|  |  |
| --- | --- |
| DocCoverBackground | CORE FLIGHT SYSTEM  SCHEDULER  BUILD 2.2.1.0  FLIGHT SOFTWARE BUILD VERIFICATON  TEST REPORT  Flight Software Branch – Code 582  Version 1.0 |

Signatures

Submitted by:



Approved by:



Plan Update History

| Version | Date | Description | Affected Pages |
| --- | --- | --- | --- |
| Draft | 7/3/2017 | Draft for review | 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 6

3 Build Verification Test Preparation 7

3.1 ScenArio Development 7

3.2 Procedure Development and Execution 7

3.3 Test Products 7

4 Build Verification Test Execution 8

4.1 Testbed Overview 8

4.2 Requirements Verification Matrix 9

4.3 Requirements Partially Tested 9

4.4 Requirements/Functionality Deferred 9

4.5 Requirements/Functionality Deferred for Mission Testing 9

5 Build Verficiaton Test Results 10

5.1 Overall Assessment 10

5.2 Procedure Description 10

5.3 Analysis Requirements Verification 11

5.4 DCRs 11

5.4.1 DCRs Verified 11

5.4.2 Outstanding DCRs 11

5.5 Notes 12

Appendix A - RTTM 13

Appendix B - Command, Telemetry, and Events Verification Matrix 14

# Introduction

## Document Purpose

This Test Report describes the test results from the Core Flight System (cFS) Scheduler (SCH) Flight Software (FSW) Test Team build 2.2.1.0 verification testing. It is used to verify that the SCH FSW has been tested in a manner that validates that it satisfies the functional and performance requirements defined within the cFS SCH 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)

* 582-2008-036 cFS Scheduler Requirements Document, Version 1.1
* 582-2008-012 cFS Deployment Guide, Version 3.1

**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 DCRs. 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 have 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\sstrege\AppData\Local\Microsoft\Windows\Temporary%20Internet%20Files\Content.Outlook\L299AK4Q\Operational_Interface) (OI), [Applications](#Application) (APP), and other cFE-based systems.



**Figure 2-1 cFS System Context**

The Scheduler (SCH) application is responsible for implementing a Time Division Multiplexer (TDM) scheduling mechanism for hosted system applications. SCH provides the ability to deterministically schedule application processing at specific times within a 1 second period, AND those scheduling opportunities are synchronized to the Spacecraft Clock.

The figure below shows major interfaces between the Scheduler task and other core Flight Executive (cFE) and core Flight System (cFS) tasks. Note that although it isn’t shown explicitly, all task-to-task communications are accomplished via the cFE Software Bus task.

Inputs to the Scheduler Application include:

1) Major Frame signal to drive the Scheduler Application

2) Minor frame signal,

3) Commands originating from outside sources such as the ground or Stored Command Processor.

Outputs from the Scheduler application include:

1) Schedule Commands which can include Housekeeping requests, wake-up calls etc.

2) Event Messages and

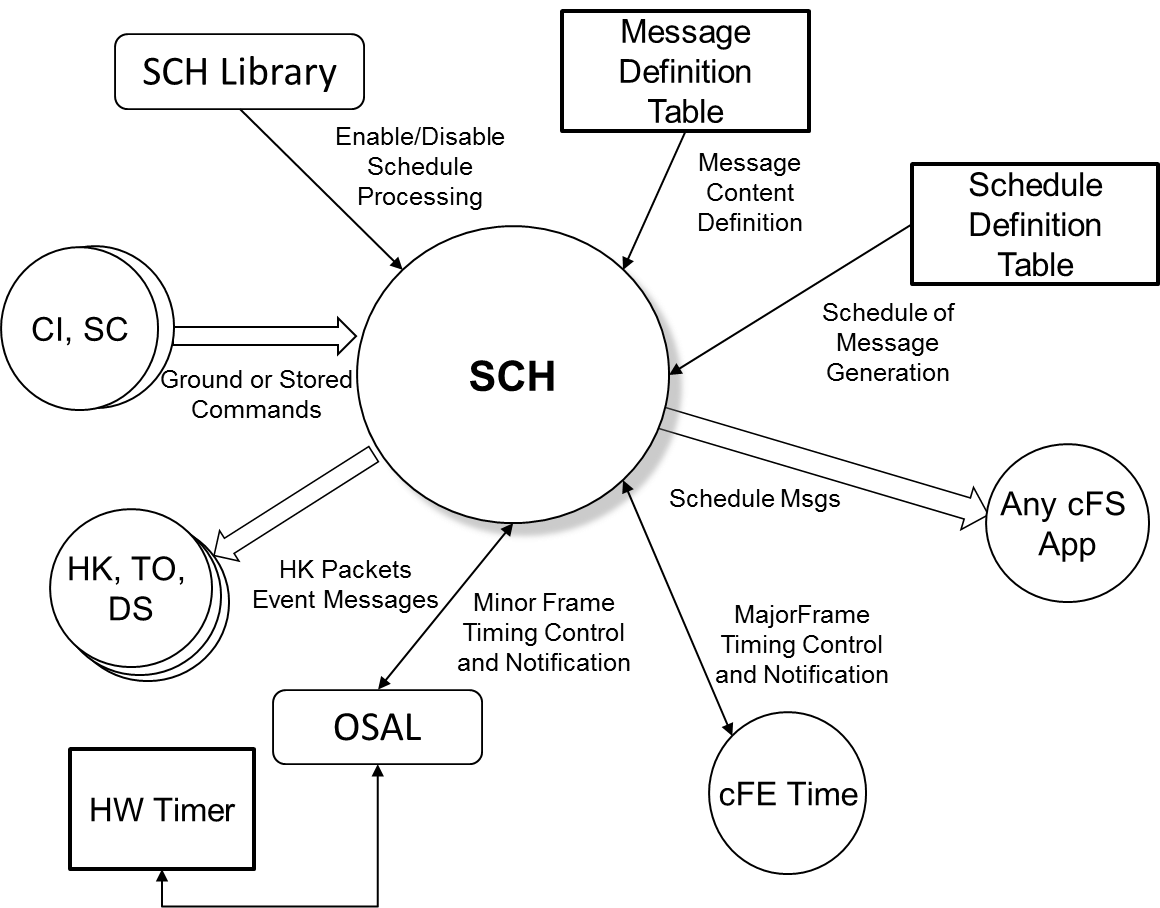
3) Scheduler Housekeeping telemetry

Two tables are used by the Scheduler App:

1) The Schedule Definition Table (SDT) defines that major and minor frame activities that are to be performed.

2) The Message Definition Table (MDT) defines the messages that are sent as part of the activities.

An activity is defined in the SDT by: an Activity Type (currently only supports “Send Message Activity Type”), Activity State (“Unused”, “Enabled”, “Disabled”), Activity Frequency (e.g. – number of seconds between activity executions), Activity Offset (e.g. – phase shift, in seconds, of first execution) and Activity Group/Multi-Group identifiers.



**Figure 2-2 cFS SCH Context**

## Test History

SCH 1.0.1.0 – Build Verification Testing completed 4/2/2009 by Ezinne Uzo-Okoro

SCH 2.0.0.0 – Build Verification Testing completed 9/3/2009 by Walt Moleski

SCH 2.1.2.0 – Build Verification Testing completed 8/16/2011 by Walt Moleski

SCH 2.2.0.0 – Build Verification Testing completed 8/7/2012 by Walt Moleski

SCH 2.2.1.0 – Build Verification Testing completed 7/3/2017 by Walt Moleski

## Testing Overview

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

* 1 test application: tst\_sch
* 4 test procedures: sch\_gencmds, sch\_timing\_analysis, sch\_functional\_test, sch\_lib\_test
* Tests require use of the ASIST Ground Station.

The tst\_sch test application is used to send schedule requests for the output of the CFE\_TIME 1 Hz signal, and cFE and SCH’s housekeeping data. This was useful when performing build verification testing since it provided great control over the sequence of steps. When deployed for a mission, the Scheduler Application would provide this request. In addition, the test application also provides the ability to force Scheduler to miss slots during processing. TST\_SCH contains the following ground commands that are used by the SCH test procedures and/or contained in the Scheduler Message Definition Table:

* TST\_SCH\_NOOP

This command increments the TST\_SCH command counter.

* TST\_SCH\_ResetCtrs

This command sets the TST\_SCH command and error counters to a zero value.

* TST\_SCH\_MissSlots

This command is used to force SCH to skip some slots while processing the minor frame. A slot skip is achieved by calling the CFE\_Time\_ExternalTone() function.

* TST\_SCH\_EnableSchProc

This command is used to call the SCH\_EnableProcessing() function contained in the SCH\_LIB shared library. The call is only performed if the SCH\_LIB\_PRESENCE configuration parameter is set to 1 (one).

* TST\_SCH\_DisableSchProc

This command is used to call the SCH\_DisableProcessing() function contained in the SCH\_LIB shared library. The call is only performed if the SCH\_LIB\_PRESENCE configuration parameter is set to 1 (one).

* TST\_SCH\_GetSchState

This command is used to call the SCH\_GetProcessingState() function contained in the SCH\_LIB shared library. The call is only performed if the SCH\_LIB\_PRESENCE configuration parameter is set to 1 (one).

The 4 SCH test procedures do the following:

|  |  |
| --- | --- |
| **Procedure** | **Description** |
| SCH\_GenCmds | The purpose of this test is to verify the Scheduler (SCH) Application general commands function properly. The NOOP and Reset Counters commands will be tested as well as invalid commands to see if the SCH application handles these appropriately. |
| SCH\_Timing\_Analysis | The purpose of this test is to verify the Scheduler (SCH) Application responds correctly to changes in the schedule and message definition tables. It also verifies the timing of the SCH schedule and message definition table processing. This test also verifies that the SCH application handles anomalies appropriately. |
| SCH\_Functional\_Test | The purpose of this test is to verify the Scheduler (SCH) Application responds correctly and timely to definition changes in the schedule and message table in order to the enabling/disabling of individual activities. This test also verifies that the SCH application handles anomalies appropriately. |
| SCH\_Lib\_Test | The purpose of this test is to verify that the SCH application supports the schedule library functions properly. These functions allow an external application to control the enabling/disabling of the SCH application’s activity processing. |

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. |
| load\_start\_app | Procedure to load and start a user application from the $WORK/apps/cpux directory. |
| 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\_tlmwait | Directive that waits for the specified telemetry condition to be met |
| ftp\_file | To ftp a file to/from the FSW/GSW. |
| get\_tbl\_to\_cvt | Directive that issues the CFE\_TBL\_Dump command and downloads the file to the ground and inserts it in the supplied table telemetry packet. |
| create\_tbl\_file\_from\_cvt | Directive that creates a CFE\_TBL load file from the supplied table telemetry packet. |
| load\_table | Directive that transfers the supplied file to the specified cpu and issues the CFE\_TBL\_Load command. |

## Version Information

|  |  |
| --- | --- |
| Item | Version |
| SCH Requirements | 1.1 |
| SCH Application | 2.2.1.0 |
| TST\_SCH Application | 2.2.0.0 |
| cFE | 6.5.0.0 |
| OSAL | 4.2.0.0 |
| ASIST | 20.2 |
| VxWorks | 6.9 |

# Build Verification Test Preparation

## ScenArio Development

No new scenarios developed for build verification test 2.2.1.0. All scenarios are stored on the MKS server, in cFS-Repository SCH test-and-ground directory within the test-review-packages subdirectory in the Scenarios folder. It should be noted that as SCH requirement evolve these scenarios are not updated to reflect any changes made.

## Procedure Development and Execution

This build test was completed by running 4 test procedures. 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.<ext>.

## Test Products

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

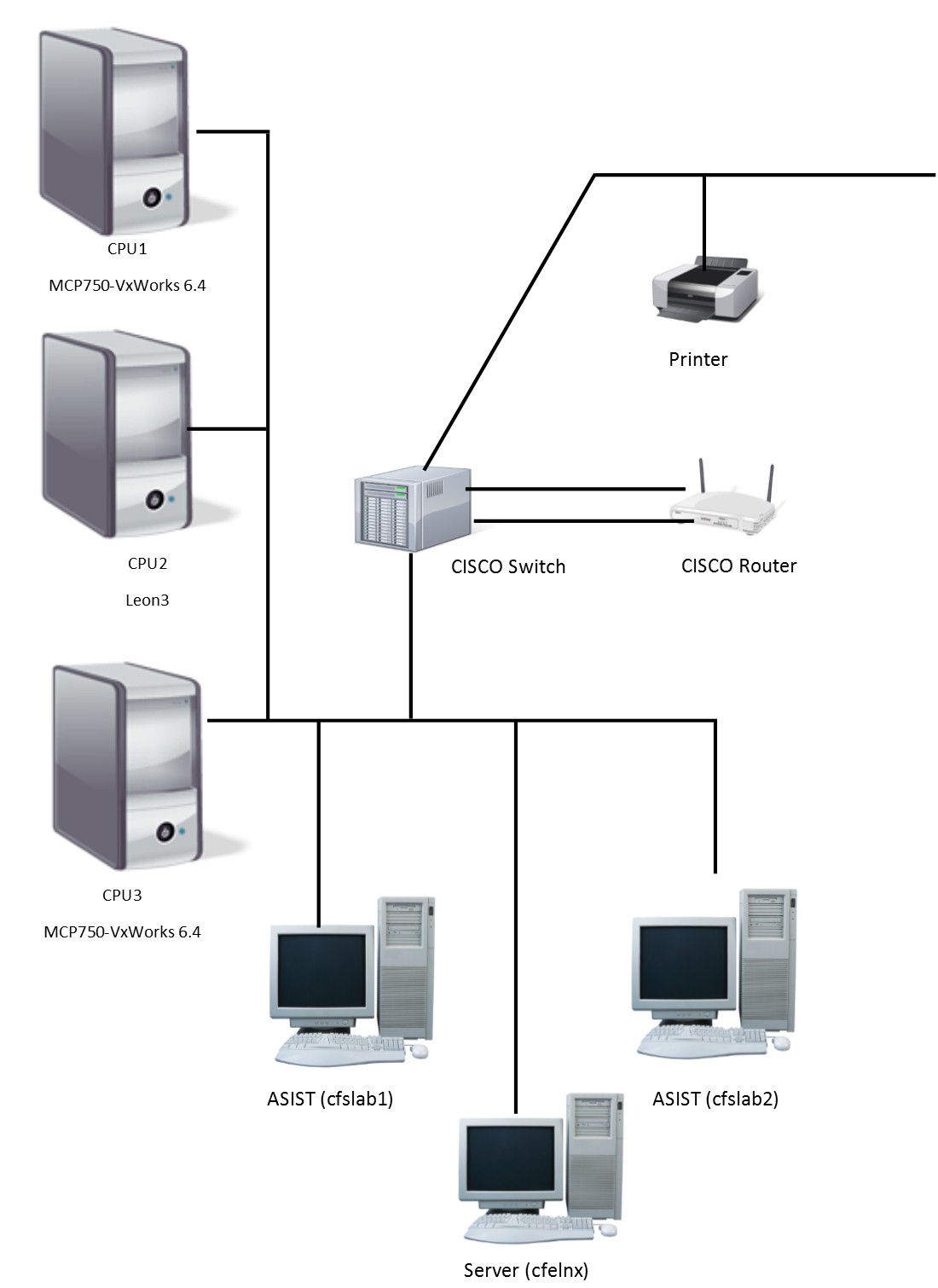
* Logs with the .loge extension list all events sent by the flight software
* Logs with the .logr extension list all requirements that passed validation by demonstration
* Logs with the .logp extension lists all prints that are generated by the test procedure
* Logs with the .logf extension lists everything 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 SCH test-and-ground directory.

# Build Verification Test Execution

## Testbed Overview

SCH 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 9.7g 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

|  |  |
| --- | --- |
|  | Scheduler (SCH) |
| Requirements Tested Passed | 48 |
| Requirements Tested Failed | 0 |
| Requirements Tested Partially | 0 |
| Total Tested | 48 |
| Deferred | 0 |
| Total | 48 |

## Requirements Partially Tested

No requirements were partially tested.

## Requirements/Functionality Deferred

No requirements were deferred.

## Requirements/Functionality Deferred for Mission Testing

No functionality was deferred to mission testing.

# Build Verficiaton Test Results

## Overall Assessment

During this build test of the SCH Application the software behaved as expected with the exception of the OS Timer Accuracy, which prompted the Scheduler Application to send slots at a minimum of 16.667 ms (10ms is expected). This however, is an artifact of the test environment and is not caused by the SCH Application. Below is a summary of the results:

* 47 requirements passed via demonstration
* 1 requirement was validated by analysis.
* 4 DCRs were verified.

## Procedure Description

| **Procedure** | **Description** | **Requirements tested** |
| --- | --- | --- |
| SCH\_GenCmds | The purpose of this test is to verify the Scheduler (SCH) general commands function properly. The NOOP and Reset Counters commands will be tested as well as invalid commands to see if the SCH application handles these appropriately. | SCH1000; SCH1001; SCH1002; SCH1004; SCH1005; SCH8000; SCH9000; SCH9001 |
| SCH\_Timing\_Analysis | The purpose of this test is to verify the Scheduler (SCH) Application responds correctly to changes in the schedule and message definition tables. It also verifies the timing of the SCH schedule and message definition table processing. This test also verifies that the SCH application handles anomalies appropriately. | SCH1002; SCH1004; SCH1005; SCH2000; SCH2001; SCH2002; SCH3000; SCH3001; SCH3001.1; SCH3002; SCH3003; SCH3003.1; SCH4000; SCH4000.1; SCH4000.2; SCH4001; SCH4001.1; SCH4001.2; SCH4002; SCH4002.1; SCH4003; SCH4003.1; SCH4004; SCH4005; SCH8000; SCH9000; SCH9001 |
| SCH\_Functional\_Test | The purpose of this test is to verify the Scheduler (SCH) Application responds correctly and timely to definition changes in the schedule and message table in order to the enabling/disabling of individual activities. This test also verifies that the SCH application handles anomalies appropriately. | SCH2000; SCH2002; SCH2003; SCH2003.1; SCH2003.2; SCH2003.3; SCH2003.4; SCH2004; SCH2005; SCH2006; SCH2006.1; SCH2006.2; SCH2006.3; SCH2006.4; SCH2006.5; SCH2007; SCH2007.1; SCH2007.2; SCH2007.3; SCH2007.4; SCH2007.5; SCH8000; SCH9000; SCH9001 |
| SCH\_Lib\_Test | The purpose of this test is to verify that the SCH application supports the schedule library functions properly. These functions allow an external application to control the enabling/disabling of the SCH application’s activity processing. | SCH8000; SCH9000; SCH9001 |

## Analysis Requirements Verification

The following requirements were verified using analysis:

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Description** | **Status** | **Justification** |
| SCH2001 | The Schedule Definition Table shall schedule activities with a minimum minor frame resolution of <PLATFORM\_DEFINED, 10> milliseconds and a major frame resolution of <PLATFORM\_DEFINED, 1> | Pass | The Timing Analysis test procedure scheduled a large amount of NOOP commands in Step 2.1. The resolution can be determined by viewing the log file. |

## DCRs

No new DCRs were generated during SCH 2.2.1.0 testing

### DCRs Verified

The following DCRs were verified during testing:

| **DCR** | **Description** | **Test Method** | **Test Approach** |
| --- | --- | --- | --- |
| 4157 | Update commands in sch\_platform\_cfg.h to fix comments | Inspection | Verified the stated fix was made to the file submitted with the DCR. |
| 4174 | |  | | --- | | SCH tables have CCSDS primary packet header that must be Big Endian on Little Endian platforms | | Inspection | The sch\_def\_msgtbl.c file contained the stated macro call on CCSDS primary headers. |
| 4248 | SCH – Scheduler task source module sch\_custom.c has local definitions of PSP functions. | Inspection | The stated PSP functions were removed from the files submitted with this DCR. |
| 145925 | SCH – CFE\_EVS\_SendEvent Format Warnings | Demonstration | The compile process produced not warnings for the SCH application. |

### Outstanding DCRs/Trac Tickets

The following outstanding DCRS are written against the Scheduler application:

Note: Trac ticket references are proceeded with a ‘#’ character.

|  |  |
| --- | --- |
| **DCR/Trac #** | **Description** |
| #21 | SCH cmake table integration. The cFE 6.5 release adds better support for building table objects from CMakeLists files.  This ticket contains an update to the SCH app to use this feature to build its table files in addition to the normal SCH build. |
| #35 | SCH occasionally hangs when CFS is stopped. When the CFS process receives a SIGINT (ctrl-c), the SCH\_MinorFrameCallback is called by the pthread\_cancel. Often this will cause that callback to hang on a lock. |
| #45 | SCH: Remove dependencies on cfe\_platform\_cfg |
| #47 | Inaccuracies in Scheduler Minor Frame Slot Timing/Processing |
| #84 | SCH Does Not Protect Scheduler Table Slots from Being Commanded. SCH currently allows any slot within the Scheduler Table to be commanded to an enable/disable state. |
| #85 | SCH Minor Frame Interrupt Processed Before Receipt of Startup Sync. The default custom.c file includes the create timer function in the early initialization function. On NICER this caused the minor frame interrupt to be processed before the SCH application has received the startup sync. |
| #86 | Consider Moving Group Macro Definitions to a Header File. Moving these macro definitions to a header file would allow ground system command definitions to include the actual group definitions vs. redefining them which can be error prone. |
| #87 | Should the Scheduler Table's Types be Configurable? MMS Made the SCH table critical. The current CFS SCH options do not allow the Scheduler's table's types to be configurable. |
| #88 | SCH Timer Configuration Issues. The SCH "Minor Frame MET Sync logic" is very dependent on the accuracy of the timer being used. There currently is no way for user's to configure the "Minor Frame MET Sync logic" to work with a variety of different timers. |
| 4121 | SCH - Add Trick Simulation Support (JSC Request) |

## Notes

The SCH\_Lib\_Test procedure was executed twice to test the default state of the SCH Library being present and a second time with the SCH Library not present. The second test required the SCH application to be recompiled with the configuration parameter SCH\_LIB\_PRESENCE set to 0 (zero).

1. RTTM

The SCH Build 2.2.1.0 RTTM can be found on the MKS server cFS-Repository and in the release package in the SCH test-and-ground directory results folder.

1. Command, Telemetry, and Events Verification Matrix

|  |  |  |
| --- | --- | --- |
| **Command** | **Test Procedure(s)** | **Notes/Comments** |
| SCH\_NOOP | sch\_gencmds |  |
| SCH\_RESETCTRS | sch\_funnctional\_test; sch\_gencmds |  |
| SCH\_ENABLEENTRY | sch\_funnctional\_test; sch\_timing\_analysis |  |
| SCH\_DISABLEENTRY | sch\_funnctional\_test; sch\_timing\_analysis |  |
| SCH\_DISABLEGROUP | sch\_funnctional\_test; sch\_lib\_test; sch\_timing\_analysis |  |
| SCH\_ENABLEGROUP | sch\_funnctional\_test; sch\_lib\_test; sch\_timing\_analysis |  |
| SCH\_ENABLESYNC | sch\_funnctional\_test; sch\_timing\_analysis |  |
| SCH\_SENDDIAG | sch\_timing\_analysis |  |

| **Telemetry** | **Test Procedure(s)** | **Notes/Comments** |
| --- | --- | --- |
| SCH\_CMDPC | sch\_gencmds; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_CMDEC | sch\_gencmds; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_SYNCTOMET | sch\_gencmds; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_MAJORFRAMESOURCE | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_ACTSUCCESSCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_ACTFAILURECTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_SLOTPROCCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_SKIPSLOTCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_MULTSLOTCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_SAMESLOTCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_BADTBLDATACTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_TBLPASSVERIFYCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_TBLFAILVERIFYCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_TBLPROCCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_VALIDMFCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_MISSMFCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_UNEXPCTDMFCTR | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_MINORSINCETONE | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_NEXTSLOT | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_LASTSYNCMETSLOT | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_IGNOREMF | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_UNEXPCTDMAJORFRAME | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| SCH\_ENTRYSTATES[] |  |  |
| SCH\_MSGIDS[] |  |  |

| **Table Telemetry** | **Test Procedure(s)** | **Notes/Comments** |
| --- | --- | --- |
| SCH\_DefaultMessageTable[].MessageBuffer[] | sch\_badmdttables; sch\_mdtloadfile |  |
| SCH\_DefaultScheduleTable[].EnableState | sch\_badsdttbls; sch\_sdt2; sch\_sdt3; sch\_sdtloadfile; |  |
| SCH\_DefaultScheduleTable[].SE\_Type | sch\_badsdttbls; sch\_functional\_test; sch\_sdt2; sch\_sdt3; sch\_sdtloadfile; |  |
| SCH\_DefaultScheduleTable[].SE\_Frequency | sch\_badsdttbls; sch\_functional\_test; sch\_sdt2; sch\_sdt3; sch\_sdtloadfile; |  |
| SCH\_DefaultScheduleTable[].Remainder | sch\_badsdttbls; sch\_sdt2; sch\_sdt3; sch\_sdtloadfile; |  |
| SCH\_DefaultScheduleTable[].MessageIndex | sch\_badsdttbls; sch\_sdt2; sch\_sdt3; sch\_sdtloadfile; |  |
| SCH\_DefaultScheduleTable[].GroupData | sch\_badsdttbls; sch\_sdt2; sch\_sdt3; sch\_sdtloadfile; |  |

|  | **Event Message Ids** | **Test Procedure(s)** | **Notes/Comments** |
| --- | --- | --- | --- |
| **1** | SCH\_INITSTATS\_INF\_EID | sch\_gencmds; sch\_functional\_test; sch\_lib\_test; sch\_timing\_analysis |  |
| **2** | SCH\_APP\_EXIT\_EID |  |  |
| **3** | SCH\_CR\_PIPE\_ERR\_EID |  |  |
| **4** | SCH\_SUB\_HK\_REQ\_ERR\_EID |  |  |
| **5** | SCH\_SUB\_GND\_CMD\_ERR\_EID |  |  |
| **7** | SCH\_SDT\_REG\_ERR\_EID |  |  |
| **8** | SCH\_MDT\_REG\_ERR\_EID |  |  |
| **9** | SCH\_SDT\_LOAD\_ERR\_EID |  |  |
| **10** | SCH\_MDT\_LOAD\_ERR\_EID |  |  |
| **11** | SCH\_ACQ\_PTR\_ERR\_EID |  |  |
| **12** | SCH\_MINOR\_FRAME\_TIMER\_CREATE\_ERR\_EID |  |  |
| **13** | SCH\_MINOR\_FRAME\_TIMER\_ACC\_WARN\_EID | sch\_gencmds; sch\_functional\_test; sch\_lib\_test; sch\_timing\_analysis |  |
| **14** | SCH\_MAJOR\_FRAME\_SUB\_ERR\_EID |  |  |
| **15** | SCH\_SEM\_CREATE\_ERR\_EID |  |  |
| **16** | SCH\_SAME\_SLOT\_EID | sch\_functional\_test; sch\_gencmds; sch\_lib\_test; |  |
| **17** | SCH\_SKIPPED\_SLOTS\_EID | sch\_functional\_test; |  |
| **18** | SCH\_MULTI\_SLOTS\_EID | sch\_functional\_test; |  |
| **19** | SCH\_CORRUPTION\_EID |  |  |
| **20** | SCH\_PACKET\_SEND\_EID |  |  |
| **21** | SCH\_NOISY\_MAJOR\_FRAME\_ERR\_EID | sch\_functional\_test; sch\_lib\_test; |  |
| **30** | SCH\_SCHEDULE\_TBL\_ERR\_EID | sch\_functional\_test; |  |
| **31** | SCH\_SCHEDULE\_TABLE\_EID | sch\_functional\_test; sch\_timing\_analysis |  |
| **32** | SCH\_MESSAGE\_TBL\_ERR\_EID | sch\_functional\_test; |  |
| **33** | SCH\_MESSAGE\_TABLE\_EID | sch\_functional\_test; |  |
| **40** | SCH\_NOOP\_CMD\_EID | sch\_gencmds; sch\_lib\_test; sch\_functional\_test; sch\_timing\_analysis |  |
| **41** | SCH\_RESET\_CMD\_EID | sch\_gencmds; sch\_functional\_test; |  |
| **42** | SCH\_ENABLE\_CMD\_EID | sch\_functional\_test; sch\_timing\_analysis |  |
| **43** | SCH\_DISABLE\_CMD\_EID | sch\_functional\_test; sch\_timing\_analysis |  |
| **44** | SCH\_ENA\_GRP\_CMD\_EID | sch\_functional\_test; sch\_lib\_test; sch\_timing\_analysis |  |
| **45** | SCH\_DIS\_GRP\_CMD\_EID | sch\_functional\_test; sch\_lib\_test; sch\_timing\_analysis |  |
| **46** | SCH\_ENA\_SYNC\_CMD\_EID | sch\_functional\_test; sch\_timing\_analysis |  |
| **47** | SCH\_SEND\_DIAG\_CMD\_EID | sch\_timing\_analysis |  |
| **50** | SCH\_ENABLE\_CMD\_ARG\_ERR\_EID | sch\_timing\_analysis |  |
| **51** | SCH\_ENABLE\_CMD\_ENTRY\_ERR\_EID | sch\_timing\_analysis |  |
| **52** | SCH\_DISABLE\_CMD\_ARG\_ERR\_EID | sch\_timing\_analysis |  |
| **53** | SCH\_DISABLE\_CMD\_ENTRY\_ERR\_EID | sch\_timing\_analysis |  |
| **54** | SCH\_ENA\_GRP\_CMD\_ERR\_EID | sch\_timing\_analysis |  |
| **55** | SCH\_ENA\_GRP\_NOT\_FOUND\_ERR\_EID | sch\_timing\_analysis |  |
| **56** | SCH\_DIS\_GRP\_CMD\_ERR\_EID | sch\_timing\_analysis |  |
| **57** | SCH\_DIS\_GRP\_NOT\_FOUND\_ERR\_EID | sch\_timing\_analysis |  |
| **58** | SCH\_CC\_ERR\_EID | sch\_gencmds; |  |
| **59** | SCH\_MD\_ERR\_EID |  |  |
| **60** | SCH\_CMD\_LEN\_ERR\_EID | sch\_gencmds; sch\_timing\_analysis |  |