|  |  |
| --- | --- |
| DocCoverBackground | 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:



Approved by:



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 5

2.3 Testing Overview 5

2.4 Version Information 8

3 Build Verification Test Preparation 9

3.1 Scenerio Development 9

3.2 Procedure Development and Execution 9

3.3 Test Products 9

4 Build Verification Test Execution 10

4.1 Testbed Overview 10

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 Verficiaton Test Results 11

5.1 Overall Assessment 11

5.2 Procedure Description 11

5.3 Analysis Requirements Verification 12

5.4 DCRs 13

5.4.1 DCRs Verified during Build Testing 13

5.4.2 Outstanding DCRs 14

5.5 Notes 14

Appendix A - RTTM 15

Appendix B - Command, Telemetry, and Events Verification Matrix 16

# Introduction

## 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.

## 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

## 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 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:\CSVar1.0.0.0\test_and_ground\results\Operational_Interface) (OI), [Applications](#Application) (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

Checksum

CI

Ground Cmds

HK Packets

OSAL

& BSP

cFE ES

Table

Checksum

Table

Memory Areas

to checksum

Tables to

checksum

Addresses of

EEPROM

And OS code

Addresses and

sizes of

Application code

And cFE core code

cFE TBL

Addresses and

sizes of

RAM tables

TO,HK,

DS

App

Checksum

Table

EEPROM

Checksum

Table

Apps to

checksum

EEPROM regions

to checksum

Memory

Checksum

Table

**Figure 2-2 cFS CS Context**

## 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

## 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. |
| 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. |

## 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 |

# Build Verification Test Preparation

## 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.

## 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>.

## 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.

# Build Verification Test Execution

## 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**

## Requirements Verification Matrix

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

## Requirements Partially Tested

No requirements were partially tested.

## Requirements/Functionality Deferred

No requirements were deferred.

## 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.

# Build Verficiaton Test Results

## 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.

## 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; |

## Analysis Requirements Verification

No requirements were verified using analysis.

## DCRs

No DCRs were generated during CS 2.4.0.0 testing.

### 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. |

### 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 |

## 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.

1. 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.

1. 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 |  |
| 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 |  |
| 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 |  |
| 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 |  |  |