Rockwell International
Rocketdyne Division

System for Anomaly and Failure Detection (SAFD) System Development

Final Report

24 July 1992

Contract NAS8-40000

Task 23

Prepared by

D. O'Reilly
Project Engineer

Approved by

C. K. Kraus
Project Manager

S. L. Stohler
Manager
SSME Technology Support Team

Rocketdyne Division
Rockwell International Corporation
6633 Canoga Avenue
Canoga Park, California 91303

Telex 698478
ROCKETDYN CNPK

10 August 1992

In Reply refer to 92RC05156

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

Attention: J. L. Moses, ER21

Reference: Transmittal of Final Technical Report, SSME Technology Test Bed System for Anomaly and Failure Detection (SAFD), Phase IV (RSS-8825-37)

Gentlemen:

Attached is the Final Technical Report for the subject task being performed as part of the SSME Technology Test Bed effort (Contract NAS8-40000).

If there are any questions concerning this report, please contact S. L. Stohler at (818) 710-3078 or D. W. O'Reilly at (205) 544-6974.

Very truly yours,

ROCKWELL INTERNATIONAL CORPORATION
Rocketdyne Division

S. L. Stohler
Manager
SSME Technology Support Team

Encl: (1) One copy Final Technical Report, RSS-8825-37.

xc: NASA/George C. Marshall Space Flight Center (1 copy each) Marshall Space Flight Center, AL 35812

Attention: L. Ingram, EB32
P. Vallely, ED14
V. Yost, EE25
G. Young, EP62
S. Douglas, ED14
H. McConnaughey, EP01
Respository, CN22D
C. Cozelos, EB42
G. Vick, EB44
T. Fox, ED14
S. Richards, HA01
R. Panciera, EB32

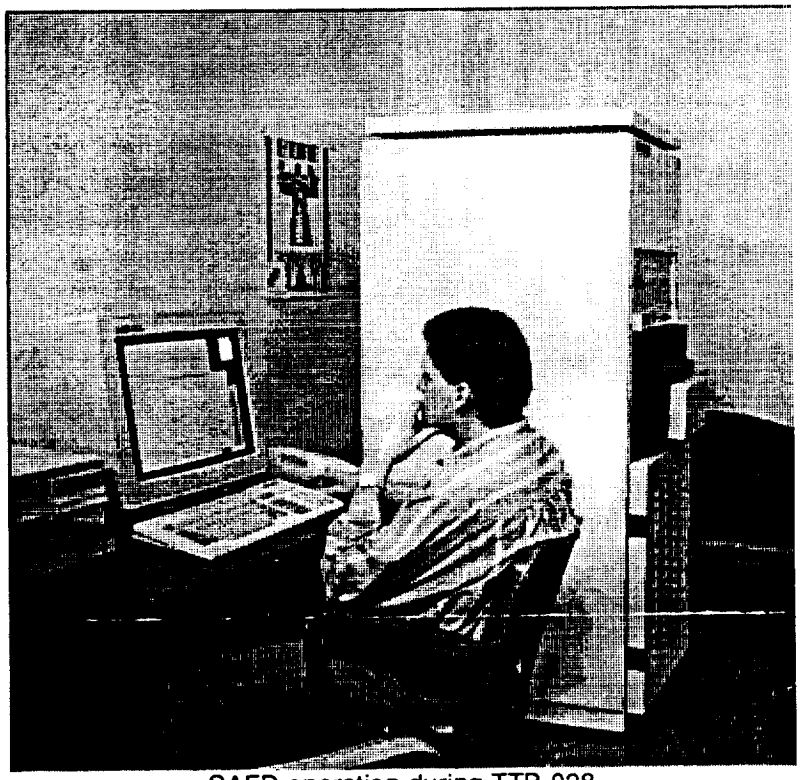
Table of Contents

List of Tables

List of Figures

Executive Summary

This task specified developing the hardware and software necessary to implement the System for Anomaly and Failure Detection (SAFD) algorithm, developed under Technology Test Bed (TTB) Task 21, on the TTB engine stand. This effort involved building two units; one unit to be installed in the Block II Space Shuttle Main Engine (SSME) Hardware Simulation Lab (HSL) at Marshall Space Flight Center (MSFC), and one unit to be installed at the TTB engine stand. Rocketdyne personnel from the HSL performed the task.



SAFD operation during TTB-028

The SAFD algorithm was developed as an improvement over the current redline system used in the Space Shuttle Main Engine Controller (SSMEC). Simulation tests and execution against previous hot fire tests demonstrated that the SAFD algorithm can detect engine failures as much as tens of seconds before the redline system recognized the failure. Although the current algorithm only operates during steady state conditions (engine not throttling), work is underway to expand the algorithm to work during transient conditions.

The algorithm currently includes 22 parameters; 16 from the Vehicle Data Table (VDT) and 6 from facility. However, only 5 of the facility parameters are available as the facility is using the available amplifiers for one of the parameters. The algorithm uses a statistical approach for generating limits for

the parameters based on mean and standard deviation. It calculates a running average of the last five samples for each parameter and compares this running average to the limits. If three (adaptation data) parameters are out of limits at the same time, SAFD requests a cut. Figure 1 illustrates establishing the limits for a typical parameter.

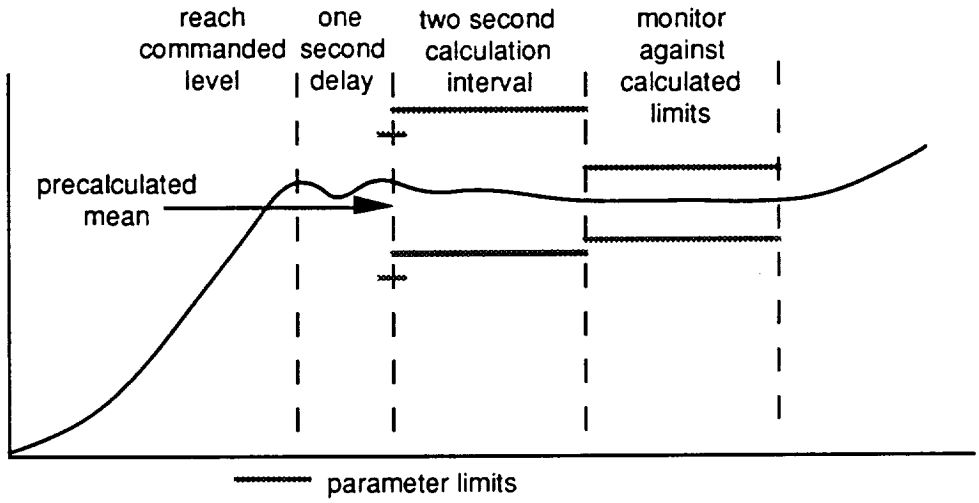


Figure 1 - SAFD Algorithm Limits

Rocketdyne completed the two systems and installed one in the HSL and one at the TTB. Figure 2 below illustrates the configuration at the TTB. SAFD acquires the VDT from a spare output on the Command And Data Simulator (CADS). This output provides the VDT after CADS has decoded it and converted it to parallel digital data. The facility provides non-VDT measurements via the SIU which converts the sensor raw analog signal to a DC voltage and amplifies it for SAFD use. The facility provides Greenwich Mean Time to a time code generator in the SAFD which decodes the time signal and generates the time stamp. SAFD requests cutoff by setting a discrete relay which is connected to the facility cutoff system.

The SAFD systems are based on Concurrent 6450 general purpose computer systems running Real Time Unix (RTU) operating system. The selection of this system was based on trade studies comparing capability and cost. Peripheral devices were purchased off-the-shelf where possible and custom built by Rocketdyne when not commercially available. The block diagram in Figure 3 illustrates the hardware configuration.

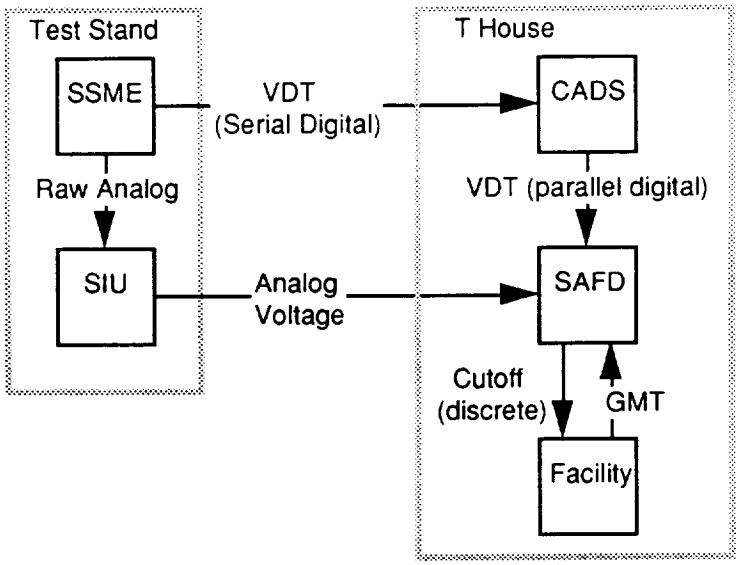


Figure 2 - SAFD TTB Configuration

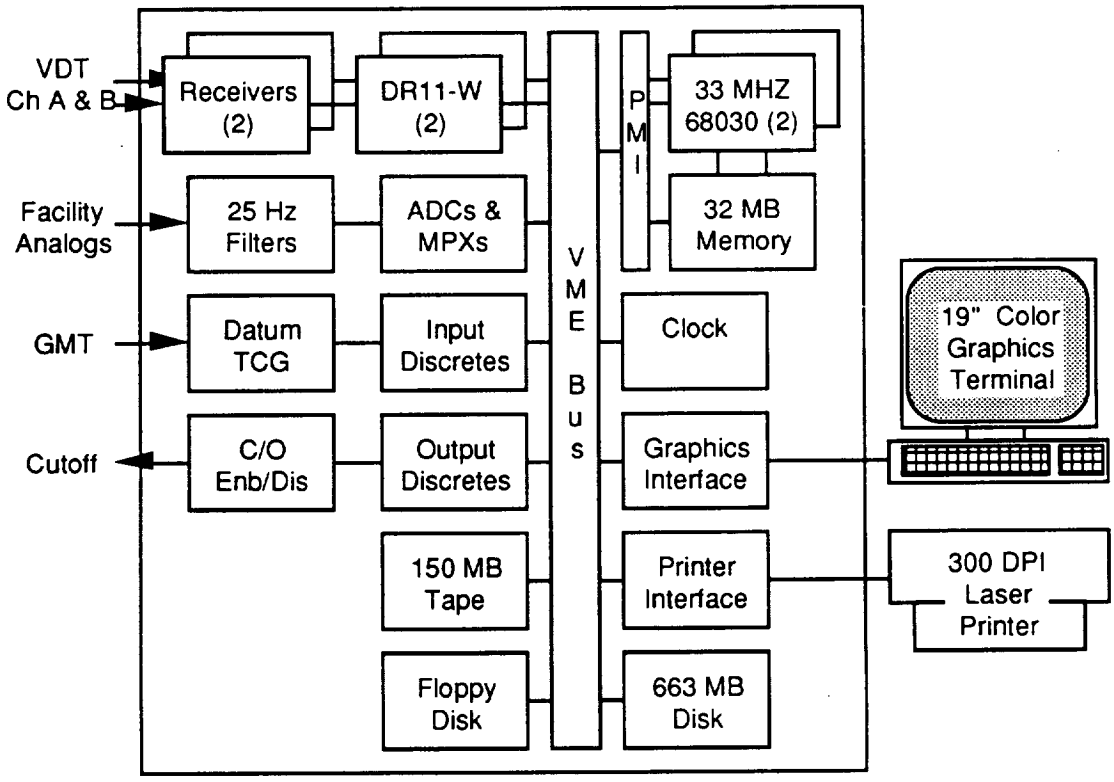


Figure 3 - SAFD Hardware Configuration

The software architecture segregates the algorithm software from the platform software. This approach derived from the fact that NASA and Rocketdyne were aware that the algorithm was being improved and therefore would change and that other algorithms were being developed. This

approach allows easily changing the algorithm software without affecting the entire system, thus limiting software maintenance and retest. It also allows faster algorithm development as the algorithm developer doesn't need to worry about timing, I/O devices, etc. This architecture also provides for running multiple distinct algorithms so that algorithms from different organizations can be executed simultaneously and without interaction. Figure 4 illustrates the architecture.

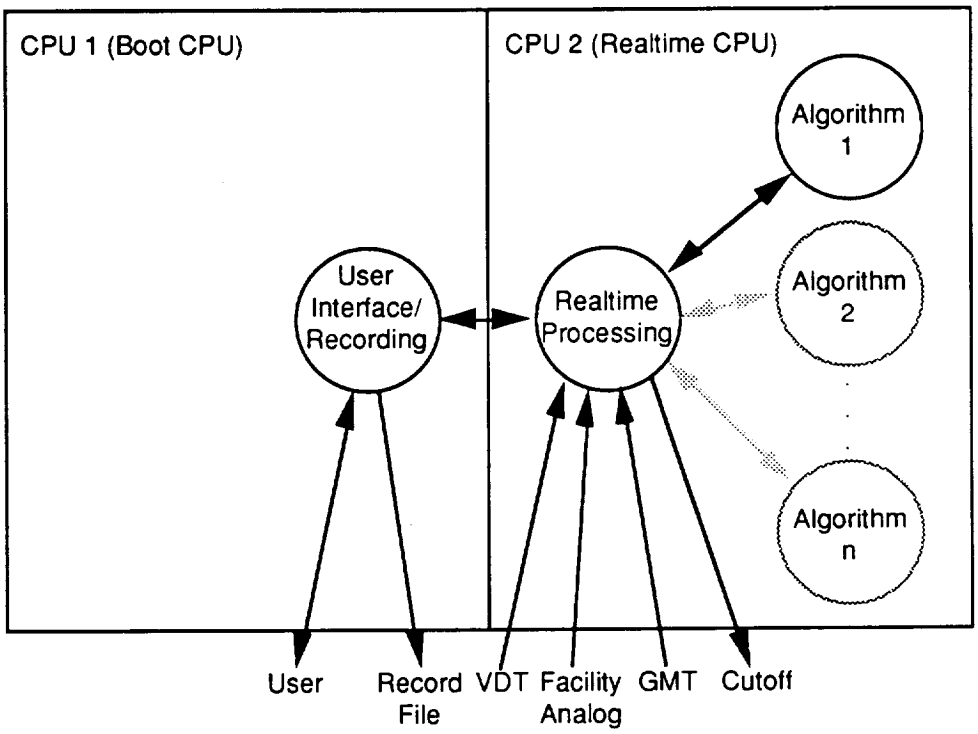


Figure 4 - SAFD System Architecture

The platform software provides basic functions such as data input and qualification, timing and scheduling, recording/playback, display, and a consistent user interface. The operating states of the system include: checkout functions, test setup and execution, test playback, and test simulation.

The checkout capability provides a confidence test for the various input and output devices. Data input checkout includes checkout of the VDT and facility parameters and the GMT. Calibration allows the user to calibrate the facility parameters. Cutoff checkout verifies the cutoff relay circuitry. Data recording checkout verifies proper operation of the recording medium. These checkouts are designed to provide a confidence test only and are not a substitute for comprehensive diagnostics.

The test setup allows the user to establish a test configuration. The system allows the user to specify the hardware configuration, the display

configuration, the default recording mode and time, and the algorithms to be executed. The parameter map defines the hardware configuration and allows adding parameters to the system without changing the platform software. The system includes a graphical display editor that allows the user to actually build the display to be used for a test. Test execution allows the user to execute these predefined test configurations.

Playback allows the user to playback a previously recorded test exactly as it was executed on the stand, and includes the capability to change the display. Thus, the user can examine parameters that were not displayed during the test by merely using the graphical display editor to build a different display.

The simulate mode allows the user to configure a test as in test setup mode, but then allows the user to execute the test configuration against data from a previously recorded test. This allows examination of the behavior of new or modified algorithms against previously recorded tests. As in the test mode, the user can record the simulation run and then view it using playback.

The architecture has proven viable in operation at the TTB and in the HSL. From December 1991 through April 1992, TTB monitored TTB tests 026 through 032. The flexibility of the display capability has allowed the users to redefine displays for individual tests and for post test analysis without having to change the software. In tests 031 and 032, the system executed algorithms from United Technologies Research Center (UTRC) along with the SAFD algorithm without software change.

As a result of the development effort, Rocketdyne identified some limitations and some strengths in vendor's products.

A number of the limitations were related to the operating system and most concerned timing. Rocketdyne discovered that the Concurrent system takes about 10ms to queue an I/O request and AST delivery times are in the 3ms range. These times are much slower than Concurrent's literature indicates. In addition to having slow response times, the system exhibits a random delay of up to 80ms that Concurrent has not been able to identify. Based on experience gained on this project, Rocketdyne concludes that the Concurrent RTU operating system executing on the Concurrent 6450 hardware is not appropriate for applications requiring absolute response times in the millisecond range unless the random delay is fixed and custom drivers are written. The SAFD project required additional effort in order to work around the limitations.

An off-the-shelf software package, Data Views from VI Corporation, provides the display capability for the system. This package proved very cost effective in that it provided all of the functionality required for display for far less than

it would have cost to develop equivalent functionality. Rocketdyne recommends this package where flexible user display capability is required.

This document discusses the operating system and Data Views in detail in sections 2.2.1 and 2.4.

The algorithm performed well in TTB tests, but the tests emphasized the importance of having the proper adaptation data for the algorithm. In the first three tests, some parameters indicated out-of-limits due to the fact that the adaptation data did not accommodate scheduled mixture ratio changes and CCV excursions. Not having the correct adaptation data inadvertently proved the viability of the algorithm in that the algorithm determined that the engine was operating abnormally. Effort is currently underway to establish a formal procedure for generating adaptation data for tests.

Operation during the TTB tests also indicated that the algorithm operation could be improved by establishing limits for time segments during the test rather than for power levels. This approach allows using closer limits because factors other than power level can be incorporated in the calculation of the limits. The software is currently being changed to implement this approach.

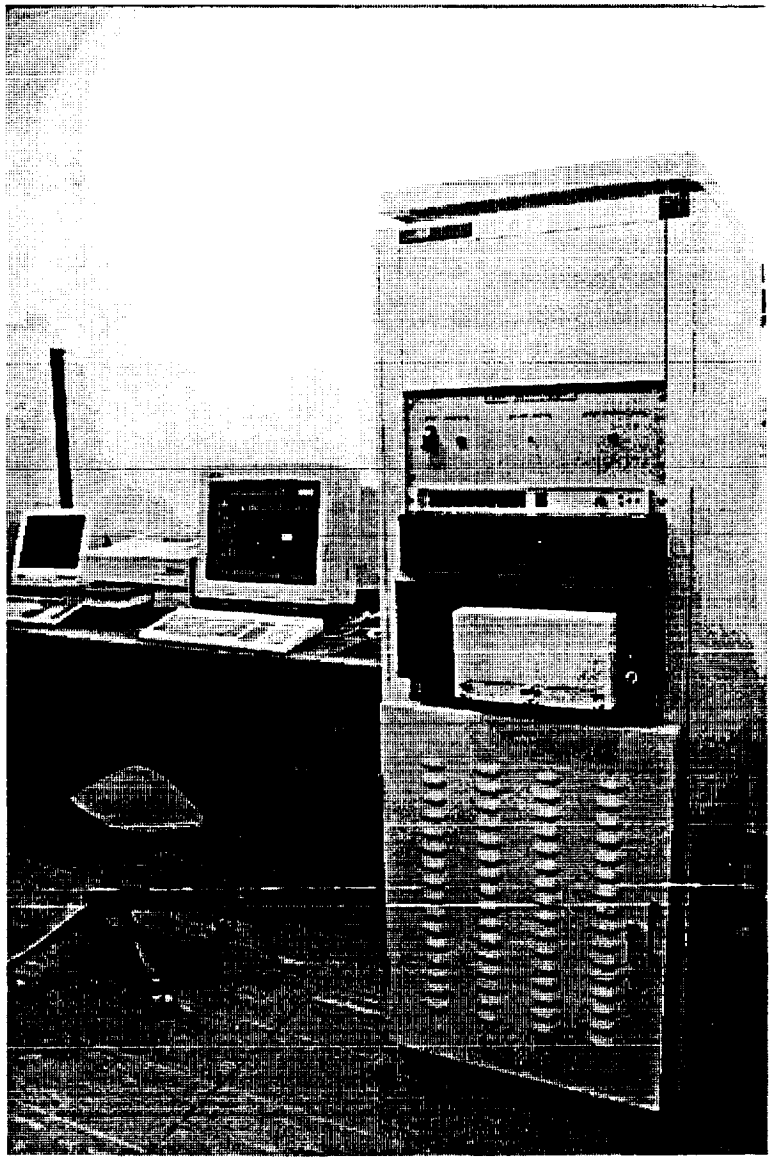
Algorithm testing in the HSL indicates the need for better sensor qualification. The algorithm is currently sensitive to channel A power failures. Several approaches have been identified to correct this. Additional effort is underway to perform testing to determine other areas where the algorithm might be sensitive to failures from which the SSMEC can recover. Rocketdyne expects that most, if not all, of these will only affect adaptation data and will not require changes to the algorithm.

The SAFD effort is now in the maintenance phase. In addition to the three efforts above, NASA and Rocketdyne are working to identify other tasks that need to be accomplished.

The SAFD project has been successful in that a versatile platform now exists for experimenting with various approaches to engine health monitoring in real time. The SAFD algorithm has been implemented on the platform as well as an algorithm from UTRC. Other benefits include the lessons learned during the development and operation of the system. Although problems were encountered during development, the approaches and architectures have proven useful concepts that can be applied to future projects.

1 Introduction

TTB STA 23 specified building the hardware and software to implement the algorithm being developed under STA 21. The task involved building two units: one to be installed in the HSL and one to be installed at the TTB. Rocketdyne personnel at the HSL performed the task.



Unit 2 at TTB

1.1 Document Overview

This report relates in detail the approaches taken, the lessons learned, and recommendation for future efforts. The report is broken down as follows:

Section 1	Introduction
Section 2	The SAFD Platform
Section 3	The SAFD Algorithm
Section 4	Other Algorithms
Section 5	Summary

1.2 Deliverables

The deliverables for the task included the two systems, including the platform and algorithm software, and appropriate documentation. Table 1 ennumerates these items.

Item	ID	Description
Doc	RHF-0032-001 Rev A	System Specification
Doc	RHF-0032-005	System Development Plan
HW	SAFD serial # 1	SAFD Hardware
HW	SAFD serial # 2	SAFD Hardware
Doc		SAFD Hardware Drawings
SW	Platform v2.0	Platform Software
Doc	RHF-0032-003	Platform Software Requirements
Doc	RHF-0032-007	Platform Software Design
Doc	RHF-0032-011	Platform Test Plan
Doc	RHF-0032-013	Platform Test Description
Doc	RHF-0032-015	Platfrom Test Report
Doc		Platform Version Description Doc
SW	Algorithm v1.0	Algorithm Software
Doc	RHF-0032-020	Algorithm Software Requirements
Doc	RHF-0032-021	Algorithm Software Design
Doc	RHF-0032-022	Algorithm Test Plan
Doc	RHF-0032-023	Algorithm Test Description
Doc	RHF-0032-024	Algorithm Test Report
Doc		Algorithm Version Description Doc

Table 1 - Deliverables

1.3 Environment

SAFD is designed to operate in the TTB environment. The system obtains VDT input from a spare VDT output in the CADS, facility measurements from the facility SIUs, and GMT from the facility GMT lines. It generates a

cut signal by closing a relay connected to the facility cutoff panel. Figure 5 illustrates the configuration of SAFD at the TTB.

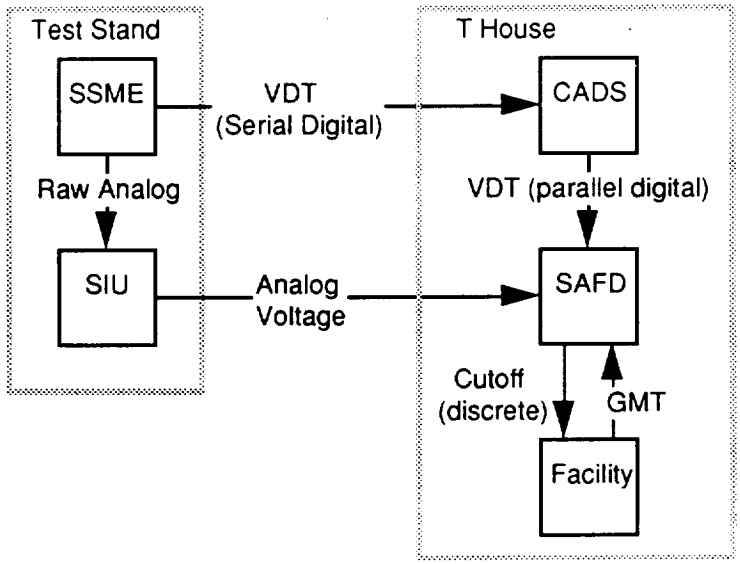


Figure 5 - SAFD TTB Configuration

1.4 System Overview

During the system definition phase, NASA and Rocketdyne agreed that it would be cost effective to separate the platform, which included the system hardware and those software functions not directly associated with the algorithm, from the algorithm implementation. The reasoning behind the decision was that the SAFD algorithm was being expanded to include transients and that at least two other efforts were underway to develop algorithms. This decision led to a system which allows multiple algorithms executing simultaneously and allows updating existing algorithms or creating new algorithms without modification of the platform software or hardware.

This modular approach led to a system where the platform handles all input/output, scaling, scheduling, recording/playback, display, and user interface as these functions are common to all algorithms. Isolating these function from the algorithms yields a stable platform upon which the algorithms can be executed. Since the algorithms do not contain generic functions, only the code directly required to implement a particular monitoring approach need be contained in the algorithm. The algorithms are thus isolated from the user and the hardware environment. Since the developer need not worry about the generic functions handled by the platform, it is easier to change existing algorithms and to create and integrate new algorithms. Figure 6 illustrates the concept.

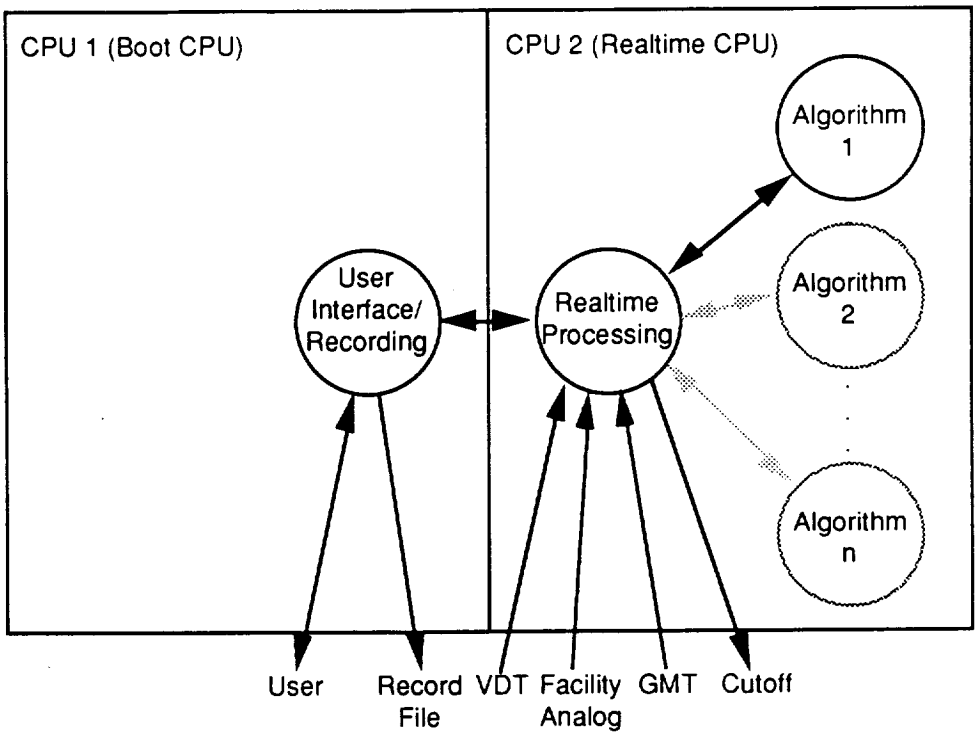
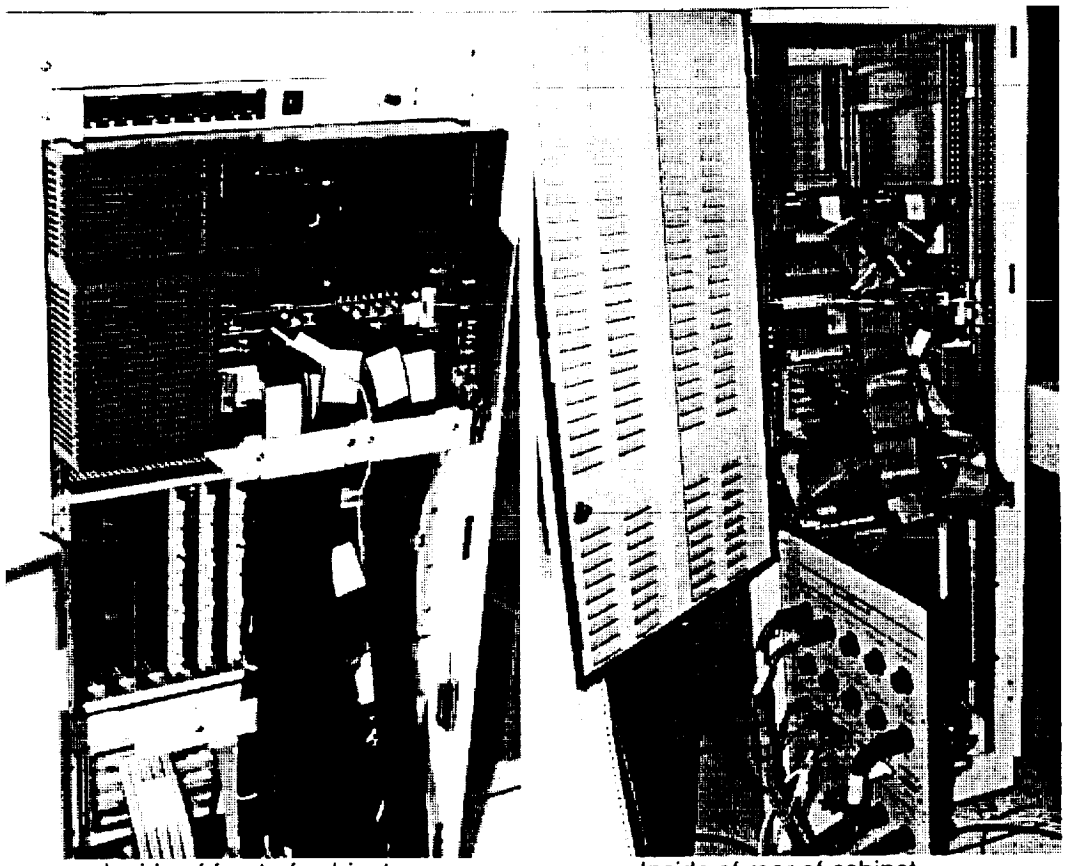


Figure 6 - SAFD System Architecture

2 SAFD Platform

Rocketdyne conducted trade studies to choose the basic hardware for the system. Vendors whose products were considered includes Digital Equipment Corporation, Intel Corporation, Sun Microsystems, Concurrent Computer Corporation, and a custom hardware configuration. The criteria included processing power, response times, and cost. The custom hardware configuration was eliminated because the consensus was that the additional software development cost of having to program low level operating system type functions would exceed the savings on hardware. Sun Microsystems was eliminated because they only support UNIX which is a non-deterministic system and therefore not suited for realtime tasks. Concurrent won the final evaluation based on processor power, expandability, compatibility, and cost.



Inside of front of cabinet Inside of rear of cabinet

2.1 SAFD Platform Hardware Description

The SAFD platform hardware includes all hardware purchased or developed under the task. The hardware is built around a Concurrent 6450 computer using off-the-shelf components where available.

Rocketdyne built custom hardware for those components not available off-the-shelf. Figure 7 shows a block diagram of the SAFD system.

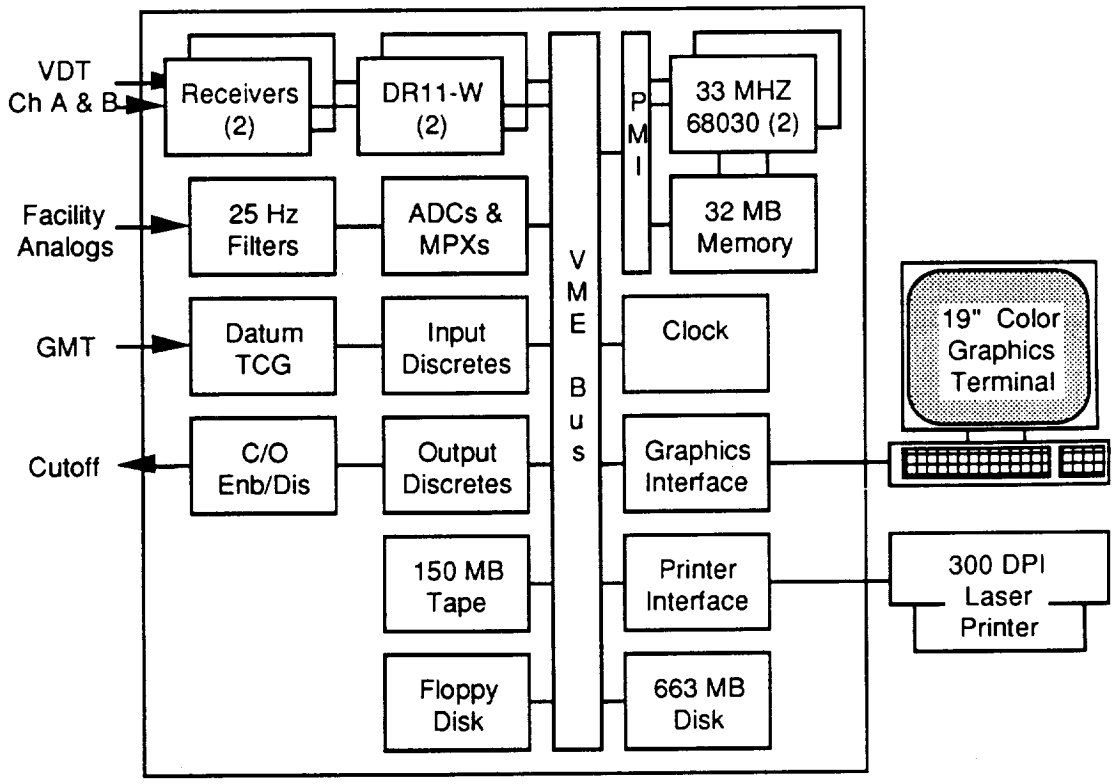


Figure 7 - SAFD Block Diagram

The major hardware components include the following:

Concurrent 6450 computer and peripherals
VDT interface
Facility analog interface
GMT TCG and interface
Cutoff logic

2.1.1 Concurrent 6450

The Concurrent 6450 is the basic building block for the SAFD hardware. It includes the following components:

Two Motorola 68030 33MHz processors
32 MB memory
660 MB disk drive
150 MB tape drive
$5^1/_4$" floppy disk
Programmable clock

The two processors are commonly referred to as the boot processor (CPU 1) and the realtime processor (CPU 2). The SAFD application executes tasks not having stringent timing requirements on CPU 1 and executes those with stringent timing requirements on CPU 2.

The programmable clock was added to the system because the line frequency clocks provided with the system did not have enough resolution for SAFD use.

2.1.2 VDT interface

The VDT interface uses the spare VDT output from the CADS. It then duplicates the signal to replace the spare that it used and drives a copy of the signal through long line drivers to receivers located in the SAFD. These receivers provide the front end to the DR11-W boards in the SAFD system. Rocketdyne custom built all of these components except the DR11-Ws which were purchased from Concurrent. Figure 8 illustrates the configuration of the VDT interface.

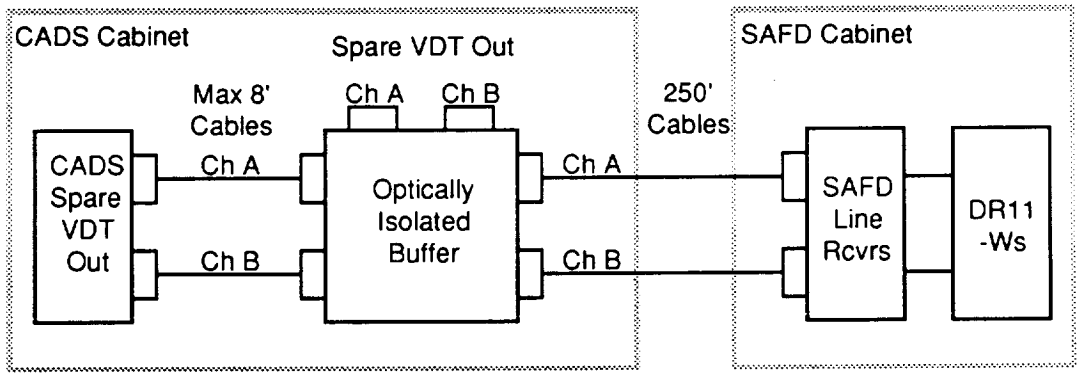


Figure 8 - VDT Interface

2.1.3 Facility Interface

The facility analog inputs are filtered by 25 Hz filters before entering the ADCs. SAFD includes an ADC card and a multiplexer accommodating 40 inputs. The inputs are expandable to 136 parameters. Rocketdyne custom built the filters and wiring harnesses while the ADC cards and multiplexers were purchased from Concurrent.

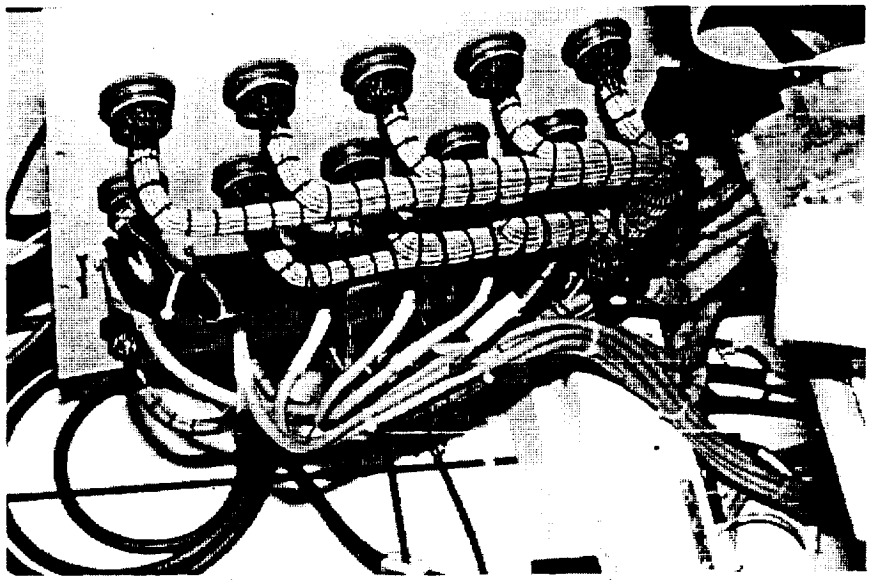
2.1.4 GMT

The GMT time pulse is acquired from the facility GMT coaxial cable by the time code generator housed in the SAFD system. The time code generator

decodes the time and converts it to parallel output which is read by the software through the input discretes.

2.1.5 Cutoff Logic

The cutoff logic enables the software to close a normally open contact to complete a circuit from the facility cutoff. In order to complete the circuit, the cutoff must be enabled at a guarded toggle switch on the front panel of SAFD.



Wiring to rear connector panel

2.2 SAFD Platform Software Description

The SAFD platform software includes all software not directly associated with an algorithm. Functions not requiring realtime response are executed on the boot processor (CPU 1). Those requiring realtime response and the algorithms are executed on the realtime processor (CPU 2). Figure 9 illustrates the software/hardware mapping for the system.

2.2.1 SAFD Platform Functions

The platform software provides the following functions for algorithms:

Parameter input, scaling, and qualification
Cutoff request to facility
Scheduling
Recording/Playback
GMT acquisition
Display
User interface

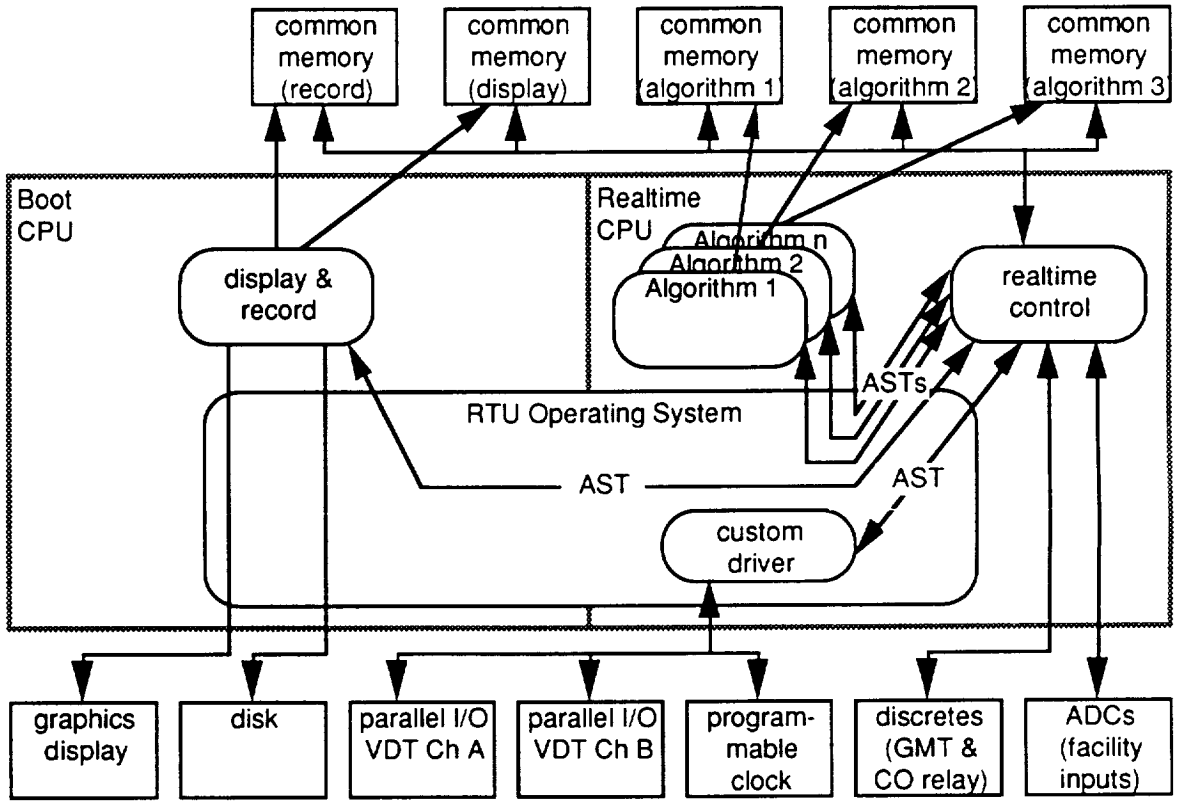


Figure 9 - Software/Hardware Mapping

The SAFD platform inputs, scales, and qualifies parameters prior to sending them to the algorithms. It determines the location, scaling, and qualification limits for the parameters via the parameter map. The parameter map is an ASCII file, built by the user, describing the parameters available to the system.

The SAFD task uses the parameter map in conjunction with the algorithm map to determine the input and output parameters for algorithms. The algorithm map tells the SAFD software what inputs and outputs are required and indicates their order. Any inputs specified in the algorithm map must exist in the parameter map or as an output in another algorithm's map. The generic main, supplied as part of the SAFD system software, then sets up the calling parameters for the algorithm routine and calls the algorithm routine. Figure 10 illustrates the mapping process for algorithm inputs and outputs.

This approach allows installation of an algorithm without modifying the software for the platform. If additional parameters are required, the user simply adds them to the parameter map (if they are not already there) and includes them in the algorithm map.

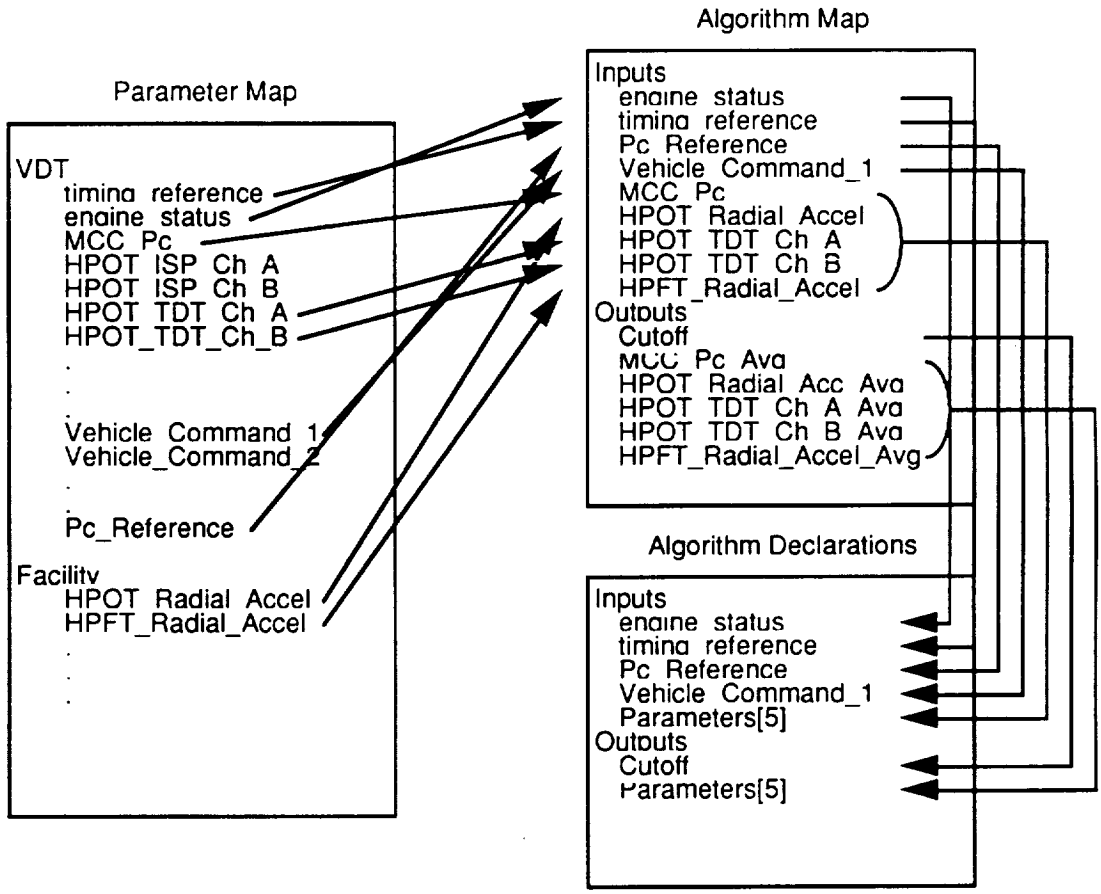


Figure 10 - Algorithm Parameter Mapping

SAFD assumes that the first output parameter from the algorithm is the cutoff request. Thus, when an algorithm detects a cutoff condition, it sets its first output parameter to non-zero. When the algorithm completes the current cycle, the SAFD task examines the parameter and, if it is non-zero, the SAFD task closes the cutoff relay.

The SAFD software allows the use of rate monotonic scheduling to schedule algorithms. Under this scheme, algorithms with shorter scheduling intervals are scheduled at a higher priority than those with longer scheduling intervals. This allows those with longer intervals to run during the idle periods between executions of those with shorter intervals. Figure 11 illustrates this concept. In order to provide maximum flexibility, the SAFD system allows the user to specify the order of the algorithms. Hence, the user may elect whether or not to use rate monotonic scheduling.

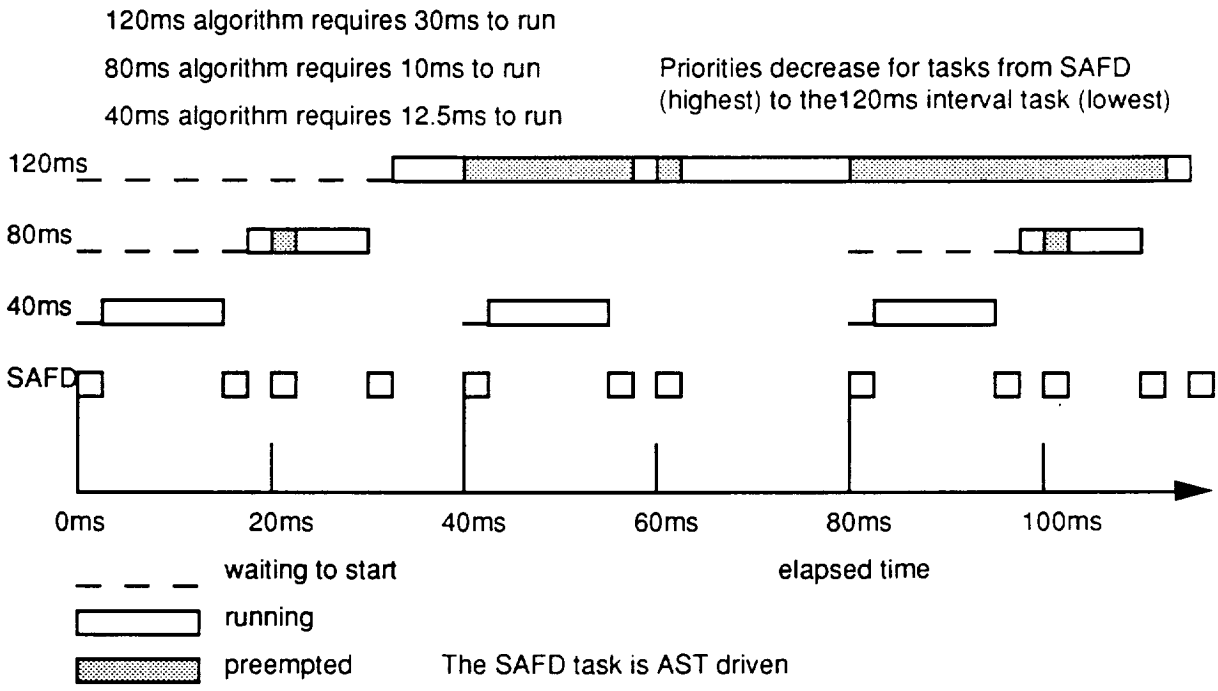


Figure 11 - Algorithm Scheduling

SAFD determines the scheduling interval from the algorithm map file. Just as in the case of input and output parameters, scheduling for algorithms can be changed without modifying the SAFD platform software.

The SAFD task also handles recording and playback functions. During test and simulate modes, if recording is on, SAFD records the following items:

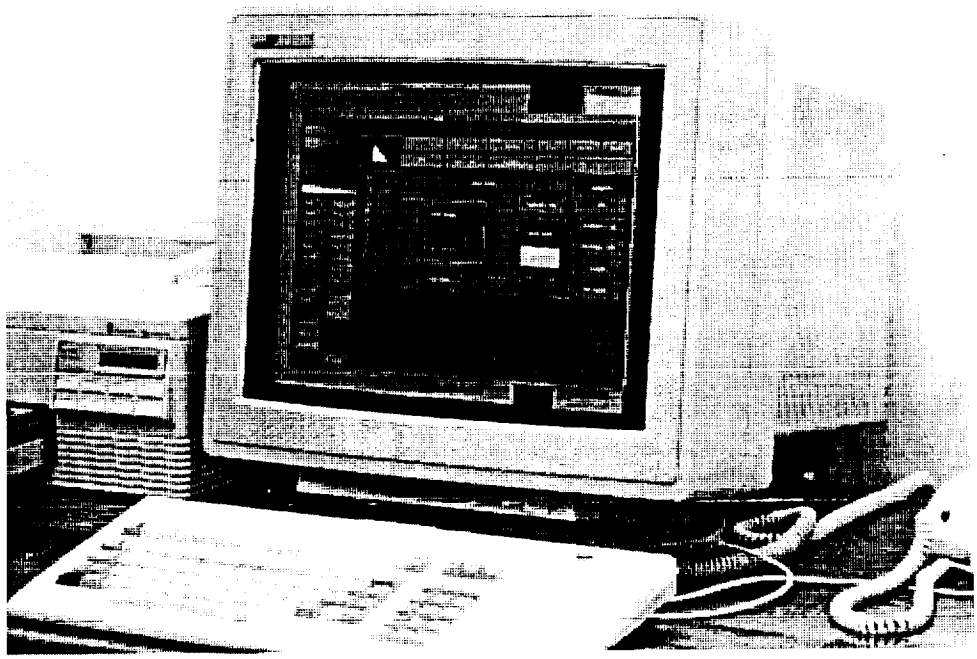
VDT tables
Facility parameters
Algorithm inputs
Algorithm outputs
SAFD status data
GMT

Note that VDT and facility parameters are all recorded, even if they are not used by an algorithm. This allows the user to play a test recording into an algorithm in simulate mode, even if that algorithm uses parameters other than those used by the algorithm active during the test. It also allows the user to view parameters in playback that were not used by the algorithm.

The SAFD task time stamps all of the data with the GMT which is acquired from the facility. This time is displayed on the screen during test, playback, and simulate.

SAFD provides extremely versatile and powerful display capability. This capability is afforded by Data Views, an off-the-shelf software package from VI Corporation. Using this package the user can build custom data displays for use in test, playback, and simulate. The package includes over 60 predefined displays such as bar charts, digital readouts, strip charts, and pie graphs. The package also allows the user to build custom displays.

The Data Views graphical editor allows the user to build the data displays interactively. The SAFD software makes all parameters defined in the parameter map and all outputs from all scheduled algorithms for the chosen test configuration available to the editor. When the user enters the editor, he simply selects the parameters to display and configures the desired display. The user can configure any number of displays for a test configuration.



Data Views editor

Since the user selects the display to use in test, playback, and simulate individually, he may use one display during the test, enter playback to view that test, and specify a different display for playback. The same capability exists for simulate mode. This avoids restricting the user to a fixed display or to viewing only parameters displayed during the test. It also allows the user to construct and use multiple displays without having to modify the SAFD or the algorithm software.

The SAFD user interface is a graphical user interface based on the Motif standard. The graphical user interface makes the system easier for the user to learn and operate.

2.2.2 SAFD Platform Operation

The SAFD platform software operates in the states and modes shown in Table 2.

State	Mode	Description
Checkout	Data Input	Checkout facility analog, VDT, and GMT inputs
	Data Recording	Checkout recording file
	Calibration	Calibrate facility parameters
	Cutoff	Checkout cutoff circuitry
Test	Setup	Build test configuration
	Execute	Execute test configuration
Playback	Setup	Setup playback configuration
	Execute	Execute playback
Simulate	Setup	Setup simulate configuration
	Execute	Execute simulation
Utilities	DVEditor	Build display

Table 2 - SAFD Platform States and Modes

The user generally invokes the various checkouts to perform confidence checks on the hardware. Calibration is the exception in that it is used to calibrate the facility parameters prior to test.

The following paragraphs describe operations during a typical test.

Prior to setting up for a test, the user must ensure that a valid parameter map exists in the test directory and that the algorithms and their appropriate maps are in the test directory. After entering the test setup mode, the user selects the appropriate parameter map, names a view (display) file (which may or may not exist), chooses the default recording mode, specifies the test length, and selects the algorithms to be executed. After entering this data, the user saves and names the test configuration. Normally, if the view file does not exist, the user will enter the DVEditor

mode to create it. Once the test configuration is saved it can be used at any time.

At some time prior to invoking the test, the user will normally calibrate the facility parameters. This is typically done early on the day of the test. To calibrate, the user must coordinate with test personnel at the TTB. The user selects the parameters to calibrate and requests data points from the test personnel, answering the prompts from SAFD as appropriate. After calibrating, the user selects "apply" and exits the calibration mode. The SAFD software saves the calibration data into the test configuration file.

Prior to the test, the user will enter the test setup mode and select the test configuration that he created and used for the calibration. He may then select "execute" to begin executing the test. Typically, the user selects "auto record" just prior to start. This causes the SAFD system to record one sample every 10 seconds until start when it begins recording all data. At the end of the test, usually after entry into post shutdown, the user turns off the recording and exits the test mode.

After the test, the user enters playback setup mode, selects the test ID and view file, and enters the playback execute mode to view the data. Currently, to facilitate data reduction, view files have been pre-built to plot each of the parameters used by the SAFD algorithm. This allows the user to simply select each view in turn, execute playback, and copy the plots. Of course, the user always has the capability to build display files to display whatever is needed in the format needed.

Simulate works like test except that, rather than using realtime data from the stand, the user specifies a previously recorded test as input. The SAFD system then schedules the selected algorithms and sends them data from the recorded test, just as though the algorithms were executing on the test stand during that test. The only difference is that there are no realtime operations and algorithms are allowed to take as long as necessary to execute.

Simulate allows the user to record the "test", just as in test mode. Therefore, the user can execute algorithms in simulate mode, record the execution, and use playback to analyze the results.

The graphical user interface, the ability to define and select displays, and the playback and simulation capability makes SAFD a powerful and versatile system for testing algorithms as well as for using them on test stands.

2.3 SAFD Platform Development Problems

The development of the SAFD system encountered a number of problems, most due to poor response times through the operating system. During the selection process, system performance was included as a criteria and the Concurrent literature indicated that the 6450 was adequate. However, the literature was apparently based on ideal examples because the times recorded by the development team for functions and response show significantly more system overhead than indicated by the literature. Although work arounds were developed for most problems, development of the work arounds significantly impacted the cost of the project. This section will address each of the problems, the work arounds for the problems, and the impact to the system.

2.3.1 Queueing I/O Requests

Rocketdyne found that queueing a read request took approximately 10 milliseconds. Since SAFD executes in a 20 millisecond loop and requires four reads every 40 milliseconds, this leaves no room for the SAFD processing. This limitation required work arounds for obtaining inputs from the ADCs (facility inputs), the DR11-Ws (VDT), and the discretes (GMT).

For the ADCs, the built-in clocks were set up to control the input. The software initializes the ADCs and maps to them. It then queues up a request to read one buffer more than required and saves the registers. Then, when ADC input is desired, the software restores the registers and strobes the clock to initiate the input. Since the registers are set up to read more than is actually input, the requested read never completes. Upon exiting, the SAFD cancels the read and closes the device.

For the DR11-Ws, which bring in the VDTs, a custom driver was written to handle the devices. As with the ADCs, the software queues a read request to read more than will ever be input, saves the registers, and restores them after each read. Since the VDTs trigger the input and the interrupt is routed through a discrete, the SAFD software only fields the interrupt.

For the discrete inputs, the software maps directly to the device and reads the input directly from the device registers.

2.3.2 Asyncronous System Trap (AST) Processing

The literature indicated that AST delivery times were less than 1 millisecond but the times measured for SAFD indicated the the times were actually closer to 3 milliseconds. The SAFD software uses three ASTs plus two ASTs per algorithm. Thus, with only one algorithm resident,

AST processing uses up to five ASTs every 40 milliseconds and therefore loses 15 milliseconds to AST processing. There was really nothing that could be done to reduce AST overhead other than to reduce the number of ASTs being used. The design already specified the minimum possible number of ASTs.

2.3.3 Single Shot Clock Requests

The manuals for the system indicated that the system clock could be set up to request a single interrupt but this proved untrue. The development team verified that the option did not work and, after consulting with Concurrent via telephone, confirmed that there was an error in the system software. Concurrent sent a fix for the bug.

2.3.4 Clock Resolution

The only clocks available from Concurrent are line frequency clocks. A line frequency clock nominally interrupts every 1/60 seconds. To accommodate realtime applications, Concurrent included the ability to speed the clocks up so that they interrupted every 1 millisecond. However, the time remaining for SAFD processing was already limited and this imposed even more system overhead as the clock had to be serviced every 1 millisecond whether or not SAFD needed it. A programmable clock was purchased from VMIC to eliminate the need for the system clock.

2.3.5 Interrupt Processing Latency

By default, all interrupts in the Concurrent system are processed by the host processor. The development team discovered that this was causing unpredictable interrupt latencies. The work around for this involved strapping the programmable clock and the VDT interrupts to interrupt level 2 on the realtime processor (CPU 2). This modification is a standard field modification offered by Concurrent to improve realtime performance.

2.3.6 VMIC Clock/VME Interrupt Logic

An incompatibility between the VMIC clock and Concurrent machine appeared as a result of employing the field mod to strap interrupts directly to the realtime processor. The incompatibility lay between the bus logic on the programmable clock board from VMIC and the VME bus in the Concurrent machine. At the time neither vendor could identify the problem. A Rocketdyne engineer identified the problem which was then determined to be a deficiency in the VMIC clock board. VMIC modified the clock boards to resolve the problem.

2.3.7 Disabling System Clock

According to the Concurrent manual, the system clock on the realtime processor can be disabled. However, in attempting to do this, the development team found that it didn't work. After consultation, Concurrent agreed that it was an error in the system software. Upon further investigation, Concurrent discovered that the feature worked if tasks ran at a lower priority. Rocketdyne implemented this work around.

2.3.8 System Delay

During development, Rocketdyne had continually observed a random delay while performing system functions. For no apparent reason, the system would gain control and not return for 40 to 80 milliseconds. The exercises with the clocks above were attempts to eliminate this delay. Concurrent was alerted early and has not been able to identify the source of the delay. Rocketdyne and Concurrent personnel are still working to identify the cause. Since the delay is random, it has not materially affected SAFD operations. The development team employed a work around that allows SAFD to continue operation in most instances.

2.4 Conclusions

The experience with SAFD, both in the lab and on the test stand, has proven the viability of the approach. The system has proven versatile and easy to use. In spite of the problems encountered during development, the end product serves the purpose.

The problems encountered with the system added significantly to the development cost. Unfortunately, they also removed many of the advantages of working with an operating system. The expectation of being able to use operating system functions rather than coding low level functions also led to eliminating custom systems from the evaluation. As it turned out, a custom system probably could have been cost competitive.

Based on experience with SAFD, Rocketdyne does not recommend using the Concurrent 6450 for realtime applications requiring response time in the millisecond range unless the "system delay" can be remedied and custom drivers are written.

Rocketdyne does, however, believe that the approach and architecture used is appropriate for the task. The idea of modularizing the components yields a system capable of supporting change and reconfiguration with a minimum of effort. It should be noted that, at least on a high level, some aspects of object based design were applied.

Rocketdyne developed the system using primarily the Ada language. The intent was to develop a system that was portable and to follow NASA's movement toward standards. However, particularly in the user interface, Ada was more expensive to use than C. Part of the problem was that the vendor did not have compatible interfaces between the languages. Since the X Windows libraries, which were used to develop the user interface, were written in C, the use of Ada here caused problems that C would not have encountered. While Ada is good for developing embedded systems, Rocketdyne would not recommend using it for developing user interfaces implemented with X Windows.

The Data Views product proved very successful. It provided the sophisticated data display capability required by the system and provided a mechanism for the user to configure displays without having to change software. Rocketdyne highly recommends this product where flexible and sophisticated user display capability is required.

The system currently supports two algorithms. Given the system overhead, this is approaching the limit of its capacity. More capacity can be added by replacing the current processors with three 68040 processors or with RISC processors. It is also possible to simply add RISC processors to the current machine.

Overall, the system has proven successful. The approach and implementation were sound and the system has performed successfully in the HSL and on the TTB.

2.5 Outstanding Issues

Some problems still exist in the system and are due to operating system deficiencies, design deficiencies, and errors.

The most critical problem outstanding in the system is the delay induced by the operating system (see paragraph 2.3.8).

The design deficiency involves the method used to qualify VDT data. The system currently qualifies VDT data by qualifying it within a range which is set by adaptation data. The current values set for the adaptation data represent the physical range of the sensor. With the range that wide, there are certain sensor failures that would not be detected by the SAFD platform and therefore would result in erroneous values being provided to the algorithms. There are basically three methods to address this problem, each with advantages and disadvantages. These are enumerated below.

Approach	Advantages	Disadvantages
Change the sensor qualification range to reflect the normal range of a working sensor during engine on phases.	No changes required to controller or SAFD software	Probably will not work for all failures. Difficult in some cases to tell whether sensor failed or parameter is out of limits
Use the failure reports in the VDT to disqualify parameters.	In most cases, sensors disqualified by controller will also be disqualified by SAFD.	Possible that for numerous failures, some failure reports will be lost. Some SAFD parameters are not qualified by controller. Requires change to SAFD software.
Modify the controller software to tag bad measurements.	In all cases, sensors disqualified by controller will also be disqualified by SAFD.	Some SAFD parameters are not qualified by controller. Requires change to controller and SAFD software.

There are known errors in the software, mostly in the user interface. However, these errors are minor in nature and pose no risk to the operation of the system. When convenient, these should be fixed.

2.6 Recommendations

Rocketdyne recommends the following tasks as follow on effort for the SAFD platform program.

The system delay can cause the system to halt. Therefore, the cause of this delay should be determined and, if possible, corrected. Currently a work around has been implemented in the software to lessen the chance that the system will halt.

The sensor qualification should be improved. As currently implemented and with the current adaptation data, a channel A power failure in the controller will cause the SAFD algorithm to erroneously request shutdown. One or more of the approaches outlined in paragraph 2.5 should be implemented to correct this deficiency.

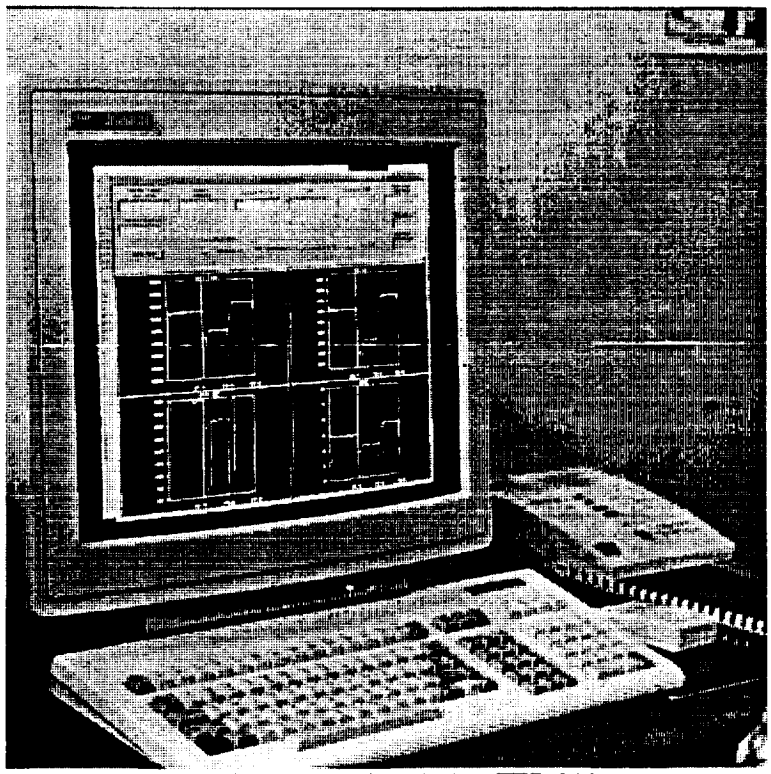
Vendor support should be maintained for both systems. Even though the systems are entering the maintenance phase, they cannot be considered mature. The system software is still evolving and there are outstanding problems. Typically vendors will only provide support for their latest revisions of software. Maintaining vendor support for the hardware insures the program against catastrophic failures in the systems. If a critical part failed without maintenance, it could be months to obtain the parts through normal purchasing channels thus delaying the program. Therefore, it is in NASA's best interest to maintain vendor support for the hardware and software.

While the system has excellent data reduction capability, added capability is desirable. One such capability is the ability to generate a report showing the parameters indicating out of limits and the time that they were reported out of limits. A crude version of this capability was developed by the test team to aid testing, but this crude version does not work properly if other algorithms are active or if the inputs or outputs in the SAFD algorithm change. An integrated capability should be added to the system.

There are a number of outstanding SPRs against the system. Although none but the system delay (mentioned above) are critical, they should eventually be closed.

3 SAFD Algorithm

The requirements for the SAFD algorithm originated with the work done at Rocketdyne in Canoga Park, California under TTB Task 21. The SAFD algorithm was developed as an improvement over the current redline system used in the Space Shuttle Main Engine Controller (SSMEC). Simulation tests and execution against previous hot fire tests demonstrated that the SAFD algorithm can detect engine failures as much as tens of seconds before the redline system recognized the failure. Although the current algorithm only operates during steady state conditions (engine not throttling), work is underway to expand the algorithm to work during transient conditions.



Algorithm operation during TTB-028

3.1 Algorithm Approach

The SAFD algorithm only monitors during mainstage, steady state (with respect to power level) operation. If it detects three (adaptation data) parameters out of limits, it will request shutdown.

The adaptation data for the algorithm includes the cutoff count (nominally three) and the following data for each parameter:

power level for this set of adaptation data
precalculated mean
precalculated standard deviation
N1 factor
N2 factor

The adaptation data for each parameter includes the above set of values for each power level at which the engine will operate during the test. The algorithm loads the adaptation data when initialized by the SAFD platform software.

Figure 12 illustrates the logical operation of the algorithm. SAFD is only active during mainstage, steady state (with respect to power level) operation. It operates in three modes while active; first instance, calculation interval, and steady state.

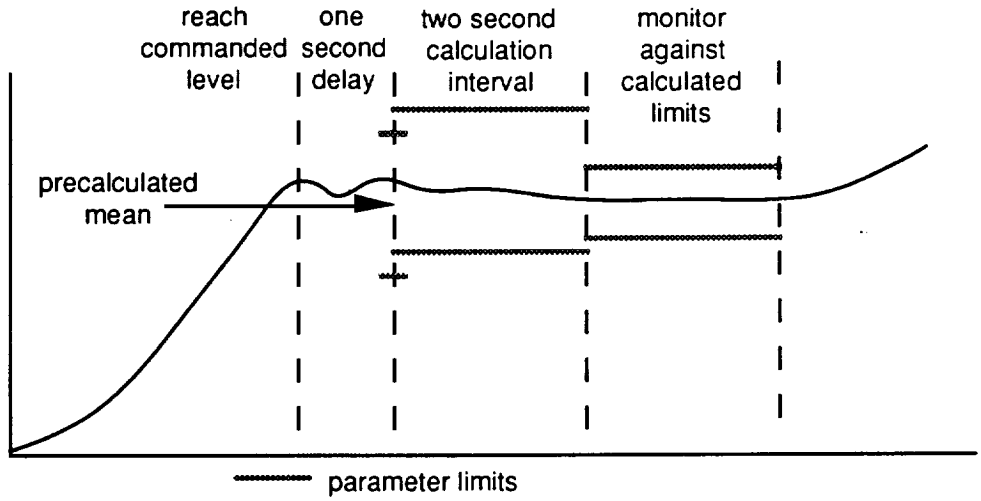


Figure 12 - Algorithm Operation

During mainstage operation, upon detecting a power level command, SAFD suspends operation until PcRef reaches the commanded power level. After a one second delay, SAFD performs a first instance check of the parameter against limits calculated from adaptation data and, if the parameter is within limits, SAFD establishes new limits to be used during the next two seconds. It calculates the limits for the first instance check as follows:

upper limit = adaptation data mean +
(adaptation data standard deviation * N1 factor)

lower limit = adaptation data mean -
(adaptation data standard deviation * N1 factor)

It calculates the limits for the two second interval as follows:

upper limit = first instance parameter value +
(adaptation data standard deviation * N1 factor)

lower limit = first instance parameter value -
(adaptation data standard deviation * N1 factor)

During the two second interval, SAFD maintains a running average of the most recent five samples of each parameter and compares that running average against the limits. If it has not accumulated five samples, it averages the samples available. At the end of two seconds, SAFD calculates steady state limits based on the deviation calculated for the two second interval, the N2 multiplier factor from the adaptation data, and the current running average. It calculates these limits as follows:

upper limit = running average at end of interval +
(standard deviation from interval +* N2 factor)

lower limit = running average at end of interval -
(standard deviation from interval * N2 factor)

SAFD remains in the steady state mode until another power level change occurs.

3.2 Algorithm Parameters

The algorithm currently includes 22 parameters; 16 from the Vehicle Data Table (VDT) and 6 from facility. However, only 5 of the facility parameters are available as the facility is using the available amplifiers for HPFTP Balance Cavity Pressure. Table 3 lists the parameters currently included in the SAFD algorithm.

Parameter	VDT	PID
HPFTP Shaft Speed Ch A	96	260
HPFTP TDT Ch A	16	231
HPFTP TDT Ch B	17	232
HPFTP Discharge Pr Ch A	29	52
HPFTP Radial Accel 90°	N/A	1953
HPFTP Coolant Liner Pr Ch A	14	53
HPFTP Balance Cavity Pr	N/A	457
HPOTP Discharge Pr Ch A	30	90
HPOTP TDT Ch A	18	233
HPOTP TDT Ch B	19	234

HPOTP ISP Pr Ch A	20	211
HPOTP Boost Pump Disch Pr Ch B	33	59
HPOTP Boost Pump Radial Accel 45°	N/A	1994
LPFTP Shaft Speed Ch A	82	32
LPOTP Pump Discharge Pr Ch A	70	209
HEX Venturi Delta Pr	N/A	8352
HEX Bypass Mix Temp	N/A	8359
MCC Pr (qualified average)	6	63
MCC Liner Cavity Pr Facility	N/A	1951
OPOV Actuator Pos	28	40
FPOV Actuator Pos	27	42
Fuel Flowmeter (qualified average)	7	100

Table 3 - SAFD Algorithm Parameters

3.3 Test Experience

From December 1991 through April of 1992, SAFD monitored tests TTB-026 through TTB-032. These tests have represented a variety of power levels and operating conditions and have confirmed the validity of the algorithm. They have also demonstrated the need for accurate adaptation data. In all cases the cutoff was disabled.

The adaptation data for the first three tests did not take into consideration the mixture ratio changes, thereby causing the SAFD algorithm to flag parameters out of limits. This demonstrated the detection capability of the algorithm in that it did flag conditions that would have been anomalous had the mixture ratio not been intentionally changed.

3.3.1 TTB-026

TTB-026 ran for full duration of 170 seconds. The test included a mixture ratio change to 6.16. Since the adaptation data was not adjusted for this perturbation, SAFD noted the following parameters out of limits during the test:

LPFTP shaft speed
Pc
Fuel Flow

However, since the three parameters were not all out of limits simultaneously, SAFD did not request a shutdown.

The facility parameters were not available for this test.

3.3.2 TTB-027

TTB-027 shut down prematurely at 40 seconds due to loss of a facility system. This test included mixture ration excursions to 5.85. Again, the SAFD adaptation data was not adjusted to accommodate this perturbation, and SAFD reported parameters out of limits. The parameters detected out of limits were:

HPFT TDT Ch B
HPOT TDT Ch A
HPOT TDT Ch B
OPOV Pos
Fuel Flow

This time there were three parameters out of limits simultaneously and SAFD requested a cut at 28.48 seconds. Since the cutoff was disabled, this did not actually shut down the engine.

Sixteen VDT parameters and five facility parameters were monitored during this test.

3.3.3 TTB-028

TTB-028 ran full duration of 210 seconds. This test included mixture ration excursions to 6.86. This test also included CCV excursions. The adaptation data generated for this test included allowances for the mixture ration changes but not for the CCV excursions. Two parameters registered out of limits. Fuel flow indicated out of limits due to an error in calculating the adaptation data for 86% power level. LPFP shaft speed indicated out of limits at the CCV excursion.

Sixteen VDT parameters and five facility parameters were monitored during this test.

3.3.4 TTB-029

TTB-029 prematurely shutdown less than 1 second after start. Therefore, no algorithm data was gathered.

3.3.5 TTB-030

TTB-030 shutdown prematurely at about 5.5 seconds. Thus, the SAFD algorithm only executed for about .5 seconds during which time there were no anomalies.

3.3.6 TTB-031

TTB-031 ran full duration of 85 seconds. The SAFD algorithm detected no anomalies.

3.3.7 TTB-032

TTB-032 ran full duration of 205 seconds. The algorithm indicated HEX Bypass Mix Temperature out of limits during the 2 second interval at 115 seconds and 128 seconds (power levels 104 and 100 respectively). The parameter limits were good for these power levels early in the test but early in the test Oxidizer Inlet Pressure was at 120 psi. When the out of limit conditions occurred, Oxidizer Inlet Pressure was at 20 psi. During the venting at 109% power level, HPOT ISP Pressure increased toward the upper limit while LPOP Discharge Pressure decreased toward the lower limit. However, neither of these parameters exceeded the limits.

3.4 Conclusions

The algorithm performed as expected in the HSL and on the TTB. However, the testing indicated two areas in which the algorithm is vulnerable.

Testing in the HSL and analysis demonstrated the algorithm's sensitivity to channel A power failures. Several approaches can remedy this problem, so it really doesn't detract from the suitability of the algorithm for engine monitoring. These approaches are addressed in section 2.5.

Performance on the TTB indicates the critical nature of the adaptation data and emphasized the necessity of ensuring that the adaptation data is correct.

Rocketdyne believes that the SAFD algorithm represents a viable approach to engine health monitoring. It demonstrated greater sensitivity to engine anomalies than the redlines currently being used. Its ability to use and generate limits based on engine operation makes it much more flexible than the current redlines. However, based on test experience, the key to success in use of the algorithm is in proper generation of the adaptation data.

3.5 Recommendations

Testing in the HSL and at the TTB validated the usefulness of the algorithm. However, this testing also indicated that areas exist that need further work.

First of all, the algorithm should be modified to accept adaptation based on time rather than power level. This allows using closer tolerances for the limits since the data can be tailored for operation at a particular time during the test rather than being solely based on power level. For example, if a test changes mixture ratio from 6.01 to 6.16 at 100% power level the limits could be set individually for operation at 6.01 mixture ratio/100% power level and 6.16 mixture ratio/100% power level rather than expanding the limits to accommodate mixture ratios from 6.01 to 6.16 at 100% power level.

The platform should be modified to guard against sensor failures that would not be caught by a simple limit check. The options for this modification were enumerated earlier in section 2.5. Note that this change should be implemented in the platform software rather than the algorithm software.

Currently, the adaptation data is acquired and entered in an informal fashion. A formal procedure should be established with the appropriate cross checks to ensure that the proper adaptation data is generated and entered for each test.

Further validation testing of the algorithm is needed, particularly in the area of avionics failures from which the controller can recover. The testing performed to data in the HSL centered primarily on verification testing, which verifies that the system performs per requirements. Additional validation testing would serve to prove that the system performs as intended. An initial set of tests have been defined by NASA and Rocketdyne, but a formal method for generating adaptation data must be established for the testing to be meaningful.

4 Other Algorithms

The SAFD platform was designed to accommodate multiple algorithms. The decision to do so was based on the fact that other algorithms were being developed through LeRC and they would require a platform. Designing the platform to accommodate multiple algorithms saves money by eliminating the need for additional platforms and reduces the complexity of the facility by only requiring one connection for the SAFD per parameter rather than requiring one connection per algorithm. It has demonstrated that capacity on tests TTB-031 and TTB-032.

No anomalies have been noted when executing the UTRC algorithms. However, the timing indicates that there is only limited room for further growth. Additional algorithms will require upgrading the processors in the SAFD platform or combining new algorithms with existing ones.

5 Summary

The SAFD project has been successful in demonstrating the viability of the SAFD platform and algorithm. The strategy of separating the two proved very successful in that the system is able to accommodate additional algorithms with little effort. The user interface has also proved convenient and represents an improvement over command line interfaces. The data reduction represents a significant improvement over similar systems in that it can easily be controlled by the user. The approach and capability of the system can be used as a pattern for future development.

Unfortunately, the Concurrent system proved to be somewhat of a disappointment. While it is adequate for the task, the excessive system overhead limits its capacity to accommodate additional algorithms. However, this deficiency can be remedied by upgrading to faster processors or adding processors.

The algorithm behaved as expected. While performance in early tests demonstrated the algorithm's ability to detect off-nominal conditions in an engine, it also demonstrated the importance of correctly determining the adaptation data.

Some outstanding problems should be fixed and some improvements should be implemented in the platform software. These include:

- Find and cure the system delay problem.
- Close outstanding SPRs.
- Implement a better qualification scheme for VDT parameters.
- Add an integrated capability to obtain a report enumerating parameters indicating out of limits during a test.

Improvements should be made in the algorithm as well and further testing is warranted. Candidate follow on work includes:

- Change the algorithm to use adaptation data based on time rather than on power level.
- Establish a formal procedure and methodology for generating adaptation data.
- Perform additional testing in the HSL to establish the algorithm's sensitivity to recoverable failures.
- Update the algorithm to incorporate the capability to monitor during transients (work done under STA 21 in Canoga Park).

Overall, the project was successful. The two systems were built and are performing well on the TTB. The algorithm has behaved as expected and

has demonstrated its sensitivity to off nominal engine operation. Most of the outstanding problems in the platform software are minor. Those that are significant are already being addressed as part of the maintenance of the system.

6 Acronyms

ADC	Analog to Digital Converter
ASCII	American National Standard Code for Information Interchange
AST	Asynchronous System Trap
DC	Direct Current
CADS	Command And Data Simulator
CCV	Coolant Control Valve
CPU	Computer Processing Unit
FPOV	Fuel Preburner Oxidizer Valve
GMT	Greenwich Mean Time
HEX	Heat Exchanger
HPFTP	High Pressure Fuel Turbopump
HPOTP	High Pressure Oxidizer Turbopump
HSL	Hardware Simulation Laboratory
ISP	Intermediate Seal Purge
I/O	Input/Output
LPFTP	Low Pressure Fuel Turbopump
LPOTP	Low Pressure Oxidizer Turbopump
MB	Megabyte
MHz	Megahertz
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
OPOV	Oxidizer Preburner Oxidizer Valve
Pc	Chamber Pressure
RISC	Reduced Instruction Set Computer
RTU	Real Time UNIX
SAFD	System for Anomaly and Failure Detection
SIU	Signal Interface Unit
SSME	Space Shuttle Main Engine
SSMEC	Space Shuttle Main Engine
STA	Special Task Assignment
TDT	Turbine Discharge Temperature
TTB	Technology Test Bed
UTRC	United Technologies Research Center
VDT	Vehicle Data Table