UAH Research Report No. ME-91-101
Date of Issue: September 1991

ENGINE DATA INTERPRETATION SYSTEM (EDIS) (PHASE II)

Prepared by:

Thomas L. Cost and Martin O. Hofmann
College of Engineering
The University of Alabama in Huntsville
Huntsville, AL 35899

Prepared for:

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Final Report on:

Contract No. NAS8-36955, Delivery Order 97
Period of Performance: October 30, 1990 to July 29, 1991

Disclaimer Statement:

"The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official NASA position, policy, or decision, unless so designated by other documentation".

Distribution Statement:

Distribution is unlimited.

UAH Research Report No. ME-91-101
Date of Issue: September 1991

ENGINE DATA INTERPRETATION SYSTEM (EDIS) (PHASE II)

Prepared by:

Thomas L. Cost and Martin O. Hofmann
College of Engineering
The University of Alabama in Huntsville
Huntsville, AL 35899

Prepared for:

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Final Report on:

Contract No. NAS8-36955, Delivery Order 97
Period of Performance: October 30, 1990 to July 29, 1991

Disclaimer Statement:

"The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official NASA position, policy, or decision, unless so designated by other documentation".

Distribution Statement:

Distribution is unlimited.

SUMMARY

A prototype of an expert system has been developed which applies qualitative constraint-based reasoning to the task of post-test analysis of data resulting from a rocket engine firing. Data anomalies are detected and corresponding faults are diagnosed. Engine behavior is reconstructed using measured data and knowledge about engine behavior. Knowledge about common faults guides but does not restrict the search for the best explanation in terms of hypothesized faults. The system contains domain knowledge about the behavior of common rocket engine components and has been configured for use with the Space Shuttle Main Engine (SSME). A graphical user interface allows an expert user to intimately interact with the system during diagnosis. The system has been applied to data taken during actual SSME tests where data anomalies had been observed.

TABLE OF CONTENTS

1. INTRODUCTION

The research described in this report is a continuation of the work performed under a previous contract, NAS8-36955, D.O. 58. This, the second, phase of the development of a knowledge-based system which assists in post-test data analysis and fault diagnosis of the Space Shuttle Main Engine (SSME), builds upon the methodologies devised and explored during the earlier contract period. Implementation and software tools are completely different, however.

The Engine Data Interpretation System (EDIS) attempts to assist the data review personnel to quickly identify significant data anomalies and to generate explanations for the anomalies in terms of abnormal component behavior and underlying faults. The review process is time-consuming and repetitive and requires an undue amount of human resources, i.e. it occupies too much of the experts' time. Special care was taken during the design to fit EDIS into the current review process without disrupting standard review procedures. EDIS can also perform the data analysis and fault diagnosis tasks autonomously without intervention by the user. In this mode EDIS might be used to produce an independent second opinion.

EDIS is unusual compared to traditional diagnostic expert systems because it presents its questions and results to the user in a manner which requires very little explanation. The combination of a special diagnostic mechanism and a state-of-the-art user interface strategy make this behavior possible.

Diagnosis is performed using a qualitative constraint model of the SSME and its components. Qualitative parameter values describe data anomalies as positive or negative deviations from the norm. Constraints model normal component behavior independent of component use within the SSME. The reasoning mechanisms synthesizes a description of SSME behavior based on the measured data, the observed anomalies, and knowledge about normal and fault behaviors of components. The behavior description makes explicit the assumptions about component health and thus generates diagnostic hypotheses. The reasoning mechanisms deals naturally with single and multiples faults, with known and unknown fault modes, and may be tuned by an experienced user. It also accepts user guidance during processing. To some extent, the reasoning process emulates an expert who envisions how the SSME is performing using the familiar schematic diagram and assumptions about component faults.

The user interface was designed to make relevant data as well as choices made by the diagnostic process immediately visible to the user. Whenever possible the user may change the suggestions brought forth by the system and thus redirect diagnostic search. Graphic images, menus, and mouse-invoked actions were used throughout the design.

EDIS does not and cannot claim to automate the review process completely. A much larger development effort would be necessary to accomplish this. EDIS, however, should be seen

as a prototype of a data review support system in that it explores methods of representing relevant expert knowledge which may readily be adapted to changes in the engine, methods of reasoning in an expert-compatible style, and methods of user-centered mixed-initiative user-system communication. The EDIS project is therefore never finished.

In the following sections we will first lay out the knowledge representation and reasoning methodologies incorporated into EDIS and then describe how to use and maintain EDIS. We will close with a note on the state of development of EDIS and a demonstration of the performance of EDIS using simple examples.

2. DIAGNOSTIC METHODOLOGY

This section consists of a journal paper which was submitted for publication after receiving clearance from NASA.

MODEL-BASED DIAGNOSIS OF THE SPACE SHUTTLE MAIN ENGINE

Martin O. Hofmann
Department of Electrical and Computer Engineering

Thomas L. Cost
Department of Mechanical Engineering

The University of Alabama in Huntsville
Huntsville, AL 35899

Michael Whitley
NASA-MSFC
Huntsville, AL 35812

ABSTRACT

The process of reviewing test data for anomalies after a firing of the Space Shuttle Main Engine (SSME) is a complex, time-consuming task. A project is under way to provide the team of SSME experts with a knowledge-based system to assist in the review and diagnosis task. A model-based approach was chosen because it can be adapted to changes in engine design, is easier to maintain, and can be explained more easily. A complex thermodynamic fluid system like the SSME introduces problems during modeling, analysis, and diagnosis which have as yet been insufficiently studied. We developed a qualitative constraint-based diagnostic system inspired by existing qualitative modeling and constraint-based reasoning methods which addresses these difficulties explicitly. Our approach is unique in that it combines various diagnostic paradigms seamlessly, such as the model-based and heuristic association-based paradigms, in order to better approximate the reasoning process of the domain experts. The end-user interface allows expert users to actively participate in the reasoning process, both by adding their own expertise and by guiding the diagnostic search performed by the system.

1. Introduction

Three main engines provide a significant portion of thrust to the Space Shuttle during liftoff. Dependable performance of the Space Shuttle Main Engines (SSME) is critical to the Shuttle missions. The SSME is a highly complicated device which is stressed to the

limit during normal operation. Therefore, each SSME is tested exhaustively for flight readiness, and new and improved designs are developed and evaluated frequently.

A comprehensive data review is performed at NASA Marshall Space Flight Center (MSFC) after each firing of a SSME, both after Shuttle launches and after ground tests. The review serves to reveal anomalies in the recorded data and, if anomalies are detected, to diagnose the underlying engine problem or fault. Shuttle launches occur only infrequently, but a SSME is testfired as often as once every three days on one of the five test stands. During the test hundreds of measurements are taken up to fifty times per second. The resulting bulk of data has to be analyzed at Marshall Space Flight Center within a turnaround time of about one day. Analysis is currently performed by a team of experts, some of whom are specialists assigned by the SSME contractor companies; others are NASA engineers. These experts are also charged with modifying and upgrading the SSME and its components as well as with improving and maintaining performance models and data reduction programs. Routine data reviews distract and take time away from these more important activities. SSME data review and fault diagnosis are good candidates for supporting and augmenting expertise-based human activities with a knowledge-based expert system. The tasks are repetitive, require expertise in short demand, and do not lend themselves to an algorithmic solution. We are developing a knowledge-based system called EDIS (Engine Data Interpretation System) which assists the review team in the analysis of engine performance data and in the diagnosis of engine faults.

EDIS incorporates a combination of diagnostic paradigms because neither heuristic rules, quantitative models, nor qualitative models alone adequately represent the scope of knowledge that human experts apply to the task of SSME data analysis and fault diagnosis. Experts are able to intuitively intermix and coordinate reasoning based on these different types of knowledge such that each contributes to the solution. The architecture of EDIS has been formulated to enable cooperation of several heterogeneous knowledge sources in order to emulate the diversity of human expert reasoning. Coordination and cooperation of different reasoning paradigms will be described elsewhere. In this paper we present the qualitative model-based component of EDIS and the diagnostic reasoning methodology which operates on the qualitative model.

The approach to device diagnosis presented in this paper is rooted in qualitative physics, model-based representation of devices, and constraint satisfaction methods. Device behavior is represented qualitatively using "qualitative constraints." Constraints model the normal (non-faulty) behavior of each component. They are derived by linearizing and simplifying physical (thermodynamic) laws. The reasoning mechanism dynamically composes device behavior from component behavior, reconciling current case-specific measurements with qualitative constraints. Anomalous device behavior gives rise to fault hypotheses. Fault diagnosis using constraint-based models is not limited to selecting from a set of predetermined faults and does not require assumptions about the number of faults present. Knowledge about fault probabilities and component fault modes is only used to focus diagnostic search.

A large number of sensors provide data on SSME performance. However large the set of parameters measured may be, it is nevertheless limited to readings from the installed sensors and therefore fixed and incomplete. No additional measurements can be made; probing is not possible. Many approaches to expert diagnosis depend on the ability to acquire additional data. The methodology presented in this paper is designed to make optimal use of available data, measured at locations distributed fairly evenly throughout the device, without the need for complete information. Because of incomplete access to critical data, cases exist where no definitive diagnosis can be generated, and instead a list of possible faults constitutes the final result of the diagnostic reasoning process.

In the following sections we will first introduce the application domain and point out its unusual characteristics. Next we will outline our approach to qualitative constraint-based modeling and diagnosis followed by our rationale for selecting this approach. We will then describe the methods for implementing the approach, discuss the results, and compare our approach to work of other researchers.

2. SSME Data Review Process

The EDIS system assists in the analysis of data from tests of the Space Shuttle Main Engine (SSME) (Cost & Hofmann, 1990). After each test of an SSME data from several hundred sensors are reviewed at NASA MSFC in order to verify SSME performance and to diagnose anomalies. A large number of graphs are printed which display parameter values versus time. The experts peruse these charts several times: first, to detect anomalies in the level or shape of the curves, then, if anomalies have been found, to correlate characteristics of other relevant measurements. For example, experts check temperature and flow rates at a pump inlet if perturbations in the pressure readings were detected at this site; they also follow anomalous readings along the fluid or gas path through the engine.

Separate sets of charts are produced for different stages of the SSME firing, i.e. the startup, main stage, and shutdown stage. During main stage the SSME settles into steady state during time intervals of constant commanded power level. The first version of EDIS is limited to analysis of steady-state data during mainstage operation. Figure 1 shows a typical chart which depicts the shaft speed of the low pressure fuel turbine. Steady-state conditions can be observed from about 100 seconds to 300 seconds and from 320 to 450 seconds after ignition. The power levels are 104% and 100%, respectively. Like most other charts it contains two curves, one for the current test and one from a previous test which the new data are compared against. Comparison charts such as this one are used to identify anomalies in the current test data. A numerical power balance model which predicts critical engine parameter values exists, but it is not used for anomaly detection because it is not reliable enough. The quantitative model also cannot simulate individual component behavior, and its fault simulation capabilities are limited.

Testing an SSME is very complicated, labor-intensive, and expensive. It is not possible to repeat a test just to get more or different measurements. Therefore the problem of

selecting optimal additional points to probe does not arise. To offset the lack of additional measurements an unusually large number of parameters is measured during each test and made available for analysis. A diagnostic technique was developed for EDIS which performs well with a large but fixed number of parameter values.

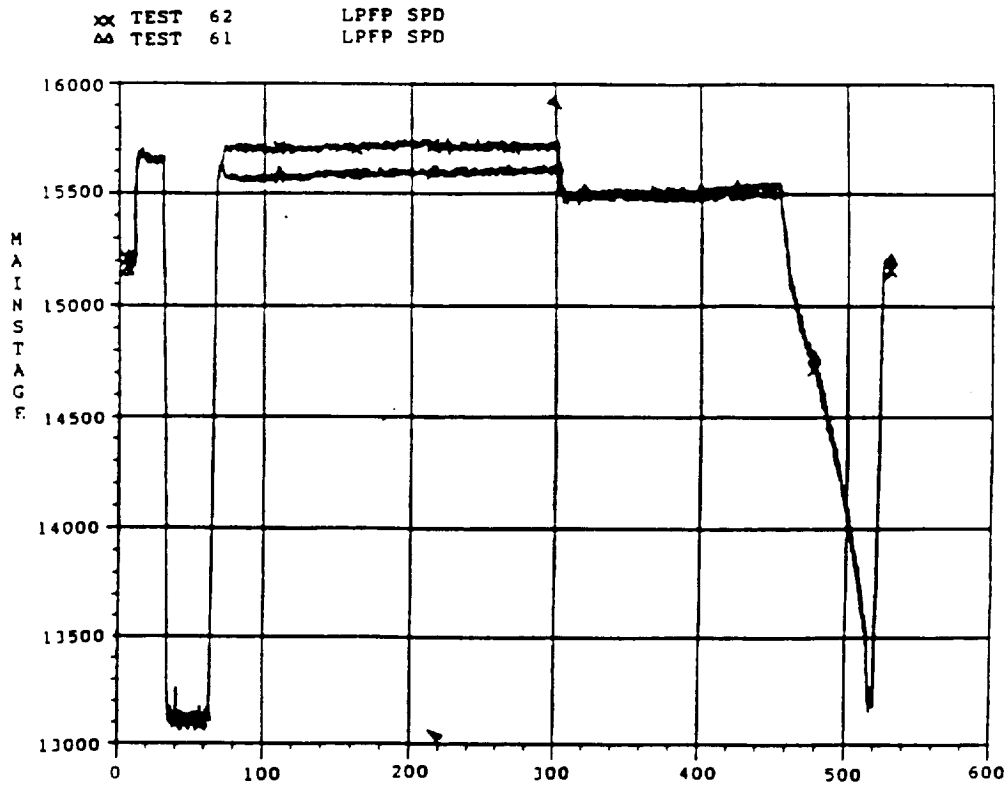

FIGURE 1. TIME FROM START COMMAND - SECONDS

Most first-generation diagnostic expert system use heuristic rules to generate and test hypotheses. Heuristic rules are inadequate for SSME diagnosis because of the complexity of the SSME, because the SSME is continually being modified, and because faults arise with low frequency and high degree of variation. Few faults can be recognized by their symptom patterns. In other words, although the review process is routine, the anomalies and faults encountered are not.

There exist many aspects of the SSME data review process which have been or will be addressed in the EDIS project but are not discussed in this paper. For example, the review

behavior of a number of thermodynamic components are supplied with EDIS. Any particular device is described in terms of its components and their interconnections. Definition of device structure and component parameters, e.g. efficiency or friction coefficients, configure the model of a particular device, see Figure 3.

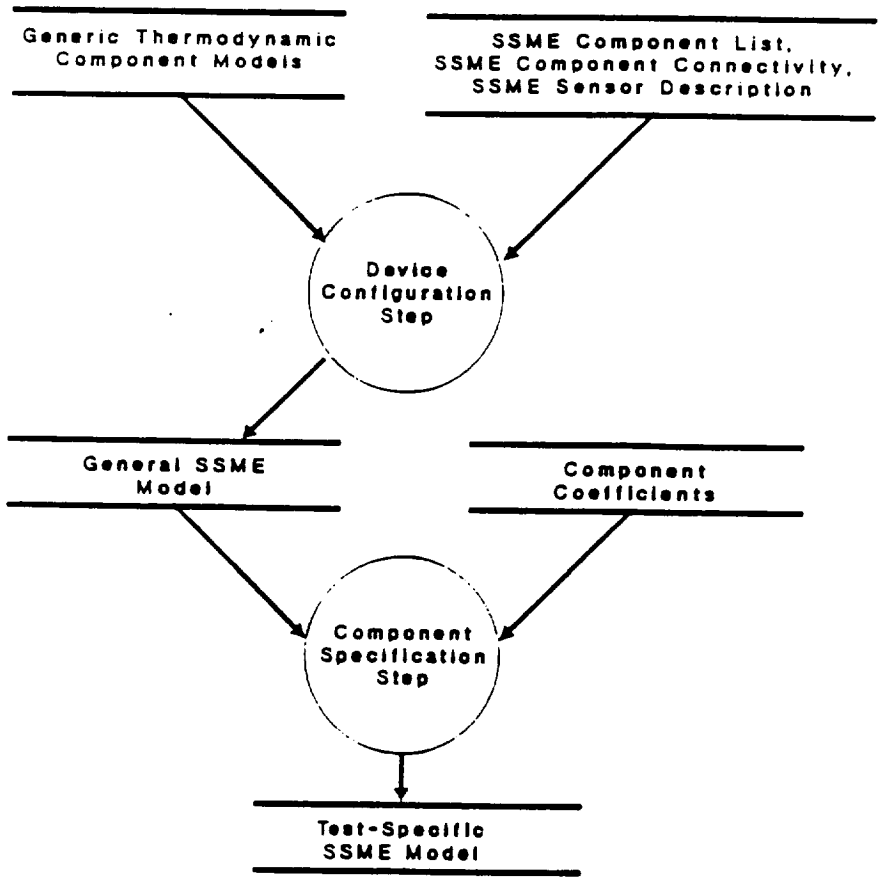

FIGURE 3

3.1 MODEL

Behavior is modeled by incremental qualitative constraints (IQCs) on system parameters. The characteristic parameters of the medium moving through the thermodynamic device constitute the set of system parameters. Typical parameters are pressures and temperatures of fluids or gases and the rotational speed of pumps and turbines. Constraints are incremental because they describe behavior as change or deviation from normal values. The incremental formulation linearizes and simplifies the constraints. Constraints are qualitative because they classify parameter values into three categories: normal, high, and low.

Constraints express qualitative proportionality between deviations of parameter values from the norm, assuming correct device behavior. Constraints are derived from mathematical system models, but unlike Iwasaki and Simon (Iwasaki & Simon, 1986), who derive a qualitative model from the differential equations describing system dynamics, we start with steady-state equations of energy and mass equilibrium. The term "behavior" thus denotes the ordinal interrelations of system parameters in steady-state, instead of an account of the dynamic changes in parameter values. Our type of analysis is labeled "comparative statics" by Kalagnanam, Simon & Iwasaki (1991) in a recent review of qualitative reasoning. Incremental deviations can also be represented quantitatively. Govindaraj (1987) reports the use of a quantitative deviation model to simulate a complicated marine steam power plant. Biswas, Hagins & Debelak (1989) formulate a quantitative process model for a fluid system. Our primary model is qualitative.

Constraints describe local behavior, i.e. they describe the interrelations of parameters associated with each particular component. Parameters are associated with a component if they are defined at one of the terminals of a component, e.g. its inlet, outlet, or shaft, or if they describe medium state within a component. IQCs are thus an example of component-centered qualitative modeling in contrast to process-centered modeling developed by Forbus (1984), which decomposes the device model into processes instead of components. A process definition collects descriptions of the influences on a substance mediated by all the components it is in contact with. In the SSME, where a substance, such as the engine fuel, is engaged in many processes at the same time, a component-centered modeling approach leads to a more satisfactory decomposition. For each component, however, behavior is described as a process which obeys thermodynamic conservation laws. Global behavior is generated by simultaneously satisfying the constraints imposed by all components. Parameters defined at the terminals of components constitute the linkage between component behaviors since their values have to satisfy the constraints of two or more connected components, see Figure 4.

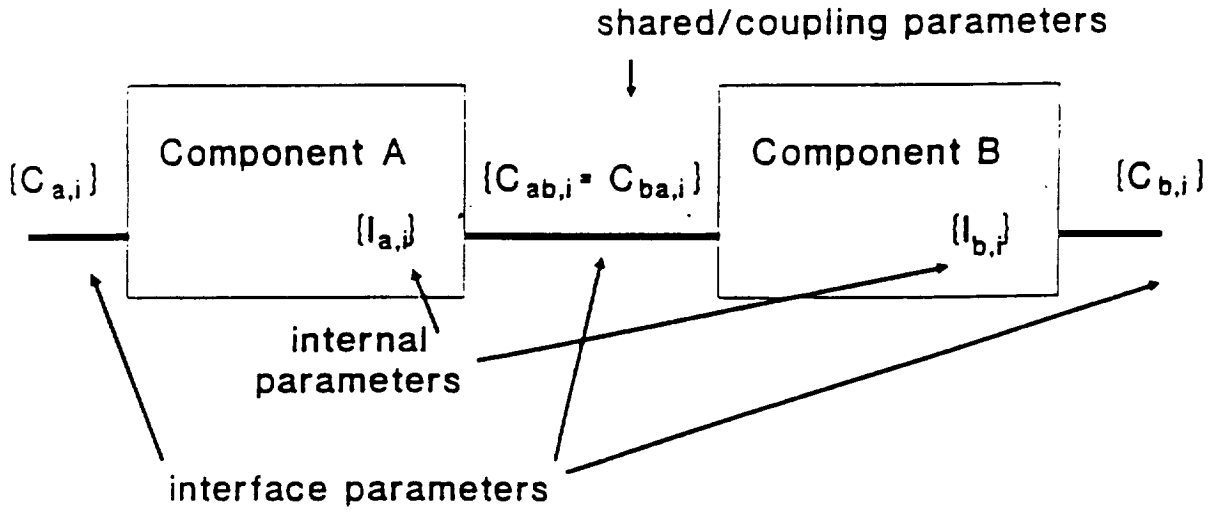

FIGURE 4

For example, the behavior of a pipe is characterized by an energy conservation equation and a mass conservation equation between the pipe inlet (index 1) and the pipe outlet (index 2).

$$\frac{V_1^2}{2g} + h_1 + \frac{p_1}{\gamma} - E_{Loss} = \frac{V_2^2}{2g} + h_2 + \frac{p_1}{\gamma} \tag{1a}$$

$$E_{Loss} = f \frac{\bar{V}^2}{2g} \frac{L}{D} \tag{1b}$$

$$\gamma_1 A_1 V_1 = \gamma_2 A_2 V_2 \tag{2}$$

$\bar{V}$ = average fluid velocity
V_i = fluid velocity
g = gravitational constant
h = height
p = pressure
γ = density
L = pipe length
D = pipe diameter
f = friction coefficient
A_i = pipe cross-sectional area

After linearization and simplification equations (1) and (2) reduce to (3) and (4) respectively. The delta operator (Δ) indicates incremental (small signal) change and K is a constant which depends on the operating point, the pipe dimensions, and the friction coefficient.

$$\Delta(p_1 - p_2) = K \cdot \Delta\bar{V} \tag{3}$$

$$\Delta V_1 = \Delta V_2 \tag{4}$$

The essence of these equations, which is captured by IQCs, is that the pressure difference between inlet and outlet is proportional to the velocity, and that the input velocity is proportional to the output velocity as long as the pipe is operating correctly. Faults which could invalidate the constraints are pipe leaks and obstructions, for example.

It has been shown by Kalagnanam et al. (1991) that the ordinal properties of the involved quantities do not change even under such strong simplifications as long as the simplifying transformations are monotonic. Our simplifications and transformations from quantitative to qualitative models therefore preserve relative magnitude of parameter values. If, for

example, the qualitative model predicts an increase in value then the quantitative model (if it existed) would also predict an increase. Invariance of ordinal properties in essence guarantees that qualitative values are predicted correctly by IQCs.

3.2 DIAGNOSIS

In the presence of faults there will be no parameter value assignment which satisfies all constraints. Davis R. (1984) describes "constraint suspension", a technique which finds the constraint (or constraints) which, if suspended, eliminate all conflicts. The procedure is basically trial and error and demands that a complete analysis be made for each hypothesis. A number of other approaches have been reported which will be discussed when appropriate in the following sections.

EDIS uses a constructive diagnostic paradigm. EDIS attempts to find the most appropriate explanation of the observed symptoms. An explanation of behavior indicates values for all critical parameters and assertions about health states of components. Such an explanation is called a "scenario." Faulty components may exhibit unconstrained behavior, all other components must obey their behavior constraints. Each scenario which takes all measured data into account and satisfies all constraints except for those imposed by components considered faulty generates a fault hypothesis. The hypothesis states that those and only those components whose behavior constraints were disregarded are the cause for the observed malfunction. Finding the qualitative behavior of a system is called "envisioning" in the qualitative modeling literature, e.g. De Kleer & Brown (1985), Forbus (1988). In general, envisioning deals with dynamic behavior of devices described as sets of states and state transitions. EDIS envisions steady-state values of critical engine parameters, i.e. engine states, but no state transitions.

Envisioning scenarios is a constraint satisfaction problem. However, constraint satisfaction alone is not powerful enough to perform diagnosis. Consider a faulty SSME and a set of measurements which renders the fault observable. If all constraints are enforced, no consistent assignment of values to constrained parameters exists. Diagnosis consists of a search for those constraints which, if revoked, permit a consistent value assignment. There are thus two levels of search involved in diagnosis: search for the violated constraint or constraints, which identify the failed component, and search for a consistent value assignment given this fault, i.e. constraint satisfaction. Both types of search are computationally intensive and are supported by knowledge about normal and fault behavior of components.

There are, in general, a vast number of scenarios which fit the measured data and satisfy a subset of component constraints. A diagnostic strategy has been defined which attempts to select a small number of "good" scenarios, i.e. plausible fault hypotheses. Scenario quality is determined from a number of factors which effectively characterize how likely a hypothesis is with respect to other hypotheses. Diagnosis is viewed as search in a space of all scenarios for the scenario which best explains the symptoms. SSME diagnosis is a

domain complex enough to render enumeration and subsequent selection of hypotheses impossible. According to the problem classification scheme described by Stefik, Aikins, Balzer, Benoit, Birnbaum, Hayes-Roth & Sacerdoti (1983) we are dealing with a big, factorable solution space calling for a step-wise generate and test methodology. Therefore, EDIS attempts to sequentially accommodate the constraints contributed by one component at a time, thereby creating partial scenarios. Partial scenarios are evaluated and compared against each other at each step.

Observed anomalies contribute to the envisionment of scenarios through a process of counterfactual reasoning (Adams 1975), which either suspends the constraint and declares the component faulty or puts the blame on anomalous data on the component interfaces. Anomalous values of parameters associated with a component do not by themselves imply a fault of this particular component. Some kinds of anomalous behavior may be due to a shift in operating point caused by a fault in some other part of the SSME. Full knowledge of all critical parameters would be necessary to judge the health of individual components. In reality, assumptions must be made and their consequences tested against all available data and knowledge of SSME behavior.

Generation of a scenario simulates device behavior since it transforms the implicit representation of component behavior as physical constraint laws into value assignments for parameters which characterize states and behaviors of components. Accounts of device behavior can be used to justify fault hypotheses to the end user.

A qualitative model, regardless of whether it is a constraint model or not, is less precise than a quantitative model. Precision becomes important during constraint testing when the relative strengths of opposing changes cannot be predicted. Two kinds of errors can be made by the scenario generator. It can assume that opposing influences cancel out when in fact they do not, i.e. the component is considered to be working correctly when it is in fact faulty; or the scenario generator can assume that a constraint is violated when, in fact, the component is working correctly. In the first case scenarios may exist which do not contain an existing fault, in the second case scenarios will contain too many faults. Our current approach is to generate separate scenarios for both assumption and to use more accurate, e.g. quantitative constraints, to test the validity of assumptions whenever possible.

EDIS operates as an assistant to the user. More difficult to implement than an autonomous expert system, an assistant system is better suited for use by domain experts who are expected to augment system expertise with their own experience in the analysis task. Also, the broad scope of this project and the uncertainty associated with the analysis process, which arise from the diversity of knowledge and from the limitations on data collection and process characterization, favors a system structure which is open to user guidance. For example, the system accepts user-generated fault hypotheses and elaborates and evaluates

them like hypotheses generated by the system itself. A graphical user interface which supports mixed-initiative processing and offers direct manipulation of diagnostic plans and hypotheses is under development.

4. Rationale

A model-based approach was chosen as the main diagnostic paradigm because it provides a declarative formalism for specifying domain knowledge which can be easily updated and improved. A model-based system can more readily be adapted to changing device configurations. Changes to SSME configuration keep happening since improvements to its design are still actively sought. In general, our approach is applicable to a variety of liquid fuel engines and fluid systems in general because the knowledge about device behavior is stored in reusable component models.

A model-based domain knowledge representation facilitates explicit formulation of reasoning strategies. Diagnostic reasoning can be subdivided into discrete tasks which are scheduled by an intelligent strategy module. The device model forms the basis for coordinated performance of the task modules. Data, strategic plans, and intermediate diagnostic results, such as a list of currently active hypotheses, can be accessed and made available to user inspection. Users can more easily understand and be involved in decisions and choices made by the system. Qualitative formulations also help to create reasoning paths which are more easily understood and directed by a user as compared to heuristic rules or quantitative approaches using equation solvers.

Models using constraints naturally represent component behavior at a detailed physical level independent of component use or function within a device. Constraint-based models support behavior simulation as well as verification. A description of device behavior using a constraint model does not presuppose a particular diagnostic methodology. Diagnosis based on a constraint model can operate without any fault assumption, e.g. GDE (de Kleer & Williams 1987), with fault assumptions but without models of fault behavior, e.g. Davis R. (1984), or can operate generatively, like in EDIS. Generative methods, i.e. methods which envision behavior, are computationally more expensive, but envisioned behavior is a useful byproduct of diagnosis because EDIS uses envisioned scenarios to explain and justify hypotheses. In order to reduce complexity, EDIS exploits restrictions on possible fault behavior options imposed by physical laws and domain heuristics. For example, energy and substances cannot be spontaneously created, and in a high pressure fluid system, such as the SSME, fluid can leak out but cannot be added to the system from the environment.

Compiled causal model representations also fail to adequately capture the intricate relations between critical parameters. Compiled causal models suffer from the same shortcomings as heuristic rule-based representation. They do not easily adapt to changes in configuration, and they cannot easily cope with the lack of direction of causal influence between processes and parameters. Constraints, however, can easily represent

Quantity-1 Relational-Operator Quantity-2

The two relational operators are

p ... "is proportional to" and

ip ..."is inversely proportional to".

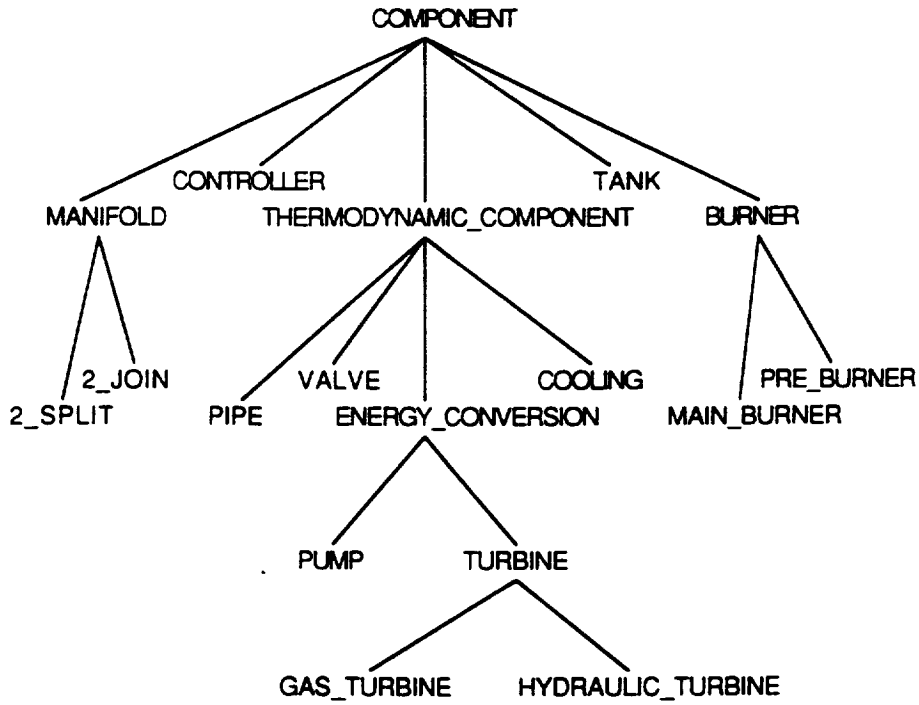

Figure 5: Component Hierarchy (partial).

A quantity is either a state parameter of the fluid or gas, such as pressure, a derived parameter, such as pressure difference, or an explicit measure of energy. We allow only two-place relations in order to simplify reasoning. More complex relations can be represented by introducing derived parameters where necessary and explaining their composition with normative constraints.

The semantics of fundamental and normative constraints and of auxiliary relations differ. A fundamental constraint captures an energy balance which must hold when the component is operating correctly. Faults, in general, are assumed to introduce losses and to invalidate fundamental constraints.

A normative constraint relates a derived quantity which appears in a fundamental or normative constraint to measurable quantities. Since this derived quantity will be normal or "constant" during normal operation, but may deviate from normal in the presence of a fault, it is called a "pseudo-constant." Note however, that a change in a pseudo-constant

by itself does not incriminate a specific component, since its value could have shifted due to a change in operating point. Normative constraints must hold only as long as the derived quantity they depend on remains constant.

Auxiliary relations describe how a change in a pseudo-constant is reflected in the quantities of the normative constraint. An auxiliary relation thus couples a normative relation to a fundamental relation via its pseudo-constant quantity. Normative constraints may be coupled to a fundamental constraint through a chain of other normative constraints in order to deal with more complex cases.

For example, the behavior of a pipe is characterized by fundamental constraints (5) and (6), which are equivalent to equations (3) and (4) above.

$$\text{p-diff}\ p\ \bar{V} \tag{5}$$

$$V_1\ p\ V_2 \tag{6}$$

p-diff = pressure difference

Equation (3) asserts that a small change in the difference between fluid pressures measured at both ends of the pipe is proportional to a change in the velocity of the fluid. The constraint was derived from a fluid energy balance, given by equations (1a) and (1b), neglecting possible differences in height and diameter of the pipe ends. Equation (4) indicates that a change in flow rate at one end must be proportional to a change at the other end. It was derived from a mass conservation law given by equation (2). The pipe has one normative constraint, defined by equation (7).

$$p_1\ p\ p_2 \tag{7}$$

Constraint (7) holds (at least) as long as the pseudo-constant "p-diff" remains constant. One can observe that the normative constraint captures a superficial rule-of-thumb analysis of pipe behavior.

Auxiliary relations are applied when a pseudo-constant has (or is suspected to have) changed and its changes have to be related to changes in the parameters of the normative constraint. In the example the auxiliary relations for p-diff are given by equations (8) and (9) and for $\bar{V}$ by equations (10) and (11).

$$p_1\ p\ p\text{-diff} \tag{8}$$

$$p_2\ ip\ p\text{-diff} \tag{9}$$

$$V_1\ p\ \bar{V} \tag{10}$$

$$V_2\ p\ \bar{V} \tag{11}$$

still exist. If contradictions are found, only faulty behavior is possible. This last rule eliminates creation of invalid hypotheses; it is equivalent to hypothesis testing. All possible and valid behaviors are generated except for physically impossible ones. Therefore, an additional fault hypothesis is generated at each step.

At the end of the reasoning process the most promising complete scenarios are presented to the user as candidate solutions. Each scenario indicates which component or components might have failed and how the failure has affected component behavior.

Next, we will present an example for a single step of the envisionment process. We will use a pipe as the component under consideration since we have already defined its behavior. Assume that a partial scenario exists which determines input pressure and velocity. The task is then to create all possible scenarios augmented by information derived from the behavior of the pipe, i.e. its average velocity, output velocity, pressure difference, and output pressure. we will demonstrate the development of a subset of behaviors. Table 1 lists all possible behaviors.

Table 1

Possible Pipe Behaviors Given $p_1 = l$, $V_1 = n$

(l = low, n = normal, h = high)

CASE		DERIVED VALUES			
No.	Type	V_2	$\bar{V}$	p-diff	P_2
1	normal	n	n	n	l
2	faulty	l	l	l	h
3	faulty	l	l	l	n
4	faulty	l	l	h	l
5	•	l	l	n	l
6	•	l	l	n	n
7	•	n	n	h	l
8	•	n	n	l	n
9	•	n	n	l	h
10	•	l	n	n	l
11	•	l	n	l	h
12	•	l	h	l	n
13	•	l	n	h	l

We will use the constraints associated with the pipe to predict values for the unspecified parameters. First, we will derive normal (non-faulty) behavior. Constraint (6) determines the output velocity to be normal. Auxiliary relations (10) and (11) agree that the average velocity should be normal also. Since constraint (5) predicts that the pressure difference is normal we can use the normative constraint (7) to derive the output pressure to be low. This is case 1 in Table 1.

If fundamental constraint (6) were violated, the output velocity would be low. It cannot be high since fluid cannot be added to the system. This is a case where a potential fault behavior is eliminated because it violates basic, domain-wide assumptions. The average velocity could be normal or low depending on the size of the deviation. In each case we need to investigate the cases for constraint (5) being complied with or violated. In the case where the average velocity is low and constraint (5) is complied with, the pressure difference will be low and the output pressure can thus be high or barely normal. These two cases are numbered 2 and 3 in Table 1.

The predictions generated in this step are added to the parent scenario (the scenario which supplied values for input pressure and velocity), and thus thirteen successor scenarios are generated. If no further data are available, each scenario is evaluated and the best ones are extended further. Evaluation would prefer case 1, because it requires no new fault hypothesis. Cases 4, 5, 10, and 13 would also be ranked high, because they predict low output pressure and flow which is consistent with a common pipe fault, a leak.

If additional data, i.e. measured values of parameters, are available, some scenarios can be eliminated. If, for example, the output pressure is known to be low, then cases 2, 3, 6, 8, 9, 11, and 12 are impossible and the number of cases to be considered has been halved. If it is known to be normal, then only cases 3, 6, 8, and 12 survive. Here, the normal case has been eliminated. Therefore, in the context of this particular parent scenario the pipe must be considered faulty. Note that in general there will be several parent scenarios, each leading to different conclusions.

The diagnostic process terminates after all components have been analyzed. If, in the case described above, the faulty pipe is the only fault required to satisfy all other constraints (using any one of its possible fault behaviors), then the completed scenario would raise the hypothesis that the SSME is malfunctioning because the pipe is faulty. If a complete scenario contains fault behaviors for two or more components, then the hypothesis implies multiple concurrent faults. Thus, no special procedures are necessary to deal with multiple faults. Scenarios which hypothesize multiple, but common, faults compete directly with single, but unusual, fault hypotheses through the heuristic evaluation function.

7. Implementation

At the beginning of the project several expert system shell products were evaluated. Important selection criteria included power of representation and inference mechanisms, ease of creating a custom user interface, portability between various hardware platforms, and ease of integration with existing and future software components. Shells which contained support for the object-oriented paradigm, i.e. support for classes, defaults, inheritance, and instantiation, were preferred. The selected shell also had to run on a personal computer. Our evaluation ranked NEXPERT-Object first and KES second. However, due to budget constraints we selected KES for the initial phase of the project. KES provides backward chaining rules, data driven demons, and a class/member (object-oriented) representation formalism. In addition we purchased a subroutine package which contains support for mathematical functions and graphical data presentation. These subroutines were integrated with KES and provide the user interface framework. KES itself was embedded into a C main program which manages execution of EDIS system tasks.

After encouraging results prompted us to proceed to a second development phase, we decided to change to more powerful tools. We chose NEXPERT as the expert system shell and Toolbook as the user interface environment, both of which run under Microsoft Windows. NEXPERT and Toolbook are linked through function calls and handler scripts using the Dynamic Link Library mechanism of Microsoft Windows.

Currently, we are concentrating on mainstage steady-state anomalies due to, for example, turbo-machinery performance degradation or pipe leaks.

8. Discussion and Future Work

The EDIS system is still in a prototype state and its performance cannot be formally evaluated yet. Instead of discussing results we will contrast our approach to related methods reported in the literature. We believe that mixing diagnostic paradigms and using incremental qualitative constraint models constitute a novel and effective way of diagnosing complex thermodynamic systems such as the SSME. EDIS attempts to solve a difficult problem in a manner which is compatible with human expert behavior. The overt behavior of the process by which diagnosis proceeds from symptom detection to fault identification has rarely before been the focus of attention. Previously, only the characteristics of the process, such as fault coverage, completeness, and efficiency, had been considered. The only exceptions were concerns for the order of questions and additional measurements. An additional benefit of the EDIS methodology is that explaining the rationale behind diagnostic behavior has become easier because system (EDIS) behavior is modeled after the behavior of human experts and meta-reasoning about goals and tool choices is made explicit.

A review of recent publications reveals that applications of expert systems to SSME behavior analysis have been largely focused on real-time detection of malfunctions and

health monitoring (Perry, 1988, Gupta and Ali, 1988). Neural networks (Luce & Govind, 1989, Whitehead, Kiech & Ali, 1990, Luce & Govind 1990) and signal processing techniques (Norman & Taniguchi, 1988, Walker & Baumgartner, 1990) have lately been proposed and developed because they accelerate processing and promise to recognize anomalies and imminent failures fast enough to avert dangerous consequences. There is no need or time to interact with a human user. EDIS, on the other hand, was designed to operate off-line in cooperation with human experts. Emphasis has been placed on smoothly integrating EDIS into the review processes. The ability to explain reasoning and the state of diagnosis and to accept directions from users are important.

De Kleer et al. (1987) have presented GDE, a method for diagnosing single and multiple faults in systems which can be modeled by interconnected modules, each characterized by constraints between input and output parameters. Essentially the same method has also been proposed by Reiter (1987) except that his derivation is based on formal logic. GDE predicts values for device parameters given some known values, e.g. measured or input values, by propagating the known values through the component interconnections and constraint expressions. Note that constraints must be non-directional, i.e. the system must be able to reason from inputs to outputs as well as from outputs to inputs. Davis R. (1984), for example, supplies "simulation" and "inference" rules for forward and backward propagation, respectively. Constraint propagation mechanisms in general are discussed by Davis E. (1987).

The diagnostic paradigm exemplified by GDE is very powerful but some caution is appropriate before recommending it for every diagnostic application. De Kleer et al. (1987) point out that complete prediction of component and system behavior is currently beyond the state-of-the-art. The SSME is a good example of a complex dynamic system whose behavior is very difficult to model and to predict. Also, GDE relies on the ability to take additional measurements until the fault or faults have been uniquely identified. GDE works well when all data at a given location can be measured. EDIS is adapted to deal with many pieces of incomplete information located quasi-randomly within the structure of the SSME. Also, the results gained from propagation can be further manipulated using different diagnostic paradigms, such as heuristics, quantitative simulation, or simply user input. The qualitative reasoning methodology used by EDIS lends itself to integration in a larger diagnostic framework, which is important when satisfactory reasoning behavior cannot be generated using one paradigm alone.

Reasoning about SSME behavior is similar to analyzing electronic circuits at the component level since fluid systems and electric circuits can be modeled by structurally equivalent equations. Stallman & Sussman (1977) presented a constraint-based approach to circuit analysis. Fluid systems have to cope with two additional problems. The structure of a fluid system, especially a high pressure system like the SSME, can easily experience structural changes, i.e. leaks. Leaks correspond to short circuits which are usually acknowledged as being hard to diagnose. The reason is that the device structure has

changed. Also, power sources are more complex in fluid systems. Pumps usually supply constant power to the system, characterized by a constant product of pressure difference times fluid velocity. These differences make it hard to apply the mechanisms developed for electric circuits directly to SSME diagnosis.

Extensions to the work presented in this paper can follow several directions. Analysis of dynamic anomalies, such as drifts, steps, and spikes, in parameter values can be included into EDIS. The SSME is known to experience such anomalies. The qualitative constraint representation will have to be augmented in order to accommodate dynamic behavior, and temporal reasoning capabilities will have to be added. A formalism like Qsim (Kuipers 1985) augmented with temporal information, e.g. time instants and interval duration, might prove to be sufficient.

Further methods and tools representing additional paradigms could be added to the reasoning framework of EDIS. Case-based reasoning would address the reoccurrence of similar faults. Case-based reasoning could supplement qualitative reasoning to envision diagnostic scenarios.

Better, more efficient ways of searching the space of possible behaviors could be explored. For example, the search does not necessarily have to follow structural linkages but could execute opportunistically, concentrating on components which are likely to contribute essential information to the solution. The search could be implemented in a parallel fashion or even cast as a simulated annealing problem if a suitable energy function is formulated.

Facilities which implement limited self-improvement or learning could be added. For example, newly discovered component failure modes could be included into the set of known modes. After that, EDIS would consider scenarios more likely which invoke the stored failure mode. Newly revealed symptom/fault associations could be added to the heuristic rules and used as shortcuts in subsequent diagnoses.

Acknowledgements

This research has been supported, in part, by NASA Contract NAS8-36955, D.O. 97.

References

Adams, E.W. 1975, The Logic of Conditionals, Dordrecht, Holland: D. Reidel.

Biswas G., Hagins W.J. and Debelak K.A. 1989, Qualitative Modeling in Engineering Applications, Proc. 1989 IEEE International Conference on Systems, Man, and Cybernetics, Cambridge, Massachusetts, November 1989, pp. 997-1002.

Cost T.L. and Hofmann M.O. 1990, Engine Data Interpretation System, UAH Research Report, No. ME-90-101, July 1990.

Davis, E. 1987, Constraint Propagation with Interval Labels, Artificial Intelligence, 32(3), 281-331.

Davis R. 1984, Diagnostic Reasoning Based on Structure and Behavior, Artificial Intelligence, 24(1-3), 347-410.

Davis, R. and Hamscher, W. 1988, Model-based Reasoning: Troubleshooting, In: Shrobe, E.H., (ed.), Exploring Artificial Intelligence, California: Morgan Kaufmann, 297-346.

De Kleer, J. and Brown J.S. 1985, A Qualitative Physics Based on Confluences, In: Bobrow D.G. , (ed.), Qualitative Reasoning about Physical Systems. Cambridge, MA: MIT Press, 7-83.

De Kleer, J., Williams, B.C. 1987, Diagnosing Multiple Faults, Artificial Intelligence, 32(1), 97-130.

Forbus K.D. 1984, Qualitative Process Theory, Artificial Intelligence, 24(1-3), 85-168.

Forbus, K.D. 1988, Qualitative Physics: Past, Present, and Future, In: Shrobe, E.H., (ed.), Exploring Artificial Intelligence, California: Morgan Kaufmann, 239-296.

Govindaraj T. 1987, Qualitative Approximation Methodology for Modeling and Simulation of Large Dynamic Systems: Applications to a Marine Power Plant, IEEE Trans. SMC, SMC-17(6), 937-955.

Gupta, U.K. and Ali, M. 1988, LEADER - An Integrated Engine Behavior and Design Analysis Based Real Time Fault Diagnostic Expert System for Shuttle Main Engine, Proc. 2nd IEA/AIE, Tullahoma, Tennessee, pp. 135-145.

Iwasaki, Y. and Simon, H. A. 1986, Causality in Device Behavior, Artificial Intelligence, 29(1), 3-32.

Kalagnanam J., Simon H. and Iwasaki Y. 1991, The Mathematical Bases for Qualitative Reasoning, IEEE Expert, 6(2), 11-19.

Kuipers, B. 1985, Commonsense Reasoning about Causality: Deriving Behavior from Structure, In: Qualitative Reasoning about Physical Systems, Bobrow, D.G., (ed.),Cambridge, Mass.: MIT Press, pp. 169-203.

Luce, H.H. and Govind, R.L. 1989, Prediction and Diagnosis of Failure in the SSME High Pressure Fuel Turbopump Using Backpropagation Neural Networks, Proc. 1st Health Monitoring Conf. for Space Propulsion Systems, Cincinnati, OH, pp. 218-237.

Luce, H.H. and Govind, R.L. 1990, Neural Network Pattern Recognizer for Detection of Failure Modes in the SSME, AIAA Paper No. 90-1893, AIAA 26th Joint Propulsion Conference.

Norman, A.M. and Taniguchi, M. 1988, Development of an Advanced Failure Detection Algorithm for the SSME, AIAA Paper No. 88-3408, AIAA 24th Joint Propulsion Conference.

Perry, J.G. 1988, An Expert Systems Approach to Turbopump Health Monitoring, AIAA paper 88-3117, AIAA 24th Joint Propulsion Conference, Boston.

Reiter, R. 1987, A Theory of Diagnosis from First Principle, Artificial Intelligence, 32(1), 57-95.

Stallman, R.M. and Sussman, G.J. 1977, Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis, Artificial Intelligence, 9,135-196.

Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F., and Sacerdoti, E. 1988, The Architecture of Expert Systems, In: Hayes-Roth, F., Waterman, D.A., and Lenat, D.B., (eds.), Building Expert Systems, Reading, Massachusetts: Addison-Wesley, 89-126.

Walker, B.K. and Baumgartner, E.T. 1990, Comparison of Nonlinear Smoothers and Nonlinear Estimators for Rocket Engine Health Monitoring, AIAA Paper No. 90-1891, AIAA 26th Joint Propulsion Conference.

Whitehead, B., Kiech, E. and Ali, M. 1990, Rocket Engine Diagnostics Using Neural Networks, AIAA Paper No. 90-1892, AIAA 26th Joint Propulsion Conference.

3. USER GUIDE

This section describes the interaction between user and system during a diagnostic consultation. Input options and output formats are explained. We also illustrate the consequences of user choices on system behavior and performance. It should be noted that EDIS operates primarily as an assistant to an expert user and as such can be directed to operate according to user-defined reasoning paths. We start with a description of the files which have to be provided for EDIS to operate correctly. Some of the information given is overly restrictive; system capabilities depend to a large degree on configuration files which may be changed by a knowledgeable user. Section 4, "Developer Guide", will describe system configuration.

The main idea behind the qualitative reasoning system is to generate an account of the behavior of the SSME in terms of its parameters and (possibly) its faults. Behavior is characterized by qualitative values of parameters (normal, high, and low) and by fault hypotheses, e.g. pipe leaks, pump efficiency drops, etc. Fault hypotheses are derived from specific behaviors, i.e. parameter assignments. Since too few parameters are measured by sensors, EDIS generates additional values to complete the behavior model. Generation of values is constrained by thermodynamic laws applicable to particular components but to a large part remains guesswork, in particular at the beginning of diagnosis when local measurements have not yet been propagated through the SSME structure. EDIS tries to pick the most promising choice from the available behavior alternatives and propagates its effects to the rest of the components. Obviously, false starts are possible which lead either to conflicts with other measurements or to highly improbable fault assumptions. In both cases EDIS will back up and select another alternative. Wrong guesses will never cause EDIS to completely discard the correct answer but it may take longer to find it. There may also exist several equally satisfying diagnoses, at least from EDIS' point of view, which may be generated in an order which depends on the choices made. Choices are controlled by heuristic evaluation functions which attempt to emulate the reasoning of a human expert. See the section on performance tuning for more information.

3.1 Preparations Before Running EDIS

EDIS reads files to

- configure the engine (SSME)	[required]
- configure the sensors	[required]
- provide test data	[required]
- provide comparison data	[required]
- provide limit data	[at least one type]
- set fault probabilities	[optional]

An end user has to prepare the test and comparison data files only. All other files do not normally change from test to test. All these files currently reside in the same directory as the EDIS source files. The list of sensors need not be changed in case of sensor failures; missing values present no problem. The configuration has to be updated to recognize new sensors which provide additional measurements, however. Limit data may be changed/updated when required. If you need to check that you have all the necessary files, refer to the "Installation" section later in this document.

Currently, only those parameters which are collected in the 2-sigma data base can be read into EDIS. They are listed in Appendix C. To generate a data file for a particular test, extract data the same way as it is done to include new data into the 2-sigma data base, using program PIDEXT. You need to save the resulting fixed-format ASCII file under any name ending in extension ".DAT". An example file (A1614.DAT) is listed in Appendix D. Create two such files: one for the current test data and one for the data from the test to compare against (most likely the previous test of this engine). Note that the comparison data are not yet used during diagnosis because we did not incorporate limits for the difference between test and comparison values.

3.2 Running EDIS

3.2.1 Cautions

1. To exit from EDIS, ALWAYS choose the EXIT option from the EXIT menu or click on a button labeled STOP, EXIT, etc.

2. When something goes wrong with NEXPERT, EDIS will lock up your computer and you have to reboot. This may happen after a reconfiguration if you are not careful. Any error or warning message from NEXPERT will cause this, e.g. missing files, compilation warnings, if the NEXPERT authorization module is not plugged in, etc.

3. When you are using the development versions of TOOLBOOK and/or NEXPERT you have the power to change ANYTHING. Refrain from doing this unless you are an experienced developer.

4. If you iconify the EDIS window you will see a NEXPERT icon. This is the NEXPERT session which is controlled by EDIS. You can open it and inspect system state if needed (recommended only for experienced users).

5. EDIS uses only one value for each parameter. This value is currently taken at 104% power level and maximum HPFT discharge temperature.

3.2.2 Conventions

A user interacts with EDIS by clicking on buttons, selecting items from menus, filling blanks, and clicking on graphical objects.

Buttons are green if they represent normal choices, e.g. normal modes of operation, acceptance of system suggestions, or simply continuation of processing. Buttons are red if they stop or change the operation of EDIS. Yellowish brown buttons can be clicked for optional activities, e.g. requesting more information.

Fields which have yellow headers with blue text require input from the user, e.g. entry of input file names. Green fields with blue text present information on the progress of EDIS. White fields with red text are displayed during periods of silent system operation.

3.2.3 Invocation

Invoke MicroSoft Windows 3.0 and click on the EDIS icon. This executes the command

<path>\TOOLBOOK\TOOLBOOK.EXE ..\NEXPERT\NXPTBK\EDIS.TBK

where <path> represents the directory path on your machine.

3.2.3.1 Introductory Screen

First, TOOLBOOK is loaded and an introductory screen appears. Click on the green START button. Now NEXPERT and the SSME configuration will be loaded. NEXPERT displays the names of the knowledge-base files being loaded. When done, you enter a screen where EDIS asks for the test and comparison data file names.

3.2.3.2 File Entry Screen

Enter the names of the files you created during the preparation step followed by <RETURN> omitting the extension".DAT". To make corrections you may return the cursor to a field by pressing <RETURN> or by clicking the mouse button in the field. EDIS will give an error message if the files do not exist. In that case you can either stop EDIS or iconify it, open a DOS window, create the files, restore EDIS, and continue normally.

There are three options listed which you may choose by clicking on the box to their left. Only the middle option is implemented at this time. It serves to reformat the 2-sigma limit file so that it can be loaded quickly by NEXPERT. Use this option after a new 2-sigma limit file has been created (check the Developer Guide for details).

Two additional buttons are provided which let you choose between assistant and autonomous mode of operation. In assistant mode (the default) the user may interact with the system during diagnosis and must be present. In autonomous mode EDIS will work quietly until a diagnosis has been found. In the following it is assumed that the system is operating in assistant mode.

Now click the Continue button to load the data files and load the SSME configuration. This may take a while. EDIS first loads the test and comparison data into TOOLBOOK (because only TOOLBOOK can load the fixed ASCII format) and then transfers the data to NEXPERT. NEXPERT then loads the SSME configuration and 2-sigma limit data and immediately executes the anomaly detection algorithm. Three types of limits may be supplied and at least one has to be supplied. Currently, this is the 2-sigma limit file. In the future, heuristic limits for the difference between current and comparison test data may be supplied and also absolute upper and/or lower limits for data may be added. Note that the 2-sigma limits currently in use are too permissive to find some anomalies. The example case will illustrate this.

3.2.3.3 Wait Screen

Before you get to the Anomaly List Screen a "Wait" Screen is displayed so can can see that something is happening. Just watch as the messages change and wait for the next screen to appear.

After this screen some strange glitches appear, i.e. irrelevant screens being displayed for a moment, just ignore those please. I have not been able to get rid of that.

3.2.3.4 Anomaly List Screen

After anomaly detection is complete, a screen is displayed which lists those parameters which were found to be anomalous, i.e. they were either too high (exceeded the +2-sigma limit) or too low (were below -2-sigma limit) as measured from the average value. Since anomaly detection is imperfect, the user may select measurements to add to or delete from the list of anomalies. To add, click on the ADD PARAMETER button and wait for a list of measurements to appear. Then scroll the list to the desired parameter and click on it (may require two clicks). Now select HIGH or LOW as new qualitative value. Click on the DONE button when done. To delete, click on the DELETE PARAM button and click on an entry in the list of anomalies; it will disappear. When finished deleting click on FINISHED. Note that you can only undo a delete action by explicitly re-adding the parameter to the anomaly list as described above.

You can immediately start diagnosis by clicking the green Diagnosis button or you may elect to go to a screen which displays a schematic of the SSME (button "Schematic"). This screen allows to inspect measured values of engine parameters and to see measured data plotted against comparison and limit data.

3.2.3.5 SSME Schematic Screen

A simplified schematic of the SSME is displayed, see Appendix E. The schematic does not quite fit on one screen. Scroll bars are provided to view the clipped portions of the drawing. Components which have anomalous parameters associated with them are drawn in red, all others in black. If an anomalous parameter is associated with several components, e.g. a pressure at the output of a pump and the input of a pipe, only one component is marked in red. We chose the more "important" component, i.e. the pump instead of the pipe, in most cases. This can be changed in the configuration file, see below.

If you click on a component (NOTE: click on a line, not whitespace) a scrollable list of parameter values will be displayed. The numbers indicate the current test values. Anomalous values are displayed in bold print. If you click on one of the entries a comparison chart is drawn which shows the relative values of current, comparison, and 2-sigma limit data. They are labeled with their numeric values. Current test data are drawn in green, comparison data in blue, and 2-sigma limit data in purple. Thus it is possible to inspect the numeric values of all currently available data. The comparison chart uses a relative scale where minimum and maximum values are always shown at the same locations and other values are placed in between according to the scale created by the spread between minimum and maximum.

Clicking on additional components will display additional parameter lists. These lists contain all parameters associated with a component, therefore there is overlap between connected components. The location of the mouse pointer determines where the list is displayed. This feature allows you to control which parts of the schematic will be obscured.

Clicking on additional list items, i.e. parameter names, replaces the comparison chart with the newly selected parameter data.

To remove any list or chart completely you have to click the button "Hide Data" which erases ALL lists and the comparison chart if displayed.

The button "Show Labels" displays labels which identify the major SSME components in the schematic. The labels can be removed by clicking "Hide Labels".

The button "Diagnosis" leaves the anomaly inspection step and starts diagnosis; it is equivalent to the "Diagnosis" button in the previous screen (the anomaly list screen).

The button "Modify Anomaly List" takes you back to the previous screen (the anomaly list screen) where you can add and delete anomalies based on the information gathered from inspecting the data values.

3.2.3.6 Diagnosis Control Screen

Here you are able to inspect and modify the diagnostic strategy. Currently, you have to accept the initial suggestion because only the qualitative reasoning mechanism is implemented.

The EXPLAIN button does nothing yet.

The EVALUATION STRATEGY button takes you to a screen which is a model of how we will let the user modify numeric strategy parameters which tune system performance. Currently you may change the displayed values but they are not used. They should be replaced by more useful parameters, see the section on tuning system performance.

The Continue button starts the diagnostic reasoning process. After a short time the "First Component" screen will appear.

3.2.3.7 First Component Screen

WARNING: The features described below may or may not be implemented in the current version of EDIS!

The performance, i.e. the time it takes to produce a diagnosis and the order in which competing diagnoses are produced, of EDIS is quite sensitive to the order in which component behavior is analyzed (and to the strategy parameters). Currently, the strategic parameters are set so that EDIS attempts to find a faulty component as soon as possible. If, for example, normal and faulty behavior for the first component are possible choices, then EDIS will assume a fault unless the faults are too unlikely. EDIS will then verify the consistency of this assumption with all other measured data and reject it if necessary, but a bad first choice leads to a lot of wasted time. Also, a component which has fewer unknown parameters will lead to fewer behavior choices which accelerates further processing.

Therefore, EDIS a) attempts to make a good guess as to where to start and b) allows the user to change the selection.

For this reason provisions were made in the system architecture to be able to add additional knowledge sources which may optimize the search process, such as heuristic rules. This is not currently implemented, however.

The component EDIS suggests is selected based only on the number of known parameter values associated with it. A pipe, for example, is currently modeled using six parameters. If EDIS knows the values of two of these six parameters it will prefer this pipe to another component where maybe two of eight parameters are known. As you can see, EDIS does not take component fault likelihood into account and a bad first guess is possible. If you

suspect this, change the initial selection by clicking on the CHANGE button; if you want to know more about the component and its associated parameters click on INSPECT; and if you agree click on ACCEPT.

If you choose INSPECT you will see the number of known and unknown parameters and the qualitative values of the parameters associated with the component.

If you choose CHANGE you may choose a different component from a scrollable list of components. For this new component you may again select INSPECT, CHANGE, or ACCEPT.

If you choose ACCEPT you will be asked whether you want to supply additional qualitative data and/or if you want to suggest a fault hypothesis for this component. A good guess on your part can speed up diagnosis. A bad guess will slow down EDIS but will not eliminate the correct diagnosis even if it contradicts your guesses. Your answer will only influence the heuristic evaluation functions so that EDIS' guesses will conform to yours initially. If you advance a fault hypothesis EDIS will first test whether your hypothesis is compatible with all measured data without having to assume additional faults. Note that the above described behavior of EDIS depends on the values of the strategic parameters.

After you choose ACCEPT EDIS will start processing. This can take a long time depending on the number of unknown parameters, e.g. ca. five minutes for four unknown parameters of a pipe.

When EDIS has finished working an a component it will show a status screen.

3.2.3.8 Diagnosis Status Screen

WARNING: The features described below may or may not be implemented in the current version of EDIS!

The status screen indicates whether a complete diagnosis has been found. If so, its quality and details are available for user inspection. If not, the current state and progress made are available.

3.2.3.9 EXIT

When you are done select EXIT from the EXIT menu to leave EDIS.

4. DEVELOPER GUIDE

4.1 System Architecture

The EDIS system consists of two main parts, a knowledge-based reasoning system written in NEXPERT and a user interface written in TOOLBOOK. Both tools use Microsoft Windows and communicate through dynamic link libraries (DLLs). The TOOLBOOK code contains instructions to load and initialize the NEXPERT session. TOOLBOOK code requests expert system processing by suggesting data and volunteering hypotheses (the NEXPERT kind). NEXPERT requests interface support by executing calls to routines which are handled by TOOLBOOK, i.e. for which TOOLBOOK has installed a handler script via a DLL call.

TOOLBOOK code reads the data files while NEXPERT code reads all other files, e.g. 2-sigma limit data, component and parameter configuration, etc. All relevant data are maintained in NEXPERT, changes made by a user interacting with the interface are immediately propagated to NEXPERT.

4.2 Installation

Install TOOLBOOK and NEXPERT first. During NEXPERT installation request that the TOOLBOOK bridge and the Windows DLL be installed. The *.DLL files will be put in the NEXPERT\NXPTBK subdirectory. All EDIS code must reside in this directory!

EDIS code consists of the files

EDIS.TBK	[TOOLBOOK code]
KB111B.TKB	[NEXPERT code: configuration
KB111C.TKB	and
PLANNER.TKB	anomaly detection;
QUALIT.TKB	qualitative
CONSTRN.TKB	diagnosis]

In addition, a set of configuration files must reside in this directory, see below (we may change this so that the configuration files reside in a subdirectory called CONFIG).

NOTE: You must have the NEXPERT authorization module plugged into your PC before executing EDIS.

NOTE: There may be a problem when you run a large application, i.e. one where too many dynamic objects are created. As far as NEXPERT by itself is concerned (without TOOLBOOK) I made the following observations: If you are not developing you may want to replace the NEXPERT.EXE file by the file BIN\STDALONE.EXE which does not contain support for editing but can handle large problems. The development version tends to crash when too many dynamic objects are being created. There is even a version supposedly for large applications called STDALONL.EXE but I could not get it to work properly.

Neuron Data has sent me a pre-release large development version, but said that there may also be a problem with the DLL version (to be used by TOOLBOOK). There will be a large DLL version available which you maybe will have to obtain when you are running EDIS on the full SSME configuration.

4.3 EDIS Configuration

4.3.1 Configuration files

You must provide files which describe the physical SSME configuration, i.e. components and interconnections, which parameters are measured by sensors, and limit data. You should provide a file which lists probabilities of common faults. See Appendix F for examples.

The physical configuration is defined by a set of files, one for each component type, i.e. NEXPERT class. For example, there are files PUMP, PIPE, VALVE, etc. All the component classes are subclasses of class "STATIC COMPONENT" in NEXPERT. Some file names do not directly correspond to class names, e.g. file "gturbine" contains the definition of GAS_TURBINE objects. Check the rule with hypothesis "CONFIGURE_COMPONENTS" to see how the files are loaded and the components are defined. A configuration file defines (among other things) the component name, its connections, and its associated parameters (only those which are measured have to be defined). The connections are specified by entering the name of the connected component in the appropriate slot. The associated parameters are specified by name which must correspond to a name in the parameter configuration file. For each measured parameter you must identify which "generic" parameter it corresponds to. For example, the LPFP_DS_PR is the output pressure, or "pout" for short, of the low pressure fuel pump. The name used must correspond to the name of a slot in the behavior class for this component. (For each component class there exists a corresponding behavior class, see the section on changing the configuration).

The component types currently supported are

Component Class	FILE	Notes
PUMP	PUMP	e.g. LPFP, HPFP
PIPE	PIPE	1-input, 1-output
VALVE	VALVE	e.g. MFV, FPV
GAS_TURBINE	GTURBINE	e.g. HPFT
TANK	TANK	No behavior definition
PRE_BURNER	PBURNER	e.g. FPB
COOLING	COOLING	e.g. MCC_COOLING
TWO_SPLIT	TWOSPLIT	pipe split: 1-input, 2-output
TWO_JOIN	TWOJOIN	pipe join: 2-input, 1-output
THREE_SPLIT	TRISPLIT	pipe split: 1-input, 3-output

One additional file deals with physical configuration. The file TERMINAL lists the names of those components which are not of interest during diagnosis. For example, the fuel tank has to be defined because it is connected to the LPFP but EDIS does not reason about faults of the tank. The tank is therefore listed in the TERMINAL file. The main reason for this file is to isolate the configuration from supporting and bounding components. However, you may list any component in this file and it will be ignored by the reasoning process. This is currently the only way to limit the reasoning process to a particular portion of the SSME as configured.

Measured parameters are defined in file PARAM1.NXP. For each parameter an object of class MEASURED_PARAMETER is created in NEXPERT. The most important entry (NEXPERT slot) is the SENSOR_PID_NUMBER which allows EDIS to transfer data from TOOLBOOK to NEXPERT without enforcing use of identical parameter names. TOOLBOOK contains a page for every measured parameter identified by the PID number. Test and comparison data are read into fields on this page at the beginning of a session, as described in the user guide. TOOLBOOK relies on the order of parameter values in the data files when assigning them to PIDs.

The 2-sigma limit data are stored in file SIGMA.NXP. It has to be created from a data base file in SYMPHONY format. The SYMPHONY file should be called SIGMA5.WR1 so that EDIS can create SIGMA.NXP from it automatically. The symphony file must list the data in columnar format, i.e. transposed from the original format. The data must be explicit, not defined as formulas. The header row and data records must be defined as a range named TRY. The range should only include the 104% MAX FUEL data recods.

The file named FAULTS lists common faults and their probabilities. You specify a component type, e.g. "pipe", a fault mode, e.g. "leak", and a probability, e.g. "0.1". The probability is used to rank competing fault hypotheses. An unknown fault, i.e. one which

has no name, has probability 0.01. The rating and ranking method is described in more detail in the section on performance tuning.

4.4 Changing the Configuration

4.4.1 Simple Configuration Changes

Only the component configuration files have to be modified in order to change the physical configuration of the SSME as long as no new component types are introduced. In fact, EDIS will work for any liquid fuel rocket engine or any high-pressure fluid system consisting of pipes, pumps, valves, etc. Neither the component types nor the behavior models are specific to the SSME. Just make sure the component configuration files are complete and consistent.

A configuration change invalidates the schematic drawing, however. It will have to be updated by hand using TOOLBOOK's author mode. Whether you use line drawings, as was done for the SSME, or other means to create the drawing, it is important that you define objects, e.g. groups of lines, which correspond to each component and give each group the name of the component as defined in NEXPERT. If this is done EDIS will be able to access and display information on the parameters associated with the component as described above.

4.4.2 Sensor Changes

Changes have to be made in both TOOLBOOK and NEXPERT if additional sensors are implemented. For NEXPERT, the configuration file PARAM1.NXP and the files which define the components associated with the newly measured parameter have to be updated. For TOOLBOOK, a page has to be added for each new parameter and the algorithm which reads the data from the ASCII file has to be modified. We regret this inconvenience but the data files do not contain any information on which parameters are contained. Also be sure to update the algorithms for both current and comparison test data. (This should be cleaned up.) The algorithm is in the book script in the handler "loadTestData".

4.4.3 Focusing on Part of the SSME

In order to focus on only part of the SSME, i.e. a subset of the components of the SSME, the fine TERMINAL can be modified to include all those components which should be neglected.

4.4.4 Adding Fault Types

Fault types are easily added in the FAULTS file. Follow the style of the entries already there. The probability should probably be around 0.1 so that it is higher than that of an unknown fault (0.01) and lower than 1.0. Also, adjust all fault probabilities in this file so

that they reflect your judgement and experience about the relative likelihoods of faults of the SSME. The absolute values are not too important but you should implement a rank ordering among faults so that the more likely ones have higher probabilities then the less likely ones.

4.4.5 Modifying the Graphical Data Presentation

You may want to to modify the graphical presentation of data on the schematic drawing screen when additional limit data become available. The script for handler "showGraph" on page "schdiag" should be clear enough to allow you to adapt the algorithm.

4.4.6 Adding A Component Type

Knowledge about new component types has to be added to NEXPERT when configuration changes call for components which have not yet been described to EDIS. This is the most difficult change in configuration and requires special care. In general, try to reuse as much of the code present and use it as a template when additions cannot be avoided. Also, be careful to add new classes and rules to the appropriate knowledge bases so that they are loaded in correct sequence. A mistake here could cause NEXPERT to display warning messages which will lock up your Windows system. Throughout we will pretend to add the type VALVE to the system to illustrate the procedure.

Choose a name for the component type, e.g. VALVE. Identify its connections to other components, e.g. medium input and output, and position control input. Identify its internal and external parameters. Here we may note that the new component is similar to an already existing component. For example, the valve is structurally similar to a pipe since both have a single medium input and output. Its behavior is, in part, a subset of the behavior of a pump, because in both cases we may choose to ignore the possibility of leaks and a pressure change takes place. Structure and behavior are implemented separately.

4.4.6.1 Structure Definition

In KB111B.TKB create a class with the name chosen for the new component type, and make it a subclass of an appropriate parent class. Display the class hierarchy starting at STATIC_COMPONENT to identify a good place. You only need to watch for connectivity similarities. For example, we make VALVE a subclass of THERMO_COMPONENT and inherit the ASSOCIATE_PARAMETERS, GENERIC_PARAMETERS, MEDIUM, MEDIUM_INPUT, MEDIUM_OUPUT, and NAME slots. We now add a slot for the connection to the controller, e.g. CONTROLLED_BY. The last four slots define which (generic) parameters in the neighboring components are equivalent to the interface parameters of a particular valve. Normally, the output pressure "pout" would be equivalent to the input pressure of the next component which might be called "pin" and you may think that this could be assumed automatically. However, some components have two inputs, such as pipe joins and preburners, and use "pin" and "pinB" or "pin_OX" as parameter

names. We recommend that you define a meta-slot for each of these slots which initializes it to the most common name, e.g. "pout" here. If you do this the configuration files need only specify deviations from this normal assumption. It is convenient to use the names of the generic parameters as defined in the behavior class in these slots, e.g. pin, pout, Vin, Vout, commanded_position. These names are the names of slots in the behavior class.

```
(@CLASS=     VALVE
        (@PROPERTIES=
                ASSOCIATE_PARAMETERS
                CONTROLLED_BY
                GENERIC_PARAMETERS
                MEDIUM
                MEDIUM_INPUT
                MEDIUM_OUTPUT
                NAME
                parameter_coupled_to_pin
                parameter_coupled_to_pout
                parameter_coupled_to_Vin
                parameter_coupled_to_Vout
                parameter_coupled_to_commanded_position
        )
)
```

4.4.6.2 Behavior Definition

Behavior definition is more difficult. In QUALIT.TKB create a behavior class named <new component>_BEHAVIOR, e.g. VALVE_BEHAVIOR, and make it a subclass of COMPONENT_BEHAVIOR. Now you have to modify some slots which apply to all behaviors, add component-specific slots, define meta-slot procedures for some of these slots, and create rules which manipulate slot values.

```
(@CLASS=     VALVE_BEHAVIOR
        (@PROPERTIES=
                add_fault_type
                commanded_position
                comp_name
                control_slot
                exchange_hypo
                fault_hypo
                fault_type
                faulty
                local_quality
                neighbors
                next_slot
```

```
                normal_quality
                num_known_params
                num_unknown_params
                open_parameters
                p_diff
                pin
                position
                pout
                temporary
                Vbar
                Vin
                Vout
        )
)
```

Behavior should be defined in terms of constraints on parameter values derived from energy and mass conservation equations. The derivation of constraints is described in chapter 2. For implementation, fundamental constraints lead to rules which detect component faults. Normative constraints and auxiliary equations together are implemented as rules which determine whether a given assignment of parameter values is physically and mathematically possible. Rules based on mathematical constraints relate derived to measured parameters, e.g. pressure difference to input and output pressures.

4.4.6.2.1 Additional Slots

In this example we added slots for input and output pressure and temperature (pin, pout, Vin, Vout), actual and commanded valve position (position, commanded_position), and a slot which lists the neighbors of the component. The "neighbors" slot should actually reside in the structure definition.

The parameter slots, e.g. pin, pout, require an "Order of Sources" and an "IF Changed" meta-slot method. The "Order of Sources" slot is the same for all parameters, copy it from another behavior class. The "If Changed" slot is used to either set other parameters and/or, more importantly, to test the possibility of the behavior specified against the behavior constraints for this component type. In some cases, the parameter may have any value and does not require constraint testing. For example, we neglect the possibility of leaks in a pump and assume that input, output, and average fluid velocity (or flow) in the pump are identical. If one of these three parameters changes so must the other two. The "If Changed" slot enforces this. Look at the Vin and Vout parameters of PUMP_BEHAVIOR for examples. We suggest to make the same assumption for the VALVE. Any leak will thus be attributed to the pipes leading to and away from a valve. If the parameter is subject to constraint testing, however, the "If Changed" slot will suggest one or several hypotheses which trigger constraint rules. These rules test if, after enough parameter values have been

determined, a behavior constraint can be shown to be violated. If so, the particular behavior is be discarded. You may want to wait until you have defined the constraint rules before you complete such a meta-slot. These constraint rules identify the impossible behaviors and mark them for later deletion. Look at the "pin" parameter of PUMP_BEHAVIOR for an example. Both cases discussed above may apply to a parameter, e.g. the Vbar parameter of PUMP_BEHAVIOR. It was chosen at random (from Vin, Vout, Vbar which are all the same) to represent fluid velocity in the constraints. Therefore it has both statements which update Vin and Vout as well as statements which trigger constraint rules.

Every component needs a "neighbors" slot with associated "Order of Sources" meta-slot which defines how to find the neighboring, i.e. connected, components. Part of this procedure is accomplished by a meta-slot in class TWO_PORT_BEHAVIOR. If the new component has fluid input and output like PIPE or VALVE you can make it a subclass of TWO_PORT_BEHAVIOR and you only have to add neighboring components which are attached to additional interfaces. A VALVE has an additional neighbor via the CONTROLLED_BY slot and we add to the meta-slot definition. Even if only fluid input and output are present, the meta-slot method has to be completed for the new component, as can be seen in class PIPE_BEHAVIOR.

4.4.6.2.2 New Slot Names

Try to reuse existing names, such as Vin, Vout, etc. as much as possible to describe component parameters. If you have to define new names, such as "position" in our VALVE example, you have to add additional information.

For each new parameter name define rule called "expand_behavior_<name>". It is best to copy this rule from an existing parameter, e.g. pin, and change the parameter name everywhere. Create a "fill..." rule for the parameter. It is best to copy this rule from an existing parameter, e.g. pout (rule fill2), and change the parameter name everywhere. In the example, "position" and "commanded_position" are new parameters. Create pair of rules "OPEN_<name>_t" and "OPEN_<name>_f" which test whether the parameter has a value yet. Compare rules "OPEN_Vin_t" and "OPEN_Vin_f", for example.

Finally, if the parameter is an interface parameter, you have to determine which parameter in any neighboring component it may made to be equivalent to. For example, Vout may be equivalent to Vin, but also VinB and Vin_OX. You have to check all existing components which my be connected to this interface. In the example "commanded_position" is a new interface parameter. Since the valve position input may only be connected to the engine controller its "commanded_position" parameter may only be equivalent to the "commanded_position" parameter of the engine controller. You need to create a "x_data_<name>" rule for each equivalence. It is best to copy this rule from an existing parameter, e.g. rule x_data_Vout1, and change the parameter name everywhere.

In the example, we need a "x_data_commanded_position" rule. Also, for each pair of possible equivalent parameters you need a rule which checks for possible value conflicts. These rules are named "x_inc_<param1>_<param2>". Again, copy and adapt an existing rule.

This seems to be unnecessarily complicated but the extra code is necessary because NEXPERT does not allow indirect specification of slot names.

4.4.6.2.3 Modified Slots

The "Order of Sources" meta-slot method of slot "add_fault_type" triggers rules which identify particular fault modes of a component, e.g. leaks or obstructions in a pipe. These rules may or may not be present but should be provided if some fault behaviors are more common than others. Write rules which identify the fault behavior and add its probability to the FAULTS file. If you do not specify the meta-slot it will be inherited from COMPONENT_BEHAVIOR which covers the cases of leaks and obstructions in a conduit. An example rule is "leak_is_present_1".

The "exchange_hypo" slot has to be initialized to a hypothesis which exchanges data values with the neighboring components. Compare slot PIPE_BEHAVIOR.exchange_hypo and hypothesis PIPE_X_DATA of rule "pipe_x_data1". These rules make sure that parameters at interfaces between components are shared. A valve shares data with three neighbors: the commanded_position with the controller in addition to pressures and velocities at the medium input and output.

The "fault_hypo" slot has to be initialized to a hypothesis which determines whether the current behavior is faulty. At this point no specific fault mode has to be recognized. In general, there will be some cases which correspond to a known fault mode or behavior and some which are unknown or "strange", all of which have to be detected with these rules. Compare rule "pump_test_fault1". These rules follow from the fundamental constraints of the component.

The "num_unknown_params" slot has to be initialized to the number of parameters associated with the component. Groups of parameters which are identical, such as Vin, Vout, and Vbar in the pump, count only once.

The "open_parameters" slot needs an "Order of Sources" meta-slot method which executes a rule that determines which parameters have not received a value yet. Compare, for example, the slot PUMP.open_parameters and the rule leading to hypothesis "CHECK_OPEN_PARAMS_PUMP". All internal and external parameters must be tested, each with its own rule. Only one of a group of identical parameters has to be checked. The rules which test each parameter should already exist at this point. You may have added some in step 4.4.6.2.2 above.

4.4.6.3 Constraint Rules

Most of the required rules have been discussed in enough detail in the previous sections. This section will add information about constraint rules.

EDIS creates possible component behaviors by enumerating all possible values for each undetermined parameter. After enumeration physically or mathematically impossible behaviors are discarded. The rules which are triggered by the "If Changed" methods of the parameter slots in a component behavior object perform this elimination task. Therefore, the left hand side of these rules matches to impossible value assignments. Sometimes it is not perfectly obvious if a behavior, i.e. or value assignment, is possible due to the uncertainty associated with qualitative values. For example, even if input and output pressure are considered normal, the pressure difference may still be considered high if the input and output pressures are just barely normal at opposite ends of the scale. The judgement depends on the somewhat arbitrary placement of the limits between normal high and low. Often, if we assume a consistent limit definition, some cases can be eliminated. In the above example, we may know that the pressure difference will be normal for all values of input and output pressure which are considered normal. This stricter assumption will eliminate more of the generated scenarios. Fewer behaviors mean faster execution. Our approach is to locate rules which are sure to eliminate only impossible cases in knowledge base QUALIT.TKB and rules which apply only under strict assumptions about choice of limits in file CONSTRN.TKB. Currently file CONSTRN.TKB is loaded automatically, but you may simply remove the load command. EDIS will then consider more cases which will, in general, slow down the system. On the other hand, it may be possible that the additional strict rules eliminate the only reasonable description of SSME behavior, although we think that this is unlikely.

Originally constraint rules were created for each component type individually. For example, PUMP_TEST_POSSIBLE_PV eliminate behaviors which incorporate impossible combinations of labels assigned to parameters PV_Product, p-diff, and Vbar. The rule is derived from the constraint PV_Product "p" p-diff + Vbar. Note that the implementation makes use of three-place (and one four-place) constraints in contrast to the theoretical exposition in Chapter 2 which allows only two-place constraints. The above constraint was derived from the mathematical relationship which defines the derived parameter PV_Product as the product of p-diff and Vbar. Since PV_Product increases with both p-diff and Vbar their relationship can be described by an additive constraint. The majority of relationships encountered can be represented by additive or subtractive constraints.

Currently, we are adding generic methods, i.e. rules, for testing additive and subtractive constraints. The If-Changed method of each parameter slot involved in such a constraint copies the relevant parameter values into slots "X", "Y", and "Z" of object "GCO" (short for generic constraint object). Additive, i.e. X=Y+Z, and subtractive, i.e. X=Y-Z, constraints can be tested by suggesting hypotheses CONSTRAINT_ADD_POSSIBLE and

CONSTRAINT_SUB_POSSIBLE, respectively. Make sure that the behavior object being examined is linked to class TEMP_BEHAVIOR as these two rules assume that. If you do not want or cannot use these generic rules you have to provide explicit rules similar to the rules with hypothesis PUMP_TEST_POSSIBLE_PV discussed above.

4.5 Tuning System Performance and Strategic Parameters

The most significant tradeoff is made when you decide whether to load the knowledge-base CONSTRN.TKB or not. The decision to load it restricts the number of solutions EDIS will pursue, see above. All other performance tuning mechanisms change only the order in which alternatives are considered. This may still turn out to be significant because the execution time to enumerate all solutions is probably too long to consider finite. Solutions which would be generated "late" are thus never generated.

Tuning may be required for selection of the first component to analyze, for selection of subsequent components to analyze, and for evaluation of partial solutions.

The first component is selected based only on the ratio of known to unknown parameters but the choice may be overridden by the user, see section 3.2.3.7.

Given a partial solution the next component to be analyzed is again chosen based on the ratio of known to unknown parameters, except that components which are connected to the last analyzed component are given precedence. This generates a less "jumpy" flow of analysis.

Partial solutions, i.e. scenarios, are evaluated using a number of contributing factors which are rated individually. A composite numeric score is accumulated and the best partial solution is chosen for further expansion until all components have been analyzed. Contributing factors are the number of faulty components in the partial scenario and the score or probability of each fault. The scores for the number of faults are defined in order in slot "quality" of object SCENARIO_EVALUATION_OBJECT. The values are 1.0 for no fault, 90.0 for one fault, 8.0 for two faults, and 1.0 for three or more simultaneous faults. For each fault, its probability is sought. If it was defined in file FAULTS the probability given there is used, if not 0.01 is used. These values are multiplied to the partial score. Finally, the score of each partial scenario is increased by the number of components is has already analyzed. This eliminates overly frequent shifts of the focus of attention.

For example, the score of a partial scenario with one fault, a pipe leak, is 90.0 times 0.1, which is the probability measure defined for pipe leaks in the current FAULTS file. The total score is thus 9.0. A further fault would reduce the score to near 1.0. In fact, a partial scenario with no faults is considered to be about as likely as one with two faults. This is basically sound, but since the scoring does not (or hardly) depend on how far the

reasoning has progressed, a single fault will be assumed at the first component analyzed, unless there is no known fault behavior which fits the available data. In the latter case the fault score of 0.01 will make the combined score 0.9 which is less than 1.0 for the no fault case. You may want to experiment and modify these numbers.

5. STATE OF DEVELOPMENT

5.1 Hardware and Sofware Setup

EDIS runs on 80386 based personal computers with color VGA display. Between ten and twenty megabytes of hard disk space are required depending on which portions of the tools are loaded. Four megabytes of RAM, a math coprocessor, and a mouse are recommended.

EDIS uses the software tools NEXPERT, an expert system shell, and TOOLBOOK, a graphical user interface design tool. Both software tools require Microsoft Windows 3.0. To run EDIS only runtime licenses are needed, but the experimental nature of EDIS makes development licenses desirable.

5.2 Interface

A large portion of the user interface has been implemented, especially the parts which deal with setup, configuration, initialization, and anomaly detection. A dynamic image of the SSME schematic is available which shows parameter values relative to limits. The interface to the qualitative diagnosis mechanism is only partially available. The explanation facility has not been coded yet.

5.3 Knowledge Base

The qualitative reasoning mechanism has been implemented and a limited number of component types are supported. No other types of reasoning are supported yet.

Currently structure models of component types PIPE, PUMP, COOLING, GAS_TURBINE, VALVE, TWO_SPLIT, THREE_SPLIT, TWO_JOIN, PRE_BURNER, and TANK are implemented. Complete behavior models exist for PIPE, PUMP, COOLING, TWO_SPLIT, THREE_SPLIT, and TWO_JOIN. In all cases only pressures and temperatures are considered, temperatures and heat transfer is ignored (which makes COOLING just like PIPE).

6. SAMPLE CASES

Two simple sample cases will be presented in this chapter. Both are based on a small configuration of pipes which is not related to the SSME but is simple enough to explain the reasoning process performed by EDIS. The configuration consists of a pipe F101 which feeds into a pipe-split M102 which, in turn, feeds into pipes F102 and F103. Note that the configuration files have to define additional components which serve to terminate the input and outputs but will not be analyzed.

In the first case three measurements are available: input pressure and flow at pipe F101 and input velocity at pipe F102. Normally, a measurement at the input of F102 would be shared with the output of M102 which is connected to the input of F102. In this example we omit this parameter sharing in order to demonstrate a special case of value propagation. All measured values are NORMAL.

In the second case an additional measurement is available: the output pressure at pipe F103. It is also NORMAL.

The examples will demonstrate how EDIS analyzes a configuration component by component, assures consistency between components, and backtracks, if necessary, when assumptions become unlikely when new measurements are encountered. The second case demonstrates the backtracking mechanism in particular. The examples will demonstrate how EDIS, using its default strategy, tries to find a single fault as soon as possible. You can see how this may lead to inefficiencies when the initial fault assumption is incorrect. Remember that this behavior depends on the strategy parameters described in a previous chapter and can be easily modified. Also, the user is given the opportunity to influence behavior and fault assumptions. Here we show how operates without user intervention. It should be clear now that the initial hypothesis generated by EDIS will contain a fault even though all measured parameter values are normal.

6.1 Case 1: No Backtracking

EDIS selects a component to analyze. At first the only criterion is derived from the ratio of the number of known parameters versus the number of unknown parameters. A pipe has 6 parameters and a pipe-split has 4. We know two of the six parameters of F101, therefore it will be analyzed first. We assume that the pressure is constant throughout the pipe-split and use the generic parameter name "pin" for it.

All possible behaviors of pipe F101 are created. Given the two measurements nine behaviors survive constraint testing. Some of them describe normal behavior but most imply a fault of the pipe. In this version of EDIS two faults are defined for a pipe: an obstruction with merit 0.2 and a leak with merit 0.1, which indicates that we believe that an obstruction is more likely than a leak. A leak is identified by a drop in flow from input to output while an obstruction causes an undue pressure drop.

Each behavior of pipe F101 generates a new partial scenario. Each scenario is rated based on the number of faults it predicts, the merit of each behavior it contains, and how many components it already describes. The basic ratings for no, one, two, and three or more faults are 1.0, 90.0, 8.0, and 1.0 respectively. The appropriate base rating is then multiplied by the rating of each behavior contained in the scenario. Normal behavior is rated 1.0 while faulty behavior is rated by the figure of merit associated with the fault. An unidentified fault has merit 0.01 and will therefore not be considered until all other options have failed.

EDIS selects the most promising partial scenario, i.e. the one with the highest rating, to expand. This is because EDIS actually performs best-first search in the space of all SSME behaviors. In this case the highest rated scenario is Scenario-4 which assumes an obstruction in pipe F101. A consequence of this fault assumption is that the output pressure of F101 will be assumed to be LOW and this value is propagated to pipe-split M102, together with the output flow which is NORMAL in this case.

Next, all possible behaviors of M102 are generated. Since two of its four parameters were determined via propagation only three or four behaviors are possible. Propagation of the output flow to F102 may lead to a conflict with the measured value at the input of F102. Scenarios which lead to such a conflict are discarded. In our case only a single behavior of M102 is consistent with the interface conditions. It predicts normal flows and low pressure.

Next, pipe F103 is analyzed. This time 15 behaviors are possible and EDIS chooses a normal, i.e. non-faulty, behavior since the scenario ratings for the two-fault cases are obviously worse. The chosen behavior predicts normal flows and pressure drop but low input and output pressures.

Finally, pipe F102 is analyzed. EDIS chooses the same behavior as for F103. Now the scenario is complete. It contains a complete account of SSME behavior in terms of all relevant parameters and the fault hypothesis that pipe F101 is obstructed. It is clear that other explanations of the measurements are possible because three measurements are far too few to lead to a unique answer. The search strategy and fault likelihoods supplied to EDIS led to this particular answer. EDIS could, if requested, generate further answers if we force it to backtrack.

6.2 Case 2: Backtracking

The second case is identical to the first except that a measurement for the output pressure of pipe F103 is added. The value supplied is NORMAL. Processing begins the same way and the first two steps are identical: EDIS considers F101 first and chooses a behavior which implies a pipe obstruction. Note that the output pressure of F101 is assumed to be LOW according to this fault hypothesis. Analysis of pipe-split M102, too, leads to the same result as before.

When EDIS reaches pipe F103, however, it cannot find a normal behavior for F103 given the propagated input values and the measured output pressure. It has to assume a second fault: a leak, for example. This drops the rating of the resulting scenarios below the rating of earlier scenarios which predict only a single fault. The bonus for covering more components is less than the penalty for a two-fault hypothesis. EDIS therefore abandons these scenarios in favor of a scenario which has the highest rating at this point. In this case it is a scenario with only F101 analyzed where F101 is again assumed to be faulty, but the fault is supposed to be a leak instead of an obstruction. There are, however, a number of behaviors possible given that the pipe has a leak. EDIS chooses one at random since they all have the same rating. It turns out that again no single fault hypothesis can be supported by this choice of behavior for pipe F101 and EDIS has to put aside its assumption in favor of a different one.

Another behavior for F101 is now chosen, again with a leak fault, and it fails also. Finally, a forth behavior of pipe F101 is selected which leads to a complete hypothesis with only one fault. Interestingly, the fault proposed is again a leak in pipe F101, supported by a different behavior of F101 given the leak. This behavior predicts LOW output flow (this is the characteristic of a leak) and also LOW output pressure. The pipe-split behavior chosen assumes that the input flow is divided so that the output flow into F103 is LOW but the output flow into F102 is (maybe just barely) NORMAL. The low flow through F103 then creates a reduced pressure drop in pipe F103 which allows for a NORMAL output pressure reading even though the input pressure was LOW. F102 sees LOW input pressure and NORMAL input flow and EDIS assumes that its output pressure and flow are LOW and NORMAL, respectively.

The resulting scenario is considered optimal by EDIS and is offered as a valid diagnosis. Note, however, that with such few measurements several other answers would have been possible. Obviously, EDIS could have derived that there is no fault in the system since all measurements are NORMAL. Or, the output flow from the pipe-split into pipe F102 could have been assumed to be LOW just as for pipe F103. Then, the behavior of F102 would have been identical to that of pipe F103. A measurement of the output pressure of F102 could distinguish between these two cases. Without additional data the choice of answers (actually, the choice of the order in which the answers are produced by EDIS) depends on the strategy parameters.

APPENDICES

A: Listing of EDIS Interface (TOOLBOOK) Code

B: Listing of EDIS Diagnosis (NEXPERT) Code

C: Parameters Used by EDIS

D: Example Data Input File

E: SSME Schematic

F: Configuration Files

APPENDIX A

LISTING OF EDIS INTERFACE (TOOLBOOK) CODE

APPENDIX B

LISTING OF EDIS DIAGNOSIS (NEXPERT) CODE

APPENDIX C

PARAMETERS USED BY EDIS

SEQ	PID	DESCRIPTION
1		TIME IN TEST THE DATA VALUES WERE TAKEN (COLUMN I IN SIGMA.WR1 FILE)
2	287	MCC PC REFERENCE
3	63	MCC PC
4	86	LPFP DS PR
5	15	LPFP DS TMP
6	32	LPFP SPEED
7	52	HPFP DS PR
8	17	MCC CLNT DS PR
9	18	MCC CLNT DS TMP
10	231	HPFT DS TMP A
11	232	HPFT DS TMP B
12	58	FPB PC
13	260	HPFP SPEED
14	53	HPFP CLNT LNR PR
15	24	MCC FUEL INJECTOR PR
16	209	LPOP DS PR
17	30	LPOP SPEED
18	90	HPOP DS PR
19	21	LOX DOME TMP
20	59	PBP DS PR
21	94	PBP DS TMP
22	233	HPOT DS TMP A
23	234	HPOT DS TMP B
24	1021	ENG FUEL INLET TMP
25	821	ENG FUEL INLET PR
26	1058	ENG OX INLET TMP
27	858	ENG OX INLET PR
28	100	FUEL FLOW
29	659	HPFP DS TMP
30	457	HPFP BAL CAV PR
31	650	HPFP COOLANT LINER TMP
32	657	HPFP DRAIN PR
33	658	HPFP DRAIN TMP
34	436	LPFT INLET PR
35	835	FUEL PRESSURANT INTERFACE PR
36	1035	FUEL PRESSURANT INTERFACE TMP
37	480	OPB PC
38	142	FPOV POSITION
39	764	HPFP SPEED
40	754	LPFP SPEED
41	327	HPOP BAL CAV PR A
42	328	HPOP BAL CAV PR B
43	595	MCC LOX INJECTOR TMP
44	395	MCC LOX INJECTOR PR
45	221	POGO PRE-CHARGE PR
46	34	HEAT EXCHANGER DS PR
47	878	HEAT EXCHANGER INTERFACE PR
48	879	HEAT EXCHANGER INTERFACE TMP
49	140	OPOV POSITION
50	734	LPOP SPEED
51	2	HPOP SPEED

APPENDIX D

EXAMPLE DATA INPUT FILE

1	A1061408									
2	14.00	498.00	2705.38	2704.49	220.23	42.56	14697.60	5099.74	3941.73	466.38
3	1697.75	1698.82	4273.90	31936.32	2996.09	14.25	336.79	4806.99	3428.25	189.88
4	6171.50	200.51	1243.60	1246.77	37.49	10.59	166.20	84.28	13862.44	89.20
5	4048.50	312.63	14.57	477.72	3878.00	2914.93	450.90	5111.35	74.96	31938.70
6	14692.32	2598.78	2408.71	192.00	3152.96	1156.08	3292.10	3283.86	811.56	61.06
7	4775.75	25451.60								
8		39.00	2705.38	2704.85	234.54	42.19	14705.32	5107.94	3944.14	459.63
9		1663.32	1669.45	4272.10	31834.24	2964.36	14.25	341.50	4628.49	3434.32
10		188.81	6204.30	198.67	1244.69	1259.13	37.06	22.12	164.71	88.42
11		13768.16	88.31	4060.75	235.43	14.64	470.88	3876.30	2795.16	444.07
12		4256.60	74.06	31883.32	14722.24	2603.12	2395.76	190.45	3153.13	1143.15
13		3267.85	3267.05	800.76	60.48	4767.38	25404.12			
14	29.00	511.00	3006.00	3005.88	216.83	42.61	15100.96	5628.14	4363.66	473.00
15	1701.55	1717.82	4877.83	33880.08	3337.15	14.25	352.22	5254.55	3886.69	194.13
16	7019.56	205.83	1328.46	1334.47	37.54	9.49	166.25	82.31	15442.36	93.85
17	4472.68	305.57	14.58	477.78	4291.92	3242.38	457.65	5097.98	77.68	33882.76
18	15116.30	2914.61	2749.96	196.21	3558.15	1302.55	3727.45	3719.26	825.33	65.72
19	5097.72	27569.06								
20		512.00	3006.00	3006.25	216.58	42.61	15105.76	5629.78	4362.45	473.00
21		1698.35	1718.36	4877.25	33842.72	3335.98	14.25	352.17	4610.97	3884.70
22		194.13	7017.92	205.87	1339.24	1335.58	37.54	9.48	166.25	82.34
23		15432.04	93.87	4475.11	298.70	14.57	477.75	4292.13	3243.44	457.51
24		5097.29	77.63	33889.44	15113.43	2916.92	2753.11	196.27	3559.43	1303.47
25		3727.45	3721.96	826.09	65.66	5101.23	27554.37			
26	391.00	430.00	3126.25	3125.24	225.30	42.68	15534.00	5926.62	4579.34	468.84
27	1737.49	1759.13	5129.11	34840.96	3479.37	14.25	357.14	4979.63	4074.56	196.16
28	7348.88	207.95	1366.45	1398.64	37.46	9.00	166.10	81.64	16041.32	96.16
29	4716.83	281.72	14.59	476.59	4500.52	3380.92	452.79	5150.70	79.23	34777.40
30	15541.45	3028.92	2882.50	197.94	3722.15	1362.45	3905.62	3905.49	848.27	67.76
31	5223.10	28427.41								
32		159.00	3126.25	3128.76	230.37	42.44	15606.40	5936.46	4590.76	466.90
33		1705.15	1734.63	5136.99	34793.04	3461.06	14.25	323.39	5186.95	4087.20
34		195.32	7451.38	207.91	1427.97	1425.40	37.17	9.01	165.25	39.86
35		15989.56	95.69	4722.56	247.91	14.61	472.02	4508.37	3249.23	453.11
36		5181.84	78.24	34753.80	15614.52	3010.45	2864.17	197.79	3727.64	1366.20
37		3912.30	3916.11	866.82	68.40	5246.64	28650.26			
38	14.00	92.00	3276.50	3276.98	241.22	42.64	16115.24	6307.92	4851.70	462.05
39	1783.56	1781.25	5467.80	35809.68	3630.74	14.25	357.14	4961.08	4318.70	197.31
40	7849.58	210.27	1436.71	1477.25	37.14	10.60	165.14	72.84	16777.96	98.71
41	5042.45	248.71	14.61	471.75	4767.57	3411.80	446.83	5509.00	80.68	35874.88
42	16123.92	3169.91	3041.99	199.78	3931.83	1441.36	4124.80	4127.73	865.74	72.13
43	5386.08	29614.15								
44		96.00	3276.50	3275.77	240.92	42.63	16119.84	6306.26	4849.88	461.51
45		1757.99	1780.72	5463.54	35853.04	3627.22	14.25	355.79	4813.92	4316.28
46		197.31	7851.22	210.28	1456.89	1482.70	37.16	9.95	165.06	70.98
47		16772.72	98.66	5039.11	262.31	14.60	471.92	4765.11	3410.32	446.77
48		5506.41	80.53	35864.30	16129.07	3167.81	3040.91	199.79	3930.56	1441.36
49		4123.58	4129.00	869.53	71.96	5384.16	29620.55			
50		300.00	3126.25	3126.67	228.57	42.58	15578.80	5929.08	4581.71	469.63
51		1719.04	1746.18	5133.38	34762.88	3483.92	14.25	356.93	5152.58	4076.33
52		195.19	7387.42	207.67	1368.43	1417.69	37.32	8.97	165.77	82.05
53		16025.40	95.92	4720.63	326.25	14.60	472.99	4501.55	3250.46	454.96
54		5160.77	78.69	34767.58	15595.31	3029.62	2877.97	197.61	3725.44	1124.03
55		3215.38	3194.94	850.74	67.57	5224.45	28455.02			
56		518.00	3006.00	3005.35	216.10	42.60	15099.36	5625.68	4364.27	473.00
57		1700.95	1718.70	4874.82	33885.28	3336.37	14.25	352.17	4599.43	3881.05
58		194.13	7013.00	205.86	1318.38	1338.30	37.55	9.40	166.28	82.56
59		15440.72	93.80	4472.20	291.63	14.57	478.13	4289.98	3242.31	456.92
60		5091.38	77.66	33880.62	15105.64	2913.31	2751.97	196.27	3557.81	1298.40
61		3724.45	3719.72	828.25	65.62	5097.64	27566.78			
62		100.00								

```
\MCC_COOLING.NAME\="MCC_COOLING"
\MCC_COOLING.MEDIUM_OUTPUT\="F109"
\MCC_COOLING.MEDIUM_INPUT\="M102"
\MCC_COOLING.MEDIUM\="FUEL"
\MCC_COOLING.ASSOCIATE_PARAMETERS\="MCC_CLNT_DS_PR"
\MCC_COOLING.GENERIC_PARAMETERS\="pout"
\NOZZLE_COOLING.NAME\="NOZZLE_COOLING"
\NOZZLE_COOLING.MEDIUM_OUTPUT\="M201"
\NOZZLE_COOLING.MEDIUM_INPUT\="M102"
\NOZZLE_COOLING.MEDIUM\="FUEL"
\NOZZLE_COOLING.ASSOCIATE_PARAMETERS\="Notknown"
**********
```

```
\FUEL_FLOW_CONTROLLER.NAME\="FUEL_FLOW_CONTROLLER"
\FUEL_FLOW_CONTROLLER.CONTROLS\="FPOV"
\FUEL_FLOW_CONTROLLER.MEASURES_AT\="F101"
\FUEL_FLOW_CONTROLLER.ASSOCIATE_PARAMETERS\="FUEL_FLOW,FPOV_POSITION"
\FUEL_FLOW_CONTROLLER.GENERIC_PARAMETERS\="Vin,commanded_position"
**********
```

```
\HPFP_CLNT_LNR_PR.SOURCE\="SYSTEM"
\HPFP_CLNT_LNR_PR.SIGMA_NAME\="HPFTP_CLT_LNR_PR_104MF"
\HPFP_CLNT_LNR_PR.SENSOR_PID_NUMBER\="53"
\HPFP_CLNT_LNR_PR.NAME\="HPFP_CLNT_LNR_PR"
\HPFP_CLNT_LNR_PR.MEASUREMENT\="SINGLE"
\HPFP_CLNT_LNR_PR.GENERIC_NAME\="P_CLT"
\HPFP_CLNT_LNR_PR.ASSOCIATE_COMPONENT\="HPFP"
\MCC_FUEL_INJECTOR_PR.SOURCE\="SYSTEM"
\MCC_FUEL_INJECTOR_PR.SIGMA_NAME\="MCC_FL_INJ_PR_104MF"
\MCC_FUEL_INJECTOR_PR.SENSOR_PID_NUMBER\="24"
\MCC_FUEL_INJECTOR_PR.NAME\="MCC_FUEL_INJECTOR_PR"
\MCC_FUEL_INJECTOR_PR.MEASUREMENT\="SINGLE"
\MCC_FUEL_INJECTOR_PR.GENERIC_NAME\="P_IN,P_OUT"
\MCC_FUEL_INJECTOR_PR.ASSOCIATE_COMPONENT\="MCC,F111"
\LPOP_DS_PR.SOURCE\="SYSTEM"
\LPOP_DS_PR.SIGMA_NAME\="LPOP_DS_PR_104MF"
\LPOP_DS_PR.SENSOR_PID_NUMBER\="209"
\LPOP_DS_PR.NAME\="LPOP_DS_PR"
\LPOP_DS_PR.MEASUREMENT\="SINGLE"
\LPOP_DS_PR.GENERIC_NAME\="P_OUT,P_IN"
\LPOP_DS_PR.ASSOCIATE_COMPONENT\="0201,HPOP"
\HPOP_DS_PR.SOURCE\="SYSTEM"
\HPOP_DS_PR.SIGMA_NAME\="HPOP_DS_PR_104MF"
\HPOP_DS_PR.SENSOR_PID_NUMBER\="90"
\HPOP_DS_PR.NAME\="HPOP_DS_PR"
\HPOP_DS_PR.MEASUREMENT\="SINGLE"
\HPOP_DS_PR.GENERIC_NAME\="P_OUT,P_IN"
\HPOP_DS_PR.ASSOCIATE_COMPONENT\="HPOP,0204"
\LOX_DOME_TMP.SOURCE\="SYSTEM"
\LOX_DOME_TMP.SIGMA_NAME\="OX_DOME_T_104MF"
\LOX_DOME_TMP.SENSOR_PID_NUMBER\="21"
\LOX_DOME_TMP.NAME\="LOX_DOME_TMP"
\LOX_DOME_TMP.MEASUREMENT\="SINGLE"
\LOX_DOME_TMP.GENERIC_NAME\="MCC"
\LOX_DOME_TMP.ASSOCIATE_COMPONENT\="MCC"
\PBP_DS_PR.SOURCE\="SYSTEM"
\PBP_DS_PR.SIGMA_NAME\="PBP_DS_PR_104MF"
\PBP_DS_PR.SENSOR_PID_NUMBER\="59"
\PBP_DS_PR.NAME\="PBP_DS_PR"
\PBP_DS_PR.MEASUREMENT\="SINGLE"
\PBP_DS_PR.GENERIC_NAME\="P_IN"
\PBP_DS_PR.ASSOCIATE_COMPONENT\="0205"
\PBP_DS_TMP.SOURCE\="SYSTEM"
\PBP_DS_TMP.SIGMA_NAME\="PBP_DS_T_104MF"
\PBP_DS_TMP.SENSOR_PID_NUMBER\="94"
\PBP_DS_TMP.NAME\="PBP_DS_TMP"
\PBP_DS_TMP.MEASUREMENT\="SINGLE"
\PBP_DS_TMP.GENERIC_NAME\="T_IN"
\PBP_DS_TMP.ASSOCIATE_COMPONENT\="0205"
\ENG_FUEL_INLET_TMP.SOURCE\="SYSTEM"
\ENG_FUEL_INLET_TMP.SIGMA_NAME\="ENG_FL_IN_T_104MF"
\ENG_FUEL_INLET_TMP.SENSOR_PID_NUMBER\="1021"
\ENG_FUEL_INLET_TMP.NAME\="ENG_FUEL_INLET_TMP"
\ENG_FUEL_INLET_TMP.MEASUREMENT\="SINGLE"
\ENG_FUEL_INLET_TMP.GENERIC_NAME\="T_IN"
\ENG_FUEL_INLET_TMP.ASSOCIATE_COMPONENT\="LPFP"
```

```
\ENG_FUEL_INLET_PR.SOURCE\="SYSTEM"
\ENG_FUEL_INLET_PR.SIGMA_NAME\="ENG_FL_IN_PR_104MF"
\ENG_FUEL_INLET_PR.SENSOR_PID_NUMBER\="821"
\ENG_FUEL_INLET_PR.NAME\="ENG_FUEL_INLET_PR"
\ENG_FUEL_INLET_PR.MEASUREMENT\="SINGLE"
\ENG_FUEL_INLET_PR.GENERIC_NAME\="P_IN"
\ENG_FUEL_INLET_PR.ASSOCIATE_COMPONENT\="LPFP"
\ENG_OX_INLET_TMP.SOURCE\="SYSTEM"
\ENG_OX_INLET_TMP.SIGMA_NAME\="ENG_OX_IN_T_104MF"
\ENG_OX_INLET_TMP.SENSOR_PID_NUMBER\="1058"
\ENG_OX_INLET_TMP.NAME\="ENG_OX_INLET_TMP"
\ENG_OX_INLET_TMP.MEASUREMENT\="SINGLE"
\ENG_OX_INLET_TMP.GENERIC_NAME\="T_IN"
\ENG_OX_INLET_TMP.ASSOCIATE_COMPONENT\="LPOP"
\ENG_OX_INLET_PR.SOURCE\="SYSTEM"
\ENG_OX_INLET_PR.SIGMA_NAME\="ENG_OX_IN_PR_104MF"
\ENG_OX_INLET_PR.SENSOR_PID_NUMBER\="858"
\ENG_OX_INLET_PR.NAME\="ENG_OX_INLET_PR"
\ENG_OX_INLET_PR.MEASUREMENT\="SINGLE"
\ENG_OX_INLET_PR.GENERIC_NAME\="P_IN"
\ENG_OX_INLET_PR.ASSOCIATE_COMPONENT\="LPOP"
\FUEL_FLOW.SOURCE\="SYSTEM"
\FUEL_FLOW.SIGMA_NAME\="FL_FLOW_104MF"
\FUEL_FLOW.SENSOR_PID_NUMBER\="100"
\FUEL_FLOW.NAME\="FUEL_FLOW"
\FUEL_FLOW.MEASUREMENT\="SINGLE"
\FUEL_FLOW.GENERIC_NAME\="V_OUT,V_IN"
\FUEL_FLOW.ASSOCIATE_COMPONENT\="F101,HPFP"
\HPFP_DS_TMP.SOURCE\="SYSTEM"
\HPFP_DS_TMP.SIGMA_NAME\="HPFP_DS_T_104MF"
\HPFP_DS_TMP.SENSOR_PID_NUMBER\="659"
\HPFP_DS_TMP.NAME\="HPFP_DS_TMP"
\HPFP_DS_TMP.MEASUREMENT\="SINGLE"
\HPFP_DS_TMP.GENERIC_NAME\="T_IN,T_OUT"
\HPFP_DS_TMP.ASSOCIATE_COMPONENT\="F102,HPFP"
\HPFP_BAL_CAV_PR.SOURCE\="SYSTEM"
\HPFP_BAL_CAV_PR.SIGMA_NAME\="HPFP_BAL_CAV_PR_104MF"
\HPFP_BAL_CAV_PR.SENSOR_PID_NUMBER\="457"
\HPFP_BAL_CAV_PR.NAME\="HPFP_BAL_CAV_PR"
\HPFP_BAL_CAV_PR.MEASUREMENT\="SINGLE"
\HPFP_BAL_CAV_PR.GENERIC_NAME\="P_BAL"
\HPFP_BAL_CAV_PR.ASSOCIATE_COMPONENT\="HPFP"
\HPFP_COOLANT_LINER_TMP.SOURCE\="SYSTEM"
\HPFP_COOLANT_LINER_TMP.SIGMA_NAME\="HPFP_CL_T_104MF"
\HPFP_COOLANT_LINER_TMP.SENSOR_PID_NUMBER\="650"
\HPFP_COOLANT_LINER_TMP.NAME\="HPFP_COOLANT_LINER_TMP"
\HPFP_COOLANT_LINER_TMP.MEASUREMENT\="SINGLE"
\HPFP_COOLANT_LINER_TMP.GENERIC_NAME\="T_CLT"
\HPFP_COOLANT_LINER_TMP.ASSOCIATE_COMPONENT\="HPFP"
\HPFP_DRAIN_PR.SOURCE\="SYSTEM"
\HPFP_DRAIN_PR.SIGMA_NAME\="HPFP_DRN_PR_104MF"
\HPFP_DRAIN_PR.SENSOR_PID_NUMBER\="657"
\HPFP_DRAIN_PR.NAME\="HPFP_DRAIN_PR"
\HPFP_DRAIN_PR.MEASUREMENT\="SINGLE"
\HPFP_DRAIN_PR.GENERIC_NAME\="P_DRAIN"
\HPFP_DRAIN_PR.ASSOCIATE_COMPONENT\="HPFP"
```

\HPFP_DRAIN_TMP.SOURCE\="SYSTEM"
\HPFP_DRAIN_TMP.SIGMA_NAME\="HPFP_DRN_T_104MF"
\HPFP_DRAIN_TMP.SENSOR_PID_NUMBER\="658"
\HPFP_DRAIN_TMP.NAME\="HPFP_DRAIN_TMP"
\HPFP_DRAIN_TMP.MEASUREMENT\="SINGLE"
\HPFP_DRAIN_TMP.GENERIC_NAME\="T_DRAIN"
\HPFP_DRAIN_TMP.ASSOCIATE_COMPONENT\="HPFP"
\LPFT_INLET_PR.SOURCE\="SYSTEM"
\LPFT_INLET_PR.SIGMA_NAME\="LPFT_IN_PR_104MF"
\LPFT_INLET_PR.SENSOR_PID_NUMBER\="436"
\LPFT_INLET_PR.NAME\="LPFT_INLET_PR"
\LPFT_INLET_PR.MEASUREMENT\="SINGLE"
\LPFT_INLET_PR.GENERIC_NAME\="P_OUT,P_IN"
\LPFT_INLET_PR.ASSOCIATE_COMPONENT\="F109,LPFT"
\FUEL_PRESSURANT_INTERFACE_PR.SIGMA_NAME\="FL_PR_INT_PR_104MF"
\FUEL_PRESSURANT_INTERFACE_PR.SENSOR_PID_NUMBER\="835"
\FUEL_PRESSURANT_INTERFACE_PR.NAME\="FUEL PRESSURANT INTERFACE PR"
\FUEL_PRESSURANT_INTERFACE_PR.MEASUREMENT\="SINGLE"
\FUEL_PRESSURANT_INTERFACE_PR.GENERIC_NAME\="NotKnown"
\FUEL_PRESSURANT_INTERFACE_TMP.SIGMA_NAME\="FL_PR_INT_T_104MF"
\FUEL_PRESSURANT_INTERFACE_TMP.SENSOR_PID_NUMBER\="1035"
\FUEL_PRESSURANT_INTERFACE_TMP.NAME\="FUEL PRESSURANT INTERFACE TMP"
\FUEL_PRESSURANT_INTERFACE_TMP.MEASUREMENT\="SINGLE"
\FUEL_PRESSURANT_INTERFACE_TMP.GENERIC_NAME\="NotKnown"
\OPB_PC.SOURCE\="SYSTEM"
\OPB_PC.SIGMA_NAME\="OPB_PC_104MF"
\OPB_PC.SENSOR_PID_NUMBER\="480"
\OPB_PC.NAME\="OPB_PC"
\OPB_PC.MEASUREMENT\="SINGLE"
\OPB_PC.GENERIC_NAME\="P_IN,P_OUT"
\OPB_PC.ASSOCIATE_COMPONENT\="HPOT,OPB"
\FPOV_POSITION.SOURCE\="SYSTEM"
\FPOV_POSITION.SIGMA_NAME\="FPOV_ACT_POS_104MF"
\FPOV_POSITION.SENSOR_PID_NUMBER\="142"
\FPOV_POSITION.NAME\="FPOV_POSITION"
\FPOV_POSITION.MEASUREMENT\="SINGLE"
\FPOV_POSITION.GENERIC_NAME\="POS"
\FPOV_POSITION.ASSOCIATE_COMPONENT\="FPOV"
\MCC_LOX_INJECTOR_TMP.SOURCE\="SYSTEM"
\MCC_LOX_INJECTOR_TMP.SIGMA_NAME\="MCC_OX_INJ_T_104MF"
\MCC_LOX_INJECTOR_TMP.SENSOR_PID_NUMBER\="595"
\MCC_LOX_INJECTOR_TMP.NAME\="MCC_LOX_INJECTOR_TMP"
\MCC_LOX_INJECTOR_TMP.MEASUREMENT\="SINGLE"
\MCC_LOX_INJECTOR_TMP.GENERIC_NAME\="T_IN_OX,T_OUT"
\MCC_LOX_INJECTOR_TMP.ASSOCIATE_COMPONENT\="MCC,MOV"
\MCC_LOX_INJECTOR_PR.SOURCE\="SYSTEM"
\MCC_LOX_INJECTOR_PR.SIGMA_NAME\="MCC_OX_INJ_PR_104MF"
\MCC_LOX_INJECTOR_PR.SENSOR_PID_NUMBER\="395"
\MCC_LOX_INJECTOR_PR.NAME\="MCC_LOX_INJECTOR_PR"
\MCC_LOX_INJECTOR_PR.MEASUREMENT\="SINGLE"
\MCC_LOX_INJECTOR_PR.GENERIC_NAME\="P_IN_OX,P_OUT"
\MCC_LOX_INJECTOR_PR.ASSOCIATE_COMPONENT\="MCC,MOV"
\POGO_PRE_CHARGE_PR.SOURCE\="SYSTEM"
\POGO_PRE_CHARGE_PR.SIGMA_NAME\="POGO_PRCHG_PR_104MF"
\POGO_PRE_CHARGE_PR.SENSOR_PID_NUMBER\="221"
\POGO_PRE_CHARGE_PR.NAME\="POGO_PRE_CAHRGE_PR"

```
\POGO_PRE_CHARGE_PR.MEASUREMENT\="SINGLE"
\POGO_PRE_CHARGE_PR.GENERIC_NAME\="P_PRE_IN"
\POGO_PRE_CHARGE_PR.ASSOCIATE_COMPONENT\="POGO"
\HEAT_EXCHANGER_DS_PR.SIGMA_NAME\="HX_DS_PR_104MF"
\HEAT_EXCHANGER_DS_PR.SENSOR_PID_NUMBER\="34"
\HEAT_EXCHANGER_DS_PR.NAME\="HEAT EXCHANGER DS PR"
\HEAT_EXCHANGER_DS_PR.MEASUREMENT\="SINGLE"
\HEAT_EXCHANGER_DS_PR.GENERIC_NAME\="NotKnown"
\HEAT_EXCHANGER_INTERFACE_PR.SIGMA_NAME\="HX_INT_PR_104MF"
\HEAT_EXCHANGER_INTERFACE_PR.SENSOR_PID_NUMBER\="878"
\HEAT_EXCHANGER_INTERFACE_PR.NAME\="HEAT EXCHANGER INTERFACE PR"
\HEAT_EXCHANGER_INTERFACE_PR.MEASUREMENT\="SINGLE"
\HEAT_EXCHANGER_INTERFACE_PR.GENERIC_NAME\="NotKnown"
\HEAT_EXCHANGER_INTERFACE_TMP.SIGMA_NAME\="HX_INT_T_104MF"
\HEAT_EXCHANGER_INTERFACE_TMP.SENSOR_PID_NUMBER\="879"
\HEAT_EXCHANGER_INTERFACE_TMP.NAME\="HEAT EXCHANGER INTERFACE TMP"
\HEAT_EXCHANGER_INTERFACE_TMP.MEASUREMENT\="SINGLE"
\HEAT_EXCHANGER_INTERFACE_TMP.GENERIC_NAME\="NotKnown"
\OPOV_POSITION.SOURCE\="SYSTEM"
\OPOV_POSITION.SIGMA_NAME\="OPOV_ACT_POS_104MF"
\OPOV_POSITION.SENSOR_PID_NUMBER\="140"
\OPOV_POSITION.NAME\="OPOV_POSITION"
\OPOV_POSITION.MEASUREMENT\="SINGLE"
\OPOV_POSITION.GENERIC_NAME\="POS"
\OPOV_POSITION.ASSOCIATE_COMPONENT\="OPOV"
\HPOP_SPEED.SOURCE\="SYSTEM"
\HPOP_SPEED.SIGMA_NAME\="HPOP_SPD_104MF"
\HPOP_SPEED.SENSOR_PID_NUMBER\="2"
\HPOP_SPEED.NAME\="HPOP_SPEED"
\HPOP_SPEED.MEASUREMENT\="SINGLE"
\HPOP_SPEED.GENERIC_NAME\="SPEED,SPEED,SPEED"
\HPOP_SPEED.ASSOCIATE_COMPONENT\="HPOP,HPOS,HPOT"
\LPOP_SPEED2.SOURCE\="SYSTEM"
\LPOP_SPEED2.SIGMA_NAME\="LPOP_SPD_104MF"
\LPOP_SPEED2.SENSOR_PID_NUMBER\="734"
\LPOP_SPEED2.RELATED_TO\="LPOP_SPEED1"
\LPOP_SPEED2.NAME\="LPOP_SPEED2"
\LPOP_SPEED2.MEASUREMENT\="CONSTITUTION"
\LPOP_SPEED2.GENERIC_NAME\="SPEED,SPEED,SPEED"
\LPOP_SPEED2.ASSOCIATE_COMPONENT\="LPOP,LPOS,LPOT"
\HPOP_BAL_CAV_PR_B.SOURCE\="SYSTEM"
\HPOP_BAL_CAV_PR_B.SIGMA_NAME\="HPOP_BCAV_PR_B_104MF"
\HPOP_BAL_CAV_PR_B.SENSOR_PID_NUMBER\="328"
\HPOP_BAL_CAV_PR_B.RELATED_TO\="HPOP_BAL_CAV_PR1"
\HPOP_BAL_CAV_PR_B.NAME\="HPOP_BAL_CAV_PR_B"
\HPOP_BAL_CAV_PR_B.MEASUREMENT\="CONSTITUTION"
\HPOP_BAL_CAV_PR_B.GENERIC_NAME\="P_BAL"
\HPOP_BAL_CAV_PR_B.ASSOCIATE_COMPONENT\="HPOP"
\HPOP_BAL_CAV_PR_A.SOURCE\="SYSTEM"
\HPOP_BAL_CAV_PR_A.SIGMA_NAME\="HPOP_BCAV_PR_A_104MF"
\HPOP_BAL_CAV_PR_A.SENSOR_PID_NUMBER\="327"
\HPOP_BAL_CAV_PR_A.RELATED_TO\="HPOP_BAL_CAV_PR1"
\HPOP_BAL_CAV_PR_A.NAME\="HPOP_BAL_CAV_PR_A"
\HPOP_BAL_CAV_PR_A.MEASUREMENT\="CONSTITUTION"
\HPOP_BAL_CAV_PR_A.GENERIC_NAME\="P_BAL"
\HPOP_BAL_CAV_PR_A.ASSOCIATE_COMPONENT\="HPOP"
```

```
\LPFP_SPEED2.SOURCE\="SYSTEM"
\LPFP_SPEED2.SIGMA_NAME\="LPFP_SPD_104MF"
\LPFP_SPEED2.SENSOR_PID_NUMBER\="754"
\LPFP_SPEED2.RELATED_TO\="LPFP_SPEED1"
\LPFP_SPEED2.NAME\="LPFP_SPEED2"
\LPFP_SPEED2.MEASUREMENT\="CONSTITUTION"
\LPFP_SPEED2.GENERIC_NAME\="SPEED,SPEED,SPEED"
\LPFP_SPEED2.ASSOCIATE_COMPONENT\="LPFP,LPFS,LPFT"
\LPFP_SPEED.SOURCE\="SYSTEM"
\LPFP_SPEED.SIGMA_NAME\="LPFTP_SPD_104MF"
\LPFP_SPEED.SENSOR_PID_NUMBER\="32"
\LPFP_SPEED.RELATED_TO\="LPFP_SPEED1"
\LPFP_SPEED.NAME\="LPFP_SPEED"
\LPFP_SPEED.MEASUREMENT\="CONSTITUTION"
\LPFP_SPEED.GENERIC_NAME\="SPEED,SPEED,SPEED"
\LPFP_SPEED.ASSOCIATE_COMPONENT\="LPFP,LPFS,LPFT"
\HPFP_SPEED2.SOURCE\="SYSTEM"
\HPFP_SPEED2.SIGMA_NAME\="HPFP_SPD_104MF"
\HPFP_SPEED2.SENSOR_PID_NUMBER\="764"
\HPFP_SPEED2.RELATED_TO\="HPFP_SPEED1"
\HPFP_SPEED2.NAME\="HPFP_SPEED2"
\HPFP_SPEED2.MEASUREMENT\="CONSTITUTION"
\HPFP_SPEED2.GENERIC_NAME\="SPEED,SPEED,SPEED"
\HPFP_SPEED2.ASSOCIATE_COMPONENT\="HPFT,HPFS,HPFP"
\HPOT_DS_TMP_B.SOURCE\="SYSTEM"
\HPOT_DS_TMP_B.SIGMA_NAME\="HPOT_T_B_104MF"
\HPOT_DS_TMP_B.SENSOR_PID_NUMBER\="234"
\HPOT_DS_TMP_B.RELATED_TO\="HPOT_DS_TMP1"
\HPOT_DS_TMP_B.NAME\="HPOT_DS_TMP_B"
\HPOT_DS_TMP_B.MEASUREMENT\="CONSTITUTION"
\HPOT_DS_TMP_B.GENERIC_NAME\="T_OUT,T_IN"
\HPOT_DS_TMP_B.ASSOCIATE_COMPONENT\="HPOT,0207"
\HPOT_DS_TMP_A.SOURCE\="SYSTEM"
\HPOT_DS_TMP_A.SIGMA_NAME\="HPOT_T_A_104MF"
\HPOT_DS_TMP_A.SENSOR_PID_NUMBER\="233"
\HPOT_DS_TMP_A.RELATED_TO\="HPOT_DS_TMP1"
\HPOT_DS_TMP_A.NAME\="HPOT_DS_TMP_A"
\HPOT_DS_TMP_A.MEASUREMENT\="CONSTITUTION"
\HPOT_DS_TMP_A.GENERIC_NAME\="T_OUT,T_IN"
\HPOT_DS_TMP_A.ASSOCIATE_COMPONENT\="HPOT,0207"
\HPFT_DS_TMP_A.SOURCE\="SYSTEM"
\HPFT_DS_TMP_A.SIGMA_NAME\="HPFT_T_A_104MF"
\HPFT_DS_TMP_A.SENSOR_PID_NUMBER\="231"
\HPFT_DS_TMP_A.RELATED_TO\="HPFT_DS_TMP1"
\HPFT_DS_TMP_A.NAME\="HPFT_DS_TMP_A"
\HPFT_DS_TMP_A.MEASUREMENT\="CONSTITUTION"
\HPFT_DS_TMP_A.GENERIC_NAME\="T_OUT,T_IN"
\HPFT_DS_TMP_A.ASSOCIATE_COMPONENT\="HPFT,F111"
\HPFT_DS_TMP_B.SOURCE\="SYSTEM"
\HPFT_DS_TMP_B.SIGMA_NAME\="HPFT_T_B_104MF"
\HPFT_DS_TMP_B.SENSOR_PID_NUMBER\="232"
\HPFT_DS_TMP_B.RELATED_TO\="HPFT_DS_TMP1"
\HPFT_DS_TMP_B.NAME\="HPFT_DS_TMP_B"
\HPFT_DS_TMP_B.MEASUREMENT\="CONSTITUTION"
\HPFT_DS_TMP_B.GENERIC_NAME\="T_OUT,T_IN"
\HPFT_DS_TMP_B.ASSOCIATE_COMPONENT\="HPFT,F111"
```

```
\LPOP_SPEED.SOURCE\="SYSTEM"
\LPOP_SPEED.SIGMA_NAME\="LPOTP_SPD_104MF"
\LPOP_SPEED.SENSOR_PID_NUMBER\="30"
\LPOP_SPEED.RELATED_TO\="LPOP_SPEED1"
\LPOP_SPEED.NAME\="LPOP_SPEED"
\LPOP_SPEED.MEASUREMENT\="CONSTITUTION"
\LPOP_SPEED.GENERIC_NAME\="SPEED,SPEED,SPEED"
\LPOP_SPEED.ASSOCIATE_COMPONENT\="LPOP,LPOS,LPOT"
\HPFP_SPEED.SOURCE\="SYSTEM"
\HPFP_SPEED.SIGMA_NAME\="HPFTP_SPD_104MF"
\HPFP_SPEED.SENSOR_PID_NUMBER\="260"
\HPFP_SPEED.RELATED_TO\="HPFP_SPEED1"
\HPFP_SPEED.NAME\="HPFP_SPEED"
\HPFP_SPEED.MEASUREMENT\="CONSTITUTION"
\HPFP_SPEED.GENERIC_NAME\="SPEED,SPEED,SPEED"
\HPFP_SPEED.ASSOCIATE_COMPONENT\="HPFT,HPFS,HPFP"
*********
```

```
\FPB.NAME\="FPB"
\FPB.GAS_OUT\="HPFT"
\FPB.FUEL_IN\="F110"
\FPB.OX_IN\="FPOV"
\FPB.ASSOCIATE_PARAMETERS\="FPB_PC"
\FPB.GENERIC_PARAMETERS\="pout"
\OPB.NAME\="OPB"
\OPB.GAS_OUT\="HPOT"
\OPB.FUEL_IN\="F108"
\OPB.OX_IN\="OPOV"
\OPB.ASSOCIATE_PARAMETERS\="Notknown"
**********
```

```
\F101.NAME\="F101"
\F101.MEDIUM_OUTPUT\="HPFP"
\F101.MEDIUM_INPUT\="LPFP"
\F101.MEDIUM\="FUEL"
\F101.ASSOCIATE_PARAMETERS\="FUEL_FLOW,LPFP_DS_TMP,LPFP_DS_PR"
\F101.GENERIC_PARAMETERS\="Vout,Tin,pin"
\F102.NAME\="F102"
\F102.MEDIUM_OUTPUT\="MFV"
\F102.MEDIUM_INPUT\="HPFP"
\F102.MEDIUM\="FUEL"
\F102.ASSOCIATE_PARAMETERS\="HPFP_DS_PR,HPFP_DS_TMP"
\F102.GENERIC_PARAMETERS\="pin,Tin"
\F107.NAME\="F107"
\F107.MEDIUM_OUTPUT\="M103"
\F107.MEDIUM_INPUT\="M201"
\F107.MEDIUM\="FUEL"
\F107.ASSOCIATE_PARAMETERS\="NotKnown"
\F108.NAME\="F108"
\F108.MEDIUM_OUTPUT\="OPB"
\F108.MEDIUM_INPUT\="M103"
\F108.MEDIUM\="FUEL"
\F108.ASSOCIATE_PARAMETERS\="NotKnown"
\F109.NAME\="F109"
\F109.MEDIUM_OUTPUT\="LPFT"
\F109.MEDIUM_INPUT\="MCC_COOLING"
\F109.MEDIUM\="FUEL"
\F109.ASSOCIATE_PARAMETERS\="MCC_CLNT_DS_PR,MCC_CLNT_DS_TMP,LPFT_INLET_PR"
\F109.GENERIC_PARAMETERS\="pin,Tin,pout"
\F110.NAME\="F110"
\F110.MEDIUM_OUTPUT\="FPB"
\F110.MEDIUM_INPUT\="M103"
\F110.MEDIUM\="FUEL"
\F110.ASSOCIATE_PARAMETERS\="NotKnown"
\F111.NAME\="F111"
\F111.MEDIUM_OUTPUT\="MCC"
\F111.MEDIUM_INPUT\="HPFT"
\F111.MEDIUM\="FUEL"
\F111.ASSOCIATE_PARAMETERS\="HPFT_DS_TMP1"
\F111.GENERIC_PARAMETERS\="Tin"
\0206.NAME\="0206"
\0206.MEDIUM_OUTPUT\="FPOV"
\0206.MEDIUM_INPUT\="M105"
\0206.MEDIUM\="OX"
\0206.ASSOCIATE_PARAMETERS\="PBP_DS_TMP,PBP_DS_PR"
\0206.GENERIC_PARAMETERS\="pin,pout"
**********
```

```
\LPFP.NAME\="LPFP"
\LPFP.MEDIUM_OUTPUT\="F101"
\LPFP.MEDIUM_INPUT\="FUEL_TANK"
\LPFP.MEDIUM\="FUEL"
\LPFP.DIRECTION\="IN"
\LPFP.COUPLED_TO\="LPFT"
\LPFP.ASSOCIATE_PARAMETERS\="ENG_FUEL_INLET_PR,ENG_FUEL_INLET_TMP,LPFP_SPEED1,LPFP_DS_PR,LPFP_DS_TMP"
\LPFP.GENERIC_PARAMETERS\="pin,Tin,omega,pout,Tout"
\HPFP.NAME\="HPFP"
\HPFP.MEDIUM_OUTPUT\="F102"
\HPFP.MEDIUM_INPUT\="F101"
\HPFP.MEDIUM\="FUEL"
\HPFP.DIRECTION\="IN"
\HPFP.COUPLED_TO\="HPFT"
\HPFP.ASSOCIATE_PARAMETERS\="HPFP_DS_PR,FUEL_FLOW,HPFP_SPEED1,HPFP_DS_TMP"
\HPFP.GENERIC_PARAMETERS\="pout,Vin,omega,Tout"
**********
```

```
\FUEL_TANK.NAME\="FUEL_TANK"
\FUEL_TANK.MEDIUM_OUTPUT\="LPFP"
\FUEL_TANK.MEDIUM\="FUEL"
\FUEL_TANK.ASSOCIATE_PARAMETERS\="HPFP_SPEED1,FPB_PC,HPFT_DS_TMP1"
\FUEL_TANK.GENERIC_PARAMETERS\="omega,pin,Tout"
\FUEL_TANK.ASSOCIATE_PARAMETERS\="ENG_FUEL_INLET_PR,ENG_FUEL_INLET_TMP"
\FUEL_TANK.GENERIC_PARAMETERS\="pout,Tout"
**********
```

```
\FUEL_TANK.NAME\="FUEL_TANK"
\F111.NAME\="F111"
\0206.NAME\="0206"
\OPB.NAME\="OPB"
\M101.NAME\="M101"
**********
```

```
\M102.NAME\="M102"
\M102.MEDIUM_OUTA\="MCC_COOLING"
\M102.MEDIUM_OUTB\="NOZZLE_COOLING"
\M102.MEDIUM_OUTC\="CCV"
\M102.MEDIUM_IN\="MFV"
\M102.MEDIUM\="FUEL"
\M102.ASSOCIATE_PARAMETERS\="Notknown"
**********
```

\M201.NAME\="M201"
\M201.MEDIUM_OUT\="F107"
\M201.MEDIUM_INA\="NOZZLE_COOLING"
\M201.MEDIUM_INB\="CCV"
\M201.MEDIUM\="FUEL"
\M201.ASSOCIATE_PARAMETERS\="Notknown"
****

```
\M101.NAME\="M101"
\M101.MEDIUM_OUTA\="HPOT_COOLING"
\M101.MEDIUM_OUTB\="HPFT_COOLING"
\M101.MEDIUM_IN\="LPFT"
\M101.MEDIUM\="FUEL"
\M101.ASSOCIATE_PARAMETERS\="Notknown"
\M103.NAME\="M103"
\M103.MEDIUM_OUTA\="F108"
\M103.MEDIUM_OUTB\="F110"
\M103.MEDIUM_IN\="F107"
\M103.MEDIUM\="FUEL"
\M103.ASSOCIATE_PARAMETERS\="Notknown"
**********
```