NASA Contractor Report 4643

SSME HPOTP Post-Test Diagnostic System Enhancement Project

Timothy W. Bickmore
Aerojet Tech Systems
Sacramento, California

Prepared for
Lewis Research Center
under Contract NAS3–25883

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program
1995

Contents

I. Introduction

An assessment of engine and component health is routinely made after each test or flight firing of a Space Shuttle Main Engine (SSME). Currently, this health assessment is done by teams of engineers who manually review sensor data, performance data, and engine and component operating histories. Based on review of information from these various sources, an evaluation is made as to the health of each component of the SSME and the preparedness of the engine for another test or flight.

The objective of this project is to further development of a computer program which automates the analysis of test data from the SSME High-Pressure Oxidizer Turbopump (HPOTP) in order to detect and diagnose anomalies. This program fits into a larger system—the SSME Post-Test Diagnostic System (PTDS)—which will eventually be extended to assess the health and status of most SSME components on the basis of test data analysis.

The HPOTP module is an expert system, which uses "rules-of-thumb" obtained from interviews with experts from NASA Marshall Space Flight Center (MSFC) to detect and diagnose anomalies. Analyses of the raw test data are first performed using pattern recognition techniques which result in features such as spikes, shifts, peaks, and drifts being detected and written to a database. The HPOTP module then looks for combinations of these features which are indicative of known anomalies, using the rules gathered from the turbomachinery experts. Results of this analysis are then displayed via a graphical user interface which provides ranked lists of anomalies and observations by engine component, along with supporting data plots for each.

I.1. The SSME Post-Test Diagnostic System Project

The post-test diagnostic system is a cooperative effort involving engineers and scientists at NASA Marshall Space Flight Center (MSFC), NASA Lewis Research Center (LeRC), Aerojet, and Science Applications International Corporation (SAIC). The system is designed to be a generic approach to automating the rocket engine data review process. A modular, distributed architecture was selected which enables modules which analyze different aspects of an engine's performance. The PTDS modules currently implemented or being developed include the following (see Figure 1):

- CAE Package — The Computer Aided Engineering package is used primarily to provide a very flexible mechanism for displaying plots of engine data. The PV~Wave command language was selected as a commercial off-the-shelf (COTS) package to fill this need.
- Relational Database Management System — A database is used to store information about tests, engines configurations, anomalies, performance parameter histories, and all PTDS analysis results. Ingres was initially selected as the COTS package to be used, but was changed to TekBase in 1993.
- Session Manager — The executive for the system which launches each of the modules as needed once test data becomes available. Implemented in C.
- Feature Extractor — Performs the pattern recognition analyses on the raw data. Implemented in C.
- HPOTP Analysis Module — Analyzes the health and performance of the SSME HPOTP. Initially implemented in Nexpert Object and C; converted to CLIPS in 1993.
- Systems Analysis Module — Analyzes the system-wide health and performance of the engine, and detects anomalies which involve more than one component. Implemented in CLIPS and scheduled for completion in 1994.

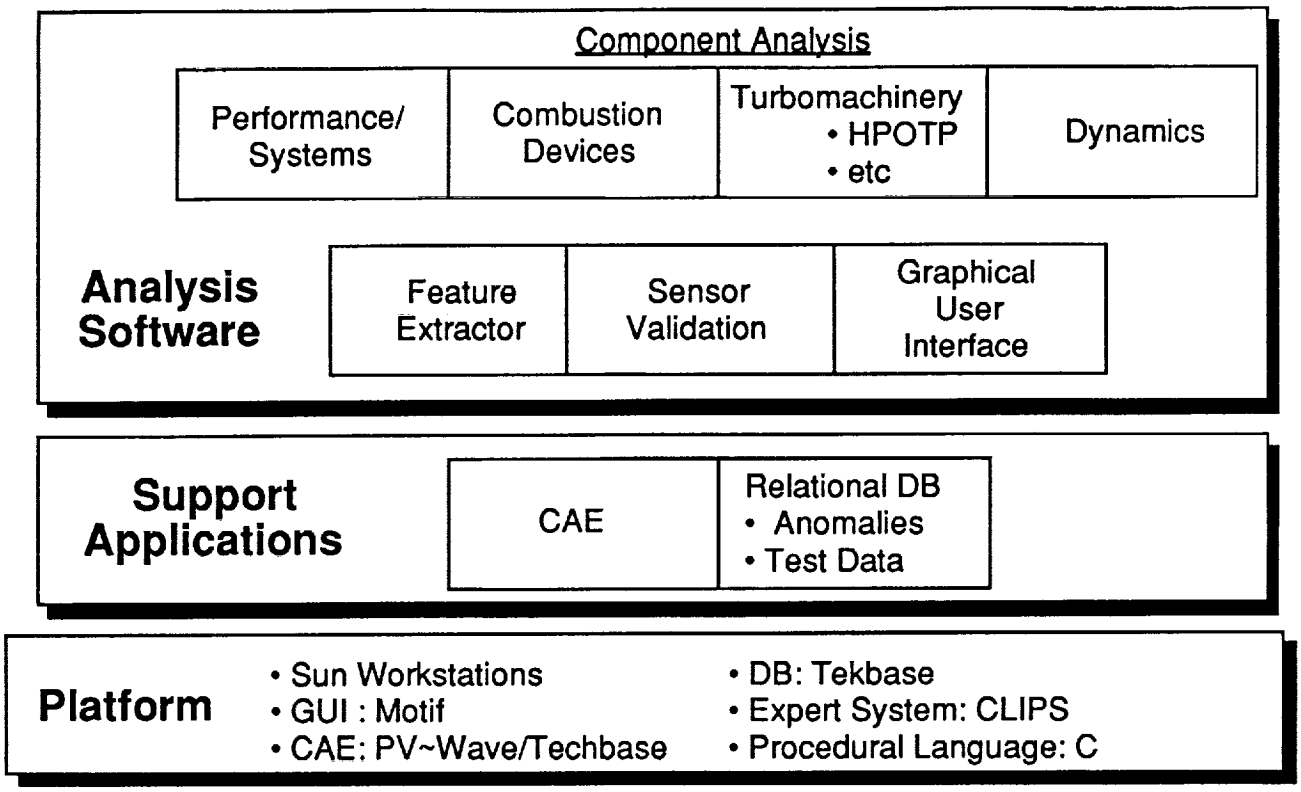

Figure 1. PTDS Architecture

In addition, modules for analysis of the other three SSME pumps, combustion devices, and dynamics data, are expected to be added to the PTDS.

I.2. The HPOTP Diagnostic System Enhancement Project

A first version of the HPOTP module was developed using the Nexpert Object expert system shell and interfaced with an Ingres relational database. In 1993 the decision was made to replace Nexpert Object with CLIPS, and replace Ingres with TekBase. CLIPS—the C Language Integrated Production System, developed at NASA Johnson—provided a more flexible language for complex knowledge representation than Nexpert. TekBase was in wide use by SSME data analysts at NASA MSFC, and the decision to port to it from Ingres was made to ensure that the PTDS would readily integrate into MSFC's operations.

These changes necessitated a substantial re-implementation of the HPOTP module. The HPOTP Diagnostic System Enhancement Project's goals were thus to perform this re-implementation, and then increase the accuracy of the system on anomalies it currently checked for, and extend it to check for additional anomalies suggested by MSFC's turbomachinery experts.

I.3. Results Obtained

Conversion

The initial re-implementation of HPOTP module from Nexpert and C to CLIPS reduced the code complexity (measured in lines of code) by over 90%. This was achieved through the greater flexibility allowed in CLIPS (allowing multiple Nexpert rules to be encoded as one CLIPS rule), and through the use of language elements which provided a "shorthand" notation for describing anomalies and their supporting plots. For example, the rule shown in Figure 2 detects significant discrepancies between the balance piston pressure difference on the current test versus a comparison test. Statement 6 (the action the rule will take when it "fires") defines the anomaly and all information required for output. Statements 7-9 define the three supporting plots required for this class of anomaly. This same rule takes up approximately two pages of Nexpert code in the original implementation.

```
   (defrule anomaly5.05.1
1.   (current_phase find_anomalies)
2.   (current_test ?testid)
3.   (comparison_test ?comptestid)
4.   (F_THLEDE ?testid ?start ?end ?PL)
5.   (F_DIFTHA ?testid "327 - 328"
            ?start2 ?end2&:(is_concurrent ?start ?end ?start2 ?end2)
               ?comptestid "327 - 328" $?)
  =>
6.   (assert (anomaly
              (class A5.05.1)
              (start ?start2)
              (end   ?end2)
              (priority 16)
              (description =(str-cat
                "The difference (327 - 328) is different at thrust level " ?PL
                " between this test and the previous."))))))

  (deffacts init5.05.1
7.  (plot (class A5.05.1)(number 1)(shutdown_delta_end 100.0)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
8.  (plot (class A5.05.1)(number 2)(use_comparison TRUE)(shutdown_delta_end 100.0)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
9.  (plot (class A5.05.1)(number 3)(cross_comparison TRUE)
          (shutdown_delta_end 100.0)(PIDs "63")(title "Thrust Profile")))
```

Figure 2. Example Diagnostic Rule

Even after the HPOTP system was extended and enhanced, it was significantly smaller than the original implementation, shrinking from 16,031 lines of Nexpert and 5,122 lines of C to 2,734 lines of CLIPS. This reduction in complexity greatly increases the maintainability of the system.

Enhancements

Many enhancements were made to the original HPOTP diagnostic system, per the recommendations of Glenn Wilmer, MSFC's turbomachinery expert consulted for this project (transcripts of all interviews with him are given in Attachment #2). The major enhancements to the system were the following:

- Embedded Feature Extraction — All of the feature extraction routines required by the HPOTP module were integrated into the CLIPS expert system shell, so that the HPOTP module can be run interactively as a stand-alone system. The primary benefit of this

enhancement is that changes to the HPOTP module can be made using a rapid-prototyping methodology, since the embedded feature extraction routines execute in seconds, whereas the PTDS feature extraction module currently takes hours to run. Attachment #5 gives a description of all of these extensions.

- Sensor Validation and Redundancy Management — Before the HPOTP module can perform anomaly detection it must first determine if the sensor data it is analyzing is valid. The original approach to sensor validation involved a simple voting scheme in which one sensor from each set of redundants was selected based on the number of "unusual" features it produced (e.g., spikes, noise, etc.). The new approach first attempts to disqualify sensors based on detection of "hard" failures (via reasonableness limits, non-variation during the firing, excessive noise, and majority voting of redundants), but then uses all remaining redundant sensors to vote on detected features. For example, nose seal leakage is indicated by a small spike in intermediate seal discharge pressure (see Figure 3). In the old approach, one of the two sensors (211 or 212) would have been selected semi-arbitrarily, and then checked to see if any spikes were indicated. In the new approach, both 211 and 212 are checked for spikes, and the results are cross-checked to see if both sensors agree. Anomalies are only reported if two or more redundant sensors agree on the indications. This has greatly reduced the false alarm rate of the system.

- Historical Statistics — The original version of the HPOTP module relied heavily on comparison of the pump-under-test to data from a prior test (ideally from the same pump or engine). Unfortunately, HPOTPs rarely go through the test program twice anymore; they typically are acceptance tested and then transferred into the flight program. At the suggestion of Glenn Wilmer, most cross-test comparisons were replaced with comparison of selected HPOTP parameters to statistical averages of many prior tests.

- Preburner Pump Bistability — The original feature extraction routine for bistability was replaced with a new one based on the SSME Green Run specification. The new test checks for 3-standard-deviation exceedances of the preburner pump head ratio (per the Green Run specification):

$$\frac{\text{PBP DS P - HPOTP DS P}}{\text{HPOTP DS P - HPOTP IN P}}$$

  but then also checks for a "system response" by ensuring that OPOV responds within the same time frame as the exceedance.

- Rotor Drag — New rules were added to detect "rotor drag", which is when the rotor hangs up following a vent or power level change and then moves slowly back into position. This is detected by first partitioning the firing up into periods of constant power level and LOX inlet pressure. The system then looks for one of these intervals following an increase in power level or decrease in LOX inlet pressure in which the balance piston pressure difference (327-328) is decreasing, 328 is increasing and 327 is decreasing. (The inverse case is also checked: 327-328 increasing, 328 decreasing, and 327 increasing following a decrease in power level or increase in LOX inlet pressure).

- Nose Seal Leakage — Rules were added and the spike feature extraction routine modified to detect "nose seal leakage" (see Figure 3).

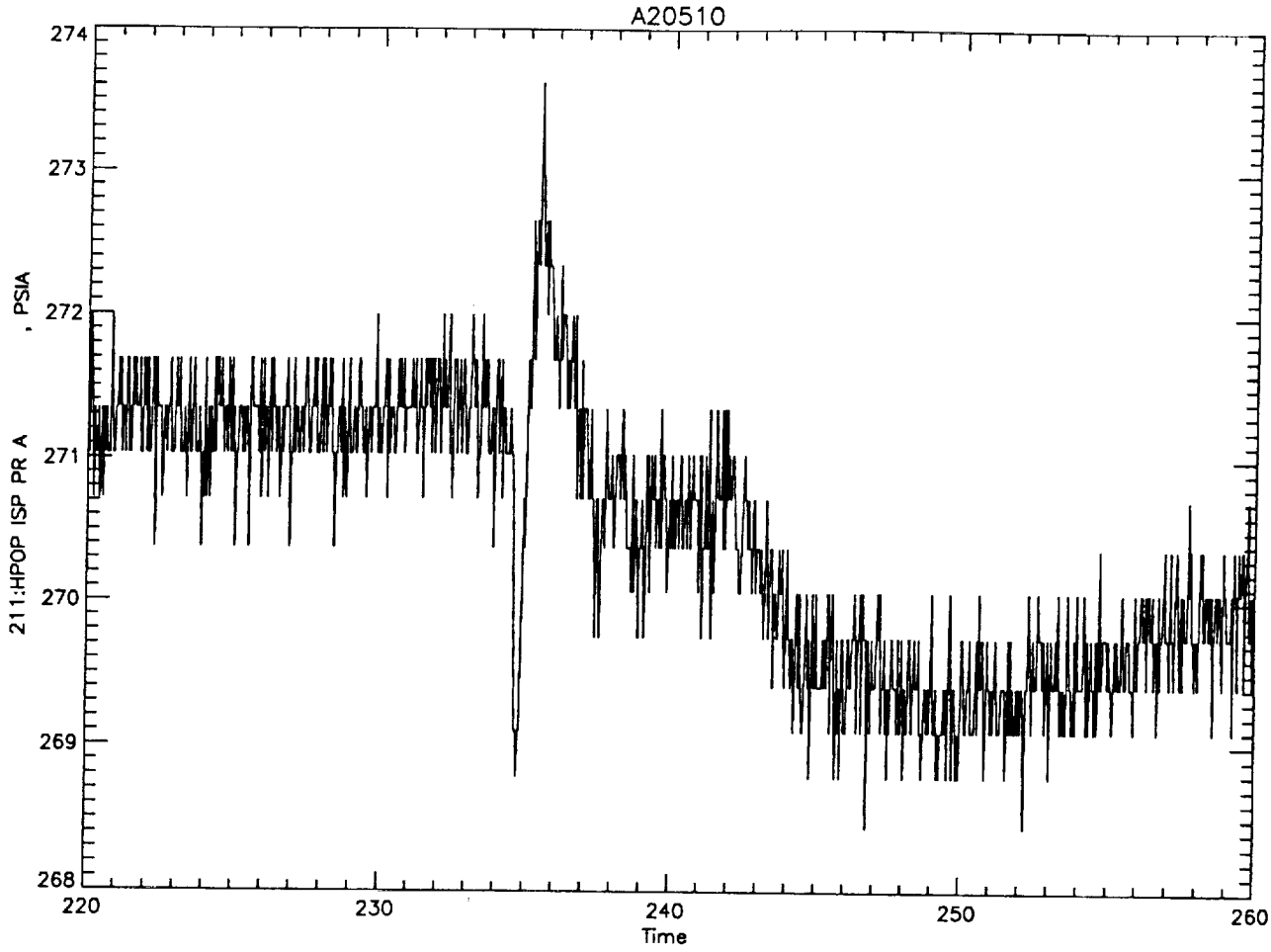

Figure 3. Example of Nose Seal Leakage

- Green Run Specifications — Rules were added to check all Green Run specifications, including:
  - ◊ Thrust profile requirements
  - ◊ LOX inlet pressure requirements
  - ◊ Turbine delta-T limits
  - ◊ Speed change limits
  - ◊ Parameter upper, lower, and difference limits

Performance

SSME data analysts at MSFC evaluated the performance of the original HPOTP module on 23 test firings in 1992. A suite of 24 test firings compiled by Glenn Wilmer was recently analyzed by the enhanced HPOTP module (Section IV of this report gives a detailed account of these results). The comparative results are shown in Table 1.

	Original	Enhanced
Number of Tests	23	24
Correct Observations	2 of 30	14 of 35
False Alarms	57	6
New Correct Observations	1	45

Table 1. Comparative Performance of HPOTP Diagnostic Modules

II. System Architecture

This section describes the overall architecture of the Enhanced HPOTP Diagnostic System. A data dictionary describing all global data structures (defglobals, deftemplates, and fact formats) is given in Attachment #6. Full listings for the CLIPS source code is given in Attachment #7.

The enhanced system is designed to run in one of two execution modes: interactive and batch. In interactive mode, the system queries a user for all needed information, computes all features dynamically as-needed, and outputs results in a textual form to the terminal. Batch mode is designed for use with the PTDS. In batch mode features are first computed by a separate module and stored in a database, then the HPOTP modules is started, reads the features in, performs its analyses and writes its results back out to the database. A user can then browse the system's results via a graphical user interface.

The major modules in the diagnostic system are each described in the following sections.

II.1. TKCLIPS

The HPOTP diagnostic system is implemented in a customized version of the CLIPS expert system shell, called "TKCLIPS" (for TeKbase CLIPS). Extensions were added to permit data to be read from and written to the TekBase database, to obtain information from the operating system, and to dynamically extract SSME test data and perform analyses.

A User's guide to TKCLIPS outlining all of the extensions is given in Attachment #5.

II.2. Executive

The executive module is primarily responsible for guiding execution of the diagnostic system through several major steps or "phases". These phases are:

initialize	Connect to database. Get test IDs and thrust profiles.
SVAL_hard_failures	Check for hard sensor failures.
SVAL_soft_failures	Determine preferred sensors via voting.
get_features	Determine features needed for diagnosis.
find_event_intervals	Determine time intervals to analyze.
find_anomalies	Diagnosis.
prepare_output	Prepare anomaly and supporting plot descriptions.
output_anomalies	Output results.
wrapup	Disconnect from database, cleanup.

II.3. Feature Extraction

This module obtains features requested by other parts of the HPOTP system. The requests (in the form of 'GetFeature' facts) are honored by either importing features from TekBase (in batch processing mode) or by computing them on-the-fly as needed (in interactive mode). Resulting features are stored as facts for use by redundancy management, sensor validation, and diagnostic routines.

All calls to import data from TekBase are located within this module.

II.4. Sensor Validation

The sensor validation module is responsible for detecting and diagnosing instrumentation anomalies and failures. It operates in two phases. First, it attempts to detect obvious, "hard" failures. These include:

- No data in the file.
- Exceeds gross noise limits.
- Pre- or Post-test reasonableness limit exceedance.
- Reasonableness limit exceedance during the test.
- Sensor trace is flat during the test (for sensor whose values are expected to vary significantly). This indicates that the sensor may have been disconnected.
- Redundancy voting, when three or more redundant transducers are available.

In its second phase of operation, the sensor validation module essentially replicates the voting scheme implemented in the original HPOTP diagnostic system implementation. This strategy entails selecting a "preferred" sensor from each set of redundants on the basis of the minimum number of erratic or spike features. These sensor preferences are not relied upon to the extent they were in the original implementation; wherever possible, features are redundancy voted (see the next section).

II.5. Redundancy Management

The redundancy management module performs redundancy management for a select group of features. Whenever one of these features is defined for a sensor (either imported from TekBase or computed dynamically), it is classified by checking it against all redundant sensors. The results of the classification can be one of the following:

For transducers with only one bridge each:

unconfirmed	If the sensor has no valid redundants.
spurious	If the sensor has valid redundants, none of which register a similar feature at the same time.
confirmed	If the sensor has a redundant which registers a similar feature at the same time. Any outlier redundants (which did not see the feature) are marked as spurious.

For transducers with two bridges:

confirmed	If seen on any bridge of 2 or more transducers. (Outlier bridges are marked as spurious.)
unconfirmed	Seen on all valid bridges of a transducer and there are no other valid transducers.
spurious	If more than one transducer is valid but only seen on one. (All bridges are marked as spurious.)

Feature equivalence is based on start times only (i.e., they must be within one second of each other), and the resulting 'confirmed' feature from two or more redundant features is formed by simply selecting one of the inputs (i.e., no attempt is made to average the values).

II.6. Statistics Module

The statistics module is responsible for computing statistical summaries of the parameters stored in the historical database, computing parameter values for the current test, and determining if any current test parameters are "outliers".

II.7. Anomaly Detection & Diagnosis

The Anomaly Detection module checks for combinations of features which are indicative of known anomalies. Results are asserted as *anomaly* records, which have descriptions of appropriate supporting plots associated with them. Results are classified as either INSTRUMENTATION, OBSERVATION, or ANOMALIES, and have a priority number associated with them indicating the degree of severity.

To simplify many of the anomaly detection rules, the test time-line is partitioned into intervals within which nothing is happening (i.e. no features start or end). The anomaly detection rules which are interested in concurrent combinations of features then simply analyze each of these time intervals separately. If the same anomaly is detected in adjacent intervals, the anomaly descriptions are combined into one spanning the entire interval.

A complete list of the diagnostic rules used in the system is given in Section III.

II.8. Green Run Specifications Check

The Green Run Specifications module checks all Green Run requirements and creates *anomaly* records whenever any violations are detected.

II.9. Supporting Plot Generation

The Plot Generation module takes abbreviated descriptions of supporting plots required for detected anomalies, and expands them into the 60 fields required by the PTDS to produce plots in the graphical user interface.

II.10. Output of Results

The Output module takes the results of the analyses, along with information about supporting plots, and either writes them to the database for later viewing (in batch mode) or prints a textual summary to the terminal (in interactive mode). This module also updates the historical database automatically (in batch mode) or if indicated by the user (in interactive mode).

All calls to export data to TekBase are located within this module.

III. Anomalies Currently Detected by the HPOTP Diagnostic System

This section provides a brief, but complete listing of the anomalies currently detected by the enhanced HPOTP diagnostic system.

III.1. General Anomalies

Rule: anomaly5.05.1
Source: SAIC final report, section 5.05
Summary: Difference 327-328 is different between current and comparison tests.
Report: "The difference (327 - 328) is different at thrust level <PL> between this test and the previous."

Rule: anomaly5.06.1
Source: SAIC final report, section 5.06
Summary: Spike seen in 327(328), not in 328(327), and no level shift in 327,328, or 327-328.
Report: "Spike seen in sensor <327|328> only, with no change in steady state pressures or pressure difference. Possible sensor or omni seal anomaly. No real rotor motion."

Rule: anomaly5.06.2
Source: SAIC final report, section 5.06
Summary: Level shift seen in 327(328) and not in 328(327).
Report: "Level shift seen in <327|328> only. Possible sensor problem, omni seal leakage problem. No real rotor motion.

Rule: anomaly5.06.3
Source: SAIC final report, section 5.06
Summary: Spike seen in 327 and 328, and level shift seen in 327-328.
Report: "Possible HPOTP momentary anomalous rotor motion.Possible HPOTP balance piston momentary shift in orifice position."

Rule: anomaly5.06.4
Source: SAIC final report, section 5.06
Summary: Level shift seen in 327 and 328 (opposite directions), and in 327-328.
Report: "Possible HPOTP anomalous rotor motion."

Rule: anomaly5.06.5
Source: SAIC final report, section 5.06
Summary: Level shift seen in 327 and 328 (same direction).
Report: "Possible HPOTP balance piston orifice position change."

Rule: anomaly5.06.7
Source: SAIC final report, section 5.06
Summary: Level shift seen in 327(328) and not in 328(327).
Report: "Statistically significant change in <327|328> but not in difference (327 - 328). Possible omni seal leakage. No real rotor motion."

Rule: anomaly5.06.9
Source: SAIC final report, section 5.06
Summary: Level shift seen in 327-328, but not in 327 or 328.
Report: "Statistically significant change in difference (327 - 328) but not in individual sensors. Not anomalous; no real rotor motion.")

Rule: anomalyRotorDrag1, anomalyRotorDrag1
Source: Wilmer
Summary:
1. Concurrent: Significant increase in 328, decrease in 327, LOX in P is flat
   Following increase in PL or decrease in LOX in P
   Duration of more than 10 seconds (or duration of current power level, if less).
2. Opposite of above case.
Report: "Possible rotor drag."

Rule: anomaly5.07.1
Source: SAIC final report, section 5.07
Summary: PBP bistability detection. Just reports result from C routine.
Report: "PBP bistability at thrust level <PL>"

Rule: anomaly5.08.1
Source: SAIC final report, section 5.08
Summary: Erratic 990, 1190 not erratic or spiking.
Report: "HPOTP erratic primary turbine seal drain pressure may indicate sensor problem or seal anomaly. No effect seen in drain temperature."

Rule: anomaly5.08.2
Source: SAIC final report, section 5.08
Summary: 1190 erratic, 990 not erratic or spiking.
Report: "HPOTP erratic primary turbine seal drain temperature may indicate sensor problem or seal anomaly. No effect seen in drain pressure."

Rule: anomaly5.08.3
Source: SAIC final report, section 5.08
Summary: 990 erratic or spiking, and 1190 erratic or spiking.
Report: "HPOTP shows concurrent jitter in both primary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly5.08.4
Source: Wilmer
Summary: Erratic or spiking 990 & 1190 in same power-level interval, but not concurrently.
Report: "HPOTP shows non-concurrent jitter in both primary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly990shift
Source: Wilmer
Summary: 990 is low (or high) in peak and equilibrium values relative to family.
Primary turbine seal drain pressure.
Report: "HPOTP primary turbine seal drain pressure is <HIGH|LOW> in peak and equilibrium values compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly990peakshift
Source: Wilmer
Summary: 990 peak value is out-of-family, but equilibrium value is OK.
Report: "HPOTP primary turbine seal drain pressure peak is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly990peakshift
Source: Wilmer
Summary: 990 equilibrium value is out-of-family, but peak value is OK.
Report: "HPOTP primary turbine seal drain pressure equilibrium value is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91shift
Source: Wilmer
Summary: 91 or 92 is low (or high) in peak and equilibrium values relative to family. Secondary turbine seal cavity pressure.
Report: "HPOTP secondary turbine seal cavity pressure is <HIGH|LOW> in peak and equilibrium values compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91peakshift
Source: Wilmer
Summary: 91/92 peak value is out-of-family, but equilibrium value is OK.
Report: "HPOTP secondary turbine seal cavity pressure peak is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91eqshift
Source: Wilmer
Summary: 91/92 equilibrium value is out-of-family, but peak value is OK.
Report: "HPOTP secondary turbine seal cavity pressure equilibrium value is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91peakwidth
Source: Wilmer
Summary: 91/92 peak width is out-of-family.
Report: "HPOTP secondary turbine seal cavity pressure peak width is <HIGH|LOW> compared to historical statistics. "

Rule: anomaly990peakwidth
Source: Wilmer
Summary: 990 peak width is out-of-family.
Report: "HPOTP primary turbine seal drain pressure peak width is <HIGH|LOW> compared to historical statistics. "

Rule: anomaly91peaktime
Source: Wilmer
Summary: 91/92 peak time is out-of-family.
Report: "HPOTP secondary turbine seal cavity pressure peak time is <HIGH|LOW> compared to historical statistics. "

Rule: anomaly990peaktime
Source: Wilmer
Summary: 990 peak time is out-of-family.
Report: "HPOTP primary turbine seal drain pressure peak time is <HIGH|LOW> compared to historical statistics. "

Rule: anomalyIMSLstart
Source: Wilmer
Summary: START value of 211/212 is out-of-family.
Report: "HPOTP intermediate seal purge pressure is <HIGH|LOW> at START compared to historical statistics. "

Rule: anomalyBalPistonFamily1
Source: Wilmer
Summary: 327 is out-of-family (at 109MAX, 104MIN, or 104Nominal NPSP).
Report: "HPOTP balance cavity pressure A is <HIGH|LOW> at <time> compared to historical statistics. B channel is within limits."

Rule: anomalyBalPistonFamily2
Source: Wilmer
Summary: 328 is out-of-family (at 109MAX, 104MIN, or 104Nominal NPSP).
Report: "HPOTP balance cavity pressure B is <HIGH|LOW> at <time> compared to historical statistics. A channel is within limits."

Rule: anomalyBalPistonFamily3
Source: Wilmer
Summary: 327 and 328 are both out-of-family (at 104MIN, 109Max, or 104Nominal NPSP).
Report: "HPOTP balance cavity pressure A is <HIGH|LOW> and channel B is <HIGH|LOW> at <time> compared to historical statistics."

Rule: anomalyLOXSlPFamily
Source: Wilmer
Summary: 951|952|953 are out-of-family.
Report: "HPOTP primary pump seal drain pressure is <HIGH|LOW> from 5 seconds to cutoff compared to historical statistics."

Rule: anomalyLOXSlTFamily
Source: Wilmer
Summary: 1187 is out-of-family.
Report: "HPOTP primary pump seal drain temperature maximum is <HIGH|LOW> compared to historical statistics.")

Rule: anomalySlingerProblem
Source: Wilmer
Summary: 1187 is out-of-family low and 951/952/953 is out-of-family high.
Report: "HPOTP primary pump seal drain temperature maximum is LOW and HPOTP primary pump seal drain pressure is HIGH compared to historical statistics. Indicates possible slinger problem.")

Rule: anomaly5.09.6
Source: SAIC final report, section 5.09
Summary: Could not compute a peak for 990.
Report: "Current test HPOTP primary turbine seal drain pressure peak missing."

Rule: anomaly5.09.12
Source: SAIC final report, section 5.09
Summary: Could not compute a peak for 91/92.
Report: "Current test HPOTP secondary turbine seal cavity pressure peak missing."

Rule: anomaly5.12.1
Source: SAIC final report, section 5.12
Summary: 91/92 is erratic, 1188 is normal.
Report: "HPOTP erratic secondary turbine seal drain pressure may indicate sensor problem or seal anomaly No effect seen in drain temperature."

Rule: anomaly5.12.2
Source: SAIC final report, section 5.12
Summary: 1188 is erratic, 91/92 is normal.
Report: "HPOTP erratic secondary turbine seal drain temperature may indicate sensor problem or seal anomaly. No effect seen in drain pressure."

Rule: anomaly5.12.3
Source: SAIC final report, section 5.12
Summary: 91/92 and 1188 are both erratic or spiking. Check for concurrent anomalies.
Report: "HPOTP shows concurrent jitter in both secondary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly5.12.4
Source: Wilmer
Summary: 91/92 and 1188 are both erratic or spiking. Check for non-concurrent anomalies.
Report: "HPOTP shows non-concurrent jitter in both secondary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly5.15.1
Source: SAIC final report, section 5.15
Summary: 951/952/953 is erratic, 1187 is normal.
Report: "HPOTP erratic primary pump seal drain pressure may indicate sensor problem or seal anomaly. No effect seen in drain temperature."

Rule: anomaly5.15.2
Source: SAIC final report, section 5.15
Summary: 1187 is erratic, 951/952/953 are normal.
Report: "HPOTP erratic primary pump seal drain temperature may indicate sensor problem or seal anomaly. No effect seen in drain pressure."

Rule: anomaly5.15.3
Source: SAIC final report, section 5.15
Summary: 951/952/953 and 1187 are both erratic or spiking. Concurrent anomaly in both sensors.
"HPOTP shows concurrent jitter in both primary pump seal drain pressure "
"and temperature. Possible seal anomaly.")

Rule: anomaly5.15.4
Source: Wilmer
Summary: 951/952/953 and 1187 are both erratic or spiking. Non-concurrent anomaly in both sensors.
Report: "HPOTP shows non-concurrent jitter in both primary pump seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly5.18.1
Source: SAIC final report, section 5.18
Summary: 211/212 are erratic or spiking.
Report: "Intermediate seal purge pressure appears erratic or spiking. Possible nose seal leakage, helium supply problem or sensor anomaly. Slight possibility of rubbing."

Rule: anomaly5.19.3
Source: Inferred from Priority table in SAIC's POST_defs.h file.
Summary: 233/234 are erratic or spiking.
Report: "Spike or erratic behavior in HPOT discharge temperature (confirmed by two sensors)."

Rule: ADeltaPfamily
Source: Inferred from examples from Wilmer.
Summary: 327-328 statistics (for any time interval) are greater than 2.5 sigma.
Report: "HPOTP balance cavity pressure delta-P is out-of-family <HIGH|LOW> during <time> conditions, compared to historical statistics. "

III.2. Green Run Specifications

Rule: GREEN_check_duration
Source: Green Run Specs, RL00461, 6 Jan 1988
Summary: The following rules determine the total time spent at 104% and 109%, and check them against the Green Run specs.
Report: "Failed HPOTP Green Run test duration criteria 3.5.1.2(a)."

Rule: GREEN_check_LPOTP_inlet
Source: Green Run Specs, RL00461, 6 Jan 1988
Summary: The following rules check the minimum required duration at MIN and MAX LOX pressurization.
Report: 1. "Failed HPOTP Green Run LPOTP inlet criteria 3.5.1.2(b). "
"(Minimum NPSP of 20+5/-0 for 5 seconds at 104% or higher.)"
2. "Failed HPOTP Green Run LPOTP inlet criteria 3.5.1.2(c). "
"(Maximum NPSP of 150+10/-0 for 10 seconds at 104% or higher.)")

Rule: GREEN_check_65_time
Source: Green Run Specs, RL00461, 6 Jan 1988
Summary: Checks the minimum bucket durations.
Report: "Failed HPOTP Green Run 65/64/63% throttle criteria 3.5.1.2(d)"

Rule: GREEN_check_limits
Source: Green Run Specs, RL00461, 6 Jan 1988
Summary: 3513II & III, Checks the various hard Green Run limits
Report: 1. "Failed HPOTP Green Run limits at START for <parameter>."
2. "Failed HPOTP Green Run <PL>% peak limits for <parameter>."
3. "Failed HPOTP Green Run <VentCondition> limits for <parameter>."

Rule: GREEN_check_IMSLStart
Source: Wilmer
Summary: Normalized form. Checks the intermediate seal purge pressure requirement at start.
Report: "Failed HPOTP Green Run intermediate seal purge pressure START criteria."

Rule: GREEN_check_DeltaT
Source: Green Run Specs, RL00461, 6 Jan 1988, Wilmer
Summary: 3513II & III, Checks the minimum turbine delta-T requirements. If limit is exceeded and turbine temps are cold, then that is offered as an explanation.
Report: 1. "Failed HPOTP Green Run <PL>% limits for turbine Delta-T. Probable cause is
cold turbine temperature (below 1300.0)."
2. "Failed HPOTP Green Run <PL>% limits for turbine Delta-T. Turbine temperature is not cold."

Rule: GREEN_check_DeltaSpeed
Source: Green Run Specs, RL00461, 6 Jan 1988
Summary: 3513II & III, Checks delta-speed requirements.
Report: "Failed HPOTP Green Run <PL>% limits for speed change."

IV. Test Results

The performance of the enhanced HPOTP diagnostic system is significantly better than the original implementation. Table 2 outlines the accuracy of the system on a suite of 24 tests selected by Glenn Wilmer. The observations he originally noted are checked in the "Expert Analyst" column. The observations made by the system are checked in the "Diagnostic System" column, and false alarms (improper diagnoses) are also noted.

Test	Anomaly	Expert Analyst	Diagnostic System
A20510	• Rotor hang up	√	√
	• IMSL nose seal leak (211/212 @ 235s)	√	√
	•Turbine press step (990 @ 150s)	√	
	• LPOTP inlet Green Run fail (Max NPSP)		√
A20547	• Bi-stability	√	√
	• Primary turbine seal delta-T (early 104%, and at 109%)	√	√
	• Nose seal leakage	√	√
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max NPSP)		√
A20548	• 2nd Turb Sl Pressures drift apart	√	
	• Nose seal leakage	√	√
	• LPOTP inlet Green Run fail (Max NPSP)		√
A20549	• Bad Bal Cav Pr CH A (may be omni seal or sensor, @75-150s)	√	
	• Drifting 2nd turb seal cav press	√	
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max NPSP)		FalseAlarm
A20551	• Slight rotor drag	√	√
	• Nose seal leakage	√	
	• LPOTP inlet Green Run fail (Max NPSP)		√
A20552	• Pri Tr Sl Delta-T Green Run fail	√	√
	• IMSL press shifts (nose seal leakage)	√	
	• LPOTP inlet Green Run fail (Max NPSP)		√
A20555	• Bal cav press shifts in cavity 2 (328)	√	
	• Nose seal leakage	√	
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max & Min)		√
	• 65/64/63% Green Run fail		√
A20558	• Bi-stability	√	√
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max & Min)		√
	• 65/64/63% Green Run fail		√
A20559	• Erratic bal cav 2 press (328)	√	
	• LPOTP inlet Green Run fail (Max & Min)		√
	• 65/64/63% Green Run fail		√
	• Green Run test duration fail		√

Table 2. HPOTP Diagnostic System Test Results

Test	Anomaly	Expert Analyst	Diagnostic System
A20561	• Very high bal cav delta-P (low cavity 2 press)	√	√
	• LPOTP inlet Green Run fail (Max NPSP)		√
A20564	• Nose seal leakage	√	
	• PBP DS P Green Run fail		√
	• LPOTP inlet Green Run fail (Max NPSP)		FalseAlarm
A20565	• 2nd turb sl nose sl or housing leakage (aka "chatter"; seal anomaly; @10-20s)	√	
	• LPOTP inlet Green Run fail (Max & Min)		√
	• 65/64/63% Green Run fail		√
	• Green Run test duration fail		√
A20566	• Rotor shift indicated - not real.	√	
	• IMSL purge pressure Green Run fail at START.		√
	• Green Run test duration fail		√
	• 65/64/63% Green Run throttle fail		√
	• LPOTP inlet Green Run fail (Max & Min)		√
A20568	• Rotor drag.	√	√
	• LPOTP inlet Green Run fail (Max NPSP)		√
	• 65/64/63% Green Run fail		√
A20571	• Unusual rotor motion after cutoff.	√	
	• LPOTP inlet Green Run fail (Max NPSP)		FalseAlarm
A20572	• Nose seall leakage.	√	
	• LPOTP inlet Green Run fail (Max NPSP)		FalseAlarm
	• PBP Bistability		√
A20573	• Nose seal leakage.	√	
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max NPSP)		√
	• 65/64/63% Green Run throttle fail		√
A20576	• Sec trb sl cav press - channels drift apart.	√	
	• Rotor drag.		√
	• LPOTP inlet Green Run fail (Max NPSP)		√
A20577	• Bal cav pres (2) drifts up (328).	√	
	• Rotor drag		√
	• Green Run test duration fail		√
	• 65/64/63% Green Run throttle fail		√
	• LPOTP inlet Green Run fail (Max & Min)		√
A20578	• IMSL prg press rise	√	
	• 2nd trb seal chatter (@10-25s); seal anomaly	√	√
	• LPOTP inlet Green Run fail (Max NPSP)		

Table 2. HPOTP Diagnostic System Test Results, Continued

Test	Anomaly	Expert Analyst	Diagnostic System
A20579	• Nose seal IMSL leak	√	
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max NPSP)		FalseAlarm
A20581	• Cav 1 (327) 40 psi high.	√	
	• IMSL nose seal leak	√	√
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max & Min)		√
	• 65/64/63% Green Run throttle fail		√
A20583	• Rotor drag	√	√
	• LPOTP inlet Green Run fail (Max & Min)		√
	• 65/64/63% Green Run throttle fail		√
	• Nose seal leak		FalseAlarm
A20589	• Green run turbine temp fail at 109%	√	√
	• Rotor drag		√
	• LPOTP inlet Green Run fail (Max NPSP)		√
	• 65/64/63% Green Run throttle fail		√

Table 2. HPOTP Diagnostic System Test Results, Continued

V. Conclusion

The HPOTP diagnostic system was converted from Nexpert Object and Ingres to CLIPS and TekBase. In the process, the code complexity was significantly reduced (increasing the maintainability of the system), and the accuracy and coverage of the system were significantly enhanced. Initial testing has shown that the system provides a useful tool for cross-checking manual test data analysis, and for detecting anomalies and observations which are sometimes missed.

V.1. Future Work

Although major improvements were made to the HPOTP diagnostic system under this project, there are several areas in which it could be further enhanced, including:

- Performance Models — The system should review turbine and pump efficiencies and flow coefficients. At the present time these are computed only occasionally using the SSME power balance model.

- Transient Analysis — The system currently performs steady-state (constant power level) analysis only. Several significant HPOTP anomalies occur during the startup and shutdown transients, and should eventually be addressed.

- Dynamics Data Analysis — Several significant anomalies, such as bearing wear, can only be detected through analysis of accelerometer data. Currently, this data is reviewed by a separate group at MSFC. Correspondingly, this has been targeted as a separate module for the PTDS.

- Continued Enhancement — There are several subtle instrumentation and seal anomalies which are either not addressed at all by the current system, or not addressed well enough. A list of these should be identified and prioritized for future development efforts.

SSME HPOTP
Post-Test Diagnostic System
Enhancement Project

Final Report
Attachment #1

User's Manual

GENCORP
AEROJET

Enhanced HPOTP Diagnostic System User's Manual

v2.0
January 27th, 1994

I. Introduction

This manual describes the operation of a computer program which detects and diagnoses anomalies in the High Pressure Oxidizer Turbopump (HPOTP) on the Space Shuttle Main Engine (SSME). This program is part of a larger system—the SSME Post-Test Diagnostic System—which assesses the health and performance of an SSME by analyzing data from ground tests of the engine. For a detailed technical description of this program, please see the report entitled *SSME HPOTP Post-Test Diagnostic System Enhancement Project Final Report.*

The HPOTP diagnostic system is an expert system which utilizes "rules of thumb" to identify common anomalies and observations in the data. It operates by first running "feature extraction" routines to scan the raw test data and identify any features of interest, such as spikes, shifts, peaks, or limit violations. Expert system rules then analyze combinations of these features which are indicative of known anomalies. Results from the system are stored in a relational database for future reference, and can be viewed via a graphical user interface which displays observations and supporting plots of test data for a selected test of interest. The database is also used to maintain a history of certain HPOTP parameter values for statistical analyses.

System Requirements

The HPOTP diagnostic system (and the PTDS) current runs on Sun SparcStations, and requires the following commercial software packages:

- Sun Operating System SunOS
- X Windows
- Motif
- TekBase Relational Database Management System
- PV~Wave Command Language

Full installation and setup of this system is beyond the scope of this manual. Please see your system administrator for details.

Prerequisites

To use the HPOTP diagnostic system, you should already be familiar with following:
- Basic Unix commands (see *SunOS User's Guide: Getting Started*).
- Use of a windows-based graphical user interface (see *OSF/Motif User's Guide*).
- Ability to view and update tables in TekBase (see *Kingfisher User's Guide*).

In addition, you will need to have the following environment variables defined (in your *.cshrc* file):
- *NASA_HOME* must be set to the directory containing the PTDS executables.
- *NASA_TEST_DATA* must be set to a colon-separated list of the directories in which SSME test data will be stored.
- The *NASA_HOME* directory should be added to your search path (PATH).

Your environment must also be configured for proper use of TekBase and PV~Wave. See your system administrator for details.

Finally, the datafiles for the test you wish to analyze should be accessible from the machine you are running on. Currently, both compressed and uncompressed MSFC datafile formats are supported.

Execution Modes

The HPOTP diagnostic system can be run in one of two modes—interactive or as an embedded part of the PTDS. In interactive mode, only the HPOTP will be analyzed (none of the other SSME components addressed by the PTDS), and all results are simply displayed in textual form. The PTDS is configured to run as a "batch" process (typically run overnight) which analyzes all SSME components in parallel. When run as part of the PTDS, the HPOTP system writes all of its results to the database for later viewing via graphical user interface.

II. Interactive Mode Execution

To run the HPOTP diagnostic system in interactive mode, simply type *HPOTP_interactive* at the Unix command line. After a brief loading period, the program will ask you to enter the test ID for the current test. The test ID should be specified as a six-character string, of the form *A20551*, *A40134*, etc. The program will then ask you to specify the test ID for a comparison test (typically the prior test of the same HPOTP or engine). You can either enter a test ID or simply a carriage return if a good comparison test is not available (most analyses can be run without the use of a comparison test). Finally, the program will ask you for the name of a log file to store the analysis results in. You can either enter a valid filename, or simply a carriage return to indicate that you do not want a log file created.

The program will then perform its analysis of the HPOTP, periodically printing results as they are obtained. At the end of its execution, the program will ask if you want to update the historical database with the parameters from the current test (just answer *yes* or *no*). Program execution currently takes approximately 30 minutes.

See Attachment #3 of *SSME HPOTP Post-Test Diagnostic System Enhancement Project* for a sample session log.

III. PTDS Execution

The full SSME Post-Test Diagnostic System is run in four steps: (1) hardware configuration data entry; (2) test data analysis; (3) graphical review of results; and (4) anomaly database update and review.

This manual will only discuss steps 2 and 3. Step 1—hardware configuration data entry—currently involves the manual updating of TekBase tables (via Kingfisher) which describe known or expected performance parameters for the engine under test. This information is primarily used by the Systems analysis module of the PTDS, and is beyond the scope of this document; refer to PTDS Systems module documentation for details. Step 4—anomaly database update and review—involves a separate program which maintains a history of all SSME anomalies and observations. Refer to the PTDS Anomaly Database User's Guide for details.

Test Data Analysis

To begin full PTDS analysis of a new SSME test, once the data files are on-line, type *new_data* at the Unix command line. This will bring up the dialog shown in Figure 1. To proceed, type in the six-character test ID for the current test and the comparison test, and then click on Go with the left mouse button. This begins a full PTDS analysis of the test data. Currently, this process takes several hours, and is thus typically run overnight.

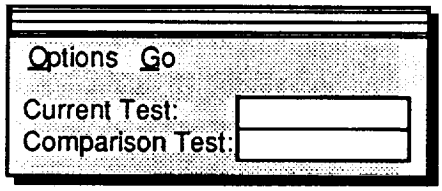

Figure 1. New_Data Dialog

Graphical Review of Results

To view the results of a PTDS analysis, or to check the progress of an analysis in progress, type *ehms* at the Unix command line. This will bring up the window shown in Figure 2. The top scrolling window displays the status of all analyses in progress in its top portion (with a check mark showing which modules have completed), and a list of all completed analyses at the bottom. To select a test to review, simply click on the test ID with the left mouse button. The system will then take a few minutes to load in the analysis results, and highlight any components on the SSME plant diagram which were found to be anomalous. To view the results of a HPOTP analysis, left-click on the HPOTP in the SSME plant diagram, to bring up the window shown in Figure 3.

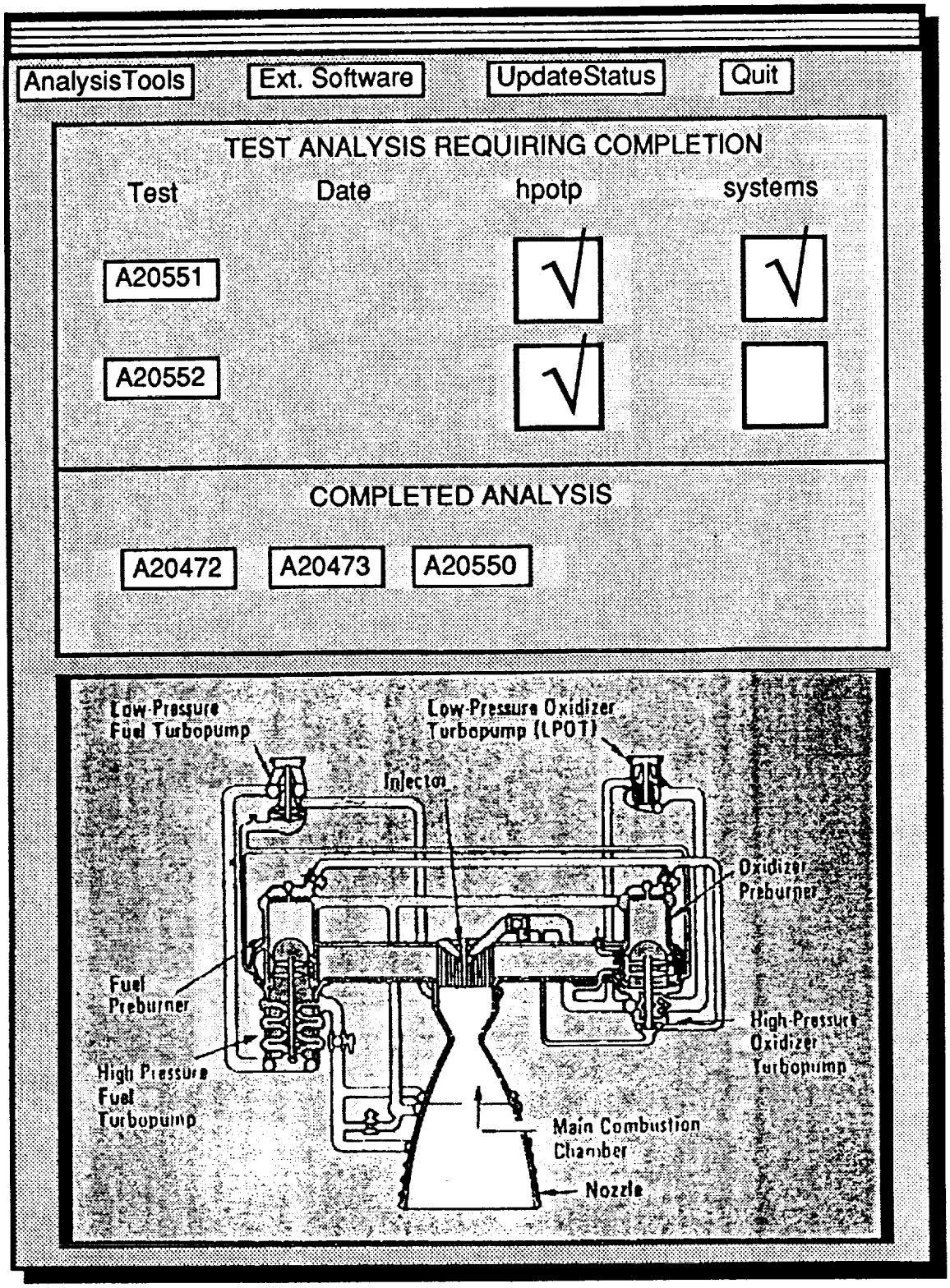

Figure 2. EHMS Window

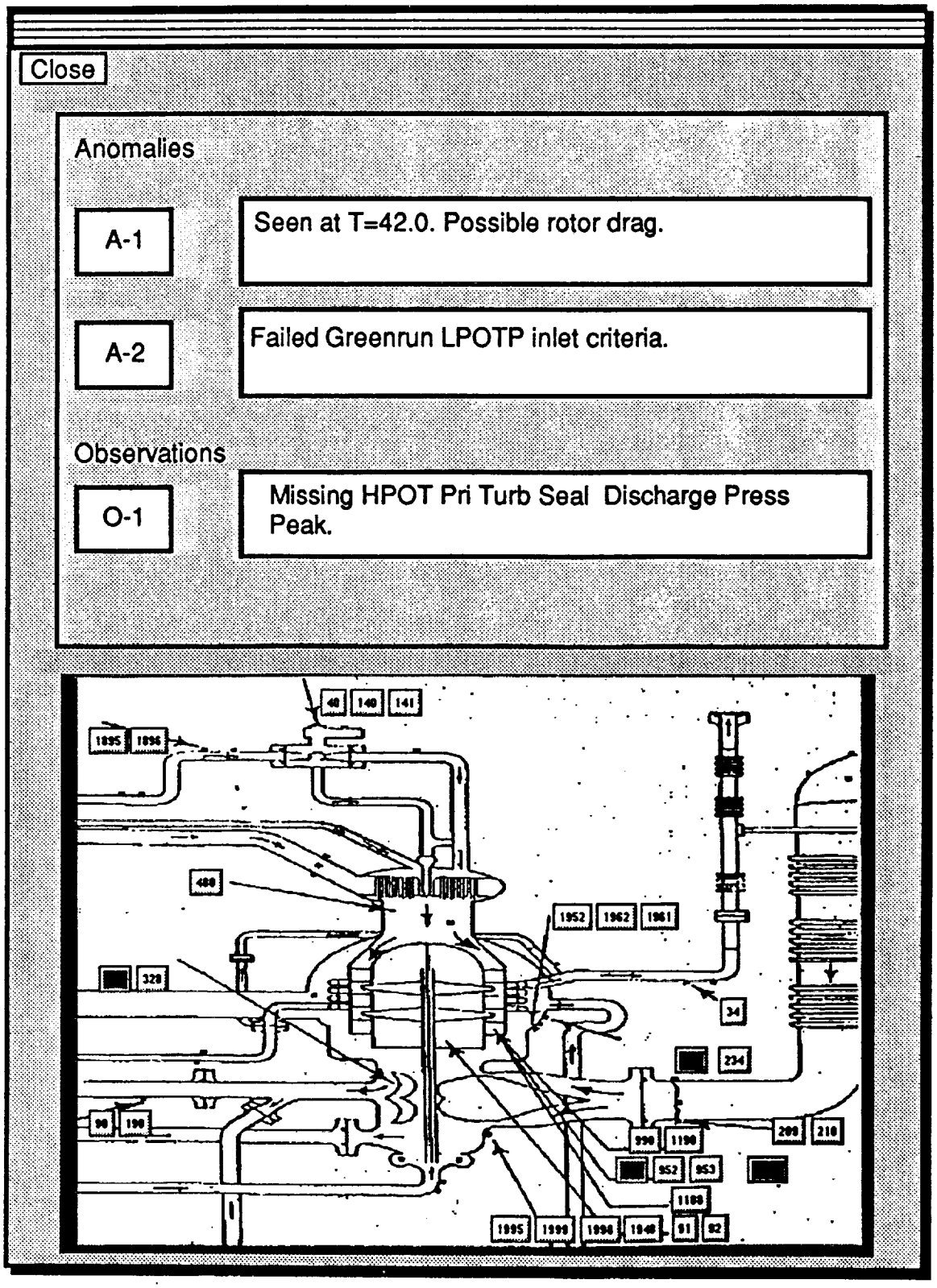

Figure 3. HPOTP Window

The top portion of the HPOTP window shows the list of anomalies found for the current test, broken down into three categories (anomalies, observations, and instrumentation) and ranked by priority. The bottom portion of the HPOTP window shows a plant diagram of the HPOTP, with buttons representing most sensors (the buttons are labeled with the sensor's Parameter ID—PID—number). To view the raw data for any sensor, simply left-click on the corresponding button. To obtain plots of data which support an anomaly or observations, simply left-click on the text of the description.

Plots are displayed in a window such as the one shown in Figure 4. The buttons along the top allow you to change the vertical or horizontal scales (range and time interval) of the display, or whether full-sample or one-second-averaged data is displayed. Note that none of these options take effect until you left-click on the Replot button.

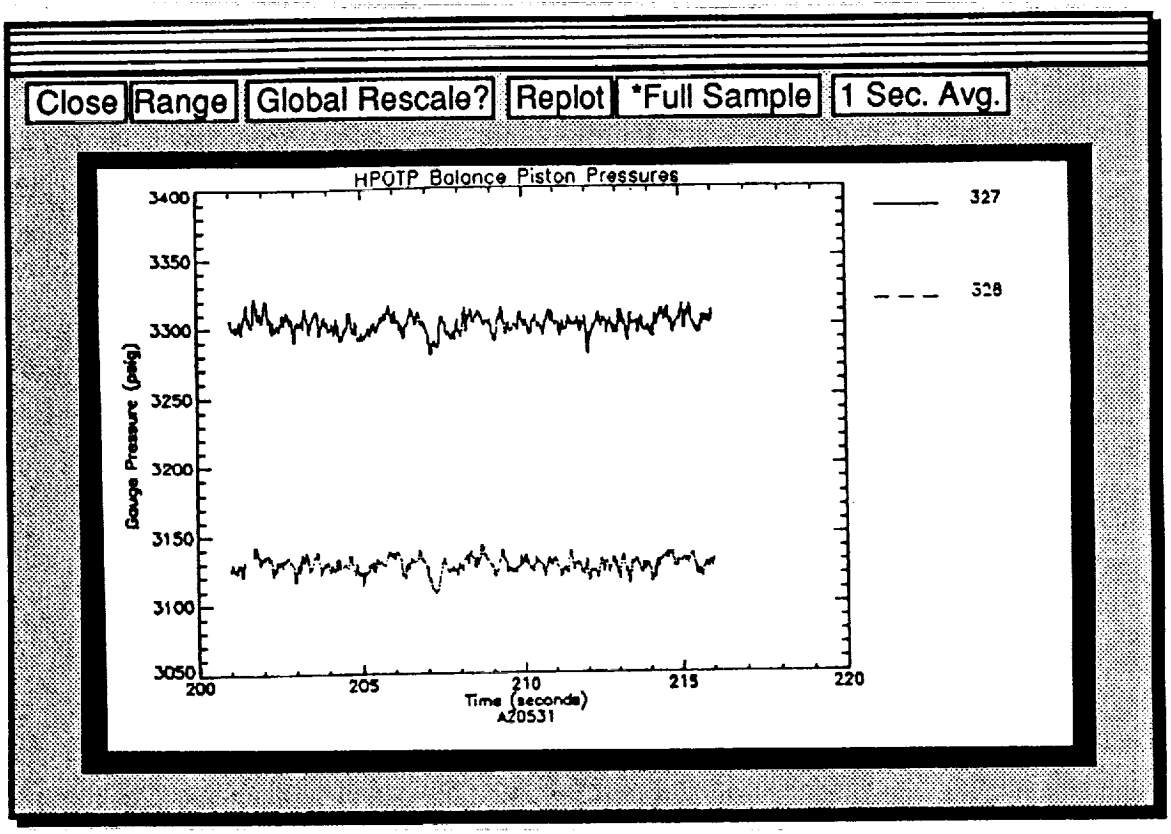

Figure 4. Plot Window

IV. Maintaining the Historical Database

The HPOTP diagnostic system utilizes a historical database of parameters for use in statistical analyses. This database is currently stored in the TekBase database named *SSME_DB* in the table named *HISTORY*. Each row in this table contains information about a single parameter value for a single test, and has the following four columns:

TESTID	The test ID.
PARAM	The name of the parameter (e.g., HPOTP_PRI_TRB_SL_DR_P).
TYPE	The type of the parameter (e.g., PEAK_WIDTH).
VALUE	The value of the parameter (e.g., 3.27).
OK_TO_USE	A Boolean (TRUE or FALSE) value which indicates if this value value should be used for future statistical analyses.

By default, OK_TO_USE values are always TRUE. However, if a HPOTP experiences a significant anomaly and you do not want some or all of its parameters used in future statistical analyses, simply set the appropriate OK_TO_USE values to FALSE (via Kingfisher).

Updating the Historical Database

A utility program is available which will update the historical database with parameters from a test without performing a full diagnostic analysis. To run this program, simply type *HPOTP_update <testID>* at the Unix command line, where <testID> is a six-character string such as A20551 or A40123.

Viewing the Historical Database

A utility program is available which will provide a quick print out of the contents of the historical database. To run it, simply type *HPOTP_history* at the Unix command line. The program will ask you for the name of a log file to store the data in. You can either enter a valid filename, or simply a carriage return to indicate that you do not want a log file created. A printout similar to the following will be output:

```
-------------------- A20571 (ENABLED)  --------------------
         PEAK_HEIGHT          HPOTP_SEC_TRB_SL_CAV_P    24.31
          PEAK_WIDTH          HPOTP_SEC_TRB_SL_CAV_P    19.67
           PEAK_TIME          HPOTP_SEC_TRB_SL_CAV_P     9.00
         PEAK_HEIGHT           HPOTP_PRI_TRB_SL_DR_P    36.97
          PEAK_WIDTH           HPOTP_PRI_TRB_SL_DR_P    22.66
           PEAK_TIME           HPOTP_PRI_TRB_SL_DR_P     8.00
              EQ_VAL          HPOTP_SEC_TRB_SL_CAV_P    11.53
              EQ_VAL           HPOTP_PRI_TRB_SL_DR_P     7.75
           START_VAL              HPOTP_INT_SL_PRG_P   188.91
            5_TO_CUT           HPOTP_PRI_PMP_SL_DR_P     0.25
         MAX_AFTER_EQ           HPOTP_PRI_PMP_SL_DR_T   427.52
         104_MIN_NPSP              HPOTP_BAL_CAV_P_A  3141.81
         104_MIN_NPSP              HPOTP_BAL_CAV_P_B  3012.06
         109_MAX_NPSP              HPOTP_BAL_CAV_P_A  3380.77
         109_MAX_NPSP              HPOTP_BAL_CAV_P_B  3258.33
         104_NOM_NPSP              HPOTP_BAL_CAV_P_A  3187.00
         104_NOM_NPSP              HPOTP_BAL_CAV_P_B  3048.27
                ...                       ...             ...
```

********** SUMMARY OF ENABLED TESTS **********

TYPE	PARAMETER	MEAN	STDDEV	N
109_MAX_NPSP	HPOTP_BAL_CAV_P_A	3286.86	77.40	11
109_MAX_NPSP	HPOTP_BAL_CAV_P_B	3076.68	178.48	11
104_MIN_NPSP	HPOTP_BAL_CAV_P_A	3053.03	75.97	22
104_MIN_NPSP	HPOTP_BAL_CAV_P_B	2863.38	143.00	22
PEAK_HEIGHT	HPOTP_SEC_TRB_SL_CAV_P	24.15	2.48	32
PEAK_WIDTH	HPOTP_SEC_TRB_SL_CAV_P	23.32	4.14	32
PEAK_TIME	HPOTP_SEC_TRB_SL_CAV_P	10.41	2.24	32
EQ_VAL	HPOTP_SEC_TRB_SL_CAV_P	12.64	1.21	34
EQ_VAL	HPOTP_PRI_TRB_SL_DR_P	8.99	1.21	34
MAX_AFTER_EQ	HPOTP_PRI_PMP_SL_DR_T	416.55	20.73	34
PEAK_HEIGHT	HPOTP_PRI_TRB_SL_DR_P	33.47	2.91	36
PEAK_WIDTH	HPOTP_PRI_TRB_SL_DR_P	25.14	5.41	36
PEAK_TIME	HPOTP_PRI_TRB_SL_DR_P	8.83	1.32	36
104_NOM_NPSP	HPOTP_BAL_CAV_P_A	3100.58	76.04	37
104_NOM_NPSP	HPOTP_BAL_CAV_P_B	2899.31	143.76	37
5_TO_CUT	HPOTP_PRI_PMP_SL_DR_P	-0.02	0.23	37
START_VAL	HPOTP_INT_SL_PRG_P	190.92	4.29	38

SSME HPOTP
Post-Test Diagnostic System
Enhancement Project

Final Report
Attachment #2

Transcripts of Interviews with Glenn Wilmer

Balance Piston Truth Table (SAIC)

Line 1: (What is described) is not real rotor motion. Seeing change in one not the other probably is due to static seal in housing or pressure shift not associated with real rotor motion. It probably is not a sensor problem.
Line 2: Level change in one and not in the other--dont know where she (SAIC) got that assumption. I don't believe we can see cup seal/washer failures or problems. Changes were made to the design to eliminate this type of problem. Also it is highly unlikely that a piece (of seal) could migrate to a pressure opening and effect that pressure. I would be skeptal that it is a cup washer, more likely it is an omni seal or a sensor problem. Data would not result in a change of the seal. All of these type of occurences are just noted. Now thaere can be other types of failures or anomolies. For instance there could be a secondary turbine seal. (Description of primary and secondary turbine seals and intrapropellent seal with helium purge) The???garbled???seal was put in backwards (I think he was saying one of the carbon interpropellent seals) and it had some pressure anomolies associated with the test program.
OK (back to the table) a change in one and not in the other looks like a level shift. One of the things we see is a gradual shift rather than a step shift. How does SAIC handle the gradual change? Its much more common to see axial rotor hang-up and Ive seen it on one test recently. This was in the test program.It is not as simple as being able to define these well defined things and say that this is unusual. Here are the individual measurments 327 and 328. You see when we come up here to 104 this is flat and this is flat and now after we come out of the bucket this one has a gradual decreasing trend and this one has a gradual increasing trend. You can see how subtle it is. The delta looks like this. This is the pump discharge pressure and basically it is just to show power level changes. Here you can see the the 1st 104 and you can see that they are basically constant. You can see in the delta there is a little change there. But here you see this curve this is what I call rotor drag. This rotor is hanging up and its moving into place slowly. This should have come on down and come streight across had it been normal. See during the 3G throttle here this looks bad but I really cannot make a statement as to whether or not there is any drag because the pressure is changing as well. This should be a streight line. The characteristic of balance pistons for most pumps is when you increase power level the delta P

decreases. Thats true except when you go well below 100% power level. What you see is decrease in power level results in increase in delta P. You see it starts to increase then reverses itself. Thats a normal trend. And this power level is below 100%. So the thing we see here is rotor drag and no one really gets hung up about rotor drag. No pun intended. What happens when you hang the bearing up the bearing reacts with additional load and that additional load can be great or relatively small. To date we have not been able to accociate any bearing damage with rotor drag or hang up. The rotor is moving but slowly. The only way to tell if the rotor is hung up is when it breaks free. It has to break free. The data we have been looking at is from Test 9020584. Looking at the second line again should not imply a cup washer problem.

Discussed the + and - as indicating a change and 0 means no change.

Line 3: Discussed that delta level shift should not be 0. In first case with 327+ and 328- the delta level shift should be +. Second case 327- and 328+ the delta level shift should be -. The postulate depends on defination of spike. If we are looking at a single data point then it is probably a data problem--its not real--have to look at it over several data points to determine if there is a real pressure change. If you look at a single point it will probably be erronous in fact I know it will be erronous. To me this was the fundumental problem with the mechanics of doing this. How do you recognize a spike or any change for that matter. For instance you could take 5 sec of data and you are going to call any thing above 3 sigma a spike. Then that still give you the possibility of sperious data spikes identified in that. Then you got to have some other type of rules that say how long does that spike last. The real problem is how the machine-the program will look at it and identify and id a potential real rotor motion vs just the data. Again I go back to the delta here. There are more things than just the delta than just what I showed you. I can't really tie anything to this-as far as rotor motion is concerned-as far as ananomolous motion is concerner. It doesnt look very pretty. But thats the human eyeball based on looking at it for a long time. So how would your program do this? Now the way your program would have to work it to do statistics on every test develop a data base and that is one critisium of what Pam did. Is that she didnt have an on going data base build up. Where basically you take different paramaters and do statisitcs on that avverage value and sigma and then be able to compare every test that you analyize with the average. Compare each parameter to the

statisical data base. What I would like to see is a model where I have the choice of selecting what tests I want to compare it with. You could choose tests of the same pump or same engine or the last test. You need the data base that has every test that has been done. You get pump to pump variations, engine to engine variations and test stand variations. Now test stand varitions are something that we may want to consider. Maybe do on test stand basis. (More discussions on how beneficial a complete data base would be) Every test in the data base. Back to line 3. This does represent real rotor motor motion but there is more to it than just that. She sees a spike up and a cooresponding spike down and there should be a spike in this also. Now you got to look at is this potentially a real rotor motion and for this to be real the rotor has to move in the proper direction. For instance, if you had 109% and it indicates it wants to move to the pump end that can't happan. The pump can hang up and then move in the proper direction but it can't move in the other direction. If we saw 327+ and 328- thats a real situation going from 104 to 109 but if we saw 327- and 328+ there is something wrong with the data because you cant have that kind of movement going from 104 to 109. The reason being is you can hang up or your can move to the proper position. If you hang up you will move toward the pump end. If you are in the proper position you won't move past that point. You wont move any further. In either case you wont move back toward the turbine end. Now if you are going from 109 to 104 and you see this, the 327+ and the 328-, is an unreal senerio where with the power level change 327 should go down and 328 should go up. So you got to have the ability to look at the data and determine is it is real. The only thing -- I've never really seen a spike up and a spike down would say I've moved to that position and moved back. Why would you move back. Now again trainsants are not considered by this model right? Right. So if you are hung up and you move there would be no reason, no force to move you back so this would represent not a real senerio. If we ignore the second part of Line 3 and look at the + at 327 and - at 328 would that be a real be a real senerio? There is a possibility. For instance say, if you had a turbine anomoly, somthing that effected the turbine efficiency - and then it fell back into place then that could be a real senerio. I've never seen that happen. Having this in here would cover a possible senerio. Now having said that the logic has to consider the power level change. A spike senerio is more appropriate to a power level change. This is the kind of thing that make it diffcult to try to do this kind of thing- - there can be so many possibilities as to the cause and then you have to look to see if you can justify the cause. A spike senerio here

associated with an efficiency change or a turbine that corrects itself, in fact we see that with the ATD where there are changes in effeciency. That was another question I had for you. Do you plan to do this for the ATD pumps? In the long term we would like to. The alternate pump has different characteristics and this won't work with it. ATD seal are so different this won't work. The ATD uses hydrogen to cool bearings and as a consequence is the fluid that goes through the drains and seals. The current pump bearings are cooled with LOX and the turbine exhaust is what goes through the seals. So there is an entirely different characteristic. More discussion on ATD. Back to the Truth Table--discussion on the (Line 4) + & - on level shift as we were trying to figure out what Pam really ment--See the delta is 327-328 so if this one goes up and this one goes down then this should go up and its possibly ananolus rotor motion considering this is constant power level. Now the other caviought is that you have to look at the venting. If its venting you will see ?????. Tape 1 ends.

Current to previous is greater than 0. Is this a delta or is this a ratio?? Current - previous is positive that means the current it hotter. It has to be a negative. So current is hotter, now this peak, current - is less than 0 so that means that current peak pressure is less. Hot gas coming through on the seal is causing the seal to grow. Thats why the pressure goes up. At the same time youve got the thermal feed into this massive piece of metal from the hot turbine tips which causes the the shaft to grow. But the effect would be if you had hot turbine temps the seal would grow faster and eventually you would have a highter peak pressure and then the shaft would catch up your equlibrium pressure would be lower. The possibility exists that ????? We'er saying that the seal is growing faster than the shaft. So really the peak all things being equal if you have high turbine temps initially that this thing would grow even faster. I'm not sure that without some kind of analysis to back that up that you can draw that kind of conclusion. So the peak should be highter and this should be lower. But I'm really against making that kind of statement. You have so many other variables -- the potential for different pumps, different purges, and I don't know what it serves to make that kind of assumption. I think its best to stick with peak-data base, equlibrium-database and erratic nature and not tie to turbine temps. The only thing I'd try to tie to turbine temps is delta P. And I don't know where that is in her package. Now is that whole page tied to turbine temps? The only one that isn't is item 4. Offset 1 times offset 2 greater than 0? This must be the peak column and

this must be the equlibrium column. This says from previous test in both peak and equlibrium valve may be change in seal clearance or sensor calibration. I don't really care what the peak is or the equlibrium is as long as it resides within the family. I dont see what utilarian value it has. Its tied back to turbine temp again but outside of that I don't see any difference in this. So again what we're looking at is primary turbine seal pressure peak and equlibrium data it would be best just to compare the steady state data with a data base and perform statistics. Would that be more valuable to you as a flag if something like a seal pressure peak fell out of the norm. To me what I really do is look at the data and say is it in the family? And I look at the character of the data-- the peak and the equlibrium and even though I don't have an average value in my head or a sigma value in my head I know if it is in the family. If its in the family it passes. A statistical data base would do the same thing that I do and the erratic test would do the same. So I wouldn' tie to turbine temp or any specific test. I'd compare to data base.

Table 5 --- This - the first two lines deal with missing data. Line 2 is missing the previous test data so if we were dealing with a comparision to a data base we wouldn't worry about previous test. Also with a data base the user could define and accept the comparison so it wouldn't matter if the previous were there or not. Again to me the user should have the ability to define what to compare to. Also need to be able to compare to Green Runs Spec Requirements. Its not only pertnent that it fall within the family but its also pertnent that it meet green run requirements. I think that this cut width is something she used to define how long the peak pressure occured. That doesn't have any meaning to me either. These peaks generally occur between 15 and 25 seconds for both the secondary and primary. I don't ever recall one falling out of the 25 seconds. But if is were less that 50 seconds-- where you want to look for the peak is between 5 and 50 seconds. It would never,never fall outside of 50 seconds. That is so far past where the peak would ever fall that I'll never have one that I miss using these times. I dont understand this peak height. What it almost sounds like to me is she is looking at the signal for the data. I don't know where she gets the sigma between this test and the previous test. But again I think we can dismiss this one like we did the other--with the peak and the steady state against the statistical data base. And you dont worry about any direct comparison between any test as far as this parameter is concerned. I wonder if cut width is some value she can

put in like one sigma--again to me it is not important when the peak occurs, I know it will occur before 50 seconds, and I want to know what the peak is. To me its not meaningful when the peak occurs. The fact that one occurs at 15 sec and another occurs at 20 sec doesn't really tell me anything about the health of the seal. The most important information is in the peak itself. That gives me some indication of the maximum clearance that the seal sees during operation. From that you generate an indication of the health of the seal. Peak 990 ????I think here she put in whatever she could think of. Full width at half max doesn't have any meaing to me. It doesn't tell me anything. Its really very easy: you look at the max, the steady state and the characteristic of the data and those three things and thats all. F om that you can see if you have a transducer problem, a seal problem, wether the seal is operating within the family--that type of thing. Same thing for equlibruim line. The same identical thing applies to pid 91 and 92 as does to 99. Look at the erratic nature however you choose to do that. Is it there or not there. Then go the the statistical data base for peak and steady state at some timeframe like 150 or 200 sec where you choose and look a peak, steady state and erratic nature. And so the same thing applies to the secondary as applied to the primary. Now, the 990 is a single measurement, its a facility measurement, not flown. 91 is the red line so it is a Secondary turbine seal cavity pressure so it is a single transducer with 2 bridges. 91 &92. You will probably want to think in terms of looking at both to make sure that both bridges are ok. So what would apply to 91 would apply to 92. Now the statistics have to be done with max and min, plus and minus 2 and 3 sigma because the two failures you would want to catch are the failure of the frozen primary seal in which the pressure went to 0 so that would be a failure at both peak and equlibrium to follow the data base. And the other one was the secondary turbine seal that was put in backwards. Now in the one will depend on how you define erratic. Erratic defined as cyclic would cause us to miss this anomoly. You would want the model to catch this one. The defination is the hard part. If you define erratic to tight then every test will come up anomolus. If to loose you will miss catching something that should be caught.

Table 6 --- Put in the - on 91 and 92 - I look at this the same as the other one. A problem of erratic defination. Also the defination of spike comes in here. Comparison between 91 and 92. What you need to use 92 for is a sanity check with 91 and visa versa. Whether you are looking at spikes, erratic behavior or changes. They should be

very close to each other. But sometime we see 92 and 91 separate and that normal and happened on a flight engine. If they differ you want to see why. Do you have a bridge failure and you have a problem or you have just one bridge fail and you dont have a real problem. How would you figure it out? If one looks normal and the other is erratic I would say that it is normal because one is ok. Now if both of them are abnormal together and they agree with each other then of course I would say there is something abnormal with it. Example of flight data provided. Problem is to define a normal trace and then be able to pick up something abnormal. Another example: A spike in both 91 and 92, first thing I would say because both are showing the spike that it is a real occurance. On the ground you have 990 so you would want to go back and see if the spike existed in that seal. If yes that explains the secondary it does not explain the primary. Then you have to flag a problem with the primary seal drain or instrumentation. So on the ground you do have the upstream pressure you can look at. If it exists in 91 and 92 I would be extreamly suprised if it doesn't exist in 990. So if it is a ground test and you see anomolous behavior in 91 and 92 we should look at 990 to verify that a real problem occured.

LOX Seal Table 8 951 is one of the pids that addresses the pressure in the lox drain. 1187 is that temperature that I showed you that you can pick out whether its a new pump or not. 95% of the time it runs at 160 upwards to 450. I think from a erratic criteria needs to be able to descriminate. The thing that bothers me here is that 951 could be erratic for cause and it would not necessarly cause 1187 temp measurement to be erratic. So should not try to couple the two measurements. The thing to do would be to classify either or both as being erratic and go from there. I believe that I would do 951 like we did the other pressures. Establish the data base and you compare each test against that database. You also analyze for erratic behavior and if it by itself fall out then flag it. The temperature of the erratic test is appropriate. There are two characteristics and one or the other is always there. What you look for is something different. The two characteristics are the new pump temp and the used pump temp referenced above. Again discussion on how to define erratic behavior. How do you know what is different? The temp unless you did the whole test profile generate a running average and sigma with time then compare a given test running with time for the average and 2 sigma. Now that gets you into doing every data point with average and sigma. We do that with certian parameters. But that gets you into a big database cause you'll have every data point

in that file. Make for a large computer! Is this something so critical that it would warrent this type of database?? NO. To me all the seals are very reliable and within their own families very reproducable. You see very few anomolies. These seals are very well behavied. The only thing you want is to have confidence that (the temperature measurements) the program sees it as normal. What you may want to do is always look at the temp at 200 sec and see that it falls between 350 and 400. Again what you might do is establish a normal as above and if it falls out of that range flag it. If it passes its erratic behavior criteria and its 200 sec criteria then you can say you have a normal seal. That would be a good approach to that one. In the lox seal there are a lot of variations that are normal that you have to accomodate. The lox seal on a new pump has a characteristic that you wont see the second time the pump is run. The reason for that is the wear in characteristics of Kel-f. It interfers with the labrath teeth when its new making a positive no leak seal. So on the first test you'll see low drain line temps. and then after its worn in the Kel-f will have groves so it will have a fixed clearance. So this again is a -- anything above 160 at start is an indication of a new seal. Also a new seal has a very well behavied character. It also can be somewhat erratic and thats acceptable too. Thats the wearing in process. The next test it will come up to 400 to 450 and have a very smooth characteristic. You can see slight change with power level change. The temp is a very good indicator of the performance of the LOX seal. Basically there are no problems with the Rocketdyne LOX seal. We have have some anomolies occur. In one case the seal was put in backwards and in another case a housing had a drilled passage that allowed hydrogen to get in to the coolant circut for the primary seal which freezes up the primary seal completely. Then we see a couple of small anomolies with the intermediate seals but basically there are no problems with the Rocketdyne seal packages. Now the seconday and primary turbine seal drain line temps and pressures (description of seals) measured outside the pump. Secondary seal temp comes in around 450 and it runs up to 800-900 and slight decrease at 109%. The primary is hotter. So here it is typically 1000-1100 at the bucket and this is measured also in the drainline. It is about 200 cooler than the secondary. This is the primary turbine seal drain line pressure measurment. Same location as temp. Varying clearance up to about 20-25 sec where clearance opens and then begins to close down as the shaft warms up. Takes about 150 sec to reach thermal equlibrium in seal package. There is a green run established for peak pressure and steady state. I've never seen one

fail green run requirement. The LOX drain temp is measured in the LOX drain outside the pump. This is the secondary turbine seal cavity pressure. Its measured internal to the pump. This is a red line measurement. There is a flight redline on this measurment of 100 psi max. Thats why the peak pressure is so important because this is the last?? pressure obtained during runing of the pump and its again at that 15-20 sec timeframe. and if that exceeded the 100 psi it would shut the whole engne down. The average pressure for this measurement is 26 psi with a 2 sigma variation. Also there is a green run requirement. Intermediate seal purge pressure now this pressure is measured in the TCA down stream of the sonic ports. The shaft will grow with heat input which closes down the clearance so as soon as you start you have a thermal effect. This typically has a soak back effect similar to the LOX drain. The characteristics of this seal vary greatly from pump to pump and even from engine position to engine position in flight. We have alot of what I call flat top characteristics they come up and then are basically constant. It occurs more often in flight than on the ground. The more typical is you come up and you go through the 150 thermal plane and you reach a peak and may start to decrease again. You see there are may different characters to it. This to me is difficult thing to catch in the program because there are so many things thatvary greatly but are normal. Ok back to the table--Primary turbine seal temp and pressure -- erratic 990 temp not erratic no spike. The essence of the whole thing is what is erratic. Showed examples. Normal is smooth. Again this is where you could apply the 3 sigma to the data base at a particular time(s). This could be a very useful tool for determining erratic behavior. It could be a seal anomoly (what Pam described) or very definately a vibration effect. I would say that depending on how erratic is defined is critical. Next case--Erratic temp not erratic ? no spike -- all shes doing is saying erratic anything--just covered the bases. Thats all. All combinations of erratic not erratic with or without spikes addressed. I guess I wouldnt see a need to change it but again it depends on how erratic is defined and tested for. Can it really call out an anomolus condition vs normal.

Primary turbine seal. Pressure peak and equlibrium checks. 233A is more discussion on being able choose test(s) to compare to. Doesn't work with previous test. Problem if trying to Pratt pump to RD pump. Compare to previous peak height and offset. Better to do data base comparison on previous. These measurements are dependent on the thrust profile and will vary with profile so must choose a good

comparison. Must establish ground rule. Mutiple application or ground test or flight. Again establish data base and compare current against data base at 150 sec and run statictics. I would like to see at least a peak and some point defined as equlibrium or steady state looked at and compared against the database. 150 or 200 sec would be appropriate for steady state. More discussion on table--peak measurements shifted in opposite directions--mirror images--this is impossible. Again there is so much variation from test to test it is much more meaningful to compare to database and run stats. Delta T across primary turbine seal is a green run requirement to assure that the hydrogen and hot gas mix together in mixing chamber. Description of mixing. Keeps seal cool and maintain a non-interference clearance. A green run requirement was established for the between the difference in the primary drain line temperature and and the turbine discharge temperature. The goal was 400 degrees. The real requirement is in the green run spec. I never saw any reason to try to coorelate the peak pressure with the turbine discharge temperature. Lots of discussion of why to tie the two parameters together.----Just look at individual data and compare to family. Discussion about offsets listed in table. Maybe this is the reason why some things didn't work.

- I never had any experience with this part of the program.
- Darryl Gaddy said several things didn't work.
- Look at peak, equilibrium, compare to statistical DB; make no correlation to turbine temps. There's no reason to do that.
- Primary turbine seal delta-T, may be appropriate to look at turbine temps, especially if you've failed a requirement. For example, if you fail green run requirement, then look at turbine temps and if turbine temps are cold, then don't flag the problem as a pump problem. That's the only thing I would use turbine temps for.
- I just look at peak and steady-state.

- Offset1, offset2, current, previous difference; turbine temps. Difference 233, previous to current.
- Peak offset 1 < 0, peak offset 2 < 0, turbine temps higher; closed down. Turbine temps are higher current to previous. Peak is less. Previous has higher peak.

9/14-9/15 Greenrun spec

- 585: This is not a launch pump greenrun test; this is a fuel pump greenrun test. This is a slave pump.
- You can see those 2 pressures approaching one another. That is the sanity check that you do when you see this... You are at steady-state. Ideally you would be straight across. This is decreasing slightly. This is rotor drag; indicating that the rotor is hung up slightly (very slight). Always flag this.
- If something falls outside the norm, I will flag it. Example, nose seal leakage; if it is not a flight pump it is not a reason to consider the pump unacceptable. But I will still flag it to management to let them know. For example, there is only one observation on this flight; one of the fuel pump coolant lines went up ??? psi.
- This is the same pump, rotor drag on both tests.

- See those loops; that's bistability. You don't see it? Incipient bistability; there is no system response, but see 3sigma violated by a data spike. Here is the system involvement, and one instance of bistability.
- If a pump is bistable at 65% it will be more bistable at 64% and even more at 63%. So if you see bistability at 64% and don't see any at 63% that is an argument against "their" approach (Rocketdyne/SAIC?). Must use OPOV involvement.
- It only has to be bistable at one power level to fail greenrun spec.
- Then look at pump DS P, and see it in the raw data.
- See how Pc is involved? There should be OPOV involvement.
- If you take the data from this test and expand it, you would see it in MCC Pc, would see it in PBP DS P, would see it in presure ratio across pump, see it in OPOV commands. This is a good case of PBP bistability.
- Expand the PIDs SAIC uses to use these.
- The one test that SAIC picked up that we didn't catch was one that had one small indication that our chart wouldn't pick up. That's the one disadvantage of our approach.

- The systems people used to do the Rocketdyne approach; we lost the people who used to do this. The new guys don't do that anymore.
- If the engine doesn't go to 65% don't need to do a bistability investigation.

Greenrun...

- Nominal inlet pressure for LOX side is 100psi, fuel is 30psi. Look at these, look at nominal MR, nominal interface pressure. The lowest requirement is 20 +5/-0 NPSP. This is 6 +2/-0. These are the nominal conditions for the acceptance test to be run at.
- These talk about what components are to be tested.
- HPOTP: Talks about pre-test. These are the time requirements: 250sec min, 50 sec at FPL, rest at 104%. 150 above 100%; 50 at 109%; rest at 104%. Greenrun is 300 sec. That's minimum amount of time they can do the thrust bucket, PBP bistablity, venting of LOX & fuel, repress of LOX side.
- LPOP follows HPOTP.
- T DS Ts; 4 time points: START; Peak transient (10-25s); engine NPSP LOX side 20-25, at ≈450sec; (at 150 they start pressurizing back up); at 140-150 after 175sec at 109%. Turbine DS Ts are there mainly to provide ensured margins to the flight redline.
- Speed change from nominal. Going from nominal to low; going from nomial to high. This doesn't really have a reason for being. Still in greenrun though.
- Speed file is available sometime after test has been run and dynamics people have generated it.
- PBP DS P ; a systems requirement. 70/40. 75/15 at 109%. Ensures that pump is producing enough pressure to offload both preburners and not have the engine thrust limit. Have only seen this failed by one pump; it was accepted because it was marginal.
- ISP Prg P; at start is 180 min. This has been revised. This is 5 psi higher than LCC which is 5 psi higher than redline. The redline is 170.
- There are 4 distinct times for greenrun spec. But if you exceed these requirements at any time, should be flagged. If you fail the peak transient (370) test anytime after 150 sec at 104% or 109%, you should fail the pump.
- HPOTP module should check greenrun tests.

- Turbine temps: should check greenrun tests on these; but don't use for anything else. Only tie early delta-t to turbine temps; use turbine temps to explain why the delta-t was low.

- 510: IMSL prg P. This has the nose seal leak characteristic, and some rubbing. Need to identify this. I haven't seen pure rubbing signature in a long time.
- Example: The fact that it didn't change—intake to rotor—should have moved and didn't. Not indicated in 3G throttle. My interpretation is hung at 104%, stayed hung at 109%, then broke free at 80%, then looks normal below this. Clear indication of rotor hanging.
- ISP prg P; see it come up here and drop back is nose seal leakage.

- Look at engine He interface pressure (upstream of the sonic orifice) at 210s, its coming up, nominal, then at 243 an anomaly(?). So, need to look at upstream pressure, the 937 PID, to see if it can explain a pressure change in ISP prg P.

Flight data

- Get 1sample/sec during ascent. Then we get full 40ms data after orbit.
- Don't fly all sensors on used greenrun.
  Dont fly: Primary turbine seal drain line temp, p; sec turb seal drain line temp; lox drain temp and press, balance cavity
  Fly: sec turb seal cav p is redline; ISP prg pres is redline, hpop ds p, pbp ds p, both turbine temps;
- Sec turbine seal cav P: 12psi steady-state is average for flight. 26psi is average on ground.
- ISP P: Steady-state as low as 220, 230; as high as 400. Very seldom see it that high.

Engine 1

- LPOP DS P:
- PBP DS P: Tailoff of solid burn shows up, but don't see on HPOP DS P.

Engine 2

- HPFT turbine temp sensor failure. Have seen it wander up. This one failed hard (within one 40ms cycle). Had the other sensor failed by wandering up (it takes 3 strikes above the redline) it would have cut the engine. Normally channel B runs higher than A on HPFTP. This one had channel A higher. This means that channel A's redline is adjusted based on difference between A and B on ground test.
- Sec Seal: Peak occured before 5 seconds. I've never seen one do that. Should be flagged.
- ISP: Normal

Engine 3

- ISP anomaly:
- Coolant liner: has some icing.
- Pump accepted for flight, but has abnormal spiking character. Should flag. When I saw it on greenrun I though it was instrumentation because it was fairly constant frequency. Very repeatable behavior.

Table 14

- Same category as PIDs 91 & 92. ISP prg press. PID 11 erratic & PID 12 erratic. Looks like a matrix of possibilities. Use one against the other for sanity check. Also, you know what normal looks like, so if one is normal and they disagree, you know which one is abnormal.
- T=0 pressure. Take into consideration. Also use steady-state number.
- Steady-state doesn't coincide in the same way as P turb sl drain line pressure (990). It does tend to reach a steady-state. If you took points at 150 sec, you would see 200 to 400 as normal. Waveform of curve also can vary considerably. So, will have high sigmas for steady-state, but low sigmas for engine start. Start number tells us our seal clearance. Very useful data. The

steady-state doesn't tell us much: based on thermal condition of pump, seal clearances. Clearance decreases to as low as 1-2mil, so very small changes cause big variation in pressure.

- Acceptable anomalies (not of real concern): nose seal rubbing and light rubbing(?). Both look like erratic data for short period of time. Rubbing may occur for one cycle. Causes pressure to go up, as seal rubs generates small amnt of heat, causes ring to grow and presure to drop. Rarely it will repeat (oscillate).
- Nose seal leakage: Common. Pressure will drop 2-3 psi, will stay down for some period of time, then will seal and presure goes back up.
- Vibration will show up as major erratic behavior. Large excursions of data over short time intervals. Not a problem caused by the seals, but one the seals will respond to.

- Look at data from statistical standpoint, erratic standpoint, and flagging anything outside of normal.

Data Package Walk-Through

- Charts are numbered in the order I look at them.
- Sec T sl cav P: redline. Interested in: Peak prior to 50 seconds; steady-state pressure after 150 seconds (on greenrun), although still moving at 150 due to thermal response. Chart #7.
- Chart #4. P T sl dr line P: ground test only. Peak and steady-state of interest. This one is relatively high. 32-35 is normal.
- ISP Prg P #6: Most important at T=0. This is very high, 197-198. PID937 is related; He interface pressure, drives this. For flight and greenrun PID937 is usually flat (around 750), so don't usually look at it. ISP Prg P is basically flat after 25sec. Can vary 220-400 at peak time. See power level changes, vent & repress.
- P & sec dr line temps: are flown. Secondary is erratic due to instrumentation. Should pick this out as erratic behavior. Primary usually runs at 420 and secondary at 450. Just check to see if it falls within the family. On secondary is between 750 and 900, primary between 850 and 1050. If high will cross-check with turbine temps to explain.
- Chart #1, Primary LOX sl dr ln P: Basically stationary at +/-0.5psig. Anything different is an anomaly. 951, 952, 953. Have tendancy to drift, so problems are usually instrumentation. This is normal data.
- Chart #8, BalCav 327, 328: Delta between them important. If you see a change in the delta, come back and look at the raw data to see if they're moving in the right direction. Both of these are decreasing at different rates indicates a vent is occurring.
- Chart #14, MCC Pc: Get power-level changes.
- Chart #19 LOX P T DS Ts: These are well behaved and cool. At min NPSP max turbine temp is 1300 degrees. I don't use these too much for anything specifically. See power level changes and vents. Can see changes in one and not the other and can't explain with OPOV changes. Happen so frequently we just overlook it.
- Chart #21, Facility LOX flow: I never look at these; too erratic.

- Chart # PBP DS P: There are system requirements on this: at 109% max NPSP, 7520 requirement. I look at as an indicator of bistability. I can also see the speed of the pump through here. Otherwise I don't use this much. 341-334 is deltaP.
- Chart #28 HPOP DS P: 334-209 is deltaP. This is a controlled parameter. The HPOTP module and systems module should interact to ensure that changes in this are due to commanded changes. Really a function of the LOX-side engine resistance.
- Chart #29 LPOP DS P: Shows strong influence of vent. Is the low pressure orifice of the balance cavity sink (DS) pressure(?).
- Chart #30 MCC Ox Inj T: Closest thing to a HPOP DS T. In the LOX dome of the MCC. I just check to see if its in the family. Under 200 at 109% is normal.
- Bal Cav DeltaP: This one has a slight drag indication. I would go to chart #8 to check if they are changing at the same rates. Very subjective (don't compute differentials). This is a very clean pump. HPOP DS P plotted just to have a power level reference.
- P T SI DeltaT: 233 - 1190 (drain temp). This meets all greenrun req'ts. Goal is 400 deg; this meets but is very late in test. This one is almost perfect.
- Chart #33: This is the primary chart that we use to detect PBP bistability.
  Rocketdyne: Takes 5 seconds of data from 65, 64, and 63%, and do a statistical analysis. Any data below 3sigma is bistability.
  Wilmer: DeltaP across PBP vs. Pc should be linear. Bistability would show up as changes in slope.
  SAIC: Also looks at OPOV and Pc.
- LOX drain T: Check at engine start, should be 150-160 with new seal, anything over 200 is warm.

Low Pressure Package
- Chart #2 Eng Ox inlet T: Just check to see if normal. 160-167 normal.
- 209, 210: HPOP In P
- Chart #4: Vibration levels for LPOP. Just get vent confirmation.
- Chart # : Eng Ox in T vs. HPOP In P: Can see the vent.
- Chart #12: LPOP Spd: 5420 max speed greenrun.

- NPSP on LOX side: important for cavitation. I don't think you could pick up cavitation on HPOP at steady-state. DS P constant by definition. You would see turbine temps go up. Should be handled by systems module.

This Test
- No bistability, no unusual rotor motion, deltaP for P T sl is normal (goal is 400).
- Very nominal test.

- Would like supporting plots for bistability to show 3-sigma floor chart, and "furrball" chart (PID 341 - PID 334 vs. PID 163).
- Rotor drag is less probable after thermal equilibrium.
- On comparison tests: We very rarely have the same pump come through twice. Typically a pump will go through greenrun, then directly into the flight program.
- Can cross check IMSL purge pressure with PID 937 (He supply) to explain unusual deviations.
- If delta-T is low, check if due to poor coolant system or low 233/234.
- On a new pump, Kef-F is just wearing in, so see Priamry LOX pump drain line temperature and pressure very high (250-300 temp, 14.7 press) compared to worn-in pumps. After just one start, temperature will usually be below 160 and pressure 1-1.5 psig.
- Other PIDs of interest:
  ◊ MCC Ox INJ T — Closest thing there is to LOX pump discharge Temperature.
  ◊ PBP DS T
  ◊ OPOV — To check unexplained speed changes (was the system involved?).
  ◊ Eng OX In Temp — Just a sanity check. Should be 164-167R.
- Would like to see analysis of pump and turbine efficiences and head and flow coefficients included in system.
- Screen run = 80sec test.
- Acceptance test = 550sec.
- Nose seal leakage usually shows up as a small dip down in 211/212 followed by a return to normal. Rubbing (less common than nose seal leakage) shows up as a spike up followed by a spike down (also on 211/212).

SSME HPOTP
Post-Test Diagnostic System
Enhancement Project

Final Report
Attachment #3

Example Run of System in Interactive Mode

```
Performing ConstantThrust analysis for A20547.
Thrust Profile for A20547:    7 -   15 @ 100%
Thrust Profile for A20547:   18 -   25 @ 104%
Thrust Profile for A20547:   31 -   39 @  65%
Thrust Profile for A20547:   42 -   50 @  64%
Thrust Profile for A20547:   53 -   61 @  63%
Thrust Profile for A20547:   68 -  358 @ 104%
Thrust Profile for A20547:  361 -  410 @ 109%
Shutdown for A20547 at  420

*** Checking for hard sensor failures.
Performing DataAvailability analysis for A20547.
Disqualifying HPOT Discharge Temp (518) for test A20547. Does not exist.
Disqualifying HPOT Discharge Temp (521) for test A20547. Does not exist.
Disqualifying HPOT Discharge Temp (519) for test A20547. Does not exist.
Disqualifying HPOT Discharge Temp (522) for test A20547. Does not exist.
Performing NoisySensor analysis for A20547.
Performing DisconnectedSensor analysis for A20547.
Performing SensorAmbientLimits analysis for A20547.
Performing SensorReasonableness analysis for A20547.
Performing SensorRedundancy analysis for A20547.

*** Determining preferred sensors.
Performing SteadyState analysis for A20547.
LOX Vent Profile:  362 -  366 : ENGONPSP Changing from 138.77 to 145.49 @ 109%
LOX Vent Profile:  366 -  370 : ENGONPSP Changing from 145.49 to 145.84 @ 109%
LOX Vent Profile:  370 -  410 : ENGONPSP Changing from 145.84 to  76.33 @ 109%
LOX Vent Profile:   69 -  217 : ENGONPSP Changing from  83.45 to  83.49 @ 104%
LOX Vent Profile:  217 -  289 : ENGONPSP Changing from  83.49 to  23.01 @ 104%
LOX Vent Profile:  289 -  305 : ENGONPSP Changing from  23.01 to  21.16 @ 104%
LOX Vent Profile:  305 -  358 : ENGONPSP Changing from  21.16 to 131.98 @ 104%

Performing SensorAnomaly analysis for A20547.
Using sensor ENGONPSP for Engine LOX Inlet NPSP (special calc), on test A20547.
Using sensor 1190 for HPOTP Pri Turbine Seal Drain Temp, on test A20547.
Using sensor 1188 for HPOTP Sec Turbine Seal Drain Temp, on test A20547.
Using sensor 1187 for HPOTP Pri Pump Seal Drain Temp, on test A20547.
Using sensor  990 for HPOTP Pri Turbine Seal Drain Press, on test A20547.
Using sensor  953 for HPOTP Pri Pump Seal Drain Press, on test A20547.
Using sensor  937 for Engine Helium Interface Press, on test A20547.
Using sensor  860 for Engine LOX Inlet Press, on test A20547.
Using sensor  328 for HPOTP Bal Cav Press B, on test A20547.
Using sensor  327 for HPOTP Bal Cav Press A, on test A20547.
Using sensor  234 for HPOT Discharge Temp, on test A20547.
Using sensor  212 for HPOTP Int Seal Purge Press, on test A20547.
Using sensor  210 for HPOP Inlet Press, on test A20547.
Using sensor  141 for OPOV Actuator Position, on test A20547.
Using sensor   92 for HPOTP Sec Turbine Seal Cavity Press, on test A20547.
Using sensor  190 for HPOP Discharge Press, on test A20547.
Using sensor   59 for PBP Discharge Press, on test A20547.
Using sensor   24 for MCC Hot Gas Injection Press, on test A20547.
Using sensor    2 for HPOTP Speed, on test A20547.

*** Determining diagnostic features.
Performing ComparisonDifferences analysis.
Performing BalancePistonShifts analysis.
Performing DrainPeaks analysis.
Performing FamilyChecks analysis for A20547.
Performing GreenRun analysis.
Performing IsFlat analysis for A20547.
Performing Bistable analysis for A20547.
```

```
Historical Value of HPOTP_BAL_CAV_P_B is 3076.67 +/- 535.42 (3 StdDevs).
Historical Value of HPOTP_BAL_CAV_P_A is 3286.85 +/- 232.19 (3 StdDevs).
Historical Value of HPOTP_BAL_CAV_P_B is 2856.61 +/- 428.65 (3 StdDevs).
Historical Value of HPOTP_BAL_CAV_P_A is 3051.97 +/- 233.04 (3 StdDevs).
Historical Value of HPOTP_SEC_TRB_SL_CAV_P is 10.35 +/- 6.77 (3 StdDevs).
Historical Value of HPOTP_SEC_TRB_SL_CAV_P is 23.36 +/- 12.58 (3 StdDevs).
Historical Value of HPOTP_SEC_TRB_SL_CAV_P is 24.14 +/- 7.55 (3 StdDevs).
Historical Value of HPOTP_PRI_PMP_SL_DR_T is 416.12 +/- 62.69 (3 StdDevs).
Historical Value of HPOTP_PRI_TRB_SL_DR_P is 9.02 +/- 3.62 (3 StdDevs).
Historical Value of HPOTP_SEC_TRB_SL_CAV_P is 12.65 +/- 3.67 (3 StdDevs).
Historical Value of HPOTP_PRI_TRB_SL_DR_P is 8.80 +/- 3.97 (3 StdDevs).
Historical Value of HPOTP_PRI_TRB_SL_DR_P is 25.18 +/- 16.45 (3 StdDevs).
Historical Value of HPOTP_PRI_TRB_SL_DR_P is 33.36 +/- 8.60 (3 StdDevs).
Historical Value of HPOTP_PRI_PMP_SL_DR_P is -0.013 +/- 0.686 (3 StdDevs).
Historical Value of HPOTP_BAL_CAV_P_B is 2895.90 +/- 432.84 (3 StdDevs).
Historical Value of HPOTP_BAL_CAV_P_A is 3099.70 +/- 230.79 (3 StdDevs).
Historical Value of HPOTP_INT_SL_PRG_P is 190.84 +/- 12.96 (3 StdDevs).

Value at 104% Nominal LOX NPSP for HPOTP_BAL_CAV_P_B is 3022.69.
Value at 104% Nominal LOX NPSP for HPOTP_BAL_CAV_P_A is 3137.48.
Value at 104% Min LOX NPSP for HPOTP_BAL_CAV_P_B is 3004.09.
Value at 104% Min LOX NPSP for HPOTP_BAL_CAV_P_A is 3073.93.
Max Value After Thermal Equilibrium for HPOTP_PRI_PMP_SL_DR_T is 430.63.
5-to-Cut Value for HPOTP_PRI_PMP_SL_DR_P is -0.119.
Start Value for HPOTP_INT_SL_PRG_P is 193.88.
Equilibrium Value for HPOTP_PRI_TRB_SL_DR_P is 7.76.
Equilibrium Value for HPOTP_SEC_TRB_SL_CAV_P is 12.02.
Peak Time for HPOTP_PRI_TRB_SL_DR_P is 10.0.
Peak Width for HPOTP_PRI_TRB_SL_DR_P is 23.60.
Peak Height for HPOTP_PRI_TRB_SL_DR_P is 37.38.
Peak Time for HPOTP_SEC_TRB_SL_CAV_P is 12.0.
Peak Width for HPOTP_SEC_TRB_SL_CAV_P is 21.74.
Peak Height for HPOTP_SEC_TRB_SL_CAV_P is 24.31.

*** Determining time intervals to analyze.

*** Diagnosing.

*** Preparing anomaly descriptions.

*** Writing data.

----------POSTULATE----------
NAME:              post_A20547_HPOTP_10
POST_NUMBER:       10
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          1
PROBLEM:           HPOT Discharge Temp (518) disqualified. Does not exist.
RULE:              gen32
TYPE:              INSTRUMENTATION

----------POSTULATE----------
NAME:              post_A20547_HPOTP_9
POST_NUMBER:       9
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          1
PROBLEM:           HPOT Discharge Temp (521) disqualified. Does not exist.
RULE:              gen33
TYPE:              INSTRUMENTATION
```

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_8
POST_NUMBER:       8
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          1
PROBLEM:           HPOT Discharge Temp (519) disqualified. Does not exist.
RULE:              gen34
TYPE:              INSTRUMENTATION

---------------POSTULATE------------
NAME:              post_A20547_HPOTP_7
POST_NUMBER:       7
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          1
PROBLEM:           HPOT Discharge Temp (522) disqualified. Does not exist.
RULE:              gen35
TYPE:              INSTRUMENTATION

---------------POSTULATE------------
NAME:              post_A20547_HPOTP_6
POST_NUMBER:       6
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          8
PROBLEM:           Seen between T=71.00 and 121.00. Possible rotor drag.
RULE:              ARotorDrag
TYPE:              ANOMALIES
START TIME:        71.0
END TIME:          121.0

---------------POSTULATE------------
NAME:              post_A20547_HPOTP_5
POST_NUMBER:       5
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          6
PROBLEM:           Failed HPOTP GreenRun LPOTP inlet criteria 3.5.1.2(c). (Maximum NPSP
                      of 150+10/-0 for 10 seconds at 104% or higher.)
RULE:              G3512c
TYPE:              ANOMALIES
START TIME:        0.0
END TIME:          419.7998046875

---------------POSTULATE------------
NAME:              post_A20547_HPOTP_4
POST_NUMBER:       4
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          6
PROBLEM:           Seen at T=409.54. Failed HPOTP GreenRun 109% limits for turbine
                   Delta-T. Probable cause is cold turbine temperature (below 1300.0).
RULE:              G3513DT
TYPE:              ANOMALIES
START TIME:        409.5400085449219
END TIME:          410.0
```

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_3
POST_NUMBER:       3
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          6
PROBLEM:           Seen at T=115.66. Failed HPOTP GreenRun 104% limits for turbine
                      Delta-T. Turbine temperature is not cold.
RULE:              G3513DT
TYPE:              ANOMALIES
START TIME:        115.6599807739258
END TIME:          115.7799758911133

---------------POSTULATE------------
NAME:              post_A20547_HPOTP_2
POST_NUMBER:       2
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          6
PROBLEM:           Seen between T=18.00 and 25.00. Failed HPOTP GreenRun 104% limits
                   for turbine Delta-T. Probable cause is cold turbine temperature
                   (below 1300.0).
RULE:              G3513DT
TYPE:              ANOMALIES
START TIME:        18.0
END TIME:          25.0

---------------POSTULATE------------
NAME:              post_A20547_HPOTP_1
POST_NUMBER:       1
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          8
PROBLEM:           Seen at T=56.34. PBP bistability at thrust level 63.
RULE:              A5.07.1
TYPE:              ANOMALIES
START TIME:        56.34000778198242
END TIME:          56.34000778198242

---------------POSTULATE------------
NAME:              post_A20547_HPOTP_0
POST_NUMBER:       0
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          6
PROBLEM:           Seen between T=68.00 and 358.00. Intermediate seal purge pressure
                   appears erratic or spiking. Possible nose seal leakage, helium
                   supply problem or sensor anomaly. Slight possibility of rubbing.
RULE:              A5.18.1
TYPE:              ANOMALIES
START TIME:        68.0
END TIME:          358.0

*** Wrapping up.
```

SSME HPOTP
Post-Test Diagnostic System
Enhancement Project

Final Report
Attachment #4

Example Output of System to Generate Supporting Plots

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_6
POST_NUMBER:       6
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          18
PROBLEM:           Seen between T=71.00 and 121.00. Possible rotor drag.
RULE:              ARotorDrag
TYPE:              ANOMALIES
START TIME:        71.0
END TIME:          121.0
---------------PLOT_INFO------------
NAME: post_A20547_HPOTP_6
POST_NUMBER: 6
MODULE: HPOTP
CUR_TESTID: A20547
SUBTITLE1: A20547
YTITLE1: Gauge Pressure (psig)
PID1: 327
WHICH_TEST1: 0
LEG_LABEL1: 327
PID2: 328
WHICH_TEST2: 0
LEG_LABEL2: 328
TITLE1: HPOTP Balance Piston Pressures
FULL_SAMPLE1: 1
NUM_CURVES1: 2
XTITLE1: Time (seconds)
START_TIME1: 66.0
END_TIME1: 81.0
SUBTITLE2: A20547
YTITLE2: Gauge Pressure (psig)
PID3: 327
WHICH_TEST3: 0
LEG_LABEL3: 327
PID4: 328
WHICH_TEST4: 0
LEG_LABEL4: 328
TITLE2: HPOTP Balance Piston Pressures
FULL_SAMPLE2: 0
NUM_CURVES2: 2
XTITLE2: Time (seconds)
SUBTITLE3: A20547
YTITLE3: MCC PC (psia)
PID5: 63
WHICH_TEST5: 0
LEG_LABEL5: 63
TITLE3: Thrust Profile
FULL_SAMPLE3: 0
NUM_CURVES3: 1
XTITLE3: Time (seconds)
NUM_PLOTS: 3
```

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_5
POST_NUMBER:       5
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          16
PROBLEM:           Failed HPOTP GreenRun LPOTP inlet criteria 3.5.1.2(c). (Maximum NPSP
                      of 150+10/-0 for 10 seconds at 104% or higher.)
RULE:              G3512c
TYPE:              ANOMALIES
START TIME:        0.0
END TIME:          419.7998046875
---------------PLOT_INFO------------
NAME: post_A20547_HPOTP_5
POST_NUMBER: 5
MODULE: HPOTP
CUR_TESTID: A20547
SUBTITLE1: A20547
YTITLE1: Pressure (psia)
PID1: 858
WHICH_TEST1: 0
LEG_LABEL1: 858
PID2: 859
WHICH_TEST2: 0
LEG_LABEL2: 859
PID3: 860
WHICH_TEST3: 0
LEG_LABEL3: 860
TITLE1: Engine Ox Inlet Pressure
FULL_SAMPLE1: 0
NUM_CURVES1: 3
XTITLE1: Time (seconds)
SUBTITLE2: A20547
YTITLE2: MCC PC (psia)
PID4: 63
WHICH_TEST4: 0
LEG_LABEL4: 63
TITLE2: Thrust Profile
FULL_SAMPLE2: 0
NUM_CURVES2: 1
XTITLE2: Time (seconds)
NUM_PLOTS: 2
```

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_4
POST_NUMBER:       4
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          16
PROBLEM:           Seen at T=409.54. Failed HPOTP GreenRun 109% limits for turbine
                   Delta-T. Probable cause is cold turbine temperature (below 1300.0).
RULE:              G3513DT
TYPE:              ANOMALIES
START TIME:        409.5400085449219
END TIME:          410.0
---------------PLOT_INFO------------
NAME: post_A20547_HPOTP_4
POST_NUMBER: 4
MODULE: HPOTP
CUR_TESTID: A20547
SUBTITLE1: A20547
PID1: 233
WHICH_TEST1: 0
LEG_LABEL1: 233
PID2: 234
WHICH_TEST2: 0
LEG_LABEL2: 234
TITLE1: HPOT Discharge Temps
FULL_SAMPLE1: 0
NUM_CURVES1: 2
XTITLE1: Time (seconds)
START_TIME1: 404.5400085449219
END_TIME1: 414.5400085449219
SUBTITLE2: A20547
PID3: 1190
WHICH_TEST3: 0
LEG_LABEL3: 1190
TITLE2: HPOTP Primary Turbine Seal Drain Temp
FULL_SAMPLE2: 0
NUM_CURVES2: 1
XTITLE2: Time (seconds)
START_TIME2: 404.5400085449219
END_TIME2: 414.5400085449219
SUBTITLE3: A20547
PID4: 63
WHICH_TEST4: 0
LEG_LABEL4: 63
TITLE3: Thrust Profile
FULL_SAMPLE3: 0
NUM_CURVES3: 1
XTITLE3: Time (seconds)
NUM_PLOTS: 3
```

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_3
POST_NUMBER:       3
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          16
PROBLEM:           Seen at T=115.66. Failed HPOTP GreenRun 104% limits for turbine
                      Delta-T. Turbine temperature is not cold.
RULE:              G3513DT
TYPE:              ANOMALIES
START TIME:        115.6599807739258
END TIME:          115.7799758911133
---------------PLOT_INFO------------
NAME: post_A20547_HPOTP_3
POST_NUMBER: 3
MODULE: HPOTP
CUR_TESTID: A20547
SUBTITLE1: A20547
PID1: 233
WHICH_TEST1: 0
LEG_LABEL1: 233
PID2: 234
WHICH_TEST2: 0
LEG_LABEL2: 234
TITLE1: HPOT Discharge Temps
FULL_SAMPLE1: 0
NUM_CURVES1: 2
XTITLE1: Time (seconds)
START_TIME1: 110.6599807739258
END_TIME1: 120.6599807739258
SUBTITLE2: A20547
PID3: 1190
WHICH_TEST3: 0
LEG_LABEL3: 1190
TITLE2: HPOTP Primary Turbine Seal Drain Temp
FULL_SAMPLE2: 0
NUM_CURVES2: 1
XTITLE2: Time (seconds)
START_TIME2: 110.6599807739258
END_TIME2: 120.6599807739258
SUBTITLE3: A20547
PID4: 63
WHICH_TEST4: 0
LEG_LABEL4: 63
TITLE3: Thrust Profile
FULL_SAMPLE3: 0
NUM_CURVES3: 1
XTITLE3: Time (seconds)
NUM_PLOTS: 3
```

```
--------------POSTULATE------------
NAME:              post_A20547_HPOTP_2
POST_NUMBER:       2
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          16
PROBLEM:           Seen between T=18.00 and 25.00. Failed HPOTP GreenRun 104% limits
                       for turbine Delta-T. Probable cause is cold turbine temperature
                       (below 1300.0).
RULE:              G3513DT
TYPE:              ANOMALIES
START TIME:        18.0
END TIME:          25.0
--------------PLOT_INFO------------
NAME: post_A20547_HPOTP_2
POST_NUMBER: 2
MODULE: HPOTP
CUR_TESTID: A20547
SUBTITLE1: A20547
PID1: 233
WHICH_TEST1: 0
LEG_LABEL1: 233
PID2: 234
WHICH_TEST2: 0
LEG_LABEL2: 234
TITLE1: HPOT Discharge Temps
FULL_SAMPLE1: 0
NUM_CURVES1: 2
XTITLE1: Time (seconds)
START_TIME1: 13.0
END_TIME1: 23.0
SUBTITLE2: A20547
PID3: 1190
WHICH_TEST3: 0
LEG_LABEL3: 1190
TITLE2: HPOTP Primary Turbine Seal Drain Temp
FULL_SAMPLE2: 0
NUM_CURVES2: 1
XTITLE2: Time (seconds)
START_TIME2: 13.0
END_TIME2: 23.0
SUBTITLE3: A20547
PID4: 63
WHICH_TEST4: 0
LEG_LABEL4: 63
TITLE3: Thrust Profile
FULL_SAMPLE3: 0
NUM_CURVES3: 1
XTITLE3: Time (seconds)
NUM_PLOTS: 3
```

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_1
POST_NUMBER:       1
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          18
PROBLEM:           Seen at T=56.34. PBP bistability at thrust level 63.
RULE:              A5.07.1
TYPE:              ANOMALIES
START TIME:        56.34000778198242
END TIME:          56.34000778198242
---------------PLOT_INFO------------
NAME: post_A20547_HPOTP_1
POST_NUMBER: 1
MODULE: HPOTP
CUR_TESTID: A20547
SUBTITLE1: A20547
YTITLE1: Pressure (psia)
PID1: 59
WHICH_TEST1: 0
LEG_LABEL1: 59
TITLE1: Preburner Pump Discharge Pressure Ch B
FULL_SAMPLE1: 1
NUM_CURVES1: 1
XTITLE1: Time (seconds)
START_TIME1: 51.34000778198242
END_TIME1: 61.34000778198242
SUBTITLE2: A20547
YTITLE2: PCNT
PID2: 176
WHICH_TEST2: 0
LEG_LABEL2: 176
TITLE2: OPOV Command
FULL_SAMPLE2: 1
NUM_CURVES2: 1
XTITLE2: Time (seconds)
START_TIME2: 51.34000778198242
END_TIME2: 61.34000778198242
SUBTITLE3: A20547
YTITLE3: MCC PC (psia)
PID3: 63
WHICH_TEST3: 0
LEG_LABEL3: 63
TITLE3: Thrust Profile
FULL_SAMPLE3: 1
NUM_CURVES3: 1
XTITLE3: Time (seconds)
START_TIME3: 51.34000778198242
END_TIME3: 61.34000778198242
NUM_PLOTS: 3
```

```
---------------POSTULATE------------
NAME:              post_A20547_HPOTP_0
POST_NUMBER:       0
MODULE:            HPOTP
TEST_ID:           A20547
PRIORITY:          16
PROBLEM:           Seen between T=68.00 and 358.00. Intermediate seal purge pressure
                      appears erratic or spiking. Possible nose seal leakage, helium
                      supply problem or sensor anomaly. Slight possibility of rubbing.
RULE:              A5.18.1
TYPE:              ANOMALIES
START TIME:        68.0
END TIME:          358.0
---------------PLOT_INFO------------
NAME: post_A20547_HPOTP_0
POST_NUMBER: 0
MODULE: HPOTP
CUR_TESTID: A20547
SUBTITLE1: A20547
YTITLE1: Pressure (psia)
PID1: 211
WHICH_TEST1: 0
LEG_LABEL1: 211
PID2: 212
WHICH_TEST2: 0
LEG_LABEL2: 212
TITLE1: HPOP Int Seal Purge Pr
FULL_SAMPLE1: 0
NUM_CURVES1: 2
XTITLE1: Time (seconds)
SUBTITLE2: A20547
PID3: 937
WHICH_TEST3: 0
LEG_LABEL3: 937
TITLE2: Engine Helium Interface Pr
FULL_SAMPLE2: 0
NUM_CURVES2: 1
XTITLE2: Time (seconds)
SUBTITLE3: A20547
YTITLE3: Pressure (psia)
PID4: 92
WHICH_TEST4: 0
LEG_LABEL4: 92
PID5: 93
WHICH_TEST5: 0
LEG_LABEL5: 93
PID6: 951
WHICH_TEST6: 0
LEG_LABEL6: 951
PID7: 952
WHICH_TEST7: 0
LEG_LABEL7: 952
PID8: 953
WHICH_TEST8: 0
LEG_LABEL8: 953
TITLE3: Adjacent Pressures (Sec Trb Sl Cav P & Pri Pump Sl Dr P)
FULL_SAMPLE3: 0
NUM_CURVES3: 5
XTITLE3: Time (seconds)
NUM_PLOTS: 3
```

SSME HPOTP
Post-Test Diagnostic System
Enhancement Project

Final Report
Attachment #5

TKCLIPS User Function Summary

Command-line CLIPS 5.1 with integrated TekBase, SSME datafile, and feature extraction interfaces.

T.W. Bickmore, Aerojet 1/18/94

The following can be used either at the command line or on the right hand side of rules.

TekBase Interface Commands

Limitations

- Support is only available for the following TekBase data types: SHORT, LONG, REAL, STRING, BOOL, and CHAR. The data types DATE, TIME, and COMPLEX are currently unsupported (attempts to import from these fields result in the value "???" getting asserted; attempts to export to these fields will result in a TekBase error).
- Support is only available for the following CLIPS data types: STRING, SYMBOL, INTEGER, and FLOAT. The data types EXTERNAL_ADDRESS and INSTANCE are currently unsupported. Attempts to export these values to TekBase will result in an error.
- An attempt is made to provide appropriate type conversions between CLIPS and TekBase where feasible, but not every combination is supported. The "standard" conversions are the following:

TekBase to CLIPS

TekBase	CLIPS
SHORT	INTEGER
LONG	INTEGER
REAL	FLOAT
STRING	STRING
BOOL	INTEGER
CHAR	STRING

- The Sun environment variable 'QYHOST' must be set to the name of the TekBase server.

Commands

(DB_connect) – Attempts to connect to the TekBase server. Returns logical TRUE if successful, FALSE otherwise. Note: This does NOT open a database.

(DB_disconnect) – Disconnects from the TekBase server.

(DB_exec <commandstring>) – Executes the specified command string. Only TQL commands which do not involve the transfer of data between the server and client may be executed with this function. Supported commands include:

CREATE	GRANT	JOIN
DELETE	GROUP	LOCK
DOCUMENT	IMPORT	MERGE
DROP	INDEX	OPEN
EXPORT	INTERSECT	RENAME

REPORT	SET	UNUSE
RUN	SORT	UPDATE
SAVE	SUBTRACT	USE
SELECT	UNION	
SEQUENCE	UNLOCK	

(DB_get <tablename> <columnnames> <conditionstring>) – Retrieves data from TekBase, asserting each retrieved row as a fact in CLIPS. The format of the asserted facts is

(<tablename> <column1value> <column2value> …)

where <tablename> is always a symbol, and the column values are converted as specified above. <columnnames> must be a multi-valued CLIPS field, each value of which must be a symbol or a string specifying the name of a column/field in the specified table (note: if only a single column of input is needed, the columnname need not be multi-valued). <conditionstring> is a string containing a legal TQL select condition (e.g. "ATOMIC# > 3"). Note that string values in the condition string must be escaped (e.g., "COLOR = \"RED\""). DB_get returns logical TRUE if it successfully imports at least one fact/row, FALSE otherwise.

(DB_put <tablename> <columnnames> <values>) - Adds a row of values to the specified table in TekBase. <tablename> can be either a symbol or a string. <columnnames> must be a multi-valued CLIPS field, each value of which must be a symbol or a string specifying the name of a column/field the specified table (note: if only a single column of the table needs to be output, the columnname need not be multi-valued). <values> must be a multi-valued CLIPS field (unless only a single column of data is being written). The values in this third argument are written to the specified columnnames. If the number of columnnames and values disagree, or if a conversion between a CLIPS value and the corresponding TekBase column cannot be performed an error will result. DB_put returns logical TRUE if a row is successfully added to the specified table, FALSE otherwise.

Environment Access Commands

(get_param) -- Returns the the 1st command line parameter used when CLIPS was invoked (if it is not a '-flag'), otherwise NIL.

(get_env <variable>) -- Returns the value of the specified variable in the Sun environment, or "".

SSME DataFile Access and Feature Extraction Commands

The following all attempt to access SSME data flatfiles directly and return their results as asserted facts. All return 1 if successful, 0 if any error occured in processing. The environment variable 'NASA_TEST_DATA' must define a colon-separated list of paths to search for the datafiles (e.g., setenv NASA_TEST_DATA /data1:/data2:/hd1:/hd2). All input and output parameters are floats, except as specified below:

<descr>, <units>, <testid>, <pid>	String values.
<label>	Symbol values.
<windowsize>, <minPts>, <smoothWinSiz>, <stepSize>, <thrust_level>	Fixed values.
<withinErrorBars>, <DiffbyOffset>	"True" | "False".
<checkType>	"either pid" | "difference"
<limitType>	"upper" | "lower"
<dataType>	0 (Full Sample) | 1 (1sec Averaged) | 2+ (Smoothed, value=windowsize)

Information about a sensor:

```
(DATA_info <testid> <pid>)
  -> (PIDINFO <testid> <pid> <start> <end> <descr> <units>
  <rate> <shutdown>)
```

Import raw data:

```
(DATA_get <testid> <pid> <start> <end> <label>)
   -> (DATA <label> <val> <sensor_reading>*)
   -> (TIME <label> <val> <time_value>*)
```

Import averaged data:

```
(DATA_average <testid> <pid> <start> <end> <label>)
   -> (DATA <label> <val>...)
      (TIME <label> <val>...)
      (STDDEV <label> <val>...)
```

Import smoothed data:

```
(DATA_smooth <testid> <pid> <start> <end> <windowsize>
  <label>)
   -> (DATA <label> <val>...)
      (TIME <label> <val>...)
```

Compute statistical summary:

```
(DATA_stats <testid> <pid> <start> <end>)   ;;Full sample only
   -> (STATS <testid> <pid> <start> <end> <mean> <stddev>
  <min> <max>)
```

Fit a line through a signal:

```
(DATA_FitLine <testid> <pid> <dataType> <start> <end>)
   -> (LINE <testid> <pid> <start> <end> <offset> <slope>)
```

Determine thrust profile:

```
(DATA_FindConstantThrust <testid>)
   -> (F_THLEDE <testid> <start> <end> <thrust_level>)
```

```
Find erratic features:
(DATA_FindErratic <testid> <pid> <start> <end>
  <absStdDevThresh>)
   -> (F_ERRAT <testid> <pid> <start> <end>)

Find level shift features:
(DATA_FindLevelShift <testid> <pid> <dataType> <start> <end>
  <pidRange>)
   -> (F_LEVSH <testid> <pid> <start> <end> <lastmag> <delta>)

Find peak features:
(DATA_FindPeak <testid> <pid> <dataType> <start> <end>
     <minPeak> <minPts>)
   -> (F_PEAK <testid> <pid> <time> <peak> <width(fwhm)>
     <offset>)

Find different than features:
(DATA_FindDifferentThan <testid1> <pid1> <start1> <end1>
  <testid2> <pid2> <start2> <end2>
                          <dataType> <numSigmasThresh>)
   -> (F_DIFTHA <testid1> <pid1> <start1> <end1> <testid2>
  <pid2>
                    <withinErrorBars> <DiffbyOffset> <offset>
  <offset_sigma>)

Compute piece-wise linear model:
If <directional_only> is 0, then any signficant change in slope is
  noted. Otherwise, only changes through zero are noted.
(DATA_FindPieceWise <testid> <pid> <dataType> <start> <end>
     <stepSize> <threshold> <directional_only>)
   -> (SEGMENT <testid> <pid> <start> <end> <startval>
     <endval>)

Compute piece-wise linear model of the difference of two signals:
If <directional_only> is 0, then any signficant change in slope is
  noted. Otherwise, only changes through zero are noted.
(DATA_DeltaPieceWise <testid1> <pid1> <start1> <end1>
  <testid2> <pid2> <start2> <end2>
       <dataType> <stepSize> <threshold> <directional_only>)
   -> (DSEGMENT <testid1> <pid1> <testid2> <pid2> <start>
         <end> <startval> <endval>)

Find spike features:
<perCent> is portion of <Range> to use as a threshold.
(DATA_FindSpike <testid> <pid> <start> <end> <bitToggle>
  <Range> <perCent>) ;;Full sample only
   -> (F_SPIKE <testid> <pid> <start> <end> <magnitude>)

SAIC method for Bistability:
Only call for PLs 65% or below.
(DATA_DetectBistability <testid> <start> <end>)
   -> (F_BISTAB <testid> <start> <end>)

New, improved method for Bistability:
(DATA_DetectGreenBistability <testid> <start> <end>
     "209"|"210"  "140"|"141")
   -> (F_GREENBISTAB <testid> <start> <end>)
```

```
Check hard limits:
<is_max> = 1 indicates that threshold is on maximum deviation, else
  minimum.
<minPts> is minimum number of consecutive data points which must exceed
  limit before report.
(DATA_CheckUpperLimit <testid> <pid> <start> <end> <limit>
      <minPts>)
(DATA_CheckLowerLimit <testid> <pid> <start> <end> <limit>
      <minPts>)
(DATA_CheckDifference <testid> <pid> <pid2> <start> <end>
     <limit> <minPts> <is_max>)
   -> (F_RLVIOL <testid> <pid> {<pid2>|nil} <start> <end>
          <checkType> <limitType> <redline>)

Tell when a PID is within specified bounds:
(DATA_FindInRange <testid> <pid> <start> <end> <low> <high>
  <minPts> <label>)
   -> (F_INRANGE <testid> <pid> <start> <end> <label>)

Find IsFlat features:
(DATA_IsFlat <testid> <pid> <dataType> <start> <end>
     <NumSigmasForFlat>)
   -> (F_ISFLAT <testid> <pid> <start> <end> <offset> <slope>
          <offsetsig> <slopesig>)

Find DeltaLevelShift features:
(DATA_DeltaLevelShift <testid1> <pid1> <start1> <end1>
     <testid2> <pid2> <start2> <end2> <dataType>)
   -> (F_LEVSH <testid> <pid1-pid2> <start> <end> <lastmag>
          <delta>)

Find DeltaDifferentThan features:
(DATA_DeltaDifferentThan <testid1> <pid1a> <pid1b> <start1>
  <end1> <testid2> <pid2a> <pid2b> <start2> <end2> <dataType>
  <numSigmasThresh>)
   -> (F_DIFTHA <testid1> <pid> <start1> <end1> <testid2>
          <compPid2> <withinErrorBars> <DiffbyOffset>
          <offset> <offset_sigma>)

Find Noise features:
(DATA_FindNoise <testid> <pid> <start> <end> <sigmaThresh>)
  -> (F_NOISE <testid> <pid> <start> <end>)
```

SSME HPOTP
Post-Test Diagnostic System
Enhancement Project

Final Report
Attachment #6

Data Dictionary

HPOTP Diagnostic System
Data Dictionary

Documentation of All Global Data Structures Used in the System
T.W. Bickmore, Aerojet 1994

Globals

(defglobal ?*DB_enabled* = FALSE)
Controls whether connection with TekBase should even be attempted.

(defglobal ?*DB_features* = TRUE)
Controls whether features are imported from TekBase (else computed).

(defglobal ?*interactive* = TRUE)
Controls whether I/O is configured for a user sitting at the terminal, or for batch mode.

(defglobal ?*DB_output* = TRUE)
Controls whether result hypotheses should be written to TekBase.

(defglobal ?*DEBUG* = TRUE)
Causes diagnostic info to be printed to stdout.

(defglobal ?*DB_connected* = FALSE)
Indicates whether a connection to TekBase has been established.

Fact Formats

(comparison_interval <current_testid> <start1> <end1><comparison_testid> <start2> <end2>)
Describes times of corresponding thrust between two tests, for comparison.

(comparison_test <testid>)
Used in all modules. Describes the comparison test ID (if any). This is always asserted with some value during HPOTP_plot execution.

(confirmed <relation> <testid> <pid> <start_time> <other_arg>*)
(unconfirmed <relation> <testid> <pid> <start_time> <other_arg>*)
(spurious <relation> <testid> <pid> <start_time> <other_arg>*)
Used in most modules. States results of redundancy voting on features.

(current_anomaly <postulate>)
Used by HPOTP_plot. Describes the current postulate being worked on.

(current_phase <phase>)
(phases <phase>*)
Used in all modules. Describes the current phase of processing.
Current phases (and their sequence) are:

initialize	Get test IDs and profiles.
SVAL_hard_failures	Check for hard sensor failures.
SVAL_soft_failures	Determine preferred sensors.
get_features	Determine diagnostic features.
find_event_intervals	Determine time intervals to analyze.
find_anomalies	Diagnosis.
prepare_output	Prepare anomaly and plot descriptions.
output_anomalies	Output result.
wrapup	Disconnect, cleanup.

(CURRENT_STAT <testid> <type> <param> <value>)
Used in HPOTP_stats, HPOTP_anom.Describes the value of a family-tracked parameter for the current test. Type can currently be one of:
5_TO_CUT MAX_AFTER_EQ 104_MIN_NPSP 109_MAX_NPSP 104_NOM_NPSP
PEAK_HEIGHT, PEAK_WIDTH, PEAK_TIME, EQ_VAL

(current_test <testid>)
Used in all modules. Describes the current test ID.

(current_time <testid> <time>)
Temporary fact used by HPOTP_event.

(emit_plot_info <postulate> <field> <value>)
Used by HPOTP_plot. Specified a single field of PLOTINFO for output.

(event_interval <testid> <start> <end>
<power_level>|TRANSIENT)
Used in HPOTP_event, HPOTP_anom. Describes a time interval within which nothing significant happens.

(event_time <testid> <time>)
Used by HPOTP_event. Declares the time of a 'significant' event with which to partition the test.

(FAMILY_tally <type> <parameter> <N> <sum> <sum-of-squares>)
Used by HPOTP_stat. Temporary fact used by HPOTP_stat.

(FAMILY_STAT <type> <parameter> <mean> <stddev> <N>)
Used by HPOTP_stat. Summarizes the mean and standard deviation of a parameter.

(FeatureRelations <relation>*)
Used by HPOTP_features.Defines the relations of all feature-facts.

(GetFeature <featureClass> [<testid>])
Used by most modules, acted on by HPOTP_features. Causes features to be either imported from TekBase or computed.
Current classes are:
ComparisonDifferences, BalancePistonShifts, DrainPeaks,
FamilyChecks, GreenRun, IsFlat, Bistable, TestIDs
ConstantThrust, SensorAnomaly, DataAvailability, SensorRedundants

(GREEN_limit <parameter> <type> <min>|nil <max>|nil)
Describes a greenrun spec limit.
<type> can be one of:
START, 104_MIN_NPSP, 109_MAX_NPSP, 104_MAX_NPSP, 109_MIN_NPSP, PEAK

(IO_table_fields <table> <type> <field>*)
Used in HPOTP_IO. Describes the types of all fields in a TekBase table.

(IO_output <table> <field>* %VALUES% <value>*)
Used in HPOTP_IO. Describes a row of data to be output to TekBase. (When a fact of this form is asserted, any unspecified fields are given default values, and then the row is output to TekBase).

(IO_update_DB yes|no)
Used in HPOTP_IO. Response from user as to whether HISTORY should be updated.

(linear_behavior <testid> <start> <end> <power_level> <LOXinletAtStart> <LOXinletAtEnd>)
Defines a period of constant thrust and linear LOX inlet pressure.

(LoadedFeatures <testid>)
Used by HPOTP_features. States whether features have already been loaded from TekBase.

(LOXInletP <testid> <PID>*)
Used by HPOTP_features, and others. Declares which PIDs are used to determine the LOX vent profile.

(num_postulates <N>)
Used by HPOTP_plot. Describes the number of postulates output.

(num_plots <N>)
Used by HPOTP_plot. Describes the number of plots output for the current anomaly.

(PID_ytitle <PID>* <title>)
Used by HPOTP_plot. Defines the YTITLE fields to use for plots.

(plot_ditto <new_class> <old_class>)
Used in HPOTP_plot, HPOTP_anom, HPOTP_greenrun, HPOTP_sval States that all anomalies of new_class should have the same supporting plots as those for old_class.

(plot_info_slots <name> <slot>*)
(plot_info_values <name> <value>*)
Used by HPOTP_plot, HPOTP_IO. Describes values to be output to PLOTINFO for a particular anomaly. Slots and values correspond by position.

(prepare_plot <N>)
Used by HPOTP_plot. Control fact describing the current plot being generated.

(pump_equlibrium_interval <testid> <start> <end>)
Used by most modules. Declares the steady-state time interval immediately following pump thermal equilibrium.

(pump_equilibrium_time <time>)
Defines the minimum time for pump thermal equilibrium.

(RED_redundancy_check <relation>*)
Used by HPOTP_redundancy. Defines the feature relations which will be redundancy voted.

(sensor_map <external_PID> <internal_PID>)
Used by HPOTP_features. Defines a translation to be performed on any field of a feature-fact.

(sensor_status <testid> <PID> <parameter> valid|invalid|preferred)
Used by most modules. Declares the current status of a sensor data channel.

(shutdown_time <testid> <shutdown>)
Used by most modules. Declares the best guess of the time of the engine shutdown command.

(time_at <power_level> <duration>)
Used by HPOTP_greenrun. Describes the total time spent at the specified duration.

(transducer_status <testid> <param> <label> valid|invalid <pid1> [<pid2>])
Defines the status of a transducer with one or two bridges (pid1 and pid2).

(turbine_equlibrium_interval <testid> <start> <end>)
Used by most modules. Declares the steady-state time interval immediately following turbine thermal equilibrium.

(turbine_equlibrium_time <time>)
Defines the minimum time for turbine thermal equilibrium.

In addition to the above, the following TekBase tables are imported:
HISTORY
F_THLEDE
F_DIFTHA
F_LEVSH
F_ERRAT
F_SPIKE
F_ISFLAT
F_BISTAB
F_PEAK
F_RLVIOL
F_NOISE
POSTUL
PIDINFO
TESTINFO

Finally, results of many CLIPS user functions may be asserted as facts. See Attachment 5 for details.

DefTemplates

```
(deftemplate  anomaly
  (field  class)
  (field  type            (default  ANOMALIES))
  (field  start           (default  nil))
  (field  end             (default  nil))
  (field  priority        (default  -1))
  (field  description     (default  "HPOTP  anomaly."))
  (field  postnum)
  (field  name)
  (field  append_time     (default  TRUE))
  (field  repeatable      (default  TRUE)))
```

```
Defines an anomaly for output to TekBase.
Class is simply a label for the type of anomaly.
Type ::= ANOMALIES | OBSERVATION | INSTRUMENTATION
Start, End ::= Start and end times (or nil).
Priority, as per POSTUL table.
Description is a textual description.
Postnum is a unique number for a specific anomaly (filled in by
  HPOTP_plot).
Name is a unique label required by ehms (filled in by HPOTP_plot).
Append_time is a Boolean flag. If TRUE, 'Seen at T=<start>.' is added to
  the front of description before the anomaly is output.
Repeatable is a Boolean flag. If FALSE, only one anomaly of this class
  will be reported per test.
```

```
(deftemplate  plot
  (field  class)
  (field  number)
  (field  use_comparison      (default  FALSE))
  (field  cross_comparison    (default  FALSE))
  (field  full_sample         (default  FALSE))
  (field  choose_redundant    (default  FALSE))
  (field  anomaly_delta_start)
  (field  anomaly_delta_end)
  (field  shutdown_delta_end)
  (field  end)
  (field  title               (default  "???"))
  (multi-field  PIDs))
```

Defines a plot to support a class of anomaly. Up to three plots per anomaly class can be defined (indexed by number).
Class ties this to the anomaly with the same value.
Number is the plot number (1, 2, or 3).
PIDs is the list of PID numbers to be shown on the plot.
Title is the title of the plot (X and Ytitles are automatically generated).
Choose_redundant is a Boolean flag. If TRUE, only one of the PIDs will be plotted.
Use_comparison is a Boolean flag. If TRUE, data for the comparison test only will be plotted, otherwise data for the current test only will be used.
Cross_comparison is a Boolean flag. If TRUE, all PIDs will be plotted for both the current test and the comparison test on the same plot.
Full_sample is a Boolean flag. If TRUE, data will be initially displayed at full sample rate, otherwise one-second average data will be displayed.
The plot time interval defaults to START to SHUTDOWN unless the following are given:
end -- Fixes the end time to the specified time.
shutdown_delta_end -- Fixes the end time relative to SHUTDOWN.
anomaly_delta_start -- Fixes the plot start time relative to the anomaly start time.
anomaly_delta_end -- Fixes the plot end time relative to the anomaly end time.

```
(deftemplate  sensor
  (field  parameter)
  (field  desc)
  (field  range)
  (field  bit_toggle)
  (field  stddev)
  (field  reasonable_limit)
  (field  comparison_threshold)
  (field  cross_test_threshold)
  (field  erratic_threshold)
  (field  ambient_upper_limit)
  (field  ambient_lower_limit)
  (field  is_flat_at_ambient       (default  FALSE))
  (field  spike_factor             (default  0.07))
  (field  should_not_be_flat       (default  TRUE))
  (multi-field  PIDs))
```

Defines a measured parameter.
Parameter is a string representation.
Desc is used for outputs to users.
Range is normal operating range.
The following control which sensor validation algorithms are run:
reasonable_limit, comparison_threshold, cross_test_threshold,
erratic_threshold, ambient_upper_limit, ambient_lower_limit,
spike_factor, should_not_be_flat

Salience Bands

10,000	Immediate-action utilities.
2,000	Sensor validation hard failure detection.
1,000	Feature extraction.
900	Redundancy management.
800	Historical statistics (collection, extraction, comparison).
0	Default (e.g., most anomaly detection rules).
-10,000	Change Phase

SSME HPOTP
Post-Test Diagnostic System
Enhancement Project

Final Report
Attachment #7

CLIPS Program Listings

FILE: HPOTP_batch.bat
CLIPS batch file to load and run HPOTP diagnostic system.

```
(unwatch all)
(load "HPOTP_global.clp")
(load "HPOTP_exec.clp")
(load "HPOTP_features.clp")
(load "HPOTP_redundancy.clp")
(load "HPOTP_stats.clp")
(load "HPOTP_sval.clp")
(load "HPOTP_event.clp")
(load "HPOTP_anom.clp")
(load "HPOTP_greenrun.clp")
(load "HPOTP_plot.clp")
(load "HPOTP_IO.clp")
(set-strategy breadth)
(reset)

(bind ?*DEBUG* TRUE)
(bind ?*DB_enabled* TRUE)
(bind ?*interactive* FALSE)
(bind ?*DB_features* TRUE)
(bind ?*DB_output* TRUE)
(run)
```

FILE: HPOTP_global.clp
Global definitions and utility functions.

```
;;; HPOTP Diagnostic Module
;;; Global Definitions & Utilities
;;;
;;; Creation 8/10/93  T.W. Bickmore

;;; -------------------------       GLOBALS       ----------------------

;;; Controls whether connection with TekBase should even be attempted.
(defglobal ?*DB_enabled* = FALSE)

;;; Controls whether features are imported from TekBase (else computed).
(defglobal ?*DB_features* = TRUE)

;;; Controls whether I/O is configured for a user sitting at the terminal,
;;; or for batch mode.
(defglobal ?*interactive* = TRUE)

;;; Controls whether result hypotheses should be written to TekBase.
(defglobal ?*DB_output* = TRUE)

;;; Causes diagnostic info to be printed to stdout.
(defglobal ?*DEBUG* = TRUE)

;;; Indicates whether a connection to TekBase has been established.
(defglobal ?*DB_connected* = FALSE)


;;; ------------------------- TEMPLATES -----------------------------
;;;
;;;

;;; DEFTEMPLATE ANOMALY
;;; Defines an anomaly for output to TekBase.
;;; Class is simply a label for the type of anomaly.
;;; Type ::= ANOMALIES | OBSERVATION | INSTRUMENTATION
;;; Start, End ::= Start and end times (or nil).
;;; Priority, as per POSTUL table.
;;; Description is a textual description.
;;; Postnum is a unique number for a specific anomaly (filled in by HPOTP_plot).
;;; Name is a unique label required by ehms (filled in by HPOTP_plot).
;;; Append_time is a Boolean flag. If TRUE, 'Seen at T=<start>.' is added to
;;;   the front of description before the anomaly is output.
;;; Repeatable is a Boolean flag. If FALSE, only one anomaly of this class
;;;   will be reported per test.
(deftemplate anomaly
  (field class)
  (field type        (default ANOMALIES))  ;;or OBSERVATION or INSTRUMENTATION
  (field start       (default nil))
  (field end         (default nil))
  (field priority    (default -1))
  (field description (default "HPOTP anomaly."))
  (field postnum)
  (field PID)
  (field name)
  (field append_time (default TRUE)) ;;Adds 'Seen at T=X' to string.
  (field repeatable  (default TRUE)))
```

```
;;; DEFTEMPLATE PLOT
;;; Defines a plot to support a class of anomaly. Up to three plots per
;;; anomaly class can be defined (indexed by number).
;;; Class ties this to the anomaly with the same value.
;;; Number is the plot number (1, 2, or 3).
;;; PIDs is the list of PID numbers to be shown on the plot.
;;; Title is the title of the plot (X and Ytitles are automatically generated).
;;; Choose_redundant is a Boolean flag. If TRUE, only one of the PIDs will be
;;;   plotted.
;;; Use_comparison is a Boolean flag. If TRUE, data for the comparison test only
;;;   will be plotted, otherwise data for the current test only will be used.
;;; Cross_comparison is a Boolean flag. If TRUE, all PIDs will be plotted for both
;;;   the current test and the comparison test on the same plot.
;;; Full_sample is a Boolean flag. If TRUE, data will be initially displayed at
;;;   full sample rate, otherwise one-second average data will be displayed.
;;; The plot time intervaldefaultsto START to SHUTDOWN unless the following are given:
;;;  end -- Fixes the end time to the specified time.
;;;  shutdown_delta_end -- Fixes the end time relative to SHUTDOWN.
;;;  anomaly_delta_start--Fixes the plot starttime relative to the anomaly start time.
;;;  anomaly_delta_end -- Fixes the plot end time relative to the anomaly end time.
(deftemplate plot
  (field class)
  (field number)
  (field use_comparison (default FALSE))
  (field cross_comparison (default FALSE))
  (field full_sample (default FALSE))
  (field choose_redundant (default FALSE))
  (field anomaly_delta_start)
  (field anomaly_delta_end)
  (field shutdown_delta_end)
  (field end) ;;absolute
  (field title (default "???"))
  (multi-field PIDs))

;;; DEFTEMPLATE SENSOR
;;; Defines a measured parameter.
;;; Parameter is a string representation.
;;; Desc is used for outputs to users.
;;; Range is normal operating range.
;;; The following control which sensor validation algorithms are run:
;;;   reasonable_limit, comparison_threshold, cross_test_threshold,
;;;   erratic_threshold, ambient_upper_limit, ambient_lower_limit,
;;;   spike_factor, should_not_be_flat
(deftemplate sensor
  (field parameter)
  (field desc)
  (field range)                          ;;Typical range experienced
  (field bit_toggle)
  (field stddev)                  ;;Typical during steady-state
  (field reasonable_limit)        ;;Upper
  (field comparison_threshold)           ;;For same-test redundants
  (field cross_test_threshold)           ;;For cross-test
  (field erratic_threshold)       ;;Number of stddevs
  (field ambient_upper_limit)            ;;Pre- and Post- test Limit check
  (field ambient_lower_limit)
  (field is_flat_at_ambient (default FALSE))
  (field spike_factor (default 0.07))    ;;Percent of range for threshold.
  (field should_not_be_flat (default TRUE))
  (multi-field PIDs))
```

```
;;; ------------------------ GLOBAL STATIC KNOWLEDGE --------------------
;;;

;;; DEFFACTS INIT-CONSTANTS
;;; Time delay intervals to reach thermal equilibrium.
(deffacts init-constants
  (turbine_equilibrium_time 150.0)
  (pump_equilibrium_time    200.0))

;;; DEFFACTS INIT-SENSORS
;;; Information about all PIDs.
(deffacts init-sensors
  (sensor (PIDs "2") (parameter "HPOTP_SPD") (desc "HPOTP Speed"))

  (sensor (PIDs "24") (parameter "MCC_HOT_GAS_INJ_P")
       (desc "MCC Hot Gas Injection Press")
       (range 3500.0) (bit_toggle 3.75) (stddev 6.0) (erratic_threshold 2.0)
        (ambient_upper_limit 40.0) (ambient_lower_limit -5.3))

  (sensor (PIDs "59") (parameter "PBP_DS_P") (desc "PBP Discharge Press")
       (range 7500.0) (bit_toggle 20.5) (stddev 24.0)
        (ambient_upper_limit 130.0) (ambient_lower_limit -5.3)
       (is_flat_at_ambient TRUE))

  (sensor (PIDs "90" "190") (parameter "HPOP_DS_P") (desc "HPOP Discharge Press")
       (range 4500.0) (bit_toggle 15.0) (stddev 10.0) (erratic_threshold 6.0)
        (ambient_upper_limit 130.0) (ambient_lower_limit 60.0))

  (sensor (PIDs "91" "92") (parameter "HPOTP_SEC_TRB_SL_CAV_P")
       (desc "HPOTP Sec Turbine Seal Cavity Press")
       (range 10.0) (bit_toggle 0.66) (stddev 0.6) (comparison_threshold 3.0)
       (cross_test_threshold 3.0) (erratic_threshold 3.0)
        (ambient_upper_limit 18.0) (ambient_lower_limit 12.0))
  (family_check "PEAK_HEIGHT" "HPOTP_SEC_TRB_SL_CAV_P" 3.0 3.0)
  (family_check "PEAK_WIDTH"  "HPOTP_SEC_TRB_SL_CAV_P" 3.0 3.0)
  (family_check "PEAK_TIME"   "HPOTP_SEC_TRB_SL_CAV_P" 3.0 3.0)
  (family_check "EQ_VAL"      "HPOTP_SEC_TRB_SL_CAV_P" 3.0 3.0
       turbine_equilibrium_interval)

  (sensor (PIDs "140" "141") (parameter "OPOV_ACT_POS") (desc "OPOV Actuator Position")
       (bit_toggle 0.08))

  (sensor (PIDs "209" "210") (parameter "HPOP_IN_P") (desc "HPOP Inlet Press")
       (range 350.0) (bit_toggle 1.28) (stddev 1.2) (comparison_threshold 3.0)
       (cross_test_threshold 3.0) (erratic_threshold 4.0)
       (should_not_be_flat FALSE)
        (ambient_upper_limit 130.0) (ambient_lower_limit 40.0))

  (sensor (PIDs "211" "212") (parameter "HPOTP_INT_SL_PRG_P")
       (desc "HPOTP Int Seal Purge Press")
       (range 100.0) (bit_toggle 1.28) (stddev 14.0) (comparison_threshold 3.0)
       (cross_test_threshold 3.0) (erratic_threshold 16.0)
        (ambient_upper_limit 210.0) (ambient_lower_limit -28.0)
       (spike_factor 0.02))
  (family_check "START_VAL"         "HPOTP_INT_SL_PRG_P" 3.0 3.0)

  (sensor (PIDs "233" "234" "518" "521" "519" "522") (parameter "HPOT_DS_T")
       (desc "HPOT Discharge Temp")
       (range 1250.0) (bit_toggle 4.25) (stddev 16.5) (comparison_threshold 3.0)
       (cross_test_threshold 3.0) (erratic_threshold 4.0)
        (ambient_upper_limit 530.0) (ambient_lower_limit 350.0))
  (bridges "HPOT_DS_T" "518" "521")
  (bridges "HPOT_DS_T" "519" "522")
```

```
(sensor (PIDs "327")(parameter "HPOTP_BAL_CAV_P_A")
     (desc "HPOTP Bal Cav Press A")
     (range 3500.0)(bit_toggle 0.595)(stddev 13.5)
      (ambient_upper_limit 170.0)(ambient_lower_limit 0.0))
(family_check "104_MIN_NPSP" "HPOTP_BAL_CAV_P_A" 3.0 3.0)
(family_check "109_MAX_NPSP" "HPOTP_BAL_CAV_P_A" 3.0 3.0)
(family_check "104_NOM_NPSP" "HPOTP_BAL_CAV_P_A" 3.0 3.0)

(sensor (PIDs "328")(parameter "HPOTP_BAL_CAV_P_B")
     (desc "HPOTP Bal Cav Press B")
     (range 3500.0)(bit_toggle 0.595)(stddev 13.5)
      (ambient_upper_limit 170.0)(ambient_lower_limit 0.0))
(family_check "104_MIN_NPSP" "HPOTP_BAL_CAV_P_B" 3.0 3.0)
(family_check "109_MAX_NPSP" "HPOTP_BAL_CAV_P_B" 3.0 3.0)
(family_check "104_NOM_NPSP" "HPOTP_BAL_CAV_P_B" 3.0 3.0)

(sensor (PIDs "858" "859" "860")(parameter "ENG_OX_IN_P")
     (desc "Engine LOX Inlet Press")
     (range 130.0)(bit_toggle 0.25)(stddev 20.1)
     (should_not_be_flat FALSE)
      (ambient_upper_limit 200.0)(ambient_lower_limit 10.0))

(sensor (PIDs "937")(parameter "ENG_HE_INT_P")
     (should_not_be_flat FALSE)
      (desc "Engine Helium Interface Press"))

(sensor (PIDs "951" "952" "953")(parameter "HPOTP_PRI_PMP_SL_DR_P")
     (desc "HPOTP Pri Pump Seal Drain Press")
     (range 5.0)(bit_toggle 0.012)(stddev 0.07)(comparison_threshold 5.0)
     (should_not_be_flat FALSE)
     (cross_test_threshold 5.0)(erratic_threshold 0.08)
      (ambient_upper_limit 34.7)(ambient_lower_limit -5.3))
(family_check "5_TO_CUT" "HPOTP_PRI_PMP_SL_DR_P" 3.0 3.0)

(sensor (PIDs "990")(parameter "HPOTP_PRI_TRB_SL_DR_P")
     (desc "HPOTP Pri Turbine Seal Drain Press")
     (range 30.0)(bit_toggle 0.012)(stddev 2.7)(cross_test_threshold 3.0)
     (erratic_threshold 1.0)
      (ambient_upper_limit 34.7)(ambient_lower_limit -5.3))
(family_check "PEAK_HEIGHT" "HPOTP_PRI_TRB_SL_DR_P" 3.0 3.0)
(family_check "PEAK_WIDTH"  "HPOTP_PRI_TRB_SL_DR_P" 3.0 3.0)
(family_check "PEAK_TIME"   "HPOTP_PRI_TRB_SL_DR_P" 3.0 3.0)
(family_check "EQ_VAL"      "HPOTP_PRI_TRB_SL_DR_P" 3.0 3.0
     turbine_equilibrium_interval)

(sensor (PIDs "1187")(parameter "HPOTP_PRI_PMP_SL_DR_T")
     (desc "HPOTP Pri Pump Seal Drain Temp")
     (range 250.0)(bit_toggle 0.15)(stddev 17.5)
     (cross_test_threshold 5.0)(erratic_threshold 2.0)
      (ambient_upper_limit 480.0)(ambient_lower_limit 110.0))
(family_check "MAX_AFTER_EQ" "HPOTP_PRI_PMP_SL_DR_T" 3.0 3.0
     turbine_equilibrium_interval)

(sensor (PIDs "1188")(parameter "HPOTP_SEC_TRB_SL_DR_T")
     (desc "HPOTP Sec Turbine Seal Drain Temp")
     (range 250.0)(bit_toggle 0.22)(stddev 19.5)(erratic_threshold 5.0)
      (ambient_upper_limit 800.0)(ambient_lower_limit 400.0))

(sensor (PIDs "1190")(parameter "HPOTP_PRI_TRB_SL_DR_T")
     (desc "HPOTP Pri Turbine Seal Drain Temp")
     (range 650.0)(bit_toggle 0.22)(stddev 19.6)(erratic_threshold 3.0)
      (ambient_upper_limit 1020.0)(ambient_lower_limit 380.0))

(sensor (PIDs "ENGONPSP")(parameter "ENG_LOX_NPSP")
```

```
        (should_not_be_flat FALSE)
        (desc "Engine LOX Inlet NPSP (special calc)"))
  )


;;; ------------------------- UTILITY FUNCTIONS ---------------------------
;;;

;;; DEFFUNCTION EQTIME
;;; Returns TRUE if two times are approximately equal.
(deffunction eqtime (?t1 ?t2)
  (< (abs (- ?t1 ?t2)) 0.005)) ;;0.050))

;;; DEFFUNCTION APPROX-EQ
;;; Returns TRUE if two values are approximately equal (within a stated tolerance).
(deffunction approx-eq (?v1 ?v2 ?tolerance)
  (< (abs (- ?v1 ?v2)) ?tolerance))

;;; DEFFUNCTION IS_CONCURRENT
;;; Returns TRUE if two time intervals (start1, end1, start2, end2) overlap.
;;; Overlapping on a single data point doesn't count.
(deffunction is_concurrent (?s1 ?e1 ?s2 ?e2)
  (and (< ?s1 ?e2)
       (< ?s2 ?e1)))

;;; DEFFUNCTION EVENT_IN_INTERVAL
;;; Returns TRUE if a time is within an interval (including endpoints).
(deffunction event_in_interval (?event ?istart ?iend)
  (or (eqtime ?event ?istart)
      (eqtime ?event ?iend)
      (and (< ?istart ?event)(> ?event ?iend))))

;;; DEFFUNCTION XOR
;;; Exclusive-or.
(deffunction xor (?a ?b)
  (or (and ?a (not ?b))(and ?b (not ?a))))

;;; DEFFUNCTION PID_RATE
;;; Returns the samples-per-second of a PID, given the PID number as a string.
(deffunction PID_rate (?pid)
  (if (>= (nth 1 (str-explode ?pid)) 300) then
      50
  else
      25))

;;; DEFFUNCTION CHECK_DB_CONNECTION
;;; Ensures that tkclips is connected to TekBase and SSME_DB is open.
;;; Returns TRUE if successful, FALSE if this cannot be achieved
;;; (also halts the program).
(deffunction check_DB_connection ()
  (if (not ?*DB_connected*) then
     (bind ?*DB_connected* (DB_connect))
     (if ?*DB_connected* then
         (bind ?*DB_connected* (DB_exec "OPEN SSME_DB"))
     ))
  (if (not *?DB_connected*) then
    (fprintout t "Failed to connect to TekBase. Aborting...")
    (halt))
  *?DB_connected*)

;;; DEFFUNCTION IS_TRUE
;;; Returns TRUE if its argument is TRUE.
;;; (Used for testing global variables.)
```

```
(deffunction is_true (?boolean)
  (not (not ?boolean)))

;;; DEFFUNCTION STDDEV
;;; Returns the standard deviation of a sample, given the
;;; number of samples, the sum and the sum-of-squares of the samples.
(deffunction stddev (?N ?sum ?sum-of-squares)
  (sqrt (/ (- ?sum-of-squares (* ?N (/ ?sum ?N) (/ ?sum ?N)))
           (- ?N 1))))
```

FILE: HPOTP_exec.clp
Diagnostic System Executive

```
;;; HPOTP Diagnostic Module
;;; Phase Executive & Utilities
;;;
;;; Creation 8/10/93  T.W. Bickmore


;;;-------------------------- PHASES -------------------------------------

;;; DEFFACTS EXEC_PHASES
;;; Defines the series of phases for normal execution, and a user printout
;;; message for each.
(deffacts EXEC_phases
  (phases initialize         "SSME HPOTP Diagnostic System"
         SVAL_hard_failures         "Checking for hard sensor failures."
         SVAL_soft_failures         "Determining preferred sensors."
         get_features               "Determining diagnostic features."
         find_event_intervals       "Determining time intervals to analyze."
         find_anomalies             "Diagnosing."
          prepare_output            "Preparing anomaly descriptions."
          output_anomalies          "Writing data."
          wrapup                    "Wrapping up.")
  (current_phase initialize))


;;; DEFRULE EXEC_CHANGE_PHASE
;;; Changes from one phase to the next when all processing has stopped.
(defrule EXEC_change_phase
  (declare (salience -10000))
  ?f <- (current_phase ?curr)
  (phases $? ?curr ? ?next ?label $?)
  =>
  (retract ?f)
  (assert (current_phase ?next)))

;;; DEFRULE EXEC_PRINT_PHASE
;;; Prints out a message whenever a new phase is entered.
(defrule EXEC_print_phase
  (declare (salience 10000))
  (current_phase ?phase)
  (phases $? ?phase ?label $?)
  =>
  (fprintout t crlf "*** " ?label crlf))


;;;----------------------- Get Test IDs & Profiles -------------------------

;;; DEFRULE EXEC_GET_TESTIDS
;;; Issues a request for the current & comparison test IDs.
(defrule EXEC_get_testIDs
  (declare (salience 10))
  (current_phase initialize)
  =>
  (assert (GetFeature TestIDs)))
```

```
;;; DEFRULE EXEC_ASK_TO_LOG
;;; When in interactive mode, asks the user if he/she wants to create a record
;;; of the processing (log file).
(defrule EXEC_ask_to_log
  (declare (salience 8))
  (current_phase initialize)
  =>
  (if (is_true ?*interactive*) then
    (fprintout t "File name for transcript (<CR> for none): ")
    (bind ?filename (readline))
    (fprintout t crlf)
    (if (> (length ?filename) 0) then
      (dribble-on ?filename)
      (assert (EXEC_dribbling)))))

;;; DEFRULE EXEC_GET_PROFILES
;;; Issues a request for the thrust profiles for the current and comparison tests.
(defrule EXEC_get_profiles
  (current_phase initialize)
  (current_test|comparison_test ?testid)
  =>
  (assert (GetFeature ConstantThrust ?testid)))


;;;---------------------- Wrap Up ---------------------------------

;;; DEFRULE EXEC_WRAPUP
;;; This should be the last rule in the system to fire. It shuts off
;;; the log file, and exits CLIPS (unless in DEBUG mode).
(defrule EXEC_wrapup
  (declare (salience -10000))
  (current_phase wrapup)
  =>
  (dribble-off)
  (if (not ?*DEBUG*) then
      (exit)))

;;; DEFRULE EXEC_DISCONNECT
;;; Disconnects from TekBase at program completion.
(defrule EXEC_disconnect
  (declare (salience -9999))
  (current_phase wrapup)
  (test (is_true ?*DB_connected*))
  =>
  (DB_disconnect))

;;; DEFRULE EXEC_UPDATE_RESOURCE_BOARD
;;; If being batch processed (not interactive mode) this makes a call
;;; to an external C program to update the resource board for the PTDS session
;;; manager to notify it that HPOTP has completed processing.
(defrule EXEC_update_resource_board
  (current_phase wrapup)
  (test (not ?*interactive*))
  =>
  (system (str-cat "HPOTP_update_RSRC " (get_param))))
```

FILE: HPOTP_features.clp
Feature Extraction Rules

```
;;; HPOTP Diagnostic Module
;;; Feature Extraction Functions
;;;
;;; Creation 8/10/93  T.W. Bickmore

;;; Summary:
;;; These rules obtain features requested by other parts of the HPOTP
;;; system. The requests (in the form of 'GetFeature' facts) are honored
;;; by either importing features from TekBase (in batch processing mode)
;;; or by computing them on-the-fly as needed.

;;;------------------------ UTILITY FUNCTIONS / RULES ----------------

;;; DEFFUNCTION GET_HPOTP_DATA
;;; Imports records from TekBase tables for the HPOTP module.
;;; ?underscore_test -- Defines whether the TESTID field in the table
;;;    has an underscore in its name (TEST_ID) or not.
;;; $?columns -- The list of columns to import. Note that TESTID is always
;;;    pre-pended to this list.
(deffunction get_HPOTP_data (?table ?testid ?underscore_test $?columns)
  (if (check_DB_connection) then
    (if ?underscore_test then
      (DB_get ?table (mv-append TEST_ID ?columns)
        (str-cat "TEST_ID='" ?testid "' AND MODULE='HPOTP'"))
    else
      (DB_get ?table (mv-append TESTID $?columns)
        (str-cat "TESTID='" ?testid "' AND MODULE='HPOTP'")))))

;;; DEFRULE FEAT_DISPLAY_CURRENT_STATUS
;;; In interactive mode this displays status messages to the user.
(defrule FEAT_display_current_status
  (declare (salience 1000))
  (GetFeature ?type&~TestIDs $?testid)
  (test (is_true ?*interactive*))
  =>
  (if (= (length $?testid) 0) then
       (fprintout t "Performing " ?type " analysis." crlf)
  else
       (fprintout t "Performing " ?type " analysis for " (nth 1 $?testid) "," crlf)))

;;; DEFRULE FEAT_REMOVE_FEATURE_REQUEST
;;; Retracts a feature request after FEAT rules have had a chance to honor it.
(defrule FEAT_remove_feature_request
  (declare (salience 995))
  ?f <- (GetFeature $?)
  =>
  (retract ?f))

;;; DEFFACTS FEATURE-RELATIONS
;;; Defines the relations used to represent features.
(deffacts Feature-Relations
  (FeatureRelations F_ERRAT F_LEVSH F_PEAK F_INRANGE F_DIFTHA F_SPIKE F_THLEDE
       F_BISTAB F_GREENBISTAB F_RLVIOL F_ISFLAT F_NOISE SEGMENT))
```

```
;;; DEFRULE DEBUG_PRINT_FEATURES
;;; In DEBUG mode this prints out every feature obtained.
(defrule DEBUG_print_features
  (declare (salience 9999))
  (FeatureRelations $? ?rel $?)
  (?rel $?stuff)
  (test (is_true ?*DEBUG*))
  =>
  (bind $?stuff (mv-append ?rel $?stuff))
  (fprintout t $?stuff crlf))

;;; DEFFACTS INIT_SENSOR_MAPS
;;; This defines any transformations that must be made to TekBase feature
;;; fields in order to make them consistent with those computed on-the-fly.
(deffacts init_sensor_maps
  (sensor_map "327a - 328a" "327 - 328"))

;;; DEFRULE FEAT_MAP_SENSORS
;;; This performs the transformations specified in INIT_SENSOR_MAPS.
(defrule FEAT_map_sensors
  (declare (salience 10000))
  (FeatureRelations $? ?rel $?)
  ?f <- (?rel $?s1 ?pid $?s2)
  (sensor_map ?pid $?replacement)
  =>
  (retract ?f)
  (assert (?rel $?s1 $?replacement $?s2)))


;;;---------------------- TestID Determination ----------------------

;;; DEFRULE FEAT_GETTESTIDS_1
;;; If not in interactive mode, this attempts to import the comparison test
;;; ID from TekBase (asserted previously by WPREV).
(defrule FEAT_GetTestIds_1
  (declare (salience 998))
  (GetFeature TestIDs)
  (test (not ?*interactive*))
  =>
  (bind ?testid (get_param))
  (assert (current_test ?testid))
  (if (check_DB_connection) then
    (DB_get POSTUL (mv-append PROBLEM POST_NUMBER)
      (str-cat "TEST_ID='" ?testid "' AND MODULE='HPOTP' AND POST_NUMBER=1000"))))

;;; DEFRULE FEAT_EXTRACT_COMPARISON_TEST
;;; Extracts the comparison test ID defined by WPREV (in batch mode).
(defrule FEAT_extract_comparison_test
  (declare (salience 998))
  (POSTUL ?string 1000)
  (test (eq (nth 1 (str-explode ?string)) "Test"))
  =>
  (assert (comparison_test =(nth 2 (str-explode ?string)))))

;;; DEFULE FEAT_GETTESTIDS_2
;;; In interactive mode, this queries the user for the current and comparison
;;; test IDs.
(defrule FEAT_GetTestIDs_2
  (declare (salience 998))
  (GetFeature TestIDs)
  (test (is_true ?*interactive*))
  =>
  (fprintout t "Enter Test ID to analyze (e.g. A10583): ")
  (assert (current_test =(readline)))
```

```
  (fprintout t "Enter comparison Test ID (or <CR> for none): ")
  (bind ?comp (readline))
  (fprintout t crlf)
  (if (> (str-length ?comp) 0) then
      (assert (comparison_test ?comp))))

;;;----------------------- TekBase Feature Load -------------------------
;;; Just loads all features upon 1st request...

;;; DEFFACTS INIT_TEKBASE_FEATURES
;;; Defines the feature classes which can be honored by features imported from
;;; TekBase.
(deffacts init_TekBase_features
  (TekBase_features ConstantThrust BalancePistonShifts DrainPeaks IsFlat Bistable
                    ComparisonDifferences SensorAnomaly DataAvailability
                     SensorRedundants))

;;; DEFRULE FEAT_LOADTEKBASEFEATS_1
;;; If import of TekBase features is enabled, this loads in ALL TekBase features
;;; for the specified test when the FIRST such request is made.
(defrule FEAT_LoadTekBaseFeats_1
  (declare (salience 998))
  (TekBase_features $? ?feature $?)
  (GetFeature ?feature ?testid $?)
  (test (is_true ?*DB_features*))
  (not (LoadedFeatures ?testid))
  =>
  (if (or ?*DEBUG* ?*interactive*) then
       (fprintout t "Loading features from TekBase..." crlf))
  (assert (LoadedFeatures ?testid))
  (if (check_DB_connection) then
  (get_HPOTP_data F_BISTAB ?testid FALSE
    (mv-append FIT_START FIT_END))
  (get_HPOTP_data F_DIFTHA ?testid FALSE
    (mv-append SENSOR START_TIME END_TIME COMP_TESTID COMP_SENSOR COEF_W_ERR_B
               DIF_BY_OFFSE OFFSET OFFSET_SIGMA))
  (get_HPOTP_data F_ERRAT ?testid FALSE
    (mv-append SENSOR START_TIME END_TIME))
  (get_HPOTP_data F_ISFLAT ?testid FALSE
    (mv-append SENSOR START_TIME END_TIME OFFSET SLOPE OFFSET_SIGMA SLOPE_SIGMA))
  (get_HPOTP_data F_LEVSH ?testid FALSE
    (mv-append SENSOR START_TIME END_TIME LAST_MAG DELTA))
  (get_HPOTP_data F_NOISE ?testid FALSE
    (mv-append SENSOR START_TIME END_TIME))
  (get_HPOTP_data F_PEAK ?testid FALSE
    (mv-append SENSOR TAPH PEAK_HT FWHM))
  (get_HPOTP_data F_RLVIOL ?testid FALSE
    (mv-append SENSOR PAIR_SENSOR VIOLAT_START VIOLAT_END CHECK_TYPE LIMIT_TYPE
               REDLINE))
  (get_HPOTP_data F_SPIKE ?testid FALSE
    (mv-append SENSOR START_TIME END_TIME MAGNITUDE))
  (get_HPOTP_data F_THLEDE ?testid FALSE
    (mv-append START_TIME END_TIME THRUST_LEVEL))
  (get_HPOTP_data F_NOISE ?testid FALSE
    (mv-append SENSOR START_TIME END_TIME))
  (DB_get PIDINFO (mv-append TEST_ID PID START_TIME END_TIME DESCR UNITS RATE)
    (str-cat "TEST_ID='" ?testid "'"))
  (DB_get TESTINFO (mv-append TEST_ID ENG_SHUTDOWN)
    (str-cat "TEST_ID='" ?testid "'"))
  (if (or ?*DEBUG* ?*interactive*) then
       (fprintout t "Finished loading features." crlf crlf))
  ))
```

```
;;; DEFRULE FEAT_FAILTEKBASELOAD
;;; This detectsthe situation inwhich HPOTP attempts to load featuresfrom TekBase, but
;;; failed.In this case the system hasno means of recovery, so we just print a message
;;; and halt.
(defrule FEAT_FailTekBaseLoad
  (declare (salience 997))
  (LoadedFeatures ?testid)
  (not (TESTINFO ?testid $?))
  =>
  (fprintout t crlf "*** Failed to load information from TekBase. Aborting." crlf)
  (halt))


;;;---------------------- PID INFO ----------------------------------

;;; DEFRULE FEAT_GETPIDINFO
;;; Responds to requests for information about all defined PIDs.
;;; (FeatureClass=DataAvailability)
(defrule FEAT_GetPIDInfo
  (declare (salience 998))
  (GetFeature DataAvailability ?testid)
  (test (not ?*DB_features*))
  (sensor_status ?testid ?pid ?param ~invalid $?)
  =>
  (if (is_true ?*DEBUG*) then (fprintout t "DATA_info " ?testid "," ?pid crlf))
  (DATA_info ?testid ?pid))


;;;---------------------- THRUST PROFILE FEATURES  --------------------

;;; DEFRULE FEAT_FINDCONSTANTTHRUST
;;;Responds to requests forthrust profiles forspecified tests by computing on-the-fly.
;;; (FeatureClass=ConstantThrust)
(defrule FEAT_FindConstantThrust
  (declare (salience 998))
  (GetFeature ConstantThrust ?testid)
  (test (not ?*DB_features*))
  =>
  (if (is_true ?*DEBUG*) then (fprintout t "DATA_FindConstantThrust " ?testid crlf))
  (DATA_FindConstantThrust ?testid)
  (DATA_info ?testid "287"))

;;; DEFRULE FEAT_DISPLAY_THRUST_PROFILE
;;;In interactive mode thisprints out the thrust profile for the current (and possibly
;;; comparison) tests.
(defrule FEAT_Display_Thrust_Profile
  (declare (salience 999))
  (F_THLEDE ?testid ?start ?end ?PL)
  (test (and ?*interactive* (not ?*DEBUG*)))
  =>
  (format t "Thrust Profile for %6s: %4.0f - %4.0f @ %3d%%%n"
       ?testid ?start ?end ?PL))
```

```
;;; DEFRULE FEAT_DETERMINE_COMPARISON_INTERVALS
;;; Determines periods of equivalent thrust between the current and comparison tests.
;;; Uses same method as SAIC's GetThrustLevelIntersection
(defrule FEAT_determine_comparison_intervals
  (declare (salience 998))
  (current_test ?current)
  (test (not *?DB_features*))
  (comparison_test ?comparison)
  (F_THLEDE ?current ?s1 ?e1 ?PL)
  (F_THLEDE ?comparison ?s2 ?e2 ?PL)
  (test (is_concurrent ?s1 ?e1 ?s2 ?e2))
  =>
  (assert (comparison_interval ?current ?s1 ?e1 ?comparison ?s2 ?e2 ?PL)))

;;; DEFRULE FEAT_DETERMINE_SHUTDOWN
;;; Asserts the shutdown time of the current and comparison tests.
(defrule FEAT_determine_shutdown
  (declare (salience 998))
  (or (PIDINFO ?testid "287" ? ? ? ? ? ?shutdown)
      (TESTINFO ?testid ?shutdown))
  =>
  (if (or ?*interactive* ?*DEBUG*) then
      (format t "Shutdown for %6s at %4.0f%n" ?testid ?shutdown))
  (assert (shutdown_time ?testid ?shutdown)))


;;;---------------------- LOX VENT PROFILE FEATURES ----------------------

;;; DEFRULE FEAT_FINDVENTPROFILE
;;; Responds to requrests for LOX Vent profile (Class=SteadyState),
;;; by computing it on-the-fly using a piecewise-linear curve.
(defrule FEAT_FindVentProfile
  (declare (salience 998))
  (GetFeature SteadyState ?testid)
  (test (not ?*DB_features*))
  (F_THLEDE ?testid ?start ?end ?)
  (LOXInletP ?testid $? ?pid $?)
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_FindPieceWise " ?testid "," ?pid ",75,"
                 (+ ?start 1) "," ?end ",50,0.7,1" crlf))
  (DATA_FindPieceWise ?testid ?pid 150 (+ ?start 1) ?end 100 0.3 1))

;;; DEFRULE FEAT_FIND_STEADY_STATE
;;; Determines periods of constant thrust and linear LOX inlet pressure.
(defrule FEAT_FindSteadyState
  (declare (salience 898))
  (confirmed|unconfirmed SEGMENT ?testid ?pid&"ENGONPSP"|"858"|"859"|"860"
       ?segstart ?segend ?startval ?endval)
  (F_THLEDE ?testid ?start ?end ?PL)
  (test (is_concurrent ?start ?end ?segstart ?segend))
  =>
  (if (and (or ?*interactive* ?*DEBUG*)
           (or (> (- ?segstart ?start) 2.0)(> (- ?end ?segend) 2.0))) then
      (format t
       "LOX Vent Profile: %4.0f - %4.0f : %3s Changing from %6.2f to %6.2f @ %3d%%%n"
          ?segstart ?segend ?pid ?startval ?endval ?PL))
  (assert (linear_behavior ?testid ?segstart ?segend ?PL ?startval ?endval)))
```

```
;;; DEFRULE FEAT_FINDMINMAXNPSP
;;; Responds to requests for LOX Vent profile by determining periods
;;; of minimum, maximum, and nominal LOX inlet conditions during 104% and 109%
;;; power levels.
(defrule FEAT_FindMinMaxNPSP
  (declare (salience 998))
  (GetFeature SteadyState ?testid)
  (F_THLEDE ?testid ?start ?end ?PL&:(>= ?PL 104))
  (LOXInletP ?testid $? ?pid $?)
  =>
  (if (< ?PL 109) then ;;104%
      (DATA_FindInRange ?testid ?pid ?start ?end 20.0 25.0 12 "104_MIN_NPSP")
      (DATA_FindInRange ?testid ?pid ?start ?end 150.0 160.0 12 "104_MAX_NPSP")
      (DATA_FindInRange ?testid ?pid ?start ?end 75.0 85.0 12 "104_NOM_NPSP")
  else
      (DATA_FindInRange ?testid ?pid ?start ?end 20.0 25.0 12 "109_MIN_NPSP")
      (DATA_FindInRange ?testid ?pid ?start ?end 150.0 160.0 12 "109_MAX_NPSP")))


;;;---------------------- BALANCE PISTON SHIFT FEATURES -----------------

;;; DEFRULE FEAT_FINDBALANCEPISTONSHIFTS_2A
;;; Responds to requests for BalancePistonShifts features by computing level shifts in
;;; 327 and 328.
(defrule FEAT_FindBalancePistonShifts_2A
  (declare (salience 998))
  (GetFeature BalancePistonShifts)
  (test (not ?*DB_features*))
  (current_test ?current)
  (sensor_status ?current ?pid&"327"|"328" ? ~invalid $?)
  (sensor (PIDs $? ?pid $?)(range ?range&~nil))
  (F_THLEDE ?current ?start ?end ?PL)
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_FindLevelShift " ?current "," ?pid ",1,"
              ?start "," ?end "," ?range crlf))
  (DATA_FindLevelShift ?current ?pid 1 ?start ?end ?range))

;;; DEFRULE FEAT_FINDBALANCEPISTONSHIFTS_2B
;;; Responds to requests for BalancePistonShifts features by computing deltalevel
;;;  shifts in 327 and 328.
(defrule FEAT_FindBalancePistonShifts_2B
  (declare (salience 998))
  (GetFeature BalancePistonShifts)
  (test (not ?*DB_features*))
  (current_test ?current)
  (sensor_status ?current "327" ? ~invalid $?)
  (sensor_status ?current "328" ? ~invalid $?)
  (F_THLEDE ?current ?start ?end ?PL)
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_DeltaLevelShift " ?current ",327," ?start "," ?end ","
       ?current ",328," ?start "," ?end ",1" crlf))
  (DATA_DeltaLevelShift ?current "327" ?start ?end ?current "328" ?start ?end 1))
```

```
;;; DEFRULE FEAT_FINDBALANCEPISTONSHIFTS_2B
;;; Responds to requests for BalancePistonShifts features by computing
;;; deltadifferenthan
;;; for 327 and 328 between current and comparison tests.
(defrule FEAT_FindBalancePistonShifts_2C
  (declare (salience 998))
  (GetFeature BalancePistonShifts)
  (test (not ?*DB_features*))
  (current_test ?current)
  (comparison_test ?comparison)
  (sensor_status ?current "327" ? ~invalid $?)
  (sensor_status ?current "328" ? ~invalid $?)
  (sensor_status ?comparison "327" ? ~invalid $?)
  (sensor_status ?comparison "328" ? ~invalid $?)
  (sensor (PIDs $? 327 $?)(range ?range&~nil))
  (comparison_interval ?current ?s1 ?e1 ?comparison ?s2 ?e2 ?)
  =>
  (if (is_true ?*DEBUG*) then
     (fprintout t "DATA_DeltaDifferentThan " ?current ",327,328," ?s1 "," ?e1
       "," ?comparison ",327,328," ?s2 "," ?e2 ",1," ?range crlf))
  (DATA_DeltaDifferentThan ?current "327" "328" ?s1 ?e1 ?comparison
       "327" "328" ?s2 ?e2 1 ?range))

;;; DEFRULE FEAT_FINDBALANCEPISTONSEGMENTS
;;; Responds to requests for BalancePistonShifts features by computing a piece-wise
;;; linear model of the difference between 327 and 328 (used for rotor drag
;;; detection).
;;;NOT in TekFeatures...
(defrule FEAT_FindBalancePistonSegments
  (declare (salience 998))
  (GetFeature BalancePistonShifts)
  (current_test ?current)
  (sensor_status ?current "327" ? ~invalid $?)
  (sensor_status ?current "328" ? ~invalid $?)
  (linear_behavior ?current ?start ?end $?)
  =>
  (DATA_DeltaPieceWise ?current "327" (+ ?start 1) ?end
                       ?current "328" (+ ?start 1) ?end
                     1 3 1.0 0))

;;; DEFRULE FEAT_CHECKBALANCEPISTONSLOPES
;;; Computes the slope of segments detected by the piece-wise linear model
;;; computed above.
(defrule FEAT_CheckBalancePistonSlopes
  (declare (salience 998))
  (GetFeature BalancePistonShifts)
  (current_test ?testid)
  (DSEGMENT ?testid "327" ?testid "328" ?start ?end ? ?)
  =>
  (DATA_FitLine ?testid "327" 0 ?start ?end)
  (DATA_FitLine ?testid "328" 0 ?start ?end))
```

```
;;;---------------- DRAIN TEMP & PRESS PEAK FEATURES -------------

;;; DEFRULE FEAT_GETDRAINPEAKS_2A
;;; Responds to requests for DrainPeaks by computing them on-the-fly.
(defrule FEAT_GetDrainPeaks_2A
  (declare (salience 998))
  (GetFeature DrainPeaks)
  (current_test|comparison_test ?testid)
  (test (not ?*DB_features*))
  (sensor_status ?testid ?pid&"91"|"92"|"990" ? ~invalid $?)
  (shutdown_time ?testid ?shutdown)
  =>
  (if (is_true ?*DEBUG*) then
     (fprintout t "DATA_FindPeak " ?testid "," ?pid
       ",1,0," ?shutdown ",2,8" crlf))
  (DATA_FindPeak ?testid ?pid 1 0 ?shutdown 2 8))

;;; DEFRULE FEAT_JUSTFIRSTPEAK
;;; HPOTP is only interested in the first peak for 91, 92, and 990, so
;;; this removes all subsequent ones to avoid confusion.
(defrule FEAT_JustFirstPeak
  (declare (salience 998))
  (GetFeature DrainPeaks)
  (F_PEAK ?testid ?pid&"91"|"92"|"990" ?time $?)
  ?f <- (F_PEAK ?testid ?pid ?time2&:(> ?time2 ?time) $?)
  =>
  (retract ?f))


;;;--------------------- FAMILY CHECKS (not covered elsewhere) ----------
;;; Responds to FamilyChecks request by computing all features needed
;;; to compute historically-tracked parameters.

;;; DEFRULE FEAT_GETCURRENTEQUILIBRIUM
;;; Computes the avarage value needed for EQ_VAL parameters.
(defrule FEAT_GetCurrentEquilibrium
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (family_check "EQ_VAL" ?param ? ? ?interval)
  (?interval ?testid ?start ?end)
  (sensor_status ?testid ?pid ?param preferred)
  =>
  (DATA_stats ?testid ?pid ?start ?end))

;;; DEFRULE FEAT_GETCURRENTSTART
;;; Computes the average value between -1.0 and 0.0 needed for START_VAL parameters.
(defrule FEAT_GetCurrentStart
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (family_check "START_VAL" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  =>
  (DATA_stats ?testid ?pid -1.0 0.0))

;;; DEFRULE FEAT_GETCURRENT5TOCUT
;;; Computes the average value between 5.0 and cutof needed for 5_TO_CUT parameters.
(defrule FEAT_GetCurrent5toCut
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (family_check "5_TO_CUT" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (shutdown_time ?testid ?shut)
  =>
  (DATA_stats ?testid ?pid 5.0 ?shut))
```

```
;;; DEFRULE FEAT_GETCURRENTMAX
;;; Computes the maximum value between thermal equilibrium and shutdown for
MAX_AFTER_EQ parameters.
(defrule FEAT_GetCurrentMax
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (family_check "MAX_AFTER_EQ" ?param ? ? ?interval)
  (sensor_status ?testid ?pid ?param preferred)
  (shutdown_time ?testid ?shut)
  (?interval ?testid ?start ?)
  =>
  (DATA_stats ?testid ?pid ?start ?shut))

;;; DEFRULE FEAT_GETCURRENT104MIN
;;;Computes the averagevalue of a parameter during minimumLOX inlet conditions at104%.
(defrule FEAT_GetCurrent104MIN
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (family_check "104_MIN_NPSP" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end "104_MIN_NPSP")
  (not (confirmed|unconfirmed F_INRANGE ?testid ?
              ?start2&:(< ?start2 ?start) ? "104_MIN_NPSP"))
  =>
  (DATA_stats ?testid ?pid ?start ?end))

;;; DEFRULE FEAT_GETCURRENT109MAX
;;;Computes the averagevalue of a parameter during maximumLOX inlet conditions at109%.
(defrule FEAT_GetCurrent109MAX
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (family_check "109_MAX_NPSP" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end "109_MAX_NPSP")
  (not (confirmed|unconfirmed F_INRANGE ?testid ?
              ?start2&:(< ?start2 ?start) ? "109_MAX_NPSP"))
  =>
  (DATA_stats ?testid ?pid ?start ?end))

;;; DEFRULE FEAT_GETCURRENT104NOM
;;;Computes the averagevalue of a parameter during nominalLOX inlet conditions at 104%
;;; (defined as 75.0 - 85.0 NPSP).
(defrule FEAT_GetCurrent104NOM
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (family_check "104_NOM_NPSP" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end "104_NOM_NPSP")
  (not (confirmed|unconfirmed F_INRANGE ?testid ?
              ?start2&:(< ?start2 ?start) ? "104_NOM_NPSP"))
  =>
  (DATA_stats ?testid ?pid ?start ?end))
```

```
;;; DEFRULE FEAT_GETFAMILYCHECKS
;;; Retrieves all historically-tracked parameters from the HISTORY table.
(defrule FEAT_GetFamilyChecks
  (declare (salience 998))
  (GetFeature FamilyChecks ?testid)
  (test (is_true ?*DB_enabled*))
  (family_check ?type ?param $?)
  =>
  (if (check_DB_connection) then
     (DB_get HISTORY
          (mv-append "TESTID" "TYPE" "PARAM" "VALUE" "OK_TO_USE")
          (str-cat "TYPE='" ?type "' AND PARAM='" ?param
               "' AND TESTID<>'" ?testid  "' AND OK_TO_USE=TRUE"))))


;;;---------------------------- BISTABLE FEATURES ----------------------

;;; DEFRULE FEAT_GETBISTABLE_2
;;; Calls the C routine to detect PBP bistability for power levels at or
;;; below 65%.
(defrule FEAT_GetBistable_2
  (declare (salience 998))
  (GetFeature Bistable ?testid)
  (F_THLEDE ?testid ?start ?end ?PL&:(and (<= ?PL 65)(> ?PL 0)))
  (sensor_status ?testid ?pid1&"210"|"209" ? preferred)
  (sensor_status ?testid ?pid2&"140"|"141" ? preferred)
  =>
  (if (is_true ?*DEBUG*) then (fprintout t "DATA_DetectGreenBistable,"
       (+ ?start 3) "," ?end "," ?pid1 "," ?pid2 crlf))
  (DATA_DetectGreenBistability ?testid (+ ?start 3.0) ?end ?pid1 ?pid2))


;;;--------------------------- HARD SENSOR FAILURE FEATURES ------------

;;; DEFRULE FEAT_GETDISCONNECTED_2
;;; Determines if a sensor is 'flat' between START and SHUTDOWN (an
;;; indication that the sensor is not connected).
;;; Currently NOT available from TekBase...
(defrule FEAT_GetDisconnected_2
  (declare (salience 998))
  (GetFeature DisconnectedSensor ?testid)
  (sensor_status ?testid ?pid ? ~invalid $?)
  (shutdown_time ?testid ?shut)
  (sensor (PIDs $? ?pid $?)(should_not_be_flat TRUE))
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_IsFlat " ?testid "," ?pid ",1,0,6,3" crlf))
  (DATA_IsFlat ?testid ?pid 1 0.0 ?shut 3))

;;; DEFRULE FEAT_GETGROSSNOISE
;;; Determines if a sensor exceeds gross noise limits.
;;; Currently NOT available from TekBase...
;;; Bad if stddev > 2*normal stddev during steady state
(defrule FEAT_GetGrossNoise
  (declare (salience 998))
  (GetFeature NoisySensor ?testid)
  (sensor_status ?testid ?pid ? ~invalid $?)
  (sensor (PIDs $? ?pid $?)(stddev ?stddev&~nil))
  (F_THLEDE ?testid ?start ?end ?)
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_FindNoise " ?testid "," ?pid ","
              ?start "," ?end "," (* ?stddev 2.0) crlf))
  (DATA_FindNoise ?testid ?pid ?start ?end (* ?stddev 3.0)))
```

```
;;; DEFRULE FEAT_GETREASONABLENESS
;;; Determines if a sensor exceeds reasonablness limits during the test.
(defrule FEAT_GetReasonableness
  (declare (salience 998))
  (GetFeature SensorReasonableness ?testid)
  (sensor_status ?testid ?pid ? ~invalid $?)
  (sensor (PIDs $? ?pid $?)(reasonable_limit ?limit&~nil))
  (shutdown_time ?testid ?shut)
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_CheckUpperLimit " ?testid "," ?pid ",0,"
              ?shut "," ?limit ",2" crlf))
  (DATA_CheckUpperLimit ?testid ?pid 0.0 ?shut ?limit 2))

;;; DEFRULE FEAT_GETAMBIENTLIMITS
;;; Determines if a sensor exceeds reasonableness limits (upper or lower) before
;;; (-6.0 to 0.0) or after (shutdown+30.0 - shutdown+36.0) the test.
(defrule FEAT_GetAmbientLimits
  (declare (salience 998))
  (GetFeature SensorAmbientLimits ?testid)
  (sensor_status ?testid ?pid ? ~invalid $?)
  (sensor (PIDs $? ?pid $?)(ambient_upper_limit ?upper)(ambient_lower_limit ?lower))
  (shutdown_time ?testid ?shut)
  =>
  (if (neq ?upper nil) then
      (if (is_true ?*DEBUG*) then
          (fprintout t "DATA_CheckUpperLimit " ?testid "," ?pid
                       ",-6,0," ?upper ",2" crlf)
          (fprintout t "DATA_CheckUpperLimit " ?testid "," ?pid "," (+ ?shut 30)
                       "," (+ ?shut 36) "," ?upper ",2" crlf))
      (DATA_CheckUpperLimit ?testid ?pid -6.0 0.0 ?upper 2)
      (DATA_CheckUpperLimit ?testid ?pid (+ ?shut 30.0) (+ ?shut 36.0) ?upper 2))
  (if (neq ?lower nil) then
      (if (is_true ?*DEBUG*) then
          (fprintout t "DATA_CheckLowerLimit " ?testid "," ?pid
                       ",-6,0," ?lower ",2" crlf)
          (fprintout t "DATA_CheckLowerLimit " ?testid "," ?pid "," (+ ?shut 30)
                       "," (+ ?shut 36) "," ?lower ",2" crlf))
      (DATA_CheckLowerLimit ?testid ?pid -6.0 0.0 ?lower 2)
      (DATA_CheckLowerLimit ?testid ?pid (+ ?shut 30.0) (+ ?shut 36.0) ?lower 2)))


;;;------------------------- SENSOR PREFERENCE FEATURES ----------------

;;; DEFRULE FEAT_GETSENSORNANOMALIES_1
;;; Computes Erratic and Spike features on-the-fly for each period of linear engine
;;; behavior.  These are also used during diagnosis...
(defrule FEAT_GetSensorAnomalies_1
  (declare (salience 998))
  (GetFeature SensorAnomaly ?testid)
  (test (not ?*DB_features*))
  (sensor_status ?testid ?pid ? ~invalid $?)
  (sensor (PIDs $? ?pid $?)(erratic_threshold ?erratic&~nil)(spike_factor ?percent)
          (bit_toggle ?toggle&~nil)(range ?range&~nil)(stddev ?stddev))
  (linear_behavior ?testid ?start ?end $?)
  =>
  (bind ?erratic_thresh (* ?erratic 4.0))
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_FindErratic " ?testid "," ?pid ","
                   ?start "," ?end "," ?erratic_thresh crlf)
      (fprintout t "DATA_FindSpike " ?testid "," ?pid "," (+ ?start 1.4) "," ?end ","
              ?toggle "," ?range "," ?percent crlf))
  (DATA_FindErratic ?testid ?pid ?start ?end ?erratic_thresh)
  (DATA_FindSpike ?testid ?pid (+ ?start 1.4) ?end ?toggle ?range ?percent))
```

```
;;;---------------------- REDUNDANT CHANNEL CHECKS -------------------

;;; DEFRULE FEAT_GETSENSORANOMALIES_2
;;; Computes different-than features between redundant parameters.
(defrule FEAT_GetSensorAnomalies_2
  (declare (salience 998))
  (GetFeature SensorRedundants ?testid)
  (test (not ?*DB_features*))
  (sensor (PIDs $? ?pid1 $? ?pid2 $?)(comparison_threshold ?tolerance&~nil))
  (sensor_status ?testid ?pid1 ? ~invalid $?)
  (sensor_status ?testid ?pid2 ? ~invalid $?)
  (F_THLEDE ?testid ?start ?end ?)
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "DATA_FindDifferentThan " ?testid "," ?pid1 ","
                   ?start "," ?end "," ?testid "," ?pid2
                   "," ?start "," ?end ",1," ?tolerance crlf))
  (DATA_FindDifferentThan ?testid ?pid1 ?start ?end
                          ?testid ?pid2 ?start ?end 1 ?tolerance))


;;;---------------------- COMPARISON TEST DIFFERENCES ----------------

;;; DEFRULE FEAT_GETCOMPARISONDIFFERENCES
;;; Computes different-than features between current and comparison tests.
(defrule FEAT_GetComparisonDifferences
  (declare (salience 998))
  (GetFeature ComparisonDifferences)
  (test (not ?*DB_features*))
  (current_test ?current)
  (comparison_test ?comparison)
  (sensor_status ?current ?pid1 ?param preferred $?)
  (sensor (PIDs $? ?pid1 $?)(cross_test_threshold ?cross_test_threshold&~nil))
  (sensor_status ?comparison ?pid2 ?param preferred $?)
  (comparison_interval ?current ?s1 ?e1 ?comparison ?s2 ?e2 ?)
  =>
  (if (is_true ?*DEBUG*) then
     (fprintout t "DATA_FindDifferentThan " ?current "," ?pid1 ","
                  ?s1 "," ?e1 "," ?comparison ","
                  ?s2 "," ?e2 ",1," ?cross_test_threshold))
  (DATA_FindDifferentThan ?current ?pid1 ?s1 ?e1 ?comparison
                  ?pid2 ?s2 ?e2 1 ?cross_test_threshold))


;;;---------------------- Determine LOX Inlet ----------------------------
;;; Rules to determine which PID to use to determine the LOX inlet profile.
;;; Prefer to use ENGONPSP, but if it is not available, use  858, 859, 860.

;;; DEFRULE FEAT_GREEN_DETERMINE_LOXIN
;;; Checks to see if ENGONPSP exists.
(defrule FEAT_Green_Determine_LOXIn
  (declare (salience 1000))
  (current_test|comparison_test ?testid&~"")
  =>
  (DATA_info ?testid "ENGONPSP"))

;;; DEFRULE FEAT_GREEN_USENPSP
;;; Asserts that ENGONPSP will be used for LOX inlet calculations.
(defrule FEAT_Green_UseNPSP
  (declare (salience 1000))
  (PIDINFO ?testid "ENGONPSP" $?)
  =>
  (assert (LOXInletP ?testid "ENGONPSP")))
```

```
;;; DEFRULE FEAT_GREEN_USELOXIN
;;; Asserts that 858, 859, and 860 will be used for LOX inlet calculations.
(defrule FEAT_Green_UseLOXIn
  (declare (salience 999))
  (current_test ?testid)
  (not (LOXInletP ?testid $?))
  =>
  (assert (LOXInletP ?testid "858" "859" "860"))
  (fprintout t "Engine LOX NPSP (special calcs) not available for " ?testid
               ". Using 858/859/860." crlf))


;;;------------------------ GREENRUN SPEC FEATURES ----------------------------

;;; DEFRULE FEAT_GREEN_START
;;; Determines if limits at START have been exceeded.
(defrule FEAT_Green_start
  (declare (salience 998))
  (GetFeature GreenRun)
  (current_test ?testid)
  (GREEN_limit ?param START ?min ?max)
  (sensor_status ?testid ?pid ?param preferred)
  =>
  (if (neq ?max nil) then
      (DATA_CheckUpperLimit ?testid ?pid -1.0 0.0 ?max 2))
  (if (neq ?min nil) then
      (DATA_CheckLowerLimit ?testid ?pid -1.0 0.0 ?min 2)))

;;; DEFRULE FEAT_GREEN_IMSL_START
;;; Computes avarage values for 211, 212, and 937 at START (-1 to 0).
(defrule FEAT_Green_IMSL_start
  (declare (salience 998))
  (GetFeature GreenRun)
  (current_test ?testid)
  (sensor_status ?testid ?IMSLpid&"211"|"212" ? preferred)
  (sensor_status ?testid "937" ? preferred)
  =>
  (DATA_stats ?testid ?IMSLpid -1.0 0.0)
  (DATA_stats ?testid "937"   -1.0 0.0))

;;; DEFRULE FEAT_GREEN_LIMITS
;;; Determines if limits have been exceeded during a specified power level.
(defrule FEAT_Green_limits
  (declare (salience 998))
  (GetFeature GreenRun)
  (current_test ?testid)
  (shutdown_time ?testid ?shut)
  (GREEN_limit ?param ?PL&~START ? ?min ?max)
  (F_THLEDE ?testid ?start ?end ?PL2&:(approx-eq ?PL ?PL2 2.0))
  (sensor_status ?testid ?pid ?param preferred)
  =>
  (if (neq ?max nil) then
      (DATA_CheckUpperLimit ?testid ?pid ?start ?end ?max 2))
  (if (neq ?min nil) then
      (DATA_CheckLowerLimit ?testid ?pid ?start ?end ?min 2)))
```

```
;;; DEFRULE FEAT_GREEN_DELTAT
;;; Determines if the minimum delta-T requirements are ever violated.
(defrule FEAT_Green_deltaT
  (declare (salience 998))
  (GetFeature GreenRun)
  (current_test ?testid)
  (shutdown_time ?testid ?shut)
  (sensor_status ?testid ?turbDS&"233"|"234"|"518"|"519"|"521"|"522" ? preferred)
  (sensor_status ?testid "1190" ? preferred)
  =>
  (if (is_true ?*DEBUG*) then (fprintout t "DATA_CheckDifference for DT" crlf))
  (DATA_CheckDifference ?testid ?turbDS "1190" 6.0 ?shut 280.0 2 0)
  (DATA_CheckDifference ?testid ?turbDS "1190" 6.0 ?shut 370.0 2 0))

;;; DEFRULE FEAT_GREEN_DELTASPEED
;;; Computes HPOTP speed during periods of change in LOX inlet pressure.
(defrule FEAT_Green_DeltaSpeed
  (declare (salience 998))
  (GetFeature GreenRun)
  (current_test ?testid)
  (sensor_status ?testid "2" ~invalid)
  (confirmed|unconfirmed SEGMENT ?testid ?pid&"ENGONPSP"|"858"|"859"|"860"
              ?start&:(> ?start 10.0) ?end ? ?)
  =>
  (DATA_stats ?testid 2 (- ?start 0.5) (+ ?start 0.5)))

;;; DEFRULE FEAT_GREEN_PC104
;;; Determines periods when the thrust level is 104% or greater.
(defrule FEAT_Green_PC104
  (declare (salience 998))
  (GetFeature GreenRun)
  (current_test ?testid)
  (shutdown_time ?testid ?shut)
  =>
  (DATA_CheckUpperLimit ?testid "287" 6.0 ?shut 3126.0 2))

;;; DEFRULE FEAT_GREEN_COLD_TURBINE
;;; Determines periods when the HPOT turbine discharge temp is "cold" (below 1300).
(defrule FEAT_Green_cold_turbine
  (declare (salience 998))
  (GetFeature GreenRun)
  (current_test ?testid)
  (shutdown_time ?testid ?shut)
  (sensor_status ?testid ?pid&"233"|"234"|"518"|"519"|"521"|"522" ? preferred)
  =>
  (DATA_CheckLowerLimit ?testid ?pid 6.0 ?shut 1300.0 2))
```

FILE: HPOTP_redundancy.clp
Feature Redundancy Management

```
;;; HPOTP Diagnostic Module
;;; Feature Redundancy Management Module
;;;
;;; Creation 8/10/93  T.W. Bickmore
;;;
;;; Summary:
;;; Performs redundancy management for a select group of features.
;;; All input features have the form:
;;;   (<relation> <testid> <pid> <start-time> <args>*)
;;; Following redundancy processing, the features will have the form:
;;;   (<status> <relation> <testid> <pid> <start-time> <args>*)
;;; Where <status> can have the following values:
;;;  *For sensors with only one bridge each...
;;;   unconfirmed -- If the PID has no valid redundants.
;;;   spurious    -- If the PID has valid redundants, none of which
;;;                  register a similar feature at the same time.
;;;   confirmed   -- If the PID has a redundant which registers a similar
;;;                  feature at the same time. Any outlier redundants (which
;;;                  did not see the feature) are marked as spurious.
;;;  *For sensors with two bridges...
;;;   confirmed   -- If seen on any bridge of 2 or more transducers.
;;;                  (Outlier bridges are marked as spurious.)
;;;   unconfirmed -- Seen on all valid bridges of a transducer and there
;;;                  are no other valid transducers.
;;;   spurious    -- If more than one transducer is valid but only seen on one.
;;;                  (All bridges are marked as spurious.)
;;;
;;; Notes:
;;; * Feature equivalence is based on start times only (must be within
;;;   one second of each other).
;;; * The resulting 'confirmed' feature from two or more redundant
;;;   features is formed by simply selecting one of the inputs (i.e.,
;;;   no attempt is made to average the values).
;;;
;;; Salience Range: 900

;;; DEFFACTS RED_REDUNDANCY_FEATURES
;;; Defines the feature relations which redundancy management is applicable to.
(deffacts RED_redundancy_features
  (RED_redundancy_check F_ERRAT F_LEVSH F_PEAK SEGMENT
       F_SPIKE F_ISFLAT F_NOISE F_INRANGE))
```

```
;;; ************************ Transducers Without Multiple Bridges ********************

;;; DEFRULE RED_UNCONFIRMED1
;;; Detects when a feature for a sensor without multiple bridges is unconfirmed.
(defrule RED_unconfirmed1
  (declare (salience 900))
  (RED_redundancy_check $? ?relation $?)
  ?f <- (?relation ?testid ?pid $?args)
  (sensor_status ?testid ?pid ?param ~invalid)
  (not (bridges ? $? ?pid $?))
  (not (sensor_status ?testid ~?pid ?param ~invalid))
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "Status of [" ?relation "," ?testid "," ?pid "," $?args
                   "] is unconfirmed." crlf))
  (retract ?f)
  (assert (unconfirmed ?relation ?testid ?pid $?args)))

;;; DEFRULE RED_SPURIOUS1
;;; Detects when a feature for a sensor without multiple bridges is spurious.
(defrule RED_spurious1
  (declare (salience 899))
  (RED_redundancy_check $? ?relation $?)
  ?f <- (?relation ?testid ?pid1 ?start1 $?args1)
  (not (bridges ? $? ?pid1 $?))
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "Status of [" ?relation "," ?testid "," ?pid1 "," $?args1
            "] is spurious." crlf))
  (retract ?f)
  (assert (spurious ?relation ?testid ?pid1 ?start1 $?args1)))

;;; DEFRULE RED_CONFIRMED1A
;;; Detects when a feature for a sensor without multiple bridges is confirmed,
;;; and asserts that all redundant sensors which disagree with the feature
;;; are spurious.
(defrule RED_confirmed1a
  (declare (salience 901))
  (RED_redundancy_check $? ?relation $?)
  ?f1 <- (?relation ?testid ?pid1 ?start1 $?args1)
  (sensor_status ?testid ?pid1 ?param ~invalid)
  (sensor_status ?testid ?pid2&~?pid1 ?param ~invalid)
  (not (bridges ? $? ?pid1 $?))
  (?relation ?testid ?pid2 ?start2&:(approx-eq ?start1 ?start2 1.5) $?args2)
  ;;Is confirmed, now look for outliers...
  (sensor_status ?testid ?pid3&~?pid1&~?pid2 ?param ~invalid)
  (not (?relation ?testid ?pid3 ?start3&:(approx-eq ?start1 ?start3 1.5) $?))
  (not (RED_ok ?relation ?testid ?pid3 ?start3&:(approx-eq ?start1 ?start3 1.5)))
  =>
  (assert (spurious ?relation ?testid ?pid3 ?start1)))
```

```
;;; DEFRULE RED_CONFIRMED1B
;;; Detects when a feature for a sensor without multiple bridges is confirmed.
(defrule RED_confirmed1b
  (declare (salience 900))
  (RED_redundancy_check $? ?relation $?)
  ?f1 <- (?relation ?testid ?pid1 ?start1 $?args1)
  (sensor_status ?testid ?pid1 ?param ~invalid)
  (sensor_status ?testid ?pid2&~?pid1 ?param ~invalid)
  (not (bridges ? $? ?pid1 $?))
  (or ?f2 <- (?relation ?testid ?pid2 ?start2&:(approx-eq ?start1 ?start2 1.5)
             $?args2 )
      ?f2 <- (confirmed ?relation ?testid ?pid2
             ?start2&:(approx-eq ?start1 ?start2 1.5) $?args2 ))
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "Status of [" ?relation "," ?testid "," ?pid1
       "," $?args1 "] is confirmed." crlf))
  (retract ?f1 ?f2)
  (assert (confirmed ?relation ?testid ?pid1 ?start1 $?args1)
          (RED_ok ?relation ?testid ?pid2 ?start2)))


;;; ************************ Transducers With 2 Bridges ********************

;;; DEFRULE RED_UNCONFIRMED2
;;; Detects when a sensor with multiple bridges has an unconfirmed feature.
;;; Seen on both bridges of a transducer, and no other valid transducers exist.
(defrule RED_unconfirmed2
  (declare (salience 900))
  (RED_redundancy_check $? ?relation $?)
  ?f <- (?relation ?testid ?pid ?start $?args)
  (sensor_status ?testid ?pid ?param ~invalid)
  (or (bridges ?param ?pid ?pidb)
      (bridges ?param ?pidb ?pid))
  (?relation ?testid ?pidb ?startb&:(approx-eq ?start ?startb 1.5) $?)
  (not (transducer_status ?testid ?param ? valid ~?pid&~?pidb $?))
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "Status of [" ?relation "," ?testid "," ?pid "," $?args
              "] is unconfirmed." crlf))
  (retract ?f)
  (assert (unconfirmed ?relation ?testid ?pid $?args)))

;;; DEFRULE RED_SPURIOUS2
;;; Detects when a sensor with multiple bridges has a spurious feature.
;;; Only seen on one transducer (any bridge combination). Other valid transducers
exist.
(defrule RED_spurious2
  (declare (salience 898))
  (RED_redundancy_check $? ?relation $?)
  ?f1 <- (?relation ?testid ?pid1 ?start1 $?args1)
  (sensor_status ?testid ?pid1 ?param ~invalid)
  (or (bridges ?param ?pid1 ?pidb)
      (bridges ?param ?pidb ?pid1))
  =>
  (retract ?f1)
  (assert (spurious ?relation ?testid ?pid1 ?start1)))
```

```
;;; DEFRULE RED_CONFIRMED2
;;; Detects when a sensor with multiple bridges has a confirmed feature.
;;; Seen on any bridge of 2 or more transducers.
(defrule RED_confirmed2
  (declare (salience 900))
  (RED_redundancy_check $? ?relation $?)
  ?f1 <- (?relation ?testid ?pid1 ?start1 $?args1)
  (sensor_status ?testid ?pid1 ?param ~invalid)
  (sensor_status ?testid ?pid2&~?pid1 ?param ~invalid)
  (or (bridges ?param $? ?pid1 $?)      ;;One of the sensors shares a transducer.
      (bridges ?param $? ?pid2 $?))
  (not (bridges ?param ?pid1 ?pid2))    ;;These two sensors do not share a transducer.
  (not (bridges ?param ?pid2 ?pid1))
  (or ?f2 <- (?relation ?testid ?pid2 ?start2&:(approx-eq ?start1 ?start2 1.5)
              $?args2 )
      ?f2 <- (confirmed ?relation ?testid ?pid2
              ?start2&:(approx-eq ?start1 ?start2 1.5) $?args2 ))
  =>
  (if (is_true ?*DEBUG*) then
      (fprintout t "Status of [" ?relation "," ?testid "," ?pid1
       "," $?args1 "] is confirmed." crlf))
  (retract ?f1 ?f2)
  (assert (confirmed ?relation ?testid ?pid1 ?start1 $?args1)
          (RED_ok ?relation ?testid ?pid1 ?start1)
          (RED_ok ?relation ?testid ?pid2 ?start2)))

;;; DEFRULE RED_CONFIRMED_SPURIOUS_BRIDGE
;;; Detects when a sensor with multiple bridges has a confirmed feature, and
;;; asserts that any bridges which did not see the feature are spurious.
;;; Confirmed feature found, but one bridge did not detect.
(defrule RED_confirmed_spurious_bridge
  (declare (salience 901))
  (RED_redundancy_check $? ?relation $?)
  ?f1 <- (?relation ?testid ?pid1 ?start1 $?args1)
  (sensor_status ?testid ?pid1 ?param ~invalid)
  (sensor_status ?testid ?pid2&~?pid1 ?param ~invalid)
  (or (bridges ?param $? ?pid1 $?)     ;;One of the sensors shares a transducer.
      (bridges ?param $? ?pid2 $?))
  (not (bridges ?param ?pid1 ?pid2))  ;;These two sensors do not share a transducer.
  (not (bridges ?param ?pid2 ?pid1))
  (?relation ?testid ?pid2 ?start2&:(approx-eq ?start1 ?start2 1.5) $?args2 )
  ;;Confirmed features, now find spurious bridges...
  (or (bridges ?param ?pid1|?pid2 ?pidS)
      (bridges ?param ?pidS ?pid1|?pid2))
  (sensor_status ?testid ?pidS ?param ~invalid)
  (not (?relation ?testid ?pidS ?start3&:(approx-eq ?start1 ?start3 1.5) $?))
  (not (RED_ok ?relation ?testid ?pidS ?start3&:(approx-eq ?start1 ?start3 1.5)))
  =>
  (assert (spurious ?relation ?testid ?pidS ?start2)))
```

FILE: HPOTP_stats.clp
Family Statistics Management

```
;;; HPOTP Diagnostic Module
;;; Family statisics collection and comparison routines.
;;;
;;; Creation 8/10/93  T.W. Bickmore

;;;---------------------- HISTORICAL (FAMILY) STATISTICS --------------
;;; Relations:
;;;   (FAMILY_tally  <type> <parameter> <N> <sum> <sum-of-squares>)
;;;   (FAMILY_STAT   <type> <parameter> <mean> <stddev> <N>)
;;;   (CURRENT_STAT <testid> <type> <parameter> <value>)
;;;   (out_of_family <testid> <type> <parameter> HIGH|LOW <num-sigmas>)
;;;   (family_check <type> <parameter> <n-sig-lo> <n-sig-hi> [<interval>])


;;; --------------- COMPUTE THE MEAN AND STDDEV OF THE FAMILY  --------

;;; DEFRULE STAT_GETFAMILYCHECKS
;;; Initializes FAMILY_tally facts for all family checks (counters which
;;; will be used to compute statistics).
(defrule STAT_GetFamilyChecks
  (declare (salience 800))
  (family_check ?type ?param $?)
  =>
  (assert (FAMILY_tally ?type ?param 0 0 0)))

;;; DEFRULE STAT_TALLY_FAMILY_STATS
;;; Increments FAMILY_tally counter facts for all family checks (adds
;;; data point for one test to the sum and sum-of-squares tallies).
(defrule STAT_tally_family_stats
  (declare (salience 799))
  ?f1 <- (HISTORY ?testid ?type ?param ?value ?ok&:(approx-eq ?ok 1 0.1))
  ?f2 <- (FAMILY_tally ?type ?param ?N ?sum ?sum-of-squares)
  =>
  (retract ?f1 ?f2)
  (assert (FAMILY_tally ?type ?param =(+ ?N 1)
       =(+ ?sum ?value) =(+ ?sum-of-squares (* ?value ?value)))))

;;; DEFRULE STAT_COMPUTEFAMILY_STATS
;;; Computes the mean and standard deviation of family checked parameters.
(defrule STAT_computefamily_stats
  (declare (salience 796))
  (logical (FAMILY_tally ?type ?param ?N ?sum ?sum-of-squares))
  =>
  (if (>= ?N 2) then
      (bind ?mean (/ ?sum ?N))
      (bind ?stddev (stddev ?N ?sum ?sum-of-squares))
     (assert (FAMILY_STAT ?type ?param ?mean ?stddev ?N))
      (if (or ?*DEBUG* ?*interactive*) then
          (fprintout t "Historical Value of " ?param " is "
              ?mean " +/- " (* 3.0 ?stddev) " (3 StdDevs)." crlf))))
```

```
;;; ----------------- EXTRACT ANALYSIS VALUES FROM CURRENT TEST TO GO IN DB ---------

;;; DEFRULE STAT_GET_PEAK_HEIGHT_STAT
;;; Determines the PEAK_HEIGHT statistics for the current test.
(defrule STAT_get_peak_height_stat
  (declare (salience 795))
  (current_test ?testid)
  (family_check "PEAK_HEIGHT" ?param ?pk-n-sig-lo ?pk-n-sig-hi $?)
  (sensor_status ?testid ?pid ?param ~invalid)
  (confirmed|unconfirmed F_PEAK ?testid ?pid ? ?currpeak $?)
  =>
  (if (or ?*interactive* ?*DEBUG*) then (fprintout t "Peak Height for "
       ?param " is " ?currpeak "." crlf))
  (assert (CURRENT_STAT ?testid "PEAK_HEIGHT" ?param ?currpeak)))

;;; DEFRULE STAT_GET_PEAK_TIME_STAT
;;; Determines the PEAK_TIME statistic for the current test.
(defrule STAT_get_peak_time_stat
  (declare (salience 795))
  (current_test ?testid)
  (family_check "PEAK_TIME" ?param ?pk-n-sig-lo ?pk-n-sig-hi $?)
  (sensor_status ?testid ?pid ?param ~invalid)
  (confirmed|unconfirmed F_PEAK ?testid ?pid ?time $?)
  =>
  (if (or ?*DEBUG* ?*interactive*) then (fprintout t "Peak Time for "
       ?param " is " ?time "." crlf))
  (assert (CURRENT_STAT ?testid "PEAK_TIME" ?param ?time)))

;;; DEFRULE STAT_GET_PEAK_WIDTH_STAT
;;; Determines the PEAK_WITH statistics for the current test.
(defrule STAT_get_peak_width_stat
  (declare (salience 795))
  (current_test ?testid)
  (family_check "PEAK_WIDTH" ?param ?pk-n-sig-lo ?pk-n-sig-hi $?)
  (sensor_status ?testid ?pid ?param ~invalid)
  (confirmed|unconfirmed F_PEAK ?testid ?pid ? ? ?width $?)
  =>
  (if (or ?*interactive* ?*DEBUG*) then (fprintout t "Peak Width for "
       ?param " is " ?width "." crlf))
  (assert (CURRENT_STAT ?testid "PEAK_WIDTH" ?param ?width)))

;;; DEFRULE STAT_GET_EQUILIBRIUM_STAT
;;; Determines the EQ_VAL statistics for the current test.
;;; Equlibirium value is the average of all data points during the first steady-state
;;; interval following the specified delay time for thermal equilbrium.
(defrule STAT_get_equilibrium_stat
  (declare (salience 795))
  (current_test ?testid)
  (family_check "EQ_VAL" ?param ? ? ?interval)
  (sensor_status ?testid ?pid ?param preferred)
  (?interval ?testid ?start ?end)
  (STATS ?testid ?pid ?sstart&:(eqtime ?sstart ?start)
       ?send&:(eqtime ?send ?end) ?value $?)
  =>
  (if (or ?*interactive* ?*DEBUG*) then (fprintout t
       "Equilibrium Value for " ?param " is " ?value "." crlf))
  (assert (CURRENT_STAT ?testid "EQ_VAL" ?param ?value)))

;;; DEFRULE STAT_GET_START_STAT
;;; Determines the START statistics for the current test
;;; (data averaged from -1.0 to 0.0).
(defrule STAT_get_START_stat
  (declare (salience 795))
  (current_test ?testid)
```

```
  (family_check "START_VAL" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (STATS ?testid ?pid ?sstart&:(eqtime ?sstart -1.0)
       ?send&:(eqtime ?send 0.0) ?value $?)
  =>
  (if (or ?*interactive* ?*DEBUG*) then (fprintout t "Start Value for "
       ?param " is " ?value "." crlf))
  (assert (CURRENT_STAT ?testid "START_VAL" ?param ?value)))

;;; DEFRULE STAT_GETCURRENT5TOCUT
;;; Determines the 5_TO_CUT statistics for the current test.
;;; Data averaged from 5.0 to cutoff.
(defrule STAT_GetCurrent5toCut
  (declare (salience 795))
  (current_test ?testid)
  (family_check "5_TO_CUT" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (shutdown_time ?testid ?shut)
  (STATS ?testid ?pid ?sstart&:(eqtime ?sstart 5.0) ?send&:(eqtime ?send ?shut)
             ?value $?)
  =>
  (if (or ?*interactive* ?*DEBUG*) then (fprintout t "5-to-Cut Value for "
       ?param " is " ?value "." crlf))
  (assert (CURRENT_STAT ?testid "5_TO_CUT" ?param ?value)))

;;; DEFRULE STAT_GETCURRENTMAX
;;; Determines the MAX_AFTER_EQ statistics for the current test.
;;; (Maximum value between a specified delay time for thermal equilibrium
;;; and cutoff.)
(defrule STAT_GetCurrentMax
  (declare (salience 795))
  (current_test ?testid)
  (family_check "MAX_AFTER_EQ" ?param ? ? ?interval)
  (sensor_status ?testid ?pid ?param preferred)
  (shutdown_time ?testid ?shut)
  (?interval ?testid ?start ?)
  (STATS ?testid ?pid ?sstart&:(eqtime ?sstart ?start) ?send&:(eqtime ?send ?shut)
             $? ?value)
  =>
  (if (or ?*DEBUG* ?*interactive*) then
       (fprintout t "Max Value After Thermal Equilibrium for "
             ?param " is " ?value "." crlf))
  (assert (CURRENT_STAT ?testid "MAX_AFTER_EQ" ?param ?value)))

;;; DEFRULE STAT_GETCURRENT104MIN
;;; Determines the 104_MIN_NPSP statistics for the current test.
(defrule STAT_GetCurrent104MIN
  (declare (salience 795))
  (current_test ?testid)
  (family_check "104_MIN_NPSP" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end "104_MIN_NPSP")
  (STATS ?testid ?pid ?sstart&:(eqtime ?sstart ?start) ?send&:(eqtime ?send ?end)
             ?value $?)
  =>
  (if (or ?*DEBUG* ?*interactive*) then (fprintout t "Value at 104% Min LOX NPSP for "
       ?param " is " ?value "." crlf))
  (assert (CURRENT_STAT ?testid "104_MIN_NPSP" ?param ?value)))
```

```
;;; DEFRULE STAT_GETCURRENT109MAX
;;; Determines the 109_MAX_NPSP statistics for the current test.
(defrule STAT_GetCurrent109MAX
  (declare (salience 795))
  (current_test ?testid)
  (family_check "109_MAX_NPSP" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end "109_MAX_NPSP")
  (STATS ?testid ?pid ?sstart&:(eqtime ?sstart ?start) ?send&:(eqtime ?send ?end)
             ?value $?)
  =>
  (if (or ?*DEBUG* ?*interactive*) then (fprintout t "Value at 109% Min LOX NPSP for "
       ?param " is " ?value "." crlf))
  (assert (CURRENT_STAT ?testid "109_MAX_NPSP" ?param ?value)))

;;; DEFRULE STAT_GETCURRENT104NOM
;;; Determines the 104_NOM_NPSP statistics for the current test.
(defrule STAT_GetCurrent104NOM
  (declare (salience 795))
  (current_test ?testid)
  (family_check "104_NOM_NPSP" ?param ? ?)
  (sensor_status ?testid ?pid ?param preferred)
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end "104_NOM_NPSP")
  (STATS ?testid ?pid ?sstart&:(eqtime ?sstart ?start) ?send&:(eqtime ?send ?end)
             ?value $?)
  =>
  (if (or ?*DEBUG* ?*interactive*) then
       (fprintout t "Value at 104% Nominal LOX NPSP for "
       ?param " is " ?value "." crlf))
  (assert (CURRENT_STAT ?testid "104_NOM_NPSP" ?param ?value)))


;;; ---------------------- CHECK OUT-OF-FAMILY CONDITIONS -----------------------

;;; DEFRULE STAT_OUT_OF_FAMILY_HIGH
;;; Detects when a parameter from the current test is more than N-sigma above the
;;; mean for the family.
(defrule STAT_out_of_family_high
  (declare (salience 795))
  (FAMILY_STAT  ?type ?param ?pk-mean ?pk-stddev ?)
  (CURRENT_STAT ?testid ?type ?param ?current)
  (family_check ?type ?param ?pk-n-sig-lo ?pk-n-sig-hi $?)
  (test (> ?current (+ ?pk-mean (* ?pk-n-sig-hi ?pk-stddev))))
  =>
  (assert (out_of_family ?testid ?type ?param HIGH
             =(/ (- ?current ?pk-mean) ?pk-stddev))))

;;; DEFRULE STAT_OUT_OF_FAMILYLOW
;;; Detects when a parameter from the current test is less than N-sigma below the
;;; mean for the family.
(defrule STAT_out_of_familylow
  (declare (salience 795))
  (FAMILY_STAT ?type ?param ?pk-mean ?pk-stddev ?)
  (CURRENT_STAT ?testid ?type ?param ?current)
  (family_check ?type ?param ?pk-n-sig-lo ?pk-n-sig-hi $?)
  (test (< ?current (- ?pk-mean (* ?pk-n-sig-lo ?pk-stddev))))
  =>
  (assert (out_of_family ?testid ?type ?param LOW
             =(/ (- ?pk-mean ?current) ?pk-stddev))))
```

FILE: HPOTP_sval.clp
Sensor Validation

```
;;; HPOTP Diagnostic Module
;;; SENSOR VALIDATION Routines
;;;
;;; Creation 8/10/93  T.W. Bickmore
;;;

;;; Summary:
;;; Works in two passes --
;;; 1. Hard Failure Detection. Attempts to disqualify on the basis
;;;    of excessive noise, reasonabless limit exceedances, deviations
;;;    from a majority of redundants, etc.
;;; 2. Soft Failure Detection. Attempts to determine a preferred
;;;    sensor from each set of remaining redundants based on
;;;    number of spikes and erratic behavior.  This is of limited
;;;    usefulness now, since most features are redundancy-voted.


;;; ******************** UTILITY RULES  ************************

;;; DEFFUNCTION SVAL_DISQUALIFY
;;; Performs all actions which must be taken when a sensor is disqualified.
;;; This assumes that the sensor_status fact has just been retracted.
(deffunction SVAL_disqualify (?testid ?pid ?desc ?param ?start ?end ?why)
  (bind ?class (gensym))
  (assert (sensor_status ?testid ?pid ?param invalid ?why)
          (anomaly (class ?class)(priority 8)(start ?start)(end ?end)
              (type INSTRUMENTATION)(PID ?pid)
              (description =(str-cat ?desc " (" ?pid ") disqualified. " ?why))))
  (if (neq ?start nil) then
      (assert (plot (class ?class)(number 1)(PIDs ?pid)(title ?desc)
              (anomaly_delta_start -5.0)(anomaly_delta_end +5.0)(full_sample TRUE))))
  (if (is_true ?*interactive*) then
      (fprintout t "Disqualifying " ?desc " (" ?pid ") for test "
              ?testid ". " ?why crlf)))


;;; ******************** TRANSDUCER STATUS UPDATE ***************
;;; To accomodate transducers with multiple bridges.
;;; Transducer is invalid when all of its bridges are invalid.

;;; DEFRULE SVAL_FAIL_TRANSDUCER1
;;; Invalidates a transducer with only 1 bridge.
(defrule SVAL_fail_transducer1
  (sensor_status ?testid ?pid1 ?param invalid $?)
  ?f <- (transducer_status ?testid ?param ?label valid ?pid1)
  =>
  (retract ?f)
  (assert (transducer_status ?testid ?param ?label invalid ?pid1)))


;;; DEFRULE SVAL_FAIL_TRANSDUCER2
;;; Invalidates a transducer with two bridges (when both are bad).
(defrule SVAL_fail_transducer2
  (sensor_status ?testid ?pid1 ?param invalid $?)
  (sensor_status ?testid ?pid2 ?param invalid $?)
  ?f <- (transducer_status ?testid ?param ?label valid ?pid1 ?pid2)
  =>
  (retract ?f)
  (assert (transducer_invalid ?testid ?param ?label invalid ?pid1 ?pid2)))
```

```
;;; ******************** SENSOR STATUS INITIALIZATION  ********************

;;; DEFRULE SVAL_ASSUME_SENSORS_VALID
;;; Asserts at program start-up that all sensors are assumed to be valid.
(defrule SVAL_assume_sensors_valid
  (declare (salience 10000))
  (sensor (PIDs $? ?pid $?)(parameter ?param))
  (current_test|comparison_test ?testid&~"")
  =>
  (assert (sensor_status ?testid ?pid ?param valid)))

;;; DEFRULE ASSUME_TRANSDUCER_VALID1
;;; Asserts at program start-up that all transducers with one bridge are valid.
(defrule SVAL_assume_transducer_valid1
  (declare (salience 10000))
  (sensor (PIDs $? ?pid $?)(parameter ?param))
  (current_test|comparison_test ?testid&~"")
  (not (bridges ?param $? ?pid $?))
  =>
  (assert (transducer_status ?testid ?param =(gensym) valid ?pid)))

;;; DEFRULE ASSUME_TRANSDUCER_VALID2
;;; Asserts at program start-up that all transducers with two bridges are valid.
(defrule SVAL_assume_transducer_valid2
  (declare (salience 10000))
  (sensor (PIDs $? ?pid $?)(parameter ?param))
  (current_test|comparison_test ?testid)
  (bridges ?param ?pid ?pid2)
  =>
  (assert (transducer_status ?testid ?param =(gensym) valid ?pid ?pid2)))


;;; ******************** HARD FAILURE PHASE ********************
;;; Requires that current_test be asserted, and thrust profile determined...

;;; DEFRULE SVAL_SEE_IF_EXISTS
;;; Issues a request to see if all sensors exist in the datafile.
(defrule SVAL_see_if_exists
  (declare (salience 100))
  (current_phase SVAL_hard_failures)
  (current_test|comparison_test ?testid)
  =>
  (assert (GetFeature DataAvailability ?testid)))

;;; DEFRULE SVAL_DISQUALIFY_IF_DOESNT_EXIST
;;; Disqualifies a sensor if data for it does not exist.
(defrule SVAL_disqualify_if_doesnt_exist
  (declare (salience 90))
  (current_phase SVAL_hard_failures)
  ?f <- (sensor_status ?testid ?pid ?param ~invalid $?)
  (not (PIDINFO ?testid ?pid $?))
  (sensor (parameter ?param)(desc ?desc))
  =>
  (retract ?f)
  (SVAL_disqualify ?testid ?pid ?desc ?param nil nil "Does not exist."))

;;; DEFFACTS SVAL_INIT_HARD_CHECKS
;;; List of feature classes used to detect hard failures.
(deffacts SVAL_init_hard_checks ;;In reverse order of execution...
  (SVAL_hard_checks SensorRedundancy SensorReasonableness SensorAmbientLimits
       DisconnectedSensor NoisySensor))
```

```
;;; DEFRULE SVAL_GET_HARD_FEATURES
;;; Issues a request for each class of features for hard sensor failure detection.
;;; This is done at low salience so that one feature class is obtained and analyzed
;;; before the next is requested.  This avoids performing unnecessary feature
;;; extraction on already-disqualified sensors.
(defrule SVAL_get_hard_features
  (declare (salience -10))
  (current_phase SVAL_hard_failures)
  (current_test|comparison_test ?testid)
  (SVAL_hard_checks $? ?check $?)
  =>
  (assert (GetFeature ?check ?testid)))

;;; DEFRULE SVAL_TOO_NOISY
;;; Disqualifies a sensor on the basis of exceesive noise.
;;; This rule acts BEFORE redundancy management kicks in.
(defrule SVAL_too_noisy
  (declare (salience 2000))
  (current_phase SVAL_hard_failures)
  ?f <- (sensor_status ?testid ?pid ?param ~invalid $?)
  (F_NOISE ?testid ?pid ?start ?end)
  (sensor (parameter ?param)(desc ?desc))
  =>
  (retract ?f)
  (SVAL_disqualify ?testid ?pid ?desc ?param ?start ?end "Exceede noise limits."))

;;; DEFRULE SVAL_DISCONNECTED
;;; Disqualifies a sensor if it appears to be flat for the duration of the firing.
;;; This rule acts BEFORE redundancy management kicks in...
(defrule SVAL_disconnected
  (declare (salience 2000))
  (current_phase SVAL_hard_failures)
  ?f <- (sensor_status ?testid ?pid ?param ~invalid $?)
  (shutdown_time ?testid ?shut)
  (F_ISFLAT ?testid ?pid ?st&:(eqtime ?st 0) ?sh&:(eqtime ?sh ?shut) $?)
  (sensor (parameter ?param)(desc ?desc))
  =>
  (retract ?f)
  (SVAL_disqualify ?testid ?pid ?desc ?param ?st ?sh "Apparently disconnected."))

;;; DEFRULE SVAL_UNREASONABLE_1
;;; Disqualifies a sensor if it exceeds reasonableness limits during the test.
(defrule SVAL_unreasonable_1
  (declare (salience 1999))
  (current_phase SVAL_hard_failures)
  ?f <- (sensor_status ?testid ?pid ?param ~invalid)
  (sensor (PIDs $? ?pid $?)(reasonable_limit ?limit&~nil))
  (shutdown_time ?testid ?shut)
  (F_RLVIOL ?testid ?pid ? ?start ?end ? ? ?lim&:(eqtime ?lim ?limit))
  (sensor (parameter ?param)(desc ?desc))
  =>
  (retract ?f)
  (SVAL_disqualify ?testid ?pid ?desc ?param ?start ?end
       "Exceeds reasonableness limits during test."))
```

```
;;; DEFRULE SVAL_UNREASONABLE_2
;;; Disqualifies a sensor if it exceed reasonableness limits before the test.
(defrule SVAL_unreasonable_2
  (declare (salience 2000))
  (current_phase SVAL_hard_failures)
  ?f <- (sensor_status ?testid ?pid ?param ~invalid)
  (sensor (PIDs $? ?pid $?)(ambient_upper_limit ?upperlimit&~nil)
       (ambient_lower_limit ?lowerlimit&~nil))
  (shutdown_time ?testid ?shut)
  (F_RLVIOL ?testid ?pid ? ?s1&:(< ?s1 0.0) ?e1 $?
       ?lim&:(or (eqtime ?lim ?upperlimit)(eqtime ?lim ?lowerlimit)))
  (sensor (parameter ?param)(desc ?desc))
  =>
  (retract ?f)
  (SVAL_disqualify ?testid ?pid ?desc ?param ?s1 ?e1
       "Exceeds reasonableness limits before test."))

;;; DEFRULE SVAL_UNREASONABLE_3
;;; Disqualifies a sensor if it exceed reasonableness limits after the test.
(defrule SVAL_unreasonable_3
  (declare (salience 1998))
  (current_phase SVAL_hard_failures)
  ?f <- (sensor_status ?testid ?pid ?param ~invalid)
  (sensor (PIDs $? ?pid $?)(ambient_upper_limit ?upperlimit&~nil)
       (ambient_lower_limit ?lowerlimit&~nil))
  (shutdown_time ?testid ?shut)
  (F_RLVIOL ?testid ?pid ? ?s1&:(> ?s1 ?shut) ?e1 $?
       ?lim&:(or (eqtime ?lim ?upperlimit)(eqtime ?lim ?lowerlimit)))
  (sensor (parameter ?param)(desc ?desc))
  =>
  (retract ?f)
  (SVAL_disqualify ?testid ?pid ?desc ?param ?s1 ?e1
       "Exceeds reasonableness limits after test."))


;;; DEFRULE SVAL_INIT_STRIKES
;;; Initializes the number 'strikes' against a sensor. This is used during hard
;;; failure detection to record significant discrepances between this sensor and its
;;; redundants. It is further used in soft failure detection.
(defrule SVAL_init_strikes
  (current_phase SVAL_hard_failures)
  (sensor_status ?testid ?pid ?param ~invalid $?)
  (sensor (PIDs $? ?pid $?)(parameter ?param))
  =>
  (assert (sensor_strikes ?testid ?pid ?param 0)))

;;; DEFRULE SVAL_TALLY_REDUNDANT_DIFFERENCE
;;; Updates sensor 'strikes' in response to a significant discrepancy between
;;; redundants.
(defrule SVAL_tally_redundant_differences
  (current_phase SVAL_hard_failures)
  ?f1 <- (F_DIFTHA ?testid ?pid1 ? ? ?testid ?pid2 $?)
  ?f2 <- (sensor_strikes ?testid ?pid1 ?param ?sofar1)
  ?f3 <- (sensor_strikes ?testid ?pid2 ?param ?sofar2)
  =>
  (retract ?f1 ?f2 ?f3)
  (assert (sensor_strikes ?testid ?pid1 ?param =(+ ?sofar1 1))
          (sensor_strikes ?testid ?pid2 ?param =(+ ?sofar2 1))))
```

```
;;; DEFRULE SVAL_DISQUALIFY_REDUNDANTS
;;; Disqualifies a sensor on the basis of being an outlier from a majority of
;;; redundants.
(defrule SVAL_disqualify_redundants
  (declare (salience -10))
  (current_phase SVAL_hard_failures)
  ?f1 <- (sensor_strikes ?testid ?s1 ?param ?s1count)
  (sensor_strikes ?testid ~?s1 ?param ?) ;;At least 1 other redundant exists.
  (not (sensor_strikes ?testid ? ?param ?s2count&:(>= ?s2count ?s1count)))
             ;;No others with same or larger count.
  ?f2 <- (sensor_status ?testid ?s1 ?param ~invalid $?)
  (sensor (parameter ?param)(desc ?desc))
  =>
  (retract ?f1 ?f2)
  (SVAL_disqualify ?testid ?s1 ?desc ?param nil nil
       "Differs from majority of redundants."))


;;; ******************************** CHECK SOFT FAILURES **********************
;;; Determine "best" of remaining redundant sensors.

;;; DEFRULE SVAL_GET_VENT_PROFILE
;;; Issues a request for vent profile determination (required by Spike and Erratic).
(defrule SVAL_get_vent_profile
  (declare (salience 1000))
  (current_phase SVAL_soft_failures)
  (current_test|comparison_test ?testid)
  =>
  (assert (GetFeature SteadyState ?testid)))

;;; DEFRULE SVAL_GET_ERRATIC_FEATURES
;;; Issues a request for Spike and Erratic features.
(defrule SVAL_get_erratic_features
  (declare (salience 10))
  (current_phase SVAL_soft_failures)
  (current_test|comparison_test ?testid)
  =>
  (assert (GetFeature SensorAnomaly ?testid)))

;;; DEFRULE SVAL_TALLY_ABNORMAL_STRIKES
;;; Counts any 'spurious' features (spikes or erratics) into a sensor's strikes.
(defrule SVAL_tally-abnormal-strikes
  (current_phase SVAL_soft_failures)
  ?f1 <- (spurious ? ?testid ?pid $?)
  ?f2 <- (sensor_strikes ?testid ?pid ?param ?sofar)
  =>
  (retract ?f1 ?f2)
  (assert (sensor_strikes ?testid ?pid ?param =(+ ?sofar 1))))


;;; DEFRULE SVAL_TALLY_CROSS_TEST_DIFFERENCE
;;; Counts any significant deviations between this test and a comparison into a
sensor's strikes.
(defrule SVAL_tally_cross_test_differences
  (current_phase SVAL_soft_failures)
  (current_test ?testid)
  (comparison_test ?compareid)
  ?f1 <- (F_DIFTHA ?testid ?pid1 ? ? ?compareid ?pid2 $?)
  ?f2 <- (sensor_strikes ?testid ?pid1 ?param ?sofar1)
  ?f3 <- (sensor_strikes ?testid ?pid2 ?param ?sofar2)
  =>
  (retract ?f1 ?f2 ?f3)
  (assert (sensor_strikes ?testid ?pid1 ?param =(+ ?sofar1 1))
          (sensor_strikes ?testid ?pid2 ?param =(+ ?sofar2 1))))
```

```
;;; DEFRULE SVAL_PREFER_CLEAN
;;; Determines the preferred sensor out of each set of valid redundants, based on
;;; minimum strike count.
(defrule SVAL_prefer_clean
  (declare (salience -20))
  (current_phase SVAL_soft_failures)
  (sensor_strikes ?testid ?s1 ?param ?s1count)
  (not (sensor_strikes ?testid ? ?param ?s2count&:(< ?s2count ?s1count)))
  (not (sensor_status ?testid ? ?param preferred $?))
  ?f <- (sensor_status ?testid ?s1 ? ~invalid $?)
  (sensor (PIDs $? ?s1 $?)(desc ?text))
  =>
  (if (is_true ?*interactive*) then
    (format t "Using sensor %4s for %s, on test %s.%n" ?s1 ?text ?testid))
  (retract ?f)
  (assert (sensor_status ?testid ?s1 ?param preferred)     ))
```

FILE: HPOTP_event.clp
Analysis Time Interval Determination

```
;;; HPOTP Diagnostic Module
;;; Event Detection & Time Paritioning Routines
;;;
;;; Creation 8/10/93  T.W. Bickmore

;;; Summary:
;;; To simplify many of the anomaly detection rules, the test time-line
;;; is partitioned into intervals within which nothing is happening
;;; (i.e. no features start or end).  The anomaly detection rules which
;;; are interested in concurrent combinations of features  can then
;;; simply analyze each of these time intervals separately.
;;; The results of this analysis is asserted as the following:
;;;  (event_interval <testid> <start> <end> <power_level>|TRANSIENT)
;;;
;;; If the same anomaly is detected in adjacent intervals, the anomaly
;;; descriptions are combined into one spanning the entire interval.

;;;------------------------ Anomaly Feature Request --------------------

;;; DEFRULE EVENT_GET_FEATURES
;;; Issues requests for all features used in diagnosis and greenrun spec
;;; analyses.
(defrule EVENT_get_features
  (current_phase get_features)
  (current_test ?testid)
  =>
  (assert (GetFeature ComparisonDifferences)
          (GetFeature BalancePistonShifts)
          (GetFeature DrainPeaks)
          (GetFeature FamilyChecks ?testid)
          (GetFeature GreenRun)
          (GetFeature IsFlat ?testid)
          (GetFeature Bistable ?testid)))


;;;------------------------ Anomaly Time Interval Rules ----------------

;;; DEFRULE EVENT_FIND_EVENT_TIMES
;;; Determines when events of significance occur (start and end times
;;; of a specific class of features).
(defrule EVENT_find_event_times
  (current_phase get_features)
  (current_test ?testid)
  (or (F_ERRAT|F_ISFLAT|F_LEVSH|F_SPIKE|F_DIFTHA ?testid ? ?start ?end $?)
      (F_BISTAB|F_THLEDE|F_GREENBISTAB ?testid ?start ?end $?))
  =>
  (assert (event_time ?testid ?start)
          (event_time ?testid ?end)))

;;; DEFRULE EVENT_FIND_PUMP_EQUILIBRIUM_INTERVAL
;;; Finds 1st steady-state segment during which pump should be in thermal equilibrium.
(defrule EVENT_find_pump_equlibrium_interval
  (pump_equilibrium_time ?eqt)
  (current_test|comparison_test ?testid)
  (F_THLEDE ?testid ?start ?end&:(< ?eqt ?end) ?)
  (not (F_THLEDE ?testid ?start2&:(< ?start2 ?start) ?end2&:(< ?eqt ?end2) ?))
  =>
  (assert (pump_equilibrium_interval ?testid =(max ?start ?eqt) ?end)))
```

```
;;; DEFRULE EVENT_FIND_TURBINE_EQULIBRIUM_INTERVAL
;;; Finds 1st steady-state segment during which turbine should be in thermal
;;; equilibrium.
(defrule EVENT_find_turbine_equlibrium_interval
  (turbine_equilibrium_time ?eqt)
  (current_test|comparison_test ?testid)
  (F_THLEDE ?testid ?start ?end&:(< ?eqt ?end) ?)
  (not (F_THLEDE ?testid ?start2&:(< ?start2 ?start) ?end2&:(< ?eqt ?end2) ?))
  =>
  (assert (turbine_equilibrium_interval ?testid =(max ?eqt ?start) ?end)))

;;; ------------------------ FIND_EVENT_INTERVALS ----------------------
;;; Determine smallest time intervals between events.
;;; Rules work from the time of the earliest event towards the lastest event,
;;; incrementing by the smallest time interval between events.

;;; DEFRULE EVENT_FIND_FIRST_INTERVAL
;;; Finds the earliest event time.
(defrule EVENT_find_first_interval
  (current_phase find_event_intervals)
  ?f1 <- (event_time ?testid ?first)
  (not (event_time ?testid ?t2&:(< ?t2 ?first)))
  (not (current_time ?testid ?))
  =>
  (retract ?f1)
  (assert (current_time ?testid ?first)))

;;; DEFRULE EVENT_FIND_STEADY_STATE_EVENT_INTERVALS
;;; Finds the "next" event to increment to during steady-state.
(defrule EVENT_find_steady_state_event_intervals
  (current_phase find_event_intervals)
  ?f1 <- (current_time ?testid ?start)
  ?f2 <- (event_time ?testid ?end)
  (not (event_time ?testid ?t2&:(< ?t2 ?end)))
  (F_THLEDE ?testid ?plstart ?plend ?PL)
  (test (is_concurrent ?plstart ?plend ?start ?end))
  =>
  (if (is_true ?*DEBUG*) then (fprintout t "Event Interval " ?testid ": "
              ?start " - " ?end " @ " ?PL crlf))
  (assert (event_interval ?testid ?start ?end ?PL)
          (current_time ?testid ?end))
  (retract ?f1 ?f2))

;;; DEFRULE EVENT_FIND_TRANSIENT_STATE_EVENT_INTERVALS
;;; Finds the "next" event to increment to during power-level transient.
(defrule EVENT_find_transient_state_event_intervals
  (current_phase find_event_intervals)
  ?f1 <- (current_time ?testid ?start)
  ?f2 <- (event_time ?testid ?end)
  (not (event_time ?testid ?t2&:(< ?t2 ?end)))
  (not (F_THLEDE ?testid ?plstart
              ?plend&:(is_concurrent ?plstart ?plend ?start ?end) ?))
  =>
  (if (is_true ?*DEBUG*) then (fprintout t "Event Interval " ?testid ": "
              ?start " - " ?end " @ TRANSIENT" crlf))
  (assert (event_interval ?testid ?start ?end TRANSIENT)
          (current_time ?testid ?end))
  (retract ?f1 ?f2))
```

```
;;; ------------------------- FIND_ANOMALIES ----------------------------

;;; DEFRULE EVENT_COMBINE_ADJACENT_ANOMALIES
;;; If the same anomaly is detected in adjacent intervals, the anomaly
;;; descriptions are combined into one spanning the entire interval.
(defrule EVENT_combine_adjacent_anomalies
  (current_phase find_anomalies)
  ?f1 <- (anomaly (class ?num) (start ?start1&~nil) (end ?end1) (repeatable TRUE))
  ?f2 <- (anomaly (class ?num) (start ?start2&~nil&:(eqtime ?end1 ?start2))
                  (end ?end2))
  (test (neq ?f1 ?f2))
  =>
  (retract ?f2)
  (modify ?f1
     (end ?end2)))

;;; DEFRULE EVENT_REMOVE_REDUNDANT_ANOMALIES
;;; If an anomaly class is defined as non-repeatable, this removes redundant
;;; occurrences.
(defrule EVENT_remove_redundant_anomalies
  (current_phase find_anomalies)
  (anomaly (class ?num)(start ?start&~nil)(repeatable FALSE))
  ?f <- (anomaly (class ?num)(start ?start2&:(> ?start2 ?start)))
  =>
  (retract ?f))
```

FILE: HPOTP_anom.clp
Anomaly Detection Rules

```
;;; PTDS HPOTP Diagnostic Module
;;; HPOTP Anomalies
;;;
;;; Creation: 8/10/93  T.W. Bickmore

;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.05.1
;;; Source: SAIC final report, section 5.05
;;; Summary: Difference 327-328 is different between current and comparison tests.
;;; Status: Wilmer questions usefulness, keeping for now for evaluation.

(defrule anomaly5.05.1
  (current_phase find_anomalies)
  (current_test ?testid)
  (comparison_test ?comptestid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (F_DIFTHA ?testid "327 - 328"
            ?start2 ?end2&:(is_concurrent ?start ?end ?start2 ?end2)
           ?comptestid "327 - 328" $?)
  =>
  (assert (anomaly
            (class A5.05.1)
            (start ?start2)
            (end   ?end2)
            (priority 16)
            (description =(str-cat
                "The difference (327 - 328) is different at thrust level " ?PL
                " between this test and the previous.")))))

(deffacts init5.05.1
  (plot (class A5.05.1)(number 1)(shutdown_delta_end 100.0)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
  (plot (class A5.05.1)(number 2)(use_comparison TRUE)(shutdown_delta_end 100.0)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
  (plot (class A5.05.1)(number 3)(cross_comparison TRUE)(shutdown_delta_end 100.0)
        (PIDs "63")(title "Thrust Profile"))
)
```

```
;;;---------------------------------------------------------------------------
;;; Rule: anomaly5.06.1
;;; Source: SAIC final report, section 5.06
;;; Summary: Spike seen in 327(328), not in 328(327), and no level shift in 327,328,
;;; or 327-328.

(defrule anomaly5.06.1
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (or (and
        (confirmed|unconfirmed F_SPIKE ?testid ?sensor&"327" ?s1
              ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
        (not (confirmed|unconfirmed F_SPIKE ?testid "328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2)  $?)))
      (and
        (confirmed|unconfirmed F_SPIKE ?testid ?sensor&"328" ?s1
              ?e1&:(is_concurrent ?start ?end ?s1 ?e1) )
        (not (confirmed|unconfirmed F_SPIKE ?testid "327" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2)  $?))))
  (not (confirmed|unconfirmed F_LEVSH ?testid "327"|"328"|"327 - 328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  =>
  (assert (anomaly
            (class A5.06.1)
            (start ?start)
            (end ?end)
            (priority 16)
            (description  =(str-cat "Spike seen in sensor " ?sensor
                      " only, with no change in steady state "
            "pressures or pressure difference. Possible sensor or omni seal anomaly. "
                      "No real rotor motion."))))))

(deffacts init5.06.1
  (plot (class A5.06.1)(number 1)(anomaly_delta_start -5.0)
        (anomaly_delta_end +10.0)(full_sample TRUE)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
  (plot (class A5.06.1)(number 2)(shutdown_delta_end 100.0)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
  (plot (class A5.06.1)(number 3)(shutdown_delta_end 100.0)
        (PIDs "63")(title "Thrust Profile"))
)
```

```
;;;---------------------------------------------------------------------------
;;; Rule: anomaly5.06.2
;;; Source: SAIC final report, section 5.06
;;; Summary: Level shift seen in 327(328) and not in 328(327).
;;; Modifications:
;;; 12/15/93 Changed to eliminate cup washer as possible cause (from Wilmer).

(defrule anomaly5.06.2
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (or (and
        (confirmed|unconfirmed F_LEVSH ?testid ?sensor&"327" ?s1
              ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
        (not (confirmed|unconfirmed F_LEVSH ?testid "328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)))
      (and
        (confirmed|unconfirmed F_LEVSH ?testid ?sensor&"328" ?s1
              ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
        (not (confirmed|unconfirmed F_LEVSH ?testid  "327" ?start2
              ?end2&:(is_concurrent ?start ?end ?start2 ?end2) $?))))
  (confirmed|unconfirmed F_LEVSH ?testid "327 - 328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly
            (class 5.06.2)
            (start ?start)
            (end ?end)
            (priority 16)
            (description  =(str-cat "Level shift seen in " ?sensor
                      " only. Possible sensor problem, omni seal leakage problem. "
                      "No real rotor motion.")))))

(deffacts init5.06.2
  (plot_ditto A5.06.2 A5.06.1))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.06.3
;;; Source: SAIC final report, section 5.06
;;; Summary: Spike seen in 327 and 328, and level shift seen in 327-328.
;;; Modifications:
;;; 12/15/93 Changed to require +/- delta levelshift (from Wilmer).
;;; 1/5/94   Split into two rules (from Wilmer).

(defrule anomaly5.06.3A
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_SPIKE ?testid "327" ?s1
              ?e1&:(is_concurrent ?start ?end ?s1 ?e1) ?mag327)
  (confirmed|unconfirmed F_SPIKE ?testid "328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) ?mag328)
  (test (xor (> ?mag327 0.0)(> ?mag328 0.0)))
  (not (confirmed|unconfirmed F_LEVSH ?testid "327"|"328"| ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  (confirmed|unconfirmed F_LEVSH ?testid "327 - 328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly (class A5.06.3A)
                   (start ?start)
                   (end ?end)
                   (priority 18)
                   (description
              "Possible HPOTP momentary anomalous rotor motion."))))

(defrule anomaly5.06.3B
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_SPIKE ?testid "327" ?s1
              ?e1&:(is_concurrent ?start ?end ?s1 ?e1) ?mag327)
  (confirmed|unconfirmed F_SPIKE ?testid "328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) ?mag328)
  (test (not (xor (> ?mag327 0.0)(> ?mag328 0.0))))
  (not (confirmed|unconfirmed F_LEVSH ?testid "327"|"328"| ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  (confirmed|unconfirmed F_LEVSH ?testid "327 - 328" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly (class A5.06.3B)
                   (start ?start)
                   (end ?end)
                   (priority 18)
                   (description  =(str-cat
       "Possible HPOTP balance piston momentary shift in orifice position."))))

(deffacts init5.06.3
  (plot_ditto A5.06.3A A5.06.1)
  (plot_ditto A5.06.3B A5.06.1))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.06.4
;;; Source: SAIC final report, section 5.06
;;; Summary: Level shift seen in 327 and 328 (opposite directions), and in 327-328.
;;; Modifications:

(defrule anomaly5.06.4
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_LEVSH ?testid "327" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) ? ?delta327)
  (confirmed|unconfirmed F_LEVSH ?testid "328" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) ? ?delta328)
  (test (xor (> ?delta327 0.0)(> ?delta328 0.0)))
  (confirmed|unconfirmed F_LEVSH ?testid "327 - 328" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly (class A5.06.4)
                   (start ?start)
                   (end ?end)
                   (priority 18)
                   (description "Possible HPOTP anomalous rotor motion."))))

(deffacts init5.06.4
  (plot_ditto A5.06.4 A5.06.1))

;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.06.5
;;; Source: SAIC final report, section 5.06
;;; Summary: Level shift seen in 327 and 328 (same direction).

(defrule anomaly5.06.5
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_LEVSH ?testid "327" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) ? ?delta327)
  (confirmed|unconfirmed F_LEVSH ?testid "328" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) ? ?delta328)
  (test (not (xor (> ?delta327 0.0)(> ?delta328 0.0))))
  =>
  (assert (anomaly (class A5.06.5)
                   (start ?start)
                   (end ?end)
                   (priority 18)
                   (description
       "Possible HPOTP balance piston orifice position change."))))

(deffacts init5.06.5
  (plot_ditto A5.06.5 A5.06.1))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.06.7
;;; Source: SAIC final report, section 5.06
;;; Summary: Level shift seen in 327(328) and not in 328(327).

(defrule anomaly5.06.7
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (or (and (confirmed|unconfirmed F_LEVSH ?testid ?sensor&"327" ?s2
             ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
           (not (confirmed|unconfirmed F_LEVSH ?testid "328" ?s2
             ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)))
      (and (confirmed|unconfirmed F_LEVSH ?testid ?sensor&"328" ?s2
             ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
           (not (confirmed|unconfirmed F_LEVSH ?testid "327" ?s2
             ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))))
  =>
  (assert (anomaly (class A5.06.7)
                   (start ?start)
                   (end ?end)
                   (priority 16)
              (description =(str-cat "Statistically significant change in " ?sensor
        " but not in difference (327 - 328). Possible omni seal leakage."
       " No real rotor motion."))))))

(deffacts init5.06.7
  (plot_ditto A5.06.7 A5.06.1))

;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.06.9
;;; Source: SAIC final report, section 5.06
;;; Summary: Level shift seen in 327-328, but not in 327 or 328.

(defrule anomaly5.06.9
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid  ?start ?end ~TRANSIENT)
  (not (confirmed|unconfirmed F_LEVSH ?testid "327"|"328" ?s2
             ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  (confirmed|unconfirmed F_LEVSH ?testid  "327 - 328" ?s2
             ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly (class A5.06.9)
                   (start ?start)
                   (end ?end)
                   (priority 16)
              (description =(str-cat
      "Statistically significant change in difference (327 - 328)"
      " but not in individual sensors. Not anomalous; no real rotor motion."))))))

(deffacts init5.06.9
  (plot_ditto A5.06.9 A5.06.1))
```

```
;;;---------------------------------------------------------------------------
;;; Rule: anomalyRotorDrag1, anomalyRotorDrag1
;;; Source: Wilmer
;;; Summary:
;;; 1. Concurrent: Significant increase in 328, decrease in 327, LOX in P is flat
;;;    Following increase in PL or decrease in LOX in P
;;;    Duration of more than 10 seconds (or duration of current power level, if less).
;;; 2. Opposite of above case.

(deffunction rate_of_change (?time1 ?time2 ?value1 ?value2)
  (/ (- ?value2 ?value1)(- ?time2 ?time1)))


(defrule anomalyRotorDrag1
  (current_test ?testid)
  ;;Check slope of difference 327-328:
  (DSEGMENT ?testid "327" ?testid "328" ?stD ?etD ?svD ?evD)
  (test (< (rate_of_change ?stD ?etD ?svD ?evD) -0.2))

  ;;Check slopes of 327 & 328 individually:
  (LINE ?testid "327" ?st327 ?et327 ? ?slope327&:(< ?slope327 -0.08))
  (test (is_concurrent ?stD ?etD ?st327 ?et327))
  (LINE ?testid "328" ?st328 ?et328 ? ?slope328&:(> ?slope328 0.08))
  (test (and (is_concurrent ?stD ?etD ?st328 ?et328)
             (> (abs (- ?slope327 ?slope328)) 0.2)))

  ;;Make sure LOX vent is flat:
  (confirmed|unconfirmed SEGMENT ?testid "ENGONPSP"|"858"|"859"|"860"
       ?stv ?etv ?svv ?evv)
  (test (< (abs (rate_of_change ?stv ?etv ?svv ?evv)) 0.2))
  (test (is_concurrent ?stD ?etD ?stv ?etv))

  ;;Check duration of event:
  (F_THLEDE ?testid ?stp ?etp ?newPL)
  (test (and (is_concurrent ?stp ?etp ?stD ?etD)
             (>= (- ?etD ?stD)
                 (min 10.0 (max 4.0 (- (- ?etp ?stp) 4.0))))))

  ;;Check to see if following the appropriate event:
  (or (and                    ;;Following increase in PL
        (F_THLEDE ?testid ?stprev
              ?etprev&:(<= ?etprev ?stp) ?oldPL&:(> ?newPL ?oldPL))
        (not (F_THLEDE ?testid ?stx
              ?etx&:(and (<= ?etx ?stp)(>= ?stx ?etprev)) ?)))

       ;;1st power level.
      (not (F_THLEDE ?testid ?stprev&:(< ?stprev ?stp) $?))

      (and                                          ;;Following Vent
        (confirmed|unconfirmed SEGMENT ?testid "ENGONPSP"|"858"|"859"|"860"
              ?stprev ?etprev&:(eqtime ?etprev ?stv) ?svprev ?evprev)
        (test (< (rate_of_change ?stprev ?etprev ?svprev ?evprev) -0.1))))
  =>
  (bind ?start (max ?stD ?stv))
  (bind ?end   (min ?etD ?etv))
  (assert (anomaly (class ARotorDrag)
              (start ?start)
              (end ?end)
              (priority 18)
              (description "Possible rotor drag."))))
```

```
(defrule anomalyRotorDrag2
  (current_test ?testid)
  (DSEGMENT ?testid "327" ?testid "328" ?stD ?etD ?svD ?evD)
  (test (> (rate_of_change ?stD ?etD ?svD ?evD) 0.2))

  (LINE ?testid "327" ?st327 ?et327 ? ?slope327&:(> ?slope327 0.08))
  (test (is_concurrent ?stD ?etD ?st327 ?et327))

  (LINE ?testid "328" ?st328 ?et328 ? ?slope328&:(< ?slope328 -0.08))
  (test (and (is_concurrent ?stD ?etD ?st328 ?et328)
             (> (abs (- ?slope327 ?slope328)) 0.2)))

  (confirmed|unconfirmed SEGMENT ?testid "ENGONPSP"|"858"|"859"|"860"
       ?stv ?etv ?svv ?evv)
  (test (< (abs (rate_of_change ?stv ?etv ?svv ?evv)) 0.2))
  (test (is_concurrent ?stD ?etD ?stv ?etv))

  (F_THLEDE ?testid ?stp ?etp ?newPL)
  (test (and (is_concurrent ?stp ?etp ?stD ?etD)
             (>= (- ?etD ?stD)
                 (min 10.0 (max 4.0 (- (- ?etp ?stp) 4.0))))))

  (or (and                      ;;Following decrease in PL
        (F_THLEDE ?testid ?stprev
              ?etprev&:(<= ?etprev ?stp) ?oldPL&:(< ?newPL ?oldPL))
        (not (F_THLEDE ?testid ?stx
              ?etx&:(and (<= ?etx ?stp)(>= ?stx ?etprev)) ?)))

      (and                                          ;;Following Vent
        (confirmed|unconfirmed SEGMENT ?testid "ENGONPSP"|"858"|"859"|"860"
              ?stprev ?etprev&:(eqtime ?etprev ?stv) ?svprev ?evprev)
       (test (> (rate_of_change ?stprev ?etprev ?svprev ?evprev) 0.1))))
  =>
  (bind ?start (max ?stD ?stv))
  (bind ?end   (min ?etD ?etv))
  (assert (anomaly (class ARotorDrag)
              (start ?start)
              (end ?end)
                    (priority 18)
              (description "Possible rotor drag."))))


;;; To Do (Wilmer)...
;;; + also show difference 327-328, zoom in
;;; + show 858 if following vent
(deffacts initRotorDrag
  (plot (class ARotorDrag)(number 1)(anomaly_delta_start -5.0)(anomaly_delta_end
+10.0)(full_sample TRUE)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
  (plot (class ARotorDrag)(number 2)(shutdown_delta_end 100.0)
        (PIDs "327" "328")(title "HPOTP Balance Piston Pressures"))
  (plot (class ARotorDrag)(number 3)(shutdown_delta_end 100.0)
        (PIDs "63")(title "Thrust Profile"))
)
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.07.1
;;; Source: SAIC final report, section 5.07
;;; Summary: PBP bistability detection. Just reports result from C routine.
;;; Modifications:
;;; 1/5/94 Re-wrote bistability routine to use GreenRun (Rktdyn) as baseline, plus
;;; check against OPOV for system response.

(defrule anomaly5.07.1
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_GREENBISTAB ?testid ?start ?end)
  (F_THLEDE ?testid ?s1 ?e1 ?PL)
  (test (is_concurrent ?start ?end ?s1 ?e1))
  =>
  (assert (anomaly (class A5.07.1)
                   (start ?start)
                   (end ?end)
                   (priority 18)
              (description =(str-cat "PBP bistability at thrust level " ?PL "."))))))

;;; To Do (Wilmer)...
;;; + show 3sigma noise floor
;;; + furball chart (341 - 334) vs. 163 Pc (x)
(deffacts init5.07.1
  (plot (class A5.07.1)(number 1)(full_sample TRUE)
        (anomaly_delta_start -5.0)(anomaly_delta_end +5.0)
        (PIDs "59")(title "Preburner Pump Discharge Pressure Ch B"))
  (plot (class A5.07.1)(number 2)(full_sample TRUE)
        (anomaly_delta_start -5.0)(anomaly_delta_end +5.0)
        (PIDs "176")(title "OPOV Command"))
  (plot (class A5.07.1)(number 3)(full_sample TRUE)
        (anomaly_delta_start -5.0)(anomaly_delta_end +5.0)
        (PIDs "63")(title "Thrust Profile"))
)
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.08.1
;;; Source: SAIC final report, section 5.08
;;; Summary: Erratic 990, 1190 not erratic or spiking.
;;; Modifications:
;;;  1/6/94 Changed to check against power level interval per Wilmer.

(defrule anomaly5.08.1
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (confirmed|unconfirmed F_ERRAT ?testid "990" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1))
  (not (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1190" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  (not (anomaly (class A5.08.1)(start ?s2)
              (end ?e2&:(is_concurrent ?start ?end ?s2 ?e2))))
  =>
  (assert (anomaly (class A5.08.1)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
              "HPOTP erratic primary turbine seal drain pressure may "
                     "indicate sensor problem or seal anomaly. "
                     "No effect seen in drain temperature."))))))

(deffacts init5.08.1
  (plot (class A5.08.1)(number 1)
        (PIDs "990")(title "HPOT Primary Turbine Seal Drain Pr"))
  (plot (class A5.08.1)(number 2)
        (PIDs "1190")(title "HPOT Primary Turbine Seal Drain Temp"))
  (plot (class A5.08.1)(number 3)
        (PIDs "233" "234" "518" "519" "521" "522")
       (title "HPOT Turbine Discharge Temp")
       (choose_redundant TRUE))
)
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.08.2
;;; Source: SAIC final report, section 5.08
;;; Summary: 1190 erratic, 990 not erratic or spiking.
;;; Modifications:
;;;  1/6/94 Changed to check against power level interval per Wilmer.

(defrule anomaly5.08.2
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (confirmed|unconfirmed F_ERRAT ?testid "1190" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1))
  (not (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "990" ?s2
            ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  (not (anomaly (class A5.08.2)(start ?s2)
            (end ?e2&:(is_concurrent ?start ?end ?s2 ?e2))))
  =>
  (assert (anomaly (class A5.08.2)
                   (start ?start)
                   (end ?end)
                   (priority 16)
              (description =(str-cat
       "HPOTP erratic primary turbine seal drain temperature may "
                     "indicate sensor problem or seal anomaly. "
                     "No effect seen in drain pressure."))))))

(deffacts init5.08.2
  (plot_ditto A5.08.2 A5.08.1))

;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.08.3
;;; Source: SAIC final report, section 5.08
;;; Summary: 990 erratic or spiking, and 1190 erratic or spiking.

(defrule anomaly5.08.3
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "990" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1190" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly (class A5.08.3)
                   (start ?start)
                   (end ?end)
                   (priority 16)
              (description =(str-cat
       "HPOTP shows concurrent jitter in both primary turbine seal drain "
                     "pressure and temperature. Possible seal anomaly."))))))

(deffacts init5.08.3
  (plot_ditto A5.08.3 A5.08.1))
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.08.4
;;; Source: Wilmer
;;; Summary:  Erratic or spiking 990 & 1190 in same power-level interval, but not
;;; concurrently.

(defrule anomaly5.08.4
  (declare (salience -5))
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "990" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1190" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  (not (anomaly (class A5.08.3|A5.08.4)(start ?s3)
              (end ?e3&:(is_concurrent ?start ?end ?s3 ?e3))))
  =>
  (assert (anomaly (class A5.08.4)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
       "HPOTP shows non-concurrent jitter in both primary turbine seal drain "
                     "pressure and temperature. Possible seal anomaly."))))))

(deffacts init5.08.4
  (plot_ditto A5.08.4 A5.08.1))
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly990shift
;;; Source: Wilmer
;;; Summary: 990 is low (or high) in peak and equilibrium values relative to family.
;;;          Primary turbine seal drain pressure.

(defrule anomalyA990shift
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "990" ?param preferred)
  (out_of_family ?testid "PEAK_HEIGHT" ?param ?value ?pk-n-sig)
  (out_of_family ?testid "EQ_VAL"  ?param ?value ?eq-n-sig)
  (turbine_equlibrium_interval ?testid ?start ?end)
  =>
  (assert (anomaly (class A990shift)
                   (start 0.0)
                   (end   ?end)
                   (priority 16)
                 (append_time FALSE)
                (description =(str-cat "HPOTP primary turbine seal drain pressure is "
                       ?value
                       " in peak ("
                      (format nil "%0.2f" ?pk-n-sig )
                      " sigma) and equilibrium ("
                      (format nil "%0.2f" ?eq-n-sig )
                      " sigma) values compared to historical statistics. "
                      "May be change in seal clearance or sensor calibration."))
                (repeatable FALSE))))

(deffacts initA990shift
  (plot (class A990shift)(number 1)
        (PIDs "990")(title "HPOT Primary Turbine Seal Drain Pr"))
  (plot (class A990shift)(number 2)
       (PIDs "1190")(title "HPOT Primary Turbine Seal Drain Temp"))
  (plot (class A990shift)(number 3)
        (PIDs "233" "234" "518" "519" "521" "522")
       (title "HPOT Turbine Discharge Temp")
       (choose_redundant TRUE)))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly990peakshift
;;; Source: Wilmer
;;; Summary:  990 peak value is out-of-family, but equilibrium value is OK.

(defrule anomalyA990peakshift
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "990" ?param preferred)
  (out_of_family ?testid "PEAK_HEIGHT" ?param ?value ?n-sig)
  (not (out_of_family ?testid "EQ_VAL"  ?param ?value ?))
  =>
  (assert (anomaly (class A990peakshift)
                   (start 0.0)
                   (end   50.0)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat
              "HPOTP primary turbine seal drain pressure peak is "
                      ?value
                     " ("
                     (format nil "%0.2f" ?n-sig )
                     " sigma)"
                      " compared to historical statistics. "
                     "May be change in seal clearance or sensor calibration."))
                (repeatable FALSE))))

(deffacts initA990peakshift
  (plot_ditto A990peakshift A990shift))

;;;-----------------------------------------------------------------------------
;;; Rule: anomaly990peakshift
;;; Source: Wilmer
;;; Summary:  990 equilibrium value is out-of-family, but peak value is OK.

(defrule anomaly990eqshift
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "990" ?param preferred)
  (out_of_family ?testid "EQ_VAL"  ?param ?value ?n-sig)
  (not (out_of_family ?testid "PEAK_HEIGHT" ?param ?value ?))
  (turbine_equlibrium_interval ?start ?end)
  =>
  (assert (anomaly (class A990eqshift)
                   (start 0.0)
                   (end   ?end)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat
       "HPOTP primary turbine seal drain pressure equilibrium value is "
                      ?value
                     " ("
                     (format nil "%0.2f" ?n-sig)
                     " sigma)"
                      " compared to historical statistics. "
                     "May be change in seal clearance or sensor calibration."))
                (repeatable FALSE))))

(deffacts initA990eqshift
  (plot_ditto A990eqshift A990shift))
```

```
;;; Rule: anomaly91shift
;;; Source: Wilmer
;;; Summary: 91 or 92 is low (or high) in peak and equilibrium values relative to
;;; family. Secondary turbine seal cavity pressure.
(defrule anomalyA91shift
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "91"|"92" ?param preferred)
  (out_of_family ?testid "PEAK_HEIGHT" ?param ?value ?pk-n-sig)
  (out_of_family ?testid "EQ_VAL"  ?param ?value ?eq-n-sig)
  (turbine_equlibrium_interval ?testid ?start ?end)
  =>
  (assert (anomaly (class A91shift)
                   (start 0.0)
                   (end   ?end)
                   (priority 16)
              (append_time FALSE)
               (description =(str-cat
              "HPOTP secondary turbine seal cavity pressure is "
                       ?value
                       " in peak ("
                      (format nil "%0.2f" ?pk-n-sig)
                      ") and equilibrium ("
                      (format nil "%0.2f" ?eq-n-sig)
                      ") values compared to historical statistics. "
                      "May be change in seal clearance or sensor calibration."))
               (repeatable FALSE))))

(deffacts initA91shift
  (plot (class A91shift)(number 1)(choose_redundant TRUE)
        (PIDs "91" "92")(title "HPOT Secondary Turbine Seal Cavity Pr"))
  (plot (class A91shift)(number 2)
       (PIDs "1188")(title "HPOTP Sec Turbine Seal Drain Temp"))
  (plot (class A91shift)(number 3)
        (PIDs "63")(title "Thrust Profile")))

;;;-------------------------------------------------------------------------------
;;; Rule: anomaly91peakshift
;;; Source: Wilmer
;;; Summary: 91/92 peak value is out-of-family, but equilibrium value is OK.
(defrule anomalyA91peakshift
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "91"|"92" ?param preferred)
  (out_of_family ?testid "PEAK_HEIGHT" ?param ?value ?n-sig)
  (not (out_of_family ?testid "EQ_VAL"  ?param ?value ?))
  =>
  (assert (anomaly (class A91peakshift)
                   (start 0.0)
                   (end   50.0)
                   (priority 16)
              (append_time FALSE)
               (description =(str-cat
              "HPOTP secondary turbine seal cavity pressure peak is "
                       ?value
                      "( "
                      (format nil "%0.2f" ?n-sig)
                       " sigma) compared to historical statistics. "
                      "May be change in seal clearance or sensor calibration."))
               (repeatable FALSE))))

(deffacts initA91peakshift
  (plot_ditto A91peakshift A91shift))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly91eqshift
;;; Source: Wilmer
;;; Summary: 91/92 equilibrium value is out-of-family, but peak value is OK.

(defrule anomaly91eqshift
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "91"|"92" ?param preferred)
  (out_of_family ?testid "EQ_VAL"  ?param ?value ?n-sig)
  (not (out_of_family ?testid "PEAK_HEIGHT" ?param ?value ?))
  (turbine_equlibrium_interval ?start ?end)
  =>
  (assert (anomaly (class A91eqshift)
                   (start 0.0)
                   (end   ?end)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat
       "HPOTP secondary turbine seal cavity pressure equilibrium value is "
                     ?value
                    " ("
                    (format nil "%0.2f" ?n-sig)
                     " sigma) compared to historical statistics. "
                    "May be change in seal clearance or sensor calibration."))
                (repeatable FALSE))))

(deffacts initA91eqshift
  (plot_ditto A91eqshift A91shift))

;;;-----------------------------------------------------------------------------
;;; Rule: anomaly91peakwidth
;;; Source: Wilmer
;;; Summary:  91/92 peak width is out-of-family.
;;;   Wilmer questions usefullness, but will keep in for now.

(defrule anomaly91peakwdith
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "91"|"92" ?param preferred)
  (out_of_family ?testid "PEAK_WIDTH"  ?param ?value ?n-sig)
  =>
  (assert (anomaly (class A91peakwidth)
                   (start 0.0)
                   (end   50.0)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat
              "HPOTP secondary turbine seal cavity pressure peak width is "
                     ?value
                    " ("
                    (format nil "%0.2f" ?n-sig)
                     " sigma) compared to historical statistics. "))
                (repeatable FALSE))))

(deffacts initA91peakwidth
  (plot_ditto A91peakwidth A91shift))
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly990peakwidth
;;; Source: Wilmer
;;; Summary:  990 peak width is out-of-family.
;;;   Wilmer questions usefulness.

(defrule anomaly990peakwdith
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "990" ?param preferred)
  (out_of_family ?testid "PEAK_WIDTH"  ?param ?value ?n-sig)
  =>
  (assert (anomaly (class A990peakwidth)
                   (start 0.0)
                   (end   50.0)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat
              "HPOTP primary turbine seal drain pressure peak width is "
                       ?value
                      " ("
                       (format nil "%0.2f" ?n-sig)
                        " sigma) compared to historical statistics. "))
                (repeatable FALSE))))

(deffacts initA990peakwidth
  (plot_ditto A990peakwidth A91shift))

;;;-------------------------------------------------------------------------------
;;; Rule: anomaly91peaktime
;;; Source: Wilmer
;;; Summary:  91/92 peak time is out-of-family.

(defrule anomaly91peaktime
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "91"|"92" ?param preferred)
  (out_of_family ?testid "PEAK_TIME"  ?param ?value ?n-sig)
  =>
  (assert (anomaly (class A91peaktime)
                   (start 0.0)
                   (end   50.0)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat
              "HPOTP secondary turbine seal cavity pressure peak time is "
                       ?value
                      " ("
                       (format nil "%0.2f" ?n-sig)
                        " sigma) compared to historical statistics. "))
                (repeatable FALSE))))

(deffacts initA91peaktime
  (plot_ditto A91peaktime A91shift))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly990peaktime
;;; Source: Wilmer
;;; Summary:  990 peak time is out-of-family.

(defrule anomaly990peaktime
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "990" ?param preferred)
  (out_of_family ?testid "PEAK_TIME"  ?param ?value ?n-sig)
  =>
  (assert (anomaly (class A990peaktime)
                   (start 0.0)
                   (end   50.0)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat
              "HPOTP primary turbine seal drain pressure peak time is "
                       ?value
                      " ("
                       (format nil "%0.2f" ?n-sig)
                        " sigma) compared to historical statistics. "))
                (repeatable FALSE))))

(deffacts initA990peaktime
  (plot_ditto A990peaktime A91shift))


;;;-----------------------------------------------------------------------------
;;; Rule: anomalyIMSLstart
;;; Source: Wilmer
;;; Summary:  START value of 211/212 is out-of-family.
;;; To Do...
;;; + check for IMSL greenrun compliance and report together (Wilmer).

(defrule anomalyIMSLstart
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "211"|"212" ?param preferred)
  (out_of_family ?testid "START_VAL" ?param ?value ?n-sig)
  =>
  (assert (anomaly (class AIMSLStart)
                   (start -1.0)
                   (end   0.0)
                   (priority 16)
              (append_time FALSE)
                (description =(str-cat "HPOTP intermediate seal purge pressure is "
                       ?value
                      " ("
                       (format nil "%0.2f" ?n-sig)
                        " sigma) at START compared to historical statistics. "))
                (repeatable FALSE))))

(deffacts initAIMSLStart
  (plot (class AIMSLStart)(number 1)(PIDs "211" "212")
       (title "HPOTP Int Seal Purge Pr")
       (anomaly_delta_start -5.0)(anomaly_delta_end +5.0)(full_sample TRUE))
  (plot (class AIMSLStart)(number 2)(PIDs "937")(title "Engine Helium Interface Pr")
       (anomaly_delta_start -5.0)(anomaly_delta_end +5.0)(full_sample TRUE))
  (plot (class AIMSLStart)(number 3)(PIDs "211" "212")
       (title "HPOTP Int Seal Purge Pr")))
```

```
;;;---------------------------------------------------------------------------
;;; Rule: anomalyBalPistonFamily1
;;; Source: Wilmer
;;; Summary: 327 is out-of-family (at 109MAX, 104MIN, or 104Nominal NPSP).

(defrule anomalyBalPistonFamily1
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "327" ?p327 ?)
  (sensor_status ?testid "328" ?p328 ?)
  (shutdown_time ?testid ?shut)
  (out_of_family ?testid ?type&"104_MIN_NPSP"|"109_MAX_NPSP"|"104_NOM_NPSP" ?p327
              ?value ?n-sig)
  (not (out_of_family ?testid ?type ?p328 ? ?))
  =>
  (assert (anomaly (class ABalPFamily1)
              (start 0)
              (end ?shut)
              (priority 16)
              (append_time FALSE)
              (repeatable FALSE)
              (description =(str-cat "HPOTP balance cavity pressure A is "
                     ?value
                     " ("
                     (format nil "%0.2f" ?n-sig)
                     " sigma) at "
                     ?type
       " compared to historical statistics. B channel is within limits."))))))

(deffacts initBalPistonFam1
  (plot (class ABalPFamily1)(number 1)(PIDs "327" "328")
       (title "HPOTP Balance Cavity Pressures"))
  (plot (class ABalPFamily1)(number 2)(PIDs "63")
       (title "Thrust Profile"))
  (plot (class ABalPFamily1)(number 3)(PIDs "858" "859" "860")
       (title "Engine LOX Inlet Pressure")))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomalyBalPistonFamily2
;;; Source: Wilmer
;;; Summary: 328 is out-of-family (at 109MAX, 104MIN, or 104Nominal NPSP).

(defrule anomalyBalPistonFamily2
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "327" ?p327 ?)
  (sensor_status ?testid "328" ?p328 ?)
  (shutdown_time ?testid ?shut)
  (out_of_family ?testid ?type&"104_MIN_NPSP"|"109_MAX_NPSP"|"104_NOM_NPSP" ?p328
              ?value ?n-sig)
  (not (out_of_family ?testid ?type ?p327 ? ?))
  =>
  (assert (anomaly (class ABalPFamily2)
              (start 0)
              (end ?shut)
              (priority 16)
              (append_time FALSE)
              (repeatable FALSE)
              (description =(str-cat "HPOTP balance cavity pressure B is "
                     ?value
                     " ("
                     (format nil "%0.2f" ?n-sig)
                     " sigma) at "
                     ?type
       " compared to historical statistics. A channel is within limits."))))))

(deffacts initBalPistonFam2
  (plot_ditto ABalPFamily2 ABalPFamily1))
```

```
;;; Rule: anomalyBalPistonFamily3
;;; Source: Wilmer
;;; Summary: 327 and 328 are both out-of-family (at 104MIN, 109Max, or 104Nominal
;;; NPSP).
(defrule anomalyBalPistonFamily3
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "327" ?p327 ?)
  (sensor_status ?testid "328" ?p328 ?)
  (shutdown_time ?testid ?shut)
  (out_of_family ?testid ?type&"104_MIN_NPSP"|"109_MAX_NPSP"|"104_NOM_NPSP" ?p328
            ?v328 ?n328)
  (out_of_family ?testid ?type ?p327 ?v327 ?n327)
  =>
  (assert (anomaly (class ABalPFamily3)
            (start 0)
            (end ?shut)
            (priority 16)
            (append_time FALSE)
            (repeatable FALSE)
            (description =(str-cat "HPOTP balance cavity pressure A is "
                  ?v327
                  " ("
                  (format nil "%0.2f" ?n327)
                  " sigma) and channel B is "
                  ?v328
                  " ("
                  (format nil "%0.2f" ?n328)
                  " sigma) at "
                  ?type
                  " compared to historical statistics."))))))

(deffacts initBalPistonFam3
  (plot_ditto ABalPFamily3 ABalPFamily1))

;;;---------------------------------------------------------------------------------
;;; Rule: anomalyLOXSlPFamily
;;; Source: Wilmer
;;; Summary: 951|952|953 are out-of-family.
(defrule anomalyLOXSlPFamily
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "951" ?param ?)
  (shutdown_time ?testid ?shut)
  (out_of_family ?testid "5_TO_CUT" ?param ?value ?n-sig)
  (sensor_status ?testid "1187" ?param1187 ?)
  (not (out_of_family ?testid "MAX_AFTER_EQ" ?param1187 LOW ?n&:(eq ?value HIGH)))
  =>
  (assert (anomaly (class ALOXSlPFamily)
            (start 0)
            (end ?shut)
            (priority 16)
            (append_time FALSE)
            (repeatable FALSE)
            (description =(str-cat "HPOTP primary pump seal drain pressure is "
                  ?value
                  " ("
                  (format nil "%0.2f" ?n-sig)
      " sigma) from 5 seconds to cutoff compared to historical statistics."))))))
(deffacts initLOXSlPFamily
  (plot (class ALOXSlPFamily)(number 1)(PIDs "951" "952" "953")
       (title "HPOTP Primary Pump Seal Drain Press"))
  (plot (class ALOXSlPFamily)(number 2)(PIDs "63")
       (title "Thrust Profile")))
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomalyLOXS1TFamily
;;; Source: Wilmer
;;; Summary: 1187 is out-of-family.

(defrule anomalyLOXS1TFamily
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "1187" ?param ?)
  (shutdown_time ?testid ?shut)
  (out_of_family ?testid "MAX_AFTER_EQ" ?param ?value ?n-sig)
  (sensor_status ?testid "951" ?param951 ?)
  (not (out_of_family ?testid "5_TO_CUT" ?param951 HIGH ?n&:(eq ?value LOW)))
  =>
  (assert (anomaly (class ALOXS1TFamily)
              (start 0)
              (end ?shut)
              (priority 16)
              (append_time FALSE)
              (repeatable FALSE)
              (description =(str-cat
              "HPOTP primary pump seal drain temperature maximum is "
                     ?value
                     " ("
                     (format nil "%0.2f" ?n-sig)
                     " sigma) compared to historical statistics."))))))

(deffacts initLOXS1TFamily
  (plot (class ALOXS1TFamily)(number 1)(PIDs "1187")
       (title "HPOTP Primary Pump Seal Drain Temp"))
  (plot (class ALOXS1TFamily)(number 2)(PIDs "63")
       (title "Thrust Profile")))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomalySlingerProblem
;;; Source: Wilmer
;;; Summary: 1187 is out-of-family low and 951/952/953 is out-of-family high.

(defrule anomalySlingerProblem
  (current_phase find_anomalies)
  (current_test ?testid)
  (sensor_status ?testid "1187" ?param1187 ?)
  (shutdown_time ?testid ?shut)
  (out_of_family ?testid "MAX_AFTER_EQ" ?param1187 LOW ?n-sig-1187)
  (sensor_status ?testid "951" ?param951 ?)
  (out_of_family ?testid "5_TO_CUT" ?param951 HIGH ?n-sig-951)
  =>
  (assert (anomaly (class ASlingerProblem)
              (start 0)
              (end ?shut)
              (priority 16)
              (append_time FALSE)
              (repeatable FALSE)
              (description =(str-cat
              "HPOTP primary pump seal drain temperature maximum is LOW ("
                     (format nil "%0.2f" ?n-sig-1187)
                     " sigma) and HPOTP primary pump seal drain pressure is HIGH ("
                     (format nil "%0.2f" ?n-sig-951)
       ") compared to historical statistics. Indicates possible slinger problem."))))))

(deffacts initSlingerProblem
  (plot (class ASlingerProblem)(number 1)(PIDs "1187")
       (title "HPOTP Primary Pump Seal Drain Temp"))
  (plot (class ASlingerProblem)(number 2)(PIDs "951" "952" "953")
       (title "HPOTP Primary Pump Seal Drain Press"))
  (plot (class ASlingerProblem)(number 3)(PIDs "63")
       (title "Thrust Profile")))

;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.09.6
;;; Source: SAIC final report, section 5.09
;;; Summary: Could not compute a peak for 990.

(defrule anomaly5.09.6
  (current_phase find_anomalies)
  (current_test ?testid)
  (not (confirmed|unconfirmed F_PEAK ?testid "990" $?))
  =>
  (assert (anomaly (class A5.09.6)
                   (repeatable FALSE)
                   (priority 6)
              (type OBSERVATION)
              (append_time FALSE)
                (description =(str-cat
       "Current test HPOTP primary turbine seal drain "
                    "pressure peak missing."))))))

(deffacts init5.09.6
  (plot (class A5.09.6)(number 1)(end 50.0)
        (PIDs "990")(title "HPOT Primary Turbine Seal Drain Pr"))
)
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.09.12
;;; Source: SAIC final report, section 5.09
;;; Summary: Could not compute a peak for 91/92.

(defrule anomaly5.09.12
  (current_phase find_anomalies)
  (current_test ?testid)
  (not (confirmed|unconfirmed F_PEAK ?testid "91"|"92" $?))
  =>
  (assert (anomaly (class A5.09.12)
                   (priority 6)
                   (repeatable FALSE)
              (type OBSERVATION)
              (append_time FALSE)
                (description =(str-cat
              "Current test HPOTP secondary turbine seal cavity "
                   "pressure peak missing.")))))

(deffacts init5.09.12
  (plot (class A5.09.12)(number 1)(choose_redundant TRUE)
        (PIDs "91" "92")(title "HPOT Sec Turbine Seal Drain Pr"))
)

;;;-----------------------------------------------------------------------------
;;; Rule: anomaly5.12.1
;;; Source: SAIC final report, section 5.12
;;; Summary: 91/92 is erratic, 1188 is normal.
;;; Modifications:
;;;  1/6/94 - Changed to check PL interval (Wilmer).

(defrule anomaly5.12.1
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (?status&confirmed|unconfirmed F_ERRAT ?testid ?pid&"91"|"92" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1))
  (not (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1188" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  =>
  (if (eq ?status confirmed) then
      (bind ?text " seal anomaly (seen in both channels).")
  else
      (bind ?text " sensor problem or seal anomaly (only seen in one channel)."))
  (assert (anomaly (class A5.12.1)
                   (start ?start)
                   (end ?end)
                   (priority 16)
              (description =(str-cat
       "HPOTP erratic secondary turbine seal drain pressure may indicate "
                     ?text
                       " No effect seen in drain temperature.")))))

(deffacts init5.12.1
  (plot (class A5.12.1)(number 1)(choose_redundant TRUE)
        (PIDs "91" "92")(title "HPOT Secondary Seal Cavity Pr"))
  (plot (class A5.12.1)(number 2)
        (PIDs "1188")(title "HPOTP Secndary Turbine Seal Drain Temp"))
  (plot (class A5.12.1)(number 3)
        (PIDs "63")(title "Thrust Profile"))
)
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.12.2
;;; Source: SAIC final report, section 5.12
;;; Summary: 1188 is erratic, 91/92 is normal.
;;; Modifications:
;;;  1/6/94 - Changed to check PL interval (Wilmer).

(defrule anomaly5.12.2
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (confirmed|unconfirmed F_ERRAT ?testid "1188" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1))
  (not (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "91"|"92" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  =>
  (assert (anomaly (class A5.12.2)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
              "HPOTP erratic secondary turbine seal drain temperature may "
                      "indicate sensor problem or seal anomaly. No "
                      "effect seen in drain pressure."))))))

(deffacts init5.12.2
  (plot_ditto A5.12.2 A5.12.1))

;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.12.3
;;; Source: SAIC final report, section 5.12
;;; Summary: 91/92 and 1188 are both erratic or spiking. Check for concurrent
;;; anomalies.

(defrule anomaly5.12.3
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "91"|"92" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1188" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly (class A5.12.3)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
       "HPOTP shows concurrent jitter in both secondary turbine seal drain "
                       "pressure and temperature. Possible seal anomaly."))))))

(deffacts initA5.12.3
  (plot_ditto A5.12.3 A5.12.1))
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.12.4
;;; Source: Wilmer
;;; Summary: 91/92 and 1188 are both erratic or spiking. Check for non-concurrent
;;; anomalies.
(defrule anomaly5.12.4
  (declare (salience -10))
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "91"|"92" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1188" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  (not (anomaly (class anomaly5.12.3|anomaly5.12.4)(start ?s3)
              (end ?e3&:(is_concurrent ?start ?end ?s3 ?e3))))
  =>
  (assert (anomaly (class A5.12.4)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
       "HPOTP shows non-concurrent jitter in both secondary turbine seal drain "
                     "pressure and temperature. Possible seal anomaly."))))))

(deffacts initA5.12.4
  (plot_ditto A5.12.4 A5.12.1))

;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.15.1
;;; Source: SAIC final report, section 5.15
;;; Summary: 951/952/953 is erratic, 1187 is normal.
;;; Modifications:
;;; 1/6/94 - Changed to check w/in power level (Wilmer).
(defrule anomaly5.15.1
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (?status&confirmed|unconfirmed F_ERRAT ?testid "951"|"952"|"953" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1))
  (not (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1187" ?s2
             ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  =>
  (if (eq ?status confirmed) then
     (bind ?text "seal anomaly (confirmed by two or more sensors)")
  else
     (bind ?text "sensor problem or seal anomaly (only reported by one sensor)."))
  (assert (anomaly (class A5.15.1)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
               "HPOTP erratic primary pump seal drain pressure may indicate "
                     ?text ". No effect seen in drain temperature."))))))

(deffacts init5.15.1
  (plot (class A5.15.1)(number 1)
        (PIDs "951" "952" "953")(title "HPOTP Primary Pump Seal Drain Pr"))
  (plot (class A5.15.1)(number 2)
        (PIDs "1187")(title "HPOTP Primary Pump Seal Drain Temp"))
  (plot (class A5.15.1)(number 3)
        (PIDs "63")(title "Thrust Profile")))
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.15.2
;;; Source: SAIC final report, section 5.15
;;; Summary: 1187 is erratic, 951/952/953 are normal.
;;; Modifications:
;;; 1/6/94 - Changed to check w/in power level (Wilmer).

 (defrule anomaly5.15.2
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (confirmed|unconfirmed F_ERRAT ?testid "1187" ?s1
              ?e1&:(is_concurrent ?start ?end ?s1 ?e1))
  (not (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "951"|"952"|"953" ?s2
              ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?))
  =>
  (assert (anomaly (class A5.15.2)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
       "HPOTP erratic primary pump seal drain temperature may indicate "
                        "sensor problem or seal anomaly.  No effect seen "
                        "in drain pressure."))))))

(deffacts init5.15.2
  (plot (class A5.15.2)(number 1)
        (PIDs "1187")(title "HPOTP Primary Pump Seal Drain Temp"))
  (plot (class A5.15.2)(number 2)(choose_redundant TRUE)
        (PIDs "951" "952" "953")(title "HPOTP Primary Pump Seal Drain Pr"))
  (plot (class A5.15.2)(number 3)
        (PIDs "63")(title "Thrust Profile"))
)

;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.15.3
;;; Source: SAIC final report, section 5.15
;;; Summary: 951/952/953 and 1187 are both erratic or spiking. Concurrent anomaly in
;;; both sensors.

(defrule anomaly5.15.3
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "951"|"952"|"953" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1187" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  =>
  (assert (anomaly (class A5.15.3)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
       "HPOTP shows concurrent jitter in both primary pump seal drain pressure "
                        "and temperature. Possible seal anomaly."))))))

(deffacts init5.15.3
  (plot_ditto A5.15.3 A5.15.1))
```

```
;;;---------------------------------------------------------------------------
;;; Rule: anomaly5.15.4
;;; Source: Wilmer
;;; Summary: 951/952/953 and 1187 are both erratic or spiking. Non-concurrent anomaly
;;; in both sensors.

(defrule anomaly5.15.4
  (declare (salience -10))
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "951"|"952"|"953" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "1187" ?s2
       ?e2&:(is_concurrent ?start ?end ?s2 ?e2) $?)
  (not (anomaly (class A5.15.3|A5.15.4)(start ?s3)
              (end ?e3&:(is_concurrent ?start ?end ?s3 ?e3))))
  =>
  (assert (anomaly (class A5.15.4)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
       "HPOTP shows non-concurrent jitter in both primary pump seal drain pressure "
                     "and temperature. Possible seal anomaly."))))))

(deffacts init5.15.4
  (plot_ditto A5.15.4 A5.15.1))


;;;---------------------------------------------------------------------------
;;; Rule: anomaly5.18.1
;;; Source: SAIC final report, section 5.18
;;; Summary: 211/212 are erratic or spiking.

(defrule anomaly5.18.1
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed|unconfirmed F_ERRAT|F_SPIKE ?testid "211"|"212" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  =>
  (assert (anomaly (class A5.18.1)
                   (start ?start)
                   (end ?end)
                   (priority 16)
                (description =(str-cat
       "Intermediate seal purge pressure appears erratic or spiking. "
              "Possible nose seal leakage, helium supply problem or sensor anomaly. "
                     "Slight possibility of rubbing."))))))

(deffacts init5.18.1
  (plot (class A5.18.1)(number 1)
        (PIDs "211" "212")(title "HPOP Int Seal Purge Pr"))
  (plot (class A5.18.1)(number 2)
       (PIDs "937")(title "Engine Helium Interface Pr"))
  (plot (class A5.18.1)(number 3)
       (PIDs "92" "93" "951" "952" "953")
       (title "Adjacent Pressures (Sec Trb Sl Cav P & Pri Pump Sl Dr P)"))
)
```

```
;;;-------------------------------------------------------------------------------
;;; Rule: anomaly5.19.3
;;; Source: Inferred from Priority table in SAIC's POST_defs.h file.
;;; Summary: 233/234 are erratic or spiking.
;;; Evaluate for now.

(defrule anomaly5.19.3
  (current_phase find_anomalies)
  (current_test ?testid)
  (event_interval ?testid ?start ?end ~TRANSIENT)
  (confirmed F_SPIKE|F_ERRAT ?testid "233"|"234"|"518"|"519"|"521"|"522" ?s1
       ?e1&:(is_concurrent ?start ?end ?s1 ?e1) $?)
  =>
  (assert (anomaly (class A5.19.3)
                   (start ?start)
                   (end ?end)
                   (priority 16)
              (description =(str-cat "Spike or erratic behavior in HPOT discharge "
                             "temperature (confirmed by two sensors)."))))))


(deffacts init5.19.3
  (plot (class A5.19.3)(number 1)(full_sample TRUE)
       (anomaly_delta_start -5.0)(anomaly_delta_end +10.0)
       (PIDs "233" "234" "518" "519" "521" "522")
       (title "HPOT Turbine Discharge Temperature")
       (choose_redundant TRUE))
  (plot (class A5.19.3)(number 2)(full_sample TRUE)
       (anomaly_delta_start -5.0)(anomaly_delta_end +10.0)
       (PIDs "140" "141")(title "OPOV Actuator Position"))
  (plot (class A5.19.3)(number 3)
       (PIDs "233" "234" "518" "519" "521" "522")
       (title "HPOT Turbine Discharge Temperature")
       (choose_redundant TRUE)))
```

```
;;;---------------------------------------------------------------------------
;;; Rule: ADeltaPfamily
;;; Source: Inferred from examples from Wilmer.
;;; Summary: 327-328 statistics (for any time interval) are greater than 2.5 sigma.

(defrule ADeltaPfamily
  (current_phase find_anomalies)
  (current_test ?testid)
  (FAMILY_STAT ?type "HPOTP_BAL_CAV_P_A" ?A-mean ?A-stddev ?)
  (FAMILY_STAT ?type "HPOTP_BAL_CAV_P_B" ?B-mean ?B-stddev ?)
  (CURRENT_STAT ?testid ?type "HPOTP_BAL_CAV_P_A" ?currentA)
  (CURRENT_STAT ?testid ?type "HPOTP_BAL_CAV_P_B" ?currentB)
  (test (>= (abs (-
                   (/ (- ?currentA ?A-mean) ?A-stddev) ;;A deviation in stddevs
                   (/ (- ?currentB ?B-mean) ?B-stddev) ;;B deviation in stddevs
                )) 2.5))
  =>
  (bind ?delta (abs (-
                   (/ (- ?currentA ?A-mean) ?A-stddev) ;;A deviation in stddevs
                   (/ (- ?currentB ?B-mean) ?B-stddev)))) ;;B deviation in stddevs
  (if (> ?delta 0) then
      (bind ?direction HIGH)
  else
      (bind ?direction LOW))
  (assert (anomaly (class ADeltaPfamily)
                   (priority 16)
                   (description =(str-cat
       "HPOTP balance cavity pressure delta-P is out-of-family "
                                ?direction
                                   " ("
                                (format nil "%0.2f" ?delta )
                                " sigma) during "
                                  ?type
                             " conditions, compared to historical statistics. "))))))

(deffacts init-ADeltaPfamily
  (plot (class ADeltaPfamily)(number 1)(PIDs "327" "328")
       (title "HPOTP Balance Cavity Pressures"))
  (plot (class ADeltaPfamily)(number 2)(PIDs "63")(title "Thrust Profile"))
  (plot (class ADeltaPfamily)(number 3)(PIDs "858" "859" "860")
       (title "ENG LOX Inlet Pressure")))
```

FILE: HPOTP_greenrun.clp
GreeRun Specification Check Rules

```
;;; HPOTP Diagnostic System
;;; Greenrun Requirements Validation
;;;
;;; Created 12/20/93 T.W. Bickmore


;;; DEFFACTS INITGREEN_LIMITS
;;; Specifies most greenrun limits found in Tables II & III of GreenRun Specification
;;; RL00461, Revision H, 1/6/88.
(deffacts initGREEN_limits
  (GREEN_limit "HPOTP_PRI_PMP_SL_DR_P"     START  nil    18.0)

  (GREEN_limit "HPOT_DS_T"                 104    "104_MIN_NPSP"       nil    1540.0)
  (GREEN_limit "HPOT_DS_T"                 104    "104_MAX_NPSP"       nil    1470.0)
  (GREEN_limit "PBP_DS_P"                  104    "104_MAX_NPSP"       7040.0 nil)
  (GREEN_limit "HPOTP_PRI_PMP_SL_DR_P"     104    "104_MIN_NPSP"       nil    16.0)
  (GREEN_limit "HPOTP_PRI_PMP_SL_DR_P"     104    "104_MAX_NPSP"       nil    16.0)
  (GREEN_limit "HPOTP_PRI_TRB_SL_DR_P"     104    PEAK                 nil    60.0)
  (GREEN_limit "HPOTP_PRI_TRB_SL_DR_P"     104    "104_MIN_NPSP"       nil    30.0)
  (GREEN_limit "HPOTP_PRI_TRB_SL_DR_P"     104    "104_MAX_NPSP"       nil    30.0)
  (GREEN_limit "HPOTP_SEC_TRB_SL_CAV_P"    104    PEAK                 nil    31.5)
  (GREEN_limit "HPOTP_SEC_TRB_SL_CAV_P"    104    "104_MIN_NPSP"       nil    22.5)
  (GREEN_limit "HPOTP_SEC_TRB_SL_CAV_P"    104    "104_MAX_NPSP"       nil    22.5)

  (GREEN_limit "HPOT_DS_T"                 109    "109_MIN_NPSP"       nil    1600.0)
  (GREEN_limit "HPOT_DS_T"                 109    "109_MAX_NPSP"       nil    1510.0)
  (GREEN_limit "PBP_DS_P"                  109    "109_MAX_NPSP"       7515.0 nil)
  (GREEN_limit "HPOTP_PRI_PMP_SL_DR_P"     109    "109_MIN_NPSP"       nil    16.0)
  (GREEN_limit "HPOTP_PRI_PMP_SL_DR_P"     109    "109_MAX_NPSP"       nil    16.0)
  (GREEN_limit "HPOTP_PRI_TRB_SL_DR_P"     109    PEAK                 nil    67.0)
  (GREEN_limit "HPOTP_PRI_TRB_SL_DR_P"     109    "109_MIN_NPSP"       nil    34.0)
  (GREEN_limit "HPOTP_PRI_TRB_SL_DR_P"     109    "109_MAX_NPSP"       nil    34.0)
  (GREEN_limit "HPOTP_SEC_TRB_SL_CAV_P"    109    PEAK                 nil    35.0)
  (GREEN_limit "HPOTP_SEC_TRB_SL_CAV_P"    109    "109_MIN_NPSP"       nil    25.0)
  (GREEN_limit "HPOTP_SEC_TRB_SL_CAV_P"    109    "109_MAX_NPSP"       nil    25.0))


;;;-------------------------------------------------------------------------------
;;; Rule: GREEN_check_duration
;;; Source: GreenRun Specs, RL00461, 6 Jan 1988
;;; Summary:
;;; The following rules determine the total time spent at 104% and 109%, and
;;; check them against the greenrun specs.

(deffacts GREEN_init_time_at_104
  (time_at 104 0.0)
  (time_at 109 0.0))
(defrule GREEN_copy_PL_times
  (declare (salience 10))
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL&:(>= ?PL 104))
  =>
  (if (>= ?PL 109) then
      (assert (one_time_at 109 =(- ?end ?start)))
  else
      (assert (one_time_at 104 =(- ?end ?start)))))
```

```
(defrule GREEN_sum_times
  (declare (salience 10))
  ?f1 <- (time_at  ?PL ?sofar)
  ?f2 <- (one_time_at ?PL ?time)
  =>
  (retract ?f1 ?f2)
  (assert (one_time_at ?PL =(+ ?sofar ?time))))

(defrule GREEN_check_duration
  (current_phase find_anomalies)
  (current_test ?testid)
  (shutdown_time ?testid ?shut)
  (or (test (< ?shut 250))
      (time_at 109 ?time&:(< ?time 50))
      (and (time_at 109 ?time109)
           (time_at 104 ?time104&:(< (+ ?time109 ?time104) 150.0))))
  =>
  (assert (anomaly
              (start 0.0)
              (end ?shut)
              (class G3512a)
              (priority 16)
              (append_time FALSE)
              (repeatable FALSE)
              (description
       "Failed HPOTP GreenRun test duration criteria 3.5.1.2(a)."))))

(deffacts init3512a
  (plot (class G3512a)(number 1)
       (PIDs "63")(title "Thrust Profile")))

;;;---------------------------------------------------------------------
;;; Rule: GREEN_check_LPOTP_inlet
;;; Source: GreenRun Specs, RL00461, 6 Jan 1988
;;; Summary:
;;; The following rules check the minimum required duration at MIN and MAX
;;; LOX pressurization.

;;; Note: FindConstantThrust clips 2 seconds from each power level,
;;;       so time requirements are shortened accordingly.
(defrule GREEN_LPOTP_inlet_OKa
  (declare (salience 10))
  (current_phase find_anomalies)
  (current_test ?testid)
  ;;(b) Minimum NPSP of 20+5/-0 for 5 seconds at 104% or higher...
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end
       "104_MIN_NPSP"|"109_MIN_NPSP")
  (test (>= (- ?end ?start) 3.0))
  =>
  (assert (GREEN_OK ?testid G3512b)))
```

```
(defrule GREEN_LPOTP_inlet_OKb
  (declare (salience 10))
  (current_phase find_anomalies)
  (current_test ?testid)
  ;;(c) Maximum inlet pressure of 150+10/-0 for 10 seconds
  ;; at 104% or higher (not for stand A3)...
  (or (test (eq (sub-string 2 2 ?testid ) "3"))
      (and
        (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end
              "104_MAX_NPSP"|"109_MAX_NPSP")
        (test (>= (- ?end ?start) 8.0))))
  =>
  (assert (GREEN_OK ?testid G3512c)))

(defrule GREEN_check_LPOTP_inletb
  (current_phase find_anomalies)
  (current_test ?testid)
  (not (GREEN_OK ?testid G3512b))
  (shutdown_time ?testid ?shut)
  =>
  (assert (anomaly
              (class G3512b)
              (priority 16)
              (start 0.0)
              (end ?shut)
              (append_time FALSE)
              (repeatable FALSE)
              (description =(str-cat
              "Failed HPOTP GreenRun LPOTP inlet criteria 3.5.1.2(b). "
       "(Minimum NPSP of 20+5/-0 for 5 seconds at 104% or higher.)")))))

(defrule GREEN_check_LPOTP_inletc
  (current_phase find_anomalies)
  (current_test ?testid)
  (not (GREEN_OK ?testid G3512c))
  (shutdown_time ?testid ?shut)
  =>
  (assert (anomaly
              (class G3512c)
              (priority 16)
              (start 0.0)
              (end ?shut)
              (append_time FALSE)
              (repeatable FALSE)
              (description =(str-cat
              "Failed HPOTP GreenRun LPOTP inlet criteria 3.5.1.2(c). "
       "(Maximum NPSP of 150+10/-0 for 10 seconds at 104% or higher.)")))))

(deffacts init3512bc
  (plot (class G3512b)(number 1)(title "Engine Ox Inlet Pressure")
        (PIDs "858" "859" "860")(choose_redundant TRUE))
  (plot (class G3512b)(number 2)
       (PIDs "63")(title "Thrust Profile"))
  (plot_ditto G3512c G3512b))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: GREEN_check_65_time
;;; Source: GreenRun Specs, RL00461, 6 Jan 1988
;;; Summary:
;;; Checks the minimum bucket durations.

;;; Note: FindConstantThrust clips 2 seconds of each power level, so must
;;;       shorten requirement from 10 seconds to 8 to accomodate.
(defrule GREEN_check_65_time
  (current_phase find_anomalies)
  (current_test ?testid)
  (shutdown_time ?testid ?shut)
  (or (not (F_THLEDE ?testid ?start ?end&:(>= (- ?end ?start) 8.0) 65))
      (not (F_THLEDE ?testid ?start ?end&:(>= (- ?end ?start) 8.0) 64))
      (not (F_THLEDE ?testid ?start ?end&:(>= (- ?end ?start) 8.0) 63)))
  =>
  (assert (anomaly
              (class G3512d)
              (priority 16)
              (start 0.0)
              (end ?shut)
              (append_time FALSE)
              (repeatable FALSE)
              (description
       "Failed HPOTP GreenRun 65/64/63% throttle criteria 3.5.1.2(d)"))))

(deffacts init3512d
  (plot_ditto G3512d G3512a))

;;;-----------------------------------------------------------------------------
;;; Rule: GREEN_check_limits
;;; Source: GreenRun Specs, RL00461, 6 Jan 1988
;;; Summary: 3513II & III
;;; Checks the various limits in the GREEN_limit table at the top of this file.

(defrule GREEN_check_start_limits
  (current_phase find_anomalies)
  (current_test ?testid)
  (GREEN_limit ?param START ?min ?max)
  (sensor_status ?testid ?pid ?param preferred)
  (sensor (parameter ?param)(desc ?text))
  (or (and
       (test (neq ?min nil))
       (F_RLVIOL ?testid ?pid nil ?vstart ?vend "either_pid" "lower"
              ?limit&:(approx-eq ?limit ?min 0.01)))
      (and
       (test (neq ?max nil))
       (F_RLVIOL ?testid ?pid nil ?vstart ?vend "either_pid" "upper"
              ?limit&:(approx-eq ?limit ?max 0.01))))
  (test (is_concurrent ?vstart ?vend -1.0 0.0))
  =>
  (bind ?class (gensym))
  (assert (anomaly
              (class ?class)
              (priority 16)
              (start -0.0)(end 0.0)
              (append_time FALSE)
              (repeatable FALSE)
              (description =(str-cat
              "Failed HPOTP GreenRun limits at START for " ?text "."))
          (plot (class ?class)(number 1)(PIDs ?pid)(title ?text)
              (anomaly_delta_start -5.0)(anomaly_delta_end +5.0))))
```

```
(defrule GREEN_check_peak_limits
  (current_phase find_anomalies)
  (current_test ?testid)
  (GREEN_limit ?param ?limitPL&~START PEAK ?min ?max)
  (sensor_status ?testid ?pid ?param preferred)
  (sensor (parameter ?param)(desc ?text))
  (F_THLEDE ?testid ?start ?end ?PL)
  (test (or (and (= ?limitPL 104)(>= ?PL 104)(< ?PL 109))
            (and (= ?limitPL 109)(>= ?PL 109))))
  (or (and
       (test (neq ?min nil))
        (F_RLVIOL ?testid ?pid nil ?vstart ?vend "either_pid" "lower"
              ?limit&:(approx-eq ?limit ?min 0.01)))
      (and
       (test (neq ?max nil))
        (F_RLVIOL ?testid ?pid nil ?vstart ?vend "either_pid" "upper"
              ?limit&:(approx-eq ?limit ?max 0.01))))
  (test (is_concurrent ?start ?end ?vstart ?vend))
  =>
  (bind ?class (gensym))
  (assert (anomaly
              (class ?class)
              (priority 16)
              (start =(max ?start ?vstart))
              (end   =(min ?end ?vend))
              (description
                =(str-cat "Failed HPOTP GreenRun " ?PL "% peak limits for "
                     ?text ".")))
        (plot (class ?class)(number 1)(PIDs ?pid)(title ?text)
              (anomaly_delta_start -5.0)(anomaly_delta_end +5.0))))

(defrule GREEN_check_limits
  (current_phase find_anomalies)
  (current_test ?testid)
  (GREEN_limit ?param ?limitPL&~START ?inlet&~PEAK ?min ?max)
  (sensor_status ?testid ?pid ?param preferred)
  (sensor (parameter ?param)(desc ?text))
  (confirmed|unconfirmed F_INRANGE ?testid ? ?start ?end ?inlet)
  (or (and
       (test (neq ?min nil))
        (F_RLVIOL ?testid ?pid nil ?vstart ?vend "either_pid" "lower"
              ?limit&:(approx-eq ?limit ?min 0.01)))
      (and
       (test (neq ?max nil))
        (F_RLVIOL ?testid ?pid nil ?vstart ?vend "either_pid" "upper"
              ?limit&:(approx-eq ?limit ?max 0.01))))
  (test (is_concurrent ?start ?end ?vstart ?vend))
  =>
  (bind ?class (gensym))
  (assert (anomaly
              (class ?class)
              (priority 16)
              (start =(max ?start ?vstart))
              (end   =(min ?end ?vend))
              (description
                =(str-cat "Failed HPOTP GreenRun " ?inlet " limits for " ?text ".")))
        (plot (class ?class)(number 1)(PIDs ?pid)(title ?text)
              (anomaly_delta_start -5.0)(anomaly_delta_end +5.0))))
```

```
;;;-----------------------------------------------------------------------------
;;; Rule: GREEN_check_IMSLStart
;;; Source: Wilmer
;;; Summary: Normalized form.
;;; Checks the intermediate seal purge pressure requirement at start.

(defrule GREEN_check_IMSLStart
  (current_phase find_anomalies)
  (current_test ?testid)
  (STATS ?testid "211"|"212" ?s1&:(eqtime ?s1 -1.0) ?e1&:(eqtime ?e1 0.0) ?imsl $?)
  (STATS ?testid "937"     ?s2&:(eqtime ?s2 -1.0) ?e2&:(eqtime ?e2 0.0) ?he   $?)
  (test (< (/ (* ?imsl 730.0)(+ ?he 14.7)) 180.0))
  =>
  (assert (anomaly (class AIMSLStart)
                   (start -5.0)
                   (end 5.0)
              (priority 16)
              (append_time FALSE)
              (repeatable FALSE)
                   (description
       "Failed HPOTP GreenRun intermediate seal purge pressure START criteria."))))

(deffacts initAIMSLStart
  (plot (class AIMSLStart)(number 1)(PIDs 211 212)
       (title "HPOTP Intermediate Seal Purge Press"))
  (plot (class AIMSLStart)(number 2)(PIDs 937)(title "Eng Helium Interface Press")))

;;;-----------------------------------------------------------------------------
;;; Rule: GREEN_check_DeltaT
;;; Source: GreenRun Specs, RL00461, 6 Jan 1988, Wilmer
;;; Summary: 3513II & III
;;; Checks the minimum turbine delta-T requirements. If limit is exceeded and turbine
;;; temps are cold, then that is offered as an explanation.

(defrule GREEN_check_DeltaT1
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (F_RLVIOL ?testid ?tpid&"233"|"234"|"518"|"519"|"521"|"522" "1190" ?vstart ?vend
              "difference" "lower" ?limit)
  (test (is_concurrent ?start ?end ?vstart ?vend))
  (test (or (and (>= ?PL 104)(< ?PL 109)(approx-eq ?limit 280.0 0.1))
            (and (>= ?PL 109)(approx-eq ?limit 370.0 0.1))))
  (not (anomaly (class G3513DT)(start ?s2)
       (end ?e2&:(is_concurrent ?start ?end ?s2 ?e2))))
  (F_RLVIOL ?testid "233"|"234"|"518"|"519"|"521"|"522" nil ?rs ?re
              "either_pid" "lower" ?rl&:(approx-eq ?rl 1300.0 1.0))
  (test (is_concurrent ?vstart ?vend ?rs ?re))
  =>
  (assert (anomaly
              (class G3513DT)
              (priority 16)
              (start =(max ?start ?vstart))
              (end =(min ?end ?vend))
              (description =(str-cat "Failed HPOTP GreenRun " ?PL
                             "% limits for turbine Delta-T."
              " Probable cause is cold turbine temperature (below 1300.0)." )))))
```

```
(defrule GREEN_check_DeltaT2
  (current_phase find_anomalies)
  (current_test ?testid)
  (F_THLEDE ?testid ?start ?end ?PL)
  (F_RLVIOL ?testid ?tpid&"233"|"234"|"518"|"519"|"521"|"522" "1190" ?vstart ?vend
             "difference" "lower" ?limit)
  (test (is_concurrent ?start ?end ?vstart ?vend))
  (test (or (and (>= ?PL 104)(< ?PL 109)(approx-eq ?limit 280.0 0.1))
            (and (>= ?PL 109)(approx-eq ?limit 370.0 0.1))))
  (not (anomaly (class G3513DT)(start ?s2)
       (end ?e2&:(is_concurrent ?start ?end ?s2 ?e2))))
  (not (F_RLVIOL ?testid "233"|"234"|"518"|"519"|"521"|"522" nil ?rs ?re
             "either_pid" "lower"
       ?rl&:(and (approx-eq ?rl 1300.0 1.0)(is_concurrent ?vstart ?vend ?rs ?re))))
  =>
  (assert (anomaly
             (class G3513DT)
             (priority 16)
             (start =(max ?start ?vstart))
             (end =(min ?end ?vend))
             (description =(str-cat "Failed HPOTP GreenRun " ?PL
       "% limits for turbine Delta-T. Turbine temperature is not cold."))))))

(deffacts initG3513DT
  (plot (class G3513DT)(number 1)(PIDs 233 234)(title "HPOT Discharge Temps")
       (anomaly_delta_start -5.0)(anomaly_delta_end +5.0))
  (plot (class G3513DT)(number 2)(PIDs 1190)
       (title "HPOTP Primary Turbine Seal Drain Temp")
       (anomaly_delta_start -5.0)(anomaly_delta_end +5.0))
  (plot (class G3513DT)(number 3)(PIDs 63)(title "Thrust Profile")))
```

```
;;;-------------------------------------------------------------------------
;;; Rule: GREEN_check_DeltaSpeed
;;; Source: GreenRun Specs, RL00461, 6 Jan 1988
;;; Summary: 3513II & III
;;; Checks delta-speed requirements.

(defrule GREEN_check_DeltaSpeed
  (current_phase find_anomalies)
  (current_test ?testid)
  (linear_behavior ?testid ?start&:(> ?start 10.0) ?end ?PL ?inlet1 ?inlet2)
  (STATS ?testid "2" ?time1a ?time1b ?mean1 $?)
  (test (event_in_interval ?start ?time1a ?time1b))
  (STATS ?testid "2" ?time2a ?time2b ?mean2 $?)
  (test (event_in_interval ?end ?time2a ?time2b))
  (test (or (and (>= ?PL 104)(< ?PL 109)
              (or (and (approx-eq ?inlet2 25.0 5.0)
                       (> (- ?mean2 ?mean1) 800.0))
                  (and (approx-eq ?inlet2 150.0 10.0)
                       (< (- ?mean2 ?mean1) -400))))
         (and (>= ?PL 109)
              (or (and (approx-eq ?inlet2 25.0 5.0)
                       (> (- ?mean2 ?mean1) 1440.0))
                  (and (approx-eq ?inlet2 150.0 10.0)
                       (< (- ?mean2 ?mean1) -720) )))))
  =>
  (assert (anomaly
             (class G3513DS)
             (priority 16)
             (start ?start)
             (end  ?end)
             (description =(str-cat "Failed HPOTP GreenRun " ?PL
                    "% limits for speed change.")))))

(deffacts initG3513DS
  (plot (class G3513DS)(number 1)(PIDs 2)(title "HPOTP Speed")
       (anomaly_delta_start -5.0)(anomaly_delta_end +5.0))
  (plot (class G3513DS)(number 2)(PIDs 858 859 860)(choose_redundant TRUE)
       (title "Eng LOX Inlet Pressure"))
  (plot (class G3513DS)(number 3)(PIDs 63)(title "Thrust Profile")))
```

FILE: HPOTP_plot.clp
Supporting Plot Generation

```
;;; HPOTP Diagnostic Module
;;; Plot Generation for PLOTINFO.
;;;
;;; Creation 8/10/93  T.W. Bickmore

;;; ------------------------- UTILTY RULES/FUNCTIONS -----------------
;;;

;;; DEFFUNCTION SUBSCRIPT
;;; Takes a symbol and a number and produces a new symbol whose
;;; name is the symbol concatenated to the number.
(deffunction subscript (?name ?subscript)
  (nth 1 (str-explode (str-cat ?name ?subscript))))

;;; DEFRULE PLOT_EMIT_INFO
;;; Adds information about a single PLOTINFO field into the list of fields for output.
(defrule PLOT_emit_info
  (declare (salience 100))
  (current_phase prepare_output)
  ?f1 <- (emit_plot_info ?postulate ?slot ?value)
  ?f2 <- (plot_info_slots ?postulate $?slots)
  ?f3 <- (plot_info_values ?postulate $?values)
  =>
  (retract ?f1 ?f2 ?f3)
  (assert (plot_info_slots ?postulate $?slots ?slot)
          (plot_info_values ?postulate $?values ?value)))

;;; ---------------------- INITIALIZATION ----------------------------

;;; DEFRULE PLOT_COPY_DATA
;;; Copies information from one anomaly class to another, as directed
;;; by 'plot_ditto' commands.
(defrule PLOT_copy_data
  (current_phase initialize)
  (plot_ditto ?new ?old)
  (plot (class ?old)
        (use_comparison ?uc)
        (cross_comparison ?cc)
        (full_sample ?fs)
        (end ?end)
        (number ?n)
        (anomaly_delta_start ?as)
        (anomaly_delta_end ?ae)
       (choose_redundant ?ch)
        (shutdown_delta_end ?se)
        (title ?title)
        (PIDs $?pids))
  =>
  (assert (plot (class ?new)
        (use_comparison ?uc)
        (cross_comparison ?cc)
        (full_sample ?fs)
        (number ?n)
        (end ?end)
       (choose_redundant ?ch)
        (anomaly_delta_start ?as)
        (anomaly_delta_end ?ae)
        (shutdown_delta_end ?se)
        (title ?title)
        (PIDs $?pids))))
```

```
;;; DEFRULE PLOT_CHOOSE_REDUNDANT_PID
;;; This selects the preferred PID for any plots which have their 'choose_redundant'
;;; flag set.
(defrule PLOT_choose_redundant_pid
  (declare (salience 10000))
  (current_phase prepare_output)
  (current_test ?testid)
  ?f <- (plot (choose_redundant TRUE)
              (PIDs $? ?pid $?))
  (sensor_status ?testid ?pid preferred)
  =>
  (modify ?f
     (choose_redundant FALSE)
     (PIDs ?pid)))


;;; ---------------------- ASSIGN POSTULATE --------------------------

;;; DEFFACTS PLOT_INIT_POSTNUM
;;; Initializes the number of postulates to zero.
(deffacts PLOT_init_postnum
  (num_postulates 0))


;;; DEFRULE PLOT_ASSIGN_POSTNUMS
;;; Assigns a postulate number and name to each anomaly.
(defrule PLOT_assign_postnums
 (declare (salience -100))
 (current_phase prepare_output)
 (current_test ?testid)
 ?f1 <- (num_postulates ?n)
 ?f2 <- (anomaly (postnum nil)(class ?class))
 =>
 (bind ?postnum (+ ?n 1))
 (bind ?postulate (str-cat "post_" ?testid "_HPOTP_" ?n))
 (retract ?f1)
 (modify ?f2
     (postnum ?n)
     (name ?postulate))
 (assert (num_postulates ?postnum)))
```

```
;;; DEFRULE PLOT_INIT_INFO
;;; Initializes PLOTINFO fields and various temporary facts to
;;; begin plot generation for a particular anomaly.
;;; Only done if at least one plot spec exists.
(defrule PLOT_init_info
 (current_phase prepare_output)
 (current_test ?testid)
 (anomaly (postnum ?post_num&~nil)(name ?postulate)(class ?class))
 (plot (class ?class))  ;;At least one must exist.
 (not (plot_info_slots ?postulate $?))
 =>
 (assert (plot_info_slots ?postulate NAME POST_NUMBER MODULE CUR_TESTID)
         (plot_info_values ?postulate ?postulate ?post_num HPOTP ?testid)
         (num_plots 0)
         (pid_index 0)
         (current_anomaly ?postulate)))

;;; DEFRULE PLOT_INIT_INFO_COMPARISON
;;; Outputs the comparison test ID, if available.
(defrule PLOT_init_info_comparison
 (current_phase prepare_output)
 (comparison_test ?testid)
 (current_anomaly ?postulate)
 (anomaly (name ?postulate)(class ?class))
 (or (plot (class ?class)(use_comparison TRUE))
     (plot (class ?class)(cross_comparison TRUE)))
 (plot (class ?class))  ;;At least one must exist.
 =>
 (assert (emit_plot_info ?postulate PREV_TESTID ?testid)))


;;; ---------------------- GENERATE PLOTS --------------------------

;;; DEFRULE PLOT_HANDLE_ONE_PLOT
;;; Asserts a control fact to start generation of the next plot for
;;; the current anomaly.
(defrule PLOT_handle_one_plot
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (num_plots ?sofar)
  (anomaly (name ?postulate)(class ?class))
  (plot (class ?class)
        (number ?n&=(+ ?sofar 1)))
  =>
  (assert (prepare_plot ?n)))

;;; DEFRULE PLOT_FINISH_PLOT
;;; Updates control facts when all information for the current plot
;;; has been specified.
(defrule PLOT_finish_plot
  (declare (salience -50))
  (current_phase prepare_output)
  ?f1 <- (prepare_plot ?n)
  ?f2 <- (num_plots ?sofar)
  =>
  (retract ?f1 ?f2)
  (assert (num_plots ?n)))
```

```
;;; DEFRULE PLOT_WRAPUP
;;; Updates control facts and output the number of plots when plot
;;; generation for the current anomaly has been completed.
(defrule PLOT_wrapup
  (declare (salience -80))
  (current_phase prepare_output)
  (current_test ?testid)
  ?f1 <- (current_anomaly ?postulate)
  (anomaly (name ?postulate))
  ?f2 <- (num_plots ?num_plots)
  ?f3 <- (pid_index ?)
  =>
  (retract ?f1 ?f2 ?f3)
  (assert (emit_plot_info ?postulate NUM_PLOTS ?num_plots)))

;;; -------------------- PLOT TIME RANGE --------------------------

;;; DEFRULE PLOT_EMIT_PLOT_TIMES_1
;;; Outputs the start and end time for the current plot, when specified as
;;; deltas from the anomaly start and end times.
(defrule PLOT_emit_plot_times_1 ;;for Delta from anomaly time
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class)(start ?anom_start))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (anomaly_delta_start ?astart_delta&~nil)
        (anomaly_delta_end ?aend_delta&~nil))
  =>
  (assert (emit_plot_info ?postulate =(subscript START_TIME ?n)
              =(+ ?anom_start ?astart_delta))
          (emit_plot_info ?postulate =(subscript END_TIME ?n)
              =(+ ?anom_start ?aend_delta))))

;;; DEFRULE PLOT_EMIT_PLOT_TIMES_2
;;; Outputs the start and end time for the current plot, from start to
;;; shutdown (or end) for a current vs. comparison test plot.
(defrule PLOT_emit_plot_times_2 ;;use comparison test
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (anomaly_delta_start nil)
        (anomaly_delta_end nil)
        (end ?end)
        (cross_comparison ?cross)
        (use_comparison   ?use&:(or ?cross ?use))
        (shutdown_delta_end ?shut_delta))
  (current_test ?current_tid)
  (TESTINFO ?current_tid $? ?current_shutdown)
  (comparison_test ?comparison_tid)
  (TESTINFO ?comparison_tid $? ?comparison_shutdown)
  =>
  (if (eq ?shut_delta nil) then (bind ?shut_delta 0.0))
  (if (eq ?end nil) then
      (bind ?end (+ (max ?current_shutdown ?comparison_shutdown) ?shut_delta)))
  (assert (emit_plot_info ?postulate =(subscript START_TIME ?n) 0.0)
          (emit_plot_info ?postulate =(subscript END_TIME ?n)  ?end)))
```

```
;;; DEFRULE PLOT_EMIT_PLOT_TIMES_3
;;; Outputs the start and end time for the current plot, from start to
;;; shutdown (or end) for a current test only plot.
(defrule PLOT_emit_plot_times_3 ;;no comparison test
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (anomaly_delta_start nil)
        (anomaly_delta_end nil)
        (end ?end)
        (use_comparison FALSE)
        (cross_comparison FALSE)
        (shutdown_delta_end ?shut_delta))
  (current_test ?current_tid)
  (TESTINFO ?current_tid $? ?shutdown)
  =>
  (if (eq ?shut_delta nil) then (bind ?shut_delta 0.0))
  (if (eq ?end nil) then
      (bind ?end (+ ?shutdown ?shut_delta)))
  (assert (emit_plot_info ?postulate =(subscript START_TIME ?n) 0.0)
          (emit_plot_info ?postulate =(subscript END_TIME ?n)  ?end)))

;;; ----------------- PLOT MISCELLANY ----------------------------

;;; DEFRULE PLOT_EMIT_MISCELLANY
;;; Outputs FULL_SAMPLEn, NUM_CURVESn, and XTITLEn fields for the current plot.
(defrule PLOT_emit_miscellany
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (cross_comparison ?usecomp)
        (full_sample ?fullsample)
        (PIDs $?pids))
  =>
  (if ?fullsample then (bind ?fullsample 1) else (bind ?fullsample 0))
  (if ?usecomp then (bind ?factor 2) else (bind ?factor 1))
  (assert (emit_plot_info ?postulate =(subscript FULL_SAMPLE ?n) ?fullsample)
          (emit_plot_info ?postulate =(subscript NUM_CURVES ?n)
              =(* (length $?pids) ?factor))
          (emit_plot_info ?postulate =(subscript XTITLE ?n) "Time (seconds)")))
```

```
;;; DEFRULE PLOT_EMIT_PIDS_1
;;; Outputs PIDn, WHICH_TESTn, and LEG_LABELn fields for the current plot,
;;; for the current test only (no comparison).
(defrule PLOT_emit_pids_1 ;;no comparison test
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (cross_comparison FALSE)
        (use_comparison ?usecomp)
        (PIDs $?pids))
  ?f <- (pid_index ?index)
  (not (output_plot_pids ?postulate ?n))
  =>
  (retract ?f)
  (bind ?i 1)
  (if ?usecomp then (bind ?which 1) else (bind ?which 0))
  (while (<= ?i (length $?pids))
     (assert (emit_plot_info ?postulate =(subscript PID (+ ?i ?index))
                      =(str-cat (nth ?i $?pids)))
             (emit_plot_info ?postulate =(subscript WHICH_TEST (+ ?i ?index)) ?which)
             (emit_plot_info ?postulate =(subscript LEG_LABEL (+ ?i ?index))
                      =(str-cat (nth ?i $?pids))))
     (bind ?i (+ ?i 1)))
  (assert (output_plot_pids ?postulate ?n)
          (pid_index =(+ ?index (length $?pids)))))

;;; DEFRULE PLOT_EMIT_PIDS_1
;;; Outputs PIDn, WHICH_TESTn, and LEG_LABELn fields for the current plot,
;;; for current and comparison tests.
(defrule PLOT_emit_pids_2 ;;comparison test
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (cross_comparison TRUE)
        (PIDs $?pids))
  ?f <- (pid_index ?index)
  (not (output_plot_pids ?postulate ?n))
  =>
  (retract ?f)
  (bind ?i 1)
  (while (<= ?i (length $?pids))
     (bind ?curr (- (* ?i 2) 1))
     (bind ?prev (* ?i 2))
     (assert ;;Current test...
             (emit_plot_info ?postulate =(subscript LEG_LABEL ?curr)
                      =(str-cat (nth ?i $?pids) " (C)"))
             (emit_plot_info ?postulate =(subscript PID ?curr)
                      =(str-cat (nth ?i $?pids)))
             (emit_plot_info ?postulate =(subscript WHICH_TEST ?curr) 0)
             ;;Comparison test...
             (emit_plot_info ?postulate =(subscript LEG_LABEL ?prev)
                      =(str-cat (nth ?i $?pids) " (P)"))
             (emit_plot_info ?postulate =(subscript PID ?prev)
                      =(str-cat (nth ?i $?pids)))
             (emit_plot_info ?postulate =(subscript WHICH_TEST ?prev) 1))
     (bind ?i (+ ?i 1)))
  (assert (output_plot_pids ?postulate ?n)
          (pid_index =(+ ?index (* (length $?pids) 2)))))
```

```
;;; ---------------------- PLOT TITLES -------------------------------

;;; DEFRULE PLOT_EMIT_TITLE
;;; Outputs the title for the current plot.
(defrule PLOT_emit_title
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class)(start ?start)
       (append_time ?append))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (title ?title))
  =>
  (assert (emit_plot_info ?postulate =(subscript TITLE ?n) ?title)))

;;; DEFRULE PLOT_EMIT_SUBTITLE_1
;;; Outputs the subtitle for the current plot when there is no comparison test.
(defrule PLOT_emit_subtitle_1 ;;no comparison test
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (current_test ?tid)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (cross_comparison FALSE)
        (use_comparison FALSE))
  =>
  (assert (emit_plot_info ?postulate =(subscript SUBTITLE ?n) ?tid)))

;;; DEFRULE PLOT_EMIT_SUBTITLE_2
;;; Outputs the subtitle for the current plot when there is a comparison test.
;;; (For cross-plots.)
(defrule PLOT_emit_subtitle_2 ;;comparison test
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (current_test ?curr_tid)
  (comparison_test ?prev_tid)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (cross_comparison TRUE))
  =>
  (assert (emit_plot_info ?postulate =(subscript SUBTITLE ?n)
                          =(str-cat ?curr_tid "(C) " ?prev_tid "(P)"))))

;;; DEFRULE PLOT_EMIT_SUBTITLE_3
;;; Outputs the subtitle for the current plot when there is a comparison test.
;;; (For comparison test only plots.)
(defrule PLOT_emit_subtitle_3 ;;comparison test
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (comparison_test ?prev_tid)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (use_comparison TRUE))
  =>
  (assert (emit_plot_info ?postulate =(subscript SUBTITLE ?n)
                          =(str-cat ?prev_tid "(P)"))))
```

```
;;; DEFRULE PLOT_EMIT_YTITLE
;;; Outputs the YTITLEn field for the current plot.
;; Assumes all PIDs on plot have same label...
(defrule PLOT_emit_ytitle
  (current_phase prepare_output)
  (current_anomaly ?postulate)
  (anomaly (name ?postulate)(class ?class))
  (prepare_plot ?n)
  (plot (class ?class)
        (number ?n)
        (PIDs ?pid $?))
  (PID_ytitle $? ?pid $? ?ylabel)
  =>
  (assert (emit_plot_info ?postulate =(subscript YTITLE ?n) ?ylabel)))

;;; DEFFACTS PLOT_INIT_YTITLES
;;; Declares the YTITLE values to use for each PID.
(deffacts PLOT_init_ytitles
  (PID_ytitle "327" "328" "990" "Gauge Pressure (psig)")
  (PID_ytitle "63"  "MCC PC (psia)")
  (PID_ytitle "24" "59" "90" "91" "92" "209" "210" "211" "212" "334" "341"
       "858" "859" "860" "951" "952" "953" "Pressure (psia)")
  (PID_ytitle "176" "PCNT")
  (PID_ytitle "233" "234" "1187" "1188" "1190" "Temperature (deg R)")
  (PID_ytitle "2" "Speed (RPM)")
)
```

FILE: HPOTP_IO.clp
Anomaly Output Rules

```
;;; HPOTP Diagnostic Module
;;; Anomaly Output Rules
;;;
;;; Creation 8/10/93  T.W. Bickmore

;;; ------------------------- Utility Functions & Rules ------------------

;;; DEFFUNCTION IO_TIME_TAG_DESC
;;; Prepends the string 'Seen at T=<time>.' to anomaly descriptions which
;;; have their append_time flags set, and which have a defined start time.
(deffunction IO_time_tag_desc (?desc ?start ?end ?append_time)
  (if (and ?append_time (neq ?start nil)(neq ?end nil)) then
      (if (approx-eq ?start ?end 1.0) then
       (str-cat "Seen at T=" (format nil "%0.2f" ?start) ". " ?desc)
      else
       (str-cat "Seen between T=" (format nil "%0.2f" ?start) " and "
                          (format nil "%0.2f" ?end) ". " ?desc))
  else
      ?desc))

;;; ------------------------- TEKBASE TABLE POPULATION -------------------
;;;
;;; These rules output a record to TekBase when a fact of the following form is
;;; asserted:
;;;    (IO_output <table <fieldname>* %VALUES% <value>*)


;;; DEFFACTS IO_INIT_TABLES
;;; Defines the types of all fields in TekBase tables which will be written to.
;;; (Primarily to ensure that an appropriate default value is written to all
;;; unspecified fields.)
(deffacts IO_init_tables
  (IO_table_fields POSTUL STRING NAME TEST_ID MODULE FMODE PROBLEM TYPE PID)
  (IO_table_fields POSTUL FIXED  POST_NUMBER PRIORITY)
  (IO_table_fields POSTUL FLOAT  START_TIME STOP_TIME)

  (IO_table_fields PLOTINFO STRING NAME PLOT_TYPE MODULE CUR_TESTID PREV_TESTID
            TITLE1 TITLE2 TITLE3 SUBTITLE1 SUBTITLE2 SUBTITLE3 XTITLE1 XTITLE2 XTITLE3
            YTITLE1 YTITLE2 YTITLE3
            PID1 PID2 PID3 PID4 PID5 PID6 PID7 PID8 PID9 PID10
           LEG_LABEL1 LEG_LABEL2 LEG_LABEL3 LEG_LABEL4 LEG_LABEL5
           LEG_LABEL6 LEG_LABEL7 LEG_LABEL8 LEG_LABEL9 LEG_LABEL10)
  (IO_table_fields PLOTINFO FIXED  POST_NUMBER NUM_PLOTS
              FULL_SAMPLE1 FULL_SAMPLE2 FULL_SAMPLE3
              NUM_CURVES1 NUM_CURVES2 NUM_CURVES3
            WHICH_TEST1 WHICH_TEST2 WHICH_TEST3 WHICH_TEST4 WHICH_TEST5
            WHICH_TEST6 WHICH_TEST7 WHICH_TEST8 WHICH_TEST9 WHICH_TEST10)
  (IO_table_fields PLOTINFO FLOAT
          START_TIME1 START_TIME2 START_TIME3 END_TIME1 END_TIME2 END_TIME3)
)

;;; DEFRULE IO_ADD_DEFAULTS
;;; Adds appropriate default values to all unspecified fields in a TekBase table.
(defrule IO_add_defaults
  ?f <- (IO_output ?table $?columns %VALUES% $?values)
  (IO_table_fields ?table ?type $? ?missing $?)
  (test (not (member ?missing $?columns)))
  =>
  (retract ?f)
  (if (eq ?type FIXED) then
      (bind ?value 0)
```

```
  else (if (eq ?type FLOAT) then
       (bind ?value 0.0)
  else
       (bind ?value "")))
  (assert (IO_output ?table $?columns ?missing %VALUES% $?values ?value)))

;;; DEFRULE IO_OUTPUT_TO_TABLE
;;; Once all defaults have been added, this actually writes a record out to TekBase.
(defrule IO_output_to_table
  (declare (salience -5))
  ?f <- (IO_output ?table $?columns %VALUES% $?values)
  =>
  (retract ?f)
  (if (check_DB_connection) then
    (DB_put ?table $?columns $?values))
  (if (is_true ?*DEBUG*) then
    (fprintout t ">>Writing to " ?table " :" crlf "> " $?columns crlf
               "> " $?values crlf)))

;;; ---------------------------- Anomalies to TekBase --------------------------
;;; Output to pid_info & postulates...

;;; DEFRULE IO_OUTPUT_POSTULATE
;;; Outputs a record to POSTUL table corresponding to an anomaly record.
(defrule IO_output_postulate
  (current_phase output_anomalies)
  (test (is_true ?*DB_output*))
  (current_test ?testid)
  (anomaly (class ?label)
           (start ?start)
           (end ?end)
           (priority ?priority)
           (postnum ?n)
           (name ?name)
          (type ?type)
          (PID ?pid)
          (append_time ?append_time)
           (description ?desc))
  =>
  (bind ?desc (IO_time_tag_desc ?desc ?start ?end ?append_time))
  (if (eq ?start nil) then (bind ?start 0.0))
  (if (eq ?end nil)   then (bind ?end 0.0))
  (if (eq ?PID nil)   then (bind ?PID ""))
  (assert (IO_output POSTUL
          NAME TEST_ID MODULE POST_NUMBER PRIORITY START_TIME
                       STOP_TIME PROBLEM TYPE PID
          %VALUES%
          ?name ?testid "HPOTP" ?n ?priority ?start ?end
                       ?desc ?type ?pid)))

;;; DEFRULE IO_ENSURE_OUTPUT_COMPID
;;; Establishes a default value for the comparison test of "".
(defrule IO_ensure_output_compid
  (declare (salience 100))
  (current_phase output_anomalies)
  (not (comparison_test ?))
  =>
  (assert (comparison_test "")))
```

```
;;; DEFRULE IO_OUTPUT_PLOTINFO
;;; Outputs a record to the PLOTINFO table corresponding to information in
;;; anomaly, plot, plot_info_slots, and plot_info_values facts.
(defrule IO_output_plotinfo
  (declare (salience 10))
  (current_phase output_anomalies)
  (test (is_true ?*DB_output*))
  (current_test ?testid)
  (comparison_test ?comptestid)
  (anomaly (class ?label)
           (start ?start)
           (end ?end)
           (priority ?priority)
           (postnum ?n)
           (name ?name))
  (plot (class ?label)(number ?maxn))
  (not (plot (class ?label)(number ?n2&:(> ?n2 ?maxn))))
  (plot_info_slots ?name $?slots)
  (plot_info_values ?name $?values)
  =>
  (assert (IO_output PLOTINFO
          NAME POST_NUMBER PLOT_TYPE MODULE CUR_TESTID PREV_TESTID NUM_PLOTS $?slots
          %VALUES%
           ?name ?n          TBD        "HPOTP" ?testid   ?comptestid ?maxn $?values)))

;;;---------------------------- OUTPUT ANOMALIES TO TERMINAL ----------------

;;; DEFRULE IO_INTERACTIVE_OK
;;; Determines if an anomaly should be displayed to the user in interactive mode,
;;; using the same priority threshold used by ehms.
(deffunction IO_interactive_OK (?type ?priority)
  (or (and (eq ?type INSTRUMENTATION)
           (> ?priority 0))
      (and (eq ?type OBSERVATION)
           (> ?priority 1))
      (and (eq ?type ANOMALIES)
           (> ?priority 5))))

;;; DEFRULE IO_PRINT_ANOMALIES_1
;;; Displays an anomaly to terminal in interactive mode, which does not have any plot
information
;;; associated with it.
(defrule IO_print_anomalies_1
  (current_phase output_anomalies)
  (current_test ?tid)
  (anomaly (class ?label)(priority ?priority)(start ?start)(end ?end)
              (append_time ?append_time)
           (description ?desc)(name ?postulate)(postnum ?number)(type ?type))
  (test (or ?*DEBUG*
            (and ?*interactive* (IO_interactive_OK ?type ?priority))))
  (not (plot_info_slots ?postulate $?))
  =>
  (bind ?desc (IO_time_tag_desc ?desc ?start ?end ?append_time))
  (fprintout t  crlf
     "---------------POSTULATE------------"        crlf)
  (if (is_true *?DEBUG*) then (fprintout t
     "NAME:              " ?postulate               crlf
     "POST_NUMBER:       " ?number   crlf
     "MODULE:            HPOTP"                     crlf
     "TEST_ID:           " ?tid                     crlf))
  (fprintout t
     "PRIORITY:          " ?priority        crlf
     "PROBLEM:           " ?desc                  crlf
     "RULE:              " ?label               crlf
```

```
   "TYPE:              " ?type                    crlf)
(if (and (neq ?start nil)(neq ?end nil)) then
  (fprintout t
   "START TIME:        " ?start crlf
   "END TIME:          " ?end crlf)))
```

```
;;; DEFRULE IO_PRINT_ANOMALIES_2
;;; Displays an anomaly to the terminal in interactive mode, which has plot
;;; information associated with it.
(defrule IO_print_anomalies_2
  (current_phase output_anomalies)
  (current_test ?tid)
  (anomaly (class ?label)(priority ?priority)(start ?start)(end ?end)
             (append_time ?append_time)
          (description ?desc)(name ?postulate)(postnum ?number)(type ?type))
  (test (or ?*DEBUG*
            (and ?*interactive* (IO_interactive_OK ?type ?priority))))
  (plot_info_slots ?postulate $?slots)
  (plot_info_values ?postulate $?values)
  =>
  (bind ?desc (IO_time_tag_desc ?desc ?start ?end ?append_time))
  (fprintout t  crlf
     "---------------POSTULATE------------"       crlf)
  (if (is_true *?DEBUG*) then (fprintout t
     "NAME:              " ?postulate               crlf
     "POST_NUMBER:       " ?number   crlf
     "MODULE:            HPOTP"                     crlf
     "TEST_ID:           " ?tid                     crlf))
  (fprintout t
     "PRIORITY:          " ?priority        crlf
     "PROBLEM:           " ?desc                    crlf
     "RULE:              " ?label                crlf
     "TYPE:              " ?type                 crlf)
  (if (and (neq ?start nil)(neq ?end nil)) then
    (fprintout t
     "START TIME:        " ?start crlf
     "END TIME:          " ?end crlf))
  (if (not (not ?*DEBUG*)) then
    (fprintout t
     "---------------PLOT_INFO------------"       crlf)
    (bind ?i 1)
    (while (<= ?i (length $?slots))
      (fprintout t (nth ?i $?slots) ": " (nth ?i $?values) crlf)
      (bind ?i (+ ?i 1)))))


;;; DEFRULE IO_PRINT_ALLS_WELL
;;; Displays a message to the user if no displayable anomalies were found in
;;; interactive mode.
(defrule IO_print_alls_well
  (current_phase output_anomalies)
  (test (or ?*interactive* ?*DEBUG*))
  (current_test ?testid)
  (not (anomaly (type ?type)
       (priority ?priority&:(IO_interactive_OK ?type ?priority))))
  =>
  (fprintout t crlf "------ No anomalies or observations found for test "
       ?testid "------" crlf))
```

```
;;;-------------------------- UPDATE HISTORICAL DB --------------------
;;; Updates the historical database if:
;;; 1. This test is not already in the database
;;;    AND
;;; 2. The system is running in batch mode,
;;;      OR
;;;    The system is running in interactive mode and the user approves.

;;; DEFRULE IO_UPDATE_CHECK
;;; Attempts to retrieve statistical information about this test from
;;; TekBase to see if it has already been updated.
(defrule IO_update_check
  (current_phase output_anomalies)
  (current_test ?testid)
  (test (is_true ?*DB_enabled*))
  =>
  (if (check_DB_connection) then
    (DB_exec "SET %Unique TRUE")
    (DB_get HISTORY
      (mv-append "TESTID")
      (str-cat "TESTID='" ?testid "'"))))

;;; DEFRULE IO_ASK_ABOUT_UPDATE
;;; In interactive mode, if the current test is not in the historical database,
;;; this asks the user if they would like to add it.
(defrule IO_ask_about_update
  (declare (salience -5))
  (current_phase output_anomalies)
  (current_test ?testid)
  (not (HISTORY ?testid))
  (test (and ?*DB_enabled* ?*interactive*))
  =>
  (fprintout t crlf "Would you like to update the historical database? (y/n):")
  (assert (IO_update_DB =(read))))

;;; DEFRULE IO_DO_UPDATE
;;; If the current test is not in the historical database, and either the system
;;; is running in batch mode, or the user OK'd it, the historical database is updated.
(defrule IO_do_update
  (declare (salience -10))
  (current_phase output_anomalies)
  (current_test ?testid)
  (not (HISTORY ?testid))
  (or (test (not ?*interactive*))
      (IO_update_DB y|Y|yes|YES))
  (CURRENT_STAT ?testid ?type ?param ?value)
  =>
  (if (check_DB_connection) then
    (DB_put HISTORY
       (mv-append TESTID TYPE PARAM VALUE OK_TO_USE)
       (mv-append ?testid ?type ?param ?value 1))))
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1995	Final Contractor Report

4. TITLE AND SUBTITLE

SSME HPOTP Post-Test Diagnostic System Enhancement Project

5. FUNDING NUMBERS

WU-584-03-11
C-NAS3-25883

6. AUTHOR(S)

Timothy W. Bickmore

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Aerojet Tech Systems
P.O. Box 13222
Sacramento, California 95813-6000

8. PERFORMING ORGANIZATION REPORT NUMBER

E-9372

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR-4643

11. SUPPLEMENTARY NOTES

Project Manager, June Zakrajsek, Space Propulsion Technology Division, NASA Lewis Research Center, organization code 5310, (216) 977-7470.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories 15, 16, and 20

This publication is available from the NASA Center for Aerospace Information, (301) 621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

An assessment of engine and component health is routinely made after each test or flight firing of a Space Shuttle Main Engine (SSME). Currently, this health assessment is done by teams of engineers who manually review sensor data, performance data, and engine and component operating histories. Based on review of information from these various sources, an evaluation is made as to the health of each component of the SSME and the preparedness of the engine for another test or flight. The objective of this project is to further development of a computer program which automates the analysis of test data from the SSME High-Pressure Oxidizer Turbopump (HPOTP) in order to detect and diagnose anomalies. This program fits into a larger system—the SSME Post-Test Diagnostic System (PTDS)—which will eventually be extended to assess the health and status of most SSME components on the basis of test data analysis. The HPOTP module is an expert system, which uses "rules-of-thumb" obtained from interviews with experts from NASA Marshall Space Flight Center (MSFC) to detect and diagnose anomalies. Analyses of the raw test data are first performed using pattern recognition techniques which result in features such as spikes, shifts, peaks, and drifts being detected and written to a database. The HPOTP module then looks for combination of these features which are indicative of known anomalies, using the rules gathered from the turbomachinery experts. Results of this analysis are then displayed via a graphical user interface which provides ranked lists of anomalies and observations by engine component, along with supporting data plots for each.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Space Shuttle Main Engine; Expert systems; Data reduction	169
	16. PRICE CODE: A08

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration

Lewis Research Center
21000 Brookpark Rd.
Cleveland, OH 44135-3191