**<MKSID-5088>**



core Flight System (cFS) Checksum (CS) Application

Requirements Document

Version 1.3

**February 26, 2017**

<MKSID-5091>

1.0       Introduction

1.1        Document Purpose

The core Flight System (cFS) Checksum (CS) Application has been developed by the Flight Software Systems Branch (FSB) of the Software Engineering Division (SED) at NASA’s Goddard Space Flight Center.  The purpose of this requirements specification is to define the requirements to be satisfied by the Checksum Application.

This application is developed for re-use.  For this reason, several nomenclatures are used in this document to identify configurations for a mission. The cFS is specified as a multi-platform product. Mission-specific features and customization requirements which are applicable for all platforms are tagged with <MISSION\_DEFINED>.  Platform-specific features and customizations requirements are tagged with either “<PLATFORM\_DEFINED>” or “<OPTIONAL>.”  Additional nomenclature is used along with the tag to specify a cFS default value for the platform-specific feature: “<PLATFORM\_DEFINED, Default\_Value>”.  Reference platforms (single processor and multi-processor architectures) are defined to supply the default cFS application configuration.  These configurations define the “maximum” cFS Application deployments such that any refined deployment is a subset of a reference platform.

1.2        Document Scope

The scope of this document is limited to the specification of requirements for the Checksum Software requirements.  These include functional, performance, qualification, and design requirements.

1.3        Document Organization

This document is organized into three additional sections:

Section 2 gives the Checksum context.

Section 3 documents the Checksum system design decisions and constraints.

Section 4 contains the Checksum functional and performance requirements.

1.4        Relevant Documents

1.4.1     Parent Documents

cFS Checksum Application Heritage Analysis    582-2007-028  
   
1.4.2     Reference Documents

Operating System Abstraction Layer (OSAL) Library API   
cFE Application Developer’s Guide  582-2007-001  
cFE User’s Guide

2.0       **cFS checksum Application 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
   1. OS code segment
   2. cFE code segment
   3. Application’s code segment
   4. Tables
   5. 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 OSAL/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.

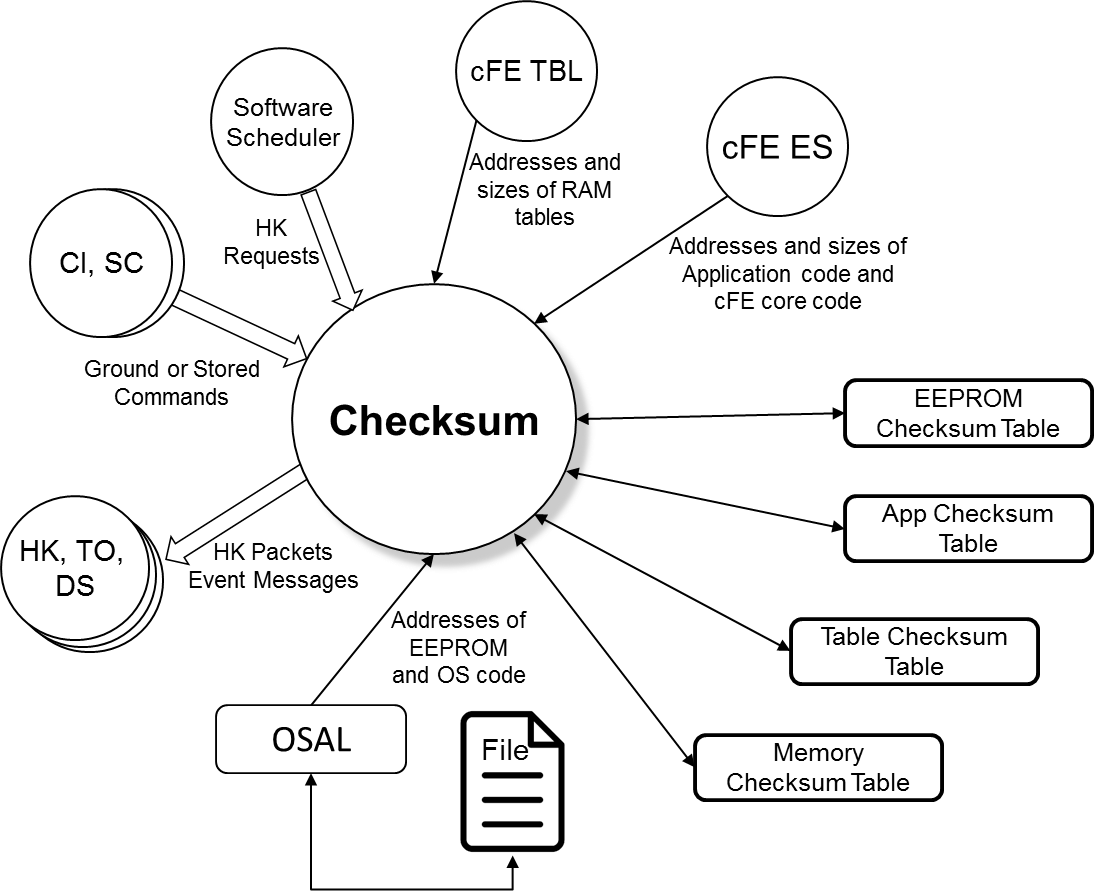
Outputs from the Checksum application include:

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 1.0** – cFS CS Context

   
2.1        Assumptions

The following list summarizes the assumptions made by the cFS Checksum Application:

* cFE API and OSAL are  being used
* Baseline checksums are computed on initialization
* The code segments for the Applications to be checksummed, must be specified to the CS Application
* The Tables to be checksummed must be specified to the CS Application
* The Non-volatile memory regions to be checksummed, must be specified to the CS Application
* Other Memory regions (User-defined memory regions) that are required to be checksummed, must be specified to the CS Application

3.0       Design Specifications

The Checksum Application’s requirements and design are based on the results of the cFS heritage analysis effort.  The results of the heritage analysis are documented in the cFS Checksum Application Heritage Analysis document.  
CS provides the capability to further segregate the non-volatile filesystem region into smaller segments in order to provide better resolution when isolating a checksum miscompare.

**3.1 Design Constraints**

The cFS architecture is based on using a file system.  When files are loaded into non-volatile and volatile memory, it is unknown where the files will be located.  In addition the cFS architecture allows for applications to be started and stopped at runtime, making the static memory regions harder to determine than in various heritage missions which defined static memory segments for the code, data and tables.   
   
Recent experience with the VxWorks file system performance has resulted in the removal of file system checksumming requirements.  The checksum application, however, is being designed such that adding the checksumming of the file system could easily be added as it is very similar to the checksumming of tables.  
   
**4.0       Subsystem Requirements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | ReqID | Text | Rational | Heritage Reference |
| 5860 | CFS-450 | The cFS shall verify the integrity of static memory. | Want to make sure that static remains unchanged. | SDO |

<MKSID-5339>

5.0       Detailed Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | ReqID | Text | Rational | Heritage Reference |
|  |  | 5.1        Basic Requirements  The following requirements are basic requirements of Checksum.  Some of them are included here to avoid repeating these requirements for each applicable requirement. |  |  |
| 5864 | CS1000 | Upon receipt of a No-Op command, CS shall increment the CS Valid Command Counter and generate an event message. | Debug command to verify application is alive. | LRO, SDO |
| 5866 | CS1001 | Upon receipt of a Reset command, CS shall reset the following housekeeping variables to a value of zero:   1. Valid Ground Command Counter 2. Ground Command Rejected Counter 3. Non-volatile CRC Miscompare Counter 4. OS Code Segment CRC Miscompare Counter 5. cFE Code Segment CRC Miscompare Counter 6. Application CRC Miscompare Counter 7. Table CRC Miscompare Counter 8. User-defined Memory CRC Miscompare Counter 9. Checksum Pass Counter (number of passes through all of the checksum areas) | Important for testing and on-orbit flight operations in order to start with a “clean slate”. | LRO, SDO |
| 5868 | CS1002 | For all CS commands, if the length contained in the message header is not equal to the expected length, CS shall reject the command and issue an event message. | Basic command verification in the event of SEU or memory corruption. | LRO, SDO |
| 5870 | CS1003 | If CS accepts any command as valid, CS shall execute the command, increment the CS Valid Command Counter and issue an event message. | Operators require feedback on command execution. | LRO, SDO |
| 5872 | CS1004 | If CS rejects any command, CS shall abort the command execution, increment the CS Command Rejected Counter and issue an error event message. | Operators require feedback on command execution. | LRO, SDO |
| 5874 | CS1005 | CS shall use the <MISSION\_DEFINED> CRC algorithm to compute the CRCs for any segment. | Want to provide the flexibility for a mission to define the CRC algorithm that is used. | New |
|  |  | 5.2        Non-Volatile Memory  These requirements are related to verifying the integrity of the non-volatile memory. Note that non-volatile memory is treated like flat memory. The flat memory can be broken up into segments of any size as a segment is defined by an address and number of bytes. |  |  |
| 5890 | CS2001 | The Checksum App shall calculate CRCs for each Table-Defined Non-volatile segment and compare them against the corresponding baseline Non-volatile segment CRCs if:   1. Checksumming (as a whole) is Enabled 2. Non-volatile segment checksumming is Enabled 3. Checksumming for the individual Non-volatile segment is Enabled | Need to verify Non-volatile memory. Note that each segment within a non-volatile region can have a different size since the segment is defined as an address and number of bytes. | LRO, SDO |
| 5898 | CS2001.1 | If the Non-volatile segment CRC is not equal to the corresponding baseline CRC, CS shall increment the Non-volatile CRC Miscompare Counter and send an event message. | Since the location of files loaded to Non-volatile is unknown apriori, there is no way to determine which Non-volatile segment or segments will be affected. | LRO |
| 146068 | CS2001.2 | If the table-defined segement is invalid, CS shall send an error event message. | Need to alert ground of an invalid segment. | SDO (loosely) |
| 5900 | CS2002 | Upon receipt of a Enable Non-volatile Checksumming command, CS shall enable non-volatile checksumming. | Enable checksumming of all of the non-volatile memory segments defined in the table. | LRO |
| 5902 | CS2003 | Upon receipt of a Disable Non-volatile Checksumming command, CS shall disable non-volatile checksumming. | Disable checksumming of all of the non-volatile memory segments defined in the table. | LRO |
| 5904 | CS2004 | Upon receipt of a Enable Non-volatile Segment command, CS shall enable checksumming of the command-specified non-volatile segment. | Enable checksumming of a particular segment. | LRO |
| 5906 | CS2005 | Upon receipt of a Disable Non-volatile Segment command, CS shall disable checksumming of the command-specified non-volatile segment. | Disable checksumming of a particular segment. | LRO |
| 5908 | CS2006 | Upon receipt of a Recompute Non-volatile Checksum Segment command, CS shall:   1. Recompute the baseline checksum for the command-specified non-volatile segment 2. Set the Recompute In Progress Flag to TRUE | Would be used after non-vol memory is updated in order to regenerate the baseline. | LRO |
| 5910 | CS2006.1 | If CS is already processing a Recompute CRC command or a One Shot CRC command, CS shall reject the command. | Both the One Shot and Recompute are done in a background task so only one may be performed at a time. |  |
| 146066 | CS2006.2 | Once the baseline CRC is computed, CS shall:   1. Generate an informational event message containing the baseline CRC 2. Set the Recompute In Progress Flag to FALSE | Gives the ground indication not only that the CRC baseline was calculated but what the value is. | New |
| 5912 | CS2007 | Upon receipt of a Report Non-volatile Checksum Segment command, CS shall send an event message containing the baseline checksum for the command-specified non-volatile segment. | Provides the ground with the baseline being used. | LRO |
| 5914 | CS2008 | Upon receipt of a Get Non-volatile Checksum Segment command, CS shall send an event message containing the segment number for the command-specified non-volatile address. | Provides the ground with ability to map the address to segment. Helpful since other commands use segment ID. | LRO |
| 5916 | CS2009 | If a command-specified segment is invalid (for any of the non-volatile memory commands where segment is a command argument), CS shall reject the command and send an event message. | Need to handle the case where an invalid segment is specified for any of the non-volatile commands. | LRO |
| 5918 | CS2010 | CS shall provide the ability to dump the baseline CRCs and status for the non-volatile memory segments via a dump-only table. | Need the ability to get all of the non-volatile checksums. Easiest to use the cFE Table services dump-only table feature. | New |
|  |  | **5.3        Volatile Memory – OS and cFE Code Segments**  The Checksum Application provides the ability to checksum the OS and cFE Code Segments. |  |  |
| 5942 | CS3000 | Checksum shall calculate CRC for the OS code segment and compare them against the corresponding baseline OS code segment CRC if:   1. Checksumming (as a whole) is Enabled 2. OS segment checksumming is Enabled | Need to verify the OS code segment. | New |
| 5950 | CS3000.1 | If the OS code segment CRC is not equal to the baseline OS code segment CRC, CS shall increment the OS Code Segment CRC Miscompare Counter and send an event message. |  | New |
| 5952 | CS3002 | Upon receipt of an Enable OS code segment command, CS shall enable checksumming of the OS Code segment. | Enable checksumming of the OS code segment. | New |
| 5954 | CS3003 | Upon receipt of a Disable OS code segment command, CS shall Disable checksumming of the OS Code segment. | Disable checksumming of the OS code segment. | New |
| 5956 | CS3004 | Upon receipt of a Recompute OS code segment CRC command, CS shall:   1. Recompute the baseline CRC for the OS code segment 2. Set the Recompute In Progress Flag to TRUE | May want to recompute OS code segment in the event of a modification to the OS code segment. | New |
| 5958 | CS3004.1 | Once the baseline CRC is computed, CS shall:   1. Generate an event message containing the baseline CRC 2. Set the Recompute In Progress Flag to FALSE | Gives the ground indication not only that the CRC baseline was calculated but what the value is. | New |
| 5960 | CS3004.2 | If CS is already processing a Recompute CRC command or a One Shot CRC command, CS shall reject the command. | Both the One Shot and Recompute are done in a background task so only one may be performed at a time. | New |
| 5962 | CS3005 | Upon receipt of a Report OS code segment CRC command, CS shall send an event message containing the baseline OS code segment CRC. | Provides the ability to view the OS code segment baseline CRC. | New |
| 5964 | CS3006 | Checksum shall calculate CRC for the cFE code segment and compare them against the corresponding baseline cFE code segment CRC if:   1. Checksumming (as a whole) is Enabled 2. cFE segment checksumming is Enabled | Need to verify the cFE code segment. | New |
| 5974 | CS3006.1 | If the cFE code segment CRC is not equal to the baseline cFE code segment CRC, CS shall increment the cFE Code Segment CRC Miscompare Counter and send an event message. |  | New |
| 5976 | CS3007 | Upon receipt of an Enable cFE code segment command, CS shall enable checksumming of the cFE Code segment. | Enable checksumming of the cFE code segment. | New |
| 5978 | CS3008 | Upon receipt of a Disable cFE code segment command, CS shall Disable checksumming of the cFE Code segment. | Disable checksumming of the cFE code segment. | New |
| 5980 | CS3009 | Upon receipt of a Recompute cFE Code Segment CRC command, CS shall:   1. Recompute the baseline CRC for the cFE Code Segment 2. Set the Recompute In Progress Flag to TRUE | May want to recompute cFE code segment in the event of a modification to the cFE code segment. | New |
| 5982 | CS3009.1 | Once the baseline CRC is computed, CS shall:   1. Generate an event message containing the baseline CRC 2. Set the Recompute In Progress Flag to FALSE | Gives the ground indication not only that the CRC baseline was calculated but what the value is. | New |
| 5984 | CS3009.2 | If CS is already processing a Recompute CRC command or a One Shot CRC command, CS shall reject the command. | Both the One Shot and Recompute are done in a background task so only one may be performed at a time. | New |
| 5986 | CS3010 | Upon receipt of a Report cFE code segment CRC command, CS shall send an event message containing the baseline cFE code segment CRC. | Provides the ability to view the cFE code segment baseline CRC | New |
|  |  | 5.4        Volatile Memory – Application Code Segments  The Checksum Application provides the ability to checksum the application code segments. An Application Code Segment Table is used to define the applications to checksum. |  |  |
| 6010 | CS4000 | Checksum shall calculate CRCs for each Table-Defined Application’s code segment and compare them against the corresponding Application’s baseline code segment CRC if:   1. Checksumming (as a whole) is Enabled 2. App code segment checksumming is Enabled 3. Checksumming of the individual Application Code Segment is Enabled | Need to verify each Application’s code segment. Note that CS depends on ES to provide the information as to which applications are running. | SDO (loosely) |
| 6018 | CS4000.1 | If the Application’s code segment CRC is not equal to the corresponding Application’s baseline code segment CRC, CS shall increment the Application Code Segment CRC Miscompare Counter and send an event message. | In practice, when a new application is being loaded, checksumming of Application code segements should be disabled prior to the load and then enabled after the load. | SDO (loosely) |
| 6020 | CS4000.2 | If the table-defined Application code segment is invalid, CS shall send an event message and skip that Application code segment. | This may be a result of an invalid Application code segment table or a deleted application. | SDO (loosely) |
| 6022 | CS4001 | Upon receipt of an Enable Application checksumming command, CS shall enable checksumming of all Application Code segments. | Enable checksumming of all of the Application code segments defined in the table. | SDO (loosely) |
| 6024 | CS4002 | Upon receipt of a Disable Application checksumming command, CS shall Disable checksumming of all Application Code segments. | Disable checksumming of all of the Application code segments defined in the table. | SDO (loosely) |
| 6026 | CS4003 | Upon receipt of an Enable Application code segment command, CS shall enable checksumming of the command-specified Application. | Enable checksumming of a particular Application code segment. | SDO (loosely) |
| 6028 | CS4004 | Upon receipt of a Disable Application code segment command, CS shall Disable checksumming of the command-specified Application. | Disable checksumming of a particular Application Code segment. This may be particularly useful when reloading an existing application. | SDO (loosely) |
| 6030 | CS4005 | Upon receipt of a Recompute Application Code Segment CRC command, CS shall:   1. Recompute the baseline CRC for the Application 2. Set the Recompute In Progress Flag to TRUE | Would be used after an Application code segment is updated in order to regenerate the baseline. | SDO (loosely) |
| 6032 | CS4005.1 | Once the baseline CRC is computed, CS shall:   1. Generate an event message containing the baseline CRC 2. Set the Recompute In Progress Flag to FALSE | Gives the ground indication not only that the CRC baseline was calculated but what the value is. | New |
| 6034 | CS4005.2 | If CS is already processing a Recompute CRC command or a One Shot CRC command, CS shall reject the command. | Both the One Shot and Recompute are done in a background task so only one may be performed at a time. |  |
| 6040 | CS4006 | Upon receipt of a Report Application code segment CRC command, CS shall send an event message containing the baseline Application code segment CRC. | Provides the ground with the baseline being used. | SDO (loosely) |
| 6042 | CS4007 | If the command-specified Application is invalid (for any Application Code Segment command where the Application is a command argument, CS shall reject the command and send an event message. | Need to handle the case where an invalid Application is specified for any of the Application code segment commands. | SDO (loosely) |
| 6044 | CS4008 | CS shall provide the ability to dump the baseline CRCs and status for the Application code segment memory segments via a dump-only table. | Need the ability to get all of the application code segment checksums. Easiest to use the cFE Table services dump-only table feature. | SDO |
|  |  | 5.5        Volatile Memory – Tables  The Checksum Application provides the ability to checksum the Application’s Tables. A Checksum Table is used to define the tables that are required to be to checksummed. |  |  |
| 6060 | CS5000 | Checksum shall calculate CRCs for each Table-Defined Table and compare them against the corresponding Table’s baseline CRC if:   1. Checksumming (as a whole) is Enabled 2. Table checksumming is Enabled 3. Checksumming of the Individual Table is Enabled | Need to verify each Table CRC. Note that CS depends on ES to provide the information as to which Tables to checksum. | SDO (loosely) |
| 6068 | CS5000.1 | If the Table’s CRC is not equal to the corresponding Table’s baseline CRC and the table has not been modified (thru a table load), CS shall increment the Table CRC Miscompare Counter and send an event message. | cFE Tables services provides an indication that a table was modified, a checksum miscompare when a table was not modified via a table load, then there was a checksum failure. | SDO (loosely) |
| 6070 | CS5000.2 | If the Table’s CRC is not equal to the corresponding Table’s baseline CRC and the table has been modified (thru a table load), CS shall recompute the table baseline CRC. | If a table is changed via a table load, CS needs to recompute the baseline CRC. | SDO |
| 6072 | CS5000.3 | If the table-defined Table is invalid, CS shall send an event message and skip that Table. | This may be a result of an invalid Table table or a deleted table. | SDO (loosely) |
| 6074 | CS5001 | Upon receipt of an Enable Table Checksumming command, CS shall enable checksumming of all Tables. | Enable checksumming of all of the Tables defined in the table. | SDO (loosely) |
| 6076 | CS5002 | Upon receipt of a Disable Table Checksumming command, CS shall Disable checksumming of all Tables. | Disable checksumming of all of the Tables defined in the table. | SDO (loosely) |
| 6078 | CS5003 | Upon receipt of an Enable Table Name command, CS shall enable checksumming of the command-specified Table. | Provides control over enable/disable status of each table. |  |
| 6080 | CS5004 | Upon receipt of a Disable Table Name command, CS shall Disable checksumming of the command-specified Table. | Provides control over enable/disable status of each table. |  |
| 6082 | CS5005 | Upon receipt of a Recompute Table CRC Command, CS shall:   1. Recompute the baseline CRC for the command-specified table 2. Set the Recompute In Progress Flag to TRUE | If a table is modified, CS needs to recompute a baseline CRC. |  |
| 6084 | CS5005.1 | Once the baseline CRC is computed, CS shall:   1. Generate an event message containing the baseline CRC 2. Set the Recompute In Progress Flag to FALSE | Gives the ground indication not only that the CRC baseline was calculated but what the value is. |  |
| 6086 | CS5005.2 | If CS is already processing a Recompute CRC command or a One Shot CRC command, CS shall reject the command. | Both the One Shot and Recompute are done in a background task so only one may be performed at a time. |  |
| 6088 | CS5006 | Upon receipt of a Report Table CRC command, CS shall send an event message containing the baseline Table CRC for the command-specified table. | Provides the ground with the baseline being used. | SDO |
| 6090 | CS5007 | If the command-specified Table in invalid (for any CS Table command where a table name is a command argument), CS shall reject the command and send an event message. | Need to handle the case where an invalid Table is specified. |  |
| 6092 | CS5008 | CS shall provide the ability to dump the baseline CRCs and status for the tables via a dump-only table. | Need the ability to get all of the table checksums. Easiest to use the cFE Table services dump-only table feature. | New |
|  |  | 5.6        Volatile Memory – User-Defined Memory (Memory)  The Checksum Application provides the ability to checksum the user-defined memory. A User-defined Memory Table is used to define the memory to checksum. |  |  |
| 6116 | CS6000 | Checksum shall calculate CRCs for each Table-Defined User-Defined Memory and compare them against the corresponding baseline CRC if:   1. Checksumming (as a whole) is Enabled 2. User-Defined Memory checksumming is Enabled 3. Checksumming of the Individual Memory segments is Enabled | Need to verify each Table CRC. Note that CS depends on ES to provide the information as to which Tables to checksum. | SDO (loosely) |
| 6124 | CS6000.1 | If the User-Defined Memory’s CRC is not equal to the corresponding baseline CRC, CS shall increment the User-Defined Memory CRC Miscompare Counter and send an event message. |  | SDO (loosely) |
| 6126 | CS6000.2 | If the table-defined Memory is invalid, CS shall send an event message. | This may be a result of an invalid User-Defined Memory area. | SDO (loosely) |
| 6128 | CS6001 | Upon receipt of an Enable User-Defined Memory Checksumming command, CS shall enable checksumming of all User-Defined Memory. | Enable checksumming of all of the User-Defined Memory defined in the table. | SDO (loosely) |
| 6130 | CS6002 | Upon receipt of a Disable User-Defined Memory Checksumming command, CS shall Disable checksumming of all User-Defined Memory. | Disable checksumming of all of the User-Defined Memory defined in the table. | SDO (loosely) |
| 6132 | CS6003 | Upon receipt of an Enable User-Defined Memory Item command, CS shall enable checksumming of the command-specified Memory. |  | New |
| 6134 | CS6004 | Upon receipt of a Disable User-Defined Memory Item command, CS shall Disable checksumming of the command-specified Memory. |  | New |
| 6136 | CS6005 | Upon receipt of a Recompute User-Defined Memory CRC command, CS shall:   1. Recompute the baseline CRC for the command-specified User-Defined Memory 2. Set the Recompute In Progress Flag to TRUE |  | New |
| 6138 | CS6005.1 | Once the baseline CRC is computed, CS shall:   1. Generate an event message containing the baseline CRC 2. Set the Recompute In Progress Flag to FALSE | Gives the ground indication not only that the CRC baseline was calculated but what the value is. | New |
| 6140 | CS6005.2 | If CS is already processing a Recompute CRC command or a One Shot CRC command, CS shall reject the command. | Both the One Shot and Recompute are done in a background task so only one may be performed at a time. |  |
| 6142 | CS6006 | Upon receipt of a Report User-Defined Memory CRC command, CS shall send an event message containing the baseline CRC for the command-specified User-Defined Memory. | Provides the ground with the baseline being used. | SDO |
| 6144 | CS6007 | If the command-specified User-Defined Memory is invalid (for any of the User-Defined memory commands where the memory ID is a command argument), CS shall reject the command and send an event message. | Need to handle the case where an invalid User-Defined Memory is specified. |  |
| 6146 | CS6008 | CS shall provide the ability to dump the baseline CRCs and status for all the User-Defined Memory via a dump-only table. | Need the ability to get all of the User-Defined Memory checksums. Easiest to use the cFE User-Defined Memory services dump-only User-Defined Memory feature. | New |
| 146062 | CS6009 | Upon receipt of a Get User-Defined Memory Entry ID command, CS shall send an informational event message containing the User-Defined Memory Table Entry ID for the command-specified Memory Address. | Provides the ground with table information without having to perform a full table dump. | LRO |
| 146064 | CS6009.1 | If the command-specified Memory Address cannot be found within the User-Defined Memory Table, CS shall send an informational event message. | Need to alert ground of failure to find memory address. | LRO |
|  |  | 5.7        Checksumming Rates  In order to ensure that the Checksum Application does not hog the CPU, limits to the amount of data that gets processed per execution cycle need to be defined. |  |  |
| 6170 | CS7000 | The CS software shall limit the amount of bytes processed during each of its execution cycles to a maximum of <PLATFORM\_DEFINED> bytes. | Want to make sure that CS does not hog the CPU. | SDO, LRO |
|  |  | 5.8        General Checksum Commands  The following are the commands that are supported in order to control the checksum application. These commands do not depend on the regions of memory (e.g. non-volatile, application code segment etc.). |  |  |
| 6186 | CS8000 | Upon receipt of an Enable Checksum command, CS shall start calculating CRCs and compare them against the baseline CRCs. | Provides global control over CS | LRO, SDO |
| 6188 | CS8001 | Upon receipt of a Disable Checksum command, CS shall stop calculating CRCs and comparing them against the baseline CRCs. | Provides global control over CS. Note that this supersedes the enable/disable status of each region’s enable/disable status AND the enable status of each element within a region (e.g. Even if App code segment X is Enabled, CS will not perform checksumming operation. If Table checksumming is Enabled, CS will not perform checksumming. | LRO, SDO |
| 6190 | CS8002 | Upon receipt of a One Shot command, CS shall:   1. Calculate the CRC starting at the command-specified address for the command-specified bytes at the command-specified rate (Max Bytes Per Cycle) 2. Set the One Shot In Progress Flag to TRUE | Provides a generic capability to compute a checksum for any memory. | LRO, SDO |
| 6192 | CS8002.1 | Once the CRC is computed CS shall:   1. Issue an event message containing the CRC 2. Set the One Shot In Progress Flag to FALSE |  | LRO, SDO |
| 6194 | CS8002.2 | If CS is already processing a One Shot CRC command or a Recompute CRC command, CS shall reject the command. | Both the One Shot and Recompute are done in a background task so only one may be performed at a time. |  |
| 146071 | CS8002.3 | If the command-specified rate is zero, CS shall calculate the CRC at the <PLATFORM\_DEFINED> rate (Max Bytes Per Cycle). | Allow use of the default checksum rate. | New |
| 6196 | CS8003 | Upon receipt of a Cancel One Shot command, CS shall stop the current One Shot calculation. | In the event that a memory region is too large, requiring too much time, cancelling the calculation may be required. | LRO, SDO |
|  |  | 5.9        Status Reporting |  |  |
| 6325 | CS9000 | CS shall generate a housekeeping message containing the following:   1. Valid Ground Command Counter 2. Ground Command Rejected Counter 3. Overall CRC enable/disable status 4. Total Non-volatile Baseline CRC 5. OS code segment Baseline CRC 6. cFE code segment Baseline CRC 7. Non-volatile CRC Miscompare Counter 8. OS Code Segment CRC Miscompare Counter 9. cFE Code Segment CRC Miscompare Counter 10. Application CRC Miscompare Counter 11. Table CRC Miscompare Counter 12. User-Defined Memory CRC Miscompare Counter 13. Last One Shot Address 14. Last One Shot Size 15. Last One Shot Checksum 16. Checksum Pass Counter (number of passes thru all of the checksum areas) 17. Current Checksum Region (Non-volatile, OS code segment, cFE code segment etc.) 18. Non-volatile CRC enable/disable status 19. OS Code Segment CRC enable/disable status 20. cFE Code Segment CRC enable/disable status 21. Application CRC enable/disable status 22. Table CRC enable/disable status 23. User-Defined Memory CRC enable/disable status 24. Last One Shot Rate 25. Recompute In Progress Flag 26. One Shot In Progress Flag |  |  |
|  |  | 5.10        Initialization Requirements  The following are the requirements associated with Checksum on an Application reset, cFE Processor Reset or a cFE Power-on Reset. |  |  |
| 6252 | CS9001 | Upon any Initialization of the CS Application (cFE Power On, cFE Processor Reset or CS Application Reset), CS shall initialize the following data to Zero:   1. Valid Ground Command Counter 2. Ground Command Rejected Counter 3. Non-volatile CRC Miscompare Counter 4. OS Code Segment CRC Miscompare Counter 5. cFE Code Segment CRC Miscompare Counter 6. Application CRC Miscompare Counter 7. Table CRC Miscompare Counter 8. User-Defined Memory CRC Miscompare Counter 9. Recompute In Progress Flag 10. One Shot In Progress Flag | No information is preserved across a cFE Processor reset or CS Application Reset. | Derived |
| 6270 | CS9002 | Upon a cFE Power On Reset, if the segment's <PLATFORM\_DEFINED> Power-On Initialization state is set to Enabled, CS shall compute baseline CRCs for the following regions:   1. OS code segment 2. cFE code segment | Need to compute a baseline which is used to compare against when background checking the checksums. | LRO, SDO |
| 6276 | CS9003 | Upon a cFE Power On Reset, if the Non-Volatile <PLATFORM\_DEFINED> Power-On Initialization state is set to Enabled, CS shall compute baseline CRCs for Non-volatile segments based on the corresponding table definition for up to <PLATFORM\_DEFINED> segments. | Need to compute a baseline which is used to compare against when background checking the checksums. | LRO |
| 6278 | CS9003.1 | If the address range for any of the Non-volatile segments is Invalid, CS shall send an event message and disable Non-volatile Checksumming. | Table validation includes verifying that the memory ranges are within limits. | New |
| 6280 | CS9003.2 | CS shall send an event message and disable Non-volatile Checksumming, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states. | New |
| 6288 | CS9004 | Upon a cFE Power On Reset, if the Non-Volatile <PLATFORM\_DEFINED> Power-On Initialization state is set to Enabled, CS shall compute the baseline CRC for the total of all of non-volatile segments. | Need to have a checksum for the entire image. Note that the CRCs for each of the non-volatile segments specified in the table are added together to arrive at this number. | SDO |
| 6290 | CS9005 | Upon a cFE Power On Reset, if the Application <PLATFORM\_DEFINED> Power-On Initialization state is set to Enabled, CS shall compute baseline CRCs for the Application code segments region based on the corresponding table definition for up to a <PLATFORM\_DEFINED> Applications. | Need to compute baselines for the Applications specified in the table. The platform-defined value could be equal to the max number of apps defined by cFE ES but could be less. | SDO (loosely) |
| 6292 | CS9005.1 | CS shall send an event message and disable Application code segment Checksumming, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states. | New |
| 6300 | CS9006 | Upon a cFE Power On Reset, if the Tables <PLATFORM\_DEFINED> Power-On Initialization state is set to Enabled, CS shall compute baseline CRCs for the tables specified in the corresponding table definition for up to <PLATFORM\_DEFINED> tables. | A table is used to define the tables that should be checksummed. Baseline checksums are computed for the tables specified in the table. The platform-defined value could be equal to the max tables defined by cFE TBL but could be less. | SDO (loosely) |
| 6302 | CS9006.1 | CS shall send an event message and disable Table Checksumming, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states. | New |
| 6310 | CS9007 | 261508, CS shall compute baseline CRCs for the User-Defined memory region based on the corresponding table definition for up to <PLATFORM\_DEFINED> memory segments. | Need to calculate baseline for all User-defined memory segments specified in a table | SDO (loosely) |
| 6312 | CS9007.1 | If the address range for any of the User-Defined Memory is Invalid, CS shall send an event message and disable User-Defined Memory Checksumming. | Table validation includes verifying that the memory ranges are within limits | New |
| 6314 | CS9007.2 | CS shall send an event message and disable Checksumming of the User-Defined Memory, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states. | New |
| 146074 | CS9008 | Upon a cFE Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to True, CS shall preserve the following:   1. OS Code Segment Checksumming State 2. cFE Code Segment Checksumming State 3. Non-volatile Checksumming State 4. Application Code Segment Checksumming State 5. Table Checksumming State 6. User-Defined Memory Checksumming State | Allows ground to preserve configured checksum states over a reset | New |
| 146102 | CS9009 | Upon a cFE Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to False, CS shall perform initialization in accordance with a Power On reset. | Allows ground to handle all resets in the same manner | New |
| 146076 | CS9010 | Upon a cFE Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to True and the segment's state is set to Enabled, CS shall compute baseline CRCs for the following regions:   1. OS code segment 2. cFE code segment | Allows ground to preserve configured checksum state over a reset | New |
| 146080 | CS9011 | Upon a Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to True and the Non-volatile Checksumming State is Enabled, CS shall compute baseline CRCs for Non-volatile segments based on the corresponding table definition for up to <PLATFORM\_DEFINED> segments. | Allows ground to preserve configured checksum state over a reset | New |
| 146082 | CS9011.1 | If the address range for any of the Non-volatile segments is Invalid, CS shall send an event message and disable Nonvolatile. | Table validation includes verifying that the memory ranges are within limits | New |
| 146084 | CS9011.2 | CS shall send an event message and disable Non-volatile Checksumming, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states | New |
| 146086 | CS9012 | Upon a cFE Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to True and the Non-volatile Checksumming State is Enabled, CS shall compute the baseline CRC for the total of all of non-volatile segments. | Allows ground to preserve configured checksum state over a reset | New |
| 146088 | CS9013 | Upon a cFE Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to True and the Application Code Segment Checksumming State is Enabled, CS shall compute baseline CRCs for the Application code segments region based on the corresponding table definition for up to a <PLATFORM\_DEFINED> Applications. | Allows ground to preserve configured checksum state over a reset | New |
| 146090 | CS9013.1 | CS shall send an event message and disable Application code segment Checksumming, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states. | New |
| 146092 | CS9014 | Upon a a cFE Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to True and the Table Checksumming State is Enabled, CS shall compute baseline CRCs for the tables specified in the corresponding table definition for up to <PLATFORM\_DEFINED> tables. | Allows ground to preserve configured checksum state over a reset | New |
| 146094 | CS9014.1 | CS shall send an event message and disable Table Checksumming, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states. | New |
| 146096 | CS9015 | Upon a cFE Processor Reset or CS Application Reset, if the <PLATFORM\_DEFINED> PRESERVE\_STATES\_ON\_PROCESSOR\_RESET Flag is set to True and the User-Defined Memory Checksumming State is Enabled, CS shall compute baseline CRCs for the User-Defined memory region based on the corresponding table definition for up to <PLATFORM\_DEFINED> memory segments. | Allows ground to preserve configured checksum state over a reset | New |
| 146098 | CS9015.1 | If the address range for any of the User-Defined Memory is Invalid, CS shall send an event message and disable User-Defined Memory Checksumming. | Table validation includes verifying that the memory ranges are within limits | New |
| 146100 | CS9015.2 | CS shall send an event message and disable Checksumming of the User-Defined Memory, if the state is not one of the following:   1. enabled 2. disabled 3. empty | Table validation includes verifying that the table contains valid initial states. | New |

<MKSID-5353>

<MKSID-5361>

 <MKSID-5367>

<MKSID-5369>

<MKSID-5381>

<MKSID-5383>

<MKSID-5385>

<MKSID-5387>

<MKSID-5391>

<MKSID-5393>

<MKSID-5395>

<MKSID-5397>

<MKSID-5399>

<MKSID-5401>

<MKSID-5403>

<MKSID-5405>

<MKSID-5407>

<MKSID-5409>

<MKSID-5411>

<MKSID-5415>

<MKSID-5417>

<MKSID-5427>

<MKSID-5431>

<MKSID-5433>