously powered on
;;              The power indicator for that LRU is now off
;;      THEN
;;              Notify the operator that the LRU has lost power
;;              Conclude the LRU is not available due to loss of power
;;      END

        (sub-phase  tacan  availability)
        (engaged-system ?sys)
        ?x <- (tacan-status  ?sys  ?lru  range  ~power-off)
        ?y <- (tacan-status  ?sys  ?lru  bearing  ~power-off)
        (tacan-flag  ?sys  power  ?lru  off)
        =>
        (assert  (event  tacan  off-nominal  alt
                "TACAN " ?lru " has lost power"))
        (assert  (status-light  tacr  ?lru  off))
        (assert  (status-light  tacb  ?lru  off))
        (assert  (tacan-status-changed))
        (retract  ?x)
        (retract  ?y)
        (assert  (tacan-status  ?sys  ?lru  range  power-off))
        (assert  (tacan-status  ?sys  ?lru  bearing  power-off)))

;-----------------------------------------------------------------------------
```

```
(defrule  tacan-power-on

;;      IF
;;              For the engaged system
;;              A TACAN LRU was previously powered off
;;              The power indicator for that LRU is now on
;;      THEN
;;              Notify the operator that the LRU has been powered on
;;              Conclude the LRU has the status indicated by the fail
;;                      and deselect indicators
;;      END

        (sub-phase  tacan  availability)
        (engaged-system ?sys)
        ?x <- (tacan-status  ?sys  ?lru  range  power-off)
        ?y <- (tacan-status  ?sys  ?lru  bearing  power-off)
        (tacan-flag  system  power  ?lru  on)
        (tacan-flag  ?sys  deselect  ?lru  ?desel-flag)
        (tacan-fail-flag  ?lru  range  ?range-fail)
        (tacan-fail-flag  ?lru  bearing  ?bearing-fail)
        (tacan-lru-quality  ?lru  range  ?range-status)
        (tacan-lru-quality  ?lru  bearing  ?bearing-status)
        =>
        (assert  (event  tacan  off-nominal  alt
                "TACAN " ?lru " has been powered on"))
        (assert  (tacan-status-changed))
        (retract  ?x)
        (retract  ?y)
        (if  (eq  ?desel-flag  on)
            then
                (assert  (status-light  tacr  ?lru  deselect))
                (assert  (status-light  tacb  ?lru  deselect))
                (assert  (tacan-status  ?sys  ?lru  range  deselect))
                (assert  (tacan-status  ?sys  ?lru  bearing  deselect))
            else
                (if  (eq  ?range-fail  on)
                    then
                        (assert  (status-light  tacr  ?lru  fail))
                        (assert  (tacan-status  ?sys  ?lru  range  fail))
                    else
                        (assert  (status-light  tacr  ?lru  ?range-status))
                        (assert  (tacan-status  ?sys  ?lru  range  avail)))
                (if  (eq  ?bearing-fail  on)
                    then
                        (assert  (status-light  tacb  ?lru  fail))
                        (assert  (tacan-status  ?sys  ?lru  bearing  fail))
                    else
                        (assert  (status-light  tacb  ?lru  ?bearing-status))
                        (assert  (tacan-status  ?sys  ?lru  bearing  avail)))))

;-------------------------------------------------------------------------------

(defrule  tacan-failed

;;      IF
;;              For the engaged system
;;              A TACAN LRU measurement was available
;;              The fail flag for that measurement is on
;;      THEN
;;              Notify the operator of the failure
;;              Conclude that the measurement is no longer available
;;                      due to failure
;;      END

        (sub-phase  tacan  availability)
        (engaged-system ?sys)
        ?x <- (tacan-status  ?sys  ?lru  ?measurement  avail)
        (tacan-fail-flag  ?lru  ?measurement  on)
        (measurement-name  ?name&tacr|tacb  ?measurement)
        =>
        (assert  (event  tacan  off-nominal  alt
                "TACAN " ?lru " " ?measurement " failed by RM"))
        (assert  (status-light  ?name  ?lru  fail))
        (assert  (tacan-status-changed))
        (retract  ?x)
        (assert  (tacan-status  ?sys  ?lru  ?measurement  fail)))

;---------------------------------------------------------------------------

(defrule  tacan-reselected

;;      IF
;;              For the engaged system
;;              A TACAN LRU has been unavailable due to
;;                      failure or deselect
;;              The deselect flag for that LRU is off
;;              Both fail flags for that LRU are off
;;      THEN
;;              Notify the operator of crew reselection
;;              Conclude the LRU is now available in range and
;;                      bearing
;;      END

        (sub-phase  tacan  availability)
        (engaged-system ?sys)
        ?x <- (tacan-status  ?sys  ?lru  range  ?range-status)
        ?y <- (tacan-status  ?sys  ?lru  bearing  ?bearing-status)
        (test  (||  (eq ?range-status fail)
                    (eq ?bearing-status fail)
                    (eq ?range-status deselect)
                    (eq ?bearing-status deselect)))
        (tacan-flag  ?sys  deselect  ?lru  off)
        (tacan-fail-flag  ?lru  range  off)
        (tacan-fail-flag  ?lru  bearing  off)
        (tacan-lru-quality  ?lru  range  ?range-quality)
        (tacan-lru-quality  ?lru  bearing  ?bearing-quality)
        =>
        (assert (event tacan off-nominal alt
              "Crew reselected TACAN " ?lru " in the " ?sys))
        (assert  (status-light  tacr  ?lru  ?range-quality))
        (assert  (status-light  tacb  ?lru  ?bearing-quality))
        (assert  (tacan-status-changed))
        (retract  ?x)
        (retract  ?y)
        (assert  (tacan-status  ?sys  ?lru  range  avail))
        (assert  (tacan-status  ?sys  ?lru  bearing  avail)))


;;----------------------------------------------------------------
```

```
(defrule  tacan-locked

;;       IF
;;               For the engaged system
;;               No LRUs were previously locked on
;;               An LRU is locked on a measurement
;;       THEN
;;               Notify the operator that TACAN is locking on
;;       END

         (sub-phase  tacan  availability)
         ?x <- (prev-tacan-lock  ?measurement  off)
         (tacan-lock  ?lru  ?measurement  on)
         =>
         (assert  (event  tacan  nominal  alt
                 "TACAN " ?lru " is locking onto " ?measurement))
         (assert  (tacan-status-changed))
         (retract  ?x)
         (assert  (prev-tacan-lock  ?measurement  on)))

;-------------------------------------------------------------------------------

(defrule  tacan-no-locked

;;       IF
;;               An LRU was previously locked on a measurement
;;               No LRU is locked on a measurement
;;       THEN
;;               Nofity the operator that TACAN lost lock
;;       END

         (sub-phase  tacan  availability)
         ?x <- (prev-tacan-lock  ?measurement  on)
         (tacan-lock  1  ?measurement  off)
         (tacan-lock  2  ?measurement  off)
         (tacan-lock  3  ?measurement  off)
         =>
         (assert  (event  tacan  nominal  alt
                 "TACAN lost lock on " ?measurement))
         (assert  (tacan-status-changed))
         (retract  ?x)
         (assert  (prev-tacan-lock  ?measurement  off)))


;;*****************************************************************************
;;
;;;  GROUP
;;       TACAN LRU Quality     (3.8.3)
;;
;;       This group checks LRU measurement errors to determine which LRUs
;;       have a problem and what the problem is.
;;
;;;  CONTROL FACTS
;        (sub-phase  tacan  quality)
;;
;;;  CONTAINING GROUP
;;       TACAN
;;
```

```
;;*************************************************************************

(defrule tacan-cone-of-confusion-on-ignore-bearing

;;      IF
;;          In the cone of confusion
;;       THEN
;;          Ignore bearing measurements

        (declare (salience 10))
        (sub-phase tacan quality-rating)
        (cone on)
=>
        (assert (temporary-rating 1 bearing none))
        (assert (temporary-rating 2 bearing none))
        (assert (temporary-rating 3 bearing none)))

;;-----------------------------------------------------------------

(defrule tacan-no-quality-due-to-channel-change

;;      IF
;;          An LRU is tuned to a different channel than it was previously
;;      THEN
;;          That LRU has no quality rating for range or bearing

        (declare (salience 10))
        (sub-phase tacan quality-rating)
        (tacan-channel ?lru ?channel)
        ?x <- (prev-tacan-channel ?lru ~?channel)
=>
        (retract ?x)
        (assert (temporary-rating ?lru bearing none))
        (assert (temporary-rating ?lru range none))
        (assert (prev-tacan-channel ?lru ?channel)))

;;-----------------------------------------------------------------

(defrule tacan-use-gnd-minus-ob-errors

;;      IF
;;          The HSTD is good
;;      THEN
;;          The selected errors for each measurement are the
;;                 GND-Onboard errors

        (declare (salience 9))
        (sub-phase tacan quality-rating)
        (hstd good)
        (tacan-error ?lru ?measurement slope ?status-s)
        (tacan-error ?lru ?measurement bias ?status-b)
        (tacan-error ?lru ?measurement noise ?status-n)
=>
        (assert (selected-tacan-error ?lru ?measurement slope ?status-s))
        (assert (selected-tacan-error ?lru ?measurement bias ?status-b))
        (assert (selected-tacan-error ?lru ?measurement noise ?status-n)))

;;-----------------------------------------------------------------

(defrule tacan-use-relative-errors
;;     IF
;;         The HSTD is not good
;;     THEN
;;         The selected errors for each measurement are the relative
;;                 errors

      (declare (salience 9))
      (sub-phase tacan quality-rating)
      (hstd ~good)
      (rel-tac ?pair-a ?measurement ?error ?status-a)
      (rel-tac ?pair-b&~?pair-a ?measurement ?error ?status-b)
      (common-lru ?pair-a ?pair-b ?lru)
      (min-miscompare ?status-a ?status-b ?best-status)
      (not (selected-tacan-error ?lru ?measurement ?error ?))
=>
      (assert (selected-tacan-error ?lru ?measurement ?error
                                    ?best-status)))

;;-----------------------------------------------------------

(defrule tacan-no-quality-rating-part-1

;;     IF
;;         The hstd is good
;;         For the engaged system
;;         A TACAN LRU is commfaulted or unlocked in the measurement
;;     THEN
;;         Set temporary rating to NONE

      (declare (salience 8))
      (sub-phase tacan quality-rating)
      (hstd good)
      (engaged-system ?sys)
      (or (tacan-status ?sys ?lru ?measurement commfault)
          (tacan-lock ?lru ?measurement off))
      (not (temporary-rating ?lru ?measurement ?))
=>
      (assert (temporary-rating ?lru ?measurement none)))

;;------------------------------------------------------------

(defrule tacan-no-quality-rating-part-2

;;     IF
;;         The HSTD is not good
;;         For the engaged system
;;         A measurement from LRU A is commfaulted or unlocked
;;         The same measurement from LRU B is commfaulted or unlocked
;;     THEN
;;         Set temporary rating to none

      (declare (salience 8))
      (sub-phase tacan quality-rating)
      (engaged-system ?sys)
      (hstd ~good)
      (or (tacan-status ?sys ?lru-a ?measurement commfault)
          (tacan-lock ?lru-a ?measurement off))
      (or (tacan-status ?sys ?lru-b&~?lru-a ?measurement commfault)
          (tacan-lock ?lru-b&~?lru-a ?measurement off))
      (lrus-in-pair ?pair ?lru-a ?lru-b)
      (excluded-lru ?pair ?lru-desired)
      (not (temporary-rating ?lru-desired ?measurement ?))
=>
      (assert (temporary-rating ?lru-desired ?measurement none)))

;;----------------------------------------------------------------

(defrule tacan-temporary-quality-for-noise-bias-slope

;;      IF
;;          An LRU has a particular rating based on considering
;;                selected errors of noise, bias, and slope
;;      THEN
;;          Conclude that the LRU has that rating

      (declare (salience 7))
      (sub-phase tacan quality-rating)
      (selected-tacan-error ?lru ?measurement slope ?s-quality)
      (selected-tacan-error ?lru ?measurement bias ?b-quality)
      (selected-tacan-error ?lru ?measurement noise ?n-quality)
      (not (temporary-rating ?lru ?measurement ?))
      (tacan-quality ?s-quality ?b-quality ?n-quality ?total-quality)
=>
      (assert (temporary-rating ?lru ?measurement ?total-quality)))

;;-------------------------------------------------------------

(defrule tacan-determine-lru-rating-part-1

;;      IF
;;          HSTD is good
;;      THEN
;;          Measurement rating = temporary rating
;;          Potential dilemma flag = off

      (declare (salience 6))
      (sub-phase tacan quality)
      (hstd good)
      ?x <- (temporary-rating ?lru ?measurement ?rating)
=>
      (retract ?x)
      (assert (tacan-lru-quality ?lru ?measurement ?rating))
      (assert (potential-dilemma-flag ?lru ?measurement off)))

;;--------------------------------------------------------------

(defrule tacan-determine-lru-rating-part-2

;;      IF
;;          For the engaged system
;;          The HSTD is not good
;;          All three measurements available and locked
;;      THEN
;;          A's measurement rating = better rating (of good,
;;              suspect, or bad) between temporary ratings for
;;              AB and AC's relative errors
;;          Potential dilemma flag = off

      (declare (salience 6))
```

```
        (sub-phase tacan quality)
        (engaged-system ?sys)
        (hstd ~good)
        (tacan-status ?sys ?lru-a ?measurement avail)
        (tacan-lock ?lru-a ?measurement on)
        (tacan-status ?sys ?lru-b&~?lru-a ?measurement avail)
        (tacan-lock ?lru-b ?measurement on)
        (tacan-status ?sys ?lru-c&~?lru-b&~?lru-a ?measurement avail)
        (tacan-lock ?lru-c ?measurement on)
        (lrus-in-pair ?pair-ab ?lru-a ?lru-b)
        (lrus-in-pair ?pair-ac ?lru-a ?lru-c)
        (temporary-rating ?lru-b ?measurement ?rating-b)
        (temporary-rating ?lru-c ?measurement ?rating-c)
        (min-compare ?rating-b ?rating-c ?best)
        (not (potential-dilemma-flag ?lru-a ?measurement ?))
        ?x <- (tacan-lru-quality ?lru-a ?measurement ?)
=>
        (retract ?x)
        (assert (tacan-lru-quality ?lru-a ?measurement ?best))
        (assert (potential-dilemma-flag ?lru-a ?measurement off)))

;;----------------------------------------------------------------

(defrule tacan-determine-lru-rating-part-3

;;      IF
;;          For the engaged system
;;          The HSTD is not good
;;          Two measurements are available and locked
;;          Both measurement's previous ratings are equal
;;      THEN
;;          Measurement rating for both measurements = temporary
;;                rating for their relative error
;;          Set potential dilemma flag to ON
;;      END

        (declare (salience 6))
        (sub-phase tacan quality)
        (hstd ~good)
        (engaged-system ?sys)
        (tacan-status ?sys ?lru-a ?measurement avail)
        (tacan-lock ?lru-a ?measurement on)
        (tacan-status ?sys ?lru-b&~?lru-a ?measurement avail)
        (tacan-lock ?lru-b ?measurement on)
        (or (tacan-status ?sys ?lru-c&~?lru-b&~?lru-a ?measurement ~avail)
            (tacan-lock ?lru-c&~?lru-b&~?lru-a ?measurement off))
        ?x <- (tacan-lru-quality ?lru-a ?measurement ?rating-a)
        ?y <- (tacan-lru-quality ?lru-b ?measurement ?rating-a)
        (not (potential-dilemma-flag ?lru-a ?measurement ?))
        (not (potential-dilemma-flag ?lru-b ?measurement ?))
        (temporary-rating ?lru-a ?measurement ?trating-a)
        (temporary-rating ?lru-b ?measurement ?trating-b)
=>
        (retract ?x ?y)
        (assert (tacan-lru-quality ?lru-a ?measurement ?trating-a))
        (assert (tacan-lru-quality ?lru-b ?measurement ?trating-b))
        (assert (potential-dilemma-flag ?lru-a ?measurement on))
        (assert (potential-dilemma-flag ?lru-b ?measurement on)))

;;----------------------------------------------------------------
```

```
(defrule tacan-determine-lru-rating-part-4

;;      IF
;;          For the engaged system
;;          The HSTD is not good
;;          Two measurements (A + B) are available and locked
;;          Measurement A previous rating is better than
;;             measurement B previous rating
;;      THEN
;;          Set measurement A rating = previous measurement A
;;             rating
;;          Set measurement B rating = temporary rating for the
;;             AB relative error
;;          Set potential dilemma flag to OFF
;;      END

        (declare (salience 6))
        (sub-phase tacan quality)
        (hstd ~good)
        (engaged-system ?sys)
        (tacan-status ?sys ?lru-a ?measurement avail)
        (tacan-lock ?lru-a ?measurement on)
        (tacan-status ?sys ?lru-b&~?lru-a ?measurement avail)
        (tacan-lock ?lru-b ?measurement on)
        (or (tacan-status ?sys ?lru-c&~?lru-b&~?lru-a ?measurement ~avail)
            (tacan-lock ?lru-c&~?lru-b&~?lru-a ?measurement off))
        (tacan-lru-quality ?lru-a ?measurement ?rating-a)
        ?x <- (tacan-lru-quality ?lru-b ?measurement ?rating-b)
        (min-miscompare ?rating-a ?rating-b ?rating-a)
        (not (potential-dilemma-flag ?lru-a ?measurement ?))
        (not (potential-dilemma-flag ?lru-b ?measurement ?))
        (temporary-rating ?lru-b ?measurement ?status-rel)
=>
        (retract ?x)
        (assert (tacan-lru-quality ?lru-b ?measurement ?status-rel))
        (assert (potential-dilemma-flag ?lru-a ?measurement off))
        (assert (potential-dilemma-flag ?lru-b ?measurement off)))

;;------------------------------------------------------------

(defrule tacan-determine-lru-rating-part-5

;;      IF
;;          For the engaged system
;;          The HSTD is not good
;;          Only measurement A is available and locked
;;          Measurement A's previous rating = none
;;          A's raw data noise (spread) is greater than 1/2
;;             RM threshold
;;      THEN
;;          A's measurement rating for = Noise
;;          Set potential dilemma flag to OFF
;;      END

        (declare (salience 6))
        (sub-phase tacan quality)
        (hstd ~good)
        (engaged-system ?sys)
```

```
      (tacan-status ?sys ?lru-a ?measurement avail)
      (tacan-lock ?lru-a ?measurement on)
      (or (tacan-status ?sys ?lru-b&~?lru-a ?measurement ~avail)
          (tacan-lock ?lru-b&~?lru-a ?measurement off))
      (or (tacan-status ?sys ?lru-c&~?lru-b&~?lru-a ?measurement ~avail)
          (tacan-lock ?lru-c&~?lru-b&~?lru-a ?measurement off))
      ?x <- (tacan-lru-quality ?lru-a ?measurement none)
      (selected-error ?lru-a ?measurement noise o50|over)
=>
      (retract ?x)
      (assert (tacan-lru-quality ?lru-a ?measurement noise))
      (assert (potential-dilemma-flag ?lru-a ?measurement off)))

;;-------------------------------------------------------------

(defrule tacan-quality-rating-change

;;      IF
;;           A measurement rating has changed
;;      THEN
;;           Notify the operator of the change and potential
;;             dilemma condition based on the potential
;;             dilemma flag status
;;      END

      (declare (salience 5))
      (sub-phase tacan quality)
      ?x <- (last-tacan-quality ?lru-a ?measurement ?old)
      (tacan-lru-quality ?lru-a ?measurement ?new&~?old)
      (potential-dilemma-flag ?lru-a ?measurement ?flag)
      (measurement-name ?name&tacr|tacb ?measurement)
=>
      (retract ?x)
      (assert (last-tacan-quality ?lru-a ?measurement ?new))
      (assert (event tacan off-nominal alt
                "Tacan " ?lru-a ?measurement
                " quality has changed from " ?old " to " ?new))
      (assert (status-light ?name ?lru-a ?new))
      (if (eq ?flag on)
          then
            (assert (event tacan off-nominal alt
                      "ONAV can't determine which TACAN LRU"
                      " caused the TACAN " ?lru-a " "
                      ?measurement " quality change"))))

;;-------------------------------------------------------------

(defrule tacan-dilemma-cleanup

      (declare (salience 4))
      (sub-phase tacan clean-up)
      ?x <- (potential-dilemma-flag ? ? ?)

=>

      (retract ?x))

;;-------------------------------------------------------------
```

```
(defrule tacan-temporary-rating-cleanup

      (declare (salience 4))
      (sub-phase tacan clean-up)
      ?x <- (temporary-rating ? ? ?)

=>

      (retract ?x))

;;**************************************************************************
;;
;;;  GROUP
;;      TACAN Filter Flag Changes    (3.8.4)
;;
;;      This group watches for changes in the TACAN data-good flags and
;;      filter flags.
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  watch-flags)
;;
;;;  CONTAINING GROUP
;;      TACAN
;;
;;**************************************************************************

(defrule  tacan-filter-flag-changed

;;      IF
;;              For the engaged system
;;              The current value of a TACAN filter flag is anything but
;;                      off  AND
;;              The value of the flag is different from its previous value
;;      THEN
;;              Note the new value
;;              Notify the operator if the new value is "process"
;;      END

        (sub-phase  tacan  watch-flags)
        (engaged-system ?sys)
        (filter-flag  ?sys  ?meas&tacr|tacb  ?flag&~off)
        ?x <- (prev-filter-flag  ?sys  ?meas  ~?flag)
        (measurement-name  ?meas  ?measurement)
        =>
        (retract  ?x)
        (assert  (prev-filter-flag  ?sys  ?meas  ?flag))
        (if  (eq  ?flag  process)
            then
                (assert  (event  tacan  nominal  alt
                        "Processing TACAN " ?measurement))))

;-------------------------------------------------------------------------

(defrule  tacan-end-measurement-processing

;;      IF
;;              For the engaged system
;;              The current value of a TACAN filter flag is off  AND
;;              The previous value is not off  AND
;;              Either
;;                      The corresponding data good flag is off  OR
;;                      MSBLS is being processed
;;      THEN
;;              Conclude and indicate that the processing of
;;                TACAN measurement has ended
;;      END

        (sub-phase  tacan  watch-flags)
        (engaged-system ?sys)
        (filter-flag  ?sys  ?meas&tacr|tacb  off)
        ?x <- (prev-filter-flag  ?sys  ?meas  ~off)
        (measurement-name  ?meas  ?measurement)
        (or  (data-good  ?sys  ?meas  off)
             (filter-flag  ?sys  mlsr|mlsa|mlse  process|edit))
        =>
        (retract  ?x)
        (assert  (prev-filter-flag  ?sys  ?meas  off))
        (assert  (event  tacan  nominal  alt
                "TACAN " ?measurement " processing turned off " ))
        (assert  (status-light ?meas 1 off))
        (assert  (status-light ?meas 2 off))
        (assert  (status-light ?meas 3 off)))

;------------------------------------------------------------------------------

(defrule  tacan-data-good-flag-changed

;;      IF
;;              For the engaged system
;;              The current value of a TACAN data-good flag is different from
;;                      its previous value
;;      THEN
;;              Notify the operator of the new value
;;      END

        (sub-phase  tacan  watch-flags)
        (engaged-system ?sys)
        (data-good  ?sys  ?meas&tacr|tacb  ?flag)
        ?x <- (prev-data-good  ?sys  ?meas  ~?flag)
        (measurement-name  ?meas  ?measurement)
        =>
        (retract  ?x)
        (assert  (prev-data-good  ?sys  ?meas  ?flag))
        (assert  (event  tacan  nominal  alt
                "TACAN " ?measurement " data-good flag is " ?flag)))

;------------------------------------------------------------------------------

(defrule  tacan-dilemma-occurred

;;      IF
;;              For the engaged system
;;              TACAN dilemma flag is on for either measurement
;;      THEN
;;              Warn the operator that a TACAN dilemma ocurred
;;      END

        (sub-phase  tacan  watch-flags)
        (engaged-system ?sys)
        (tacan-dilemma  ?measurement  on)
        =>
        (assert  (event  tacan  off-nominal  alt
                "TACAN " ?measurement " is in dilemma")))



;;************************************************************************
;;
;;;  GROUP
;;      Toggle Tacan Recommendations (3.8.5)
;;
;;      This group determines whether or not the TACAN ground station has
;;      a problem.  If so, and if a backup is available, then toggling
;;      is recommended.
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  toggle)
;;
;;;  CONTAINING GROUP
;;      TACAN
;;
;;************************************************************************

(defrule tacan-gnd-station-problem-1

;;      IF
;;              For the engaged system
;;              At least 2 LRUs are locked onto the same measurement  AND
;;              All locked LRUs are exhibiting the same problem
;;      THEN
;;              Conclude the ground station has a problem and a toggle
;;                is needed
;;      END

        (sub-phase  tacan  toggle)
        (engaged-system ?sys)
        (tacan-lock  ?lru-a  ?measurement  on)
        (tacan-lru-quality  ?lru-a  ?measurement  ?status&noise|bias)
        (tacan-lock  ?lru-b&~?lru-a  ?measurement  on)
        (tacan-lru-quality  ?lru-b  ?measurement  ?status)
        (or  (tacan-lock  ?lru-c  ?measurement  off)
             (tacan-status  ?sys  ?lru-c&~?lru-a&~?lru-b  ?measurement  ~avail)
             (and  (tacan-lock  ?lru-c&~?lru-b&~?lru-a  ?measurement  on)
                   (tacan-status  ?sys  ?lru-c  ?measurement  avail)
                   (tacan-lru-quality  ?lru-c  ?measurement  ?status)))
        =>
        (assert  (event  tacan  off-nominal  alt
                "All locked TACAN LRUs have a " ?measurement
                   " " ?status))
        (assert  (need-a-toggle)))

;-------------------------------------------------------------------------

(defrule  tacan-gnd-station-problem-2

;;      IF
;;              For the engaged system
;;              Only 1 LRU is available  AND
;;                That LRU is locked  AND
;;                That LRU has an error
;;        THEN
;;                Notify the operator that the ground station has a problem
;;                Conclude a toggle is needed
;;        END

        (sub-phase  tacan  toggle)
        (engaged-system ?sys)
        (tacan-status ?sys ?lru-a ?measurement avail)
        (tacan-lock  ?lru-a  ?measurement  on)
        (tacan-lru-quality  ?lru-a  ?measurement  ?status&noise|bias)
        (tacan-status  ?sys  ?lru-b  ?measurement  ~avail)
        (tacan-status  ?sys  ?lru-c&~?lru-b  ?measurement  ~avail)
        =>
        (assert (event tacan off-nominal alt
                 "locked LRU has a " ?measurement " " ?status))
        (assert  (need-a-toggle)))

;------------------------------------------------------------------------

(defrule  tacan-one-locked-at-130k

;;        IF
;;                Only one LRU is locked  AND
;;                That LRU has an error  AND
;;                The altitude is less than 130 kft and greater than 5 kft
;;        THEN
;;                Notify the operator that the ground station has a problem
;;                Conclude a toggle is needed
;;        END

        (sub-phase  tacan  toggle)
        (tacan-lock  ?lru-a  ?measurement  on)
        (tacan-lru-quality  ?lru-a  ?measurement  ?status&noise|bias)
        (tacan-lock  ?lru-b  ?measurement  off)
        (tacan-lock  ?lru-c&~?lru-b  ?measurement  off)
        (altitude  ?alt)
        (test  (< ?alt 130000))
        (test  (> ?alt 5000))
        =>
        (assert (event tacan off-nominal alt
                 "locked LRU has a " ?measurement " " ?status
                 " at altitude less than 130k ft"))
        (assert  (need-a-toggle)))

;------------------------------------------------------------------------

(defrule  tacan-none-locked-at-130k

;;        IF
;;                No LRUs are locked  AND
;;                The altitude is less than 130 kft and greater than 5 kft
;;        THEN
;;                Notify the operator that the ground station has a problem
;;                Conclude a toggle is needed
;;        END

        (sub-phase  tacan  toggle)
        (tacan-lock  1  ?measurement  off)
        (tacan-lock  2  ?measurement  off)
        (tacan-lock  3  ?measurement  off)
        (altitude  ?alt)
        (test  (< ?alt 130000))
        (test  (> ?alt 5000))
        =>
        (assert (event tacan off-nominal alt
                 "No LRU's are locked in " ?measurement
                 " at altitude less than 130k ft"))
        (assert  (need-a-toggle)))

;-----------------------------------------------------------------------

(defrule  tacan-do-a-toggle

;;      IF
;;              A toggle is needed  AND
;;              Toggle capability is available
;;      THEN
;;              Request a toggle
;;      END

        ?x <- (need-a-toggle)
        (toggle-available  yes)
        (desired-tacan  ?current-slot)
        (same-area  ?current-slot  ?new-slot)
        =>
        (bind ?channel (lookup-tacan ?new-slot))
        (retract  ?x)
        (assert  (recommend  tacan  toggle  off-nominal  alt
                 "Need" " to toggle TACAN to " ?channel
                 " please confirm")))

;-----------------------------------------------------------------------

(defrule  tacan-dont-do-a-toggle

;;      IF
;;              A toggle is needed  AND
;;              Toggle capability is not available
;;      THEN
;;              Don't do the toggle
;;      END

        ?x <- (need-a-toggle)
        (toggle-available  no)
        =>
        (retract  ?x))

;;*************************************************************************
;;
;;;  GROUP
;;      LRU's for Deselect (3.8.6.1)
;;
;;      This group looks at problems with the LRUs to determine which
;;      ones might need to be deselected.
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  deselect)
;;
```

```
;;;  CONTAINING GROUP
;;      Deselect TACAN LRU
;;
;;*****************************************************************************

(defrule  tacan-kill-old-suggestion

;;       IF
;;               TACAN status has changed  AND
;;               Part of an old deselect suggestion still exists
;;       THEN
;;               Remove that part of the deselect suggestion
;;       END

         (declare  (salience  10))
         (sub-phase  tacan  deselect)
         (tacan-status-changed)
         ?x <- (suggested-deselect  $?)
         =>
         (retract  ?x))

;-----------------------------------------------------------------------------

(defrule  tacan-dsel-prep-done

;;       IF
;;               TACAN status has changed  AND
;;               No previous deselect suggestion exists
;;       THEN
;;               Remove the note about the TACAN status changing
;;       END

         (declare  (salience  10))
         (sub-phase  tacan  deselect)
         ?x <- (tacan-status-changed)
         (not  (suggested-deselect $?))
         =>
         (retract  ?x))

;-----------------------------------------------------------------------------
(defrule  tacan-dilemma

;;       IF
;;               For the engaged system
;;               TACAN RM is in dilemma  AND
;;               One LRU is known to be bad  AND
;;               Another LRU is known to be good
;;       THEN
;;               Try deselecting the bad LRU
;;       END

         (sub-phase  tacan  deselect)
         (engaged-system ?sys)
         (tacan-dilemma  ?measurement  on)
         (tacan-status  ?sys  ?lru-a  ?measurement  avail)
         (tacan-lru-quality  ?lru-a  ?measurement  noise|bias)
         (tacan-status  ?sys  ?lru-b&~?lru-a  ?measurement  avail)
         (tacan-lru-quality  ?lru-b  ?measurement  good)
         =>
         (assert  (need-to-deselect  ?lru-a)))
```

```
;-----------------------------------------------------------------------------

(defrule  tacan-two-against-one

;;      IF
;;              Two LRUs have a problem  AND
;;              The third LRU is good  AND
;;              The problem with the two bad LRUs is such that TACAN RM
;;                      may fail the good LRU
;;      THEN
;;              Try deselecting the two bad LRUs
;;      END

        (sub-phase  tacan  deselect)
        (tacan-lru-quality  ?lru-a  ?measurement  bias)
        (tacan-lru-quality  ?lru-b&~?lru-a  ?measurement  bias)
        (tacan-lru-quality  ?lru-c  ?measurement  good)
        (lrus-in-pair  ?pair  ?lru-a  ?lru-b)
        (rel-tac  ?pair  ?measurement  bias  under)
        =>
        (assert  (need-to-deselect  ?lru-a))
        (assert  (need-to-deselect  ?lru-b)))

;-----------------------------------------------------------------------------

(defrule  tacan-not-2-locked

;;      IF
;;              For the engaged system
;;              2 LRUs are not locked  AND
;;              1 LRU is locked  AND
;;              The data good flag is off  AND
;;              The altitude is less than 130 kft and greater than 5 kft
;;      THEN
;;              Try deselecting the 2 unlocked LRUs
;;      END

        (sub-phase  tacan  deselect)
        (engaged-system ?sys)
        (tacan-lock  ?lru-a  ?measurement  off)
        (tacan-lock  ?lru-b&~?lru-a  ?measurement  off)
        (tacan-lock  ?lru-c  ?measurement  on)
        (tacan-lru-quality ?lru-c ?measurement good)
        (measurement-name  ?meas&tacr|tacb  ?measurement)
        (data-good  ?sys  ?meas  off)
        (altitude  ?alt)
        (test  (< ?alt 130000))
        (test  (> ?alt 5000))
        =>
        (assert  (need-to-deselect  ?lru-a))
        (assert  (need-to-deselect  ?lru-b)))

;-----------------------------------------------------------------------------

(defrule  tacan-noisy-lru

;;      IF
;;              An LRU has excessive noise
;;      THEN
;;              Try deselecting that LRU
;;      END

        (sub-phase  tacan  deselect)
        (tacan-lru-quality  ?lru  ?measurement  noise)
        =>
        (assert  (need-to-deselect  ?lru)))

;------------------------------------------------------------------------

(defrule  tacan-rm-failed-wrong-lru

;;      IF
;;              For the engaged system
;;              One LRU has a problem  AND
;;              Another LRU is good  AND
;;              TACAN RM has failed the good one
;;      THEN
;;              Try deselecting the bad one
;;      END

        (sub-phase  tacan  deselect)
        (engaged-system ?sys)
        (tacan-lru-quality  ?lru-a  ?measurement  bias|noise)
        (tacan-status  ?sys  ?lru-b&~?lru-a  ?measurement  fail)
        (tacan-lru-quality  ?lru-b  ?measurement  good)
        =>
        (assert  (need-to-deselect  ?lru-a)))

;------------------------------------------------------------------------


(defrule  tacan-deselect-the-lru-due-to-no-go

;;      IF
;;              The selected measurement from RM is not good
;;                enough to go for tacan
;;              Deselecting an LRU will remedy the situation
;;      THEN
;;              Recommend deselection of the LRU
;;      END

        (sub-phase  tacan  deselect)
        (tacan-error  ?lru-a  ?measurement  raw over)
        (tacan-lock ?lru-a  ?measurement  on)
        (tacan-error  ?lrub&~?lru-a  ?measurement  raw under)
        (tacan-lock ?lru-b  ?measurement  on)
        (not  (need-to-deselect  ?lru-a))
        =>
        (assert  (need-to-deselect  ?lru-a)))


;;**************************************************************************
;;
;;;  GROUP
;;      Deselect Configurations (3.8.6.2)
;;
;;      This group takes the initial suggestions from the previous group
;;      and determines which deselect combinations should be tried.  Each
;;       combination is proposed as a separate configuration.  There are
;;       up to seven possible combinations.
;;
;;;  CONTROL FACTS
;        (sub-phase  tacan  deselect)
;;
;;;  CONTAINING GROUP
;;       Deselect TACAN LRU
;;
;;*************************************************************************

(defrule  tacan-try-zero-deselects

;;       IF
;;                 Any LRUs have been proposed for deselection
;;       THEN
;;                 Propose a configuration where no LRUs are deselected
;;                         (i.e. the onboard configuration is left like it is)
;;       END

         (sub-phase  tacan  deselect)
         (need-to-deselect  $?)
         =>
         (bind  ?config  (gensym))
         (assert  (number-deselected  ?config  0))
         (assert  (possible-tacan-configuration  ?config  1  off))
         (assert  (possible-tacan-configuration  ?config  2  off))
         (assert  (possible-tacan-configuration  ?config  3  off)))

;------------------------------------------------------------------------

(defrule  tacan-try-one-deselect

;;       IF
;;                 An LRU has been proposed for deselection
;;       THEN
;;                 Propose a configuration where that LRU is the only one
;;                         deselected
;;       END

         (sub-phase  tacan  deselect)
         (need-to-deselect  ?lru-a)
         (lrus-in-pair  ?  ?lru-a  ?lru-b)
         (lrus-in-pair  ?  ?lru-a  ?lru-c&~?lru-b)
         =>
         (bind  ?config  (gensym))
         (assert  (number-deselected  ?config  1))
         (assert  (possible-tacan-configuration  ?config  ?lru-a  on))
         (assert  (possible-tacan-configuration  ?config  ?lru-b  off))
         (assert  (possible-tacan-configuration  ?config  ?lru-c  off)))

;------------------------------------------------------------------------
(defrule  tacan-try-two-deselects

;;       IF
;;                 For the engaged system
;;                 An LRU has been proposed for deselection  AND
;;                 Another LRU is not commfaulted, deselected, or powered off
;;       THEN
;;                 Propose a configuration where both LRUs are deselected
;;      END

        (sub-phase  tacan  deselect)
        (engaged-system ?sys)
        (need-to-deselect  ?lru-a)
        (lrus-in-pair  ?  ?lru-a  ?lru-b)
        (lrus-in-pair  ?  ?lru-a  ?lru-c&~?lru-b)
        (tacan-status  ?sys  ?lru-b  range  ~commfault&~deselect&~power-off)
        =>
        (bind  ?config  (gensym))
        (assert  (number-deselected  ?config  2))
        (assert  (possible-tacan-configuration  ?config  ?lru-a  on))
        (assert  (possible-tacan-configuration  ?config  ?lru-b  on))
        (assert  (possible-tacan-configuration  ?config  ?lru-c  off)))

;-----------------------------------------------------------------------------

(defrule  tacan-eliminate-duplicate-configurations

;;      IF
;;              Two proposed configurations are identical
;;      THEN
;;              Eliminate one of the proposed configurations
;;      END

        (declare  (salience  5))
        (sub-phase  tacan  deselect)
        ?x1 <- (possible-tacan-configuration  ?config-a  1  ?dsel-1)
        ?x2 <- (possible-tacan-configuration  ?config-a  2  ?dsel-2)
        ?x3 <- (possible-tacan-configuration  ?config-a  3  ?dsel-3)
        ?x4 <- (number-deselected  ?config-a  $?)
        (possible-tacan-configuration  ?config-b&~?config-a  1  ?dsel-1)
        (possible-tacan-configuration  ?config-b  2  ?dsel-2)
        (possible-tacan-configuration  ?config-b  3  ?dsel-3)
        =>
        (retract  ?x1  ?x2  ?x3  ?x4))



;;****************************************************************************
;;
;;;  GROUP
;;      Predict Availability and Configuration Data (3.8.6.3 & 3.8.6.4)
;;
;;      This group of rules predicts how TACAN RM will respond to a proposed
;;      deselection configuration.  This prediction consists of the bias
;;      and noise on the selected range and bearing measurements, the range
;;      and bearing data good flags, and the range and bearing dilemma
;;      indicators.
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  deselect)
;;
;;;  CONTAINING GROUP
;;      Deselect TACAN LRU
;;
;;****************************************************************************

(defrule  tacan-predict-available
;;        IF
;;                 For the engaged system
;;                 An LRU is not deselected in a proposed configuration  AND
;;                 That LRU is available in the real world
;;        THEN
;;                 Predict that the LRU will be available in the proposed
;;                          configuration
;;        END

          (sub-phase  tacan  deselect)
          (engaged-system ?sys)
          (possible-tacan-configuration  ?config  ?lru  off)
          (tacan-status  ?sys  ?lru  ?measurement  avail)
          =>
          (assert (predicted-tacan  status  ?config  ?lru  ?measurement  avail)))

;-------------------------------------------------------------------------------

(defrule  tacan-predict-not-available-1

;;        IF
;;                 An LRU is deselected in a proposed configuration
;;        THEN
;;                 Predict that the LRU will not be available in the proposed
;;                          configuration
;;        END

          (sub-phase  tacan  deselect)
          (possible-tacan-configuration  ?config  ?lru  on)
          =>
          (assert  (predicted-tacan  status  ?config  ?lru  range  not-avail))
          (assert  (predicted-tacan  status  ?config  ?lru  bearing  not-avail)))

;-------------------------------------------------------------------------------

(defrule  tacan-predict-not-available-2

;;        IF
;;                 For the engaged system
;;                 An LRU is not available in the real world
;;        THEN
;;                 Predict that the LRU will not be available in any proposed
;;                          configuration
;;        END

          (sub-phase  tacan  deselect)
          (engaged-system ?sys)
          (possible-tacan-configuration  ?config  ?lru  ?)
          (tacan-status  ?sys  ?lru  ?measurement  ~avail)
          =>
          (assert
             (predicted-tacan  status  ?config  ?lru  ?measurement  not-avail)))

;-------------------------------------------------------------------------------

(defrule  tacan-predict-data-good-two-locked

;;        IF
;;                 Two LRUs are available in a proposed configuration  AND
;;              Both LRUs are currently locked onto a measurement
;;      THEN
;;              Predict the data good flag for that measurement will be
;;                      on in the proposed configuration
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  status  ?config  ?lru-a  ?measurement  avail)
        (predicted-tacan  status  ?config  ?lru-b&~?lru-a  ?measurement  avail)
        (tacan-lock  ?lru-a  ?measurement  on)
        (tacan-lock  ?lru-b  ?measurement  on)
        =>
        (assert  (predicted-tacan  data-good  ?config  ?measurement  on)))

;-------------------------------------------------------------------------------

(defrule  tacan-predict-data-good-one-locked

;;      IF
;;              At least one LRU is available in a proposed configuration  AND
;;              That LRU is locked onto a measurement  AND
;;              The two-lock flag for that measurement is off
;;      THEN
;;              Predict the data good flag for that measurement will be
;;                      on in the proposed configuration
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  status  ?config  ?lru  ?measurement  avail)
        (tacan-lock  ?lru  ?measurement  on)
        (two-lock-flag  ?measurement  off)
        =>
        (assert  (predicted-tacan  data-good  ?config  ?measurement  on)))

;-------------------------------------------------------------------------------

(defrule  tacan-predict-data-good-one-avail

;;      IF
;;              Only one LRU is available in a proposed configuration  AND
;;              That LRU is locked onto a measurement
;;      THEN
;;              Predict the data good flag for that measurement will be
;;                      on in the proposed configuration
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  status  ?config  ?lru-a  ?measurement  avail)
        (predicted-tacan  status  ?config  ?lru-b  ?measurement  not-avail)
        (predicted-tacan  status  ?config  ?lru-c&~?lru-b
                                                ?measurement  not-avail)
        (tacan-lock  ?lru-a  ?measurement  on)
        =>
        (assert  (predicted-tacan  data-good  ?config  ?measurement  on)))

;-------------------------------------------------------------------------------

(defrule  tacan-predict-data-good-off

;;      IF
;;              No rule has predicted the data good flag for a measurement
;;                      will be on in a proposed configuration
;;      THEN
;;              Predict the data good flag for that measurement will be off
;;                      in the proposed configuration
;;      END

        (declare  (salience  -1))
        (sub-phase  tacan  deselect)
        (predicted-tacan  status  ?config  ?  ?measurement  ?)
        (not  (predicted-tacan  data-good  ?config  ?measurement  ?))
        =>
        (assert  (predicted-tacan  data-good  ?config  ?measurement  off)))

;-----------------------------------------------------------------------------

(defrule  tacan-predict-dilemma

;;      IF
;;              Exactly two LRUs are available and locked for a measurement
;;                      in a proposed configuration  AND
;;              The relative bias between the two LRUs exceeds the RM
;;                      threshold
;;      THEN
;;              Predict that RM will declare a dilemma in the proposed
;;                      configuration
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  status  ?config  ?lru-a  ?measurement  avail)
        (tacan-lock  ?lru-a  ?measurement  on)
        (predicted-tacan  status  ?config  ?lru-b&~?lru-a  ?measurement  avail)
        (tacan-lock  ?lru-b  ?measurement  on)
        (lrus-in-pair  ?pair  ?lru-a  ?lru-b)
        (excluded-lru  ?pair  ?lru-c)
        (or  (predicted-tacan  status  ?config  ?lru-c  ?measurement  not-avail)
             (tacan-lock  ?lru-c  ?measurement  off))
        (tacan-relative-difference  ?pair  ?measurement  bias  over)
        =>
        (assert  (predicted-tacan  dilemma  ?config  ?measurement  on)))

;-----------------------------------------------------------------------------

(defrule  tacan-predict-no-dilemma

;;      IF
;;              No rule has yet predicted that RM will declare a dilemma
;;                      in a proposed configuration
;;      THEN
;;              Predict that RM will not declare a dilemma in the proposed
;;                      configuration
;;      END

        (declare  (salience -1))
        (sub-phase  tacan  deselect)
        (predicted-tacan  status  ?config  ?  ?measurement  ?)
        (not  (predicted-tacan  dilemma  ?config  ?measurement  ?))
        =>
        (assert  (predicted-tacan  dilemma  ?config  ?measurement  off)))
```

```
;---------------------------------------------------------------------------

(defrule  tacan-predict-error-1-level

;;       IF
;;               The data good flag is on for a measurement in a proposed
;;                       configuration  AND
;;               One LRU is available and locked  AND
;;               The other two LRUs are either unavailable or unlocked
;;       THEN
;;               Predict the selected measurement bias and noise is the
;;                       same as that of the available LRU
;;       END

        (sub-phase  tacan  deselect)
        (predicted-tacan  data-good  ?config  ?measurement  on)
        (predicted-tacan  status  ?config  ?lru-a  ?measurement  avail)
        (tacan-lock  ?lru-a  ?measurement  on)
        (lrus-in-pair  ?  ?lru-a  ?lru-b)
        (lrus-in-pair  ?  ?lru-a  ?lru-c&~?lru-b)
        (or  (predicted-tacan  status  ?config  ?lru-b  ?measurement  ~avail)
             (tacan-lock  ?lru-b  ?measurement  off))
        (or  (predicted-tacan  status  ?config  ?lru-c  ?measurement  ~avail)
             (tacan-lock  ?lru-c  ?measurement  off))
        =>
        (bind  ?bias  (tacan-error  ?lru-a  ?measurement  bias))
        (assert  (predicted-tacan  bias  ?config  ?measurement  ?bias))
        (bind  ?noise  (tacan-error  ?lru-a  ?measurement  noise))
        (assert  (predicted-tacan  noise  ?config  ?measurement  ?noise)))

;---------------------------------------------------------------------------

(defrule  tacan-predict-error-2-level

;;       IF
;;               The data good flag is on for a measurement in a proposed
;;                       configuration  AND
;;               Two LRU's are available and locked  AND
;;               The other LRU is either unavailable or unlocked
;;       THEN
;;               Predict the selected measurement bias and noise is the
;;                       average of the available LRUs
;;       END

        (sub-phase  tacan  deselect)
        (predicted-tacan  data-good  ?config  ?measurement  on)
        (predicted-tacan  status  ?config  ?lru-a  ?measurement  avail)
        (tacan-lock  ?lru-a  ?measurement  on)
        (predicted-tacan  status  ?config  ?lru-b&~?lru-a  ?measurement  avail)
        (tacan-lock  ?lru-b  ?measurement  on)
        (lrus-in-pair  ?pair  ?lru-a  ?lru-b)
        (excluded-lru  ?pair  ?lru-c)
        (or  (predicted-tacan  status  ?config  ?lru-c  ?measurement  ~avail)
             (tacan-lock  ?lru-c  ?measurement  off))
        =>
        (bind  ?bias-a  (tacan-error  ?lru-a  ?measurement  bias))
        (bind  ?bias-b  (tacan-error  ?lru-b  ?measurement  bias))
        (bind  ?bias  (/  (+  ?bias-a  ?bias-b)  2.0))
        (assert  (predicted-tacan  bias  ?config  ?measurement  ?bias))
        (bind  ?noise-a  (tacan-error  ?lru-a  ?measurement  noise))
        (bind  ?noise-b  (tacan-error  ?lru-b  ?measurement  noise))
        (bind  ?noise  (/  (sqrt  (+  (* ?noise-a ?noise-a)
                                      (* ?noise-b ?noise-b)))  2.0))
        (assert  (predicted-tacan  noise  ?config  ?measurement  ?noise)))

;-----------------------------------------------------------------------------

(defrule  tacan-predict-error-3-level

;;      IF
;;              The data good flag is on for a measurement in a proposed
;;                      configuration  AND
;;              All LRUs are available and locked for that measurement
;;      THEN
;;              Predict the selected measurement bias and noise is the
;;                      same as what is currently being selected by RM.
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  data-good  ?config  ?measurement  on)
        (predicted-tacan  status  ?config  1  ?measurement  avail)
        (tacan-lock  1  ?measurement  on)
        (predicted-tacan  status  ?config  2  ?measurement  avail)
        (tacan-lock  2  ?measurement  on)
        (predicted-tacan  status  ?config  3  ?measurement  avail)
        (tacan-lock  3  ?measurement  on)
        =>
        (bind  ?bias  (tacan-error  0  ?measurement  bias))
        (assert  (predicted-tacan  bias  ?config  ?measurement  ?bias))
        (bind  ?noise  (tacan-error  0  ?measurement  noise))
        (assert  (predicted-tacan  noise  ?config  ?measurement  ?noise)))


;;*****************************************************************************
;;
;;;  GROUP
;;      Choose Best Configuration (3.8.6.5)
;;
;;      This group of rules compares proposed configurations and chooses
;;      the one that should give the best performance
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  deselect)
;;
;;;  CONTAINING GROUP
;;      Deselect TACAN LRU
;;
;;*****************************************************************************

(defrule  tacan-dont-want-dilemma

;;      IF
;;              A proposed configuration will result in a dilemma in either
;;                      measurement
;;      THEN
;;              Veto that configuration
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  dilemma  ?config  ?measurement  on)
        =>
        (assert  (vetoed  ?config)))

;----------------------------------------------------------------------------

(defrule  tacan-need-range-data

;;      IF
;;              A proposed configuration does not have range data
;;      THEN
;;              Veto that configuration
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  data-good  ?config  range  off)
        =>
        (assert  (vetoed  ?config)))

;----------------------------------------------------------------------------

(defrule  tacan-dont-have-bearing

;;      IF
;;              A proposed configuration does not have bearing data
;;      THEN
;;              Assume the crosstrack state error under the proposed
;;                      configuration will be the same as the current
;;                      crosstrack state error
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  data-good  ?config  bearing  off)
        =>
        (bind  ?bearing-bias  (/  (state-error  pass  w)  200.0))
        (assert  (predicted-tacan  bias  ?config  bearing  ?bearing-bias))
        (assert  (predicted-tacan  noise  ?config  bearing  0.0)))

;----------------------------------------------------------------------------

(defrule  tacan-predict-state-effect

;;      IF
;;              A configuration has not been vetoed
;;      THEN
;;              Predict the effect of the proposed configuration on the
;;                      state error
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  bias  ?config  range  ?range-bias)
        (predicted-tacan  noise  ?config  range  ?range-noise)
        (predicted-tacan  bias  ?config  bearing  ?bearing-bias)
        (predicted-tacan  noise  ?config  bearing  ?bearing-noise)
        (number-deselected  ?config  ?n-desel)
        (not  (vetoed  ?config))
        =>
        (bind  ?e1  ?range-bias)
        (bind  ?e2  ?range-noise)
        (bind  ?e3  (*  200.0  ?bearing-bias))
        (bind  ?e4  (*  200.0  ?bearing-noise))
        (bind  ?e5  (*  5000.0  ?n-desel))
        (bind  ?effect  (sqrt  (+  (*  ?e1  ?e1)
                                   (*  ?e2  ?e2)
                                   (*  ?e3  ?e3)
                                   (*  ?e4  ?e4))))
        (bind  ?effect  (+  ?effect  ?e5))
        (assert  (predicted-tacan  state-effect  ?config  ?effect)))

;------------------------------------------------------------------------------

(defrule  tacan-pick-smallest-state-effect

;;      IF
;;              One configuration has a smaller predicted state error
;;                      than another
;;      THEN
;;              Veto the configuration with the larger state error
;;      END

        (sub-phase  tacan  deselect)
        (predicted-tacan  state-effect  ?config-a  ?effect-a)
        (predicted-tacan  state-effect  ?config-b  ?effect-b)
        (test  (<  ?effect-a  ?effect-b))
        =>
        (assert  (vetoed  ?config-b)))

;------------------------------------------------------------------------------

(defrule  tacan-select-a-configuration

;;      IF
;;              All configurations that are going to be vetoed have been
;;                      vetoed
;;      THEN
;;              Select the only one left as the chosen configuration

        (declare  (salience  -2))
        (sub-phase  tacan  deselect)
        (predicted-tacan  state-effect  ?config  $?)
        (not  (vetoed  ?config))
        =>
        (assert  (chosen-configuration  ?config)))

;------------------------------------------------------------------------------

(defrule  tacan-confirm-a-deselect

;;      IF
;;              An LRU is deselected in the chosen configuration
;;      THEN
;;              Confirm the deselect suggestion
;;      END

        (sub-phase  tacan  deselect)
        (chosen-configuration  ?config)
        (possible-tacan-configuration  ?config  ?lru  on)
        =>
        (assert  (suggested-deselect  ?lru  confirmed)))
```

```
;---------------------------------------------------------------------------

(defrule  tacan-deny-a-deselect

;;       IF
;;               The initial deselect determination suggested deselecting
;;                       an LRU  AND
;;               That LRU is not deselected in the chosen configuration
;;       THEN
;;               Deny the deselect suggestion
;;       END

         (sub-phase  tacan  deselect)
         (chosen-configuration  ?config)
         (possible-tacan-configuration  ?config  ?lru  off)
         (need-to-deselect  ?lru)
         =>
         (assert  (suggested-deselect  ?lru  denied)))

;---------------------------------------------------------------------------

(defrule  tacan-deselect-confirmed

;;       IF
;;               A deselect suggestion has been confirmed
;;       THEN
;;               Send the recommendation to the operator
;;       END

         (declare  (salience  1))
         (sub-phase  tacan  deselect)
         (suggested-deselect  ?lru  confirmed)
         =>
         (assert  (recommend  tacan  deselect-tacan  off-nominal  alt
                 "Need to deselect TACAN LRU " ?lru)))

;---------------------------------------------------------------------------

(defrule  tacan-deselect-shortcut

;;       IF
;;               An LRU has been suggested for deselection  AND
;;               That suggestion has already been confirmed or denied
;;       THEN
;;               Withdraw the suggestion
;;       END

         (sub-phase  tacan  clean-up)
         ?x <- (need-to-deselect  ?lru)
         (suggested-deselect  ?lru  $?)
         =>
         (retract  ?x))

;---------------------------------------------------------------------------

(defrule  tacan-deselect-cleanup

;;       IF
;;               All work on all deselects has been completed  AND
;;               A temporary fact generated during the deselect determination
;;                      still exists
;;       THEN
;;               Remove the fact
;;       END

         (sub-phase  tacan  clean-up)
         ?x <- (possible-tacan-configuration|
                 number-deselected|
                 predicted-tacan|
                 vetoed|
                 chosen-configuration  $?)
         =>
         (retract  ?x))

;;****************************************************************************
;;
;;;  GROUP
;;      Reselect TACAN LRU (3.8.7)
;;
;;      This group determines when to recommend reselected a TACAN LRU
;;
;;;  CONTROL FACTS
;       (sub-phase  tacan  reselect)
;;
;;;  CONTAINING GROUP
;;      TACAN
;;
;;****************************************************************************

(defrule  tacan-reselect-a-tacan

;;       IF
;;               For the engaged system
;;               A TACAN LRU is unavailable in a measurement due to
;;                       RM-declared failure or deselect  AND
;;               The LRU is locked and good in range  AND
;;               The LRU is locked and good in bearing
;;       THEN
;;               Recommend reselecting the LRU
;;       END

         (sub-phase  tacan  reselect)
         (engaged-system ?sys)
         (tacan-status  ?sys  ?lru  ?measurement  fail|deselect)
         (tacan-lock  ?lru  range  on)
         (tacan-lru-quality  ?lru  range  good)
         (tacan-lock  ?lru  bearing  on)
         (tacan-lru-quality  ?lru  bearing  good)
         =>
         (assert  (recommend  tacan  reselect-tacan  off-nominal  alt
                 "Need to reselect TACAN LRU " ?lru " in the " ?sys)))


;;****************************************************************************
;;
;;;  GROUP
;;      TACAN AIF Determination (3.8.8)
;;
```

```
;;       This group of rules determines when the TACAN AIF switch should be
;;       changed, and what the new value should be.
;;
;;;   CONTROL FACTS
;        (sub-phase  tacan  aif-change)
;;
;;;   CONTAINING GROUP
;;       TACAN
;;
;;*************************************************************************

(defrule tacan-selected-tacan-is-acceptable

;;       IF
;;               For the engaged system
;;               The selected measurement was previously no-go
;;               The measurement error from every available and locked LRU
;;                       is less than the corresponding state error  AND
;;       THEN
;;               Change the selected measurement to "go"
;;       END

         (sub-phase tacan aif-change)
         (engaged-system ?sys)
         ?x <- (selected-tacan  ?measurement  no-go)
         (or  (and  (tacan-error  1  ?measurement  raw  under)
                    (tacan-lock  1  ?measurement on)
                    (tacan-status  ?sys  1  ?measurement  avail))
                    (tacan-lock  1  ?measurement off)
                    (tacan-status  ?sys  1  ?measurement  ~avail))
         (or  (and  (tacan-error  2  ?measurement  raw  under)
                    (tacan-lock  2  ?measurement on)
                    (tacan-status  ?sys  2  ?measurement  avail))
                    (tacan-lock  2  ?measurement off)
                    (tacan-status  ?sys  2  ?measurement  ~avail))
         (or  (and  (tacan-error  3  ?measurement  raw  under)
                    (tacan-lock  3  ?measurement on)
                    (tacan-status  ?sys  3  ?measurement  avail))
                    (tacan-lock  3  ?measurement off)
                    (tacan-status  ?sys  3  ?measurement  ~avail))
         =>
         (retract  ?x)
         (assert  (selected-tacan  ?measurement  go)))

;------------------------------------------------------------------------

(defrule  tacan-selected-tacan-is-unacceptable

;;       IF
;;               For the engaged system
;;               The selected TACAN measurement was previously "go"  AND
;;               The error from any available and locked LRU is unacceptable
;;       THEN
;;               Change the selected measurement to "no-go"
;;       END

         (sub-phase tacan aif-change)
         (engaged-system ?sys)
         ?x <- (selected-tacan  ?measurement  go)
         (tacan-error  ?lru  ?measurement  raw  over)
```

```
        (tacan-lock  ?lru  ?measurement  on)
        (tacan-status  ?sys  ?lru  ?measurement  avail)
        =>
        (retract  ?x)
        (assert  (selected-tacan  ?measurement  no-go)))

;----------------------------------------------------------------------------

(defrule  tacan-to-auto

;;      IF
;;              The pass is engaged
;;              Range and bearing data good flags are on  AND
;;              No toggle has been requested  AND
;;              No TACAN deselects have been recommended  AND
;;              No delta-state is in work  AND
;;              Selected range and bearing errors are acceptable  AND
;;              Range and bearing edit ratios are less than 1  AND
;;              TACAN is currently inhibited
;;      THEN
;;              Recommend that TACAN go to AUTO mode
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system pass)
        (data-good  pass  tacr  on)
        (data-good  pass  tacb  on)
        (not  (need-a-toggle))
        (not  (suggested-deselect  ?  confirmed))
        (not  (need-delta-state  $?))
        (selected-tacan  range  go)
        (selected-tacan  bearing  go)
        (edit-ratio  pass  tacr  under)
        (edit-ratio  pass  tacb  under)
        (aif  pass  tacan  inhibit)
        =>
        (assert  (recommend  tacan  tacan-to-auto  nominal  alt
                "TACAN" " is good and can be placed in AUTO")))

;----------------------------------------------------------------------------

(defrule  tacan-to-auto-no-bearing

;;      IF
;;              The pass is engaged
;;              Range data-good is on  AND
;;              Bearing data-good is off  AND
;;              No toggle has been requested  AND
;;              No TACAN deselects have been requested  AND
;;              No delta state is in work  AND
;;              Selected range error is acceptable  AND
;;              Range edit ratio is less than 1  AND
;;              TACAN is currently inhibited
;;      THEN
;;              Recommend TACAN be put in AUTO
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system pass)
        (data-good  pass  tacr  on)
        (data-good  pass  tacb  off)
        (not  (need-a-toggle))
        (not  (suggested-deselect  ?  confirmed))
        (not  (need-delta-state  $?))
        (selected-tacan  range  go)
        (edit-ratio  pass  tacr  under)
        (aif  pass  tacan  inhibit)
        =>
        (assert  (recommend  tacan  tacan-to-auto  nominal  alt
                "TACAN" " is good and can be placed in AUTO")))

;----------------------------------------------------------------------------

(defrule  tacan-to-auto-end-force

;;      IF
;;              The pass is engaged
;;              TACAN is being forced  AND
;;              Range and bearing edit ratios are less than 1
;;      THEN
;;              Recommend TACAN be put in auto
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system pass)
        (aif  pass  tacan  force)
        (edit-ratio  pass  tacr  under)
        (edit-ratio  pass  tacb  under)
        =>
        (assert  (recommend  tacan  end-force  nominal  alt
                "TACAN" " should be returned to AUTO")))

;----------------------------------------------------------------------------

(defrule  tacan-auto-after-update

;;      IF
;;              For the engaged system
;;              Range and bearing data-good flags are on  AND
;;              No toggle has been requested  AND
;;              No TACAN deselects have been requested  AND
;;              A delta-state is in work  AND
;;              Selected range and bearing errors are acceptable  AND
;;              TACAN is currently inhibited
;;      THEN
;;              Recommend TACAN be put in AUTO after the delta-state
;;                      is complete
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system ?sys)
        (data-good  ?sys  tacr  on)
        (data-good  ?sys  tacb  on)
        (not  (need-a-toggle))
        (not  (suggested-deselect  ?  confirmed))
        (need-delta-state  $?)
        (selected-tacan  range  go)
        (selected-tacan  bearing  go)
        (aif  ?sys  tacan  inhibit)
        =>
        (assert  (recommend  tacan  auto-after-update  nominal  alt
                "TACAN is good and can be put in AUTO after the "
                "delta-state is complete")))

;------------------------------------------------------------------------------

(defrule  tacan-inhibit-bad-tacan

;;      IF
;;              The pass is engaged
;;              No delta-state is in work  AND
;;              State error is good or suspect AND
;;              TACAN is not inhibited  AND
;;                  Range edit ratio is greater than 1
;;              OR
;;                  (   Bearing edit ratio is greater than 1
;;                      while vehicle is not in the cone of confusion
;;                  )
;;      THEN
;;              Recommend TACAN be inhibited
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system pass)
        (not  (need-delta-state  $?))
        (gnd-state  pass  worst-axis  ~over)
        (aif  pass  tacan  ~inhibit)
        (or  (edit-ratio  pass  tacr  over)
             (and  (edit-ratio  pass  tacb  over)
                   (cone  off)))
        =>
        (assert  (recommend  tacan  inhibit-bad-tacan  off-nominal  alt
                "TACAN" " should be inhibited")))

;------------------------------------------------------------------------------

(defrule  tacan-error-before-tacan

;;      IF
;;              For the engaged system
;;              At least one LRU is locked in range  AND
;;              Neither range nor bearing is being processed  AND
;;              The status of the state error is different from
;;                      what it was on the previous cycle
;;      THEN
;;              Note the current status of the state error
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system ?sys)
        (prev-tacan-lock  range  on)
        (filter-flag  ?sys  tacr  ~process)
        (filter-flag  ?sys  tacb  ~process)
        (gnd-state  ?sys  worst-axis  ?status)
        ?x <- (error-before-tacan  ~?status)
        =>
        (retract  ?x)
        (assert  (error-before-tacan  ?status)))

;------------------------------------------------------------------------------
```

```
(defrule  tacan-error-after-tacan

;;      IF
;;              For the engaged system
;;              TACAN is being processed  AND
;;              The state error is worse now than before TACAN was processed
;;      THEN
;;              Recommend TACAN be inhibited
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system ?sys)
        (error-before-tacan  ?before)
        (filter-flag  ?sys  tacr|tacb  process)
        (gnd-state  ?sys  worst-axis  ?after&~?before)
        (max-miscompare  ?before  ?after  ?after)
        =>
        (assert  (recommend  tacan  inhibit-bad-tacan  off-nominal  alt
                "TACAN made the state error worse.  It needs to be "
                "inhibited")))

;-----------------------------------------------------------------------------

(defrule  tacan-to-force

;;      IF
;;              The pass is engaged
;;              Range and bearing data-good flags are on  AND
;;              No toggle has been requested  AND
;;              No TACAN deselects have been requested  AND
;;              No delta-state is in work  AND
;;              Selected range and bearing errors are acceptable  AND
;;              Either range or bearing edit ratio is greater than 1  AND
;;              TACAN is not being forced
;;      THEN
;;              Recommend forcing TACAN
;;      END

        (sub-phase  tacan  aif-change)
        (engaged-system  pass)
        (data-good  pass  tacr  on)
        (data-good  pass  tacb  on)
        (gnd-state  pass  worst-axis  over)
        (not  (need-a-toggle))
        (not  (suggested-deselect  ?  confirmed))
        (not  (need-delta-state  $?))
        (selected-tacan  range  go)
        (selected-tacan  bearing  go)
        (edit-ratio  pass  tacr|tacb  over)
        (aif  pass  tacan  ~force)
        =>
        (assert  (recommend  tacan  force-tacan  off-nominal  alt
                "TACAN" " is good and should be forced")))

;-----------------------------------------------------------------------------
```

3.9 Baro Altitude

```
;;*****************************************************************************
;;
;;;  GROUP
;;      Baro Altitude (3.9)
;;
;;      This group checks baro altitude, and recommends (output)
;;      a setting for the baro AIF switch.
;;
;;;  CONTROL FACTS
;       (sub-phase baro ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;*****************************************************************************


;;;  FACTS

(deffacts  monitoring-baro-phases        ; These facts define the sequence of
                                         ; sub-phases within the monitoring
                                         ; phase of baro
        (first-sub-phase  baro  monitoring  quality)
                                         ; First sub-phase is quality checks
        (next-sub-phase  baro  quality  flag-status)
                                         ; Then comes flag status
)

(deffacts  analysis-baro-phases          ; These facts define the sequence of
                                         ; sub-phases within the analysis
                                         ; phase of baro
        (first-sub-phase  baro  analysis  recommendation)
                                         ; The only sub-phase is recommendation
)


(deffacts  initial-baro-facts            ; These facts represent assumptions
                                         ; about baro before any data is received
        (baro-status  unknown)           ; The quality of baro measurements is
                                         ; unknown
        (prev-filter-flag  pass  baro  off)
                                         ; Baro is not being processed in the
                                         ; PASS
        (prev-filter-flag  bfs  baro  off)
                                         ; Baro is not being processed in the BFS
)


;;*****************************************************************************
;;
;;;  GROUP
;;      Baro Measurement Quality (3.9.1)
;;
;;      This group of rules determines whether or not baro alitude measurements
;;      are good.  If they are bad, the rules attempt to determine the reason.
;;
;;;  CONTROL FACTS
;       (sub-phase  baro  quality)
;;
;;;  CONTAINING GROUP
```

```
;;      Baro Altitude
;;
;;**************************************************************************

(defrule baro-ok-to-perform-baro-checks
;;      IF
;;              Mach is greater than 5 OR
;;              in mach jump region
;;      THEN
;;              Do not perform any baro checking
;;      END

        ?x <-(sub-phase baro quality)
             (or (mach-jump on)
                 (mach-number ?n&:(> ?n 5.0)))
         =>
              (retract ?x))

;--------------------------------------------------------------------------

(defrule  baro-is-good-bfs

;;      IF
;;              For the bfs system
;;              The HSTD is good
;;              Baro was previously not known to be good
;;              |delta-sel| <= |delta-z| + 500
;;      THEN
;;              Baro is good
;;      END

        (sub-phase  baro  quality)
        (hstd  good)
        ?x <- (baro-status  ~good)
        (baro-gnh ?delta-sel)
        (engaged-system bfs)
        (test (<= (abs ?delta-sel)
                  (+ (abs (state-error bfs u)) 500.0)))
        =>
        (assert  (status-light  baro  0  good))
        (retract  ?x)
        (assert  (baro-status  good)))

;--------------------------------------------------------------------------

(defrule  baro-is-bad-bfs

;;      IF
;;              For the bfs system
;;              The HSTD is good
;;              Baro was previously good or unknown
;;              |delta-sel| > |delta-z| + 500
;;      THEN
;;              Baro is bad
;;      END

        (sub-phase  baro  quality)
        (hstd  good)
        ?x <- (baro-status  ?prev-status&good|unknown)
        (baro-gnh ?delta-sel)
        (engaged-system bfs)
        (test (> (abs ?delta-sel)
                 (+ (abs (state-error bfs u)) 500.0)))
        =>
        (assert  (status-light  baro  0  bad))
        (if  (eq ?prev-status good)
            then
                (assert  (event baro off-nominal mach "Air" " data is bad")))
        (retract  ?x)
        (assert  (baro-status  bad)))

;-----------------------------------------------------------------------------

(defrule  baro-is-good-pass

;;      IF
;;              For the pass system
;;              The HSTD is good
;;              Baro was previously not known to be good
;;              |delta-sel| <= |delta-z| + 500
;;      THEN
;;              Baro is good
;;      END

        (sub-phase  baro  quality)
        (hstd  good)
        ?x <- (baro-status  ~good)
        (delta-z ?delta-z)
        (baro-gnh ?delta-sel)
        (engaged-system pass)
        (test (<= (abs ?delta-sel)
                  (+ (abs ?delta-z) 500.0)))
        =>
        (assert  (status-light  baro  0  good))
        (retract  ?x)
        (assert  (baro-status  good)))

;-----------------------------------------------------------------------------

(defrule  baro-is-bad-pass

;;      IF
;;              For the pass system
;;              The HSTD is good
;;              Baro was previously good or unknown
;;              |delta-sel| > |delta-z| + 500
;;      THEN
;;              Baro is bad
;;      END

        (sub-phase  baro  quality)
        (hstd  good)
        ?x <- (baro-status  ?prev-status&good|unknown)
        (delta-z ?delta-z)
        (baro-gnh ?delta-sel)
        (engaged-system pass)
        (test (> (abs ?delta-sel)
                 (+ (abs ?delta-z) 500.0)))
        =>
        (assert  (status-light  baro  0  bad))
```

```
         (if  (eq ?prev-status good)
              then
                   (assert  (event baro off-nominal mach "Air" " data is bad")))
         (retract  ?x)
         (assert  (baro-status  bad)))

;-----------------------------------------------------------------------------

(defrule  baro-roll-reversal

;;       IF
;;                 Baro is bad
;;                 The vehicle is executing a roll-reversal
;;       THEN
;;                 Baro is bad because of roll-reversal
;;       END

         (sub-phase  baro  quality)
         ?x <- (baro-status  bad)
         (roll-rate  high)
         =>
         (assert  (status-light  baro  0  roll))
         (assert  (event baro off-nominal mach
                          "Air" " data is bad due to roll reversal"))
         (retract  ?x)
         (assert  (baro-status  roll-reversal)))

;-----------------------------------------------------------------------------

(defrule baro-crew-call

;;       IF
;;                 HSTD is not good
;;       THEN
;;                 ADTA is crew call
;;       END

         (sub-phase baro quality)
         (hstd ~good)
         (not  (ADTA crew-call))
         =>
         (assert (ADTA crew-call))
         (assert (status-light baro 0 crew))
         (assert (event baro off-nominal mach
                         "Air" " data is crew call")))

;-----------------------------------------------------------------------------

(defrule baro-not-crew-call

;;       IF
;;                 HSTD is good
;;       THEN
;;                 ADTA is not crew call
;;       END

         (sub-phase baro quality)
         (hstd good)
         ?x <- (ADTA crew-call)
         =>
```

```
        (retract ?x)
        (assert (status-light baro 0 blank))
        (assert (event baro off-nominal mach
                       "Air" " data is not crew call")))

;;*****************************************************************************
;;
;;;  GROUP
;;      Baro Flag Status (3.9.2)
;;
;;      This group watches for changes in the baro altitude filter flag.  It
;;      also watches to see if the change is caused by entering or leaving
;;      the Mach jump region.
;;
;;;  CONTROL FACTS
;       (sub-phase  baro  flag-status)
;;
;;;  CONTAINING GROUP
;;      Baro Altitude
;;
;;*****************************************************************************
;;
;;  BARO FLAG STATUS
;;

(defrule  baro-enter-mach-jump

;;      IF
;;              The vehicle was not previously in the mach jump region
;;              The vehicle is now in the mach jump region
;;      THEN
;;              Notify the operator that the mach jump region has been entered
;;      END

        (sub-phase  baro  flag-status)
        (not  (in-mach-jump))
        (mach-jump  on)
        =>
        (assert  (in-mach-jump))
        (assert  (event baro nominal mach "Entering" " Mach jump region")))

;-----------------------------------------------------------------------------

(defrule  baro-leave-mach-jump

;;      IF
;;              The vehicle was previously in the mach jump region
;;              The vehicle is now out of the mach jump region
;;      THEN
;;              Notify the operator that the mach jump region has been exited
;;      END

        (sub-phase  baro  flag-status)
        ?x <- (in-mach-jump)
        (mach-jump  off)
        =>
        (retract  ?x)
        (assert  (event baro nominal mach "Leaving" " Mach jump region")))

;-----------------------------------------------------------------------------
```

```
(defrule  baro-filter-flag-changed

;;       IF
;;               For the engaged system
;;               The current value of the baro filter flag
;;                     is different from its previous value
;;       THEN
;;               Conclude that the value has changed
;;               Notify the operator of the new value
;;       END

         (sub-phase  baro  flag-status)
         (engaged-system ?sys)
         (filter-flag  ?sys  baro  ?flag)
         ?x <- (prev-filter-flag  ?sys  baro  ~?flag)
         =>
         (retract  ?x)
         (assert  (prev-filter-flag  ?sys  baro  ?flag))
         (assert  (event baro nominal mach "air data status is "
                   ?flag)))

;;**************************************************************************
;;
;;;  GROUP
;;      Baro Recommendations - Ground Available (3.9.3)
;;
;;      This group recommends a setting for the AIF switch when the ground
;;      state is available.
;;
;;;  CONTROL FACTS
;       (sub-phase  baro  recommendation)
;;
;;;  CONTAINING GROUP
;;      Baro Altitude
;;
;;**************************************************************************

(defrule  baro-to-auto

;;       IF
;;               For the pass system
;;               Baro is good
;;               The baro edit ratio is less than 1
;;               Baro is inhibited
;;       THEN
;;               Baro is go for nav
;;       END

         (sub-phase  baro  recommendation)
         (baro-status  good)
         (engaged-system pass)
         (edit-ratio  pass  baro  under)
         (aif  pass  baro  inhibit)
         =>
         (assert  (recommend  baro  baro-to-auto  nominal  mach
                  "Air" " data is go for nav")))

;-------------------------------------------------------------------------
```

```
(defrule  baro-to-force

;;       IF
;;               For the pass system
;;               Baro is good
;;               The baro edit ratio is greater than 1
;;               Baro is not being forced
;;       THEN
;;               Recommend forcing baro
;;       END

        (sub-phase  baro  recommendation)
        (baro-status  good)
        (engaged-system pass)
        (edit-ratio  pass  baro  over)
        (aif  pass  baro  ~force)
        =>
        (assert  (recommend  baro  baro-to-force  off-nominal  mach
                "Need" " to force air data to nav")))

;------------------------------------------------------------------------------

(defrule  baro-end-force

;;       IF
;;               For the pass system
;;               Baro is good
;;               The baro edit ratio is less than 1
;;               Baro is being forced
;;       THEN
;;               Recommend returning baro to auto
;;       END

        (sub-phase  baro  recommendation)
        (baro-status  good)
        (engaged-system pass)
        (edit-ratio  pass  baro  under)
        (aif  pass  baro  force)
        =>
        (assert  (recommend  baro  end-baro-force  off-nominal  mach
                "Need" " to return air data to auto for nav")))

;---------------------------------------------------- ------------------------

(defrule  baro-to-inhibit

;;       IF
;;               For engaged system
;;               Baro is bad
;;               The vehicle is not in the Mach jump region
;;               Baro is not inhibited
;;       THEN
;;               Recommend that baro be inhibited
;;       END

        (sub-phase  baro  recommendation)
        (baro-status  ~good&~unknown)
        (mach-jump  off)
        (engaged-system ?sys)
        (aif  ?sys  baro  ~inhibit)
        =>
        (assert  (recommend  baro  inhibit-baro  off-nominal  mach
                 "Air"  " data is no-go and should be inhibited")))

;-----------------------------------------------------------------
```

3.10 Microwave Scan Beam Landing System

```
;;*****************************************************************************
;;
;;;  GROUP
;;      MSBLS (3.10)
;;
;;      This group monitors MSBLS data, recommends (output) which
;;      of the three LRUs should be used, and whether MSBLS
;;      should be used or not.
;;
;;;  CONTROL FACTS
;       (sub-phase  msbls  ?)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;*****************************************************************************


;;;  FACTS

(deffacts monitoring-msbls-phases               ; Defines the sequence of
                                                ; sub-phases in the monitoring
                                                ; phase of the MSLBS section.
        (first-sub-phase  msbls  monitoring  availability)
                                                ; First sub-phase is a check
                                                ; for LRU availability
        (next-sub-phase  msbls  availability  lockon)
                                                ; Then comes a check for lock
        (next-sub-phase  msbls  lockon  quality)
                                                ; Then comes LRU quality check
        (next-sub-phase  msbls  quality  watch-flags)
                                                ; Last is a flag-status check
)

(deffacts  analysis-msbls-phases                ; These facts define the
                                                ; sequence of sub-phases in the
                                                ; analysis phase of MSBLS
        (first-sub-phase  msbls  analysis  recommendation)
                                                ; First is recommendations
                                                ; based on LRU quality
        (next-sub-phase  msbls  recommendation  watch-state)
                                                ; Last is recommendation based
                                                ; on effects on state error
)


(deffacts  initial-msbls-facts          ; These facts represent assumptions
                                        ; about MSBLS before any data is
                                        ; received
        (msbls-status  1  avail)                ; LRU 1 is available
        (msbls-status  2  avail)                ; LRU 2 is available
        (msbls-status  3  avail)                ; LRU 3 is available
        (msbls-num-avail  3)                    ; All 3 LRUs are available
        (msbls-num-locked  range  0)            ; No LRUs are locked in range
        (msbls-num-locked  azimuth  0)          ; No LRUs are locked in azimuth
        (msbls-num-locked  elevation  0)        ; No LRUs locked in elevation
        (last-msbls-report  1  range  bias  unknown)
                                                ; Do not know if LRU 1 range
                                                ; has a bias
        (last-msbls-report  1  range  noise  unknown)
```

```
                                        ; Do not know if LRU 1 range
                                        ; has a noise
(last-msbls-report  1  azimuth  bias  unknown)
                                        ; Do not know if LRU 1 azimuth
                                        ; has a bias
(last-msbls-report  1  azimuth  noise  unknown)
                                        ; Do not know if LRU 1 azimuth
                                        ; has a noise
(last-msbls-report  1  elevation  bias  unknown)
                                        ; Do not know if LRU 1 elevation
                                        ; has a bias
(last-msbls-report  1  elevation  noise  unknown)
                                        ; Do not know if LRU 1 elevation
                                        ; has a noise
(last-msbls-report  2  range  bias  unknown)
                                        ; Do not know if LRU 2 range
                                        ; has a bias
(last-msbls-report  2  range  noise  unknown)
                                        ; Do not know if LRU 2 range
                                        ; has a noise
(last-msbls-report  2  azimuth  bias  unknown)
                                        ; Do not know if LRU 2 azimuth
                                        ; has a bias
(last-msbls-report  2  azimuth  noise  unknown)
                                        ; Do not know if LRU 2 azimuth
                                        ; has a noise
(last-msbls-report  2  elevation  bias  unknown)
                                        ; Do not know if LRU 2 elevation
                                        ; has a bias
(last-msbls-report  2  elevation  noise  unknown)
                                        ; Do not know if LRU 2 elevation
                                        ; has a noise
(last-msbls-report  3  range  bias  unknown)
                                        ; Do not know if LRU 3 range
                                        ; has a bias
(last-msbls-report  3  range  noise  unknown)
                                        ; Do not know if LRU 3 range
                                        ; has a noise
(last-msbls-report  3  azimuth  bias  unknown)
                                        ; Do not know if LRU 3 azimuth
                                        ; has a bias
(last-msbls-report  3  azimuth  noise  unknown)
                                        ; Do not know if LRU 3 azimuth
                                        ; has a noise
(last-msbls-report  3  elevation  bias  unknown)
                                        ; Do not know if LRU 3 elevation
                                        ; has a bias
(last-msbls-report  3  elevation  noise  unknown)
                                        ; Do not know if LRU 3 elevation
                                        ; has a noise
(msbls-lru-quality  1  range  none)     ; no rating on LRU 1 range
(msbls-lru-quality  1  azimuth  none)   ; no rating on LRU 1 azimuth
(msbls-lru-quality  1  elevation  none) ; no rating on LRU 1 elevation
(msbls-lru-quality  2  range  none)     ; no rating on LRU 2 range
(msbls-lru-quality  2  azimuth  none)   ; no rating on LRU 2 azimuth
(msbls-lru-quality  2  elevation  none) ; no rating on LRU 2 elevation
(msbls-lru-quality  3  range  none)     ; no rating on LRU 3 range
(msbls-lru-quality  3  azimuth  none)   ; no rating on LRU 3 azimuth
(msbls-lru-quality  3  elevation  none) ; no rating on LRU 3 elevation
(error-before-msbls  under)             ; state error before MSBLS
```

```
                                                  ; is within limits
        (prev-filter-flag  pass  mlsr  off)       ; not processing range
        (prev-filter-flag  pass  mlsa  off)       ; not processing azimuth
        (prev-filter-flag  pass  mlse  off)       ; not processing elevation
        (prev-data-good  pass  mlsr  off)         ; range data-good is off
        (prev-data-good  pass  mlsa  off)         ; azimuth data-good is off
        (prev-data-good  pass  mlse  off)         ; elevation data-good is off
)


;;**************************************************************************
;;
;;;  GROUP (3.10.1)
;;      MSBLS Availability
;;
;;      This group determines which LRUs are available.  It also determines
;;      why the unavailable LRUs are unavailable.
;;
;;;  CONTROL FACTS
;       (sub-phase  msbls  availability)
;;
;;;  CONTAINING GROUP
;;      MSBLS
;;
;;**************************************************************************

(defrule  msbls-commfault

;;      IF
;;              A MSBLS LRU was not previously commfaulted  AND
;;              The LRU is powered on AND
;;              The commfault flag for that LRU is now on
;;      THEN
;;              Notify the operator that the LRU is commfaulted (unless
;;                    the whole string is down)
;;              Conclude the LRU is no longer available due to commfault
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-status  ?lru  avail|fail)
        (msbls-flag  commfault  ?lru  on)
        (string-commfault  pass  ?lru  ?string-flag)
        =>
        (if  (eq  ?string-flag  off)
            then
                (assert  (event msbls off-nominal alt "Commfault MSBLS " ?lru)))
        (assert  (status-light  mlsr  ?lru  commfault))
        (assert  (status-light  mlsa  ?lru  commfault))
        (assert  (status-light  mlse  ?lru  commfault))
        (retract  ?x)
        (assert  (msbls-status  ?lru  commfault)))

;------------------------------------------------------------------------

(defrule  msbls-commfault-clear

;;      IF
;;              A MSBLS LRU was previously commfaulted  AND
;;              The commfault flag for that LRU is now off
;;      THEN
;;              Notify the operator that the commfault has cleared (unless
;;                      the whole string was down)
;;              Conclude the LRU has the status indicated by the fail flag
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-status  ?lru  commfault)
        (msbls-flag  commfault  ?lru  off)
        (msbls-flag fail ?lru range ?flagr)
        (msbls-flag fail ?lru azimuth ?flaga)
        (msbls-flag fail ?lru elevation ?flage)
        (prev-string-cf  pass  ?lru  ?string-flag)
        (msbls-lru-quality  ?lru  range  ?range-status)
        (msbls-lru-quality  ?lru  azimuth  ?azimuth-status)
        (msbls-lru-quality  ?lru  elevation  ?elevation-status)
        =>
        (if  (eq  ?string-flag off)
            then
                (assert  (event msbls off-nominal alt
                        "Commfault clear on MSBLS " ?lru)))
        (retract  ?x)
        (if  (||  (eq ?flagr on)
                  (eq ?flaga on)
                  (eq ?flage on))
            then
                (assert  (status-light  mlsr  ?lru  fail))
                (assert  (status-light  mlsa  ?lru  fail))
                (assert  (status-light  mlse  ?lru  fail))
                (assert  (msbls-status  ?lru  fail))
            else
                (assert  (status-light  mlsr  ?lru  ?range-status))
                (assert  (status-light  mlsa  ?lru  ?azimuth-status))
                (assert  (status-light  mlse  ?lru  ?elevation-status))
                (assert  (msbls-status  ?lru  avail))))

;------------------------------------------------------------------------------

(defrule  msbls-failed

;;      IF
;;              A MSBLS LRU was previously available  AND
;;              The fail flag for that LRU is now on
;;      THEN
;;              Notify the operator of the LRU failure
;;              Conclude the LRU is no longer available due to RM failure
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-status ?lru avail)
        (msbls-flag fail ?lru range|azimuth|elevation on)
        =>
        (assert  (event msbls off-nominal alt
                "MSBLS " ?lru " has been failed by RM"))
        (assert  (status-light  mlsr  ?lru  fail))
        (assert  (status-light  mlsa  ?lru  fail))
        (assert  (status-light  mlse  ?lru  fail))
        (retract  ?x)
        (assert  (msbls-status  ?lru  fail)))
```

```
;----------------------------------------------------------------------------

(defrule  msbls-power-off

;;      IF
;;              A MSBLS LRU was previously powered on  AND
;;              The power indicator for that LRU is now off
;;      THEN
;;              Notify the operator that the LRU has lost power
;;              Conclude the LRU is not available due to loss of power
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-status  ?lru  ~power-off)
        (msbls-flag  power  ?lru  off)
        =>
        (assert  (event msbls off-nominal alt
                "MSBLS " ?lru " has been powered off"))
        (assert  (status-light  mlsr  ?lru  off))
        (assert  (status-light  mlsa  ?lru  off))
        (assert  (status-light  mlse  ?lru  off))
        (retract  ?x)
        (assert  (msbls-status  ?lru  power-off)))

;----------------------------------------------------------------------------

(defrule  msbls-power-on

;;      IF
;;              A MSBLS LRU was previously powered off  AND
;;              The power indicator for that LRU is now on
;;      THEN
;;              Notify the operator that the LRU has been powered on
;;              Conclude the LRU has the status indicated by the fail flag
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-status  ?lru  power-off)
        (msbls-flag  power  ?lru  on)
        (msbls-flag fail ?lru range ?flagr)
        (msbls-flag fail ?lru azimuth ?flaga)
        (msbls-flag fail ?lru elevation ?flage)
        (msbls-lru-quality  ?lru  range  ?range-status)
        (msbls-lru-quality  ?lru  azimuth  ?azimuth-status)
        (msbls-lru-quality  ?lru  elevation  ?elevation-status)
        =>
        (assert  (event msbls off-nominal alt
                "MSBLS " ?lru " has been powered on"))
        (retract  ?x)
        (if  (||  (eq ?flagr on)
                  (eq ?flaga on)
                  (eq ?flage on))
            then
                (assert  (status-light  mlsr  ?lru  fail))
                (assert  (status-light  mlsa  ?lru  fail))
                (assert  (status-light  mlse  ?lru  fail))
                (assert  (msbls-status  ?lru  fail))
            else
                (assert  (status-light  mlsr  ?lru  ?range-status))
                (assert  (status-light  mlsa  ?lru  ?azimuth-status))
                (assert  (status-light  mlse  ?lru  ?elevation-status))
                (assert  (msbls-status  ?lru  avail))))

;------------------------------------------------------------------------------

(defrule  three-msbls-avail

;;      IF
;;              All MSBLS LRUs are available
;;      THEN
;;              The number of available LRUs is 3
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-num-avail  ~3)
        (msbls-status  1  avail)
        (msbls-status  2  avail)
        (msbls-status  3  avail)
        =>
        (retract  ?x)
        (assert  (msbls-num-avail  3)))

;------------------------------------------------------------------------------

(defrule  two-msbls-avail

;;      IF
;;              MSBLS LRU A is available  AND
;;              MSBLS LRU B is available  AND
;;              MSBLS LRU C is not available
;;      THEN
;;              The number of available LRUs is 2
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-num-avail  ~2)
        (msbls-status  ?lru-a  avail)
        (msbls-status  ?lru-b&~?lru-a  avail)
        (msbls-status  ?lru-c  ~avail)
        =>
        (retract  ?x)
        (assert  (msbls-num-avail  2)))

;------------------------------------------------------------------------------

(defrule  one-msbls-avail

;;      IF
;;              MSBLS LRU A is available  AND
;;              MSBLS LRU B is not available  AND
;;              MSBLS LRU C is not available
;;      THEN
;;              The number of available LRUs is 1
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-num-avail  ~1)
        (msbls-status  ?lru-a  avail)
        (msbls-status  ?lru-b  ~avail)
        (msbls-status  ?lru-c&~?lru-b ~avail)
        =>
        (retract  ?x)
        (assert  (msbls-num-avail  1)))

;---------------------------------------------------------------------------------

(defrule  no-msbls-avail

;;      IF
;;              All MSBLS LRUs are unavailable
;;      THEN
;;              The number of available LRUs is 0
;;      END

        (sub-phase  msbls  availability)
        ?x <- (msbls-num-avail  ~0)
        (msbls-status  1  ~avail)
        (msbls-status  2  ~avail)
        (msbls-status  3  ~avail)
        =>
        (retract  ?x)
        (assert  (msbls-num-avail  0)))


;;*******************************************************************************
;;
;;;  GROUP (3.10.2)
;;      MSBLS Lockon Status
;;
;;      This group determines how many LRUs are locked onto range, azimuth,
;;      and elevation.
;;
;;;  CONTROL FACTS
;       (sub-phase  msbls  lockon)
;;
;;;  CONTAINING GROUP
;;      MSBLS
;;
;;*******************************************************************************

(defrule  check-channel

;;      IF
;;              At least one MSBLS LRU is available AND
;;              No LRU is locked on one of the measurements  AND
;;              The vehicle is below 13000 feet
;;      THEN
;;              Ask that the MSBLS channel be verified
;;      END

        (sub-phase  msbls  lockon)
        (msbls-num-avail  ~0)
        (msbls-lock  1  ?measurement  off)
        (msbls-lock  2  ?measurement  off)
        (msbls-lock  3  ?measurement  off)
        (runway  pass  ?runway)
        (alitutde  ?alt)
        (test  (<  ?alt  13000))
        =>
        (assert  (recommend msbls check-channel off-nominal alt
                "Need to verify MSLBS channel is " =(lookup-msbls ?runway))))

;-----------------------------------------------------------------------------

(defrule  three-msbls-locked

;;      IF
;;              All LRUs are available AND
;;              All LRUs are locked on a measurement
;;      THEN
;;              The number locked for that measurement is 3
;;              If the number locked was previously 0, then notify the
;;                      operator that MSBLS is locking on
;;      END

        (sub-phase  msbls  lockon)
        (msbls-num-avail 3)
        ?x <- (msbls-num-locked  ?measurement  ?old-number&~3)
        (msbls-lock  1  ?measurement  on)
        (msbls-lock  2  ?measurement  on)
        (msbls-lock  3  ?measurement  on)
        =>
        (if  (=  0  ?old-number)
            then
                (assert  (event msbls nominal alt
                        "MSLBS is locking onto " ?measurement)))
        (retract  ?x)
        (assert  (msbls-num-locked  ?measurement  3)))

;-----------------------------------------------------------------------------

(defrule  two-msbls-locked

;;      IF
;;              LRU A is locked on a measurement  AND
;;              LRU B is locked on the same measurement  AND
;;              LRU C is not lock on the measurement
;;                 or not available
;;      THEN
;;              The number of LRUs locked on that measurement is 2
;;              If the number locked was previously 0, then notify the
;;                      operator that MSBLS is locking on
;;      END

        (sub-phase  msbls  lockon)
        ?x <- (msbls-num-locked  ?measurement  ?old-number&~2)
        (msbls-lock  ?lru-a  ?measurement  on)
        (msbls-lock  ?lru-b&~?lru-a  ?measurement  on)
        (or (msbls-lock  ?lru-c  ?measurement  off)
            (msbls-num-avail 2))
        =>
        (if  (=  0  ?old-number)
            then
                (assert  (event msbls nominal alt
                        "MSLBS is locking onto " ?measurement)))
        (retract  ?x)
        (assert  (msbls-num-locked  ?measurement  2)))

;-----------------------------------------------------------------------------
```

```
(defrule  one-msbls-locked

;;      IF
;;              LRU A is locked on a measurement  AND
;;              LRU B is not locked on the measurement  AND
;;              LRU C is not locked on the measurement
;;      THEN
;;              The number of LRUs locked on that measurement is 1
;;              IF the number locked previously was 0, then notify the
;;                      operator that MSBLS is locking on
;;      END

        (sub-phase  msbls  lockon)
        ?x <- (msbls-num-locked  ?measurement  ?old-number&~1)
        (msbls-lock  ?lru-a  ?measurement  on)
        (msbls-lock  ?lru-b  ?measurement  off)
        (msbls-lock  ?lru-c&~?lru-b  ?measurement  off)
        =>
        (if  (=  0  ?old-number)
            then
                (assert  (event msbls nominal alt
                        "MSLBS is locking onto " ?measurement)))
        (retract  ?x)
        (assert  (msbls-num-locked  ?measurement  1)))

;---------------------------------------------------------------------------

(defrule  no-msbls-locked

;;      IF
;;              At least 1 LRU is available
;;              No LRU is locked on a measurement
;;      THEN
;;              The number of LRUs locked for that measurement is 0
;;              Notify the operator that MSBLS lost lock
;;      END

        (sub-phase  msbls  lockon)
        ?x <- (msbls-num-locked  ?measurement  ~0)
        (msbls-num-avail ?num)
        (test (>= ?num 1))
        (msbls-lock  1  ?measurement  off)
        (msbls-lock  2  ?measurement  off)
        (msbls-lock  3  ?measurement  off)
        =>
        (assert  (event msbls nominal alt
                        "MSLBS lost lock in " ?measurement))
        (retract  ?x)
        (assert  (msbls-num-locked  ?measurement  0)))


;;***************************************************************************
;;
;;;  GROUP (3.10.3)
;;      MSBLS Error Checks
;;
;;      This group check measurement errors and determines the quality of
;;      the three LRUs.
```

```
;;
;;;  CONTROL FACTS
;       (sub-phase  msbls  quality)
;;
;;;  CONTAINING GROUP
;;      MSBLS
;;
;;************************************************************************

(defrule initial-msbls-check

;;      IF
;;              The no quality statement has yet been made about a
;;                      measurement  AND
;;              The measurement bias is within tolerance  AND
;;              The measurement noise is within tolerance
;;      THEN
;;              The report that the measurement is good
;;      END

        (declare (salience 10))
        (sub-phase  msbls  quality)
        ?x <- (last-msbls-report  ?lru  ?measurement  bias  unknown)
        ?y <- (last-msbls-report  ?lru  ?measurement  noise  unknown)
        (msbls-error  ?lru  ?measurement  bias  under)
        (msbls-error  ?lru  ?measurement  noise  under)
        =>
        (assert  (event msbls nominal alt
                "MSBLS " ?lru " " ?measurement " is good"))
        (retract  ?x)
        (retract  ?y)
        (assert  (last-msbls-report  ?lru  ?measurement  bias  under))
        (assert  (last-msbls-report  ?lru  ?measurement  noise  under)))
;-----------------------------------------------------------------------

(defrule msbls-error-change

;;      IF
;;              Either the noise or bias on a measurement has a different
;;                      status than it did previously
;;      THEN
;;              Notify the operator of the new status
;;      END

        (sub-phase  msbls  quality)
        ?x <- (last-msbls-report  ?lru  ?measurement  ?error
                        ?old-status)
        (msbls-error  ?lru  ?measurement  ?error  ?status&~?old-status)
        (units  ?measurement  ?units)
        =>
        (if  (! (eq ?status under))
            then
                (bind  ?a  (msbls-error ?lru ?measurement ?error))
                (assert  (event msbls off-nominal alt
                        "MSBLS " ?lru " " ?measurement " has a " ?error
                        " of " ?a  ?units))
            else
                (if  (! (eq ?old-status unknown))
                    then
```

```
                              (assert  (event msbls off-nominal alt
                                      "MSBLS " ?lru " " ?measurement " " ?error
                                      " has cleared up"))))
        (retract  ?x)
        (assert  (last-msbls-report  ?lru  ?measurement  ?error  ?status)))

;--------------------------------------------------------------------------------

(defrule  msbls-lru-quality-1

;;      IF
;;              A MSBLS LRU is unavailable or unlocked in a measurement
;;      THEN
;;              That LRU has no quality rating for that measurement
;;      END

        (sub-phase  msbls  quality)
        ?x <- (msbls-lru-quality  ?lru  ?measurement  ~none)
        (or  (msbls-status  ?lru  ~avail)
             (msbls-lock  ?lru  ?measurement  off ))
        (measurement-name  ?name&mlsr|mlsa|mlse  ?measurement)
        =>
        (assert  (status-light  ?name  ?lru  none))
        (retract  ?x)
        (assert  (msbls-lru-quality  ?lru  ?measurement  none)))

;--------------------------------------------------------------------------------

(defrule  msbls-lru-quality-2

;;      IF
;;              A MSBLS LRU is available  AND
;;              The LRU is locked on a measurement  AND
;;              The noise and bias ratings on the measurement indicate
;;                      a quality rating different from the one previously
;;                      given to the LRU
;;      THEN
;;              Note the new quality rating for the LRU
;;      END

        (sub-phase  msbls  quality)
        (msbls-status  ?lru  avail)
        (msbls-lock  ?lru  ?measurement  on)
        (msbls-error  ?lru  ?measurement  bias  ?bias)
        (msbls-error  ?lru  ?measurement  noise  ?noise)
        (msbls-quality  ?bias  ?noise  ?quality)
        ?x <- (msbls-lru-quality  ?lru  ?measurement  ~?quality)
        (measurement-name  ?name&mlsr|mlsa|mlse  ?measurement)
        =>
        (assert  (status-light  ?name  ?lru  ?quality))
        (retract  ?x)
        (assert  (msbls-lru-quality  ?lru  ?measurement  ?quality)))


;;*******************************************************************************
;;
;;;  GROUP (3.10.4)
;;      MSBLS Flag Monitoring
;;
```

```
;;      This group watches for changes in the MSBLS data good flags and
;;      filter flags.
;;
;;;  CONTROL FACTS
;       (sub-phase  msbls  watch-flags)
;;
;;;  CONTAINING GROUP
;;      MSBLS
;;
;;*****************************************************************************

(defrule  msbls-filter-flag-changed

;;      IF
;;              The value of the MSBLS filter flag is different from
;;                its previous value
;;      THEN
;;              Conclude that the value has changed
;;              Notify the operator if the new value is "process"
;;      END

        (sub-phase  msbls  watch-flags)
        (filter-flag  pass  ?meas&mlsr|mlsa|mlse  ?flag&~off)
        ?x <- (prev-filter-flag  pass  ?meas  ~?flag)
        (measurement-name  ?meas  ?measurement)
        =>
        (retract  ?x)
        (assert  (prev-filter-flag  pass  ?meas  ?flag))
        (if  (eq  ?flag  process)
            then
                (assert  (event msbls nominal alt
                        " MSBLS " ?measurement
                        " filter flag changed to the "
                        ?flag " position "))))

;-----------------------------------------------------------------------------

(defrule  msbls-data-good-flag-changed

;;      IF
;;              The current value of a MSBLS data-good flag is different from
;;                      its previous value
;;      THEN
;;              Notify the operator of the new value
;;      END

        (sub-phase  msbls  watch-flags)
        (data-good  pass  ?meas&mlsr|mlsa|mlse  ?flag)
        ?x <- (prev-data-good  pass  ?meas  ~?flag)
        (measurement-name  ?meas  ?measurement)
        =>
        (retract  ?x)
        (assert  (prev-data-good  pass  ?meas  ?flag))
        (assert  (event msbls nominal alt
                "MSBLS " ?measurement " data-good flag is " ?flag)))

;-----------------------------------------------------------------------------

(defrule  msbls-dilemma
;;      IF
;;              MSBLS dilemma flag is on for any measurement
;;      THEN
;;              Warn the operator
;;      END

        (sub-phase  msbls  watch-flags)
        (msbls-dilemma  ?measurement  on)
        =>
        (assert  (event  msbls  off-nominal  alt
                "MSBLS " ?measurement " is in dilemma")))

;;****************************************************************************
;;
;;;  GROUP (3.10.5)
;;      MSBLS Recommendations
;;
;;      This group determines what actions need to be taken on the MSBLS
;;      to keep it from corrupting the nav state.
;;
;;;  CONTROL FACTS
;       (sub-phase  msbls  recommendation)
;;
;;;  CONTAINING GROUP
;;      MSBLS
;;
;;****************************************************************************

(defrule  three-level-msbls-deselect-1

;;      IF
;;              3 LRUs are available  AND
;;              2 LRUs are locked on  AND
;;              1 LRU is bad
;;      THEN
;;              Recommend deselecting the bad LRU
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  3)
        (msbls-num-locked  ?measurement  2)
        (msbls-lru-quality  ?lru-a  ?measurement  bad)
        (msbls-lru-quality  ?lru-b  ?measurement  good)
        =>
        (assert  (recommend  msbls  deselect-msbls-lru  off-nominal  alt
                "Need to power off MSBLS " ?lru-a " due to bad " ?measurement)))

;----------------------------------------------------------------------------

(defrule  three-level-msbls-force-tacan-1

;;      IF
;;              3 LRUs are available  AND
;;              2 LRUs are locked on  AND
;;              2 LRUs are bad in the same measurement
;;      THEN
;;              Recommend forcing TACAN
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  3)
        (msbls-num-locked  ?measurement  2)
        (msbls-lru-quality  ?lru-a _?measurement  bad)
        (msbls-lru-quality  ?lru-b&~lru-a ?measurement  bad)
        =>
        (assert  (recommend  msbls  force-tacan  off-nominal  alt
                "Need to force TACAN because of two bad MSBLS LRUs")))

;-----------------------------------------------------------------------------

(defrule  three-level-msbls-rm-fail

;;      IF
;;              3 LRUs are available  AND
;;              3 LRUs are locked on  AND
;;              1 LRU is bad
;;      THEN
;;              Recommend deselecting (for a noise problem) or waiting for
;;                      RM isolation (for a bias problem)
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  3)
        (msbls-num-locked  ?measurement  3)
        (msbls-lru-quality  ?lru-a  ?measurement  bad)
        (msbls-error  ?lru-a  ?measurement  bias  ?bias)
        (msbls-lru-quality  ?lru-b _?measurement  good)
        (msbls-lru-quality  ?lru-c&~lru-b  ?measurement  good)
        =>
        (if  (eq  ?bias  over)
           then
              (assert  (recommend  msbls  msbls-rm-fail  off-nominal  alt
                      "RM should fail MSBLS " ?lru-a " due to "
                      ?measurement " bias"))
           else
              (assert  (recommend  msbls  deselect-msbls  off-nominal  alt
                      "Need to power off MSBLS " ?lru-a " due to "
                      ?measurement " noise"))))

;-----------------------------------------------------------------------------

(defrule  three-level-msbls-deselect-2

;;      IF
;;              3 LRUs are available  AND
;;              3 LRUs are locked on  AND
;;              2 LRUs are bad in the same measurement
;;      THEN
;;              Recommend deselecting the bad LRUs
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  3)
        (msbls-num-locked  ?measurement  3)
        (msbls-lru-quality  ?lru-a _?measurement  bad)
        (msbls-lru-quality  ?lru-b&~lru-a   ?measurement  bad)
        (msbls-lru-quality  ?lru-c  ?measurement  good)
        =>
        (assert  (recommend  msbls  deselect-msbls-lru  off-nominal  alt
                "Need to power off MSBLS " ?lru-a " and "
                ?lru-b " due to bad " ?measurement)))

;-----------------------------------------------------------------------------

(defrule  three-level-msbls-force-tacan-2

;;      IF
;;              3 LRUs are available  AND
;;              3 LRUs are locked on  AND
;;              3 LRUs are bad on the same measurement
;;      THEN
;;              Recommend forcing TACAN
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  3)
        (msbls-num-locked  ?measurement  3)
        (msbls-lru-lock  1  ?measurement  bad)
        (msbls-lru-lock  2  ?measurement  bad)
        (msbls-lru-lock  3  ?measurement  bad)
        =>
        (assert  (recommend  msbls  force-tacan  off-nominal  alt
                "Need to force TACAN due to bad " ?measurement
                " in all MSBLS LRUs")))

;-----------------------------------------------------------------------------

(defrule  two-level-msbls-deselect

;;      IF
;;              2 LRUs are available  AND
;;              2 LRUs are locked on  AND
;;              1 LRU is bad
;;      THEN
;;              Recommend deselecting the bad LRU
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  2)
        (msbls-num-locked  ?measurement  2)
        (msbls-lru-quality  ?lru-a  ?measurement  bad)
        (msbls-lru-quality  ?lru-b  ?measurement  good)
        =>
        (assert  (recommend  msbls  deselect-msbls-lru  off-nominal  alt
                "Need to power off MSBLS " ?lru-a " due to bad " ?measurement)))

;-----------------------------------------------------------------------------

(defrule  two-level-msbls-force-tacan

;;      IF
;;              2 LRUs are available  AND
;;              2 LRUs are locked on  AND
;;              2 LRUs are bad in the same measurement
;;      THEN
;;              Recommend forcing TACAN
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  2)
        (msbls-num-locked  ?measurement  2)
        (msbls-lru-quality  ?lru-a  ?measurement  bad)
        (msbls-lru-quality  ?lru-b&~lru-a  ?measurement  bad)
        =>
        (assert  (recommend  msbls  force-tacan  off-nominal  alt
                "Need to force TACAN due to bad MSBLS " ?measurement)))

;-------------------------------------------------------------------------------

(defrule  one-level-msbls-force-tacan

;;      IF
;;              1 LRU is available  AND
;;              1 LRU is locked on  AND
;;              1 LRU is bad
;;      THEN
;;              Recommend forcing TACAN
;;      END

        (sub-phase  msbls  recommendation)
        (msbls-num-avail  1)
        (msbls-num-locked  ?measurement  1)
        (msbls-lru-quality  ?lru  ?measurement  bad)
        =>
        (assert  (recommend  msbls  force-tacan  off-nominal  alt
                "Need to force TACAN due to bad MSBLS " ?measurement)))

;-------------------------------------------------------------------------------

(defrule  do-not-force-tacan

;;      IF
;;              Forcing TACAN is recommended AND
;;              TACAN is no go
;;      THEN
;;              Cancel force TACAN recommendation AND
;;              Recommend powering off MSBLS
;;      END

        (sub-phase  msbls  recommendation)
        ?x <-  (recommend  msbls  force-tacan  off-nominal  alt $?)
        (selected-tacan ?measurement no-go)
        =>
        (retract ?x)
        (assert  (recommend  msbls  do-not-force-tacan  off-nominal  alt
           "Need to power off MSBLS because TACAN is no-go in " ?measurement)))




;;*****************************************************************************
;;
;;;  GROUP (3.10.6)
;;      Effect of MSBLS on State Errors
;;
;;      This group checks to see if MSBLS processing makes the state error
;;      worse.
;;
```

```
;;;  CONTROL FACTS
;       (sub-phase  msbls  watch-state)
;;
;;;  CONTAINING GROUP
;;
;;
;;****************************************************************************

(defrule error-before-msbls

;;      IF
;;              At least 1 lru is locked on range  AND
;;              No MSBLS is being processed
;;      THEN
;;              Remember the current worst-axis state error
;;      END

        (sub-phase  msbls  watch-state)
        (msbls-num-locked  range  ~0)
        (filter-flag  pass  mlsr  ~process)
        (filter-flag  pass  mlsa  ~process)
        (filter-flag  pass  mlse  ~process)
        (gnd-state  pass  worst-axis  ?status)
        ?x <- (error-before-msbls  ~?status)
        =>
        (retract  ?x)
        (assert  (error-before-msbls  ?status)))

;---------------------------------------------------------------------------

(defrule  error-after-msbls

;;      IF
;;              MSBLS is being processed  AND
;;              The state error is worse than before MSBLS was processed
;;      THEN
;;              Recommend forcing TACAN
;;      END

        (sub-phase  msbls  watch-state)
        (error-before-msbls  ?before)
        (filter-flag  pass  mlsr|mlsa|mlse  process)
        (gnd-state  pass  worst-axis  ?after&~?before)
        (max-miscompare  ?before  ?after  ?after)
        =>
        (assert  (recommend  msbls  force-tacan  off-nominal  alt
                "Need to force TACAN because MSBLS is causing error growth")))
```

3.11 High Speed Trajectory Determinator

```
;;******************************************************************************
;;
;;;  GROUP  (3.11)
;;      HSTD monitoring
;;
;;      These rules have the task of determining the status of the HSTD state
;;      vector.  THESE RULES DEPEND PRIMARILY ON OPERATOR INPUT.  The rules
;;      can detect when the filter is stopped, and they can detect some
;;      situations where the filter is not converged.  In addition, the
;;      operator can indicate when the filter is bad.  The operator must
;;      specify when the filter is good; the rules never do that automatically.
;;
;;
;;;  CONTROL FACTS
;       none
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;******************************************************************************


;;;  FACTS

(deffacts monitoring-hstd-phases        ; These facts list the sequence of
                                        ;  sub-phases in the monitoring phase
                                        ;  of the hstd rules.
        (first-sub-phase  hstd  monitoring  status)
                                        ; There is only 1 sub-phase: hstd-status
)


(deffacts initial-hstd-facts            ; These facts represent assumptions
                                        ;  about the HSTD vector prior to
                                        ;  receiving any data.
        (hstd stopped)                  ; The filter is not running
        (restart-time 0.0)              ; Time of last restart not yet known
)


(defrule hstd-start

;;      IF
;;              The HSTD has not been running  AND
;;              The 'stopped' indicator is off
;;      THEN
;;              Conclude the HSTD is running but has not converged
;;      END

        (sub-phase  hstd  status)
        ?x <- (hstd  stopped)
        (hstd-stop-flag  off)
        =>
        (assert  (status-light  state  ground  bad))
        (retract  ?x)
        (assert  (hstd  bad)))

;-----------------------------------------------------------------------------
```

```
(defrule hstd-bad

;;      IF
;;              The HSTD was good  AND
;;              The operator entered the HSTD-bad indicator
;;      THEN
;;              Conclude the HSTD is bad (not converged)
;;      END

        (sub-phase  hstd  status)
        ?x <- (hstd  good)
        ?y <- (operator-input  hstd  bad)
        =>
        (assert  (status-light  state  ground  bad))
        (retract  ?x)
        (retract  ?y)
        (assert  (hstd  bad)))

;-----------------------------------------------------------------------

(defrule hstd-good

;;      IF
;;              The HSTD was bad  AND
;;              The operator entered the HSTD-good indicator  AND
;;              At least 10 seconds have elapsed since the last restart
;;      THEN
;;              Conclude the HSTD is good
;;      END

        (sub-phase  hstd  status)
        ?x <- (hstd  bad)
        ?y <- (operator-input  hstd  good)
        (restart-time  ?restart-time)
        (current-time  ?time)
        (test  (>= (- ?time ?restart-time) 10.0))
        =>
        (assert  (status-light  state  ground  good))
        (retract  ?x)
        (retract  ?y)
        (assert  (hstd  good)))

;-----------------------------------------------------------------------

(defrule hstd-stopped

;;      IF
;;              The HSTD was running  AND
;;              The stopped indicator is on
;;      THEN
;;              Conclude the HSTD has been stopped
;;      END

        (sub-phase  hstd  status)
        ?x <- (hstd  ~stopped)
        (hstd-stop-flag  on)
        =>
        (assert  (status-light  state  ground  stopped))
        (retract  ?x)
        (assert  (hstd  stopped)))
```

```
;---------------------------------------------------------------------------

(defrule hstd-editing

;;;     IF
;;;             The HSTD was good  AND
;;;             Less that 3 stations are being processed  AND
;;;             A given station is not being excluded  AND
;;;             There is data coming from that station  AND
;;;             At least one good measurement of a given type was
;;;                     available from that station  AND
;;;             All of the measurements of that type from that station
;;;                     were edited by the filter
;;;     THEN
;;;             Conclude the HSTD is bad
;;;     END

        (sub-phase  hstd  status)
        ?x <- (hstd  good)
        (or  (exclude  ?station-1  on)
             (tracking-avail  ?station-1  0))
        (exclude  ?station-2&~?station-1  off)
        (tracking-avail  ?station-2  ~0)
        (tracking-good  ?station-2  ?meas  ?num-good)
        (test  (>= ?num-good 1))
        (tracking-edit  ?station-2  ?meas  ?num-good)
        =>
        (assert  (status-light  state  ground  bad))
        (retract  ?x)
        (assert  (hstd  bad)))

;---------------------------------------------------------------------------

(defrule hstd-prop

;;      IF
;;              The HSTD was good  AND
;;              The prop flag is on
;;      THEN
;;              Conclude the HSTD is bad
;;      END

        (sub-phase  hstd  status)
        ?x <- (hstd  good)
        (hstd-prop-flag  on)
        =>
        (assert  (status-light  state  ground  bad))
        (retract  ?x)
        (assert  (hstd  bad)))

;---------------------------------------------------------------------------

(defrule hstd-covariance

;;      IF
;;              The HSTD was good  AND
;;              The RSS position or velocity covariance diagonals are
;;                      too large
;;      THEN
;;              Conclude the HSTD is bad
;;      END

        (sub-phase  hstd  status)
        ?x <- (hstd  good)
        (hstd-covariance  ?  over)
        =>
        (assert  (status-light  state  ground  bad))
        (retract  ?x)
        (assert  (hstd  bad)))

;-----------------------------------------------------------------------------

(defrule hstd-restart

;;      IF
;;              The HSTD is available  AND
;;              The HSTD restart flag is on
;;      THEN
;;              Conclude the HSTD is bad
;;              Record the current time as the time of the last restart
;;      END

        (sub-phase  hstd  status)
        (hstd-status  available)
        ?x <- (hstd  ?)
        (hstd-restart-flag  on)
        ?y <- (restart-time  ?restart-time)
        (current-time  ?time&~?restart-time)
        =>
        (assert  (status-light  state  ground  bad))
        (retract  ?x)
        (assert  (hstd  bad))
        (retract  ?y)
        (assert  (restart-time  ?time)))
```

3.12 Control Flow

```
;;****************************************************
;;
;;  GROUP
;;    Control (no reference number)
;;
;;    This group handles initial start up of rule
;;    execution, and controls the phasing of rule
;;    groups.
;;
;;  CONTROL FACTS
;;    none
;;
;;  CONTAINING GROUP
;;    Entry
;;
;;****************************************************
;;;  Facts

(deffacts  control-initial-phase
        (phase  fact-assertion)
)

;;--------------------------------------------------

(deffacts  control-phases
        (next-phase  fact-assertion  monitoring)
        (next-phase  monitoring  analysis)
        (next-phase  analysis  output)
        (next-phase  output  fact-assertion)
)

;;--------------------------------------------------

(defrule  control-kickoff
        (phase  fact-assertion)
        =>
        (call  (operator-input))
        (call  (check-facts  off))
        (call  (fact-assertion))
        (call  (display-time))
        (call  (check-facts  on)))

;;----------------------------------------------------

(defrule control-change-phases
        (declare (salience -1000))
        (next-phase  ?current-phase  ?next-phase)
        (not (end-of-data $?))
        ?x <- (phase  ?current-phase)
        =>
        (retract ?x)
        (assert (phase ?next-phase)))

;;----------------------------------------------------

(defrule control-end-of-cycle
        (declare (salience -999))
        (single step)
        (phase output)
        =>
        (halt))

;;------------------------------------------------------

(defrule control-kickoff-subphase
        (declare  (salience 100))
        (phase  ?phase)
        (first-sub-phase  ?module  ?phase  ?subphase)
        =>
        (assert  (sub-phase  ?module  ?subphase)))

;;------------------------------------------------------

(defrule control-next-subphase
        (declare  (salience -100))
        ?x <- (sub-phase  ?module  ?current)
        (next-sub-phase  ?module  ?current  ?next)
        =>
        (retract  ?x)
        (assert  (sub-phase ?module  ?next)))

;;------------------------------------------------------

(defrule control-last-subphase
        (declare  (salience -200))
        ?x <- (sub-phase  $?)
        =>
        (retract  ?x))

;;------------------------------------------------------
```

3.13 Operator Input

```
;;****************************************************************************
;;
;;;  GROUP  Operator Inputs
;;
;;      This group takes the following operator inputs and makes appropriate
;;      adjustments to the fact base;
;;              stop
;;              subsystem
;;              delta-state
;;              bfs-no-go
;;              runway
;;              toggle-tacan
;;      The hstd status is handled by the hstd rules because proper handling
;;      involves coordination with other hstd flags (see hstd.r).
;;
;;;  CONTROL FACTS
;       (phase  fact-assertion)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;****************************************************************************


(defrule  operator-stop

;;      IF
;;              The operator issued the stop command
;;      THEN
;;              Retract the operator's command
;;              Halt CLIPS
;;      ENDIF

        (phase  fact-assertion)
        ?x <- (operator-input  stop)
        =>
        (retract  ?x)
        (halt))


;----------------------------------------------------------------------------

(defrule  operator-subsystem

;;      IF
;;              The operator commanded a new subsystem window
;;      THEN
;;              Retract the operator's command
;;              Reconfigure the screen to show the commanded subsystem
;;      ENDIF

        (phase  fact-assertion)
        ?x <- (operator-input  subsystem  ?number)
        =>
        (retract  ?x)
        (call  (select-subsystem  ?number)))


;----------------------------------------------------------------------------
```

```
(defrule  operator-delta-state

;;      IF
;;              The operator issued a delta-state command (position-only,
;;                      position-and-velocity, or none)  AND
;;              No delta-state was in work previously
;;      THEN
;;              Retract the operator's command
;;              If the command was anything but "none", note that a delta-state
;;                      is in work and note the type of delta-state
;;      ENDIF

        (phase  fact-assertion)
        ?x <- (operator-input  delta-state  ?type)
        (not  (need-delta-state  $?))
        =>
        (retract  ?x)
        (call  (update-configuration  delta-state  ?type))
        (if  (! (eq ?type none))
            then
                (assert  (need-delta-state  ?type))))


;---------------------------------------------------------------------------------

(defrule  operator-changed-delta-state

;;      IF
;;              The operator issued a delta-state command (position-only,
;;                      position-and-velocity, or none)  AND
;;              A delta-state was already in work
;;      THEN
;;              Retract the operator's command
;;              If the command was anything but "none", change the type
;;                      of delta-state in work; otherwise, note that no
;;                      delta-state is in work.
;;      ENDIF

        (phase  fact-assertion)
        ?x <- (operator-input  delta-state  ?type)
        ?y <- (need-delta-state  $?)
        =>
        (retract  ?x  ?y)
        (call  (update-configuration  delta-state  ?type))
        (if  (! (eq ?type none))
            then
                (assert  (need-delta-state  ?type))))


;---------------------------------------------------------------------------------

(defrule  operator-bfs-no-go

;;      IF
;;              The operator issued the BFS-NO-GO command
;;      THEN
;;              Retract the operator's command
;;              Change the BFS status to no-go
;;      ENDIF
```

```
        (phase  fact-assertion)
        ?x <- (operator-input  bfs-no-go)
        ?y <- (bfs-status  $?)
        =>
        (call  (update-configuration  bfs  no-go))
        (retract  ?x  ?y)
        (assert  (bfs-status  no-go)))


;-----------------------------------------------------------------------------------

(defrule  operator-runway-selection

;;      IF
;;              The operator has completed a runway selection
;;      THEN
;;              Change the desired runway to the specified slot
;;              Change the desired TACAN to the primary slot in the same
;;                      area as the runway

        (phase  fact-assertion)
        ?x <- (operator-input  runway  ?rw-slot)
        ?a <- (runway  desired  $?)
        ?b <- (desired-tacan  $?)
        ?c <- (desired-channel  $?)
        =>
        (retract  ?x)
        (if  (&&  (>=  ?rw-slot  1)  (<=  ?rw-slot  30))
            then
                (retract  ?a  ?b  ?c)
                (bind  ?name  (lookup-rw-name  ?rw-slot))
                (bind  ?area  (trunc  (/  (+ ?rw-slot 1) 2)))
                (bind  ?tac-slot  (-  (* ?area 2) 1))
                (bind  ?channel  (lookup-tacan  ?tac-slot))
                (assert  (runway  desired  ?rw-slot))
                (assert  (desired-tacan  ?tac-slot))
                (assert  (desired-channel  ?channel))
                (call  (update-configuration  runway  ?name))
                (call  (update-configuration  tacan  ?channel))
            else
                (assert  (event  site  nominal alt
                        "There is no runway slot " ?rw-slot " in the table"))))

;-----------------------------------------------------------------------------------

(defrule  operator-toggle-tacan

;;      IF
;;              The operator issued the TOGGLE command  AND
;;              Toggle capability is available
;;      THEN
;;              Retract the operator's command
;;              Change the desired TACAN to the other station in the
;;                      current area
;;      ENDIF

        (phase  fact-assertion)
        ?x <- (operator-input  toggle-tacan)
        (toggle-available  yes)
        ?y <- (desired-tacan  ?current-slot)
```

```
          (same-area  ?current-slot  ?other-slot)
          ?z <- (desired-channel  $?)
          =>
          (bind  ?channel  (lookup-tacan  ?other-slot))
          (call  (update-configuration  tacan  ?channel))
          (retract  ?x  ?y  ?z)
          (assert  (desired-tacan  ?other-slot))
          (assert  (desired-channel  ?channel)))


;----------------------------------------------------------------------------

(defrule  operator-cant-toggle

;;        IF
;;                The operator issued the TOGGLE command  AND
;;                Toggle capability is not available
;;        THEN
;;                Retract the operator's command
;;                Inform the operator that toggle is not available
;;        ENDIF

          (phase  fact-assertion)
          ?x <- (operator-input  toggle-tacan)
          (toggle-available  no)
          =>
          (retract  ?x)
          (assert  (event  tacan  nominal  alt
                   "No " "toggle capability at this landing site")))

;----------------------------------------------------------------------------
```

3.14 Output Management

```
;;*****************************************************************************
;;
;;;  GROUP  Output Management
;;
;;      These groups determine what needs to be displayed and how it is
;;      to be displayed.
;;
;;
;;;  CONTROL FACTS
;       (phase output)
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;*****************************************************************************

;;*****************************************************************************
;;
;;;  GROUP Event Management
;;
;;      This group manages the transmission of event notices to the message
;;      windows.  An event notice is received as a fact with the following
;;      form:
;;              (event ?subsystem ?mode ?tag $?text)
;;      where   ?subsystem = the name of the subsystem generating the event
;;              ?mode = nominal or off-nominal
;;              ?tag = alt, mach, or none
;;              $?text = the text of the message
;;
;;;  CONTROL FACTS
;       (phase output)
;;
;;;  CONTAINING GROUP
;;      Output Management
;;
;;*****************************************************************************

(defrule output-event

;;      IF
;;              An event needs to be printed
;;      THEN
;;              Print it on the main message window and the appropriate
;;                      subsystem window
;;      END

        ?x <- (event  ?subsystem  ?mode  ?tag  $?text)
        (phase  output)
        =>
        (bind ?n 1)
        (bind ?l (length $?text))
        (while (<= ?n ?l)
            (bind ?a (nth ?n $?text))
            (if  (numberp ?a)
                then
                    (call  (format message "%g" ?a))
                else
                    (call  (format message "%s" ?a)))
            (bind ?n (+ ?n 1)))
        (call  (format message "%n"))
        (call  (message  main  ?mode  event  ?tag))
        (call  (message  ?subsystem  ?mode  event  ?tag))
        (retract  ?x))


;;*****************************************************************************
;;
;;;  GROUP  Recommendation Management
;;
;;  This group of rules handles the printout of recommendations at regular
;;  intervals.  Recommendations are sent to this group from other rules
;;  in the form of a fact:
;;
;;      (recommend ?subsystem ?id ?mode ?tag $?text)
;;
;;  where       ?subsystem = the name of the subsystem generating the event
;;              ?id = name of the recommendation (to distinguish it from other
;;                      recommendations).
;;              ?mode = nominal or off-nominal
;;              ?tag = alt, mach, or none
;;              $?text = the text of the message
;;
;;  The recommendation rules also keep an internal record of active
;;  recommendations using facts of the following form:
;;
;;      (active-message ?subsystem ?id ?a ?b ?time $?text)
;;
;;  where       ?subsystem = same as recommendation subsystem
;;              ?id = same as recommendation id
;;              ?a = message number on main message window
;;              ?b = message number on subsystem message window
;;              ?time = time the recommendation was last checked
;;              $?text = the text of the message
;;
;;  For a recommendation to remain active, the rule that asserts it must
;;  re-assert it on every cycle.  If a recommendation is not asserted on
;;  a given cycle, then it is assumed to no longer be active.
;;
;;;  CONTROL FACTS
;       (phase output)
;;
;;;  CONTAINING GROUP
;;      Output Management
;;
;;*****************************************************************************

(defrule  output-recommendation
        ?x <- (recommend  ?subsystem  ?id  ?mode  ?tag  $?text)
        (not  (active-message  ?subsystem  ?id  ?  ?  ?  $?text))
        (current-time  ?time)
        (phase  output)
        =>
        (bind ?n 1)
        (bind ?l (length $?text))
        (while (<= ?n ?l)
            (bind ?a (nth ?n $?text))
            (if  (numberp ?a)
                then
                    (call  (format message "%g" ?a))
                else
                    (call  (format message "%s" ?a)))
            (bind ?n (+ ?n 1)))
        (call  (format message "%n"))
        (bind  ?a  (message  main  ?mode  recommend  ?tag))
        (bind  ?b  (message  ?subsystem  ?mode  recommend  ?tag))
        (retract  ?x)
        (assert  (active-message  ?subsystem  ?id  ?a  ?b  ?time  $?text)))

;---------------------------------------------------------------------------

(defrule  output-hold-recommendation
        ?x <- (active-message  ?subsystem  ?id  ?a  ?b  ?last-time  $?text)
        ?y <- (recommend  ?subsystem  ?id  ?  ?  $?text)
        (current-time  ?time)
        (test  (> ?time ?last-time))
        (phase  output)
        =>
        (retract  ?x)
        (retract  ?y)
        (assert  (active-message  ?subsystem  ?id  ?a  ?b  ?time  $?text)))

;---------------------------------------------------------------------------

(defrule  output-end-recommendation
        ?x <- (active-message  ?subsystem  ?id  ?a  ?b  ?last-time  $?text)
        (not  (recommend  ?subsystem  ?id  ?  ?  $?text))
        (current-time  ?time)
        (test  (> ?time ?last-time))
        (phase  output)
        =>
        (call  (erase-msg  ?a))
        (call  (erase-msg  ?b))
        (retract  ?x))



;;****************************************************************************
;;
;;;  GROUP  Status Light Management
;;
;;      These rules control updates to the status lights.  Statuses are
;;      determined by other rules and are sent to this group as facts:
;;
;;              (status-light  ?id  ?sub-id  ?value)
;;
;;      where ?id is a subsystem identifier, ?sub-id is an LRU number or
;;      component identifier, and ?value is the value to be displayed.
;;
;;;  CONTROL FACTS
;       (phase output)
;;
;;;  CONTAINING GROUP
;;      Output Management
;;
;;****************************************************************************

(deffacts output-light-locations                ; These facts define the locatio
                                        ;  (line and column number) for each
                                        ;  of the subsystems and LRUs
        (light-location runway pass 1 10)
        (light-location runway bfs 1 15)
        (light-location runway ground 1 20)
        (light-location tacan pass 2 10)
        (light-location tacan bfs 2 15)
        (light-location state pass 3 10)
        (light-location state bfs 3 15)
        (light-location state ground 3 20)
        (light-location three-state 1 6 10)
        (light-location three-state 2 6 15)
        (light-location three-state 3 6 20)
        (light-location pass-imu 1 7 10)
        (light-location pass-imu 2 7 15)
        (light-location pass-imu 3 7 20)
        (light-location bfs-imu 1 8 10)
        (light-location bfs-imu 2 8 15)
        (light-location bfs-imu 3 8 20)
        (light-location drag 0 9 10)
        (light-location tacr 1 10 10)
        (light-location tacr 2 10 15)
        (light-location tacr 3 10 20)
        (light-location tacb 1 11 10)
        (light-location tacb 2 11 15)
        (light-location tacb 3 11 20)
        (light-location tacb cone 11 0)
        (light-location baro 0 12 10)
        (light-location mlsr 1 13 10)
        (light-location mlsr 2 13 15)
        (light-location mlsr 3 13 20)
        (light-location mlsa 1 14 10)
        (light-location mlsa 2 14 15)
        (light-location mlsa 3 14 20)
        (light-location mlse 1 15 10)
        (light-location mlse 2 15 15)
        (light-location mlse 3 15 20)
        (light-location tlm 0 16 10)
)

;---------------------------------------------------------------------------


(deffacts output-display-values         ; These facts define the display values
                                        ;  for all of the possible values of
                                        ;  the status lights
        (display-value unknown  "    " normal)
        (display-value blank    "    " normal)
        (display-value none     "    " normal)
        (display-value go       " GO " normal)
        (display-value good     "GOOD" normal)
        (display-value high     "HIGH" normal)
        (display-value low      "LOW " normal)
        (display-value no-go    "NOGO" blink)
        (display-value bias     "BIAS" blink)
        (display-value resolver "RSLV" blink)
        (display-value drift    "DRFT" blink)
        (display-value velocity "VEL " blink)
        (display-value attitude "ATTD" blink)
        (display-value suspect  "SPCT" blink)
        (display-value timing   "TIME" blink)
        (display-value noise    "NOIS" blink)
        (display-value atmos     "ATMS" blink)
        (display-value mach      "MACH" blink)
        (display-value roll      "ROLL" blink)
        (display-value cone      "CONE" blink)
        (display-value commfault "COMF" inverse)
        (display-value fail      "FAIL" inverse)
        (display-value deselect "DSEL" inverse)
        (display-value off       "OFF " inverse)
        (display-value bad       "BAD " inverse)
        (display-value stopped  "STOP" inverse)
)

;------------------------------------------------------------------------------

(defrule output-update-status-light
        ?x <- (status-light  ?id  ?sub-id  ?value)
        (display-value  ?value  ?word  ?mode)
        (light-location  ?id  ?sub-id  ?row  ?column)
        (phase  output)
        =>
        (retract  ?x)
        (call (status-light ?row ?column ?mode ?word)))

;------------------------------------------------------------------------------
```

3.15 Data Tables

```
;;************************************************************************
;;
;;;  GROUP
;;      Data Tables (no reference number)
;;
;;
;;;  CONTROL FACTS
;;      None
;;
;;
;;;  CONTAINING GROUP
;;      Entry
;;
;;
;;************************************************************************

;  Common-lru is used to determine the lru that is common to two pairs
;       (common-lru ?pair-1 ?pair-2 ?lru-id)
(deffacts tables-common-lru
        (common-lru  p-1-2  p-1-3  1)
        (common-lru  p-1-3  p-1-2  1)
        (common-lru  p-2-3  p-1-2  2)
        (common-lru  p-1-2  p-2-3  2)
        (common-lru  p-1-3  p-2-3  3)
        (common-lru  p-2-3  p-1-3  3)
)

;-------------------------------------------------------------------------

;  Excluded-lru is used to determine which lru is excluded from a pair
;       (excluded-lru ?pair ?lru-id)
(deffacts tables-excluded-lru
        (excluded-lru  p-1-2  3)
        (excluded-lru  p-1-3  2)
        (excluded-lru  p-2-3  1)
)


;-------------------------------------------------------------------------

;  Lrus-in-pair is used to determine which lrus are included in a pair
;       (lrus-in-pair ?pair ?lru-a ?lru-b)
;  Note that if ?pair is the only bound variable, then there are two matches.
(deffacts tables-lrus-in-pair
        (lrus-in-pair  p-1-2  1  2)
        (lrus-in-pair  p-1-2  2  1)
        (lrus-in-pair  p-1-3  1  3)
        (lrus-in-pair  p-1-3  3  1)
        (lrus-in-pair  p-2-3  2  3)
        (lrus-in-pair  p-2-3  3  2)
)


;-------------------------------------------------------------------------

;  Min-miscompare is used to determine the "smaller" of two miscomparison
;  ratings, where the ratings are defined to be "zero", "under", "o50",
;  and "over", in that order.
;       (min-miscompare ?status-1 ?status-2 ?min-status)
(deffacts tables-min-miscompare
        (min-miscompare  zero   zero   zero )
        (min-miscompare  under  zero   zero )
        (min-miscompare  o50    zero   zero )
        (min-miscompare  over   zero   zero )
        (min-miscompare  zero   under  zero )
        (min-miscompare  under  under  under)
        (min-miscompare  o50    under  under)
        (min-miscompare  over   under  under)
        (min-miscompare  zero   o50    zero )
        (min-miscompare  under  o50    under)
        (min-miscompare  o50    o50    o50  )
        (min-miscompare  over   o50    o50  )
        (min-miscompare  zero   over   zero )
        (min-miscompare  under  over   under)
        (min-miscompare  o50    over   o50  )
        (min-miscompare  over   over   over )
)


;----------------------------------------------------------------------------

;  Max-miscompare is used to determine the "larger" of two miscomparison
;  ratings, where the ratings are defined to be "zero", "under", "o50",
;  and "over", in that order.
;       (max-miscompare ?status-1 ?status-2 ?max-status)
(deffacts tables-max-miscompare
        (max-miscompare  zero   zero   zero )
        (max-miscompare  under  zero   under)
        (max-miscompare  o50    zero   o50  )
        (max-miscompare  over   zero   over )
        (max-miscompare  zero   under  under)
        (max-miscompare  under  under  under)
        (max-miscompare  o50    under  o50  )
        (max-miscompare  over   under  over )
        (max-miscompare  zero   o50    o50  )
        (max-miscompare  under  o50    o50  )
        (max-miscompare  o50    o50    o50  )
        (max-miscompare  over   o50    over )
        (max-miscompare  zero   over   over )
        (max-miscompare  under  over   over )
        (max-miscompare  o50    over   over )
        (max-miscompare  over   over   over )
)


;----------------------------------------------------------------------------

;  Fault matrix is used to determine the IMU component that has failed
;  based on which algorithms (velocity, attitude, or ACC) are indicating
;  a miscomparison with other IMUs.
;       (fault-matrix ?vel-status ?att-status ?acc-status ?fault)
;  where each status is under, o50, or over; and ?fault is as follows:
;       good            - no fault
;       bias            - accelerometer bias or scale factor error
;       resolver        - resolver error
;       drift           - gyro drift
;       velocity        - undiagnosable velocity problem
;       attitude        - undiagnosable attitude problem
;       suspect         - undiagnosable problem
(deffacts tables-fault-matrix
        (fault-matrix  under   under   under   good     )
        (fault-matrix  o50     under   under   velocity)
        (fault-matrix  over    under   under   velocity)
        (fault-matrix  under   o50     under   attitude)
        (fault-matrix  under   over    under   attitude)
        (fault-matrix  under   under   o50     attitude)
        (fault-matrix  under   under   over    attitude)
        (fault-matrix  o50     o50     under   resolver)
        (fault-matrix  over    o50     under   resolver)
        (fault-matrix  o50     over    under   resolver)
        (fault-matrix  over    over    under   resolver)
        (fault-matrix  o50     under   o50     bias     )
        (fault-matrix  over    under   o50     bias     )
        (fault-matrix  o50     under   over    bias     )
        (fault-matrix  over    under   over    bias     )
        (fault-matrix  under   o50     o50     drift    )
        (fault-matrix  under   over    o50     drift    )
        (fault-matrix  under   o50     over    drift    )
        (fault-matrix  under   over    over    drift    )
        (fault-matrix  o50     o50     o50     suspect )
        (fault-matrix  over    o50     o50     suspect )
        (fault-matrix  o50     over    o50     suspect )
        (fault-matrix  over    over    o50     suspect )
        (fault-matrix  o50     o50     over    suspect )
        (fault-matrix  over    o50     over    suspect )
        (fault-matrix  o50     over    over    suspect )
        (fault-matrix  over    over    over    suspect )
)


;--------------------------------------------------------------------------

;  quality-table is used to determine the quality of a state
;    vector (good ,suspect, or bad) based on a comparison with
;    another state vector or the ground (zero, under, o50, or over)
(deffacts tables-quality-table
        (quality-table  zero    good)
        (quality-table  under   good)
        (quality-table  o50     suspect)
        (quality-table  over    bad )
)


;--------------------------------------------------------------------------

;  tacan-quality is used to determine the quality of a tacan lru based on
;  comparisons with the ground or other lrus.
;       (tacan-quality  ?slope  ?bias  ?noise  ?quality)
;  where ?slope and ?noise are under or over; ?bias is under, o50, or over;
;  and quality is good, bias, timing, or noise.
(deffacts tables-tacan-quality
        (tacan-quality  under   under   under   good)
        (tacan-quality  under   under   over    noise)
        (tacan-quality  under   o50     under   bias)
        (tacan-quality  under   o50     over    noise)
        (tacan-quality  under   over    under   bias)
        (tacan-quality  under   over    over    noise)
        (tacan-quality  over    under   under   timing)
        (tacan-quality  over    under   over    noise)
        (tacan-quality  over    o50     under   timing)
        (tacan-quality  over   o50    over   noise)
        (tacan-quality  over   over   under  timing)
        (tacan-quality  over   over   over   noise)
)


;-------------------------------------------------------------------------

;  msbls-quality is used to determine the quality of a msbls lru based on
;  comparisons with the ground or other lrus.
;       (msbls-quality  ?bias  ?noise  ?quality)
;  where ?bias and ?noise are under, o50, or over; and quality is good or bad
(deffacts tables-msbls-quality
        (msbls-quality  under  under  good)
        (msbls-quality  under  o50    good)
        (msbls-quality  under  over   bad )
        (msbls-quality  o50    under  good)
        (msbls-quality  o50    o50    good)
        (msbls-quality  o50    over   bad )
        (msbls-quality  over   under  bad )
        (msbls-quality  over   o50    bad )
        (msbls-quality  over   over   bad )
)


;-------------------------------------------------------------------------

;  measurement-name is used to connect the 4-character measurement name used by
;  filter flags and data good flags with the TACAN and MSBLS measurement type
(deffacts tables-measurement-names
        (measurement-name  tacr  range)
        (measurement-name  tacb  bearing)
        (measurement-name  mlsr  range)
        (measurement-name  mlsa  azimuth)
        (measurement-name  mlse  elevation)
)



;-------------------------------------------------------------------------

;  "units" is used to determine the unit name to print out for a given
;  measurement
(deffacts tables-units
        (units  range   feet)
        (units  bearing degrees)
        (units  azimuth degrees)
        (units  elevation degrees)
        (units  drag    feet)
        (units  tacr    feet)
        (units  baro    feet)
        (units  mlsr    feet)
        (units  tacb    degrees)
        (units  mlsa    degrees)
        (units  mlse    degrees)
)


;-------------------------------------------------------------------------
```

```
;;  same-area is used to determine which slot is in the same area as a
;;  given slot
(deffacts tables-same-area
        (same-area   1   2)
        (same-area   2   1)
        (same-area   3   4)
        (same-area   4   3)
        (same-area   5   6)
        (same-area   6   5)
        (same-area   7   8)
        (same-area   8   7)
        (same-area   9  10)
        (same-area  10  19)
        (same-area  11  12)
        (same-area  12  11)
        (same-area  13  14)
        (same-area  14  13)
        (same-area  15  16)
        (same-area  16  15)
        (same-area  17  18)
        (same-area  18  17)
        (same-area  19  20)
        (same-area  20  19)
        (same-area  21  22)
        (same-area  22  21)
        (same-area  23  24)
        (same-area  24  23)
        (same-area  25  26)
        (same-area  26  25)
        (same-area  27  28)
        (same-area  28  27)
        (same-area  29  30)
        (same-area  30  29)
)

;----------------------------------------------------------------------------
```

Section 4

REFERENCES

1) "Knowledge Requirements For the Onboard Navigation (ONAV) Console Expert/Trainer System, "Mission Support Directorate, Mission Planning & Analysis Division, NASA Johnson Space Center, ENTRY phase specifications, Baseline Version 1.0, October 1987, JSC internal Note #JSC-22657.

End of Document