CFS CS Requirements

Code 582
Flight Software Systems Branch

Mar 29, 2010

ID	Summary
2626	CFS CS Requirements Document

SR_Contains:

ID	ReqID	Text	Rationale	Heritage_Reference
2628		# CFS Checksum (CS) Requirements ## 1.0 Introduction ### 1.1 Document Purpose The Core Flight Software System (CFS) Checksum (CS) Application will be developed by the Flight Software Branch (FSB) of the Software Engineering Division (SED). 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 and several appendices.

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.

Appendix A contains a list of abbreviations and acronyms used in this document.

1.4 Relevant Documents

1.4.1 Parent Documents

CFS Checksum Application Heritage Analysis 582-2007-028

1.4.2 Reference Documents

1. Operating System Abstraction Layer (OSAL) Library
2. cFE Application Developer's Guide 582-2007-001
3. 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

· 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 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

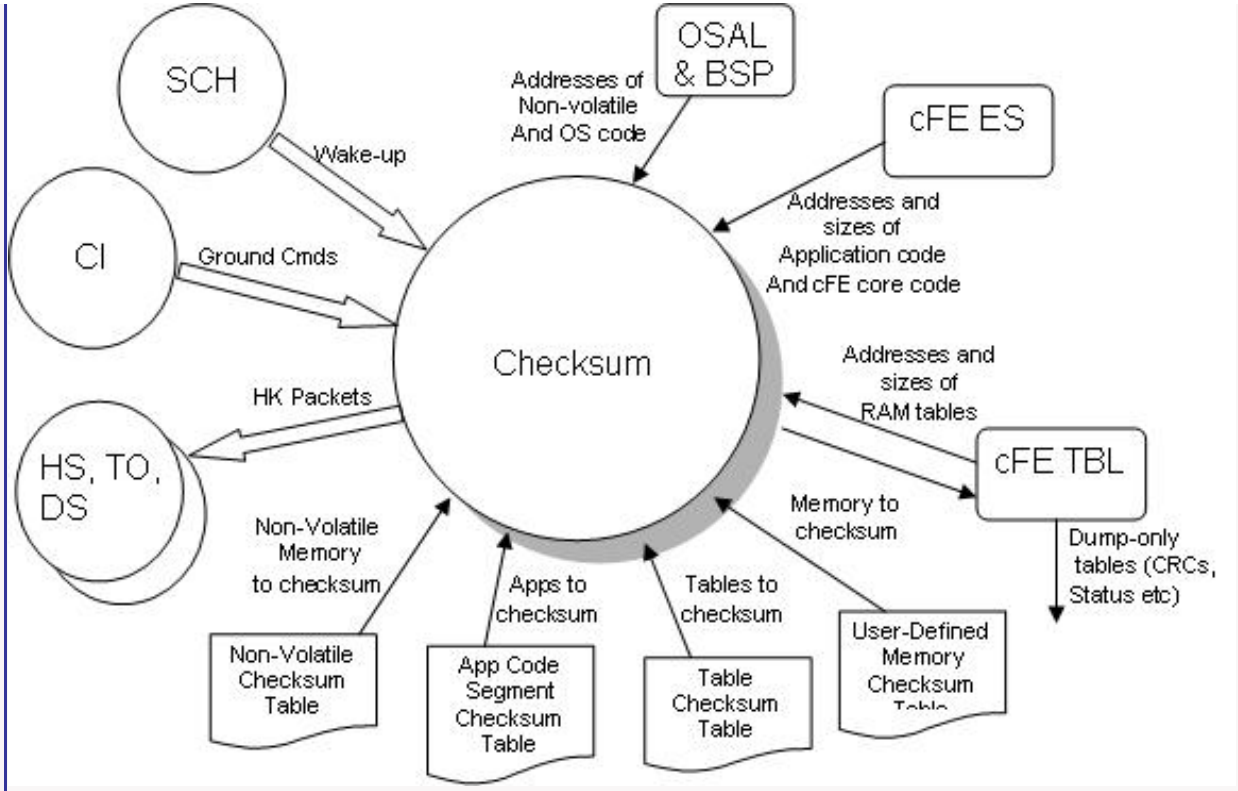

Figure 2.0 – CFS CS Context

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

2630	CFS-450	The CFS shall verify the integrity of static memory	Want to make sure that static remains unchanged.	SDO
2632		# 5.0 Detailed Requirements ## 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.		
2634	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
2636	CS1001	Upon receipt of a Reset command, CS shall reset the following housekeeping variables to a value of zero: a) Valid Ground Command Counter b) Ground Command Rejected Counter c) Non-volatile CRC Miscompare Counter d) OS Code Segment CRC Miscompare Counter e) cFE Code Segment CRC Miscompare Counter f) Application CRC Miscompare Counter g) Table CRC Miscompare Counter h) User-defined Memory CRC Miscompare Counter i) 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
2638	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
2640	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

2642	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
2644	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
2806		**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.		
2646	CS2001	The Checksum App shall calculate CRCs for each Non-volatile segment and compare them against the corresponding baseline Non-volatile segment CRCs if: a) Checksumming (as a whole) is Enabled b) Non-volatile segment checksumming is Enabled c) 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
2648	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

2650	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
2652	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
2654	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
2656	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
2658	CS2006	Upon receipt of a Recompute Non-volatile Checksum Segment command, CS shall recompute the baseline checksum for the command-specified non-volatile segment	Would be used after non-vol memory is updated in order to regenerate the baseline	LRO
4437	CS2006.1	If CS is already processing a Recompute CRC command, CS shall reject the command	Recompute is done in a background task so can only do one recompute at a time	
2660	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
2662	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

2664	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
2666	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
2808		**5.3 Volatile Memory – OS and cFE Code Segments** The Checksum Application provides the ability to checksum the OS and cFE Code Segments		
2668	CS3000	Checksum shall calculate CRC for the OS code segment and compare them against the corresponding baseline OS code segment CRC if: a) Checksumming (as a whole) is Enabled b) OS segment checksumming is Enabled	Need to verify the OS code segment	New
2670	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
2672	CS3002	Upon receipt of a Enable OS code segment command, CS shall enable checksumming of the OS Code segment	Enable checksumming of the OS code segment	New
2674	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

2676	CS3004	Upon receipt of a Recompute OS code segment CRC command, CS shall recompute the baseline CRC for the OS code segment	May want to recompute OS code segment in the event of a modification to the OS code segment	New
2678	CS3004.1	Once the baseline CRC is computed, CS shall generate an event message containing the baseline CRC	Gives the ground indication not only that the CRC baseline was calculated but what the value is.	New
4451	CS3004.2	If CS is already processing a Recompute CRC command, CS shall reject the command	Recompute done in the background so can only do one at a time	New
2680	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
2682	CS3006	Checksum shall calculate CRC for the cFE code segment and compare them against the corresponding baseline cFE code segment CRC if: a) Checksumming (as a whole) is Enabled b) cFE segment checksumming is Enabled	Need to verify the cFE code segment	New
2684	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
2686	CS3007	Upon receipt of a Enable cFE code segment command, CS shall enable checksumming of the cFE Code segment	Enable checksumming of the cFE code segment	New
2688	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
2690	CS3009	Upon receipt of a Recompute cFE Code Segment CRC command, CS shall recompute the baseline CRC for the cFE Code Segment	May want to recompute cFE code segment in the event of a modification to the cFE code segment	New

4429	CS3009.1	Once the baseline CRC is computed, CS shall generate an event message containing the baseline CRC	Gives the ground indication not only that the CRC baseline was calculated but what the value is	New
4439	CS3009.2	If CS is already processing a Recompute CRC command, CS shall reject the command	Only able to process one recompute one at a time	
2692	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
2810		**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.		
2694	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: a) Checksumming (as a whole) is Enabled b) App code segment checksumming is Enabled c) 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)

2696	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)
2698	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)
2700	CS4001	Upon receipt of a 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)
2702	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)
2704	CS4003	Upon receipt of a Enable Application code segment command, CS shall enable checksumming of the command-specified Application.	Enable checksumming of a particular Application code segment	SDO (loosely)
2706	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)

2708	CS4005	Upon receipt of a Recompute Application Code Segment CRC command, CS shall recompute the baseline CRC for the Application	Would be used after an Application code segment is updated in order to regenerate the baseline	SDO (loosely)
2710	CS4005.1	Once the baseline CRC is computed, CS shall generate an event message containing the baseline CRC	Gives the ground indication not only that the CRC baseline was calculated but what the value is.	New
4441	CS4005.2	If CS is already processing a Recompute CRC command, CS shall reject the command	Can only support one recompute CRC at a time	
2712	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)
2714	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)
2716	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
2812		**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.		

2718	CS5000	Checksum shall calculate CRCs for each Table-Defined Table and compare them against the corresponding Table's baseline CRC if: a) Checksumming (as a whole) is Enabled b) Table checksumming is Enabled c) 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)
2720	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)
2722	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
2724	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)
2726	CS5001	Upon receipt of a Enable Table Checksumming command, CS shall enable checksumming of all Tables.	Enable checksumming of all of the Tables defined in the table	SDO (loosely)
2728	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)
2730	CS5003	Upon receipt of a Enable Table Name command, CS shall enable checksumming of the command-specified Table.	Provides control over enable/ disable status of each table	
2732	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	

4431	CS5005	Upon receipt of a Recompute Table CRC Command, CS shall recompute the baseline CRC for the command-specified table	If a table is modified, CS needs to recompute a baseline CRC	
2734	CS5005.1	Once the baseline CRC is computed, CS shall generate an event message containing the baseline CRC	Gives the ground indication not only that the CRC baseline was calculated but what the value is.	
4443	CS5005.2	If CS is already processing a Recompute CRC command, CS shall reject the command	Can only support one recompute at a time	
2736	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
2738	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	
2740	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
2814		**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.		

2742	CS6000	Checksum shall calculate CRCs for each Table-Defined User-Defined Memory and compare them against the corresponding baseline CRC if a) Checksumming (as a whole) is Enabled b) User-Defined Memory checksumming is Enabled c) 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)
2744	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)
2746	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)
2748	CS6001	Upon receipt of a 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)
2750	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 Memorys defined in the table	SDO (loosely)
2752	CS6003	Upon receipt of a Enable User-Defined Memory Item command, CS shall enable checksumming of the command-specified Memory.		New
2754	CS6004	Upon receipt of a Disable User-Defined Memory Item command, CS shall Disable checksumming of the command-specified Memory.		New
2756	CS6005	Upon receipt of a Recompute User-Defined Memory CRC command, CS shall recompute the baseline CRC for the command-specified User-Defined Memory.		New
2758	CS6005.1	Once the baseline CRC is computed, CS shall generate an event message containing the baseline CRC	Gives the ground indication not only that the CRC baseline was calculated but what the value is.	New
4445	CS6005.2	If CS is already processing a Recompute CRC command, CS shall reject the command	Can only support one recompute at a time	
2760	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

2762	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	
2764	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
2816		## 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.		
2766	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
2818		## 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 (eg. non-volatile, application code segment etc).		
2768	CS8000	Upon receipt of a Enable Checksum command, CS shall start calculating CRCs and compare them against the baseline CRCs.	Provides global control over CS	LRO, SDO

2770	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 superceeds the enable/disable status of each regions's enable/ disable status AND the enable status of each element within a region (eg. 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
2772	CS8002	Upon receipt of a One Shot command, CS shall calculate the CRC starting at the command-specified address for the command-specified bytes.	Provides a generic capability to compute a checksum for any memory	LRO, SDO
2774	CS8002.1	CS shall issue an event message containing the CRC		LRO, SDO
4447	CS8002.2	If CS is already processing a One Shot CRC command, CS shall reject the command	Can only process one One Shot command at a time	
2776	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
2820		**5.9 Status Reporting**		

2778	CS9000	CS shall generate a housekeeping message containing the following: a) Valid Ground Command Counter b) Ground Command Rejected Counter c) Overall CRC enable/disable status d) Total Non-volatile Baseline CRC e) OS code segment Baseline CRC f) cFE code segment Baseline CRC g) Non-volatile CRC Miscompare Counter h) OS Code Segment CRC Miscompare Counter i) cFE Code Segment CRC Miscompare Counter j) Application CRC Miscompare Counter k) Table CRC Miscompare Counter l) User-Defined Memory CRC Miscompare Counter m) Last One Shot Address n) Last One Shot Size o) Last One Shot Checksum p) Checksum Pass Counter (number of passes thru all of the checksum areas) q) Current Checksum Region (Non-volatile, OS code segment, cFE code segment etc) r) Non-volatile CRC enable/disable status s) OS Code Segment CRC enable/disable status t) cFE Code Segment CRC enable/disable status u) Application CRC enable/disable status v) Table CRC enable/disable status w) User-Defined Memory CRC enable/disable status	Housekeeping telemetry to indicate basic CS status.	LRO, SDO
2822		**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		
2780	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: a) Valid Ground Command Counter b) Ground Command Rejected Counter c) Non-volatile CRC Miscompare Counter d) OS Code Segment CRC Miscompare Counter e) cFE Code Segment CRC Miscompare Counter f) Application CRC Miscompare Counter g) Table CRC Miscompare Counter h) User-Defined Memory CRC Miscompare Counter	No information is preserved across a cFE Processor reset or CS Application Reset.	Derived
2782	CS9002	Upon any Initialization, CS shall compute baseline CRCs for the following regions: a) OS code segment b) cFE code segment	Need to compute a baseline which is used to compare against when background checking the checksums.	LRO, SDO

2784	CS9003	Upon any Initialization, 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
2786	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
2788	CS9003.2	CS shall send an event message and disable Non-volatile Checksumming, if the state is not one of the following: a) enabled b) disabled c) empty	Table validation includes verifying that the table contains valid intial states.	New
2790	CS9004	Upon any Initialization, CS shall compute 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-vol segments sepcified in the table are added together to arrive at this number.	SDO
2792	CS9005	Upon any Initialization, 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)
2794	CS9005.1	CS shall send an event message and disable Application code segment Checksumming, if the state is not one of the following: a) enabled b) disabled c) empty	Table validation includes verifying that the table contains valid intial states.	New

2796	CS9006	Upon any Initialization, 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)
2798	CS9006.1	CS shall send an event message and disable Table Checksumming, if the state is not one of the following: a) enabled b) disabled c) empty	Table validation includes verifying that the table contains valid intial states.	New
2800	CS9007	Upon any Initialization, 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)
2802	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
2804	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: a) enabled b) disabled c) empty	Table validation includes verifying that the table contains valid intial states.	New