|  |  |
| --- | --- |
| DocCoverBackground | CORE FLIGHT SYSTEM  Stored Command  BUILD 2.4.0.0  FLIGHT SOFTWARE BUILD VERIFICATON  TEST REPORT  Flight Software Branch – Code 582  Version 1.0 |

Signatures

Submitted by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Walt Moleski /582 Date

CFS Flight Software Tester

Approved by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

David McComas/582 Date

Flight Software Branch/Head

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Susanne Strege/582 Date

CFS Flight Software Product Development Lead

Plan Update History

| Version | Date | Description | Affected Pages |
| --- | --- | --- | --- |
| 1.0 |  | Initial Release | All |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1 Introduction 1

1.1 Document Purpose 1

1.2 Applicable Documents 1

1.3 Document Organization 1

1.4 Definitions 2

2 OVERVIEW 3

2.1 Flight Data System Context 3

2.2 Test History 4

2.3 Testing Overview 4

2.4 Version Information 7

3 Build Verification Test Preparation 8

3.1 Scenerio Development 8

3.2 Procedure Development and Execution 8

3.3 Test Products 8

4 Build Verification Test Execution 9

4.1 Testbed Overview 9

4.2 Requirements Verification Matrix 10

4.3 Requirements Partially Tested 10

4.4 Requirements/Functionality Deferred 10

4.5 Requirements/Functionality Deferred for Mission Testing 10

5 Build Verficaton Test Results 11

5.1 Overall Assessment 11

5.2 Procedure Description 11

5.3 Analysis Requirements Verification 12

5.4 DCRs 14

5.4.1 DCRs Verified 14

5.4.2 Outstanding DCRs 15

5.5 Notes 16

Appendix A - RTTM 17

Appendix B - Command, Telemetry, and Events Verification Matrix 18

# Introduction

## Document Purpose

This Test Report describes the test results from the Core Flight System (CFS) Stored Command (SC) Flight Software (FSW) Test Team build 2.4.0.0 verification testing. It is used to verify that the SC FSW has been tested in a manner that validates that it satisfies the functional and performance requirements defined within the CFS SC Requirements Document. This Test Report summarizes the FSW test history, the build verification process, the build test configuration, and the test execution and results.

## Applicable Documents

Unless otherwise stated, these documents refer to the latest version.

**Parent Documents** (Mission and FSW)

* 582-2007-019 CFS Stored Commands Requirements Document, Version 1.5
* 582-2008-012 CFS Deployment Guide, Version 3.0

**Reference Documents**

All of the references below can be found on the Code 582 internal website at <http://fsw.gsfc.nasa.gov/>

* 582-2003-001 FSB FSW Test Plan Template
* 582-2004-001 FSB FSW Test Description Template
* 582-2004-002 FSB FSW Test Scenario Template
* 582-2004-003 FSB FSW Test Procedure Template
* 582-2004-004 FSB FSW Test Execution Summary Template
* 582-2004-005 FSB Test Product Peer Review Form
* 582-2000-002 FSB FSW Unit Test Standard

## Document Organization

Section 1 of this document presents some introductory material.

Section 2 provides a flight software overview and context along with the test history and testing overview.

Section 3 describes the build verification process including procedure development and execution and test products produced.

Section 4 describes the build test configuration which includes an overview of the testbed and the requirements verification matrix.

Section 5 describes the test execution and results by subsystem.

Appendix A - provides the Requirements Traceability Matrix

Appendix B - provides the Command, Telemetry, and Events Verification Matrix

## Definitions

There were 3 verifications methods used during build verification testing. They were:

* Demonstration: Show compliance with system requirement by exhibiting the required capability (e.g. by demonstrating interactive capability, display capability, print capability, etc.
* Inspection: Show compliance with a system requirement by visual verification of the software (e.g. verifying preparation for delivery, proper interfacing)
* Analysis: Perform detailed analysis of code, generated data (both intermediate data and final output data), etc., to determine compliance with system requirements.

The fields in the Requirements Verification Matrix in Section 4.3 are defined as follows:

* Requirements Tested Passed: Requirement was fully tested in a build test procedure and passed all tests.
* Requirements Tested Failed: Requirement was fully tested in a build test procedure and failed one or more aspect of the testing.
* Requirements Tested Partially: Requirement was tested partially in a build test procedure. To be fully tested, the partially tested requirement is either tested additionally in one or more other test procedures within the same build **and/or** other aspects of the requirement must be tested in a later build, due to capabilities not present in the current build
* Total Tested: Total number of requirements fully tested in a build test procedure. Includes total passed and total failed, but does **not** include requirements tested partially, **unless** (included as a separate entry) testing in multiple procedures within the same build constitutes total testing of a particular requirement. Total Requirements Tested is computed this way in order to avoid multiple counting of individual requirements that are tested partially in more than one procedure.
* Deferred: Number of requirements that were planned to be tested in current build, but were not tested due to some FSW capability or necessary system component not being present.
* Total: Total Requirements Tested + Number of Requirements Deferred

In each software test section in Section 5 there is a table of DCR’s. The state definitions are as follows:

* Opened: The DCR is currently being addressed
* Assigned: The DCR was accepted and the modification is being addressed
* InTest: The DCR was corrected and is currently in test
* Validated: The DCR was corrected and tested and has been validated, needs to have a CCB to close the DCR
* Closed: The DCR is closed and have been resolved and tested to satisfaction
* Closed with Defect: The DCR is closed and the defect is most likely assigned a differed DCR number associated with another subsystem.

# OVERVIEW

## Flight Data System Context

Figure 2-1 illustrates the CFS system context. The cFE interfaces to five external systems: an [Operating System](#Operating_System) (OS), a [Hardware Platform](#Hardware_Platform) (HP), an [Operational Interface](file:///C:\Users\wmoleski\Documents\Projects\cFE\CFS\Stored%20Command\Testing\2.4.0.0\Operational_Interface) (OI), [Applications](#Application) (APP), and other cFE-based systems.



**Figure 2-1 CFS System Context**

The figure below shows major interfaces between the Stored Command application and other core Flight Executive (cFE) and Core Flight System (CFS) applications. Although it isn’t shown explicitly, all task-to-task communications are accomplished via the cFE Software Bus (SB) application.

Inputs to the Stored Command application include: 1) Wake-up calls from the Scheduler (SCH) application which trigger processing, 2) Housekeeping requests from the Scheduler (SCH) application which trigger housekeeping data collection, 3) configuration commands from the Command Ingest (CI) application, and 4) updates to Stored Command Tables managed by the Table Services (TBL) application.

Outputs from the Stored Command application include: 1) Stored Command housekeeping messages sent to the Housekeeping (HK) application, 2) Commands sent to all applications for processing, and 3) Event messages.

SC

ATS Tables

CI

SCH

HK, DS, TO

All Apps

ATS

Status Tables

RTS Tables

RTS

Status Tables

House Keeping

Events

1 Hz Wakeup Req

Ground Commands

Read

Read/Write

Read

Read/Write

House Keeping Req

Stored

Commands

**Figure 2-1 CFS SC Context**

## Test History

SC 1.0.0.0 – Build Verification Testing completed 3/23/2009 by Walt Moleski

SC 2.0.0.0 – Build Verification Testing completed 8/31/2009 by Walt Moleski

SC 2.1.1.0 – Build Verification Testing completed 2/17/2011 by Walt Moleski

SC 2.2.0.0 - Build Verification Testing completed 9/8/2011 by Walt Moleski

SC 2.2.1.0 - Build Verification Testing completed 10/6/2011 by Walt Moleski

SC 2.3.0.0 - Build Verification Testing completed 1/11/2012 by Walt Moleski

SC 2.4.0.0 - Build Verification Testing completed 1/20/2015 by Walt Moleski

## Testing Overview

The SC application was tested during Build Verification testing using the following:

* 1 test application: tst\_sc
* 5 main test procedures: sc\_atsfunc.prc, sc\_gencmds.prc, sc\_resetnocds.prc, sc\_rtsfunc.prc, sc\_stress.prc
* 11 test procedures that are called by the main procedures: sc\_start\_apps.prc, sc\_700cmdats.prc, sc\_appendfull.prc, sc\_appoffend.prc, sc\_atsoddbyte.prc, sc\_atsoffend.prc, sc\_maxcmdats.prc, sc\_loadrts1.prc, sc\_loadrts2.prc, sc\_rtsoddbyte.prc and sc\_rtsoffend.prc
* App tests require the Advanced Spacecraft Integration and System Test (ASIST) Ground Station

The TST\_SC test application is used to send schedule requests for the output of SC’s housekeeping data to the SC application. This was useful when performing build verification testing since it provided great control over the sequence of steps. In addition, having the test application eliminated the need to modify the SCH\_LAB application and rebuild. When deployed for a mission, the Scheduler Application would provide this request. In addition, the test application has 5 ground commands defined to help with the SC testing. These commands are described below:

* TST\_SC\_NOOP
  + This command that issues an event and increments the command processed counter.
* TST\_SC\_ResetCtrs
  + This command resets the command processed and command error counters to zero (0).
* TST\_SC\_SetCounters
  + This command sets several Stored Command (SC) counters so that the SC\_ResetCtrs command can be tested and verified.
* TST\_SC\_GetCRC
  + This command generates a CRC value for the supplied data and displays this value in the TST\_SC housekeeping page for use by the test procedures.
* TST\_SC\_GetTime
  + This command retrieves the current requested time based upon the supplied type and displays this value in the TST\_SC housekeeping page.

The SC 2.4.0.0 testing was performed using 2 different configurations. Each configuration required a separate compilation with changes to the PLATFORM\_DEFINED configuration parameters. The SC 2.4.0.0 configurations are described below:

* Normal: SC compiled out of the box using the cFE default time (TAI) and Group Commands enabled.
* UTC Time/Group Disabled: SC compiled with the SC\_TIME\_TO\_USE parameter set to SC\_USE\_UTC and the SC\_ENABLE\_GROUP\_COMMANDS set to FALSE.

The 5 main SC test procedures do the following:

|  |  |
| --- | --- |
| **Procedure** | **Description** |
| sc\_atsfunc | The purpose of this test is to verify that Absolute Time Sequences (ATS) execute and function properly. |
| sc\_gencmds | The purpose of this test is to verify that the SC general commands execute and function properly. |
| sc\_resetnocds | The purpose of this test is to verify that the SC application does not save any data across a reset (Application, Processor, or Power-On). |
| sc\_rtsfunc | The purpose of this test is to verify that Relative Time Sequences (RTS) execute and function properly. |
| sc\_stress | The purpose of this test is to verify that the SC application supports the execution of ATSs and RTSs simultaneously. Also, this test verifies several conditions regarding the maximum number of commands executing simultaneously and the priority of those commands. |

The test procedures described in the table below are called by at least one of the test procedures above.

|  |  |
| --- | --- |
| **Procedure** | **Description** |
| sc\_700cmdats | This procedure creates an ATS table load image containing 700 valid commands for ATS B. |
| sc\_appendfull | This procedure creates an ATS Append table load image containing a full table of valid commands. |
| sc\_appoffend | This procedure creates an ATS Append table load image containing a full table of commands with the last command going over the end of the table buffer. |
| sc\_atsoddbyte | This procedure creates an ATS table load image file containing two odd byte commands along with another command contained in the buffer. The command that follows each odd byte command must start on an even word boundary. This means that the buffer must pad to the next word. |
| sc\_atsoffend | The purpose of this proc is to create an ATS table load file containing a full table of commands with the last command going over the end of the table buffer. Basically, the last command is incomplete. |
| sc\_loadrts1 | The purpose of this proc is to create a table load file for the first RTS table. |
| sc\_loadrts2 | The purpose of this proc is to create a table load file for the second RTS table. |
| sc\_maxcmdats | The purpose of this proc is to create an ATS table load file containing one more command than the maximum number of commands allowed in an ATS. |
| sc\_rtsoddbyte | This procedure creates a table load image file for RTS #3 containing two odd byte commands along with another command contained in the buffer. The command that follows each odd byte command must start on an even word boundary. This means that the buffer must pad to the next word. |
| sc\_rtsoffend | The purpose of this proc is to create an RTS table load file containing a full table of commands with the last command going over the end of the table buffer. Basically, the last command is incomplete. |
| sc\_start\_apps | The purpose of this proc is to start the SC and TST\_SC applications. |

The CFS Deployment Guide contains the instruction for how to set up both the CFS Flight and Ground test environment. The testers use a CFS Test Account for each build test. This account runs ASIST and is setup to contain all the files needed to test the application. These files are extracted from MKS, the source repository tool. Included in these files are test utilities. These utilities can be located in 2 places depending upon whether they are “local” or “global” utilities. The local utilities are extracted into the working prc directory ($WORK/prc). The global utilities are pointed to by ASIST in the global area defined on the test system. Additional tools utilized by the test procedures are located in the $TOOLS directory. It is assumed that test procedures and the ASIST telemetry database used for testing is built using procedure and database templates

The following utilities were used during testing:

|  |  |
| --- | --- |
| **Name** | **Description** |
| cfe\_startup | Directive combines the "start\_data\_center", "open\_tlm", and "open cmd <cpu>" ASIST startup commands. |
| close\_data\_center | Directive that closes the command and telemetry connection to the CPU being used. |
| create\_tbl\_file\_from\_cvt | Procedure that creates a load file from the specified arguments and cvt |
| load\_start\_app | Procedure to load and start a user application from the /s/opr/accounts/cfebx/apps/cpux directory. |
| load\_table | Procedure that takes the specified file and transfers the file to the specified processor and then issues a TBL\_LOAD command using the file. |
| tst\_sc (version 2.4.0.0) | Test application required to test the SC application. |
| ut\_pfindicate | Directive to print the pass fail status of a particular requirement number. |
| ut\_runproc | Directive to formally run the procedure and capture the log file. |
| ut\_sendcmd | Directive to send EVS commands Verifies command processed and command error counters. |
| ut\_sendrawcmd | Send raw commands to the spacecraft. Verifies command processed and command error counters. |
| ut\_setrequirements | A directive to set the status of the cFE requirements array. |
| ut\_setupevents | Directive to look for multiple events and increment a value for each event to indicate receipt. |
| ut\_tlmupdate | Procedure to wait for a specified telemetry point to update. |
| ut\_tlmwait | Directive that waits for the specified telemetry condition to be met |

## Version Information

|  |  |
| --- | --- |
| Item | Version |
| SC Requirements | 1.5 |
| SC Application | 2.4.0.0 |
| TST\_SC Application | 2.4.0.0 |
| CFE | 6.4.1.0 |
| ASIST | 20.2 |
| VxWorks | 6.4 |

# Build Verification Test Preparation

## Scenerio Development

No new scenarios were developed for SC 2.4.0.0 Build Verification Test. All scenarios are stored on the MKS server, in CFS-Repository SC test-and-ground directory within the Scenarios subdirectory. It should be noted that as SC requirements and BVT procedures evolve these scenarios are not updated to reflect any changes made.

## Procedure Development and Execution

This build test was completed by running 5 test procedures. All 5 test procedures were modified as a result of SC 2.4.0.0 changes and are checked in to MKS at the conclusion of build testing. All test procedures were written using the STOL scripting language. The naming convention for files created by the test procedures was: scx\_cpu<#>\_<procedure name>\_GMT.<ext>.

## Test Products

Four log files were generated for every procedure that was run. They are defined as follows:

* Logs with the .loge extension list all events sent by the flight software
* Logs with the .logr extension list all requirements that passed validation by demonstration
* Logs with the .logp extension lists all prints that are generated by the test procedure
* Logs with the .logf extension lists everything from the other logs along with the steps in the test procedure
* Logs with the .logs extension lists the SFDU information (if applicable) contained in the full log.

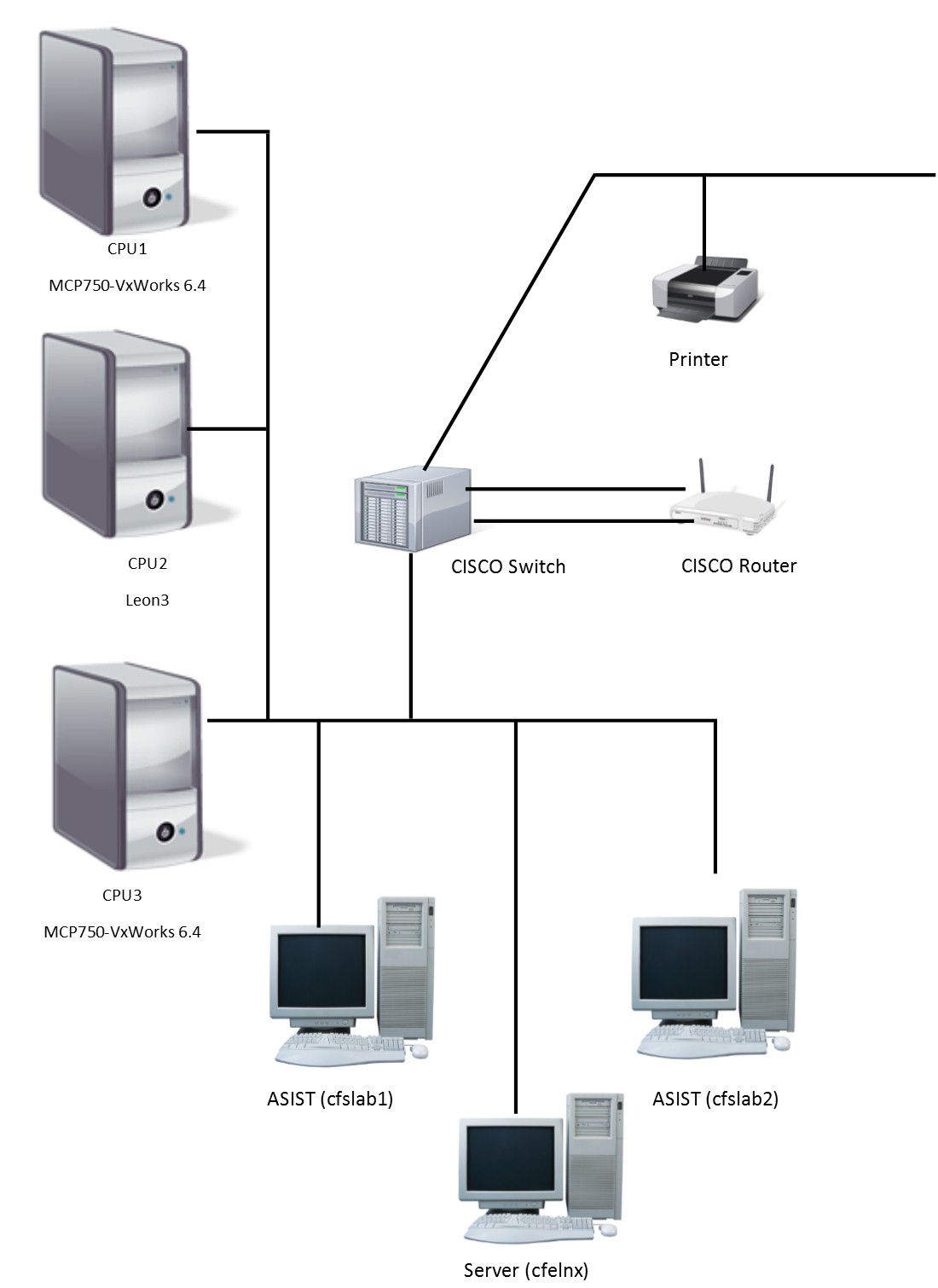
A test summary report is developed in MKS for each procedure by the tester after build testing is completed. All test products are maintained on MKS in the CFS-Repository SC test-and-ground directory.

The SC 2.4.0.0 test results contain 2 sets of test products for the sc\_atsfunc and sc\_rtsfunc tests. One set for each of the test configurations described in Section 2.3 above.

# Build Verification Test Execution

## Testbed Overview

SC FSW testing took place in the CFS FSW Development and Test Facility. A high level view of the CFS FSW Test Bed is shown in Figure 4-1. This facility is located in GSFC Building 23, Room N410. This facility consists of two ASIST workstations running ASIST version 20.2 and three MPC750 CPU boards running VxWorks 6.4. CPU1 is primarily used for development testing while CPU2 and CPU3 are used for build verification testing.



**Figure 4-1 CFS FSW Development and Testing Facility**

## Requirements Verification Matrix

|  |  |
| --- | --- |
|  | Stored Command (SC) |
| Requirements Tested Passed | 71 |
| Requirements Tested Failed | 0 |
| Requirements Tested Partially | 0 |
| Total Tested | 71 |
| Deferred | 0 |
| Total | 71 |

## Requirements Partially Tested

No requirements were partially tested.

## Requirements/Functionality Deferred

No requirements/functionality was deferred.

## Requirements/Functionality Deferred for Mission Testing

No requirements/functionality was deferred to mission testing.

# Build Verficaton Test Results

## Overall Assessment

During this build test of the SC Application the software behaved as expected with several problems still outstanding from previous testing.

Below is a summary of the results:

* 60 requirements passed via demonstration.
* 11 requirements were validated by analysis.
* 10 existing DCRs were verified.

## Procedure Description

| **Procedure** | **Description** | **Requirements tested** |
| --- | --- | --- |
| sc\_atsfunc | The purpose of this test is to verify that Absolute Time Sequences (ATS) execute and function properly. | SC1002, SC1004, SC1005, SC2000, SC2000.1, SC2000.2, SC2000.3, SC2000.4, SC2001, SC2004, SC2007, SC2007.1, SC2008, SC2008.1, SC2008.2, SC2008.3, SC2008.4, SC2008.5, SC2008.6, SC2008.7, SC2009, SC3000, SC3000.1, SC3000.3, SC3000.3.1, SC3000.3.2, SC3001, SC3001.1, SC3002, SC3002.1, SC3002.2, SC3002.3, SC3002.4, SC3003, SC3003.1, SC3003.2, SC3003.2.1, SC3003.2.2, SC3003.3, SC3003.4, SC3004, SC3005, SC8000, SC9000 |
| sc\_gencmds | The purpose of this test is to verify that the SC general commands execute and function properly. | SC1000, SC1001, SC1002, SC1004, SC1005, SC8000, SC9000 |
| sc\_resetnocds | The purpose of this test is to verify that the SC application does not save any data across a reset (Application, Processor, or Power-On). This test should NOT be executed if the configuration parameter indicating Save Critical Data is set by the Mission. | SC1004, SC2000.2, SC2000.3, SC2001, SC2002.2, SC2002.3, SC2003, SC3000, SC4000, SC4000.1, SC4004, SC8000, SC9000, SC9004, SC9005 |
| sc\_rtsfunc | The purpose of this test is to verify that Relative Time Sequences (RTS) execute and function properly. | SC1002, SC1004, SC1005, SC2002, SC2002.1, SC2002.2, SC2002.3, SC2003, SC2005, SC2006, SC4000, SC4000.1, SC4000.2, SC4001, SC4001.1, SC4001.2, SC4001.3, SC4001.3.1, SC4001.4, SC4002, SC4003, SC4004, SC4005, SC4005.1, SC8000, SC9000. SC9004 |
| sc\_stress | The purpose of this test is to verify that the SC application supports the execution of ATSs and RTSs simultaneously. Also, this test verifies several conditions regarding the maximum number of commands executing simultaneously and the priority of those commands. | SC1004, SC2000.2, SC2000.3, SC2001, SC2002, SC2002.2, SC2002.3, SC2003, SC2005, SC2005.1, SC2005.2, SC3000, SC3001, SC4000, SC4000.1, SC4003, SC4004, SC8000, SC9000, SC9004 |

## Analysis Requirements Verification

There were 11 requirements verified using analysis.

| **Requirement** | **Requirement Text** | **Analysis** |
| --- | --- | --- |
| SC2000.1 | ATS commands were executed every second with the proper delay as specified in the loaded table | Step 2.6 of the atsfunc test procedure verifies this requirement. The ATS commands executed with the proper delay between them as specified in the table load file. The load file contained a 5 second delay between the commands. |
| SC2000.4 | This requirement was tested using normal time (TAI) and UTC time. For the test, the LeapSeconds were jammed to be 5 less than the current value. In normal time, there was no affect. In UTC time, 6 commands executed in the same second. | Steps 5.4 thru 5.6 of the atsfunc test procedure verify this requirement. When TAI (CFE\_DEFAULT) time was used, a jam of the LeapSeconds did not have any effect on the ATS execution. When using UTC time, there were 6 commands that executed in the same second when the LeapSeconds was jammed from 32 to 27 seconds. |
| SC2002.1 | RTS commands executed every second as specified in the loaded table. | Step 2.7 of the rtsfunc test procedure verifies this requirement. The RTS commands executed every 1 second as specified in the loaded table. |
| SC2004 | ATS commands were executed in the proper “time” order rather than the order loaded in the table. | Step 2.6 of the atsfunc test procedure verifies this requirement. The ATS commands were executed in the proper "time" order rather than the order loaded in the table. The table load contained command 2 (SC\_NOOP), command 1 (TBL\_NOOP), and command 3 (EVS\_NOOP). |
| SC2005 | The maximum commands per second were executed as specified by the configuration parameter. | Step 2.19 of the rtsfunc and 2.17 of the stress test procedures verify this requirement. RTS #3 contained 6 commands (3 SC\_NOOP and 3 ES\_NOOP) that execute each second for 2 consecutive seconds. RTS #8 contained 4 commands (TBL\_NOOP) that executed each second for 3 seconds. The test log shows that the max number of commands (8) executed in any one second. |
| SC2005.1 | ATS commands were executed before RTS commands and the RTS commands were properly deferred. | Step 2.17 of the stress test procedure verifies this requirement. ATS B was loaded with the maximum commands per second (8) and executed for 6 seconds. The first ATS command started RTS #5 which contained 3 commands (TO\_NOOP, CI\_NOOP and TST\_SC\_NOOP) that executed for 6 consecutive seconds. The ATS commands executed first followed by the deferred RTS commands until both sequences completed. |
| SC2006 | The higher priority RTSs commands were executed first followed by the lower priority RTSs commands. | Step 2.19 of the rtsfunc test procedure verifies this requirement. RTS #3 contained 6 commands (3 SC\_NOOP and 3 ES\_NOOP) that execute each second for 2 consecutive seconds. RTS #8 contained 4 commands (TBL\_NOOP) that executed each second for 3 seconds. The test log shows that all RTS #3 commands were executed in each second along with 2 of RTS #8 commands. The remaining RTS #8 commands executed when RTS #3 completed. |
| SC3002.4 | The Switch command was executed immediately after it was dispatched. | Step 4.27 of the atsfunc test procedure waits for the switch command to execute. However, in this test run, the Switch command executed prior to getting to this step. The ATS purposely contained a switch command to an empty ATS in order to generate an error event. The error event message was captured which indicated that the switch command executed immediately since it was contained in an ATS. |
| SC4001 | RTS commands were dispatched in the order they were loaded in the table. | Step 2.6 and 2.7 of the rtsfunc test procedure verify this requirement. Three commands were loaded into RTS #2. The command order in the table was SC\_NOOP, TBL\_NOOP and EVS\_NOOP. The RTS commands were dispatched in the correct order. |
| SC4001.1 | The delay was interpreted properly. | Step 2.6 and 2.7 of the rtsfunc test procedure verify this requirement. The delay prior to the first command was 5 seconds, followed by a 1 second delay between the other commands. The test log shows these delays. |
| SC4001.2 | The delay for the first command was relative to the Start RTS command. | Step 2.6 and 2.7 of the rtsfunc test procedure verify this requirement. The first command in the RTS table was to delay for 5 seconds. The log file of the test shows that the first command executed approximately 5 seconds after the RTS was started. |

## DCRs

No new DCRs were generated during SC 2.4.0.0 testing.

### DCRs Verified

The following DCRs were verified during testing:

| **DCR** | **Description** | **Test Method** | **Test Approach** |
| --- | --- | --- | --- |
| 17270 | SC – Remove Table Compiler Warning from Default Table(s) | Analysis | No compiler warnings were generated when compiling the default tables |
| 19196 | SC – Reports Incorrect Number of RTS’s Loaded at Startup | Test Procedure | The resetnocds test procedure verifies this DCR. Normal startup of 2 RTSs occurs first in Step 1.4.2 as well as other steps. Step 4.3.2 reports 0 RTSs loaded since the default tables were removed for this step. |
| 19212 | GPM-IVV-1291 – SC – Not Consistently Verifying Message Length | Test Procedure | Stpe 2.7 of the gencmds test sends an invalid length HK request. The command was rejected as expected. |
| 19500 | SC – Update Design Charts to Reflect Removal of Pad Words from ATS and RTS Command Table Entries | Inspection | The design charts no longer reference pad words. |
| 20976 | SC – Uninitialized Variable Causes Compiler Warning | Analysis | No compiler warnings were generated when the SC fsw was compiled. |
| 21200 | Update CFS/SC documentation to reflect secret pad byte needed in RTSs and ATSs for odd length commands | Inspection | The documentation noted in the DCR has been changed to reflect the necessary information. |
| 21590 | SC – Users Guide does not provide Critical ATS Table information. | Inspection | The documentation noted in the DCR has been changed to reflect the necessary information. |
| 21740 | SC – Incorrect RTP Ctrl Table Handle Used in Table Manage Command | Analysis | Manually dumped the table using the ASIST command browser. The table dumped successfully. |
| 22621 | SC – RTS Event Message Text is Inconsistent | Test Procedure | A scan of the log files for all test procedures did not find "Relative Time Sequence" or "Absolute Time Sequence" spelled out. They were specified as RTS and ATS respectively |
| 22836 | SC tables have CCSDS primary packet headers that must be Big Endian on Little Endian platforms | Test Procedure | The default tables loaded properly on a Big Endian machine and the default RTSs loaded on a Little Endian machine at startup. |

### Outstanding DCRs

|  |  |  |
| --- | --- | --- |
| **DCR** | **Description** | **State** |
| 7337 | There is no notification from SC when the non-critical table CDSs are corrupted upon restart | On Hold |
| 7338 | The ATS and RTS Table data is restored across a reset when it has been corrupted | On Hold |
| 7339 | SC does not initialize the RTS Table data loaded when other CDSs were not restored properly | On Hold |
| 14339 | Critical data is not saved across a reset | Submitted |
| 21109 | The CFS/SC startup time with 320 RTSs on MMS is nearly 2 minutes | On Hold |
| 22014 | SC – Add Trick Simulation Support (JSC Request) | Submitted |
| 22593 | Allow duplicate command numbers in ATS loads (SC2001) | Submitted |
| 22714 | SC: Consider adding ATS filename to hk telemetry packet | Submitted |

## Notes

It should be noted that integration testing is the ultimate verification of the SC applications performance in a system-like scenario.

1. RTTM

The SC Build 2.4.0.0 RTTM can be found on the MKS server, in CFS-Repository SC test-and-ground directory results folder.

1. Command, Telemetry, and Events Verification Matrix

|  |  |  |
| --- | --- | --- |
| **Command** | **Test Procedure(s)** | **Notes/Comments** |
| SC\_NOOP | sc\_gencmds |  |
| SC\_ResetCtrs | sc\_gencmds |  |
| SC\_StartATS | sc\_atsfunc, sc\_resetcds, sc\_resetnocds, sc\_stress |  |
| SC\_StopATS | sc\_atsfunc, sc\_stress |  |
| SC\_StartRTS | sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_StopRTS | sc\_rtsfunc, sc\_stress |  |
| SC\_DisableRTS | sc\_rtsfunc |  |
| SC\_EnableRTS | sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_SwitchATS | sc\_atsfunc |  |
| SC\_JumpATS | sc\_atsfunc |  |
| SC\_ContinueATS | sc\_atsfunc |  |
| SC\_AppendATS | sc\_atsfunc |  |
| SC\_StartRTSGroup | sc\_rtsfunc | Controlled via configuration parameter |
| SC\_StopRTSGroup | sc\_rtsfunc | Controlled via configuration parameter |
| SC\_DisableRTSGroup | sc\_rtsfunc | Controlled via configuration parameter |
| SC\_EnableRTSGroup | sc\_rtsfunc | Controlled via configuration parameter |

|  |  |  |
| --- | --- | --- |
| **Telemetry** | **Test Procedure(s)** | **Notes/Comments** |
| SC\_ATSNumber | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_ATPState | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_ContATSFlag | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_ATPCmdNumber | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_SwitchPend | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_ActiveRTSs | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_NextRTS | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_CMDEC | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_CMDPC | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_RTSActvCtr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_RTSActvErr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_ATSCmdCtr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_ATSErrCtr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_RTSCmdCtr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_RTSErrCtr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_LastATSErr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_LastATSCmdErr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_LastRTSErr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_LastRTSCmdErr | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_AppendATSID | atsfunc |  |
| SC\_AppendCount | atsfunc |  |
| SC\_AppendSize | atsfunc |  |
| SC\_AppendLoads | atsfunc |  |
| SC\_FreeBytes[2] | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_NextRTSTime | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_NextATSTime | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_RTSExeStatus[RTS\_Tables/16] | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| SC\_RTSDisableStatus[RTS\_Tables/16] | sc\_atsfunc, sc\_gencmds, sc\_resetcds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| **Table Telemetry** |  |  |
| SC\_ATP\_State |  |  |
| SC\_ATSNum |  |  |
| SC\_CmdNum |  |  |
| SC\_TimIdx |  |  |
| SC\_SwitchFlag |  |  |
| SC\_ATSCMD\_Status[SC\_MAX\_ATS\_CMDS] | sc\_atsfunc |  |
| SC\_ATSInfo\_Table[2].UsageCtr |  |  |
| SC\_ATSInfo\_Table[2].CmdCtr |  |  |
| SC\_ATSInfo\_Table[2].ATSSize |  |  |
| SC\_ATS\_DATA[SC\_ATS\_BUFF\_SIZE] | sc\_atsfunc, sc\_resetcds, sc\_resetnocds, sc\_stress |  |
| SC\_RTP\_ActiveCtr |  |  |
| SC\_RTP\_NextRTS |  |  |
| SC\_RTSINFO\_Table[RTS\_Tables].Status |  |  |
| SC\_RTSINFO\_Table[RTS\_Tables].DisabledFlag |  |  |
| SC\_RTSINFO\_Table[RTS\_Tables].CmdCtr |  |  |
| SC\_RTSINFO\_Table[RTS\_Tables].CmdErrCtr |  |  |
| SC\_RTSINFO\_Table[RTS\_Tables].NextCmdTime |  |  |
| SC\_RTSINFO\_Table[RTS\_Tables].NextCmd |  |  |
| SC\_RTSINFO\_Table[RTS\_Tables].UsageCtr |  |  |
| SC\_RTS\_Data[SC\_RTS\_BUFF\_SIZE] | sc\_resetcds, sc\_resetnocds, sc\_rtsfunc , sc\_stress |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Event Message** | **Test Procedure(s)** | **Notes/Comments** |
| 1 | SC\_APP\_EXIT\_ERR\_EID |  |  |
| 2 | SC\_LEN\_ERR\_EID | sc\_atsfunc, sc\_gencmds, sc\_rtsfunc |  |
| 3 | SC\_INIT\_SB\_CREATE\_ERR\_EID |  |  |
| 4 | SC\_INIT\_SB\_SUBSCRIBE\_HK\_ERR\_EID |  |  |
| 5 | SC\_INIT\_SB\_SUBSCRIBE\_1HZ\_ERR\_EID |  |  |
| 6 | SC\_INIT\_SB\_SUBSCRIBE\_CMD\_ERR\_EID |  |  |
| 9 | SC\_INIT\_INF\_EID | sc\_atsfunc, sc\_gencmds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| 10 | SC\_REGISTER\_RTS\_TBL\_NO\_CDS\_ERR\_EID |  |  |
| 11 | SC\_REGISTER\_ATS\_TBL\_NO\_CDS\_ERR\_EID |  |  |
| 16 | SC\_REGISTER\_RTS\_INFO\_TABLE\_ERR\_EID |  |  |
| 17 | SC\_REGISTER\_RTS\_CTRL\_BLK\_TABLE\_ERR\_EID |  |  |
| 18 | SC\_REGISTER\_ATS\_INFO\_TABLE\_ERR\_EID |  |  |
| 19 | SC\_REGISTER\_ATS\_CTRL\_BLK\_TABLE\_ERR\_EID |  |  |
| 20 | SC\_REGISTER\_ATS\_CMD\_STATUS\_TABLE\_ERR\_EID |  |  |
| 21 | SC\_RTS\_LOAD\_COUNT\_INFO\_EID | sc\_atsfunc, sc\_gencmds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| 23 | SC\_STARTATS\_CMD\_INF\_EID | sc\_atsfunc, sc\_resetnocds, sc\_stress |  |
| 24 | SC\_STARTATS\_CMD\_NOT\_LDED\_ERR\_EID |  |  |
| 25 | SC\_STARTATS\_CMD\_NOT\_IDLE\_ERR\_EID | sc\_atsfunc |  |
| 26 | SC\_STARTATS\_CMD\_INVLD\_ID\_ERR\_EID | sc\_atsfunc |  |
| 27 | SC\_STOPATS\_CMD\_INF\_EID | sc\_atsfunc, sc\_stress |  |
| 28 | SC\_STOPATS\_NO\_ATS\_INF\_EID |  |  |
| 29 | SC\_ATS\_SKP\_ALL\_ERR\_EID |  |  |
| 30 | SC\_ATS\_ERR\_SKP\_DBG\_EID | sc\_atsfunc, sc\_resetnocds, sc\_stress |  |
| 31 | SC\_SWITCH\_ATS\_CMD\_INF\_EID | sc\_atsfunc |  |
| 32 | SC\_SWITCH\_ATS\_CMD\_NOT\_LDED\_ERR\_EID |  |  |
| 33 | SC\_SWITCH\_ATS\_CMD\_IDLE\_ERR\_EID | sc\_atsfunc |  |
| 34 | SC\_ATS\_SERVICE\_SWTCH\_INF\_EID | sc\_atsfunc |  |
| 35 | SC\_SERVICE\_SWITCH\_ATS\_CMD\_LDED\_ERR\_EID |  |  |
| 36 | SC\_ATS\_SERVICE\_SWITCH\_IDLE\_ERR\_EID |  |  |
| 37 | SC\_ATS\_INLINE\_SWTCH\_INF\_EID |  |  |
| 38 | SC\_ATS\_INLINE\_SWTCH\_NOT\_LDED\_ERR\_EID | sc\_atsfunc |  |
| 39 | SC\_JUMPATS\_CMD\_STOPPED\_ERR\_EID | sc\_atsfunc |  |
| 40 | SC\_JUMP\_ATS\_INF\_EID | sc\_atsfunc |  |
| 41 | SC\_JUMPATS\_CMD\_NOT\_ACT\_ERR\_EID | sc\_atsfunc |  |
| 42 | SC\_CONT\_CMD\_ERR\_EID |  |  |
| 43 | SC\_CONT\_CMD\_DEB\_EID | sc\_atsfunc |  |
| 44 | SC\_ATS\_CHKSUM\_ERR\_EID | sc\_atsfunc |  |
| 45 | SC\_ATS\_ABT\_ERR\_EID | sc\_atsfunc |  |
| 46 | SC\_ATS\_DIST\_ERR\_EID |  |  |
| 47 | SC\_ATS\_MSMTCH\_ERR\_EID |  |  |
| 48 | SC\_ATS\_SKP\_ERR\_EID |  |  |
| 49 | SC\_RTS\_DIST\_ERR\_EID |  |  |
| 50 | SC\_RTS\_CHKSUM\_ERR\_EID | sc\_rtsfunc |  |
| 51 | SC\_RESET\_DEB\_EID | sc\_gencmds |  |
| 52 | SC\_NOOP\_INF\_EID | sc\_atsfunc, sc\_gencmds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| 59 | SC\_RTS\_INVLD\_MID\_ERR\_EID |  |  |
| 60 | SC\_RTS\_LEN\_ERR\_EID | sc\_rtsfunc |  |
| 61 | SC\_RTS\_LEN\_BUFFER\_ERR\_EID | sc\_rtsfunc |  |
| 62 | SC\_RTS\_LEN\_TOO\_LONG\_ERR\_EID |  |  |
| 63 | SC\_MID\_ERR\_EID |  |  |
| 64 | SC\_INVLD\_CMD\_ERR\_EID | sc\_gencmds |  |
| 65 | SC\_GET\_ADDRESS\_RTS\_INFO\_ERR\_EID |  |  |
| 66 | SC\_GET\_ADDRESS\_RTS\_CTRL\_BLCK\_ERR\_EID |  |  |
| 67 | SC\_GET\_ADDRESS\_ATS\_INFO\_ERR\_EID |  |  |
| 68 | SC\_GET\_ADDRESS\_ATS\_CTRL\_BLCK\_ERR\_EID |  |  |
| 69 | SC\_GET\_ADDRESS\_ATS\_CMD\_STAT\_ERR\_EID |  |  |
| 70 | SC\_GET\_ADDRESS\_RTS\_ERR\_EID |  |  |
| 71 | SC\_GET\_ADDRESS\_ATS\_ERR\_EID |  |  |
| 72 | SC\_STARTRTS\_CMD\_DBG\_EID | sc\_stress |  |
| 73 | SC\_RTS\_START\_INF\_EID | sc\_atsfunc, sc\_gencmds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| 74 | SC\_STARTRTS\_CMD\_INVLD\_LEN\_ERR\_EID | sc\_rtsfunc |  |
| 75 | SC\_STARTRTS\_CMD\_NOT\_LDED\_ERR\_EID | sc\_rtsfunc |  |
| 76 | SC\_STARTRTS\_CMD\_DISABLED\_ERR\_EID | sc\_resetnocds, sc\_rtsfunc |  |
| 77 | SC\_STARTRTS\_CMD\_INVALID\_ERR\_EID | sc\_rtsfunc |  |
| 78 | SC\_STOPRTS\_CMD\_INF\_EID | sc\_rtsfunc, sc\_stress |  |
| 79 | SC\_STOPRTS\_CMD\_ERR\_EID | sc\_rtsfunc |  |
| 80 | SC\_DISABLE\_RTS\_DEB\_EID | sc\_rtsfunc |  |
| 81 | SC\_DISRTS\_CMD\_ERR\_EID | sc\_rtsfunc |  |
| 82 | SC\_ENABLE\_RTS\_DEB\_EID | sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| 83 | SC\_ENARTS\_CMD\_ERR\_EID | sc\_rtsfunc |  |
| 84 | SC\_RTS\_LNGTH\_ERR\_EID |  |  |
| 85 | SC\_RTS\_CMD\_LNGTH\_ERR\_EID |  |  |
| 86 | SC\_RTS\_COMPL\_INF\_EID | sc\_atsfunc, sc\_gencmds, sc\_resetnocds, sc\_rtsfunc, sc\_stress |  |
| 87 | SC\_ATS\_COMPL\_INF\_EID | sc\_atsfunc, sc\_stress |  |
| 88 | SC\_JUMP\_ATS\_SKIPPED\_DBG\_EID | sc\_atsfunc |  |
| 90 | SC\_REGISTER\_APPEND\_INFO\_TABLE\_ERR\_EID |  |  |
| 91 | SC\_GET\_ADDRESS\_APPEND\_INFO\_ERR\_EID |  |  |
| 92 | SC\_GET\_ADDRESS\_APPEND\_ERR\_EID |  |  |
| 93 | SC\_REGISTER\_APPEND\_TBL\_NO\_CDS\_ERR\_EID |  |  |
| 97 | SC\_UPDATE\_APPEND\_EID | sc\_atsfunc |  |
| 98 | SC\_APPEND\_CMD\_INF\_EID | sc\_atsfunc |  |
| 99 | SC\_APPEND\_CMD\_ARG\_ERR\_EID | sc\_atsfunc |  |
| 100 | SC\_APPEND\_CMD\_TGT\_ERR\_EID | sc\_atsfunc |  |
| 101 | SC\_APPEND\_CMD\_SRC\_ERR\_EID | sc\_atsfunc |  |
| 102 | SC\_APPEND\_CMD\_FIT\_ERR\_EID | sc\_atsfunc |  |
| 103 | SC\_VERIFY\_ATS\_EID | sc\_atsfunc, sc\_resetnocds, sc\_stress |  |
| 104 | SC\_VERIFY\_ATS\_NUM\_ERR\_EID | sc\_atsfunc |  |
| 105 | SC\_VERIFY\_ATS\_END\_ERR\_EID |  |  |
| 106 | SC\_VERIFY\_ATS\_PKT\_ERR\_EID | sc\_atsfunc |  |
| 107 | SC\_VERIFY\_ATS\_BUF\_ERR\_EID | sc\_atsfunc |  |
| 109 | SC\_VERIFY\_ATS\_DUP\_ERR\_EID | sc\_atsfunc |  |
| 110 | SC\_VERIFY\_ATS\_MPT\_ERR\_EID | sc\_atsfunc |  |
| 111 | SC\_TABLE\_MANAGE\_ID\_ERR\_EID |  |  |
| 112 | SC\_TABLE\_MANAGE\_RTS\_ERR\_EID |  |  |
| 113 | SC\_TABLE\_MANAGE\_ATS\_ERR\_EID |  |  |
| 114 | SC\_TABLE\_MANAGE\_APPEND\_ERR\_EID |  |  |
| 115 | SC\_STARTRTSGRP\_CMD\_INF\_EID | sc\_rtsfunc |  |
| 116 | SC\_STARTRTSGRP\_CMD\_ERR\_EID | sc\_rtsfunc |  |
| 117 | SC\_STOPRTSGRP\_CMD\_INF\_EID | sc\_rtsfunc |  |
| 118 | SC\_STOPRTSGRP\_CMD\_ERR\_EID | sc\_rtsfunc |  |
| 119 | SC\_DISRTSGRP\_CMD\_INF\_EID | sc\_rtsfunc |  |
| 120 | SC\_DISRTSGRP\_CMD\_ERR\_EID | sc\_rtsfunc |  |
| 121 | SC\_ENARTSGRP\_CMD\_INF\_EID | sc\_rtsfunc |  |
| 122 | SC\_ENARTSGRP\_CMD\_ERR\_EID | sc\_rtsfunc |  |