NASA Technical Memorandum 87638

User's Guide to Programming Fault Injection and Data Acquisition in the SIFT Environment

C. R. Elks, D. F. Green, and D. L. Palumbo

January 1987

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

1.0 INTRODUCTION

1.1 Purpose and Scope

This document describes the features, command language, and functional design of the Software Implemented Fault Tolerance (SIFT) fault injection and data acquisition systems in detail. It is also intended to assist and guide the SIFT user in defining, developing, and executing SIFT fault injection experiments or other experiments and the subsequent collection and reduction of data obtained from the data acquisition software. This document is intended to be used in conjunction with the SIFT User's Guide (NASA TM-86289) for reference to SIFT system commands, procedures, functions, and overall guidance in SIFT system programming.

1.2 Background and Description

SIFT is an experimental multi-computer system designed to provide ultra-high reliable computing service in applications where uninterruptable system performance is critical. Research in fault-tolerant computing concepts utilizing the SIFT computer system is performed at the NASA Langley Research Center's Avionic Integration Research Laboratory (AIRLAB). Vital to verifying and validating a given fault-tolerant computer system (like SIFT) is the concept of in-circuit fault injection. In-circuit fault injection allows the designer or researcher to examine system parameters such as: fault-handling behavior (i.e. fault detection, isolation, and reconfiguration), performability under fault conditions, survivability of the system in the presence of faults, and fault coverage for the system or a particular subsystem. The SIFT fault injection and data acquisition environments support the user in creating, modifying, and executing fault injection or other experiments via command driven interfaces implemented on the host environment (VAX-11/750). In particular, the SIFT Fault Injection System (FIS) furnishes the user with the

flexibility to vary fault injection parameters such as: static fault insertion (stuck at one, stuck at zero), transient fault insertion, rate of occurrence, duration of fault waveform, double fault insertion, etc. On the other hand, the SIFT Data Acquisition System (DAS) provides the capability to extract, collect, and analyze selected real-time data from the SIFT computer system during fault injection or fault-free operation. Together, the two diagnostic research support systems provide an effective work station environment for preparing, executing, analyzing, and cataloging a wide range of data from the SIFT computer system.

Fault injection, data acquisition and software development for the SIFT computer is accomplished using a DEC VAX-11/750 (AIRLAB System 9) as the host software facility. Communication between the VAX-11 computer and SIFT computer is facilitated through a VAX-11 resident interface called the SIFT Host Environment (SIFT HEVN). The programming concept for SIFT HEVN is to provide a separate DEC Command Language (DCL) command (e.g. a separate task program) for each SIFT function. This programming approach allows the SIFT system user to execute all VAX-11 DCL commands while in the SIFT environment and in addition, provides the user with the flexibility and control to modify and/or add SIFT functions for unique user applications and future SIFT programming requirements. Entry into the SIFT Host Environment requires only a SIFT account on System #9 and a VT-100 compatible terminal. Multi-user capability can be utilized for many of the SIFT programming features; however, the FIS and DAS features are SINGLE USER only.

The main section of this guide describes the FIS and DAS command languages which are used to define, develop, and execute SIFT fault injection experiments.

Appendix A describes the DAS user interface WINDOW. The WINDOW utility allows the user to define the conditions for SIFT to VAX data transfers. This appendix describes the procedures, command language, and capabilities of the WINDOW interface.

Appendix B describes the features and operations of the SIFT global clock. The SIFT Global is a common time base for the SIFT computer system and the VAX/11-750 host station.

The user should be advised that because SIFT is a dynamically evolving system, modifications to exiting functions/commands or new procedures can be expected and may not be reflected in this guide. Current information can be obtained through the VAX-11/VMS help facility which will be updated to reflect the latest changes.

1.3 Conventions Used in this Document

CONVENTION	MEANING
USER SUPPLIED	Underlining indicates command or data is to be supplied by user
( )	Parenthesis indicate optional parameters associated with command arguments
" "	Double quotes contain system supplied responses, prompts, cursors, and error messages
$	VAX-11 DCL command prompt

SIFT$	SIFT Interface program prompt
*	Editing prompt for FIS environment
COMM*AND	The asterisk indicates command can be truncated to first four letters.

2.0 SIFT FAULT INJECTION SYSTEM

2.1 Introduction

There are two methods for conducting fault injection tests (i.e., entering fault input parameters and injecting the fault using those parameters) in the SIFT Fault Injection System. The first method described is the injection of faults interactively from input at the terminal. This method allows the user to execute one type of fault at a time for as many iterations of that fault as desired. The fault type and parameters can be varied interactively for desired results. The second method is to inject faults using a file to store sequences of faults which constitute a complete fault inject test plan. This method is primarily used for large tests that can be executed automatically from beginning to end with little or no user intervention. Test plans must be developed using the Automated Research and Test System (ARTS), and detailed procedures are contained in the ARTS User's Guide. The following sections describe in detail the procedures for executing fault injections interactively and from a test plan.

2.1.1 Entering the SIFT Environment

To enter in the SIFT interface environment from a VT-100 type terminal, the user can type SIFT at the VAX-11 command prompt "$". The system will

respond by replacing the VAX-11 command prompt "$", with the SIFT prompt "SIFT$". This starts the SIFT session and the display of the SIFT computer system status. The SIFT display (see Figure 1) will appear at the top of the screen showing current processors not allocated to other users. Processor numbers already allocated to other users will not appear in the display. Processor status (ARMED, ALLOCATED, SELECTED, HALTED, and RUNNING) is indicated by various combinations of character attributes as described in reference 1.

2.2 **FIS Command**

- **Command format:** FIS (no options)
- **Command level:** Entered at the SIFT users prompt "SIFT$" .
- **Command summary:** The command FIS activates the SIFT Fault Injection System resident on the VAX-11/750. There are no parameters associated with the FIS command. The FIS command is cancelled by issuing the ENDFIS command at the SIFT user prompt "sift$".

Description

After entering the FIS command, the system will issue the message "FIS ACTIVATED" to notify the user that the command was accepted and executed, and the message "FIS:ON" will appear in the sift status display. However, if FIS is currently in use by another user, the system will issue the message "FIS IN USE". In either case, the SIFT prompt "SIFT$" will reappear after the command is executed. To exit a FIS session, the user can or must enter the ENDFIS command at the SIFT prompt "SIFT$". The ENDFIS command will delete any FIS parameters that were created by the user (see FAULT command below) and permit other users to access the FIS environment.

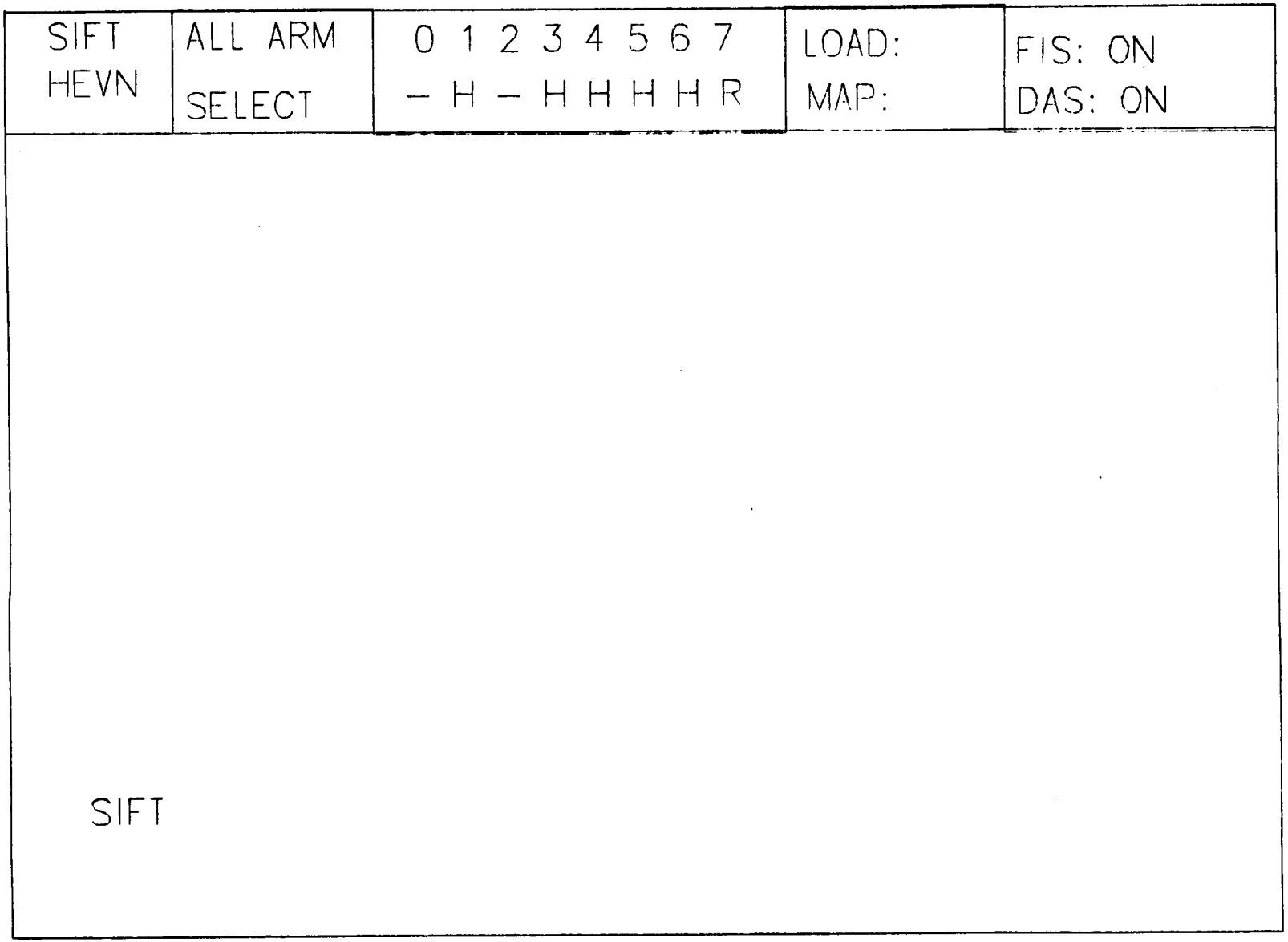


Figure 1 – SIFT Status display

TVGDIR:SIFTDIS.TVG

There are several background processes initiated by invoking FIS commands that are transparent to the SIFT user, such as, creating the FIS global section and initializing the section variables, and subsequently mapping to the SIFT global section and setting the boolean FIS true. When FIS is finished mapping to SIFT, the FIS switch in the SIFT display will be set (on). This Boolean locks out other users from entering the SIFT FIS program.

2.3 Interactive Fault Injections

2.3.1 FAULT Command

- **Command format:** FAULT (no options)
- **Command level:** Entered at the SIFT users prompt "SIFT$".
- **Command summary:** This command activates the process that allows the user to enter fault input parameters from the terminal for a subsequent fault injection. To cancel the fault command the user can enter QUIT at the FIS editing prompt "*". To save fault input parameters, the user can enter EXIT at the FIS editing prompt.

Description

To invoke the FAULT data entry screen, the user must initially invoke FIS as described above, then enter the FAULT command at the "SIFT$" prompt. The system will respond to a FAULT command by graphically displaying the fault input parameters needed for a single fault injection (see Figure 2). The single fault injection display is segmented into different fault input parameter fields (e.g. fault type, processor, board, etc.). Accordingly, each input parameter field is associated with a unique set of keywords that are valid only for the input parameter field they are associated with. For example, (see Figure 2) the input parameter field PROCESSOR has six different keywords that are valid for this field: "P1" (i.e. SIFT Processor 1), "P2",

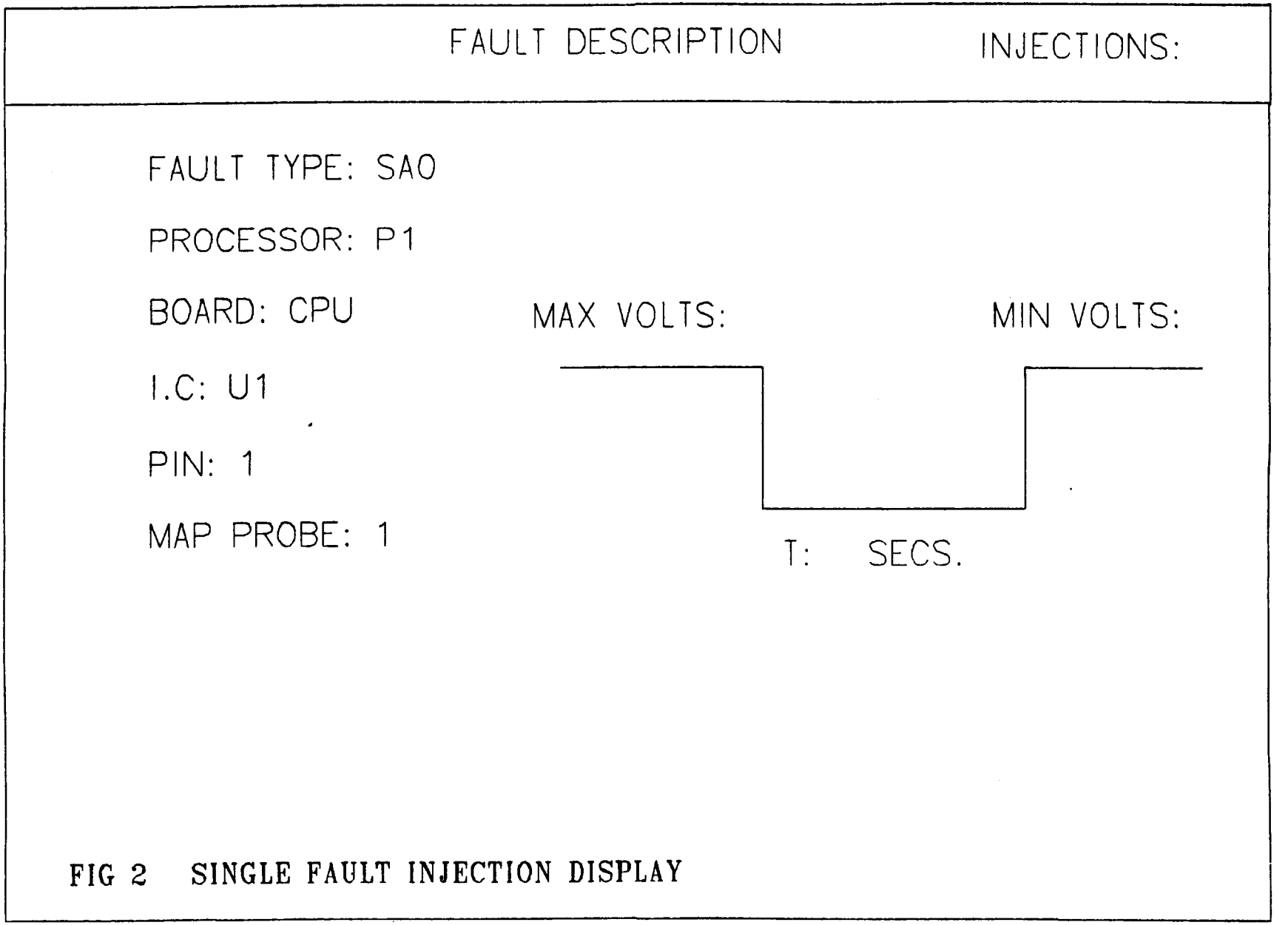


FIG 2 SINGLE FAULT INJECTION DISPLAY

TVGDIR:SINGFLT

"P3", ... and "P6". A description of the fault types and other input parameters with respect to limitations of input parameter sizes and correct format is included in the FAULT INPUT PARAMETER section. Once the FAULT command has been executed, the FIS cursor will automatically be placed at the bottom of the screen awaiting the fault data entry commands (i.e. DEF*INE command, DOU*BLE command, etc.).

2.3.1.1 **DEFINE Command**

- **Command format:** DEF (options: /1 or /2) [parameter field name]

  Options /1 - single fault entry screen

  Options /2 - double fault entry screen

  Default option is /1.

- **Command Level:** Enter command at FIS editing prompt "*".

- **Command summary:** The DEF command, when used in conjunction with the appropriate fault input parameter, allows the user to enter fault injection data into the FIS display.

  For example:

  * DEF TYPE <return> Places the cursor in the field FAULT TYPE and

  accepts one of the following keywords:

SA0 - Stuck at zero.	TR1 - Transient type 1.
SA1 - Stuck at one.	TR2 - Transient type 2.

Description

The DEFINE command executed without a fault input parameter field automatically positions the cursor at the first fault input parameter field (i.e. FAULT TYPE) inside the display for data entry. Pressing a <return> after data entry will move the cursor in the display for the next data entry. Executing a carriage return without entering data will leave the data unchanged

in that parameter field. Entering a "Q" at any parameter field will exit the display and reposition the cursor to the FIS editing prompt "*". Once the user has completed an input parameter edit session, a control Z command entered at the editing prompt "*" will save the FIS edit session and map the input fault injection parameters to the FIS global section for subsequent use by the INJECT command process.

2.3.1.1.1 **Fault Input Parameters**

This section will present and describe the fault input parameters and associated keywords for a single fault and double fault injection display. This section is organized in the following manner: fault parameter field, format, options, and associated keywords.

FAULT TYPE:

Description:

Defines the type of fault to be injected into the circuit. Currently there are four types of faults available:
"Stuck at 0", "Stuck at 1", "Transient 1", and "Transient 2".

Format:TY*PES (no options)

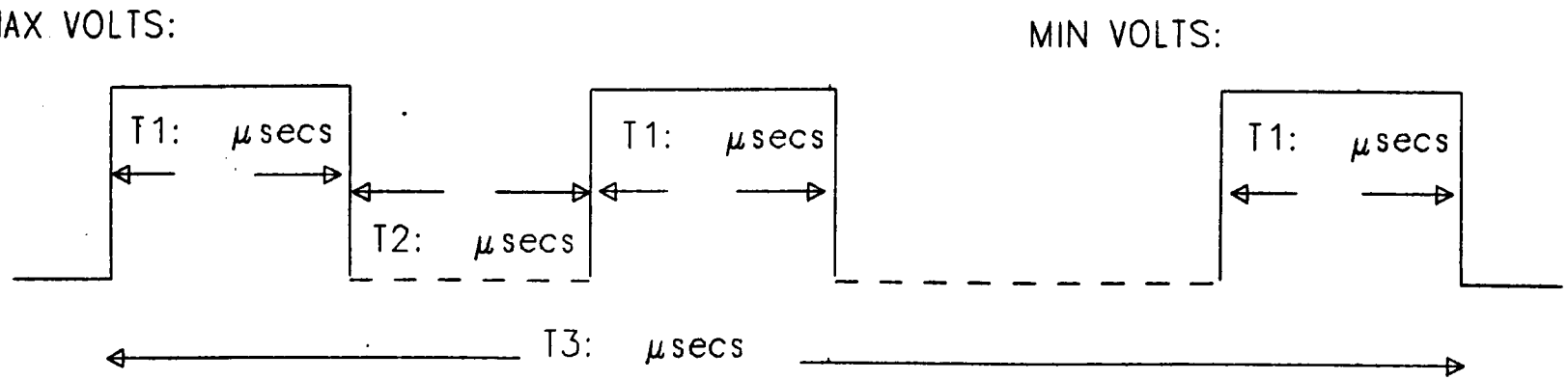
Keywords:

SA0 -"Stuck At 0" - forces the selected fault node to a logic low from 1 millisecond - 600 seconds (user defined).

SA1-"Stuck at 1"- forces the selected fault node to a logic high from 1 milliseconds - 600 seconds (user defined).

TR1-"Transient 1"- injects an intermittent pulsed waveform that can repeat over a time interval specified by the user. The TR1 waveform is in a electrically inactive high impedance state (i.e. tristated) between active pulses. (See Figure 3)

TR2- "Transient 2"- is a continuous waveform (e.g. alternating "stuck at



WHERE: T1 is the active or on-time of the TR1 intermittent fault waveform.

T2 is the in-active or tri-stated time of TR1 intermittent fault waveform

T3 is total elapsed time of TR1 intermittent fault waveform.

FIGURE 3. TRANSIENT 1 FAULT INJECTION WAVEFORM.

TVGDIR:TRANS1

zero", "stuck at one" states) that repeats over time interval specified by user. This transient waveform is electrically active in both state levels.

(See Figure 4).

PROCESSOR

Description:

Defines the SIFT working processor that is targeted for fault injection. No default parameters are associated with this input parameter field.

Format: PROC*ESSOR (no options)

Keywords: P1,P2,P3,P4,P5,P6 - SIFT working processors.

BOARD

Description:

Defines the functional circuit board that faults are to be injected into for a given SIFT processor. No default parameters are associated with this input parameter field.

Format: BOA*RD (no options)

Keywords:

CPU- Central Processing Unit

BI- Bus Interface

MIC- Memory Interface Control

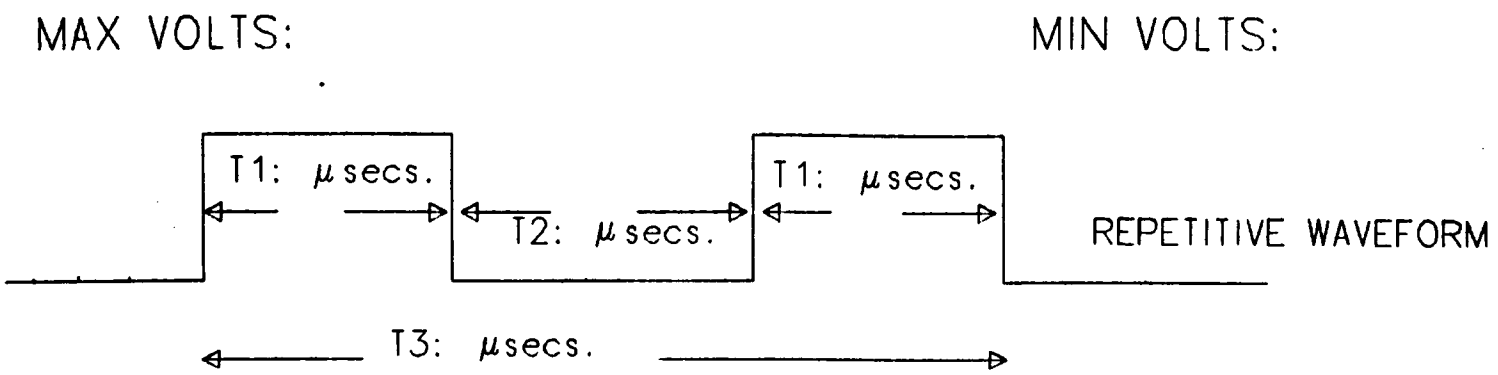
BR- Broadcast Receiver

1553- Avionics Serial Data Port (MIL-STD-1553)

MPM1- Main Processor Memory

MPM2- Main Processor Memory

TC- Timing and Control



WHERE: T1 and T2 designate an interval time that a user defined voltage level will be active for the TR2 continous fault waveform.

Note: the TR2 continous fault waveform has no in-active or tri-stated states, as the case is in the TR1 fault waveform, therefore the TR2 continous fault waveform is fully asserted for the duration of T3.

T3 is total elapsed time of TR2 continous fault waveform.

FIG. 4 TRANSIENT 2 INJECTION WAVEFORM.

TVGDIR:TRANS2

I.C.

Description

The Fault program will accept valid I.C. designations from U1-U99. However, the user should consult Appendix C "SIFT I.C. layouts" to correctly identify board and I.C. No default parameters are associated with this input parameter field.

Format: I.C. (no options)

Keywords: U1 - U99

PIN

Description

Accepts valid pin numbers from 1-99. Again, the user should consult the SIFT users data base for correct pin assignment numbers. No default parameters are associated with this input parameter field.

Format: PIN (no options)

Keywords: 1-99.

MAP PROBE

Description

The SIFT Fault Injector is a physical link to the integrated circuit or P.C. board. At present, there exists eight discrete map probes for in-circuit fault injection. To aid the experimenter in connecting to integrated circuits, each probe can accommodate two connection devices. The first, is a standard ball-clip connector; the second option is a wire-wrap pin socket. In addition to the in-circuit fault connections described above, the map probe requires a power (+5 v) connection and a ground connection from the board receiving the fault injection. (See the AIRLAB In-circuit fault injection users guide for more detail on the SIFT Fault injector operation).

Format: PRO*BE (no options)

Keywords: 1-8.

MAX VOLTS

Description

Defines the maximum voltage the user wants to inject into the circuit. Maximum voltage using the SIFT Fault Injector is 15 volts d.c. It can be incremented in .1v steps from -15Vdc to 15Vdc. Default is 0.0 volts. Note: The user should consult the SIFT data base for information on designated I.C. supply voltages.

Format: MAX*VOLTS (no options)

Keywords: -15.0Vdc to 15.0Vdc at .1Vdc incremental steps

MIN VOLTS

Description

Defines the minimum voltage the user wants to inject into the circuit. Minimum voltage using the SIFT Fault Injector is -15 volts d.c. Can be incremented in .1v steps from 15.0Vdc to -15.0Vdc. Default is 0.0 volts.

Format: MIN*VOLTS (no options)

Keywords: 15.0Vdc to -15Vdc at .1Vdc incremental steps

INJECTIONS

Description

Defines the number of times the fault is to be executed. Default is 1.

Format: Inject*ions

Keywords: 0 to 65536

TIME

Description

Defines the amount of time a FAULT TYPE waveform (i.e. SA0, SA1, TR1, and

TR2) will be activated.

Format: TIME

Keywords: T, T1, T2, T3

T: Defines the amount of time a "Stuck at zero" (SA0) or a "Stuck at one" (SA1) fault injection waveform is activated. T: can be varied from 1 millisecond to 600 seconds in 1 millisecond intervals. (See Figure 2).

T1: Defines the time a voltage pulse (either a negative or positive level) will be active for a TR1 and TR2 fault injection waveform. Can be varied from .3 microseconds to 12.8 microseconds in .1microsecond steps. (See Figures 3 and 4).

T2: Defines the inactive or tristated time between two successive voltage pulses as related to TR1 fault injection waveform. For TR2 fault injection waveform, defines the time the user defined voltage level will be active. Can be varied from .3 microseconds to 12.8 microseconds in .1 microsecond steps. (See Figures 3 and 4).

T3: Defines the total period or life-time of a TR1 and a TR2 fault injection waveform. Can be varied from .3 microseconds to 3200 microseconds in .1 microsecond steps. (See Figures 3 and 4)

TIME OUT

Description

Defines the time out period (in sec) for the fault if no reconfiguration occurs. No further data will be taken after this time.

Format: TIME OUT

Keywords: 0 to 65536 seconds.

DELAY TIME (double fault display only)

Description

Defines the amount of delay between single fault display execution and double fault display execution (i.e.defines the delay time between two sequential faults). NOT IMPLEMENTED.

Format: DELAY TIME

keywords: not avaliable yet

2.3.1.2 **DOUBLE Command**

- **Command format:** DOU (options: /DEL)
- **Command level:** "*" DOU
- **Command summary:** The DOUBLE command invokes the process that draws the double fault screen display (See figure 5). To cancel and erase the double fault screen, the option (/del) is used with DOU command.

Description

Data entry is accomplished by using the DEF/2 command with the appropriate input parameter field at the FIS editing prompt "*". For example:

* DEF/2 TYPE

Places the cursor at the FAULT TYPE input parameter field in the double fault display (assume the DOUBLE command was executed prior to DEF/2 command).

The same input fault parameters as described in Section 2.3.1 for the single fault display are valid for the double fault display. To exit a double fault edit session, the user can execute a control Z or enter an EXIT <return> command sequence. The FIS edit session will be saved and accordingly the fault input parameters will be sent to FIS global variable section for subsequent use. If the user desires to discontinue or erase a double fault screen, the command DOU with the DELETE attribute will erase the double fault screen. Fault input parameters entered into the single fault injection screen will remain intact and undisturbed.

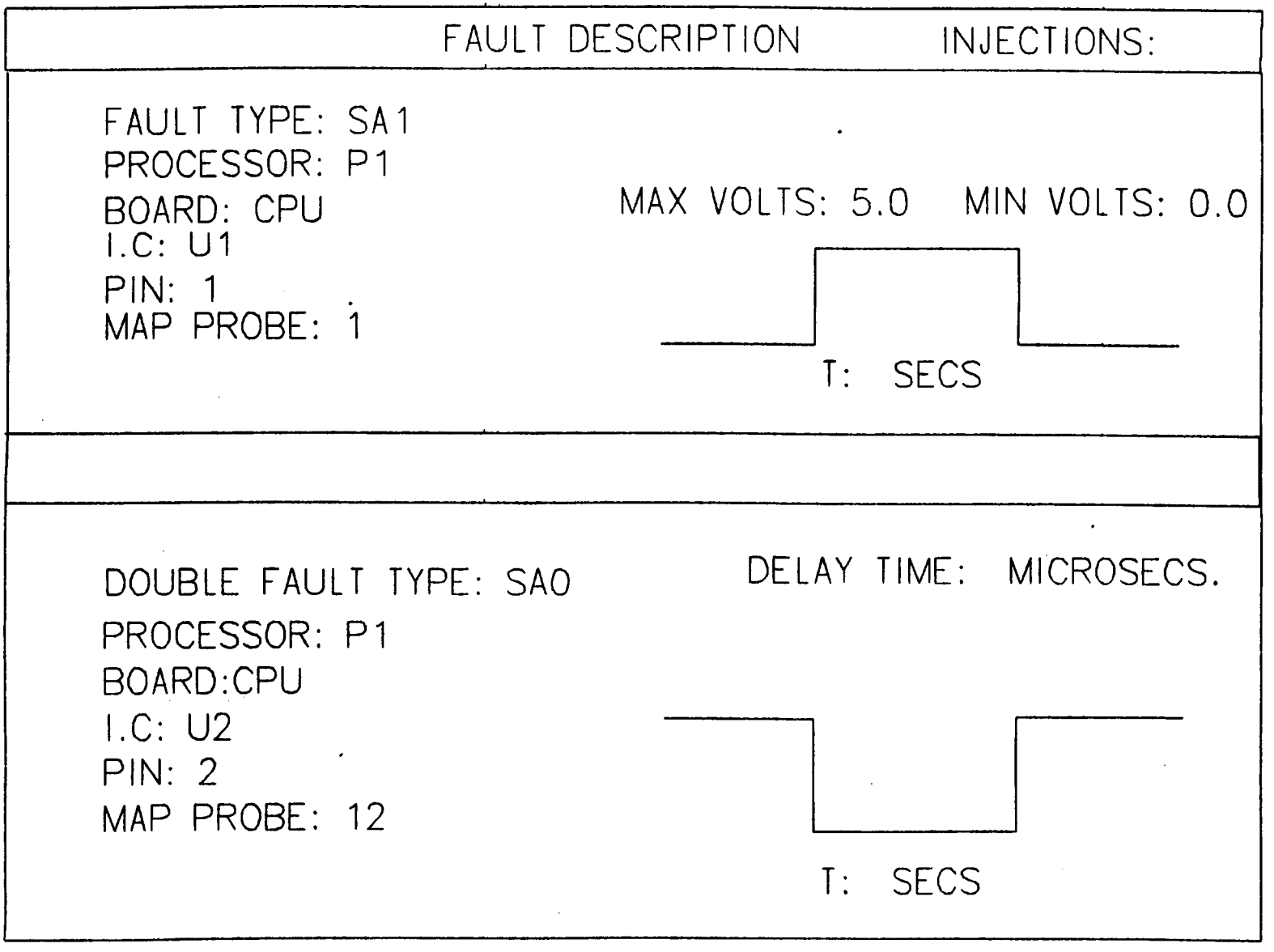


FIG. 5 DOUBLE FAULT INJECTION DISPLAY

TVGDIR:DOUDIS

2.3.1.3 **EXIT Command**

- **Command format:** EXIT or Control Z (no options)
- **Command level:** * EXIT or Control Z
- **Command summary:** The exit command saves the fault injection data into the FIS global section for future fault injections, and returns the user back to SIFT status screen.

Description

The EXIT or Control Z command allows the user to exit from a particular fault injection edit session and subsequently save the edit session parameters for later use (i.e. the fault injection). Note: If the EXIT or Control Z command is given while the double fault is in effect, the system will assume a double fault is being executed. If a single fault is in effect when EXIT or Control Z command is given, the system assumes a single fault is being issued.

2.4 **INJECT Command**

- **Command format:** Inject
- **Command level:** $SIFT Inject
- **Command summary:** The Inject command activates the In-circuit Fault Injector, which subsequently issues a fault to the targeted circuit board. To cancel the Inject command, type a ctrl Y during the Inject process.

Description

The Inject command calls the process (program) that retrieves and checks the fault input parameters from the FIS global section and subsequently sends them to the SIFT Fault Injector for in-circuit fault injection. The INJECT program is integrated with the SIFT data acquisition system (DAS) (see section 3.0). Therefore, FIS and DAS must be activated to use this facility. The in-circuit fault injection will execute and issue the message FAULT INJECTION COMPLETED. After the fault is completed, the program stores the values of the

SIFT global clock at the time the fault was submitted. This clock time is the variable SIFT_global clock in the DAS section which is available for use in processing the results of a fault injection test (see Appendix B on use of the SIFT global clock).

2.5 Fault Injection Test Plan Execution

The following commands are used to execute fault injection test plans: (See the Automated Research and Test System (ARTS) User's Guide for detailed procedures on creating and editing a fault injection test plan.)

2.5.1 **FITM*ENU** [testplan name]

- **Command format:** FITM*ENU
- **Command level:** SIFT$ FITM*ENU
- **Command summary:** Displays a menu of fault injection test (FIT) operations.

Description

When the test plan is first started, the startup menu allows the user to select a particular test unit to execute from the test plan. The selected unit is then initialized into the FIS global section. If this command is invoked after a test plan has already started (i.e., a test unit has already been initialized and has partly or completely finished execution), a menu different from the startup menu will be displayed. This menu includes a number of different operations that are only applicable after a test plan has started such as restarting a test unit at a different fault number or continuing a test from the last position in the file. This menu also displays at the top of the screen the lastest status of test plan execution.

2.5.2 FITC*ONNECT

- **Command format:** FITC*ONNECT
- **Command level:** SIFT$ FITC*ONNECT
- **Command summary:** Displays the map probe connections and SIFT circuit board layouts to assist the user in connecting the correct in-circuit fault injector probes to the SIFT processor hardware.

Description

By default the mapping for the currently initialized test unit will be displayed but other test unit mappings can also be displayed as desired. The program is completely menu driven and allows the user to select circuit board and IC layouts for display on the Megatek graphics monitor. The Megatek displays highlight the particular ICs as they are being connected and also show additional information of value about the ICs.

2.5.3 FITI*NJECT

- **Command format:** FITI*NJECT
- **Command level:** SIFT$ FITI*NJECT
- **Command Summary:** This command initiates the fault injections for the currently initialized test unit.

Description

The FITINJECT command automatically does all the other SIFT or FIS commands necessary to run the test such as enabling data acquisition, starting the SIFT processors, starting the preprocessing program, and reloading faulted processors after an injection. Each fault in the test unit is injected in sequence for the required number of iterations until all faults have been injected. After the test unit is completed, control will return to the user to select another test unit or rerun the previous test unit (done from the FITMENU command). Once a test unit is in progress it can be interrupted with Cntrl y, after which the FITMENU can be invoked to restart or repeat the test unit.

2.6 ENDFIS Command

- **Command format:** ENDFIS
- **Command level:** SIFT$ ENDFIS
- **Command summary:** The ENDFIS Command ends the current FIS session.

The ENDFIS command ends the FIS session by deleting access to the FIS global section. Note: The ENDFIS command deletes all FIS and FAULT input variables in the FIS global section; therefore, all previous user input data describing a fault injection session is lost.

2.7 SUMMARY OF FIS COMMANDS

- FIS - Activates the SIFT fault injection system if not already in use.
- FAULT - Invokes the process that allows the user to enter, edit and list fault injection data for an interactive fault injection session. Inside the fault program commands are:

  DEF[/1 or /2] [field] - Places the cursor at the parameter field specified or first parameter field if no field is specified.

  DOU - Displays the double fault entry screen for subsequent entry of second fault parameters

  DOU/DEL - Deletes the double fault entry screen and returns to single fault entry.

  EXIT or CNTRL Z - saves parameters and exits the fault program.
- INJECT - Collects the data from the FIS global section (i.e., the fault input parameters from the last FAULT command) and sends them to the SIFT fault injector for in-circuit fault injection. (Note - This command also enables data acquisition, starts SIFT, runs the preprocessor and reloads SIFT after the injection. It repeats this sequence for the number of injections specified in the FAULT program.)
- FITMENU - Displays a menu of operations for execution of a fault injection test plan.

- FITCONNECT - Displays mapping for the in-circuit fault injector probes and SIFT circuit board diagrams for making probe connections to the SIFT hardware.
- FITINJECT - Does the fault injections from the test plan.
- ENDFIS - Deactivates FIS and deletes all global variables/parameters used during the FIS session.

3.0 SIFT DATA ACQUISITION SYSTEM

3.1 Introduction

The purpose of the SIFT Data Acquisition System (DAS) is to extract, collect, and analyze selected real time data from the SIFT computer system. This process is accomplished in three main phases. The first phase is defining the specific data objects to be extracted and the conditions for sending the data (called the filter). The second phase is setting up DAS, enabling acquisition, and capturing the data. The final phase is preprocessing the data which includes reading, processing, and/or saving the data for further analysis. The phases of DAS are integrated and coordinated through VAX global data sections and interprocess communications. A thorough understanding of these phases and their interelationships is essential to have effective use of DAS for SIFT experiments. This section describes each phase of data acquisition in detail and also describes the window utility used for defining DAS requirements.

3.2 Defining Output Data and the Filter

The specific data objects that DAS is to output must be defined and made known to the system, and the conditions (filter) for sending the data must also be defined. The method for accomplishing these tasks is the window utility which places the information in a file called the window file. The details of

how to use the window utility are described fully in Appendix B; however, the general concepts of defining output data and filter are explained below.

3.2.1 Data Definition

There are four basic types of data objects used in DAS - variables, hooks, memory addresses, and permanent system variables. DAS allows up to 16 of these data objects to be defined. The entire list of objects make up what is called the window. There are five permanent system variables leaving the user with zero to 11 data objects which can be defined as any combination of variables, hooks, and memory addresses. These types and the permanent system variables are described below.

Variables: Variables are those names or labels used in the modules of the SIFT operating system and user programs that are linked with the operating system. At link time the variable names used in these modules are mapped to a specific memory location in SIFT. If a variable name is defined in the window, the contents of this memory location will be output. In order to use a variable name in the window the variable must be listed in the program map. In assembly programs, all labeled variables are accessible in the program map. In Pascal programs, only those variables declared in the outermost scope (main program level) are accessible. In addition to the operating system variables available in the VAR section of the SIFT operating system program, several frequently used variables are declared in the source file GLOBALS.SR located in the SIFT directory (logical name SIFTDIR). These variable names can be included in the window only if the object file SIFTDIR:GLOBALS.RX is linked with the operating system and user programs.

Hooks: One group of special purpose variables listed in GLOBALS.SR is used frequently in DAS - the hook variables (or just hooks). The hooks are HOOK1, HOOK2,...HOOK9, HOOKA, HOOKB; and one or more of these hooks can be

defined in the window. The purpose of a hook is to provide a convenient method of accessing local variables (i.e., not declared in the outermost scope) used in SIFT programs. For example, if the variable TIME is defined locally in a Pascal program and this program is linked with the operating system, the variable TIME will not be mapped to a specific hex address and, thus, cannot be used in the window. However, if the program contains a statement that assigns the value of TIME to one of the hooks (e.g., HOOK1 := TIME), then the variable HOOK1 can be used in the DAS window which will output the value of TIME. An alternative is to define TIME in the Pascal program to be at a specific address and use this hex address in the window. Another alternative is to declare the variable at the outermost level in the Pascal program and use that same name in the window.

Hex addresses: This type of data object is self-explanatory. If used as an entry into the DAS window, the value that is stored at the hex location will be output by the system.

Permanent system variables: The five permanent system variables in the window are: PID (processor ID), TASKID (task ID), PRESENTCONFIG (present configuration), GFRAME (global frame count), SFCOUNT (subframe count). These variables are always located in the order given and in the last five positions of the window. They cannot be changed by the user, but the data from these variables can be filtered (see filter definition below) and accessed during preprocessing (see Section 3.4).

Declaration files: If a hook is referenced in a Pascal program (e.g., HOOK1 := TIME), the hook must also be declared in the Pascal program. To declare one or more hooks, the program must contain the statement INCLUDE 'SIFTDIR:HOOKSDEC.GLO'. HOOKSDEC.GLO is a file that contains the declarations for all the hooks, and these declarations correspond to the hook definitions

contained in GLOBALS.SR. If a non-hook variable listed in GLOBALS.SR is referenced in a Pascal program, the statement INCLUDE 'SIFTDIR:SIFTDEC.GLO' must be used. If both hooks and non-hook variables listed in GLOBALS.SR are referenced in the program, then both files (HOOKSDEC.GLO and SIFTDEC.GLO) must be included in separate INCLUDE statements.

3.2.2 Filter Definition

Once the window has been defined the conditions for sending the data must be established. The set of these conditions is called the filter. The filter is composed of two main parts. The first part, the filter status, defines when data is to be sent based on actual values of the data objects in the window. The second part, the termination status, defines how data acquisition is to be terminated.

Filter status: There are three choices for the filter status: 1) send data only when there is a reconfiguration of SIFT processors, 2) send data all the time (unconditional - filter disabled), or 3) send data based on the value of specific data objects in the window or when there is a reconfiguration. These options are selected using the window utility and invoking the SET FILTER function at the main menu as described in Appendix B. The first two choices are self-explanatory but the third needs more explanation. Under this third option the user may qualify from one to six data objects in the window with a relational operator (called the qualifier or qualifier code) and a qualifier value. The filter algorithm compares the value of the window data object to the qualifier value based on the qualifier code selected (e.g., equal, less than, greater than, etc.). If the result of this relational expression is true, data is sent. If more than one data object has been qualified, the data will be sent when any qualifier is found to be true (i.e., the results of qualifiers are ORed). An example might be to establish a qualifier for the

variable TASKID so that data is sent only when TASKID = 3. Also, a qualifier could be defined for ERRORS > 0 so that data will be sent when either TASKID = 3 OR ERRORS > 0. (There is no provision for ANDing qualifiers so that data is sent only when the expression, TASKID = 3 AND ERRORS > 0, is true.) Also note that data will always be sent when there is a processor reconfiguration, regardless of the result of qualifier conditions established in this option.

Termination status: There are basically two choices for termination status: 1) terminate data after a reconfiguration, and 2) do not terminate data after a reconfiguration but continue data acquisition indefinitely. These options are made using the Window Utility and invoking the SET TERMINATION function at the main menu. The first choice means that data will be sent during the subframe that reconfiguration occurs and will continue to be sent (without filtering) for a set number of subframes after which data will stop. The number of subframes to continue data after reconfiguration is called the delay, and this number may be defined by the user to be zero or any positive integer. The default delay is 10 subframes, but can be changed in the window utility. The second option means that data will be sent during the subframe that reconfiguration occurs, but data acquisition will continue indefinitely according to whatever filter conditions have been established. Therefore, under this option data will not be sent automatically for a certain number of subframes after a reconfiguration, but will be filtered just as before the reconfiguration occurred. This option allows continuous operation of data acquisition for extended periods without being effected by processor reconfigurations.

3.3 Setting Up DAS, Enabling Acquisition and Capturing Data

The DAS environment must be properly set up and enabled in order to capture data for SIFT experiments. The setting up process involves several

steps: activating DAS, allocating and loading SIFT processors, loading the window file, and enabling data acquisition.

3.3.1 Activating DAS

The first step is to activate DAS within the SIFT environment. To activate DAS the user must already be in SIFT and type 'DAS' at the SIFT$ prompt (i.e., DAS is a SIFT command and is unknown outside the SIFT environment). However, data acquisition is a single user system. If another user has already activated DAS the second user gets a 'DAS IN USE' message and must wait until the other user is finished. Once activated, DAS runs as a subprocess in parallel with SIFT, and the SIFT display will show the message--DAS ON. The DAS command also does some functions that are transparent to the user but are nevertheless important for a complete understanding of the system. One of the most important of these functions is to create and initialize a DAS global section that is used to store information needed for acquisition and to share data between various DAS programs. Another important function is to load the SIFT processor P7 with a program (called RELAY) that filters and sends the data to the VAX. Processor P7 becomes allocated to DAS and cannot be used for any other purpose while DAS is activated. Another function done automatically by the DAS command is to mount the disk drive DMA3. This is the drive used by DAS to store all data transferred during one acquisition. Under normal operating conditions, the message indicating that DMA3 has been mounted will be displayed on the screen. However, it is possible that a previous error condition or abnormal exit from DAS will cause DMA3 not to be mounted automatically. In this case, the DAS command will display a message that requests the operator to mount the disk and will wait for the disk to be mounted before continuing to execute the DAS command. If this situation should occur, see the operator, SIFT system manager, or a systems programmer for

assistance. After DAS has been activated (shown by the DAS ON message in the SIFT display), the other DAS commands become available for use. All the DAS commands are listed in Section 3.5.

3.3.2 Allocating and Loading SIFT Processors

This step can be done before or after DAS is activated, but it must be done before loading the window file. The commands used for allocating (ALP) and loading (LOAD) are described fully in the SIFT User's Guide (NASA TM-86289). Any combination of available processors P1 through P6 can be allocated for use during data acquisition. These processors must be loaded with the SIFT operating system that has been properly linked with the user's application programs and schedule table; and the map option must have been used during the link operation (see SIFTLNK command in the SIFT User's Guide). The map option during linking produces a map file with the extension .CRF which is used to locate symbolic addresses when the window file is loaded. One point to be aware of when allocating SIFT processors for data acquisition is that there will be no reconfigurations if fewer than three processors are selected. Therefore, with fewer than three processors, data acquisition will only terminate when a terminating condition other than reconfiguration occurs (see paragraph below on terminating data).

3.3.3 Loading the Window File

After DAS is activated and the SIFT processors are allocated and loaded, the window file must be loaded. This is accomplished using the window utility. Type WINDOW and select the load option in the main menu, or type WINDOW/LOAD filename. The second method bypasses the menu and loads the window file directly. The SIFT processors that are loaded will be the processors that are selected at the time the load is done. Normally, the window file has to be loaded only once for a data acquisition session. However, the window file will

have to be reloaded if any of the following situations occur: (1) if changes are made to the allocated/selected processors; (2) if a new/revised window file is used; (3) if any of the SIFT processors are reloaded; (4) if DAS is ended with ENDDAS command and reentered at a later time. (Note - the commands INJECT AND FITINJECT automatically take care of reloading the window file on a faulted processor after the injection is done.)

3.3.4 Enabling Acquisition

After loading the window file, data acquisition can be enabled with the ENABLE [timeout] command. This command enables the system to receive data by starting the hardware and software components that work together in transferring data from the SIFT processors to the VAX. The timeout parameter given with the ENABLE command sets a length of time (in seconds) that data will be received. If no timeout value is given, the initial default is set to 10 seconds. If a timeout is given in the command, that value becomes the default for subsequent ENABLE commands. The minimum time that can be given is 0 seconds and the maximum time is 65,534 seconds (approximately 8 hours). A zero timeout means that timeout is disabled (i.e., timeout will not occur). Acquisition starts when the ENABLE [timeout] command is invoked and ends when the timeout period (if not zero) or another terminating condition occurs (see paragraph 3.3.6 below on terminating data). The acquisition period started by the ENABLE command does not imply continuous receipt of data. No data will be delivered until after the SIFT processors are started; also the actual amount of data captured can vary from 0 to 50,000 blocks on the DMA3 disk depending on the filter conditions established (see Section 3.2), the number of processors used, and the number of variables in the window. This subject is covered in more detail in the next paragraph and in Section 3.4 under structure of the data. After data acquisition has been enabled (i.e., started) with the ENABLE

command, it can be aborted using the ABORT command which stops the current acquisition. This command might be used if a zero timeout or long timeout period is given in the ENABLE command but a problem occurs and acquisition must be discontinued.

3.3.5 Capturing Data

To start capturing data after DAS has been enabled, the SIFT processors must be started with the STARTSIFT 100 command. The parameter 100 is the hex address of the starting program counter (PC) for the SIFT operating system. While the SIFT processors are running, the SIFT operating system tasks and application tasks run in predetermined interrupt periods called subframes. These subframes and the particular tasks that are scheduled to run during each subframe are defined by the scheduler (see SIFT User's Guide, Appendix A). The important point to remember for data acquisition is that values for data objects in the window are broadcast by the SIFT processors at the BEGINNING of EVERY subframe. The values transmitted are those that were valid after termination of the previous subframe. Because a subframe can be as small as 1.6 msec duration, the capability exists to transmit a considerable amount of data at very high speeds. The actual amount of data transmitted to the VAX, however, can be significantly limited and controlled using the filter mechanism discussed in Section 3.2. The filtering is done by a program (called RELAY) running in processor P7 that examines the data sent by the SIFT processors every subframe. This program either sends the data for that subframe or does not send the data depending on the results of filtering. If the filter is disabled (an option selected in the window utility), all the data will be sent automatically every subframe which will rapidly fill up the DMA3 disk. With no filtering and maximum output (1.6 msec subframes, 16 variables in the window, and 6 processors in the SIFT configuration), the disk will fill up (50,000 blocks) in approximately 4 minutes.

3.3.6 Terminating Data

Data acquisition will be terminated when any of the conditions listed below occur. Note that conditions 1 - 6 do not terminate the activation of DAS (done by the ENDDAS command), but only terminate the current acquisition started by the ENABLE command.

1) The timeout occurs (value in seconds given in the ENABLE command).
2) A reconfiguration of SIFT processors occurs (if filter termination status is set to stop data after reconfiguration).
3) An error condition is detected by DAS.
4) The DMA3 disk fills up.
5) An error condition or timeout occurs from a fault injection test.
6) The user aborts acquisition by invoking the ABORT command.
7) The user does an ENDDAS or ENDSIFT command while acquisition is still in progress.

The user will be notified of termination by a message to the screen that gives the reason for termination and also the amount of data transferred. In the case of an error condition some diagnostic information will also be given. The amount of data transferred is given by two numbers - the number of full buffers and the word count. A buffer is 16,128 words. The word count is the number of words left after the last full buffer. Thus, the total words transferred is the number of full buffers times 16,128 plus the word count. All of this data is located on DMA3 starting at block 1; this data must be preprocessed and results saved in a permanent file or the data will be lost.

3.4 Preprocessing the Data

The final phase in data acquisition is to read the data from the disk DMA3, check and/or manipulate the data (i.e., preprocess), and save the raw data or results of preprocessing in a permanent file. These functions are

accomplished by a user written program constructed from a template provided in the Pascal source code file named PREPRO.PAS. The template program, located in SIFTDIR, provides the data structures and procedure for retrieving data from the DMA3. The user must edit this file to add preprocessing functions which are unique to his or her experiment. After editing the file, the program must be compiled and linked using a command procedure PREPRO.COM. The user's preprocessing program can then be run after acquisition has been enabled and SIFT processors have been started. The program will wait until data acquisition completes and then do the preprocessing. A point to remember is that the preprocessing program must be run before the next data acquisition begins. This is necessary because each acquisition starts writing at block 1 on the disk, causing any previous acquisition data to be overwritten. If the experiment requires doing repeated data acquisitions in a loop (i.e., enabling acquisition, waiting for DAS completion, preprocessing, enabling again, preprocessing again, etc.), the data must be preprocessed during each iteration of the loop. After the entire experiment and all acquisitions are completed, only the data saved during preprocessing will be available for post processing analysis.

3.4.1 Structure of the Data

The structure of the data transferred from the SIFT processors is the same for every acquisition, but the length of the structure is determined by the number of processors in the starting SIFT configuration and the number of data objects in the window. The output data structure for each subframe is described as follows:

```
                subframe count
                current processor configuration (voted value)

        first processor in starting configuration
                first user defined data object in window
                second user defined data object in window
                  .
                  .   (additional user defined objects up to 11)
                  .
                PID              (permanent system variables)
                TASKID
                PRESENTCONFIG
                GFRAME
                SFCOUNT

         second processor in starting configuration
                first user defined data object in window
                second user defined data object in window
                  .
                  .   (additional user defined objects up to 11)
                  .
                PID              (permanent system variables)
                TASKED
                PRESENTCONFIG
                GFRAME
                SFCOUNT

            .
            .
            .

         last processor in starting configuration
                first user defined data object in window
                second user defined data object in window
                  .
                  .   (additional user defined objects up to 11)
                  .
                PID              (permanent system variables)
                TASKID
                PRESENTCONFIG
                GFRAME
                SFCOUNT
```

Each data value output is a 16 bit word and the number of words transmitted for a single subframe in SIFT is called a subframe record (see data type SUBFRAMEREC below). The first word (SF in SUBFRAMEREC) is the subframe count, a number from 0 to 26. The second word (PCONFG in SUBFRAMEREC) is the present configuration which is a value that has been voted

from each processor's copy of the PRESENTCONFIG variable. The bits 0, 1, 2, 3, 4, 5 of PCONFG represent processors P1, P2, P3, P4, P5, P6. If a bit is 0 the processor is in the configuration. If a bit is 1 that processor is not in the configuration. For example, when PCONFG = 00F1, it means that P2, P3 and P4 are in the configuration. After SF and PCONFG the remaining words are the window values from each processor in the starting configuration. Each processor outputs its own copy of the values for each data object (variable, hook or hex location) in the window. This is an array of arrays, or a two-dimensional array of window values. The total length of this structure depends on the number of processors and the length of the window (number of data objects). The smallest length possible is 7 words (1 processor times 5 window values plus 2). The maximum length is 98 words (6 processors times 16 window values plus 2). The Pascal data type declarations used for this structure in the template file PREPRO.PAS are as follows:

```
TYPE
  $WORD = [WORD] 32768..32767;    {defined in INCLUDE file SIFTDEC.PAS}

  {Names used in the user's window file}
  WINDOWVARS = (HOOK1, HOOK2, HOOK3, HOOK4, HOOK5, ERROR1, ERROR2,
                ERROR3, ERROR4, ERROR5, ERROR6, PID, TASKID, PCONFIG,
                GFRAME, SFCNT);

  SUBFRAMEREC = [ALIGNED(1)] PACKED RECORD
                SF:  $WORD;
                PCONFG:  $WORD;
                PROCDAT:  PACKED ARRAY [P1..P6, WINDOWVARS] OF $WORD;

VAR
  SFDAT:  SUBFRAMEREC
```

WINDOWVARS is a Pascal enumerated type that defines the structure and data objects in the user's window as described in Section 3.4.1. The user edits the template file PREPRO.PAS to define WINDOWVARS by listing the names of the window data objects in the same order they are listed in the window file. The names should correspond closely with the names used in the window

The names should correspond closely with the names used in the window file, but do not necessarily have to be the same. In fact, if a hex location is used in the window file it will not have a name, so a name representing the value in that location should be used in the type WINDOWVARS. The names will be used to access the subframe record to read a particular window variable for a particular processor. The subframe record is defined by type SUBFRAMEREC, which is a record that corresponds to the data structure described previously. The first and second words are the subframe count and present configuration respectively followed by the window variable data for each processor P1 through P6 in the starting configuration. Data for each subframe includes data for each processor in the starting configuration even if one or more of those processors has been reconfigured. This insures that the length of the subframe record is the same for the duration of the acquisition.

3.4.2 **Accessing the Subframe Data**

Using the example in 3.4.1 above, the data is accessed as follows: The variable SFDAT is defined in the preprocessing program to be of type SUBFRAMEREC. HOOK3 in processor P5 is then accessed by the variable name SFDAT.PROCDAT [P5, HOOK3]. The present configuration is accessed by the variable name SFDAT.PCONFG. The global frame count in processor P2 is accessed by the variable name SFDAT.PROCDAT [P2, GFRAME]. If a processor is not used in the starting configuration the user should take care not to inadvertently access that processor because the data will either be all zeros or garbage.

3.4.3 **Reading and Preprocessing**

The template file PREPRO.PAS contains a procedure that can be called which returns the data for a single subframe (a single record represented by the variable SFDAT). The user should not alter this procedure in any way

since it is dependent on logical I/O's to the DMA3 disk drive and exact calculations of where a requested record is located on the disk. Once the data is returned in SFDAT the user must write code to access the record as shown above and preprocess it in some way. This preprocessing might be reading the data into user defined variables or arrays in memory, doing computations, and storing results/raw data into a permanent file. After PREPRO.PAS is edited it must be compiled and linked by invoking the command procedure PREPRO.COM (DCL command @PREPRO). This procedure links the user's preprocessing program (named by the user) to other files located in the SIFT directory (logical name SIFTDIR). To perform preprocessing during the experiment, the user must execute the DCL command RUN filename where filename is the name of the executable file for the user's preprocessing program.

3.5 Summary of DAS Commands and Sequence of Operations

The following is a summary of DAS commands described above:

DAS - Activates the SIFT data acquisition system if not already in use.

ENDDAS - Ends a data acquisition session. The DAS global section is destroyed.

WINDOW - Invokes the window utility main menu used for creating, editing, listing, printing or loading a window file and defining the filter.

(Note - window can be used outside of DAS except the LOAD and SETUP TERMINATION functions which require DAS activation before use).

WINDOW/LOAD [window file_name] - Loads a window file directly and bypasses the main menu of the Window Utility. If file name is omitted, the program prompts for the file name. Requires DAS activation.

ENABLE [timeout] - Enables the data acquisition system to accept data. The timeout parameter sets a time limit in seconds for the acquisition if no other terminating condition occurs first. If no

timeout is given, the default timeout is 10 seconds for the first ENABLE and for subsequent ENABLEs it is the last timeout value used. A value of 0 means no timeout.

ABORT - Aborts a data acquisition that was started by the ENABLE command. Note that the ABORT command ends the current acquisition of data; it does not end the activation of DAS which is done by the ENDDAS command.

The sequence of operations for DAS is as follows:

1. Enter SIFT and type DAS to activate the data acquisition system. (Also type FIS to activate the fault injection system if applicable.)
2. Use the window utility to create a window file if not already done.
3. Allocate SIFT processors and load the sift operating system.
4. Load the window file using the window utility.
5. Use the ENABLE [timeout] command to begin an acquisition.
6. Start the SIFT processors (STARTSIFT 100).
7. Wait for the acquisition to complete (timeout, reconfiguration, etc.). Other SIFT or DCL commands may be used while acquisition is in progress.
8. When acquisition completes, execute the user's preprocessing program using the DCL RUN command. If this program is run before DAS completes, the program will be in a wait state and will start the preprocessing when acquisition completes.
9. Repeat 5 through 8 as many times as required. This may be done interactively or in a user written command procedure. (Note: The operations in 5 through 8 are done automatically by the INJECT and FITINJECT commands for injecting faults).

10. When all acquisitions of the test or experiment are finished and all preprocessing as been done, end the DAS session by the ENDDAS command.

The data acquisition system is designed to provide flexibility for many types of tests or experiments using SIFT. For example, fault injection tests are integrated with data acquisition through the INJECT command (see Section 2.0), or transient fault tests can be run over very long periods (weeks or months) by disabling timeouts. The user may want to write a command procedure to execute DAS commands, or may do DAS and SIFT commands interactively. The former method is appropriate if experiments must be done with minimum user intervention. The latter method allows interactive control of experiments and access to all DCL and SIFT commands while acquisitions are in progress.

APPENDIX A

SIFT WINDOW UTILITY

A1. INTRODUCTION

The SIFT WINDOW UTILITY is used to define the output variables for SIFT data acquisition and also to define the filter for those variables (i.e., the conditions under which data will be transferred from the SIFT processors to the VAX). The utility stores this information in a file and also provides the capability to edit the file, list the file on the screen, print the file and load the information in the file to the SIFT processors. The utility is invoked by the command WINDOW. A menu will appear on the screen to select one of the window functions: CREATE, EDIT, LOAD, LIST, PRINT, SETUP FILTER, SETUP TERMINATION, SETUP PREPROCESSOR. The following sections describe these operations.

A2. CREATE

This function creates the window file that stores the information needed for data acquisition. The information is organized as a list of variables and/or SIFT processor memory locations which are defined for SIFT output plus additional information about each variable/location. The entire list is called the window, and each variable entry in the list is called a pane. Up to 16 variables or memory locations can be defined for the window. Five of those variables are predefined by the data acquisition system (DAS) and the remaining eleven are selected by the user. The user will be prompted for all information (no commands to remember) starting with PANE 1 through PANE 11. (Note: PANES 12 - 16 already contain the predefined variables explained below).

The first information to enter for each pane is the name of a SIFT variable or a hex address. At the top of the screen there is a list of variables called the default window variables. To facilitate data entry the default variable may be selected for a particular pane number by pressing <RETURN> or by entering an asterisk (*). If a variable name is entered it is divided into two sections, the scope and the label, which are separated with a period (.). The entire name always starts with a dollar sign ($) to distinguish it from a hex address (e.g., $GLOBA.HOOK1, where GLOBA is the scope and HOOK1 is the label). The scope must be 1 to 5 characters; the scope and label combined can be up to 18 characters long but the scope plus the first five characters of the label portion must be a unique name. (Note: rules for labels are the same as rules for referencing labels in other SIFT commands - see SIFT USER'S GUIDE.) The asterisk can be used to enter a default scope or a default label. For example, if the default is $SIFTO.ERROR+2, you can enter *.ERROR+5 and the result will be $SIFTO.ERROR+5. To enter a hex address, simply enter the hex number instead of a label or default label. If no label or address is desired for a particular pane, enter NIL and this will be considered an unused pane in the overall window. Each name, hex address, or NIL will be shown in the center of the screen after it is entered.

The next information to enter for each pane depends on whether or not the information previously entered was a label or a hex address. If a hex address, then the user will be prompted for a text description or comment that describes that data. The description is limited to 30 characters. If the previous entry was a label but not a hook (see Section 3.2 for description of hook), then the next prompt will be for a description just as for a hex address. If the label previously entered was a hook, then the program will

prompt for a task name where that hook is used and a corresponding description. Up to eight task names and corresponding descriptions can be entered for each hook used. Up to 11 hooks can be defined by the user in writing SIFT application or operating system tasks and are referenced by $GLOBA.HOOK1, $GLOBA.HOOK2,..., $GLOBA.HOOK9, $GLOBA.HOOKA, $GLOBA.HOOKB. (For a definition of hooks see para 3.2.1.)

After all information has been entered for a pane, the user will be asked if changes or corrections are desired. Press <RETURN> for no or Y for yes. If yes, the edit screen will appear which allows changing any part or all of the entries made for that pane. The edit function is described in more detail in a later section. The user can make any changes desired and return to the CREATE operation by pressing Cntrl Z. Changes can only be made to the pane just entered. Previous panes cannot be changed in this mode but the entire file can be edited after it is created using the EDIT function in the main menu.

The five predefined variables are located in pane 12 through pane 16 and cannot be changed. The definitions of these variables are as follows:

PANE 12:	$GLOBA.PID	processor ID (e.g., 1,2,..6)
PANE 13:	$SIFTO.TASKID	task ID
PANE 14:	$SIFTO.PRESENTCONFIG	present processor configuration
PANE 15:	$GLOBA.GFRAME	global frame count
PANE 16:	$GLOBA.SFCOUNT	subframe count

After all information for panes 1-11 have been entered, the SETUP FILTER function will be invoked automatically in order to obtain information for filtering the data. Filtering is defined as those conditions which will cause the data for each variable in the window to be transferred from the SIFT computer to the VAX. This function is explained in detail in the next

section. The SETUP FILTER function can also be invoked from the main menu to change filter conditions in a previously created window file.

A3. SETUP FILTER

This function sets up the conditions under which SIFT data will be transferred to the VAX for subsequent processing. It is invoked automatically from the CREATE function described above to establish the filter for a newly created window file, or can be invoked from the main menu to make changes in a previously created window file. The function is completely menu driven.

The first menu that will appear in this function is to select one of three overall data acquisition modes: 1) send data only when there is a reconfiguration of SIFT processors, 2) send all data every subframe whether reconfiguration or not, or 3) send data according to variable qualifiers or reconfiguration. Options 1 and 2 transfer data for every variable/location in the window regardless of the value of any variable/location. Option 3 invokes another input screen which allows the user to enter qualifiers and qualifier values for any of the non-nil variables. A qualifier is a relational operator such as equal to (=), less than (<), greater than or equal to (>=), etc. A qualifier value is a number entered in hex (the default) or entered in decimal by preceding the number with %D. During data acquisition, the variable is compared with the qualifier value using the relational qualifier. If the result of any comparison is true, the data for all variables in the window will be transferred. If no comparison is true, data will not be transferred unless there is reconfiguration. If there is reconfiguration, data is transferred regardless of qualifiers.

Qualifiers are defined by selecting a pane number at the prompt, then entering a qualifier code (-1, 0, 1, 2, 3, 4, 5) associated with the qualifier as defined in the screen display. For example, the code 0 is associated with

the equal qualifier. The code -1 is used to delete existing qualifiers which may then be redefined using one of the codes 0 through 5. After the pane number and qualifier code has been entered, there will be a prompt for the qualifier value. The qualifier and value (in hex) is immediately shown in the display at the top of the screen. Pressing Control Z terminates the session and returns to the main menu.

Note: Changes made to a previously created window file using the SETUP FILTER function are always made to the same version of the file. Only one version of the file is maintained. If more than one version is needed, each version must be given a different name.

A4. SETUP TERMINATION

This function allows the user to override the defaults for 1) terminating data after a reconfiguration of SIFT processors, and 2) maximum subframe number in a frame. The defaults are only overridden for the current DAS session. That is, if the user exits DAS and comes back in, the defaults will again become effective.

In the default mode, if a reconfiguration occurs data will continue to be transferred for all variables/memory locations in the window for the next 10 subframes, after which data acquisition will terminate. The filter is not used to transfer these last 10 subframes after the reconfiguration; this data is sent regardless of what qualifiers have been established. Data acquisition stops after 10 subframes are sent. However, there are two options the user can choose to override this default. The user can change the number of subframes (from 0 to 65535) to continue sending data after reconfiguration, or, there is the option to continue acquisition indefinitely after a reconfiguration. For the second option data will be sent during the subframe that reconfiguration occurs, but in subsequent subframes the data is sent only if specified filter conditions are met.

The second function of SETUP TERMINATION is to change the default for the maximum subframe number in a frame. Currently, the default is 26. (Note: there are 27 subframes in a frame starting at 0 subframe through 26). This default can be overridden for the current DAS session by entering another max subframe number at the prompt. However, if a number other than the default 26 is used, the user must also change the constant MAXSUBFRAME in the file SIFTDEC.CON located in SIFTDIR, and must have made appropriate modifications to the SIFT operating system and scheduler.

Note: The SETUP TERMINATION function has no effect on any window file. Changes made using this function are only made to the DAS global section variables and are not part of any window file. Also, the changes are only effective for the current DAS session. If DAS is ended using the ENDDAS command, and DAS is reentered at a later time, the system will return to the defaults. Also, to be effective any changes made with the SETUP TERMINATION function must be loaded into the SIFT processors using the LOAD option (see para 7 below).

A5. **SETUP PREPROCESSOR**

This function gets the file specification of the user's preprocessing program that is being used for the current test/experiment. The file specification is used by the INJECT and FITINJECT commands to run the proprocessor automatically from a command procedure without user intervention.

A6. **EDIT**

The edit function allows changing any part of a previously created window file. The window file is not a text file and, therefore, cannot be changed in the editor; it can only be changed in the edit function of the window utility. Use of this function is self explanatory. Edit commands are simple and listed at the top of the screen. The pane number to be edited is selected by

pressing arrow keys (up arrow to advance the file and down arrow to go backwards). The selected pane is displayed in the center of the screen and edit commands are entered at the prompt. Changes are immediately displayed on the screen. To exit the function type Cntrl Z.

Note: Changes made to a window file using the EDIT function are always made to the same version of the file. Only one version of the file is maintained. If more than one version is needed, each version must be given a different name.

A7. **LOAD**

This function must be invoked to make a particular window file effective for a data acquisition session. If the window file is not loaded data will not be received. Before LOAD is invoked the SIFT processors (any configuration of P1 through P6) must have been allocated and loaded with the SIFT operating system, and DAS must be activated using the DAS command (SIFT display shows DAS on). The loading procedure loads the addresses of the window variables or memory locations into the currently selected SIFT processors and loads the filter information and other parameters into the acquisition processor P7. The window variables will be packed during the load. That is, any NIL panes in the window will deleted and selected variables/locations will be packed in the order they appear in the file. The window file can be loaded from the main menu, or can be loaded directly from the SIFT$ prompt using the command WINDOW/LOAD windowfilename.

A8. **LIST and PRINT**

These functions simply list the window file on the screen or print the file. Since the file is a binary file, it cannot be listed using the DCL TYPE command or printed using the DCL PRINT command.

Appendix B

The Global Clock

The Global Clock is comprised of a 16 bit counter which can be read by any SIFT processor and the System 9 VAX. The Global Clock's period can be set from the VAX to anywhere within the interval 1 μsec to 4000 seconds. Access to the Global Clock from a SIFT processor is through function GCLOCK.

FUNCTION GCLOCK: INTEGER; EXTERN;

Function GCLOCK returns the Global Clock value. All SIFT processors can access the Global Clock simultaneously without contention.

The period of the Global Clock is programmed from the VAX. The Global Clock has 3 base frequencies: 1 sec., 1 msec., and 1 μsec. A 12 bit presettable up-counter provides additional adjustment to the base frequency (i.e. periods of from 1 to 4095 units can be programmed). SIFT DCL function CLOCK allows a SIFT user to program the clock. If CLOCK is invoked with no arguments, the Global Clock is read and the following message printed:

Clock has unsigned value of XXXXX(base 10).

The CLOCK function can be called to set the desired Global Clock period. The clock period should be expressed in "E" notation in units of seconds.

To set a 10 μsec period - SIFT$ CLOCK 10E-6

To set a 1 sec period - SIFT$ CLOCK 1.0

To set a 2 msec period - SIFT$ CLOCK 2E-3

Because the Global Clock has been known to "freeze", the CLOCK function suspends for a length of time to test the operation of the Global Clock. For periods up to 999 μsec CLOCK will suspend for 50 msec. For 1.0 msec <= period <= 999 msec, CLOCK will suspend for 1.0 sec. For periods over 1.0 sec, CLOCK will suspend for 5 seconds. Of course periods in excess of 4 seconds will not

be tested. However, the user can use CLOCK to read the Global Clock after an appropriate interval and assure himself that the Global Clock is functioning. When the CLOCK task resumes it prints the following message:

Received X clock ticks during 0 ::Y.YY second timeout

If, for example, the user sets a 1 msec period, the following sequence would occur:

```
SIFT$ CLOCK 1.0E-3
Received 999 clock ticks during 0 ::1.00 second timeout
```

Using this data the user can verify the Global Clock programming. At periods below 1 msec (i.e when the timeout period is 50 msec), the correlation between the number of clock ticks received and the timeout period degrades due to VMS's inability to deliver consistent 50 msec intervals. For example, when setting a period of 1 μsec, anywhere from 40000 to 50000 clock ticks will be recorded.

REFERENCES

Green, D. F.; Palumbo, D. L.; Baltrus, D. W.: Software Implemented Fault Tolerance (SIFT) Users Guide. NASA TM-86289, 1983.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-87638		

4. Title and Subtitle	5. Report Date
User's Guide to Programming Fault Injection and Data Acquisition in the SIFT Environment	January 1987
	6. Performing Organization Code 505-66-21-01

7. Author(s)	8. Performing Organization Report No.
Carl R. Elks David F. Green Daniel L. Palumbo	

9. Performing Organization Name and Address	10. Work Unit No.
NASA Langley Research Center Hampton, Virginia 23665-5225	11. Contract or Grant No.

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, DC 20546	Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

Carl R. Elks, PRC-Kentron, Hampton, VA 23666
David F. Green, PRC-Kentron, Hampton, VA 23666
Daniel L. Palumbo, NASA Langley Research Center, Hampton, VA 23665-5225

16. Abstract

This document decribes the features, command language, and functional design of the SIFT (Software Implemented Fault Tolerance) fault injection and data acquisition interface software in detail. It is also intended to assist and guide the SIFT user in defining, developing, and executing SIFT fault injection experiments and the subsequent collection and reduction of that fault injection data. This document is intended to be used in conjunction with the SIFT User's Guide (NASA Technical Memorandum 86289) for reference to SIFT system commands, procedures and functions, and overall guidance in SIFT system programming.

17. Key Words (Suggested by Author(s))		18. Distribution Statement
SIFT	Stuck-at-faults	Unclassified - Unlimited
FIS	Intermittant faults	Subject Category 61
DAS	Transient faults	
WINDOW	Filter	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	53	A04