Code 582
Flight Software Branch

CORE FLIGHT SYSTEM CHECKSUM BUILD 2.4.0.0

FLIGHT SOFTWARE BUILD VERIFICATON TEST REPORT

Flight Software Branch – Code 582

Version 1.0

SIGNATURES

Submitted by:

WALTER MOLESKI

Digitally signed by WALTER MOLESKI
DN: c=US, o=U.S. Government, ou=NASA, ou=People, cn=WALTER MOLESKI, 0.9.2342.19200300.100.1.1=wmoleski
Date: 2017.04.24 12:36:52 -04'00'

X

Walt Moleski/582
cFS Flight Software Tester

Approved by:

SUSANNE STREGE

Digitally signed by SUSANNE STREGE
DN: c=US, o=U.S. Government, ou=NASA, ou=People, cn=SUSANNE STREGE, 0.9.2342.19200300.100.1.1=sstrege
Date: 2017.04.24 10:56:08 -04'00'

X

Susanne Strege/582
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 DOCUMENT PURPOSE

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

1.2 APPLICABLE DOCUMENTS

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

Parent Documents (Mission and FSW)

- 582-2008-036 cFS Checksum Requirements Document, Version 1.3
- 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

1.3 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

1.4 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 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.

2 OVERVIEW

2.1 FLIGHT DATA SYSTEM CONTEXT

Figure 2-1 illustrates the cFS system context. The cFE interfaces to five external systems: an Operating System (OS), a Hardware Platform (HP), an Operational Interface (OI), Applications (APP), and other cFE-based systems.

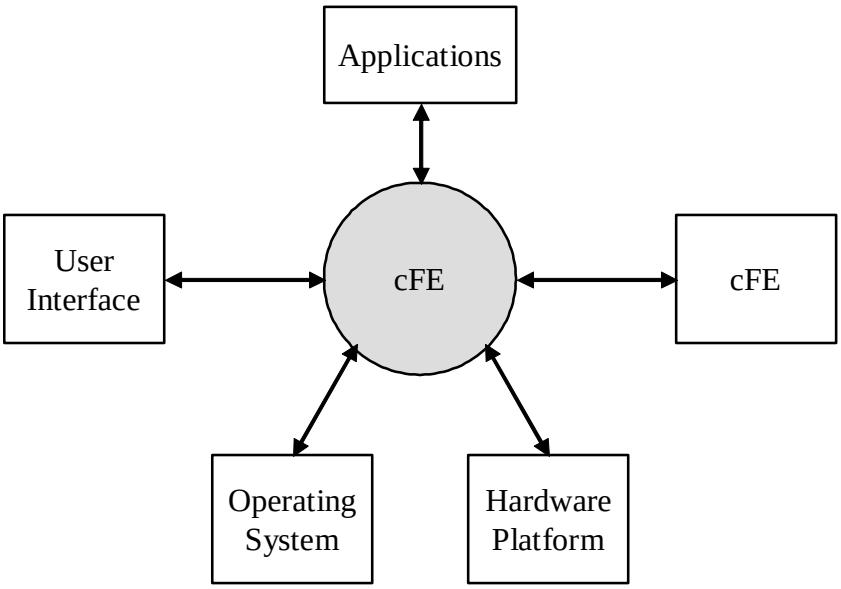

Figure 2-1 cFS System Context

The Checksum (CS) application is responsible for calculating and monitoring checksums or Cyclical Redundancy Checking (CRC) for static memory. For the purposes of this document, the term "checksum" does not dictate an algorithm but merely refers to the act of verifying memory.
The Checksum (CS) application is responsible for monitoring checksums for the following regions:

1. Non-volatile Memory (eg. EEPROM)
2. Volatile static memory
   - · OS code segment
   - · cFE code segment
   - · Application's code segment
   - · Tables
   - · User-Defined Memory ("Memory")

In order for the CS application to further decompose the regions listed above, CS will rely on various tables to supply the details. These tables will be populated by software system engineers or other software personnel. CS will, for example, use a table which specifies which Applications to monitor for checksum miscompares. Another table will be used to specify which tables CS should monitor. This type of design allows for the software systems engineers to have greater control and flexibility for defining what to checksum.

The figure below shows major interfaces between the Checksum task and other core Flight Executive (cFE) and core Flight System (cFS) applications. Note that although it isn't shown explicitly, all application-to-

application communications are accomplished via the cFE Software Bus core app. Inputs to the Checksum Application include:

1. Commands to the Checksum Application
2. Addresses of the non-volatile and OS code segments are validated by the Operating System Abstraction Layer (OSAL) / Board Support Package (BSP)
3. Addresses and sizes of the cFE core and the Applications that run on the cFE are provided by the cFE Executive Services (ES).
4. Addresses and sizes of each of the tables to be checksummed are provided by the cFE Table Services.

Outputs from the Checksum application include:

1. Checksum Application housekeeping message
2. Event messages

Tables used by the Checksum Application include:

1. Application code segment Checksum Table
2. Table Checksum Table - specifies the tables that the Checksum App should verify
3. Non-volatile Checksum Table
4. User-Defined Memory Checksum Table

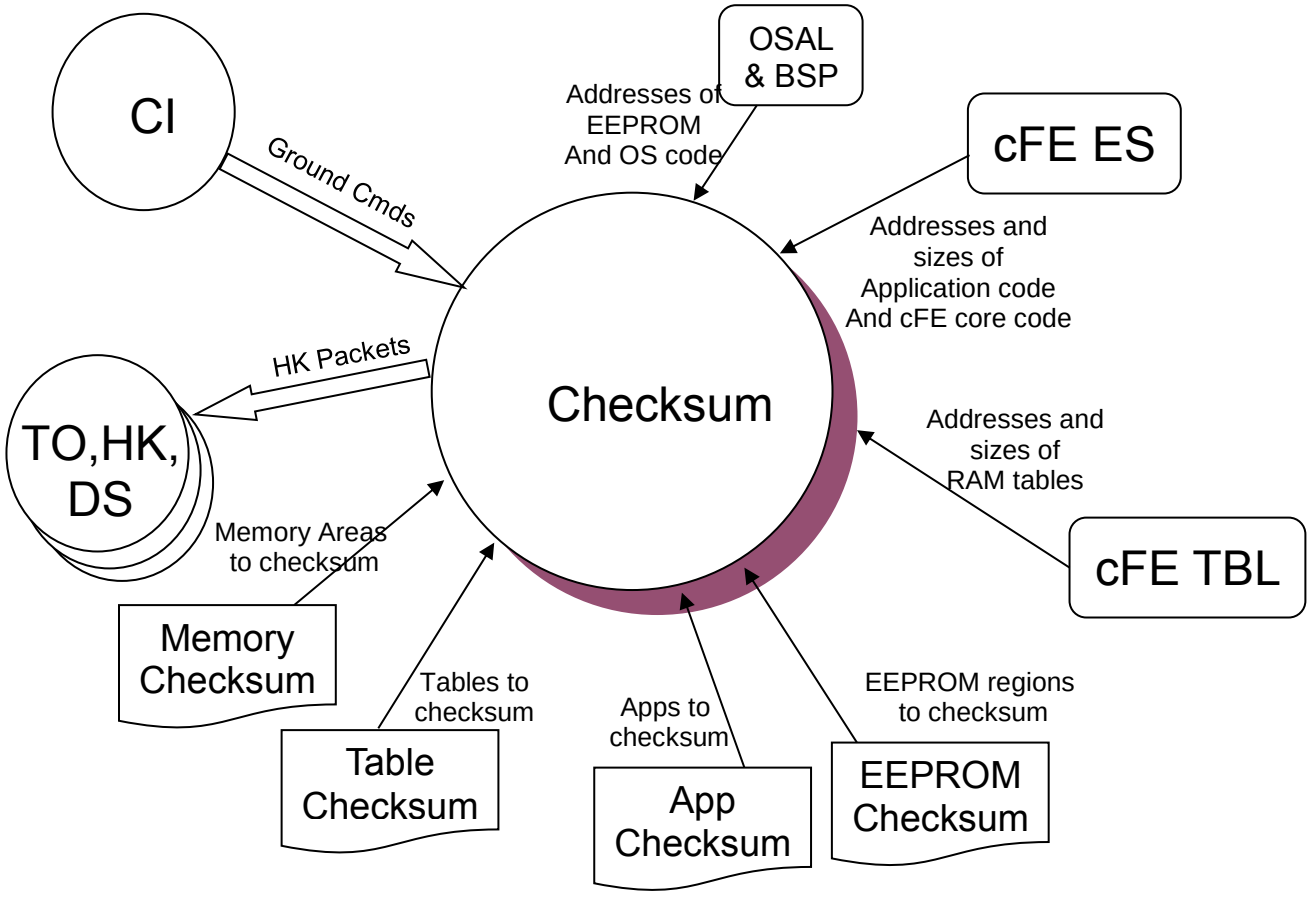

Figure 2-2 cFS CS Context

2.2 TEST HISTORY

CS 1.0.0.0 – Build Verification Testing completed 10/29/2008 by Walt Moleski
CS 2.0.0.0 – Build Verification Testing completed 9/2/2009 by Walt Moleski
CS 2.1.0.0 – Build Verification Testing completed 12/1/2010 by Walt Moleski
CS 2.1.1.0 – Build Verification Testing completed 7/21/2011 by Walt Moleski
CS 2.2.0.0 – Build Verification Testing completed 9/21/2012 by Walt Moleski
CS 2.3.1.0 – Build Verification Testing completed 3/2/2015 by Walt Moleski
CS 2.4.0.0 – Build Verification Testing completed 4/14/2017 by Walt Moleski & Joseph Gurganus

2.3 TESTING OVERVIEW

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

- 2 test applications: tst_cs and tst_cs_memtbl
- 8 main test procedures: cs_gencmds, cs_appcode, cs_corecode, cs_nvmem, cs_table, cs_reset, cs_reset2, cs_usermem
- 19 test procedures that are called by the main procedures to setup the tables are listed below along with an explanation of each procedure.
- Tests require use of the Advanced Spacecraft Integration and System Test software (ASIST) Ground Station.

The tst_cs test application is used to send schedule requests for the output of CS's housekeeping data to the CS application. 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 corrupt the CRC and memory. TST_CS has 8 ground commands that are used by the CS test procedures

- CorruptAppCRC: This command is used to corrupt the CRC of the supplied application. The argument to this command is the application name.
- CorruptTblCRC: This command is used to corrupt the CRC of the supplied table. The argument to this command is the fully-qualified table name.
- CorruptMemCRC: This command is used to corrupt the CRC of the supplied entry in the specified memory table. The arguments to this command are the memory type MemType(uint8) and the EntryID(uint16). The memory types are either EEPROM or User-Defined.
- CorruptOSCRC: This command is used to corrupt the baseline OS CRC.
- CorruptCFECRC: This command is used to corrupt the baseline cFE CRC.
- CorruptEEPROM: This command is used to corrupt a fixed area of the EEPROM memory used by the Checksum test applications in order to simulate a change to memory.
- SetMaxBytes: This command sets the Checksum (CS) application's bytes per cycle parameter to the supplied value in order to simulate segmentation. The argument to this command is NumBytes(uint32).
- SetCounters: This command sets specific CS Housekeeping parameters to a non-zero value. This command is used to test CS_ResetCtr command which resets several housekeeping parameters to zero.

The tst_cs_memtbl test application is used to setup the OS Memory Table required by the CS application for determining the validity of memory addresses and ranges. There are no commands associated with this application and it sends out five tst_cs housekeeping packets upon initialization. This information is needed in order to determine the valid RAM and EEPROM addresses to use for testing CS. Also, this test application must be started before the tst_cs application since it defines the Memory Table that tst_cs utilizes.

These 8 CS test procedures do the following:

Procedure	Description
cs_GenCmds	The purpose of this test is to verify the Checksum (CS) general commands function properly. The NOOP, Reset Counters, Enable/Disable Checksum, and One Shot commands will be tested as well as invalid commands to see if the CS application handles these appropriately.
cs_AppCode	The purpose of this test is to verify the Checksum (CS) Application Code Segments checksumming commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.
cs_CoreCode	The purpose of this test is to verify the Checksum (CS) application OS and cFE Code Segments commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.
cs_NVMem	The purpose of this test is to verify the Checksum (CS) non-volatile memory commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.
cs_Table	The purpose of this test is to verify the Checksum (CS) Application Table checksumming commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.
cs_Reset	The purpose of this test is to verify the Checksum (CS) application initializes the appropriate data items upon any initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This test also verifies that the proper notifications occur if any anomalies exist with the data items stated in the requirements. This procedure is executed twice with different platform configuration parameter settings. The CS_PRESERVE_STATES_ON_PROCESSOR_RESET = TRUE and the 6 States in ENABLED and DISABLED respectively.
cs_Reset2	The purpose of this test is to verify the Checksum (CS) application initializes the appropriate data items upon any initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This procedure is executed twice with different platform configuration parameter settings. The CS_PRESERVE_STATES_ON_PROCESSOR_RESET = FALSE and the 6 States in ENABLED and DISABLED respectively.
cs_UserMem	The purpose of this test is to verify the Checksum (CS) application User-Defined Memory checksumming commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.

The 19 test procedures described in the table below are called by the 6 main test procedures. The purpose of these procedures is to generate the table load files used during BVT.

Procedure	Description
cs_adt1	The purpose of this procedure is to generate the default Application Definition Table.
cs_adt2	The purpose of this procedure is to generate an Application Definition Table containing valid entries with empty entries in between them.
cs_adt3	The purpose of this procedure is to generate an Application Definition Table containing an entry with an invalid state.
cs_adt4	The purpose of this procedure is to generate an empty Application Definition Table.

Procedure	Description
cs_edt1	The purpose of this procedure is to generate the default EEPROM Definition Table.
cs_edt2	The purpose of this procedure is to generate an EEPROM Definition Table that contains several valid entries, and entry with an invalid address, an entry that contains an invalid range, and an entry with an invalid state.
cs_edt3	The purpose of this procedure is to generate an EEPROM Definition Table that contains entries that overlap and empty entries in between valid entries.
cs_edt4	The purpose of this procedure is to generate an empty EEPROM Definition Table.
cs_edt5	The purpose of this procedure is to generate an EEPROM Definition Table that contains only an entry with an invalid state.
cs_mdt1	The purpose of this procedure is to generate the default User-defined Memory Definition Table.
cs_mdt2	The purpose of this procedure is to generate several User-defined Memory Definition Tables. The first contains several valid entries, an entry with an invalid address, an entry with an invalid range and an entry with an invalid state. The second contains the invalid range and state entries and the third contains just the invalid state error.
cs_mdt3	The purpose of this procedure is to generate a User-defined Memory Definition Table that contains entries that overlap and empty entries in between valid entries.
cs_mdt4	The purpose of this procedure is to generate an empty User-defined Memory Definition Table.
cs_mdt5	The purpose of this procedure is to generate the User-defined Memory Definition Table used by the cs_reset test procedure.
cs_tdt1	The purpose of this procedure is to generate the default Tables Definition Table.
cs_tdt2	The purpose of this procedure is to generate a Tables Definition Table containing valid entries with empty entries in between them.
cs_tdt3	The purpose of this procedure is to generate a Tables Definition Table containing an entry with an invalid state.
cs_tdt4	The purpose of this procedure is to generate an empty Tables Definition Table.
cs_tdt5	The purpose of this procedure is to generate the Tables Definition Table used by the cs_reset test procedure.

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
close_data_center	Directive that closes the command port from the ASIST machine to the flight cpu.
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 /s/opr/accounts/cfebx/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.

2.4 VERSION INFORMATION

Item	Version
CS Requirements	1.3
CS Application	2.4.0.0
TST_CS Application	2.4.0.0
cFE	6.5.0.0
OSAL	4.2.0.0
ASIST	20.2
VxWorks	6.9

3 BUILD VERIFICATION TEST PREPARATION

3.1 SCENERIO DEVELOPMENT

There were no new scenarios developed for build verification test 2.4.0.0. All scenarios are stored on the MKS server, in cFS-Repository CS test-and-ground directory within the test-review-packages subdirectory in the Scenarios folder. It should be noted that as CS requirements evolve these scenarios are not updated to reflect any changes made.

3.2 PROCEDURE DEVELOPMENT AND EXECUTION

This build test was completed by running 8 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>_GMT.<ext>.

3.3 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 CS test-and-ground directory.

4 BUILD VERIFICATION TEST EXECUTION

4.1 TESTBED OVERVIEW

CS 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 9.7k 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

4.2 REQUIREMENTS VERIFICATION MATRIX

	Checksum (CS)
Requirements Tested Passed	114
Requirements Tested Failed	0
Requirements Tested Partially	0
Total Tested	114
Deferred	0
Total	114

4.3 REQUIREMENTS PARTIALLY TESTED

No requirements were partially tested.

4.4 REQUIREMENTS/FUNCTIONALITY DEFERRED

No requirements were deferred.

4.5 REQUIREMENTS/FUNCTIONALITY DEFERRED FOR MISSION TESTING

The following functionality was deferred to mission testing:

- RAM was the only physical memory type tested. EEPROM, Compact Flash, SSR memory was not tested. EEPROM testing was done by simulating EEPROM in RAM.

5 BUILD VERFICIATON TEST RESULTS

5.1 OVERALL ASSESSMENT

During this build test of the CS Application the software behaved as expected with several problems still outstanding from previous testing. Below is a summary of the results:

- 113 requirements passed by demonstration
- 0 requirements were validated by analysis.
- 1 requirement was validated by inspection.
- 15 DCRs were verified.

5.2 PROCEDURE DESCRIPTION

Procedure	Description	Requirements tested
CS_GenCmds	The purpose of this test is to verify the Checksum (CS) general commands function properly. The NOOP, Reset Counters, Enable/Disable Checksum, and One Shot commands will be tested as well as invalid commands to see if the CS application handles these appropriately.	CS1000; CS1001; CS1002; CS1003; CS1004; CS1005; CS8000; CS8001; CS8002; CS8002.1; CS8002.2; CS8002.3; CS8003; CS9000; CS9001
CS_AppCode	The purpose of this test is to verify the Checksum (CS) Application Code Segments checksumming commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.	CS1002; CS1003; CS1004; CS4000; CS4000.1; CS4000.2; CS4001; CS4002; CS4003; CS4004; CS4005; CS4005.1; CS4005.2; CS4006; CS4007; CS4008; CS7000; CS8000; CS8001; CS9000; CS9001
CS_CoreCode	The purpose of this test is to verify the Checksum (CS) application OS and cFE Code Segments commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.	CS1002; CS1003; CS1004; CS3000; CS3000.1; CS3002; CS3003; CS3004; CS3004.1; CS3004.2; CS3005; CS3006; CS3006.1; CS3007; CS3008; CS3009; CS3009.1; CS3009.2; CS3010; CS8000; CS8001; CS9000; CS9001
CS_NVMem	The purpose of this test is to verify the Checksum (CS) non-volatile memory commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.	CS1002; CS1003; CS1004; CS2001; CS2001.1; CS2001.2; CS2002; CS2003; CS2004; CS2005; CS2006; CS2006.1; CS2006.2; CS2007; CS2008; CS2009; CS2010; CS3003; CS3008; CS8000; CS8001; CS9000; CS9001

CS_Table	The purpose of this test is to verify the Checksum (CS) Application Table checksumming commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.	CS1002; CS1003; CS1004; CS2003; CS3003; CS3008; CS4002; CS5000; CS5000.1; CS5000.2; CS5000.3; CS5001; CS5002; CS5003; CS5004; CS5005; CS5005.1; CS5005.2, CS5006; CS5007; CS5008; CS6002; CS7000; CS8000; CS8001; CS9000; CS9001
CS_Reset	The purpose of this test is to verify the Checksum (CS) application initializes the appropriate data items upon any initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This test also verifies that the proper notifications occur if any anomalies exist with the data items stated in the requirements. This procedure is executed twice with different platform configuration parameter settings. The CS_PRESERVE_STATES_ON_PROCESSOR_RESET = TRUE and the 6 States in ENABLED and DISABLED respectively.	CS2010; CS4008; CS5008; CS6008; CS9000; CS9001; CS9002; CS9003; CS9003.1; CS9003.2; CS9004; CS9005; CS9005.1; CS9006; CS9006.1; CS9007; CS9007.1; CS9007.2; CS9008; CS9010; CS9011; CS9011.1; CS9011.2; CS9012; CS9013; CS9013.1; CS9014; CS9014.1; CS9015; CS9015.1; CS9015.2
CS_UserMem	The purpose of this test is to verify the Checksum (CS) application User-Defined Memory checksumming commands of the core Flight System (cFS). This test verifies that these commands function properly and that the CS application handles anomalies appropriately.	CS1002; CS1003; CS1004; CS2003; CS3003; CS3008; CS4002; CS5002; CS6000; CS6000.1; CS6000.2; CS6001; CS6002; CS6003; CS6004; CS6005; CS6005.1; CS6005.2; CS6006; CS6007; CS6008; CS6009; CS6009.1; CS7000; CS8000; CS8001; CS9000; CS9001
CS_Reset2	The purpose of this test is to verify the Checksum (CS) application initializes the appropriate data items upon any initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This procedure is executed twice with different platform configuration parameter settings. The CS_PRESERVE_STATES_ON_PROCESSOR_RESET = FALSE and the 6 States in ENABLED and DISABLED respectively.	CS2010; CS4008; CS5008; CS6008; CS9000; CS9001; CS9002; CS9003; CS9004; CS9005; CS9006; CS9007; CS9009;

5.3 ANALYSIS REQUIREMENTS VERIFICATION

No requirements were verified using analysis.

5.4 DCRS

No DCRs were generated during CS 2.4.0.0 testing.

5.4.1 DCRs Verified during Build Testing

The following DCRs were validated during CS 2.4.0.0 Build Testing:

DCR	Description	Test Method	Test Approach
3905	CS – Missing Doxygen in Platform Configuration File	Inspection	The stated doxygen comments were found in the submitted file for the missing macros.
3969	No telemetry in housekeeping indicating that a CS one shot or recompute is in progress	Test Procedure	Telemetry item names were updated and tested where possible in all test procedures.
3981	GPM-IVV-1356 - CS - Missing requirement to test for invalid non-volatile memory segments	Test Procedure / Inspection	Although there was not a requirement that specified invalid table segments, the test procedure contained a test for this situation. The procedure was updated to add the requirement.
3983	GPM-IVV-1355 - CS - Missing requirement for event message that reports new non-volatile memory checksum result	Test Procedure / Inspection	The updated requirements were added to the test procedure.
3984	GPM-IVV-1352 - CS - Missing requirement for Get Entry ID Memory Command	Test Procedure / Inspection	The test procedure contained a test for this situation. The procedure was updated to add the new requirements.
3990	CS Requirements Specify Unconditional Enabling of Checksumming Following Processor Reset	Test Procedure	The cs_reset test procedure was modified to verify these new requirements. The procedure tests regions that are ENABLED and skips those that are DISABLED.
4017	CS - Does Not Allow Missions to Configure Checksum Regions Following a Reset	Test Procedure	The cs_reset test procedure was modified to verify the new requirements added as a result of this and other DCR changes. If the PRESERVE_STATES configuration parameter is set to TRUE, cs_reset should be executed. Otherwise, cs_reset2 should be executed to handle the FALSE case.
4176	CS - Allow One Shot Command to Operate at a Different Rate than the Configured Background Checksum Rate (MMS Request)	Test Procedure	The new argument was added to the command and telemetry databases and tested successfully.
145923	CS - CFE_EVS_SendEvent Format Warnings	Test Procedure	No compiler warnings were generated during the make process.
145935	CS - Integrate and Implement Babelfish Ticket Fixes	Test Procedure	No compiler errors were generated during the make process.
146070	CS Requirements Need to Be Updated To Allow One Shot Command to Operate at Different Rates	Test Procedure / Inspection	The updated requirements were added to the cs_gencmds test.
146106	CS Does Not Check Return Value When Registering for Event Services	Inspection	The code submitted with this DCR was inspected. The stated solution has been implemented.
146109	CS: The commands that contain an EntryID are not 32-bit aligned	Test Procedure	The command database for ASIST was updated, compiled and loaded prior to testing. All commands passed.

146113	CS - Command Definitions Should be Defined in cmds.c	Inspection	The stated solution was found in the source files submitted with this DCR.
146120	CS Sends Error Event Message On Nominal Application Exit	Test Procedure	The cs_reset and cs_reset2 test procedures perform application resets of the CS application. There were no error events generated.

5.4.2 Outstanding DCRs

DCR or Trac #	Description	State
145976	CS Unit Tests: Delete Local Copies of UT-Assert Files After Changes In Those Files Are Released in Actual UT-Assert Library. Local copies of certain UT-Assert library files are being added to the CS unit test directory, because required stub functions were missing. This is a workaround while waiting for a new version release of UT-Assert library where these missing stub functions will eventually be added to the library.	Submitted
4111	CS - Add Trick Simulation Support (JSC Request)	Submitted
3879	Expand CS to compute CRC for each bank of EEPROM	On Hold
4075	The overall EEPROM checksum is in housekeeping telemetry. Add the overall checksums for the other table specified areas: tables, applications, and memory regions.	On Hold

5.5 NOTES

The CS_Reset and CS_Reset2 tests were executed twice with different configuration parameter settings. This was done in order to verify several new requirements that were added as a result of the implementation of the DCRs verified with CS 2.4.0.0. CS_Reset is executed with the CS_PRESERVE_STATES_ON_PROCESSOR_RESET set to TRUE and CS_Reset2 is executed with it set to FALSE. In both cases, the following configuration parameters were set to all Enabled and all Disabled for the respective executions:

CS_OSCS_CHECKSUM_STATE
CS_CFECORE_CHECKSUM_STATE
CS_EEPROM_TBL_POWERON_STATE
CS_MEMORY_TBL_POWERON_STATE
CS_APPS_TBL_POWERON_STATE
CS_TABLES_TBL_POWERON_STATE

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

APPENDIX A - RTTM

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

APPENDIX B - COMMAND, TELEMETRY, AND EVENTS VERIFICATION MATRIX

Command	Test Procedure(s)	Notes/Comments
CS_NOOP	cs_gencmds	
CS_RESETCTRS	cs_gencmds	
CS_OneShot	cs_gencmds	
CS_CancelOneShot	cs_gencmds	
CS_EnableAll	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_table; cs_usermem	
CS_DisableAll	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_table; cs_usermem	
CS_EnableCFECore	cs_corecode	
CS_DisableCFECore	cs_appcode; cs_corecode; cs_nvmem; cs_table; cs_usermem	
CS_ReportCFECore	cs_corecode	
CS_RecomputeCFECore	cs_corecode	
CS_EnableOS	cs_corecode	
CS_DisableOS	cs_appcode; cs_corecode; cs_nvmem; cs_table ; cs_usermem	
CS_ReportOS	cs_corecode	
CS_RecomputeOS	cs_corecode	
CS_EnableEeprom	cs_nvmem	
CS_DisableEeprom	cs_appcode; cs_nvmem; cs_table; cs_usermem	
CS_ReportEeprom	cs_nvmem	
CS_RecomputeEeprom	cs_nvmem	
CS_EnableEepromEntry	cs_nvmem	
CS_DisableEepromEntry	cs_nvmem; cs_usermem	
CS_GetEepromEntry	cs_nvmem	
CS_EnableMemory	cs_usermem	
CS_DisableMemory	cs_appcode; cs_table; cs_usermem	
CS_ReportMemory	cs_usermem	
CS_RecomputeMemory	cs_usermem	
CS_EnableMemoryEntry	cs_usermem	
CS_DisableMemoryEntry	cs_usermem	
CS_GetMemoryEntryID	cs_usermem	
CS_EnableTables	cs_table	
CS_DisableTables	cs_appcode; cs_table; cs_usermem	
CS_ReportTableName	cs_table	
CS_RecomputeTableName	cs_table	
CS_EnableTableName	cs_table	
CS_DisableTableName	cs_table	
CS_EnableApps	cs_appcode	
CS_DisableApps	cs_appcode; cs_table; cs_usermem	
CS_ReportAppName	cs_appcode	

CS_RecomputeAppName	cs_appcode	
CS_EnableAppName	cs_appcode	
CS_DisableAppName	cs_appcode	

Telemetry	**Test Procedure(s)**	**Notes/Comments**
CS_CMDPC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_CMDEC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_State	cs_corecode; cs_gencmds	
CS_EepromState	Cs_appcode; cs_table; cs_usermem	
CS_MemoryState	cs_appcode; cs_nvmem; cs_table	
CS_AppState	cs_nvmem; cs_table	
CS_TableState	Cs_appcode; cs_nvmem	
CS_OSState	cs_corecode; cs_nvmem; cs_usermem	
CS_CFECoreState	cs_corecode; cs_nvmem; cs_usermem	
CS_ChildTaskInUse	Cs_appcode; cs_gencmds; cs_nvmem; cs_table; cs_usermem	
CS_OneShotTaskInUse	Cs_appcode; cs_gencmds; cs_nvmem; cs_table; cs_usermem	
CS_EepromEC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_MemoryEC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_AppEC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_TableEC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_CFECoreEC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_OSEC	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
CS_CurrTable		
CS_CurrEntryInTable		
CS_EepromBaseline	cs_reset	
CS_OSBaseline	cs_corecode; cs_reset	
CS_CFECoreBaseline	cs_corecode; cs_reset	
CS_LastOneShotAddr		
CS_LastOneShotSize		
CS_LastOneShotCRC		

CS_PassCtr	cs_corecode; cs_gencmds; cs_reset	

Table Telemetry	**Test Procedure(s)**	**Notes/Comments**
CS_APP_DEF_Table.State	cs_appcode; cs_reset	There are CS_MAX_NUM_APP_TABLE_ENTRIES instances of this variable.
CS_APP_DEF_Table.Name	cs_appcode; cs_reset	
CS_APP_RESULT_Table.State	cs_appcode; cs_reset	
CS_APP_RESULT_Table.ComputedYet	cs_appcode	
CS_APP_RESULT_Table.StartAddr		
CS_APP_RESULT_Table.NumBytes		
CS_APP_RESULT_Table.BaselineCRC	cs_appcode; cs_reset	
CS_APP_RESULT_Table.ByteOffset	cs_appcode	
CS_APP_RESULT_Table.TempCRC	cs_appcode	
CS_APP_RESULT_Table.Name	cs_appcode; cs_reset	
CS_EEPROM_DEF_Table.State	cs_nvmem; cs_reset	There are CS_MAX_NUM_EEPROM_TABLE_ENTRIES instances of this variable.
CS_EEPROM_DEF_Table.StartAddr	cs_nvmem; cs_reset	
CS_EEPROM_DEF_Table.NumBytes	cs_nvmem; cs_reset	
CS_EEPROM_RESULT_Table.State	cs_nvmem; cs_reset	
CS_EEPROM_RESULT_Table.ComputedYet	cs_nvmem; cs_reset	
CS_EEPROM_RESULT_Table.StartAddr		
CS_EEPROM_RESULT_Table.NumBytes		
CS_EEPROM_RESULT_Table.BaselineCRC	cs_nvmem; cs_reset	
CS_EEPROM_RESULT_Table.ByteOffset		
CS_EEPROM_RESULT_Table.TempCRC		
CS_MEM_DEF_Table.State	cs_reset; cs_usermem	There are CS_MAX_NUM_MEMORY_TABLE_ENTRIES instances of this variable.
CS_MEM_DEF_Table.StartAddr	cs_reset; cs_usermem	
CS_MEM_DEF_Table.NumBytes	cs_reset; cs_usermem	
CS_MEM_RESULT_Table.State	cs_reset; cs_usermem	
CS_MEM_RESULT_Table.ComputedYet	cs_usermem	
CS_MEM_RESULT_Table.StartAddr	cs_usermem	
CS_MEM_RESULT_Table.NumBytes		
CS_MEM_RESULT_Table.BaselineCRC	cs_reset; cs_usermem	
CS_MEM_RESULT_Table.ByteOffset	cs_usermem	
CS_MEM_RESULT_Table.TempCRC	cs_usermem	

CS_TBL_DEF_Table.State	cs_reset; cs_table	There are CS_MAX_NUM_TABLES_TABLE_ENTRIES instances of this variable.
CS_TBL_DEF_Table.Name	cs_reset; cs_table	
CS_TBL_RESULT_Table.State	cs_reset; cs_table	
CS_TBL_RESULT_Table.ComputedYet	cs_table	
CS_TBL_RESULT_Table.StartAddr		
CS_TBL_RESULT_Table.NumBytes	cs_table	
CS_TBL_RESULT_Table.BaselineCRC	cs_reset; cs_table	
CS_TBL_RESULT_Table.ByteOffset	cs_table	
CS_TBL_RESULT_Table.TempCRC	cs_table	
CS_TBL_RESULT_Table.Name	cs_reset; cs_table	
CS_TBL_RESULT_Table.Tbl_Handle		
CS_TBL_RESULT_Table.CSOwned		

Id	Event Message	Test Procedure(s)	Notes/Comments
1	CS_INIT_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
2	CS_NOOP_INF_EID	cs_gencmds	
3	CS_RESET_DBG_EID	cs_gencmds	
4	CS_DISABLE_ALL_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_table; cs_usermem	
5	CS_ENABLE_ALL_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_table; cs_usermem	
6	CS_DISABLE_CFECORE_INF_EID	cs_appcode; cs_corecode; cs_nvmem; cs_table; cs_usermem	
7	CS_ENABLE_CFECORE_INF_EID	cs_corecode	
8	CS_DISABLE_OS_INF_EID	cs_appcode; cs_corecode; cs_nvmem; cs_table; cs_usermem	
9	CS_ENABLE_OS_INF_EID	cs_corecode	
10	CS_BASELINE_CFECORE_INF_EID	cs_corecode	
11	CS_NO_BASELINE_CFECORE_INF_EID		
12	CS_BASELINE_OS_INF_EID	cs_corecode	
13	CS_NO_BASELINE_OS_INF_EID		
14	CS_RECOMPUTE_CFECORE_STARTED_DBG_EID	cs_corecode	
15	CS_RECOMPUTE_CFECORE_CREATE_CHDTASK_ERR_EID		
16	CS_RECOMPUTE_CFECORE_CHDTASK_ERR_EID	cs_corecode	
17	CS_RECOMPUTE_OS_STARTED_DBG_EID	cs_corecode	
18	CS_RECOMPUTE_OS_CREATE_CHDTASK_ERR_EID		
19	CS_RECOMPUTE_OS_CHDTASK_ERR_EID	cs_corecode	
20	CS_ONESHOT_STARTED_DBG_EID	cs_gencmds	
21	CS_ONESHOT_CREATE_CHDTASK_ERR_EID		
22	CS_ONESHOT_CHDTASK_ERR_EID	cs_gencmds	

Id	Event Message	Test Procedure(s)	Notes/Comments
23	CS_ONESHOT_MEMVALIDATE_ERR_EID	cs_gencmds	
24	CS_ONESHOT_CANCELLED_INF_EID	cs_gencmds	
25	CS_ONESHOT_CANCEL_DELETE_CHDTASK_ERR_EID		
26	CS_ONESHOT_CANCEL_NO_CHDTASK_ERR_EID	cs_gencmds	
27	CS_EEPROM_MISCOMPARE_ERR_EID	cs_nvmem; cs_reset;	
28	CS_MEMORY_MISCOMPARE_ERR_EID	cs_usermem	
29	CS_TABLES_MISCOMPARE_ERR_EID	cs_table; cs_reset;	
30	CS_APP_MISCOMPARE_ERR_EID	cs_appcode; cs_reset;	
31	CS_CFECORE_MISCOMPARE_ERR_EID	cs_corecode; cs_reset	
32	CS_OS_MISCOMPARE_ERR_EID	cs_corecode; cs_reset	
33	CS_MID_ERR_EID		
34	CS_CC1_ERR_EID	cs_gencmds	
35	CS_EXIT_ERR_EID		
36	CS_LEN_ERR_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_table; cs_usermem	
37	CS_DISABLE_EEPROM_INF_EID	cs_appcode; cs_corecode; cs_nvmem; cs_table; cs_usermem	
38	CS_ENABLE_EEPROM_INF_EID	cs_nvmem	
39	CS_BASELINE_EEPROM_INF_EID	cs_nvmem	
40	CS_NO_BASELINE_EEPROM_INF_EID		
41	CS_BASELINE_INVALID_ENTRY_EEPROM_ERR_EID	cs_nvmem	
42	CS_RECOMPUTE_EEPROM_STARTED_DBG_EID	cs_nvmem	
43	CS_RECOMPUTE_EEPROM_CREATE_CHDTASK_ERR_EID		
44	CS_RECOMPUTE_INVALID_ENTRY_EEPROM_ERR_EID	cs_nvmem	
45	CS_RECOMPUTE_EEPROM_CHDTASK_ERR_EID	cs_nvmem	
46	CS_ENABLE_EEPROM_ENTRY_INF_EID	cs_nvmem	
47	CS_ENABLE_EEPROM_INVALID_ENTRY_ERR_EID	cs_nvmem	
48	CS_DISABLE_EEPROM_ENTRY_INF_EID	cs_nvmem	
49	CS_DISABLE_EEPROM_INVALID_ENTRY_ERR_EID	cs_nvmem	
50	CS_GET_ENTRY_ID_EEPROM_INF_EID	cs_nvmem	
51	CS_GET_ENTRY_ID_EEPROM_NOT_FOUND_ERR_EID	cs_nvmem	
52	CS_DISABLE_MEMORY_INF_EID	cs_appcode; cs_nvmem; cs_table; cs_usermem	
53	CS_ENABLE_MEMORY_INF_EID	cs_usermem	
54	CS_BASELINE_MEMORY_INF_EID	cs_usermem	
55	CS_NO_BASELINE_MEMORY_INF_EID		
56	CS_BASELINE_INVALID_ENTRY_MEMORY_ERR_EID	cs_usermem	
57	CS_RECOMPUTE_MEMORY_STARTED_DBG_EID	cs_usermem	
58	CS_RECOMPUTE_MEMORY_CREATE_CHDTASK_ERR_EID		
59	CS_RECOMPUTE_INVALID_ENTRY_MEMORY_ERR_EID	cs_usermem	
60	CS_RECOMPUTE_MEMORY_CHDTASK_ERR_EID	cs_usermem	
61	CS_ENABLE_MEMORY_ENTRY_INF_EID	cs_usermem	
62	CS_ENABLE_MEMORY_INVALID_ENTRY_ERR_EID	cs_usermem	
63	CS_DISABLE_MEMORY_ENTRY_INF_EID	cs_usermem	
64	CS_DISABLE_MEMORY_INVALID_ENTRY_ERR_EID	cs_usermem	
65	CS_GET_ENTRY_ID_MEMORY_INF_EID	cs_usermem	
66	CS_GET_ENTRY_ID_MEMORY_NOT_FOUND_ERR_EID	cs_usermem	
67	CS_DISABLE_TABLES_INF_EID	cs_appcode; cs_corecode; cs_nvmem; cs_table; cs_usermem	

Id	Event Message	Test Procedure(s)	Notes/Comments
68	CS_ENABLE_TABLES_INF_EID	cs_table	
69	CS_BASELINE_TABLES_INF_EID	cs_table	
70	CS_NO_BASELINE_TABLES_INF_EID		
71	CS_BASELINE_INVALID_NAME_TABLES_ERR_EID	cs_table	
72	CS_RECOMPUTE_TABLES_STARTED_DBG_EID	cs_table	
73	CS_RECOMPUTE_TABLES_CREATE_CHDTASK_ERR_EID		
74	CS_RECOMPUTE_UNKNOWN_NAME_TABLES_ERR_EID	cs_table	
75	CS_RECOMPUTE_TABLES_CHDTASK_ERR_EID	cs_table	
76	CS_ENABLE_TABLES_NAME_INF_EID	cs_table	
77	CS_ENABLE_TABLES_UNKNOWN_NAME_ERR_EID	cs_table	
78	CS_DISABLE_TABLES_NAME_INF_EID	cs_table	
79	CS_DISABLE_TABLES_UNKNOWN_NAME_ERR_EID	cs_table	
80	CS_DISABLE_APP_INF_EID	cs_appcode; cs_corecode; cs_nvmem; cs_table; cs_usermem	
81	CS_ENABLE_APP_INF_EID	cs_appcode	
82	CS_BASELINE_APP_INF_EID	cs_appcode	
83	CS_NO_BASELINE_APP_INF_EID		
84	CS_BASELINE_INVALID_NAME_APP_ERR_EID	cs_appcode	
85	CS_RECOMPUTE_APP_STARTED_DBG_EID	cs_appcode	
86	CS_RECOMPUTE_APP_CREATE_CHDTASK_ERR_EID		
87	CS_RECOMPUTE_UNKNOWN_NAME_APP_ERR_EID	cs_appcode	
88	CS_RECOMPUTE_APP_CHDTASK_ERR_EID	cs_appcode	
89	CS_ENABLE_APP_NAME_INF_EID	cs_appcode	
90	CS_ENABLE_APP_UNKNOWN_NAME_ERR_EID	cs_appcode	
91	CS_DISABLE_APP_NAME_INF_EID	cs_appcode	
92	CS_DISABLE_APP_UNKNOWN_NAME_ERR_EID	cs_appcode	
93	CS_COMPUTE_APP_NOT_FOUND_ERR_EID	cs_appcode	
94	CS_COMPUTE_TABLES_NOT_FOUND_ERR_EID	cs_table	
95	CS_RECOMPUTE_FINISH_EEPROM_MEMORY_INF_EID	cs_appcode; cs_corecode; cs_nvmem; cs_usermem	
96	CS_RECOMPUTE_ERROR_TABLES_ERR_EID		
97	CS_RECOMPUTE_ERROR_APP_ERR_EID		
98	CS_RECOMPUTE_FINISH_TABLES_INF_EID	cs_table	
99	CS_RECOMPUTE_FINISH_APP_INF_EID	cs_appcode	
100	CS_ONESHOT_FINISHED_INF_EID	cs_gencmds	
101	CS_VAL_EEPROM_STATE_ERR_EID	cs_appcode; cs_reset	
102	CS_VAL_EEPROM_RANGE_ERR_EID	cs_appcode; cs_gencmds; cs_nvmem; cs_reset;	
103	CS_VAL_MEMORY_STATE_ERR_EID	cs_reset; cs_usermem	
104	CS_VAL_MEMORY_RANGE_ERR_EID	cs_appcode; cs_corecode; cs_gencmds; cs_reset; cs_usermem	
105	CS_VAL_TABLES_STATE_ERR_EID	cs_appcode; cs_gencmds; cs_reset; cs_table	
106	CS_VAL_APP_STATE_ERR_EID	cs_appcode; cs_reset;	
107	CS_PROCESS_EEPROM_MEMORY_NO_ENTRIES_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
108	CS_PROCESS_APP_NO_ENTRIES_INF_EID	cs_appcode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	

Id	Event Message	Test Procedure(s)	Notes/Comments
109	CS_PROCESS_TABLES_NO_ENTRIES_INF_EID	cs_appcode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
110	CS_TBL_INIT_ERR_EID		
111	CS_TBL_UPDATE_ERR_EID		
112	CS_INIT_SB_CREATE_ERR_EID		
113	CS_INIT_SB_SUBSCRIBE_HK_ERR_EID		
114	CS_INIT_SB_SUBSCRIBE_BACK_ERR_EID		
115	CS_INIT_SB_SUBSCRIBE_CMD_ERR_EID		
116	CS_INIT_EEPROM_ERR_EID		
117	CS_INIT_MEMORY_ERR_EID		
118	CS_INIT_TABLES_ERR_EID		
119	CS_INIT_APP_ERR_EID		
120	CS_COMPUTE_TABLES_RELEASE_ERR_EID		
121	CS_COMPUTE_TABLES_ERR_EID	cs_table	
122	CS_COMPUTE_APP_ERR_EID	cs_appcode	
123	CS_UPDATE_EEPROM_ERR_EID		
124	CS_UPDATE_MEMORY_ERR_EID		
125	CS_UPDATE_TABLES_ERR_EID		
126	CS_UPDATE_APP_ERR_EID		
127	CS_OS_TEXT_SEG_INF_EID		
128	CS_COMPUTE_APP_PLATFORM_DBG_EID		
129	CS_ENABLE_TABLE_DEF_NOT_FOUND_DBG_EID		
130	CS_DISABLE_TABLE_DEF_NOT_FOUND_DBG_EID		
131	CS_ENABLE_APP_DEF_NOT_FOUND_DBG_EID		
132	CS_DISABLE_APP_DEF_NOT_FOUND_DBG_EID		
133	CS_DISABLE_MEMORY_DEF_EMPTY_DBG_EID		
134	CS_ENABLE_MEMORY_DEF_EMPTY_DBG_EID		
135	CS_DISABLE_EEPROM_DEF_EMPTY_DBG_EID		
136	CS_ENABLE_EEPROM_DEF_EMPTY_DBG_EID		
137	CS_VAL_TABLES_DEF_TBL_DUPL_ERR_EID		
138	CS_VAL_TABLES_DEF_TBL_ZERO_NAME_ERR_EID		
139	CS_VAL_TABLES_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
140	CS_VAL_APP_DEF_TBL_DUPL_ERR_EID		
141	CS_VAL_APP_DEF_TBL_ZERO_NAME_ERR_EID	cs_appcode;	
142	CS_VAL_APP_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
143	CS_VAL_MEMORY_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
144	CS_VAL_EEPROM_INF_EID	cs_appcode; cs_corecode; cs_gencmds; cs_nvmem; cs_reset; cs_table; cs_usermem	
145	CS_INIT_CDS_ERR_EID		
146	CS_EXIT_INF_EID		