Mission Name

Core Flight System (CFS)
Stored Command (SC)
Version 2.4.0.0
APPLICATION USER'S GUIDE

Flight Software Systems Branch – Code 582
Release Version 1.1 – 12/18/14
582-2012-003

Goddard Space Flight Center

Greenbelt, Maryland

National Aeronautics and
Space Administration

FORWORD

This CFS Stored Command (SC) Application User's Guide provides guidance for the Flight Operations Team (FOT) for the Core Flight System (CFS) SC Application.

This is one of a set of enhanced User's Guides planned for the CFS Product Documentation Suite. While the main audience is the FOT, the Guides also help serve the needs of flight software developers; Flight Software Sustaining Engineering (FSSE), Integration and Test, and others who support missions which use CFS.

This generic CFS SC Guide is set up so that missions can convert it into a mission-specific guide.

AUTHOR

9/5/2012

Gary M Smith

Technical Writer

Date

APPROVALS

Susanne Strege

9-5-2012

Susanne Strege / 582

Core Flight Executive (cFE) Core Flight System (CFS) / PDL

Date

Charle Wildermann

9-5-2012

Charles Wildermann / 582

Flight Software Systems Branch/Head

Date

UPDATE HISTORY

Version	Date	Description	Affected Pages
1.0	09/05/12	Release Version	All
1.1	12/18/14	Added notes about required pad byte needed for odd length commands – DCR 21200. Updated event messages to be consistent with use of ATS and RTS vs. spelling out the acronyms within the event – DCR 22621 Added new "ATP Precaution" section 3.2.6 – DCR 21590	3-6, 3-10, 3-11, A-11, A-28, A-29, A-30, A-31

CONTENTS

TABLE OF FIGURES

TABLES

This page intentionally blank

Chapter 1. **Introduction to the CFS SC User's Guide**

1.1 *Purpose and Scope of this Guide*

The primary purpose of this Application User's Guide is to help the Flight Operations Team (FOT) understand the Stored Command (SC) application.

Many other purposes will be found for this Guide, including helping mission flight software personnel populate the ground system Record Definition Language (RDL) files in the ground system used later by the FOT, e.g., Advanced Spacecraft Integration & System Test software (ASIST).

Further purposes of this Guide are to help mission developers, system integration and test team members, and Flight Software Sustaining Engineering (FSSE) to understand the Stored Command (SC) application for their own specific needs, such as using the software to perform certain hardware tests. *As delivered, this is a generic document ready to insert mission defined values to serve the needs of specific missions.*

1.2 *Acknowledgements*

This Application User's Guide relies heavily on the content of earlier heritage SC publications and presentations and interviews with flight software engineers, Appendix A is based on information from CFS SC source code as processed by Doxygen in Rich Text Format (RTF) output, and reformatted for this linear publication.

1.3 *Conventions*

- In this document, flight controller, ground controller, and the Flight Operations Team (FOT) are used interchangeably.
- In this document, the percent sign (%), when followed by a string, may indicate variable text. See Appendix A for references to the text that may be substituted in each case.
- Core Flight Executive is abbreviated cFE (lower case "c" with uppercase "FE").
- *See Appendix C for more.*

1.4 *Related Documents*

Documents used in the preparation of this Guide are listed in the table below.

Table 1 Related Documents

Item No.	Document ID	Document Source
1	N/A	Yanchik, Nicholas. *CFS_SC_Design.* CFS Stored Command (SC) Application Design Review, As Built. Greenbelt, MD: Goddard Space Flight Center, Code 582 (Flight Software Branch), 23 Dec. 2008. PPT.
2	N/A	CFS SC Requirements Document V1_4 120811. Greenbelt: NASA Goddard Space Flight Center, Code 582, Flight Software Systems Branch, 8 Dec. 2011. PDF.
3	N/A	Strege, Susanne L. CFS-SC-UsersGuide-6-7-12. Greenbelt: NASA Goddard Space Flight Center, Code 582, CFS Product Development Team, 7 June 2012. Doxygen Output via RTF.
4	N/A	NASA Goddard Space Flight Center. *Lunar Reconnaissance Orbiter (LRO) Stored Commands Task User's Guide.* Version 1. Greenbelt: Code 582, Flight Software Branch, August 1, 2008. DOC. (Please note that there are significant differences between the version of SC used in LRO and the version documented in this User Guide.)

1.5 *Assumptions*

1.5.1 Personnel

This Application User's Guide assumes the primary reader is either a new or long term member of the Flight Operations Team (FOT) at NASA or the equivalent elsewhere.

This Guide also assumes that many other audiences may find it useful, as described in *Purpose and Scope of this Guide* above.

1.5.2 CFS SC Application

The following list summarizes the assumptions made about the CFS Stored Command Application as documented in this Guide:

- The CFS SC code has not been modified.
- The CFS SC application has been configured using the standard CFS SC configuration parameters.
- Core Flight Executive, (cFE) Application Programming Interface (API), and Operating System Abstraction Layer (OSAL) are being used.
- CCSDS File Delivery Protocol (CFDP) is being used.
- A command is sent to SC by the CFS Scheduler (SCH) application to schedule the application to wake up at a 1Hz rate. This is required as the resolution of the commands is one second.

1.6 *How to Use this Document*

This Application User's Guide is set up as a learning tool before launch, with an Appendix as a reference tool after launch. It can be read offline as a paper printout, or the electronic file can be searched. It may be used along with Doxygen generated html files for more complete understanding.

Read the sections from the beginning up to just before the Appendix as needed to understand how SC works. Refer to the Appendix for specifics when setting up operational scenarios.

Experienced flight controllers may only need to browse the Guide, and use the Appendix as needed. New flight controllers may wish to get more familiar with the entire Guide well before needing to rehearse operational scenarios or set up flight software procedures ("procs").

After launch, refer to the Appendix as needed.

1.7 *Acronyms and Abbreviations*

Acronyms and abbreviations in this publication are shown in Table 2 below. Telemetry and command mnemonics, and similar terms, are not shown.

Table 2 Acronyms and Abbreviations

Acronym	Description
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASIST	Advanced Spacecraft Integration & System Test software
ATP	Absolute Time Processor
ATS	Absolute Time [tagged command] Sequence
CCSDS	Consultative Committee for Space Data Systems
cFE	Core Flight Executive
CFDP	CCSDS File Delivery Protocol
CFS	Core Flight System
CI	Command Ingest Application
CM	Configuration Management
CMD	Command
DS	Data Storage Application
EEPROM	Electrically Erasable Programmable Read-Only Memory
FOT	Flight Operations Team
FSSE	Flight Software Sustaining Engineering
FSW	Flight Software
GPM	Global Precipitation Measurement

Acronym	Description
GRP	Group
GSFC	Goddard Space Flight Center
HEX	Hexadecimal
HK	Housekeeping Application
HS	Health and Safety
HZ	Hertz
ID	Identification
ISR	Interrupt Service Routine
LRO	Lunar Reconnaissance Orbiter
MMS	Magnetospheric Multiscale Mission
MOT	Mission Operations Team
OS	Operating System
OSAL	Operating System Abstraction Layer
PDF	Portable Document Format
PDL	Project Development Lead
RAM	Random Access Memory
RDL	Record Definition Language
RTF	Rich Text Format
RTP	Relative Time Processor
RTS	Relative Time command Sequence
SB	Software Bus Service
SC	Stored Command Application
SCH	Scheduler Application
SDO	Solar Dynamics Observatory
STOL	Systems Test and Operations Language
TAI	International Atomic Time
TBL	Table
TO	Telemetry Output Application
UTCF	Universal time code format
UTC	Universal time code
UT	Universal Time

Chapter 2. Introduction to the CFS SC Application

2.1 *Heritage*

Stored Command (SC) differs somewhat in architecture than many of the other CFS apps because it has a very long and very stable reuse heritage from multiple missions.

Lunar Reconnaissance Observatory (LRO) inherited its version and changed it very slightly to be usable with the CFS architecture. The LRO team changed the bus calls, the event message calls, and very little else.

After LRO, the CFS team took LRO's Stored Command and made it significantly more CFS compliant, adding configuration parameters in place of hard coded parameters, and making changes to fit SC better into the larger CFS architecture. Others added the ability to append Absolute Time [tagged command] Sequence (ATS) commands.

cFE Table Services was updated to allow a command to be registered with the table. This command is sent to the table owning application to tell it an update is pending. This change was ported to the CFS/SC application. Both MMS and the Global Precipitation Measurement (GPM) mission have this version of SC. This feature is transparent to the ground operator and there are no additional commands to be sent from the ground (beyond load/validate/activate). This feature is available for any CFS application to use, but as of the date of publication of this Guide, only SC is using it.

2.2 *CFS Stored Command (SC) Overview*

The Stored Command (SC) application provides a spacecraft with the ability to be commanded 24 hours a day using sequences of commands that are loaded via tables by the ground controller and stored for later execution. Each stored command has a time tag, permitting the command to be released for distribution at predetermined times.

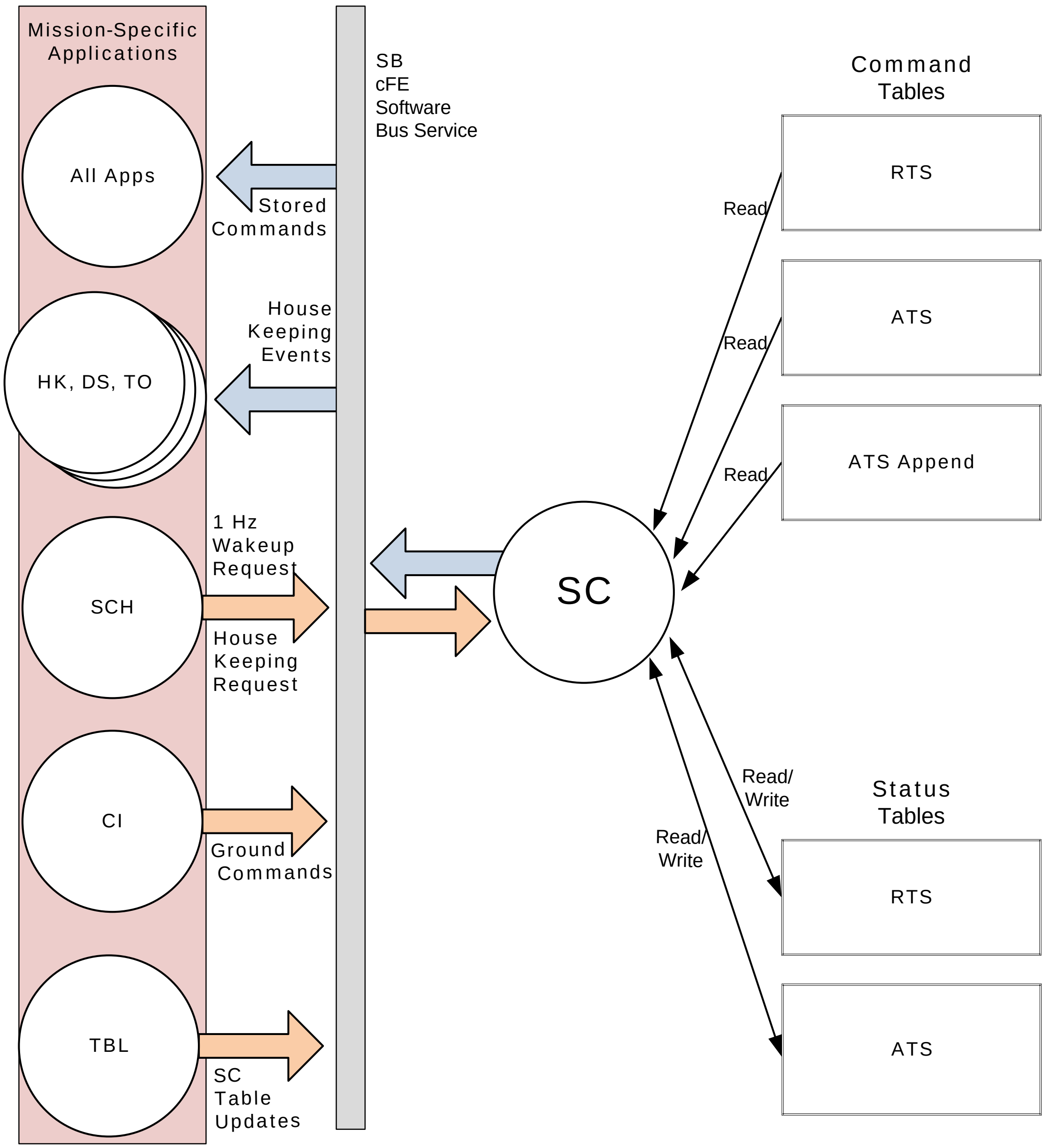
The size and number of commands is a configuration parameter. Configuration parameters are listed in Appendix A, and cannot be changed by the FOT after the SC program has been compiled.

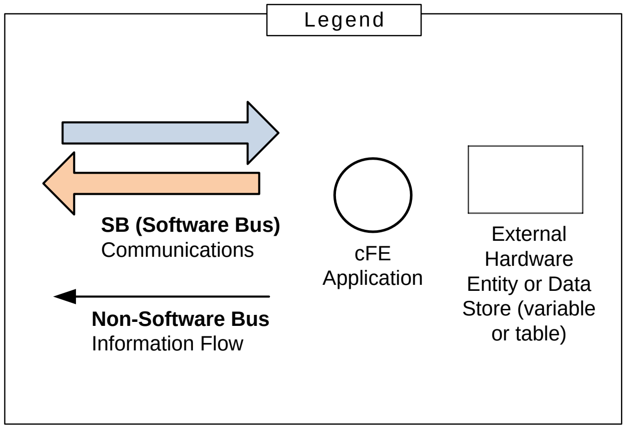
SC supports two types of time tagged command sequences. The first type, absolute time tagged command sequences (ATSs) are timed to execute at some absolute point in time, as measured by the configured time, described further in the section "Universal Time Code (UTC) vs. International Atomic Time (TAI)" below. SC supports two ATSs at one second granularity. Only one ATS can be active at one time.

The second type, relative time sequences (RTSs) are command sequences which execute at some point in time relative to the previous command in the relative time command sequence. The number of RTSs that SC supports is defined at compile time of the flight code, at one second granularity.

Offsets of zero (0) are allowed. They are executed in the same second as the command above it.

SC provides flight operators the ability to load and dump command sequences via tables.

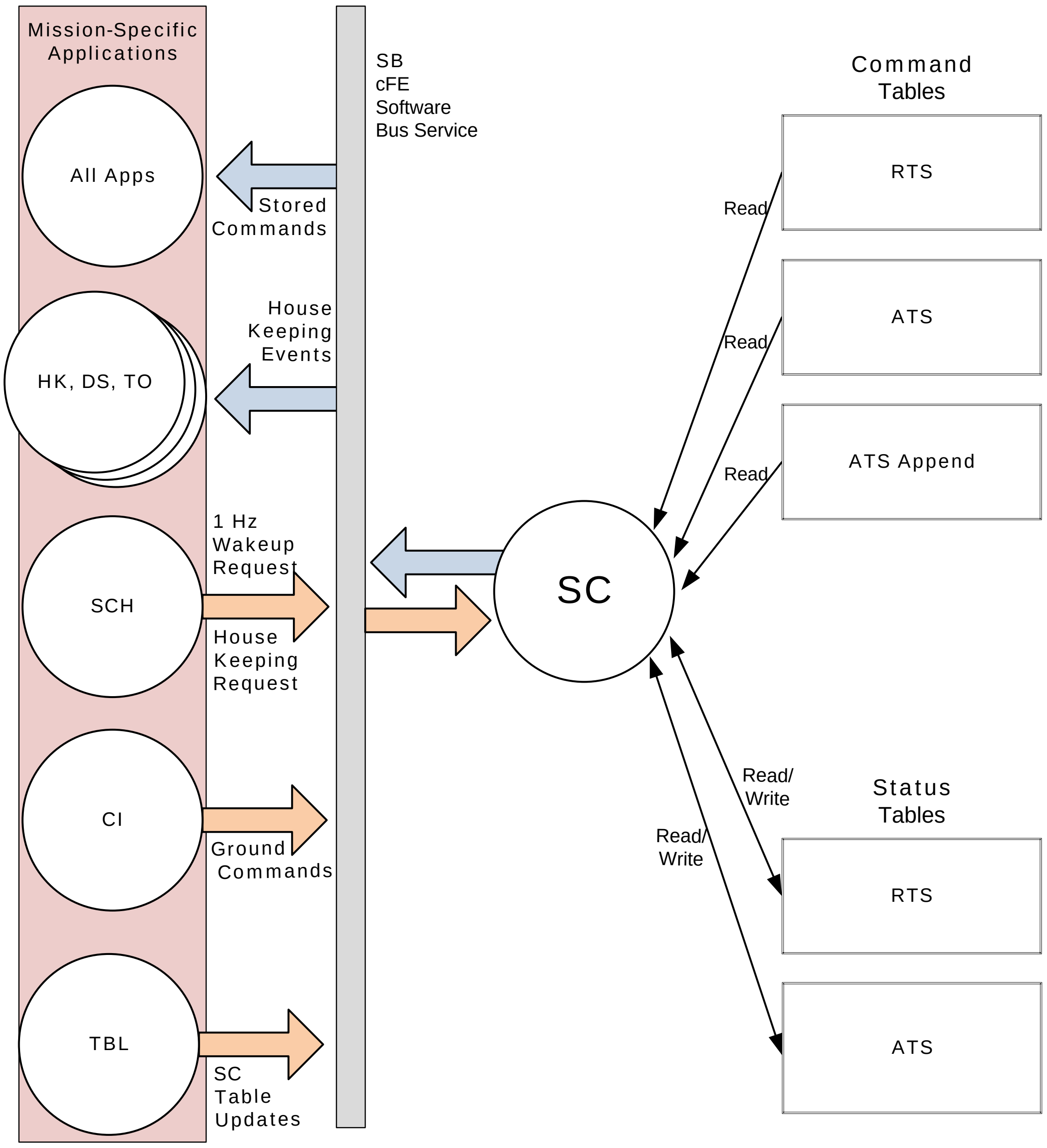


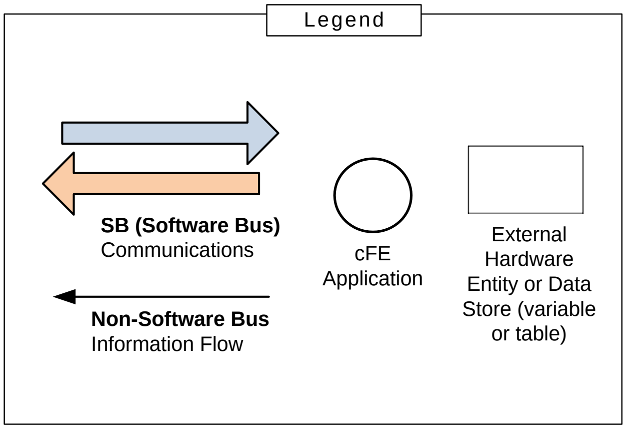

Legend
SB (Software Bus) Communications
Non-Software Bus Information Flow
cFE Application
External Hardware Entity or Data Store (variable or table)

Notes

- Everything to the right of the software bus is designed to be the same for SC on every mission.
- *To the left of the software bus are mission-specific applications. This diagram arbitrarily assumes CI, SCH, HK, DS, and TO are being used.*
- *All messaging is done over the Software Bus. Any application could be on the receiving end as long as it has subscribed to the appropriate messages.*
- *The CFS is highly configurable. This diagram shows one possible configuration. However, depending on the mission configuration, specific packets may, or may not, be routed to the SC app.*

Figure 1 below shows the software context for the CFS Stored Command (SC) Application.




Legend
SB (Software Bus) Communications
Non-Software Bus Information Flow
cFE Application
External Hardware Entity or Data Store (variable or table)

Notes

- Everything to the right of the software bus is designed to be the same for SC on every mission.
- *To the left of the software bus are mission-specific applications. This diagram arbitrarily assumes CI, SCH, HK, DS, and TO are being used.*
- *All messaging is done over the Software Bus. Any application could be on the receiving end as long as it has subscribed to the appropriate messages.*
- *The CFS is highly configurable. This diagram shows one possible configuration. However, depending on the mission configuration, specific packets may, or may not, be routed to the SC app.*

Figure 1 SC Application Software Context

2.2.1 Scheduler Application (SCH)

As shown in Figure 1 above, the Scheduler Application (SCH) sends periodic commands to SC (as defined in the SCH Schedule Table) in order to wake up the SC application so it can process the ATS and RTS tables. Additionally, SCH generates a request for SC's housekeeping message.

2.2.2 Command Ingest Application (CI)

Ground commands come from the Command Ingest Application (CI), as shown in Figure 1 above. Messages are routed to SC by the cFE Software Bus Service (SB). SC learns of ground updates to the SC tables through the cFE Table Services application (TBL).

2.2.3 Housekeeping (HK), Data Store (DS), Telemetry Output (TO)

Messages generated by SC are routed to whatever mission-specific applications subscribe to them, such as Housekeeping (HK), Telemetry Output (TO), and/or Data Storage (DS) Applications, as shown in Figure 1 above.

2.2.4 All Apps

Commands sent from the SC application are sent to all other applications that have subscribed to the command message, as shown in Figure 1 above.

2.2.5 Tables in Support of ATSs and RTSs

SC contains these tables in support of ATSs and RTSs:

Command Tables

1. RTS Tables – define the relative time sequence commands. There are separate tables for each relative time sequence
2. ATS Tables – two tables which define the absolute time sequence commands
3. ATS Append Table – command table; commands will be added to the end of the ATS being appended

Status Tables

1. RTS Status Tables - dump only tables used to report the status for all of the RTSs
2. ATS Status Tables – dump only tables used to report ATS status for both of the ATS tables

2.2.6 The Need for ATS Tables

ATS tables are a sequence of Absolute Time Sequence commands. The purpose of Absolute Time Sequence commands is to be able to specify commands to be executed at a specific time. Examples of the need for ATS commands are fixed times when the spacecraft will be over a downlinking station; a fixed time when a celestial object must be imaged; fixed times when a burn needs to be performed; or fixed times when the spacecraft needs to do housekeeping of certain data, for example.

The FOT specifies a particular time, and a particular command for each ATS command. An ATS table is a collection of those times and commands, with header and identifying information.

Depending on the mission, an entire ATS table may specify commands for a longer mission-specific period of time. Or, it may last only a much shorter mission-specific period of time, depending on how many software activities need to be specified during that time.

There are two ATS tables so that FOT can upload and configure a new one while an existing one is executing.

For Absolute Time Sequence commands, there is a field in the command that represents the absolute time in seconds that each command shown in the ATS Tables in Figure 1 above will execute.

Much more of a discussion of ATS and RTS follows in the sections below.

2.2.7 RTS Tables

RTS tables are a sequence of Relative Time Sequence commands. The purpose of Relative Time Sequence commands is to be able to specify commands to be executed at a specific time *after* ("relative to") an ATS.

For Relative Time Command Sequence commands there is a field that represents the time in seconds that the command will *delay* before executing. This delay is relative to the time when the previous Relative Time Tagged Command (RTC) was executed. In the case of the first command of the sequence, this time is relative to when the sequence was started.

More details of timing and format for RTS tables are shown in Chapter 3.

2.2.8 Dump-Only Tables

SC uses the following dump-only tables for status information which can be useful in debugging situations. These tables are defined as single buffer tables with the structures defined below.

2.2.8.1 Absolute Time Processor (ATP) Control Block Table

The ATP Control Block table contains data items to keep track of the execution of an ATS on the ATP.

See also: Table 7, ATP Control Block Table, on page A-1.

2.2.8.2 ATS Information Table

The ATS Information table contains data items that provide information about the ATS.

The ATS Information table is defined by SC_AtsInfoTable_t. The SC application defines one of these tables for each ATS table created.

See also: Table 8, ATS Information Table, on page A-1.

2.2.8.3 ATS Append Information Table

There is only one ATS Append Info Table defined and it provides information about the ATS Append table described earlier

The ATS Append Info table is defined by the same structure as the ATS Information table described above.

2.2.8.4 RTS Information Table

The RTS Information Table is used to keep the current status of each RTS table.

This table contains one record for each RTS table as defined by SC_RtsInfoEntry_t.

See also: Table 9, RTS Information Table, on page A-1.

2.2.8.5 RTP Control Block Table

The RTP Control Block table contains data items to keep track of the RTSs executing on the RTP.

See also: Table 10, RTP Control Block Table, on page A-2.

2.2.9 Universal Time Code (UTC) vs. International Atomic Time (TAI)

This is the time used by SC (CFS SC Configuration Parameter SC_TIME_TO_USE). Choices are Universal Time Code (UTC), TAI, or Default.

UTC is Universal Time Code.

TAI is International Atomic Time. Note that leap seconds are not applicable; TAI does not use leap seconds whatsoever.

Default gives the time in whatever format the cFE is set to use.

The time format (TAI or UTC) only affects ATSs, not RTSs, since RTSs contain only offsets from the time they were started.

2.3 *Design Overview*

2.3.1 What Drives SC

SC is driven by the Scheduler Application (by the 1 Hz packet and housekeeping request packets) and by ground commands.

As shown in Figure 2 below, **SC is driven off of the *1 Hz packet*** from the Scheduler Application. When the 1Hz packet is received, SC looks to see if there are RTS or ATS stored commands to execute in this second of time.

SC is also driven off the *housekeeping request packet* from the Scheduler (SCH) Application. When SC receives the housekeeping (HK) request, it processes the request, sends out the housekeeping packet, and looks for table updates to the load/dump tables that SC maintains.

SC will not update a table if that table is currently being used. If it is not being used, the ATS or RTS is updated as required.

SC is also driven by *ground commands,* and responds to them as they are received.

ATS commands are sorted by time.

2.3.2 How SC Handles Multiple Commands

If there are commands to be executed, SC will execute them up until a mission-specific number of commands (CFS SC Configuration Parameter SC_MAX_CMDS_PER_SEC).

If there are more commands than specified, SC tries to catch up during the next and subsequent seconds. ATSs gets priority, followed by RTSs.

RTSs get priority in order of their numbers (for example, 1 gets priority over 5). If, for example, the limit was eight commands per second, SC would try to send eight commands per second until it was all caught up.

Figure 2 shows the flow control for SC software commands.

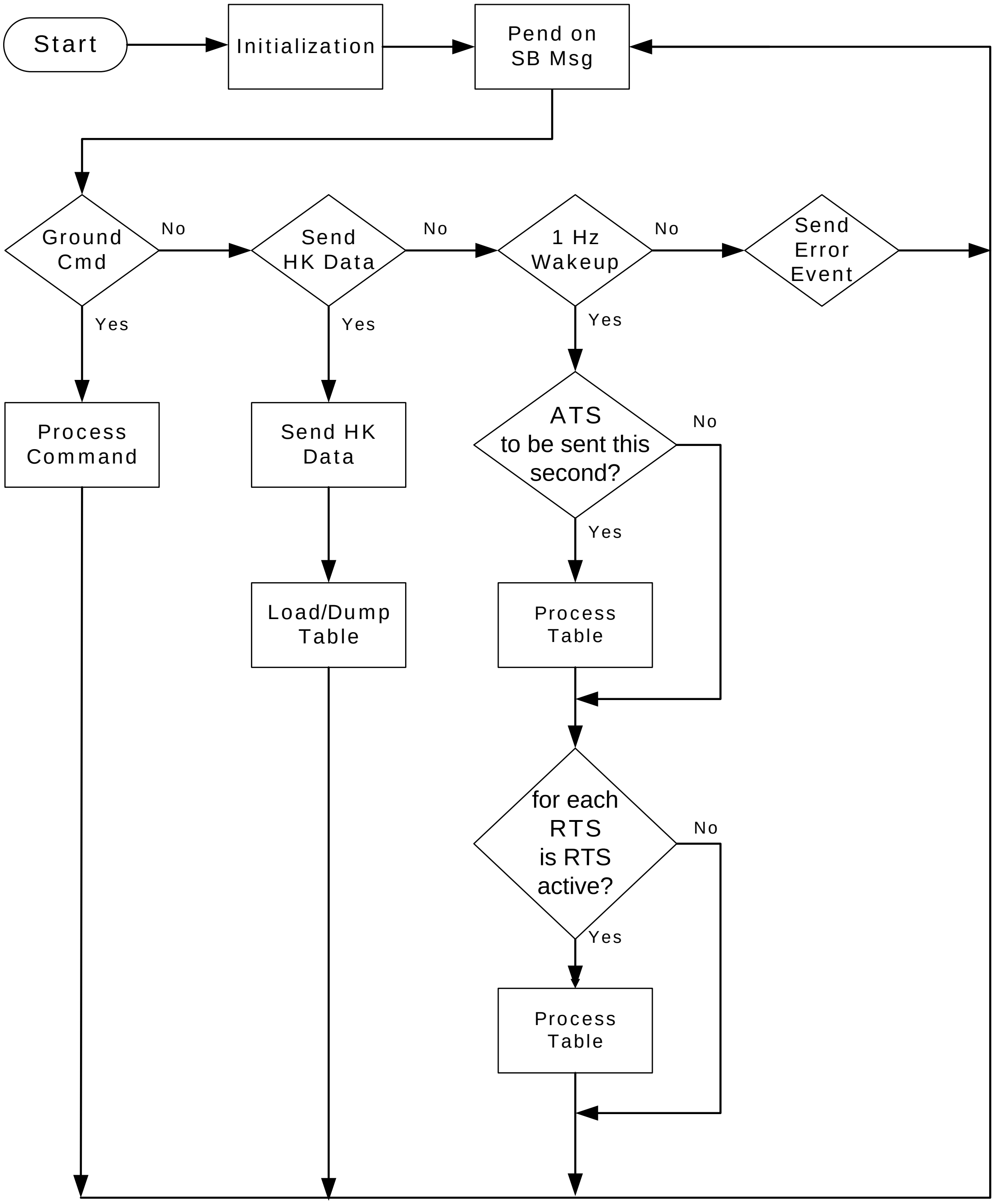


Figure 2 Flow Control Diagram

This page intentionally blank

Chapter 3. **CFS SC Normal Operations**

3.1 *Initializing SC*

There are two types of SC initializations: processor reset, and power-on reset. SC can do different things on initialization depending on the type of initialization taking place. As with any file, any RTS or ATS table file that is stored on a RAM drive will not survive a Power On reset.

- **On power on reset,** SC executes RTS 1.
- **On processor reset**, SC executes RTS 2.
- Both RTS 1 and 2 are reserved for initialization commands that can be easily changed, rather than inline, hard-coding of the startup procedures.

SC does the following at initialization, **regardless of reset type:**

- SC-specific initialization.
- Initializes all housekeeping telemetry counters.
- Initializes the RTS Information Table.
- Initializes the Absolute Time Processor (ATP).
- Initializes the RTS and ATS status tables.
- Loads default RTS tables from Electrically Erasable Programmable Read-Only Memory (EEPROM) into Random Access Memory (RAM).
- Initializes housekeeping data. ***See also:*** Table 124, Housekeeping Telemetry Initialization Summary, on page A-35.

3.2 *Using the Absolute Time Processor (ATP)*

The Absolute Time Processor (ATP) manages the execution of Absolute Time Sequences (ATSs). The ATP is used to schedule spacecraft commands for an extended period of time. The ATP manages two buffers of ATSs. Only one ATS can be active at one time.

The ATP is controlled by commands from the Software Bus. These commands can come from the ground controller, other subsystems, or SC itself.

Each command received by SC will be fully validated before it is executed and can be verified in the telemetry to determine the status of the execution.

Under normal operations one of the ATS buffers will be loaded with all the commands anticipated to operate the spacecraft for a mission-specific period of time. While one ATS buffer is being used, the second ATS buffer can be prepared for the next period.

The job of the ATP is to send out the commands in the ATS buffer in time order. Each command will be sent out to the data system when the command time tag matches the configured Time. Because the ATS command time tag only has seconds, the resolution of each ATS command is in seconds.

Commands with the same time tag are legal and will be sent to the data system in the same second. In the case where the time tags are equal, the command with the lower command number will be executed first.

When it is time for an ATS command to be executed, the corresponding command status from the ATS Command Status Table is checked. If the command status is valid (i.e. LOADED) the command will be executed. If the command status indicates any status other than LOADED, the command is not executed.

After the command status is checked and determined to be valid, the checksum on the ATS command is verified. If the checksum fails, the status of the command in the ATS Command Status Table will be marked as FAILED_CHECKSUM. If the checksum is valid, the command is sent out to the data system via the Software Bus.

If the command is targeted for SC (i.e. START RTS), the Software Bus will route the command back to the input pipeline for SC.

After the command has been sent through the Software Bus, the return status will be checked. If the Software Bus was able to send the command with no errors, the status will be marked as EXECUTED. If the Software Bus returns an error code, the status will be marked as FAILED_DISTRIBUTION. After a command has been sent through the Software Bus, the ATP sets up the next ATS command for processing. If the ATP detects the end of the ATS buffer, it will stop the ATS. After finding the next command (or stopping the ATS buffer) the ATP will return control to the main function of SC.

3.2.1 ATS Commands

3.2.1.1 Starting an ATS

This command is used to start the execution of an ATS.

Upon receiving this command, the ATP makes several checks to determine if it can execute the command.

The checks include:

- Validate the ATS buffer parameter in the packet.
- Determine that the ATS has at least one command.
- Determine that the ATP is not already executing an ATS.

If all of the checks pass, the ATP initializes the ATP control block with all of the information it needs about the ATS. The ATP will then walk down the list of ATS commands until it finds the first one with a time that is greater than or equal to the current time.

Any command that has a time tag that is less than the current time (i.e. expired) will be marked in the ATS Command Status table as SKIPPED.

Important: The ATP will not execute commands that have a time tag that is "in the past" compared to the configured time source. So if the ground controller starts an ATS that was meant for yesterday, all of the commands in the sequence will be skipped and the ATS will be stopped.

3.2.1.2 Stopping an ATS

This command is used to stop the execution of the ATP.

After stopping an ATS it can be restarted by sending the Start Absolute Time Processor command.

Assumptions. It is assumed that there is an ATS running at the time that this command is sent, but sending the command when the ATP is idle will not cause an error. If the command stops a currently running ATS, the ATP state in the ATP control block will change to IDLE.

3.2.1.3 Switching the ATS Buffer

This command is used to switch from one ATS that is currently running on the ATP to the alternate ATS.

This command must pass the following checks before it is accepted:

- The first check determines that the ATP is currently executing an ATS.
- The second check determines that there is at least one command in the alternate ATS buffer.

If the checks pass:

- The ATP will stop executing the commands in buffer A (or B) and start executing commands in buffer B (or A) at the next one-second boundary in spacecraft time.
- If there are no commands in the alternate ATS buffer, the current buffer will not be stopped.
- If there are commands in the alternate buffer and they all have bad status or time tags in the past, then all ATS processing will be stopped.

In normal operations, one ATS buffer will be running on the ATP, executing commands, while the unused command buffer can be loaded and patched with the next sequence of commands.

Because of the way the ATP skips commands that have time tags "in the past" when the ATS is started, the two ATS buffers can be built with some overlap in them. This allows a window of time during which the switch command could be sent. Figure 3 shows the overlapping buffer concept with the ATSs.

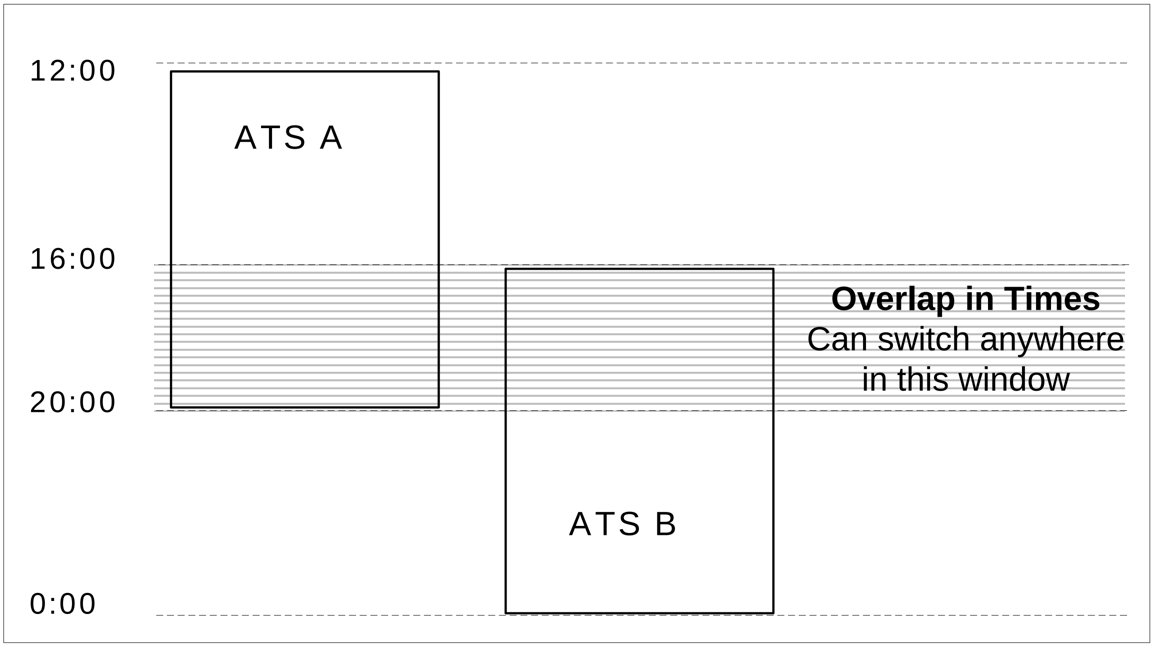


Figure 3 Overlapping ATS Buffers

As mentioned earlier, the ATP can execute commands with the same time tag.

Special switch logic comes into use when there are multiple commands per second.

Because the resolution of the time tag only goes down to one second, in order to execute more than one command in one second, the commands should have the same time tag.

Because the ATS switch is one of the most complicated SC operations, it is important to discuss the complete details of the ATS switch here.

3.2.1.3.1 *Sample Scenario*

Suppose that the ATP receives the command to Switch ATSs while one ATS is in the middle of sending eight commands out with the same time tag, and only five of these commands were sent out before the buffer is switched (assuming a switch command coming from the ground controller, not the internal switch command).

At first glance this seems to work because the other ATS buffer has an overlap with the same eight commands that want to go out in one second. This is where a problem occurs: when the new ATS is started, the ATP will walk down the list of commands until it finds a command with a time tag that is greater than or equal to the current time tag. Because the resolution of the time tags only goes down to a second, the ATP will execute all eight commands in that one second, causing five of the eight commands to be sent out twice.

In order to solve the problem of the ATS Switch sending out duplicate commands, the Switch ATS command received from the ground controller causes a wait condition until it is "safe" to switch the ATS buffer. So, when the Switch ATS command is received by the ATP, the command is validated and then a SWITCH_PEND flag is set up.

When it becomes safe for the ATS to switch (i.e. at a one second boundary in the configured time source), the switch will be serviced. This method assures that sending the Switch ATS command by the ground controller will not cause any duplicate commands to be sent out to the data system, or any missed commands. Note that all of this switch logic only comes into use when there are multiple commands per second.

The safe switch will wait until all commands during the current second have been sent out, then it will switch. If there are no commands with the same time tag in the overlap region (including the switch command) this logic does not get used. In either case, the switch can be performed without sending any duplicate commands. (Note that on some missions, the Mission Operations Team (MOT) may choose to include an ATS switch buffer command as the last command in each ATS load. If this is done, the overlap feature offers little to no benefit. See Appendix A for mission-specific values.)

There are certain conditions that can cause an ATS switch that is pending to be canceled. If the ATP is stopped by the ground controller while the ATP is waiting for a "safe" time to switch, then the switch will not occur. Also if the ATP detects the end of the ATS buffer before there is a "safe" time to switch, the switch will be canceled. Note that the typical time frame for a "switch pend" is one second or less. The operation of the switch is transparent to the ground controller.

3.2.1.4 Jumping within an ATS (ATS Jump)

This command is used to jump to a specified time in the currently running ATS.

The jump command will effectively restart the ATS at the time, in seconds of the configured time source, given in the command.

Because there is no restriction on the time given in the command, the ATP may try to restart the ATS at any time before or after the current time. In the case of the time tag being before the current time, (a backwards jump) the ATP will simply skip the commands that have been executed (or failed execution) and end up at the same location as before.

In the case of the jump time being after the current time, the ATP will skip all commands with time tags less than the jump time and start executing the ATS at the time equal to the jump time. If there are no commands with time tags equal to the jump time, the ATP will set up the ATS to wait for the first command after the jump time. When a command is skipped while doing the jump, the command's status is marked as SKIPPED unless it has already been marked as EXECUTED, FAILED_DISTRIBUTION, or FAILED_CHECKSUM. An event message will be generated for each command skipped.

3.2.2 Understanding ATP Error Handling

Error checks are built into the ATP to assure that it does not cause any ill effects on the spacecraft. Error handling within the ATP is of four general categories:

1. Ground command validation,
2. Sequence error detection,
3. Load error detection, and
4. Memory error detection.

3.2.2.1 ATP Ground Command Validation.

Each ground command sent to SC is first checked to see if the command packet length matches the expected length. Second, SC checks if the command parameters are valid. For example, the command to execute ATS number 24 will never be accepted, because it is not valid.

After the parameter checked, there is a logical check on each command. An example of the logical check is the Ground Switch command checking to see if the ATP is actually executing an ATS before a switch is attempted. If the ATP is idle, then it makes no sense to switch the ATS buffer.

The exception to this rule is the Stop ATS ground command. This command will be executed even if there is not an ATS currently running. This feature is useful in the very unlikely case that the ATP is running an ATS but does not "think" it is actually running an ATS, i.e. processor memory becomes corrupted. (The Stop ATS command simply cleans up the ATP control block regardless of the ATP status, and can be contained in the ATS. Upon receipt of a Stop ATS Command, SC stops processing the currently executing ATS and sets the state of that ATS to IDLE.)

3.2.2.2 ATP Sequence Error Detection

The ATP checks for errors within ATSs, including invalid packet IDs. It also checks for end of buffer.

The ATP also checks the ATS Command Status Table for each command to make sure that each command is loaded and ready.

3.2.2.3 ATP Load Error Detection

As mentioned in the section on ATS loads in Section 3.4 on page 3-18, the ATP also parses through the load to try to assure that the ATS loads are valid and not corrupted. If the ATS does not pass the load checks, it will be rejected. This check will eliminate many of the run-time errors before the ATS gets started.

3.2.2.4 ATP Memory Error Detection

The ATP also does memory error checking. Before each command is sent out, the checksum in the command secondary header is recomputed and compared to the checksum stored with the command. If the checksum fails, the command is not sent out and the appropriate entry in the ATS Command Status Table is set to FAILED_CHECKSUM. Also the command number is checked to make sure it matches the number in the Time Index Table.

The ATP can either continue processing commands after it receives one with a failed checksum, or stop running altogether (settable by command).

3.2.3 Understanding ATS Command Format

An ATS is a collection of ATCs. SC supports two ATSs and one ATS Append Table. ATCs do not have to be in time or ID number order.

Figure 4 shows the format of an Absolute Time Tagged Command. Each command executed by the ATP must be in this format. The fields of the commands are:

ATS Command Format		
← Command ID No. →	← Absolute Time Tag →	← CCSDS command packet →

Figure 4 Absolute Time Sequence Command Format

- Absolute Time Tag - This field is a 32 bit integer that represents the absolute time, in seconds, that each command will execute.
- Consultative Committee for Space Data Systems (CCSDS) command packet - This field is a variable length CCSDS Telecommand Packet containing a command for the data system.

> **Note:** If the command length (in bytes) is odd, a pad byte must be added to the RTS command structure so that the next command starts on a word (uint16) boundary.

Figure 5 shows a typical absolute time command within an absolute time sequence.

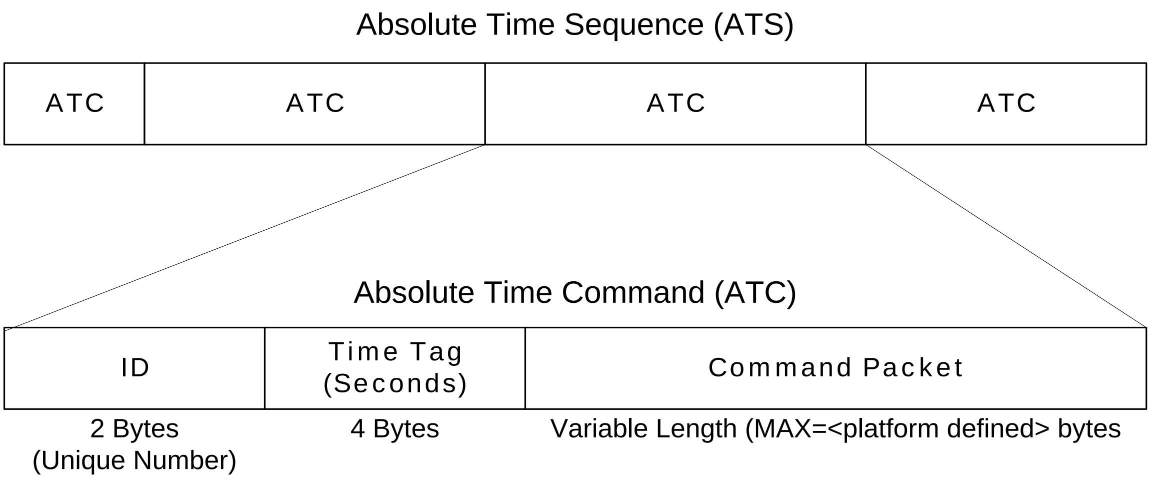

Absolute Time Sequence (ATS)
ATC
ATC
ATC
ATC
Absolute Time Command (ATC)
ID
Time Tag (Seconds)
Command Packet
2 Bytes (Unique Number)
4 Bytes
Variable Length (MAX=<platform defined> bytes

Figure 5 ATC within an ATS

3.2.4 ATP Loads

3.2.4.1 ATP Load Sequence

Before the ATP can be used, an ATS must be loaded. An ATS may be loaded into ATS Buffer A or ATS Buffer B. The load is described here in two parts: the preparation necessary on the ground controller to format an ATS load and the processing that is done by the Stored Command Processor to accept an ATS load.

3.2.4.2 Ground Preparation for an ATS Load

To prepare an ATS load, follow these steps:

1. Plan the sequence to be loaded. Address the spacecraft configuration, commands, times for the commands, size of the commands, and other SC activity.
2. Map out the ATS in Systems Test and Operations Language (STOL) Command Mnemonics with the appropriate configured time tags.
3. Translate the STOL command mnemonic procedure into a loadable file by the FOT.

3.2.4.3 Onboard Preparation of an ATS Load

The next step in the load is to send the load file containing the ATS to the flight system as a table load.

Table 3 RTS Table Load Sequence

Step	Description
1	CFTP the table (.tbl) file to the spacecraft.

2	Send the table load command.
3	Send the table validate command.
4	Send the table activate command.

If SC finds an error in the ATS, such as duplicate command numbers, invalid command lengths, or commands that run off of the end of the buffer, it will reject the ATS and send an event message that the ATS was rejected.

Figure 6 shows a *sample* ATS load. The first part shows the command mnemonics and the second part shows the actual load data as it would be put into the load packets.

Sample Absolute Time Sequence

Command Mnemonics

Command No.	Configured Time Source	*Command*	**Comment**
1	1	*SNOOPCMD*	No operation
2	30	*SCIRESET*	Reset Command Ingest (CI) Counters
5	0	*SNOOPCMD*	No operation
22	54	*SSCSWITCHA*	Switch ATS
See numbers in bold in table below	See plain numbers in table below	*See highlighted numbers in table below*	

For ATS

Binary Command Data in ASCII HEX format

(Note: cFE file header and cFE table header not shown)

First→ Byte

```
0001  0001 0000  1801  C000  0001  0100  0002 001E
0000  1801  C000  0001  0200  0005  0000 0000  1801
C000  0001  0100  0016  0036 0000  180E  C000  0001
0500  0000  0000  0000  0000  0000  0000  0000  0000
0000  0000  0000  0000  0000  0000  0000  0000  0000
0000  0000  0000  0000  0000  0000  0000  0000  0000
0000  0000  0000  0000  0000  0000  0000  0000  0000
                    .........
0000  0000  0000  0000  0000  0000  0000  0000  0000
```

Last ←Byte

Note: The mnemonics and the corresponding hex values of the commands in this example are for illustration only, and do not represent actual commands for any specific mission. See the command database for actual command mnemonics and their hex equivalents.

Figure 6 Sample ATS Load

After the load process is complete, the ATS can be started. An important item to note is that one ATS buffer can be loaded and validated while the ATP is executing the other ATS buffer.

After the ATS has executed to completion (or skipped or failed), all of the commands will have a status other than LOADED attached to them. If the ground controller were to start the ATS again, it would not output any commands. The ATS can be reloaded without actually sending all of the data up to the spacecraft. To reload an ATS, you would use ASIST or CFDP to transfer the table to the FSW. The cFE Table Manager would then load the new table. The next 1 Hz command tone would then process the table in the stored command FSW. This reloading causes the ATP to reset all of its tables and parse through the commands again.

The following information is contained in the ATS info table: the number of ATS commands in the table, the size of the ATS, and how many times the ATS has been executed. The ATP control block contains the following information: the state of the ATP, the ATS number of what is currently in execution, the command number to run if any, the timer index pointer for the current command in execution, and switch the 'pend' flag.

3.2.5 Appending an ATS

The SC application allows you to add to an existing ATS.

The user loads the desired commands into the ATS Append Table using the cFE Table Services. The new commands will be added to the end of the ATS being appended. The new commands can replace existing commands in the ATS as long as those commands have not yet been executed.

The ATS being appended can be EXECUTING or IDLE. Once the data in the Append Table is loaded, validated and activated, the SC_APPENDATS command can be sent to add the contents of the Append Table to the specified ATS.

When the SC_AppendATS command is processed, the SC application will copy the contents of the Append Table to the specified ATS and rebuild the pending execution list with the new commands.

3.2.5.1 Summary: How to Use Append ATS

Assuming that one or both onboard ATS's (ATS A and/or ATS B) are already loaded:

1. Load the Append ATS table.
2. Send the Append ATS command to SC with an argument to append to ATS A or B.
3. SC will copy the Append ATS data to the end of the selected ATS.

3.2.6 ATP Precaution

Despite the many error checks built in to the ATP to assure that it does not cause any ill effects on the spacecraft, there is a crucial safety measure that is required of all ATS tables. The ATP relies on a sentinel word of zeroes at the end of an ATS table to signal the end of the ATS table (end of data marker). cFE Table Services, when loading a new table, unconditionally fills the table working buffer with the contents of the current table prior to placing the new table file contents into the buffer. If a newly loaded ATS table does not contain the sentinel word of zeroes at the end of the table and is smaller than the table that was previously loaded, the newly loaded table has potential to execute the "old" invalid commands that were part of the prior table load. It is therefore essential for operators to place the sentinel word of zeroes at the end of each ATS table including ATS append tables.

3.3 *Using the Relative Time Sequence Processor (RTP)*

The Relative Time Sequence (RTS) Processor (RTP) is the SC component that manages the execution of RTSs. An RTS can be thought of like a subroutine in a computer program; a small set of instructions that can be reused. The Absolute Time Sequences handle the actual scheduling of the daily operations, but it is inefficient to put the same sets of commands over and over in the ATS buffer instead of calling an RTS.

An RTS can also be used to provide services for other spacecraft subsystems. The Limit Checker (LC) application, for example, can request the execution of an RTS that contains a group of commands to change the state of a particular subsystem. The RTP can manage the concurrent execution of up to 256 RTSs (CFS SC Configuration Parameter SC_NUMBER_OF_RTS).

3.3.1 Understanding RTP Memory Usage and Layout

Each RTS must be stored in a buffer that is maintained by the RTP. There are up to 256 (CFS SC Configuration parameter SC_NUMBER_OF_RTS) of these fixed size sequence buffers available; each buffer is 300 bytes (CFS SC Configuration Parameter SC_RTS_BUFF_SIZE).

Relative Time Sequences that are larger than 300 bytes (CFS SC Configuration Parameter SC_RTS_BUFF_SIZE) can be constructed by using more than one RTS buffer by using the last command in an RTS buffer to start the next RTS buffer in the sequence.

The *RTS info table* (a dump only table) maintains the following information:

- RTS Run status,
- whether the RTS is enabled or disabled,
- the next command time,
- the next command pointer in the table to the RTS,
- the command counter,
- the command error counter, and
- how many times the RTS was run.

See also: Table 9, RTS Information Table, on page A-1.

The *RTP control block* contains the following information:

- The number of RTSs that are active
- The RTS number to be run next
- These get initialized on a processor reset or power on reset.

3.3.1.1 RTS Format

An RTS consists of a series of Relative Time Tagged Commands packed in the RTS buffer. Because of the variable length commands within the RTS buffer, the RTS does not have a fixed format. The RTS sequence does have the following format restrictions:

1. The first RTS command must begin on the first word of the RTS buffer.
2. There must not be any gaps between the RTS commands.
3. A pad byte must be added to RTS commands with that an odd command length (in bytes), so that the next command starts on a word (uint16) boundary.
4. The unused bytes at the end of an RTS buffer must be filled with zeros.

3.3.1.2 RTP Internal Structures

Similar to the ATP, the RTP maintains data structures to help manage RTSs. The RTP maintains an RTS Information Table and an RTP Control Block. Figure 7 shows the basic format of these structures.

See also:

Table 9, RTS Information Table, on page A-1

Table 10, RTP Control Block Table, on page A-2

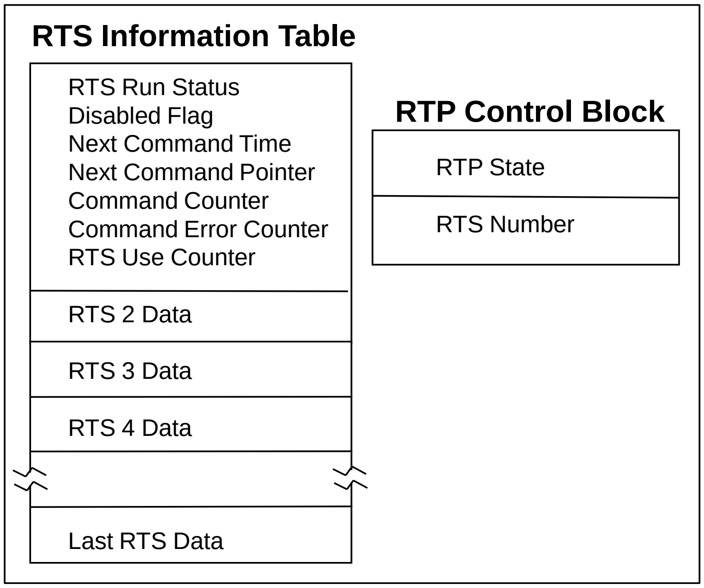


Figure 7 RTP Internal Structures

The RTS Information Table is used to keep the current status on each of the RTS buffers.

The table has one record for each RTS buffer. The table is sized by Each record in the RTS Information Table consists of the fields shown in Table 4 below.

Table 4 The RTS information table has one record for the current status of each RTS buffer

RTS Information Table Field/Flag	Purpose
rts_status	Used to keep the current status of that particular buffer. Can be EMPTY, LOADED, or EXECUTING.
disabled_flag	Indicates whether that particular RTS is disabled or enabled. If the RTS is disabled, it cannot be started.
next_command_time	The computed absolute time of the next command in the RTS that needs to execute.
next_command_pointer	An offset in 16-bit words into the RTS buffer of the next RTS command to execute.
command_counter	Counts the number of commands that have been sent out from this RTS during its most recent execution.
command_error_counter	Counts the number of errors that have occurred while sending RTS commands from this buffer. The RTS will stop when it encounters an error, and this is not configurable.

RTS Information Table Field/Flag	Purpose
rts_use_counter	Keeps track of the number of times that the particular RTS was started since it was loaded. The RTS Information Table can be dumped through the table dump facility of cFE.

The RTP Control Block is used to keep several important parameters for the RTP, shown in Table 5 below. These parameters are in SC housekeeping telemetry.

Table 5 The RTP control block keeps two important variables for the RTP

Field/Flag	Purpose
RTP state	Current state of the RTP. Can be either IDLE or EXECUTING.
RTS number	Current RTS number executing on the RTP. Only cleared out when another RTS is started, allowing the ground controller to note the last RTS that started on the RTP.

3.3.2 Commanding a Group of RTSs

Commands and event messages allow start/stop/enabling/disabling a group of RTSs. These commands are all enabled via a configuration parameter.

See also: For command details:

- Table 169, Command 13: Start a Group of RTS on page A-63
- Table 170, Command 14: Stop a Group of RTS on page A-64
- Table 171, Command 15: Disable a Group of RTS on page A-65
- Table 172, Command 16: Enable a Group of RTS on page A-66

See also: For configuration parameter which enables these commands:

- Table 139, Configuration Parameter - Define Inclusion State for RTS Group Commands on page A-46

See also: For the structure of the RTS Group Commands:

- RTS Group Commands Structure on page A-36

3.3.3 Running an RTS (RTP Normal Operation)

To run an RTS do the following:

1. First load the RTS into one of the available RTS buffers. To do this, first use the CCSDS File Delivery Protocol (CFDP) file upload procedure; then use the table load sequence.

Table 6 RTS Table Load Sequence

Step	Description
1	CFDP the table (.tbl) file to the spacecraft.
2	Send the table load command.
3	Send the table validate command.
4	Send the table activate command.
5	Ensure a minimum of five seconds (required) after the "Table Activate" command in the table load sequence and then send the Enable RTS command.

2. On a successful RTS load, the disabled flag for the sequence will be set. Once enabled, the sequence can be started. Once started, the RTP reads the delay of the first command.
3. After the number of seconds listed in the delay, the RTP will fetch the command, check the checksum of the command, and send the command out to the data system.
4. The RTP will then fetch the next command in the sequence and determine when this command needs to execute.

The RTP will continue to process commands in the RTS until one of the following conditions occurs:

1. The SCP receives a Stop RTS command from any source.
1. The RTP detects the end of the RTS sequence (zero value for the first word of the CCSDS command) or the end of the buffer.
2. The RTP encounters an RTS command with an invalid CCSDS length field. Because the RTS commands are variable length, the RTP relies on the length field in each command to find the next command. If the length field is not valid, then the RTP cannot assume that it can find the next command so it must stop the sequence.
3. The RTP encounters a command that runs past the end of the RTS buffer.
4. The RTP processes a command with a bad checksum.

The RTS will stop if it encounters the end of its buffer, so the Stop RTS command is not needed within the buffer. For example, if the last command in an RTS is a command to stop the RTS, the command will be 'sent out' to SC. Meanwhile, the RTP will detect the end of the buffer and stop the RTS. Then SC will process the Stop RTS command and stop an RTS that is already stopped. This is not an error, but it shows that the Stop RTS command is not needed within the buffer.

Each RTS can be started and run independently of the other RTSs. The only restriction is that an RTS cannot be started a second time while it is executing.

When there is more than one RTS running concurrently, each RTS has a priority associated with it. The priority is assigned by the buffer number. In other words, RTS buffer 1 has the highest priority and the last RTS buffer (up to 256) has the lowest priority. This priority only comes into play when there is more than one RTS that has commands to be executed in the *same second*.

Example. For example, if RTS 1 has a command to execute at 12:00:01 and RTS 50 has eight commands to execute at 12:00:00, all eight commands from RTS 50 will be executed before RTS 1 executes its command. However, if the eight commands from RTS 50 are scheduled to go out at 12:00:01, then the command from RTS 1 will be sent first, followed by seven commands from RTS 50. At 12:00:02, the eighth command from RTS 50 will be sent.

- This example assumes SC_MAX_CMDS_PER_SEC is set to 8. This number of course can be changed.
- This example uses RTS commands, as opposed to ATS commands, which are marked *skipped* if they don't go out in the second requested, i.e., if the current time is greater than the SEND time. For example, if one started an ATS at 12:30 that has commands to be sent at 12:00 and 12:01, the commands would be marked SKIPPED instead of being run later.
- The total number of commands allowed per second is the sum of the RTS commands and the ATS commands.

3.3.4 Using RTP Commands

SC can process external commands to configure and control the RTP. This section describes the commands.

3.3.4.1 Starting RTS

This command is used to start the execution of an RTS.

Upon receiving this command, the RTP will go through the following checks:

1. Validate the Relative Time Sequence number.
2. Make sure that the specified RTS is not executing already.
3. Make sure that the specified RTS is loaded.
4. Make sure that the rts_disabled flag is clear in the RTS Info Table.
5. Make sure that the first command in the RTS has a valid length.

Once the command is validated, the RTS will start executing the RTP. It will execute until it is terminated by one of the previously-mentioned conditions.

3.3.4.2 Stopping RTS

This command is used to stop the execution of an RTS.

The command has one parameter, RTS number. This command is valid when the specified RTS is idle. The command causes associated data structures to be reset.

3.3.4.3 Disabling RTS

This command is used to set a flag that inhibits the starting of the specified RTS.

The specified RTS will be disabled after the last command is executed.

The single parameter in this command, RTS buffer number, is checked to see if it is valid. After validation, *enabled_flag* for that particular RTS is cleared in the RTS Information Table. Once the flag indicates that the RTS is disabled, the RTS will not be allowed to start.

This command does *not* check to first see if the sequence is currently running, so it will have no effect on the execution of the sequence in progress at the time when the command is received. However, it will prevent future attempts to start the RTS.

3.3.4.4 Enabling Relative Time Sequence

This command is the opposite of the Disable RTS command.

This command will simply reset the flag, allowing the RTS to start.

3.3.5 Understanding RTP Error Handling

Like the ATP, the RTP performs many error checks. The RTP detects and handles errors in ground commands, in sequences and in memory.

- Each ground command sent to the RTP must pass validation checks on each of the parameters required for that command.
- After the parameter check, a logical check is performed to see if the command can be safely executed. For example, if a command to Start RTS number 1 is sent and RTS number 1 is already executing, then the command is rejected because that RTS is already in use. Any command that is rejected is reported by an event message as well as in telemetry.

In addition to ground command errors, the RTP detects a number of sequence errors once the RTS is running. The RTP checks for:

- Invalid packet length
- Invalid packet application ids
- Command packets that run off the end of the RTS buffer
- End of buffer

The RTP also checks for memory errors. When it is time for the command to be sent out to the data system, the checksum in the command secondary header is computed and compared to the checksum attached to the command. If the checksum fails, the command is discarded, the error is reported through an event message, and the RTS is stopped.

3.3.6 RTP Loads

3.3.6.1 Loading RTSs

In order to run an RTS on the RTP, the sequence must be loaded into an RTS buffer. The RTS can be loaded into the buffers by two methods. The first method is to load the RAM buffer. The second method is to load the EEPROM version of the RTS buffer.

Because of the relatively small size of the RTS buffers, there is no method to patch RTSs.

3.3.6.1.1 *Loading RTP Sequence to RAM buffers*

Loading an RTS to the RAM buffer is a simple task.

Although an RTS may be loaded in any of the RTS buffers, it is recommended that the RTS be loaded in one of the buffers not reserved for a pre-loaded RTS. An RTS is formatted and loaded the same way an ATS is loaded.

After the load is formatted and ready to transmit to the Flight Software, load the cFE Working Buffer with the desired RTS. The RTS commands should start at the first byte of the buffer and there should be no gaps between RTS commands. Once the Working Buffer has been loaded, dumped, and verified, SC will first check to see if the specified RTS buffer is in use. If the RTS buffer is not in use, then the RTS data is copied from the Working Buffer to the RTS buffer. The buffer is now ready to use. The RTS needs to be enabled prior to executing (starting).

Figure 8 is an example RTS load. The first part of the figure is the list of command mnemonics and the second part is the actual load data. Note the differences between the example ATS and the example RTS.

Relative Time Sequence Mnemonics for an RTS

Delay	*Command*	**Comment**
10	*SNOOPCMD*	No operation
5	*S4KTLM*	Start 4K telemetry
10	*SNOOPCMD*	No operation
5	*STLMOFF*	Turn off telemetry
2	*SNC86RTS*	End the RTS sequence
See numbers in bold in table below	*See highlighted numbers in table below*	

For RTS

Binary Command Data in ASCII HEX format

(Note: cFE file header and cFE table header not shown)

First➔ Byte

```
000A 1801 C000 0001 0001 0005 1803 C000 0001 0002
000A 1801 C000 0001 0001 0005 1803 C000 0001 0001
 0002 180A C000 0001 000B 0000 0000 0000 0000 0000
 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
                      .........
 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Last ➔Byte

Note: The mnemonics and the corresponding hex values of the commands in this example are for illustration only, and do not represent actual commands for any specific mission. See the command database for actual command mnemonics and their hex equivalents.

Figure 8 Sample RTS Load

The second method of loading Relative Time Sequences is actually performed when the Flight Software is installed in the Data System. All of the Limit Checker Sequences and some of the more commonly used sequences will be loaded into the system for instant access:

- The RTS buffers that are used for the pre-loaded RTSs can be overwritten.

- The pre-loaded RTS data is stored in EEPROM or in a file on board.
- Upon startup, the RTS data is transferred to RAM.
- When the system is running, a request to start one of the pre-loaded RTSs will run the RTS from the RAM buffer.
- Ground controllers can load a new RTS into the RAM buffer where a pre-loaded RTS is located. Now every call to that RTS buffer number will use the new RTS.
- The buffer can be reset back to the original EEPROM version by a system reset (not recommended) or by using cFE Table Manager.

3.3.6.2 RTS Processor Commands

The RTS Processor can be controlled by requests by the ground controller or from flight software applications LC or HS.

The RTS Processor can be commanded to:

- Start an RTS
- Stop an RTS
- Enable an RTS
- Disable an RTS

3.4 *ATS/RTS Loads – Helpful Information*

3.4.1 Updating ATS/RTS Loads

When creating and loading a new ATS or RTS table, it is recommended that the table to be updated is not executing. While updating an executing ATS or RTS is allowed, unexpected results may arise since the contents of the updated ATS or RTS will be executed after the table is activated.

3.4.2 Null ATS/RTS Loads (Not Valid)

Loading a NULL (zero filled file) RTS or ATS is not a valid SC operation.

In the event that the user executes a NULL ATS or RTS, an SC event message will be sent out and the following happens:

ATS: The ATS and associated index and status tables WILL be cleared.

RTS: The RTS that was being overwritten will NOT be cleared. *[Note: From an FSSE perspective about SC, this means that any previously used RTS would have to be replaced with something like a noop command, rather than just removing the RTS. This also means that the recovery procedure should include reloading such noop commands even if no RTS would otherwise be loaded so the system configuration is completely restored.]*

3.4.3 Table and File Naming Convention

SC uses two types of loadable tables: ATS and RTS tables. Both the table file names and the table names contained in the file header need to follow a specific naming convention.

3.4.4 Naming Conventions for ATSs

What matters to SC is the *name of the table itself* in the *header of the table file*.

The name should either be 'SC.ATS_TBL1' or 'SC.ATS_TBL2' provided that the name of the application is "SC" and which ATS the table is meant for (1 is for A, 2 is for B).

ATSs are not loaded at startup, nor are they loaded automatically at any other time. Therefore, neither the name of the file of the ATS, nor the file's location, are important.

3.4.5 Naming Conventions for RTSs

Because RTSs can be loaded at startup, the files for those RTSs must be in a predetermined location (CFS SC Configuration Parameter SC_RTS_FILE_NAME).

This location must be in non-volatile memory. Otherwise, the files would not exist upon a Power-On reset.

Also, the RTS table file must be named according to a specific convention (CFS SC Configuration Parameter SC_RTS_TABLE_NAME). The file name must start with the value of the (CFS SC Configuration Parameter SC_RTS_TABLE_NAME) platform configuration parameter.

Next, must be a three digit number indicating which RTS this table file is, and the last must be ".tbl". An example of this for RTS No.1, with SC_RTS_TABLE_NAME set to "RTS_TBL" would be: 'RTS_TBL001.tbl'.

In addition to the file naming convention, the name of the table contained within the table file should be the same as the file name, without the path or extension.

Remember to also have the application name prefixed to the name of the table. For the file 'RTS_TBL001.tbl', its table name should be 'SC.RTS_TBL001, if the name of the application is "SC".

3.5 *Housekeeping Telemetry Tasks*

3.5.1 Housekeeping Variables Affected By Reset

For testing and on orbit flight operations, it is important to be able to start with a clean set of housekeeping variables. Upon receipt of a Reset command, SC resets the following housekeeping variables to a value of zero:

1. Valid Command Counter
2. Command Rejected Counter
3. ATS Command Counter
4. ATS Command Error Counter
5. RTS Command Counter
6. RTS Command Error Counter
7. Number of RTS Started Counter
8. Number of RTS Started Error Counter

3.5.2 Housekeeping Telemetry to Indicate Basic SC Status.

After resetting housekeeping variables upon a Reset command, SC generates a housekeeping message containing the following:

1. Valid Command Counter

2. Command Rejected Counter
3. Total count of commands dispatched from ATSs
4. Total count of commands dispatched from RTSs
5. Total count of commands which failed dispatch from ATSs
6. Total count of commands which failed dispatch from RTSs
7. ATS Table No.1 free byte count
8. ATS Table No.2 free byte count
9. Absolute Time Command Processing State
10. Identifier of the active ATS table
11. Number of the next ATS command pending execution
12. ATS switch pending flag
13. Time the next ATS command is due to be dispatched
14. The identifier of the ATS table for which the most recent ATS command failed to dispatch
15. The identifier of most recent ATS command which failed to dispatch from the ATS tables
16. RTS table activation count
17. RTS table activation error count
18. Number of active RTSs
19. Identifier of the next RTS table to dispatch a command
20. Time the next RTS command is due to be dispatched
21. Execution status for each RTS table
22. Enable status for each RTS table
23. Identifier of the RTS table for which the most recent RTS command dispatch error occurred
24. The word offset within the RTS for the most recent RTS command which failed to dispatch
25. ATS Continue-On-Failure status
26. The last append ApId
27. The last ATS Append Table command count
28. The last appended count

Chapter 4. **Understanding SC Operational Constraints**

4.1 ***Effects of Adjusting Spacecraft Time***

SC does not react well to adjustments to spacecraft time. It is recommended that SC be idle during large time adjustments (1 second or greater). The application does not need to be suspended, but SC should not have any commands to process during large time adjustments. Simply ensure there no RTS or ATSs enabled.

Small adjustments can be tolerated if there is not an exact second tolerance for every command being executed.

Because SC application depends on the spacecraft clock to schedule itself and all of the active sequences, adjustments to spacecraft clock time (depending on what time format is configured) will have an effect on those active sequences. When required, adjusting the spacecraft clock time has different effects on active ATSs and RTSs.

4.1.1 Effect on ATSs

The effect that adjusting time has on an ATS is predictable.

Because each command has an absolute time tag associated with it, any sequence that is running will have a new frame of reference when the time is adjusted.

For example, an ATS is started at 12:00 and it is waiting to send out commands starting at 2:00 pm. If SC is configured to use UTC time, and the ATS is started and UTC time is adjusted to 1:00 pm (by adjusting spacecraft time or the Universal time code format [UTCF] or both), the first ATS command will execute in an hour instead of two hours. If SC is configured to use TAI time, and the ATS is started, and the leap seconds are changed, SC time will not be affected. (TAI time does not include leap seconds, so changing the leap seconds does not change the calculation of time for TAI.)

4.1.2 Effect on RTSs

The effect of adjusting time on Relative Time Sequences is a little more complicated.

As mentioned in the Scheduling section, the next RTS command for each sequence actually has an absolute time associated with it. With this absolute time, it is possible for SC to know when to send out each RTS command. SC starts with the absolute time, then computes a delayed time, which is used to tell SC when to send out each RTS command.

When time is adjusted on an RTS, the command that is currently waiting to execute gets "thrown off". Depending on the time adjustment, the command could go out sooner or later than expected. Once the "current" command is out, however, the remaining commands in the sequence will execute as scheduled because they are relative to the previous command.

4.2 *Avoid Over Scheduling the Stored Command Application*

Another way to "stress" SC is to send out many commands in one second from many sequences. The time tags for both ATS commands and RTS commands have one second resolution. However, there is a way to send multiple commands over the period of one second:

- For ATS commands, set the time tags to the same second.
- For RTS commands, set the delays to zero.
- As noted earlier, SC will send out the commands as fast as possible up to a certain number of commands per second.

With this in mind, it is possible to pack the ATSs and a few RTSs with commands that need to go out in the same second. When all of these sequences are run, SC will get behind in sending out the commands. SC will keep going until all of the commands are executed, but do not expect "to the second" execution accuracy. "Keep going" means that ATS commands in excess of the MAX in the same second would not get sent out (they would be skipped), but RTS commands will get executed as much as possible (and possibly get pushed off again) until SC has caught up.

4.3 *Avoid Unsorted ATS Loads*

Avoid maximum unsorted loads. Although there is no requirement for the ATS command loads to be sorted in time order, having a full command load (platform defined ATS commands) of completely unsorted commands can cause SC to block the execution of lower priority tasks until it is finished sorting.

Chapter 5. **Operations FAQs**

5.1 *Why is the smallest time unit for SC one second?*

A time granularity of one second is sufficient for fault management responses and the one second period virtually guarantees the sequential execution of commands.

5.2 *How can ground controllers make sure the SC software is alive?*

Use the debug No-Op command to verify the application is alive. Upon receipt of a No-Op command, SC increments the SC Valid Command Counter and generates an event message. Note that this only applies to ground controller commands, i.e. does not include requests that come from the scheduler.

5.3 *How are ATS commands handled by SC?*

SC accepts a variable number of variable length commands from each ATS. Each ATS command contains:

1. A command number
2. A time tag denoting the time at which to execute the command
3. A variable length command

For any ATS command which fails the Data Integrity Check Value, SC performs the following:

1. Discards the command
2. Marks the command with "Data Integrity Check Value Verification Failed"
3. Issues an event message

5.4 *What happens when SC validates a command?*

When SC accepts a command as valid, SC executes the command, increments the SC Valid Command Counter and issues an event message. If SC rejects any command, SC aborts execution of the command, increments the SC Command Rejected Counter, and issues an error event message.

5.5 *As a ground controller, how do I know if command strings are corrupted?*

For all SC commands, if the length contained in the message header is not equal to the expected length, SC rejects the command and issues an event message. Note again that this only applies to "ground commands" (i.e. does not include requests that come from the Scheduler application.)

5.6 *Can the ground controller chain ATS tables together?*

Yes. Upon receipt of a Switch ATS Command, SC does the following:

1. Terminates the processing of the current ATS table after processing all of the commands within the current second.
2. Starts processing the alternate ATS table.

5.7 *Does the next table start immediately?*

No. SC allows ATS tables to overlap each other so that the ground controller can switch at any time, but SC waits until the next second to ensure that the overlapping in ATSs doesn't cause the same commands to be executed twice.

In other words, SC begins processing the first ATS command after the next one second occurs containing a time which is greater-than-or-equal-to the current time.

5.8 *Are switch commands handled identically from the ground as in an ATS?*

No. A switch command located within an ATS is handled differently than a switch command by the ground controller. If the switch command is by the ground controller, SC waits until the next second to prevent overlapping of two ATSs from duplicating commands.

If the Switch command is located within an ATS, SC immediately executes the switch command.

5.9 *Does SC finish executing the current commands in the time slot before jumping?*

Upon receipt of a Jump Command, SC transfers execution to the command within the currently executing ATS table whose time-tag is equal to a command-specified time value.

5.10 *What happens if the ground controller specifies a jump time, but there's no command set for that time?*

If no command exists that is equal to the command-specified jump time, SC waits for the first command after the jump time.

5.11 *Does the difference in leap seconds in International Atomic Time (TAI) affect an RTS?*

TAI does not use leap seconds at all, so they would not be used in determining the time for an RTS. The time format (TAI, UTC) only affects ATSs, not RTSs, since RTSs only contain offsets from the time they were started.

5.12 *What happens if there are more than the maximum number of commands to be executed? Will they be processed in the next second or skipped?*

SC tries to catch up during the next second and subsequent seconds. ATSs gets priority, then RTSs. RTSs get priority in order of their numbers (for example, 1 gets priority over 5). If, for example, the limit was eight commands per second, SC would try to send eight commands per second until it was all caught up.

5.13 *Can an RTS start itself?*

The answer is yes. Since the RTS is just a collection of commands, an RTS can include the StartRTS command. However, it is recommended that this be the last command in the RTS.

This page intentionally blank

Appendix A CFS SC Reference

A.1 ATS and RTS Dump-Only Tables

The fields included in the ATP Control Block are shown below.

Table 7 ATP Control Block Table

Name	Description
AtpState	This field has the current state of the Absolute Time Processor. It can be IDLE or EXECUTING.
AtsNumber	This field has the number of the current or last executed ATS. The possible values are: 0 for none (after power on or cold restart), 1 for ATS A, and 2 for ATS B.
CmdNumber	This field has the current or last executed ATS command number.
TimeIndexPtr	This field has the table offset into the ATS Time Index Table of the next ATS command to execute.
SwitchPendFlag	This flag indicates that an ATS switch is pending. This flag is used for internal logic.

The fields included in the ATS Information Table are shown below.

Table 8 ATS Information Table

Name	Description
Usage Counter	A counter that increments each time the ATS has executed to completion
Command Count	The number of commands contained in the ATS
Size	The size of the ATS.

The fields of each entry in the RTS Information Table are shown below.

Table 9 RTS Information Table

Name	Description
Status flag	The status of the associated RTS. This can be EMPTY, LOADED, or EXECUTING.
Disabled flag	Indicates whether the associated RTS is Enabled or Disabled. If Disabled, the RTS cannot be started.
Command Counter	The number of commands executed in this execution of the RTS.

Name	Description
Command Error Counter	The number of errors that have occurred during this RTS execution. Currently, an RTS will stop when it encounters an error, so the counter should never increment past 1.
Next Command Time	The computed absolute time the next command will execute.
Next Command Pointer	An offset (in 16-bit words) into the RTS Table of the next command.
Usage Counter	The number of times that the associated RTS has been started since the last time it was loaded.

The fields included in the RTP Control Block Table are shown below.

Table 10 RTP Control Block Table

Name	Description
Active RTSs	The number of RTSs currently active (started).
RtsNumber	The current RTS number that is executing on the RTP. Note that this field is only cleared out when another RTS is started, allowing the ground to note the last RTS that ran on the RTP.

A.2 Event Messages

Event messages are mission independent.

A.2.1 Event Messages - ERROR

The tables in this section show **error** event messages for the SC application.

Table 11 Event ID 1 (Error)

Event ID Number:	1
Event Message:	'App terminating, Result = 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when the App exits due to a fatal error condition.
	• The Result field contains the return status from the function that caused the app to terminate.

Table 12 Event ID 2 (Error)

Event ID Number:	2
Event Message:	'Invalid msg length: ID = 0x%04X, CC = %d, Len = %d, Expected = %d'
Type:	ERROR
Cause:	This event message is issued when a command is received, but it is not of the expected length.
	• The **ID** field contains the message Id of the message.
	• The **CC** field contains the command code of the message.
	• The **Len** field contains the length of the message received.
	• The **Expected** field contains the expected length of the command .

Table 13 Event ID 3 (Error)

Event ID Number:	3
Event Message:	'Software Bus Create Pipe returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when cFE_SB_CreatePipe returns an error.
	• The **returned** field contains the result of the cFE_SB_CreatePipe call.

Table 14 Event ID 4 (Error)

Event ID Number:	4
Event Message:	'Software Bus subscribe to housekeeping returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when CFE_SB_Subscribe to the Housekeeping Request packet fails.
	• The **returned** field contains the result of the CFE_SB_Subscribe call.

Table 15 Event ID 5 (Error)

Event ID Number:	5
Event Message:	'Software Bus subscribe to 1 Hz cycle returned: 0x%08X'
Type:	ERROR

Event ID Number:	5
Cause:	- This event message is issued when CFE_SB_Subscribe to the 1 Hz Request packet fails.
	- The **`returned`** field contains the result of the CFE_SB_Subscribe call.

Table 16 Event ID 6 (Error)

Event ID Number:	6
Event Message:	'Software Bus subscribe to command returned: 0x%08X'
Type:	ERROR
Cause:	- This event message is issued when CFE_SB_Subscribe to the SC Command Request packet fails
	- The **`returned`** field contains the result of the CFE_SB_Subscribe call.

Table 17 Event ID 7 (Error)

Event ID Number:	7
Event Message:	'Dump Only Tables initialization failed, returned: 0x%08X'
Type:	ERROR
Cause:	- This event message is issued when there is a problem registering the dump only tables that SC has to use.
	- The **`returned`** field contains the result returned from SC_RegisterDumpOnlyTables.

Table 18 Event ID 8 (Error)

Event ID Number:	8
Event Message:	'Table initialization failed, returned: 0x%08X'
Type:	ERROR
Cause:	- This event message is issued when there was a problem initializing SC's tables.
	- The **`returned`** field contains the result from either SC_InitWithCDS or SC_InitNoCDS, depending on if the CDS is being used or not for SC.

Table 19 Event ID 10 (Error)

Event ID Number:	10
Event Message:	'RTS Table Registration Failed for RTS %d, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when a call to CFE_TBL_Register for an RTS table failed when SC is not configured to use the CDS.
	• The **RTS** field contains the RTS ID of the table that failed. • The **returned** field contains the error code returned from CFE_TBL_Register.

Table 20 Event ID 11 (Error)

Event ID Number:	11
Event Message:	'ATS Table Registration Failed for ATS %d, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when a call to CFE_TBL_Register for an ATS table failed when SC is not configured to use the CDS.
	• The **ATS** field contains the ATS ID of the table that failed. • The **returned** field contains the error code returned from CFE_TBL_Register.

Table 21 Event ID 12 (Error)

Event ID Number:	12
Event Message:	'Registering CDS for dump only table data failed, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when trying to register a CDS for the dump only table data failed.
	• The **returned** field contains the result from CFE_ES_RegisterCDS.

Table 22 Event ID 13 (Error)

Event ID Number:	13
Event Message:	'Registering CDS for SC_AppData failed, returned: 0x%08X'
Type:	ERROR

Event ID Number:	13
Cause:	• This event message is issued when trying to register a CDS for the SC Application Data failed.
	• The **returned** field contains the result from CFE_ES_RegisterCDS.

Table 23 Event ID 14 (Error)

Event ID Number:	14
Event Message:	'Error registering RTS %d table, Result = 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when registering an RTS table while using the CDS failed
	• The **RTS** field contains the RTS Id of the table that failed to be registered.
	• The **Result** field contains the result of the CFE_TBL_Register call.

Table 24 Event ID 15 (Error)

Event ID Number:	15
Event Message:	'Error registering ATS %d table, Result = 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when registering an ATS table while using the CDS failed.
	• The **ATS** field contains the ATS Id of the table that failed to be registered. • The **Result** field contains the result of the CFE_TBL_Register call.

Table 25 Event ID 16 (Error)

Event ID Number:	16
Event Message:	'RTS info table register failed, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when trying to register the RTS Info Table dump only table fails.
	• The **returned** field contains the result from CFE_TBL_Register.

Table 26 Event ID 17 (Error)

Event ID Number:	17
Event Message:	'RTS control block table register failed, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when trying to register the RTS control block dump only table fails.
	• The **`returned`** field contains the result from CFE_TBL_Register.

Table 27 Event ID 18 (Error)

Event ID Number:	18
Event Message:	'ATS info table register failed, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when trying to register the ATS Info Table dump only table fails.
	• The **`returned`** field contains the result from CFE_TBL_Register.

Table 28 Event ID 19 (Error)

Event ID Number:	19
Event Message:	'ATS control block table register failed, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when trying to register the ATS control block dump only table fails.
	• The **`returned`** field contains the result from CFE_TBL_Register.

Table 29 Event ID 20 (Error)

Event ID Number:	20
Event Message:	'ATS command status table register failed for ATS %d, returned: 0x%08X'
Type:	ERROR

Event ID Number:	20
Cause:	• This event message is issued when one of the ATS command status tables fails table registration.
	• The **ATS** field contains the ATS Id for the table that failed. • The **returned** field contains the result from CFE_TBL_Register.

Table 30 Event ID 24 (Error)

Event ID Number:	24
Event Message:	'Start ATS Rejected: ATS %c Not Loaded'
Type:	ERROR
Cause:	• This event message is issued when a SC_START_ATS_CC cmd failed because the specified ATS was not loaded.
	• The **ATS** field contains the ATS that was tried to be started.

Table 31 Event ID 25 (Error)

Event ID Number:	25
Event Message:	'Start ATS Rejected: ATP is not Idle'
Type:	ERROR
Cause:	This event message is issued when a SC_START_ATS_CC command was issued but there is already an ATS running.

Table 32 Event ID 26 (Error)

Event ID Number:	26
Event Message:	'Start ATS %d Rejected: Invalid ATS ID'
Type:	ERROR
Cause:	• This event message is issued when the ATS Id specified in the SC_START_ATS_CC command was invalid.
	• The **ATS** field contains the invalid ATS Id.

Table 33 Event ID 29 (Error)

Event ID Number:	29
Event Message:	'All ATS commands were skipped, ATS stopped'
Type:	ERROR
Cause:	This event message is issued when an ATS is begun, and all times for the commands in the ATS exist in the past.

Table 34 Event ID 32 (Error)

Event ID Number:	32
Event Message:	'Switch ATS Failure: Destination ATS Not Loaded'
Type:	ERROR
Cause:	This event message is issued when a SC_SWITCH_ATS_CC command is issued, but the ATS to switch to is not loaded.

Table 35 Event ID 33 (Error)

Event ID Number:	33
Event Message:	'Switch ATS Rejected: ATP is idle'
Type:	ERROR
Cause:	This event message is issued when a SC_SWITCH_ATS_CC command is issued, but there is not an ATS running from which to switch.

Table 36 Event ID 35 (Error)

Event ID Number:	35
Event Message:	'Switch ATS Failure: Destination ATS is empty'
Type:	ERROR
Cause:	This event message is issued when an ATS switch is scheduled, but there are no commands (ie not loaded) in the destination ATS.

Table 37 Event ID 36 (Error)

Event ID Number:	36
Event Message:	'Switch ATS Rejected: ATP is idle'
Type:	ERROR
Cause:	This event message is issued when an ATS switch is scheduled, but there is no ATS running This error will only occur is something gets corrupted.

Table 38 Event ID 38 (Error)

Event ID Number:	38
Event Message:	'Switch ATS Failure: Destination ATS Not Loaded'
Type:	ERROR
Cause:	This event message is issued when an ATS switch is scheduled, but there are no commands (ie not loaded) in the destination ATS.

Table 39 Event ID 39 (Error)

Event ID Number:	39
Event Message:	'Jump Cmd: All ATS commands were skipped, ATS stopped'
Type:	ERROR
Cause:	This event message is issued when a SC_JUMP_ATS_CC command was issued, and the time to jump to was passed all of the commands in the ATS.

Table 40 Event ID 41 (Error)

Event ID Number:	41
Event Message:	'ATS Jump Failed: No active ATS'
Type:	• ERROR
Cause:	• This event message is issued when a SC_JUMP_ATS_CC command was received, but there is no ATS currently running.

Table 41 Event ID 42 (Error)

Event ID Number:	42
Event Message:	‘Continue ATS On Failure command failed, invalid state: %d"‘
Type:	ERROR
Cause:	• This event message is issued when a SC_CONTINUE_ATS_ON_FAILURE_CC command was received, but the state in the command was invalid.
	• The **State** field contains the state given in the command.

Table 42 Event ID 44 (Error)

Event ID Number:	44
Event Message:	‘ATS Command Failed Checksum: Command #%d Skipped’
Type:	ERROR
Cause:	• This event message is issued when a command from an ATS is about to be sent out but it fails checksum validation.
	• The **Command** field contains the number of the command that was skipped.

Table 43 Event ID 45 (Error)

Event ID Number:	45
Event Message:	‘ATS %c Aborted’
Type:	ERROR
Cause:	• This event message is issued when a ATS command that was about to be sent out failed checksum validation, and the Continue-ATS-on_checksum-Failure flag was set to 'FALSE’.
	• The **Sequence** field contains the ATS that was stopped.

Table 44 Event ID 46 (Error)

Event ID Number:	46
Event Message:	‘ATS Command Distribution Failed, Cmd Number: %d, SB returned: 0x%08X’
Type:	ERROR

Event ID Number:	46
Cause:	• This event message is issued when an ATS command is about to be sent out, and the CFE_SB_SendMsg call failed to send it.
	• The **`Cmd number`** field contains the command number that failed. • The **`Returned`** field contains the return code from CFE_SB_SendMsg.

Table 45 Event ID 47 (Error)

Event ID Number:	47
Event Message:	'ATS Command Number Mismatch: Command Skipped, expected: %d received: %d'
Type:	ERROR
Cause:	• This event message is issued when an ATS command is about to be sent out, but its command number is not what was expected.
	• The **`expected`** field is the command number that was in the ATS control block. • The **`received`** field is the command number that was in the ATS table.

Table 46 Event ID 48 (Error)

Event ID Number:	48
Event Message:	'Invalid ATS Command Status: Command Skipped'
Type:	ERROR
Cause:	• This event message is issued when an ATS command is about to be send out, but the command isn't marked as 'SC_LOADED'.
	• The **`Status`** field contains the state of the ATS command.

Table 47 Event ID 49 (Error)

Event ID Number:	49
Event Message:	'RTS %03d Command Distribution Failed: RTS Stopped. SB returned 0x08X'
Type:	ERROR

Event ID Number:	49
Cause:	• This event message is issued when an RTS command was about to be sent out, and CFE_SB_SendMsg couldn't send the message.
	• The **RTS** field contains the RTS Id that was stopped. • The **returned** field is the return code from CFE_SB_SendMsg.

Table 48 Event ID 50 (Error)

Event ID Number:	50
Event Message:	‘RTS %03d Command Failed Checksum: RTS Stopped’
Type:	ERROR
Cause:	• This event message is issued when an RTS comand was about to be sent out, but the command failed checksum validation.
	• The **RTS** field contains the RTS Id that was stopped.

Table 49 Event ID 59 (Error)

Event ID Number:	59
Event Message:	‘RTS cmd loaded with invalid MID at %d’
Type:	ERROR
Cause:	• This event message is issued when and RTS table is loaded, but there is an invalid message Id in the command.
	• The **Start** field contains the word at which the failure occurred.

Table 50 Event ID 60 (Error)

Event ID Number:	60
Event Message:	‘RTS cmd loaded with invalid length at %d, len: %d’
Type:	ERROR
Cause:	• This event message is issued when and RTS table is loaded, but there is an invalid length in the command.
	• The **Start** field contains the word at which the failure occurred. • The **len** field contains the length in the command.

Table 51 Event ID 61 (Error)

Event ID Number:	61
Event Message:	'RTS cmd at %d runs off end of buffer'
Type:	ERROR
Cause:	• This event message is issued when and RTS table is loaded, but the command runs off the end of the buffer.
	• The **`Start`** field contains the word at which the failure occurred.

Table 52 Event ID 62 (Error)

Event ID Number:	62
Event Message:	'RTS cmd loaded won't fit in buffer at %d'
Type:	ERROR
Cause:	• This event message is issued when and RTS table is loaded, but the command won't fit in the buffer.
	• The **`Start`** field contains the word at which the failure occurred.

Table 53 Event ID 63 (Error)

Event ID Number:	63
Event Message:	'Invalid command pipe message ID: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when an invalid message Id is received in the command pipe.
	• The **`Messge Id`** field contains the erroneous message Id.

Table 54 Event ID 64 (Error)

Event ID Number:	64
Event Message:	'Invalid Command Code: MID = 0x%04X CC = %d'
Type:	ERROR

Event ID Number:	64
Cause:	• This event message is issued when an invalid command code was received in the command pipe.
	• The **`MID`** field contains the message Id for the command. • The **`CC`** field contains the erroneous command code.

Table 55 Event ID 65 (Error)

Event ID Number:	65
Event Message:	'RTS Info table failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when getting the address of the Rts Info table failed.
	• The **`returned`** field contains the result from CFE_TBL_GetAddress.

Table 56 Event ID 66 (Error)

Event ID Number:	66
Event Message:	'RTS Ctrl Blck table failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when getting the address of the Rts Control Block table failed.
	• The **`returned`** field contains the result from CFE_TBL_GetAddress.

Table 57 Event ID 67 (Error)

Event ID Number:	67
Event Message:	'ATS Info table failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when getting the address of the Ats Info table failed.
	• The **`returned`** field contains the result from CFE_TBL_GetAddress.

Table 58 Event ID 68 (Error)

Event ID Number:	68
Event Message:	'ATS Ctrl Blck table failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when getting the address of the Ats Control Block table failed.
	• The **returned** field contains the result from CFE_TBL_GetAddress.

Table 59 Event ID 69 (Error)

Event ID Number:	69
Event Message:	'ATS Cmd Status table for ATS %d failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when getting the address of an Ats Command Status table failed.
	• The **ATS** field contains the ATS Id of the Cmd Status table that failed. • The **returned** field contains the result from CFE_TBL_GetAddress.

Table 60 Event ID 70 (Error)

Event ID Number:	70
Event Message:	'RTS table %d failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when getting the address of an RTS table failed.
	• The **RTS** field contains the RTS Id of the table that failed. • The **returned** field contains the result from CFE_TBL_GetAddress.

Table 61 Event ID 71 (Error)

Event ID Number:	71
Event Message:	'ATS table %d failed Getting Address, returned: 0x%08X'
Type:	ERROR

Event ID Number:	71
Cause:	• This event message is issued when getting the address of an ATS table failed.
	• The **RTS** field contains the RTS Id of the table that failed. • The **returned** field contains the result from CFE_TBL_GetAddress.

Table 62 Event ID 74 (Error)

Event ID Number:	74
Event Message:	'Start RTS %03d Rejected: Invld Len Field for 1st Cmd in Sequence. Invld Cmd Length = %d'
Type:	ERROR
Cause:	• This event message is issued when an RTS is started but the first command has an invalid length.
	• The **Number** field contains the RTS Id that was started.

Table 63 Event ID 75 (Error)

Event ID Number:	75
Event Message:	'Start RTS %03d Rejected: RTS Not Loaded or In Use, Status: %d'
Type:	ERROR
Cause:	• This event message is issued when an RTS is tried to be started, but the RTS is not marked as SC_LOADED.
	• The **RTS** field contains the RTS ID that was rejected. • The **Status** field contains the status of that RTS.

Table 64 Event ID 76 (Error)

Event ID Number:	76
Event Message:	'Start RTS %03d Rejected: RTS Disabled'
Type:	ERROR
Cause:	• This event message is issued when a SC_START_RTS_CC command was received, but the RTS is disabled.
	• The **RTS** field contains the RTS Id of the RTS that was rejected to start.

Table 65 Event ID 77 (Error)

Event ID Number:	77
Event Message:	'Start RTS %03d Rejected: Invalid RTS ID'
Type:	ERROR
Cause:	• This event message is issued when a SC_START_RTS_CC command was received, but the RTS Id was invalid.
	• The **RTS** field contains the RTS Id of the RTS that was rejected to start.

Table 66 Event ID 79 (Error)

Event ID Number:	79
Event Message:	'Stop RTS %03d rejected: Invalid RTS ID'
Type:	ERROR
Cause:	• This event message is issued when a SC_STOP_RTS_CC command was rejected because the RTS Id given was invalid.
	• The **RTS** field contains the RTS Id of the RTS that was invalid.

Table 67 Event ID 81 (Error)

Event ID Number:	81
Event Message:	'Disable RTS %03d Rejected: Invalid RTS ID'
Type:	ERROR
Cause:	• This event message is issued when a SC_DISABLE_RTS_CC command was received, but the RTS Id given was invalid.
	• The **RTS** field contains the RTS Id of the RTS that was invalid.

Table 68 Event ID 83 (Error)

Event ID Number:	83
Event Message:	'Enable RTS %03d Rejected: Invalid RTS ID'
Type:	ERROR

Event ID Number:	83
Cause:	• This event message is issued when a SC_DISABLE_RTS_CC command was received, but the RTS Id given was invalid.
	• The **RTS** field contains the RTS Id of the RTS that was invalid.

Table 69 Event ID 84 (Error)

Event ID Number:	84
Event Message:	'Cmd Runs passed end of Buffer, RTS %03d Aborted'
Type:	ERROR
Cause:	• This event message is issued when trying to get the next RTS command to execute, and that command runs passed the end of the RTS table buffer.
	• The **RTS** field contains the RTS Id that failed.

Table 70 Event ID 85 (Error)

Event ID Number:	85
Event Message:	'Invalid Length Field in RTS Command, RTS %03d Aborted. Length: %d, Max: %d'
Type:	ERROR
Cause:	• This event message is issued when trying to get the next RTS command to execute, and that command has an illegal length value in the command.
	• The **RTS** field contains the RTS Id that failed. • The **Length** field is the length of the command. • The **Max** field is SC_PACKET_MAX_SIZE.

Table 71 Event ID 89 (Error)

Event ID Number:	89
Event Message:	'Saving AppData-CDS Flag back to CDS Failed, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when CDS Data has been restored to SC, and the Saved-To CDS flag has been reset, but could not be copied back to the CDS correctly.
	• The **returned** field contains return code of CFE_ES_CopyToCDS.

Table 72 Event ID 90 (Error)

Event ID Number:	90
Event Message:	'Append ATS info table register failed, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when trying to register the Append ATS Info Table dump only table fails.
	• The **`returned`** field contains the result from CFE_TBL_Register.

Table 73 Event ID 91 (Error)

Event ID Number:	91
Event Message:	'Append ATS Info table failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message is issued when getting the address of the Append ATS Info table failed.
	• The **`returned`** field contains the result from CFE_TBL_GetAddress.

Table 74 Event ID 92 (Error)

Event ID Number:	92
Event Message:	'Append ATS table failed Getting Address, returned: 0x%08X'
Type:	ERROR
Cause:	• This event message indicates a failure to get the table data address from cFE Table Services.
	• The **`returned`** value is the result from the call to CFE_TBL_GetAddress.

Table 75 Event ID 93 (Error)

Event ID Number:	93
Event Message:	'Append ATS Table Registration Failed, returned: 0x%08X'
Type:	ERROR

Event ID Number:	93
Cause:	• This event message indicates a failure to register the table with cFE Table Services.
	• The **returned** field is the result from the call to CFE_TBL_Register.

Table 76 Event ID 94 (Error)

Event ID Number:	94
Event Message:	'Error registering Append ATS table, Result = 0x%08X'
Type:	ERROR
Cause:	• This event message indicates a failure to register the table with cFE Table Services.
	• The **returned** value is the result from the call to CFE_TBL_Register.

Table 77 Event ID 99 (Error)

Event ID Number:	99
Event Message:	'Append ATS error: invalid ATS ID = %d'
Type:	ERROR
Cause:	• This event message is issued when a SC_APPEND_ATS_CC command has failed because the specified ATS command argument was not a valid ATS ID.
	• The event text displays the invalid ATS ID command argument.

Table 78 Event ID 100 (Error)

Event ID Number:	100
Event Message:	'Append ATS %c error: ATS table is empty'
Type:	ERROR
Cause:	• This event message is issued when a SC_APPEND_ATS_CC command has failed because the specified target ATS table is empty. The Append ATS command requires that both the source Append table and the target ATS table have valid contents.
	• The event text displays the ATS ID command argument (A or B).

Table 79 Event ID 101 (Error)

Event ID Number:	101
Event Message:	'Append ATS %c error: Append table is empty'
Type:	ERROR
Cause:	• This event message is issued when a SC_APPEND_ATS_CC command has failed because the source Append table is empty. The Append ATS command requires that both the source Append table and the target ATS table have valid contents.
	• The event text displays the ATS ID command argument (A or B).

Table 80 Event ID 102 (Error)

Event ID Number:	102
Event Message:	'Append ATS %c error: ATS size = %d, Append size = %d, ATS buffer = %d'
Type:	ERROR
Cause:	• This event message is issued when a SC_APPEND_ATS_CC command has failed because there is not room in the specified target ATS table to add the contents of the Append table.
	• The event text displays the ATS ID command argument (A or B), the size (in words) of the data in the specified ATS table, the size of the data in the Append table and the total size of the ATS table.

Table 81 Event ID 104 (Error)

Event ID Number:	104
Event Message:	'Verify ATS Table error: invalid command number: buf index = %d, cmd num = %d'
Type:	ERROR
Cause:	• This event message indicates an error during verification of an ATS or an Append ATS table due to an invalid table entry. The cause is a table entry with an invalid command number.
	• The event text describes the offset into the table in words for the invalid ATS table entry and the invalid table entry command number.

Table 82 Event ID 105 (Error)

Event ID Number:	105
Event Message:	'Verify ATS Table error: buffer full: buf index = %d, cmd num = %d, buf words = %d'

Event ID Number:	105
Type:	ERROR
Cause:	• This event message indicates an error during verification of an ATS or an Append ATS table due to an invalid table entry. The cause is a table entry that has a valid command number but the entry begins at an offset too near the end of the table buffer to provide room for even the smallest possible command packet.
	• This error can be corrected by setting the first data word which follows the last valid table entry to zero. Note that the first word in an ATS table entry is the entry command number.
	• The event text describes the offset into the table in words for the invalid ATS table entry, the valid entry command number and the total size of the table buffer in words.

Table 83 Event ID 106 (Error)

Event ID Number:	106
Event Message:	'Verify ATS Table error: invalid length: buf index = %d, cmd num = %d, pkt len = %d'
Type:	ERROR
Cause:	• This event message indicates an error during verification of an ATS or an Append ATS table due to an invalid table entry. The cause is a table entry with an invalid command packet length.
	• The event text describes the offset into the table in words for the invalid ATS table entry, the valid table entry command number and the invalid table entry command packet length in bytes.

Table 84 Event ID 107 (Error)

Event ID Number:	107
Event Message:	'Verify ATS Table error: buffer overflow: buf index = %d, cmd num = %d, pkt len = %d'
Type:	ERROR
Cause:	• This event message indicates an error during verification of an ATS or an Append ATS table due to an invalid table entry. The cause is a table entry with an otherwise valid command packet length that would extend past the end of the table buffer.
	• The event text describes the offset into the table in words for the invalid ATS table entry, the valid table entry command number and the table entry command packet length in bytes.

Table 85 Event ID 109 (Error)

Event ID Number:	109
Event Message:	'Verify ATS Table error: dup cmd number: buf index = %d, cmd num = %d, dup index = %d'
Type:	ERROR
Cause:	• This event message indicates an error during verification of an ATS or an Append ATS table due to an invalid table entry. The cause is a table entry with an otherwise valid command number that is already in use by an earlier table entry.
	• The event text describes the offset into the table in words for the invalid ATS table entry, the table entry command number and the table offset in words for the earlier entry using the same number.

Table 86 Event ID 110 (Error)

Event ID Number:	110
Event Message:	'Verify ATS Table error: table is empty'
Type:	ERROR
Cause:	• This event message indicates an error during verification of an ATS or an Append ATS table due to the table having no entries.
	• This error can only occur if the first entry in the table has a command number equal to zero - the end of data marker.

Table 87 Event ID 111 (Error)

Event ID Number:	111
Event Message:	'Table manage command packet error: table ID = %d'
Type:	ERROR
Cause:	• This event message is issued upon receipt of a table manage request command that has an invalid table ID argument.
	• The event text includes the invalid table ID.

Table 88 Event ID 112 (Error)

Event ID Number:	112
Event Message:	'RTS table manage process error: RTS = %d, Result = 0x%X'

Event ID Number:	112
Type:	ERROR
Cause:	• The expectation is that this command is sent by cFE Table Services only when the indicated RTS table has been updated. Thus, after allowing cFE Table Services an opportunity to manage the table, the call to re-acquire the table data pointer is expected to return an indication that the table data has been updated. This event message is issued upon receipt of any other function result.
	• The event text includes the RTS number and the unexpected result.

Table 89 Event ID 113 (Error)

Event ID Number:	113
Event Message:	'ATS table manage process error: ATS = %d, Result = 0x%X'
Type:	ERROR
Cause:	• The expectation is that this command is sent by cFE Table Services only when the indicated ATS table has been updated. Thus, after allowing cFE Table Services an opportunity to manage the table, the call to re-acquire the table data pointer is expected to return an indication that the table data has been updated. This event message is issued upon receipt of any other function result.
	• The event text includes the ATS number and the unexpected result.

Table 90 Event ID 114 (Error)

Event ID Number:	114
Event Message:	'ATS Append table manage process error: Result = 0x%X'
Type:	ERROR
Cause:	• The expectation is that this command is sent by cFE Table Services only when the ATS Append table has been updated. Thus, after allowing cFE Table Services an opportunity to manage the table, the call to re-acquire the table data pointer is expected to return an indication that the table data has been updated. This event message is issued upon receipt of any other function result.
	• The event text includes the unexpected result.

Table 91 Event ID 116 (Error)

Event ID Number:	116
Event Message:	'Start RTS group error: FirstID=%d, LastID=%d'
Type:	ERROR
Cause:	This event message is issued when a SC_START_RTSGRP_CC command was rejected because the RTS group definition was invalid. • First RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be greater than or equal to First RTS ID. The event text includes the invalid group definition values..

Table 92 Event ID 118 (Error)

Event ID Number:	118
Event Message:	'Stop RTS group error: FirstID=%d, LastID=%d'
Type:	ERROR
Cause:	This event message is issued when a SC_STOP_RTSGRP_CC command was rejected because the RTS group definition was invalid. • First RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be greater than or equal to First RTS ID. The event text includes the invalid group definition values.

Table 93 Event ID 120 (Error)

Event ID Number:	120
Event Message:	'Disable RTS group error: FirstID=%d, LastID=%d'
Type:	ERROR
Cause:	This event message is issued when a SC_DISABLE_RTSGRP_CC command was rejected because the RTS group definition was invalid. • First RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be greater than or equal to First RTS ID. The event text includes the invalid group definition values.

Table 94 Event ID 122 (Error)

Event ID Number:	122
Event Message:	'Enable RTS group error: FirstID=%d, LastID=%d'
Type:	ERROR
Cause:	This event message is issued when a SC_ENABLE_RTSGRP_CC command was rejected because the RTS group definition was invalid.: • First RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be 1 through SC_NUMBER_OF_RTS. • Last RTS ID must be greater than or equal to First RTS ID. The event text includes the invalid group definition values.

A.2.2 Event Messages - INFORMATION

The tables in this section show **informational** event messages for the SC application.

Table 95 Event ID 9 (Informational)

Event ID Number:	9
Event Message:	'SC Initialized. Version %d.%d.%d.%d'
Type:	INFORMATIONAL
Cause:	• This event message is issued when the App has completed initialization.
	• The **`Version`** fields contain the SC_MAJOR_VERSION, SC_MINOR_VERSION, SC_REVISION, and SC_MISSION_REV version identifiers.

Table 96 Event ID 21 (Informational)

Event ID Number:	21
Event Message:	'RTS table file load count = %d'
Type:	INFORMATIONAL
Cause:	This event message indicates the number of RTS table files successfully loaded at startup.

Table 97 Event ID 22 (Informational)

Event ID Number:	22
Event Message:	'Application Data and Tables saved on exit'
Type:	INFORMATIONAL
Cause:	This event message is issued when SC exits cleanly, but no information was saved to the CDS.

Table 98 Event ID 23 (Informational)

Event ID Number:	23
Event Message:	'ATS %c Execution Started'
Type:	INFORMATIONAL
Cause:	• This event message is issued when an ATS is started successfully.
	• The **Sequence** field contains which ATS was started.

Table 99 Event ID 27 (Informational)

Event ID Number:	27
Event Message:	'ATS %c stopped'
Type:	INFORMATIONAL
Cause:	• This event message is issued when a SC_STOP_ATS_CC command successfully stopped an ATS.
	• The **ATS** field contains the ATS that was stopped.

Table 100 Event ID 28 (Informational)

Event ID Number:	28
Event Message:	'There is no ATS running to stop'
Type:	INFORMATIONAL
Cause:	This event message is issued when a SC_STOP_ATS_CC command was issued but there was no ATS running.

Table 101 Event ID 31 (Informational)

Event ID Number:	31
Event Message:	'Switch ATS Pending'
Type:	INFORMATIONAL
Cause:	This event message is issued when a SC_SWITCH_ATS_CC command is issued and the switch is scheduled.

Table 102 Event ID 34 (Informational)

Event ID Number:	34
Event Message:	'ATS Switched from %c to %c'
Type:	INFORMATIONAL
Cause:	• This event message is issued when an ATS switch is scheduled and has been switched.
	• The **`from`** field contains old ATS. • The **`to`** field contains the new running ATS.

Table 103 Event ID 37 (Informational)

Event ID Number:	37
Event Message:	'ATS Switched from %c to %c'
Type:	INFORMATIONAL
Cause:	• This event message is issued when an ATS is scheduled in-line and the switch is successful.
	• The **`from`** field contains old ATS. • The **`to`** field contains the new running ATS.

Table 104 Event ID 40 (Informational)

Event ID Number:	40
Event Message:	'Next ATS command time in the ATP was set to %s'
Type:	INFORMATIONAL

Event ID Number:	40
Cause:	• This event message is issued when a SC_JUMP_ATS_CC command was executed successfully.
	• The **time** field contains the time of the next ATS command after the jump.

Table 105 Event ID 52 (Informational)

Event ID Number:	52
Event Message:	‘No-op command. Version %d.%d.%d.%d’
Type:	INFORMATIONAL
Cause:	• This event message is issued when a SC_NOOP_CC command has been received.
	• The **Version** fields contain the SC_MAJOR_VERSION, SC_MINOR_VERSION, SC_REVISION, and SC_MISSION_REV version identifiers.

Table 106 Event ID 73 (Informational)

Event ID Number:	73
Event Message:	‘RTS Number %03d Started’
Type:	INFORMATIONAL
Cause:	• This event message is issued when an RTS is started successfully.
	• The **Number** field contains the RTS Id that was started.

Table 107 Event ID 78 (Informational)

Event ID Number:	78
Event Message:	‘RTS %03d Aborted’
Type:	INFORMATIONAL
Cause:	• This event message is issued when an SC_STOP_RTS_CC command is received and exexuted successfully.
	• The **RTS** field contains the RTS Id of the RTS that was stopped.

Table 108 Event ID 86 (Informational)

Event ID Number:	86
Event Message:	'RTS %03d Execution Completed'
Type:	INFORMATIONAL
Cause:	• This event message is issued when an RTS completes execution.
	• The **RTS** field contains the RTS Id that completed.

Table 109 Event ID 87 (Informational)

Event ID Number:	87
Event Message:	'ATS %c Execution Completed'
Type:	INFORMATIONAL
Cause:	• This event message is issued when an ATS completes execution.
	• The **ATS** field contains the ATS that completed.

Table 110 Event ID 97 (Informational)

Event ID Number:	97
Event Message:	'Update Append ATS Table: load count = %d, command count = %d, byte count = %d'
Type:	INFORMATIONAL
Cause:	• This event message signals that the Append ATS table has been updated.
	• The displayed values include a load counter to identify this particular update, the number of valid ATS command entries in the table and the size (in bytes) of the portion of the table containing valid ATS command entries.

Table 111 Event ID 98 98 (Informational)

Event ID Number:	98
Event Message:	'Append ATS %c command: %d ATS entries appended'
Type:	INFORMATIONAL

Event ID Number:	98
Cause:	• This event signals the successful completion of an Append ATS command.
	• The event text displays the ATS selection command argument (A or B) and the number of Append table entries appended to the selected ATS.

Table 112 Event ID 103 (Informational)

Event ID Number:	103
Event Message:	'Verify ATS Table: command count = %d, byte count = %d'
Type:	INFORMATIONAL
Cause:	This event message signals the successful verification of an ATS or an Append ATS table. The displayed values indicate the number of table entries (ATS commands) and the total size of the in use table data (in bytes).

Table 113 Event ID 115 (Informational)

Event ID Number:	115
Event Message:	'Start RTS group: FirstID=%d, LastID=%d, Modified=%d'
Type:	INFORMATIONAL
Cause:	• This event message is issued following the successful execution of a SC_START_RTSGRP_CC command. • The event text includes the group definition values and the count of RTS in the group that were actually affected by the command (may be zero).

Table 114 Event ID 117 (Informational)

Event ID Number:	117
Event Message:	'Stop RTS group: FirstID=%d, LastID=%d, Modified=%d'
Type:	INFORMATIONAL
Cause:	• This event message is issued following the successful execution of a SC_STOP_RTSGRP_CC command. • The event text includes the group definition values and the count of RTS in the group that were actually affected by the command (may be zero).

Table 115 Event ID 119 (Informational)

Event ID Number:	119
Event Message:	'Disable RTS group: FirstID=%d, LastID=%d, Modified=%d'
Type:	INFORMATIONAL
Cause:	• This event message is issued following the successful execution of a SC_DISABLE_RTSGRP_CC command. • The event text includes the group definition values and the count of RTS in the group that were actually affected by the command (may be zero).

Table 116 Event ID 121 (Informational)

Event ID Number:	121
Event Message:	'Enable RTS group: FirstID=%d, LastID=%d, Modified=%d'
Type:	INFORMATIONAL
Cause:	• This event message is issued following the successful execution of a SC_ENABLE_RTSGRP_CC command. • The event text includes the group definition values and the count of RTS in the group that were actually affected by the command (may be zero).

A.2.3 Event Messages - DEBUG

The tables in this section show **debug** event messages for the SC application.

Table 117 Event ID 30 (Debug)

Event ID Number:	30
Event Message:	'ATS started, skipped %d commands'
Type:	DEBUG
Cause:	• This event message is issued when an ATS is started, and some of the times for commands in the ATS exist in the past.
	• The **`skipped`** field contains the number of commands that were skipped in the ATS.

Table 118 Event ID 43 (Debug)

Event ID Number:	43
Event Message:	'Continue-ATS-On-Failure command, State: %d'
Type:	DEBUG
Cause:	• This event message is issued when the SC_CONTINUE_ATS_ON_FAILURE_CC command was received and the state was changed successfully.
	• The State field contains the new state for the flag.

Table 119 Event ID 51 (Debug)

Event ID Number:	51
Event Message:	'Reset counters command'
Type:	DEBUG
Cause:	This event message is issued when the SC_RESET_COUNTERS_CC command was received.

Table 120 Event ID 72 (Debug)

Event ID Number:	72
Event Message:	'Start RTS #%d command'
Type:	DEBUG
Cause:	• This event message is issued when a SC_START_RTS_CC cmd is received and is successful.
	• The **RTS** field specifies the RTS ID to start.

Table 121 Event ID 80 (Debug)

Event ID Number:	80
Event Message:	'Disabled RTS %03d'
Type:	DEBUG
Cause:	• This event message is issued when a SC_DISABLE_RTS_CC command was received, and executed successfully.
	• The **RTS** field contains the RTS Id of the RTS that was disabled.

Table 122 Event ID 82 (Debug)

Event ID Number:	82
Event Message:	'Enabled RTS %03d'
Type:	DEBUG
Cause:	• This event message is issued when a SC_ENABLE_RTS_CC command was received, and executed successfully.
	• The **`RTS`** field contains the RTS Id of the RTS that was enabled.

Table 123 Event ID 88 (Debug)

Event ID Number:	88
Event Message:	'Jump Cmd: Skipped %d ATS commands'
Type:	DEBUG
Cause:	• This event message is issued a Jump Command is issued and Some of the ATS commands were marked as skipped.
	• The **`Skipped`** field contains the number of ATS commands that were skipped when jumping forward in time.

A.3 Housekeeping Telemetry Initialized at Reset

On either power-on or processor reset SC initializes the following Housekeeping data. The data is initialized zero unless otherwise noted.

Table 124 Housekeeping Telemetry Initialization Summary

Housekeeping Data	Initialized To
Valid command counter	0
Command rejected counter	0
Total count of commands dispatched from ATSs	0
Total count of commands dispatched from RTSs	0
Total count of commands which failed dispatch from ATSs	0
Total count of commands which failed dispatch from RTSs	0
ATS Table No.1 free byte count[1]	0

[1] Technically, ATS Table No.1 free byte count gets initialized to zero (0) upon startup, but on every housekeeping cycle the telemetry point is updated to be the amount of free bytes left in the ATS. If no ATS is loaded, the value recorded in telemetry would be (SC_ATS_BUFF_SIZE * SC_BYTES_IN_WORD).

Housekeeping Data	Initialized To
ATS Table No.2 free byte count[2]	0
Absolute Time Command processing state	Set to disabled
Identifier of the active ATS table	Set to none
Number of the next ATS command pending execution	0
ATS switch pending flag	0
Time of the next ATS command dispatch	0
The identifier of the ATS table for which the most recent ATS command failed to dispatch	0
The identifier of the most recent ATS command which failed to dispatch from the ATS tables	0
RTS table activation count	0
RTS table activation error count	0
Number of active RTSs	0
Identifier of the next RTS table to dispatch a command	0
The time the next RTS command is due to be dispatched	0
Execution status for each RTS table	Set to idle
Status for each RTS table	Set to disabled
Identifier of the RTS table for which the most recent RTS command dispatch error occurred	0
The word offset within the RTS for the most recent RTS command which failed to dispatch	0
ATS Continue-On-Failure status	Platform defined
The last append ApId	0
The last ATS Append Table command count	0
The last appended count	0

A.4 RTS Group Commands Structure

This section discusses the structure of the RTS group commands. The commands allow groups of RTSs to be started, stopped, disabled, or enabled.

See also: Section 3.3.2, Commanding a Group of RTSs, on page 3-13

The following table shows the structure of the RTS group command.

[2] Technically, ATS Table No.2 free byte count gets initialized to zero (0) upon startup, but on every housekeeping cycle the telemetry point is updated to be the amount of free bytes left in the ATS. If no ATS is loaded, the value recorded in telemetry would be (SC_ATS_BUFF_SIZE * SC_BYTES_IN_WORD).

Table 125 Structure of the RTS Group Command

Type	Name	Description
uint8	CmdHeader [CFE_SB_CMD_HDR_SIZE]	
uint16	FirstRtsId	ID of the first RTS to act on, 1 through SC_NUMBER_OF_RTS.
uint16	LastRtsId	ID of the last RTS to act on, 1 through SC_NUMBER_OF_RTS.

A.5 Telemetry and Housekeeping

A.5.1 Command, HK, 1Hz, Telemetry Msg ID

This section details message ID for number, description, housekeeping and telemetry. CFS SC default values are provided below, but note that the information is highly mission-specific.

Table 126 Command, HK, 1Hz, Telemetry Msg ID Detail

Message ID	CFS SC Value	Mission-Specific Value	Description
SC_CMD_MID	(0x18A9)	Mission Defined	Msg ID for commands to SC
SC_SEND_HK_MID	(0x18AA)	Mission Defined	Msg ID to request SC HK
SC_1HZ_WAKEUP_MID	(0x18AB)	Mission Defined	Msg ID to receive the 1Hz
SC_HK_TLM_MID	(0x08AA)	Mission Defined	Msg ID on which to send down telemetry

A.5.2 Housekeeping Packet Structure

The following table shows the structure detail of housekeeping packets.

Table 127 Housekeeping (HK) Packet Telemetry Structure

Type	Name	Description	CFS SC Default Telemetry Mnemonic(s)	Mission-Specific Telemetry Mnemonic
uint16	AppendByteCount	Size of CMD entries in current Append ATS table	$sc_$cpu_SC_Append ByteCount	Mission Defined
uint16	AppendCmdArg	ATS selection argument from most recent Append ATS command	$sc_$cpu_SC_Append CmdArg	Mission Defined

Type	Name	Description	CFS SC Default Telemetry Mnemonic(s)	Mission-Specific Telemetry Mnemonic
uint16	AppendEntryCount	Number of CMD entries in current Append ATS table	$sc_$cpu_SC_Append EntryCount	Mission Defined
uint16	AppendLoadCount	Total number of Append ATS table loads	$sc_$cpu_SC_Append LoadCount	Mission Defined
uint32	AtpCmdNumber	Current command number	$sc_$cpu_SC_AtpCmd Num	Mission Defined
uint32	AtpFreeBytes [SC_NUMBER_OF_ATS]	Free Bytes in each ATS.	$sc_$cpu_SC_AtpFree Bytes	Mission Defined
uint8	AtpState	Current ATP state valid values are: 2 = IDLE, 5 = EXECUTING	$sc_$cpu_SC_AtpState	Mission Defined
uint16	AtsCmdCtr	Total ATS CMD counter counts commands sent by the ATS	$sc_$cpu_SC_ATSCM DPC	Mission Defined
uint16	AtsCmdErrCtr	Total ATS CMD Error counter command errors in the ATS	$sc_$cpu_SC_ATSCM DEC	Mission Defined
uint8	AtsNumber	Current ATS number 1 = ATS A, 2 = ATS B	$sc_$cpu_SC_AtsNum ber	Mission Defined
uint16	CmdCtr	Counts ground requests.	$sc_$cpu_SC_CMDPC	Mission Defined
uint16	CmdErrCtr	Counts request errors.	$sc_$cpu_SC_CMDEC	Mission Defined
uint8	ContinueAtsOnFailureFlag	In the event of ATS execution failure (ATS command fails checksum), the ATS execution will continue if this flag is set to TRUE and will stop if this flag is set to FALSE.	$sc_$cpu_SC_ContOn Fail	Mission Defined
uint16	LastAtsErrCmd	Last ATS errant command number	$sc_$cpu_SC_AtsLast ErrCmd	Mission Defined
uint16	LastAtsErrSeq	Last ATS errant sequence number values: 1 or 2.	$sc_$cpu_SC_AtsLast ErrSeq	Mission Defined

Type	Name	Description	CFS SC Default Telemetry Mnemonic(s)	Mission-Specific Telemetry Mnemonic
uint16	LastRtsErrCmd	The OFFSET in the RTS buffer of the command that had an error. It will be a WORD value. That is, if the first command had an error, this value would be 0, if the second command started at int8 10 in the buffer, this value would be 5.	$sc_$cpu_SC_RtsLast ErrCmd	Mission Defined
uint16	LastRtsErrSeq	Last RTS errant sequence number	$sc_$cpu_SC_RtsLast ErrSeq	Mission Defined
uint32	NextAtsTime	Next ATS command time (seconds).	$sc_$cpu_SC_NextAts Time	Mission Defined
uint32	NextRtsTime	Next RTS command absolute time	$sc_$cpu_SC_NextRts Time	Mission Defined
uint16	NumRtsActive	Number of RTSs currently active	$sc_$cpu_SC_NumRts Active	Mission Defined
uint16	Padding16		$sc_$cpu_SC_Pad	Mission Defined
uint8	Padding8		$sc_$cpu_SC_Pad8	Mission Defined
uint16	RtsActiveCtr	Increments when an RTS is started without error.	$sc_$cpu_SC_RTSAC TPC	Mission Defined
uint16	RtsActiveErrCtr	Increments when an attempt to start an RTS fails.	$sc_$cpu_SC_RTSAC TEC	Mission Defined
uint16	RtsCmdCtr	Counts TOTAL RTS commands that were sent out from ALL active RTSs.	$sc_$cpu_SC_RTSCM DPC	Mission Defined
uint16	RtsCmdErrCtr	Counts TOTAL number of errors from ALL RTSs that are active.	$sc_$cpu_SC_RTSCM DEC	Mission Defined

Type	Name	Description	CFS SC Default Telemetry Mnemonic(s)	Mission-Specific Telemetry Mnemonic
uint16	RtsDisabledStatus [(SC_NUMBER_OF_RTS +15)/16]	RTS disabled status bit map where each uint16 represents 16 RTS numbers. • Array index numbers and bit numbers use base zero indexing, but RTS numbers use base one indexing. • Thus, the LSB (bit zero) of uint16 array index zero represents RTS number 1, and bit one of uint16 array index zero represents RTS number 2, etc. • If an RTS is ENABLED, then the corresponding bit is zero. • If an RTS is DISABLED, then the corresponding bit is one.	$sc_$cpu_SC_RtsDisabled	Mission Defined

Type	Name	Description	CFS SC Default Telemetry Mnemonic(s)	Mission-Specific Telemetry Mnemonic
uint16	RtsExecutingStatus[(SC_NUMBER_OF_RTS+15)/16]	RTS executing status bit map where each uint16 represents 16 RTS numbers. • Array index numbers and bit numbers use base zero indexing, but RTS numbers use base one indexing. • Thus, the LSB (bit zero) of uint16 array index zero represents RTS number 1, and bit one of uint16 array index zero represents RTS number 2, etc. • If an RTS is IDLE, then the corresponding bit is zero. • If an RTS is EXECUTING, then the corresponding bit is one.	$sc_$cpu_SC_RtsExecuting	Mission Defined
uint16	RtsNumber	Next RTS number	$sc_$cpu_SC_NextRtsNum	Mission Defined
uint16	SwitchPendFlag	Is an ATS switch pending? 0 = NO, 1 = YES This means that the ATS switch is waiting until a safe time.	$sc_$cpu_SC_SwitchPendFlag	Mission Defined

A.6 SC Configuration Parameters

The tables in this section show configuration parameters used to configure the CFS Stored Command Application either for each platform or for a mission as a whole.

Configuration parameters are included for reference, but they cannot be changed by the FOT after the SC application has been compiled.

From an FSSE perspective, configuration parameters exist in source code in sc_config.h.

Table 128 Configuration Parameter - Max Buffer Size for an Append ATS in uint16s

Configuration Parameter	SC_APPEND_BUFF_SIZE
CFS SC Default	32
Mission-Specific Default	Mission Defined
Purpose	Max Buffer Size for an Append ATS in uint16s
Description	Max buffer size for an Append ATS in uint16s
Limits	This parameter cannot be larger than SC_ATS_BUFF_SIZE.

Table 129 Configuration Parameter - Append ATS Table Filename

Configuration Parameter	SC_APPEND_FILE_NAME
CFS SC Default	"/ram/sc_append.tbl"
Mission-Specific Default	Mission Defined
Purpose	Append ATS Table Filename
Description	This name describes the default append ATS filename loaded at startup. Often the default Append ATS file contains only a single unused table entry, and is used only to initialize the table state as having data that may be patched.
Limits	SC requires that this name be defined, but otherwise places no limits on the definition. If the named table file does not exist at run time, or the table fails validation, then the table load will fail. Refer to Operating System (OS)_MAX_PATH_LEN for filename length limits.

Table 130 Configuration Parameter - Append ATS Table Object Names

Configuration Parameter	SC_APPEND_TABLE_NAME
CFS SC Default	"APPEND_TBL"

Mission-Specific Default	Mission Defined
Purpose	Append ATS Table Object Names
Description	Unique table object names are required for each table registered with cFE Table Services. This is the table object name for the Append ATS table.
Limits	SC requires that this name be defined, but otherwise places no limits on the definition. Refer to CFE_TBL_MAX_NAME_LENGTH for specific information on limits related to table object names.

Table 131 Configuration Parameter - Name of the Append ATS Information Table

Configuration Parameter	SC_APPENDINFO_TABLE_NAME
CFS SC Default	"APPINF_TBL"
Mission-Specific Default	Mission Defined
Purpose	Name of the Append ATS Information Table
Description	Name of the Append ATS Information Table
Limits	Must be less than CFE_TBL_MAX_NAME_LENGTH

Table 132 Configuration Parameter - Max Buffer Size for an ATS in uint16s

Configuration Parameter	SC_ATS_BUFF_SIZE
CFS SC Default	40000
Mission-Specific Default	Mission Defined
Purpose	Max buffer size for an ATS in uint16s
Description	The max size of an ATS buffer in words (not bytes)
Limits	This parameter can't be larger than an unsigned 16 bit integer (65535)

Table 133 Configuration Parameter - Name Prefix of ATS Cmd Status Table

Configuration Parameter	SC_ATS_CMD_STAT_TABLE_NAME
CFS SC Default	"ATSCMD_TBL"
Mission-Specific Default	Mission Defined
Purpose	Name Prefix of ATS Cmd Status Table
Description	The prefix of the ATS Command Status table names. Note that actual table names will have a 1 digit number postfixed to it depending on the ATS number
Limits	Total length must be less than CFE_TBL_MAX_NAME_LENGTH

Table 134 Configuration Parameter - Name of the ATP Control block table

Configuration Parameter	SC_ATS_CTRL_TABLE_NAME
CFS SC Default	"ATPCTR_TBL"
Mission-Specific Default	Mission Defined
Purpose	Name of the ATP Control block table
Description	
Limits	Must be less than CFE_TBL_MAX_NAME_LENGTH

Table 135 Configuration Parameter - ATS Table Filenames

Configuration Parameter	SC_ATS_FILE_NAME
CFS SC Default	"/ram/sc_ats"
Mission-Specific Default	Mission Defined
Purpose	ATS Table Filenames

Description	• Filenames for the ATS tables loaded at startup are constructed by appending a one digit table identifier plus the extension ".tbl" to the base portion of the filename defined here. • The default definitions will create ATS filenames as follows: "/cf/apps/sc_ats1.tbl or /cf/apps/sc_ats2.tbl"
Limits	SC requires that this name be defined, but otherwise places no limits on the definition. If the named table file does not exist at run time, or the table fails validation, then the table load will fail. Refer to OS_MAX_PATH_LEN for filename length limits.

Table 136 Configuration Parameter - ATS Table Object Names

Configuration Parameter	SC_ATS_TABLE_NAME
CFS SC Default	"ATS_TBL"
Mission-Specific Default	Mission Defined
Purpose	ATS Table Object Names
Description	• Unique table object names are required for each table registered with cFE Table Services. ATS table object names are constructed by appending a one digit table identifier to the base portion of the object name defined here. • The default definitions will create ATS object names as follows: "ATS_TBL1 or ATS_TBL2"
Limits	SC requires that this name be defined, but otherwise places no limits on the definition. Refer to CFE_TBL_MAX_NAME_LENGTH for specific information on limits related to table object names.

Table 137 Configuration Parameter - Name of the ATS Information Table

Configuration Parameter	SC_ATSINFO_TABLE_NAME
CFS SC Default	"ATSINF_TBL"
Mission-Specific Default	Mission Defined
Purpose	Name of the ATS Information Table
Description	
Limits	Must be less than CFE_TBL_MAX_NAME_LENGTH

Table 138 Configuration Parameter - Define Default of Continue-ATS-On-Checksum-Failure Flag

Configuration Parameter	SC_CONT_ON_FAILURE_START
CFS SC Default	FALSE
Mission-Specific Default	Mission Defined
Purpose	Define Default State of Continue-ATS-On-Checksum-Failure Flag
Description	This parameter specifies the default state to continue an ATS when a command in the ATS fails checksum validation
Limits	Must be TRUE or FALSE

Table 139 Configuration Parameter - Define Inclusion State for RTS Group Commands

Configuration Parameter	SC_ENABLE_GROUP_COMMANDS
CFS SC Default	TRUE
Mission-Specific Default	Mission Defined
Purpose	Define inclusion state for RTS group commands
Description	This parameter specifies the inclusion state for the following RTS group commands: • Start RTS group • Stop RTS group • Enable RTS group • Disable RTS group RTS group commands affect a range of consecutive RTS numbers. When set to TRUE, this definition results in the inclusion of the group command handlers into the SC source code.
Limits	Must be defined as TRUE or FALSE

Table 140 Configuration Parameter - Last RTS that will be Sent with an Event ID 73 Message

Configuration Parameter	SC_LAST_RTS_WITH_EVENTS
CFS SC Default	246
Mission-Specific Default	Mission Defined
Purpose	The last RTS that will be sent with an Event ID 73 message
Description	When all RTSs are started, the Event ID 73 event message is sent out. This parameter suppresses that message for all RTSs over this number
Limits	This parameter needs to be less than or equal to SC_NUMBER_OF_RTS

Table 141 Configuration Parameter - Max Number of Commands in each ATS

Configuration Parameter	SC_MAX_ATS_CMDS
CFS SC Default	2500
Mission-Specific Default	Mission Defined
Purpose	Max number of commands in each ATS
Description	The maximum number of commands that are allowed in each ATS
Limits	This parameter can't be larger than an unsigned 16 bit integer (65535).

Table 142 Configuration Parameter - Max Number of Commands Per Second

Configuration Parameter	SC_MAX_CMDS_PER_SEC
CFS SC Default	8
Mission-Specific Default	Mission Defined
Purpose	Max number of commands per second

Description	Maximum number of commands that can be sent out by SC in any given second.
Limits	This parameter can't be larger than an unsigned 16 bit integer (65535), but should be kepoot relatively small to avoid SC hogging the CPU

Table 143 Configuration Parameter - Mission-Specific Version Number for SC Application

Configuration Parameter	SC_MISSION_REV
CFS SC Default	1
Mission-Specific Default	Mission Defined
Purpose	Mission-specific version number for SC application
Description	An application version number consists of four parts: major version number, minor version number, revision number and mission-specific revision number. The mission-specific revision number is defined here and the other parts are defined in "sc_version.h".
Limits	Must be defined as a numeric value that is greater than or equal to zero.

Table 144 Configuration Parameter - Number of RTSs

Configuration Parameter	SC_NUMBER_OF_RTS
CFS SC Default	256
Mission-Specific Default	Mission Defined
Purpose	Number of RTSs
Description	The number of RTSs allowed in the system
Limits	This parameter will dictate the size of the RTS Info Table. This parameter can't be larger than 999.

Table 145 Configuration Parameter - Maximum Packet Size

Configuration Parameter	SC_PACKET_MAX_SIZE

CFS SC Default	250
Mission-Specific Default	Mission Defined
Purpose	Maximum Packet Size
Description	This parameter specifies the maximum size for an ATS or RTS command.
Limits	This parameter must be greater than or equal to SC_PACKET_MIN_SIZE and less than or equal to CFE_SB_MAX_SB_MSG_SIZE.

Table 146 Configuration Parameter - Minimum Packet Size

Configuration Parameter	SC_PACKET_MIN_SIZE
CFS SC Default	8
Mission-Specific Default	Mission Defined
Purpose	Minimum Packet Size
Description	This parameter specifies the minimum size for an ATS or RTS command.
Limits	This parameter must be greater than or equal to CFE_SB_CMD_HDR_SIZE and less than or equal to CFE_SB_MAX_SB_MSG_SIZE.

Table 147 Configuration Parameter - Command Pipe Depth

Configuration Parameter	SC_PIPE_DEPTH
CFS SC Default	12
Mission-Specific Default	Mission Defined
Purpose	Command Pipe Depth
Description	Maximum number of messages that will be allowed in the SC command pipe at one time. Used during initialization in the call to CFE_SB_CreatePipe

Limits	This parameter must be greater than zero and less than or equal to CFE_SB_MAX_PIPE_DEPTH.

Table 148 Configuration Parameter - Name of the RTP Control Block Table

Configuration Parameter	SC_RTP_CTRL_TABLE_NAME
CFS SC Default	"RTPCTR_TBL"
Mission-Specific Default	Mission Defined
Purpose	Name of the RTP Control block table
Description	
Limits	Must be less than CFE_TBL_MAX_NAME_LENGTH

Table 149 Configuration Parameter - Max Buffer Size for an RTS in uint16s

Configuration Parameter	SC_RTS_BUFF_SIZE
CFS SC Default	300
Mission-Specific Default	Mission Defined
Purpose	Max buffer size for an RTS in uint16s
Description	The max size of an RTS buffer in WORDS (not bytes)
Limits	This parameter can't be larger than an unsigned 16 bit integer (65535).

Table 150 Configuration Parameter - RTS Table Filenames

Configuration Parameter	SC_RTS_FILE_NAME
CFS SC Default	"/boot/sc_rts"

Mission-Specific Default	Mission Defined
Purpose	RTS Table Filenames
Description	Filenames for the RTS tables loaded at startup are constructed by appending a one digit table identifier plus the extension ".tbl" to the base portion of the filename defined here. The default definitions will create RTS filenames as follows: "/cf/apps/sc_rts001.tbl, /cf/apps/sc_rts002.tbl, etc"
Limits	SC requires that this name be defined, but otherwise places no limits on the definition. If the named table file does not exist at run time, or the table fails validation, then the table load will fail. Refer to OS_MAX_PATH_LEN for filename length limits.

Table 151 Configuration Parameter - RTS Table Object Names

Configuration Parameter	SC_RTS_TABLE_NAME
CFS SC Default	"RTS_TBL"
Mission-Specific Default	Mission Defined
Purpose	RTS Table Object Names
Description	• Unique table object names are required for each table registered with cFE Table Services. RTS table object names are constructed by appending a three digit table identifier to the base portion of the object name defined here. • The default definitions will create RTS object names as follows: "RTS_TBL001, RTS_TBL002, etc"
Limits	SC requires that this name be defined, but otherwise places no limits on the definition. Refer to CFE_TBL_MAX_NAME_LENGTH for specific information on limits related to table object names.

Table 152 Configuration Parameter - Name of the RTS Information Table

Configuration Parameter	SC_RTSINFO_TABLE_NAME
CFS SC Default	"RTSINF_TBL"
Mission-Specific Default	Mission Defined

Purpose	Name of the RTS Information Table
Description	
Limits	Must be less than CFE_TBL_MAX_NAME_LENGTH

Table 153 Configuration Parameter - Save Data to CDS Compiler Switch

Configuration Parameter	SC_SAVE_TO_CDS
CFS SC Default	FALSE
Mission-Specific Default	Mission Defined
Purpose	Save data to CDS compiler switch
Description	Compile switch that tells SC that we should save data over a processor or application reset by using the Critical Data Store (CDS). Set the value to TRUE or FALSE.
Limits	None

Table 154 Configuration Parameter - Define the Time SC Should Use for its Commands

Configuration Parameter	SC_TIME_TO_USE
CFS SC Default	SC_USE_UTC
Mission-Specific Default	Mission Defined
Purpose	Define the TIME SC should use for its commands
Description	This parameter defines what type of time SC should use for sending out its commands
Limits	Must be SC_USE_CFE_TIME, SC_USE_TAI, or SC_USE_UTC

A.6.1 Internal Command Code Cross Reference

The table below allows one to find the command number from the internal code name. The table is sorted in alphabetical order by internal SC command code name.

Table 155 SC Internal Command Code to Command Number Cross Reference

Internal SC Command Code Name	CMD No.	Name
SC_APPEND_ATS_CC	11	Append to an ATS table
SC_CONTINUE_ATS_ON_FAILURE_CC	10	Set the Continue-On-Checksum-Failure flag
SC_DISABLE_RTS_CC	6	Disable an RTS
SC_DISABLE_RTSGRP_CC	15	Disable a group of RTS
SC_ENABLE_RTS_CC	7	Enable an RTS
SC_ENABLE_RTSGRP_CC	16	Enable a group of RTS
SC_JUMP_ATS_CC	9	Jump the time in the running ATS
SC_MANAGE_TABLE_CC	12	Request from cFE Table Services to manage a table
SC_NOOP_CC	0	Noop
SC_RESET_COUNTERS_CC	1	Reset Counters
SC_START_ATS_CC	2	Start an ATS
SC_START_RTS_CC	4	Start an RTS
SC_START_RTSGRP_CC	13	Start a group of RTS
SC_STOP_ATS_CC	3	Stop an ATS
SC_STOP_RTS_CC	5	Stop an RTS
SC_STOP_RTSGRP_CC	14	Stop a group of RTS
SC_SWITCH_ATS_CC	8	Switch the running ATS

A.6.2 SC Command Detailed Descriptions

The tables below detail SC commands.

Table 156 Command 0: Noop

CFS SC Command Code	0
Command Name	Noop
Description	Implements the Noop command that insures the SC app is alive.
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_NOOP

Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment. • The Event ID 52 informational event message will be generated when the command is received.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment. • Error specific event message Event ID 2 (or as mission defined)
Criticality	None

Table 157 Command 1: Reset Counters

CFS SC Command Code	1
Command Name	Reset Counters
Description	Resets the SC housekeeping counters.
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_ResetCtrs
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will be cleared. • The Event ID 23 informational event message will be generated when the command is received.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected.

Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment. • Error specific event message Event ID 2 (or as mission defined)
Criticality	None

Table 158 Command 2: Start an ATS

CFS SC Command Code	2
Command Name	Start an ATS
Description	Starts the specified ATS
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_StartAts
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • ATP is not idle • ATS specified is not loaded • Invalid ATS ID • All command were skipped
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment. • Error specific event message
Criticality	None

Table 159 Command 3: Stop an ATS

CFS SC Command Code	3

Command Name	Stop an ATS
Description	Stops the specified ATS
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_StopAts
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment. • The Event ID 27) event message will be generated.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment.
Criticality	None

Table 160 Command 4: Start an RTS

CFS SC Command Code	4
Command Name	Start an RTS
Description	Starts the specified RTS.
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_StartRts
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment. • The Event ID 72, Debug, will be sent

Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • Invalid command field in first RTS command • RTS not loaded • RTS already running • RTS is disabled • Invalid RTS ID
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment. • $sc_$cpu_SC_RTSACTEC (or as mission defined) - RTS activate error counter will increment. • Error specific event message
Criticality	None

Table 161 Command 5: Stop an RTS

CFS SC Command Code	5
Command Name	Stop an RTS
Description	Stops the specified RTS
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_StopRts
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment. • The Event ID 78 will be sent.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected. • RTS ID is invalid
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment. • Error specific event message

Criticality	None

Table 162 Command 6: Disable an RTS

CFS SC Command Code	6
Command Name	Disable an RTS
Description	Disables the specified RTS
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_DisableRts
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment • The Event ID 80 will be sent
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • RTS ID is invalid
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment • Error specific event message
Criticality	None

Table 163 Command 7: Enable an RTS

CFS SC Command Code	7
Command Name	Enable an RTS
Description	Enables the specified RTS
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_EnableRts

Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment • The Event ID 82 will be sent
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • RTS ID is invalid
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment • Error specific event message
Criticality	None

Table 164 Command 8: Switch the Running ATS

CFS SC Command Code	8
Command Name	Switch the Running ATS
Description	Switches the running ATS and the ATS no running.
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_SwitchAts
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) command counter will increment. • The Event ID 31 will be sent.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected. • Destination ATS is not loaded. • There is no currently running ATS from which to switch.

Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment. • Error specific event message
Criticality	None

Table 165 Command 9: Jump the Time in the Running ATS

CFS SC Command Code	9
Command Name	Jump the time in the running ATS
Description	Moves the 'current time' pointer in the ATS to another time
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_JumpAts
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment • The Event ID 40 will be sent
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • All ATS Cmds were skipped in the jump, ATS is shut off • No ATS is active
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment • Error specific event message
Criticality	None

Table 166 Command 10: Set the Continue-On-Checksum-Failure Flag

CFS SC Command Code	10
Command Name	Set the Continue-On-Checksum-Failure flag

Description	Sets the flag which specifies whether or not to continue processing an ATS if one of the commands in the ATS fails checksum validation before being sent out.
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_ContAtsOnFailure
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) - command counter will increment • The Event ID 43 will be sent
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • Invalid State specified
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) - command error counter will increment • Error specific event message
Criticality	None

Table 167 Command 11: Append to an ATS table

CFS SC Command Code	11
Command Name	Append to an ATS table
Description	Adds contents of the Append table to the specified ATS table
CFS SC Default Command Mnemonic(s)	$sc_$cpu_SC_AppendATS
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified with the following telemetry: • $sc_$cpu_SC_CMDPC (or as mission defined) command counter will increment

Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • ATS specified is not loaded • Invalid ATS ID • Append table contents too large to fit in ATS free space
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC (or as mission defined) command error counter will increment • Error specific event message
Criticality	None

Table 168 Command 12: Request from cFE Table Services to Manage a Table

CFS SC Command Code	12
Command Name	Request from cFE Table Services to manage a table
Description	This command signals a need for the host application (SC) to allow cFE Table Services to manage the specified table. For loadable tables, this command indicates that a table update is available. For dump only tables, this command indicates that cFE Table Services wants to dump the table data. In either case, the host application must call the table manage API function so that the pending function can be executed within the context of the host. Note: There is no reason for this command to be sent from any source other than cFE Table Services.
CFS SC Default Command Mnemonic(s)	None
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified via: • cFE Table Services housekeeping telemetry
Error Conditions	This command may fail for the following reason(s): • Invalid table ID • Unexpected result during manage of loadable table
Failure Evidence	Evidence of failure may be found in the following telemetry: • cFE Table Services housekeeping telemetry • Error specific SC event message
Criticality	None

Table 169 Command 13: Start a Group of RTS

CFS SC Command Code	13
Command Name	START a group of RTS
Description	The load state for an RTS may be LOADED or NOT LOADED. The enable state for an RTS may be ENABLED or DISABLED. The run state for an RTS may be STARTED or STOPPED. This command STARTS each RTS in the specified group that is currently LOADED, ENABLED and STOPPED.
CFS SC Default Command Mnemonic(s)	None
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified via: • $sc_$cpu_SC_CMDPC - command counter will increment. • SC_STARTRTSGRP_CMD_INF_EID event will indicate the number of RTS in the group that were actually STARTED by the command.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • Invalid group definition, first RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be greater than or equal to first RTS ID • If the group definition is valid the command will report success, regardless of whether any RTS in the group is actually STARTED by the command.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC - command error counter will increment • The SC_LEN_ERR_EID event will indicate invalid command packet length. • The SC_STARTRTSGRP_CMD_ERR_EID event will indicate invalid group definition.
Criticality	None
See also:	SC_STOP_RTSGRP_CC

Table 170 Command 14: Stop a Group of RTS

CFS SC Command Code	14
Command Name	STOP a group of RTS
Description	The load state for an RTS may be LOADED or NOT LOADED. The enable state for an RTS may be ENABLED or DISABLED. The run state for an RTS may be STARTED or STOPPED. This command STOPS each RTS in the specified group that is currently STARTED.
CFS SC Default Command Mnemonic(s)	None
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified via: • $sc_$cpu_SC_CMDPC - command counter will increment. • The SC_STOPRTSGRP_CMD_INF_EID event will indicate the number of RTS in the group that were actually STOPPED by the command.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • Invalid group definition, first RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be greater than or equal to first RTS ID • If the group definition is valid the command will report success, regardless of whether any RTS in the group is actually STOPPED by the command.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC - command error counter will increment • The SC_LEN_ERR_EID event will indicate invalid command packet length. • The SC_STOPRTSGRP_CMD_ERR_EID event will indicate invalid group definition.
Criticality	None
See also:	SC_START_RTSGRP_CC

Table 171 Command 15: Disable a Group of RTS

CFS SC Command Code	15
Command Name	Disable a Group of RTS
Description	The enable state for an RTS may be ENABLED or DISABLED. This command sets the enable state for the specified group of RTS to DISABLED.
CFS SC Default Command Mnemonic(s)	None
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified via: • $sc_$cpu_SC_CMDPC - command counter will increment • The SC_DISRTSGRP_CMD_INF_EID event will indicate the number of RTS in the group that were changed from ENABLED to DISABLED by the command
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • Invalid group definition, first RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be greater than or equal to first RTS ID • If the group definition is valid the command will report success, regardless of whether the group contained an RTS that was not already DISABLED.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC - command error counter will increment • The SC_LEN_ERR_EID event will indicate invalid command packet length. • The SC_DISRTSGRP_CMD_ERR_EID event will indicate invalid group definition.
Criticality	None
See also:	SC_ENABLE_RTSGRP_CC

Table 172 Command 16: Enable a Group of RTS

CFS SC Command Code	16
Command Name	Enable a Group of RTS
Description	The enable state for an RTS may be ENABLED or DISABLED. This command sets the enable state for the specified group of RTS to ENABLED.
CFS SC Default Command Mnemonic(s)	None
Mission-Specific Command Mnemonic(s)	Mission Defined
Command Verification	Successful execution of this command may be verified via: • $sc_$cpu_SC_CMDPC - command counter will increment • The SC_ENARTSGRP_CMD_INF_EID event will indicate success and display the number of RTS that were changed from DISABLED to ENABLED by the command.
Error Conditions	This command may fail for the following reason(s): • Command packet length not as expected • Invalid group definition, first RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be 1 through SC_NUMBER_OF_RTS • Invalid group definition, last RTS ID must be greater than or equal to first RTS ID • If the group definition is valid the command will report success, regardless of whether the group contained an RTS that was not already ENABLED.
Failure Evidence	Evidence of failure may be found in the following telemetry: • $sc_$cpu_SC_CMDEC - command error counter will increment • The SC_LEN_ERR_EID event will indicate invalid command packet length. • The SC_ENARTSGRP_CMD_ERR_EID event will indicate invalid group definition.
Criticality	None
See also:	SC_DISABLE_RTSGRP_CC

This page intentionally blank

Appendix B Mission-Specific Reference

Mission specific SC-related documentation is located at (mission defined).

B.1 ATS Definition Tables

Documentation of ATS Definition Tables is located at (mission defined).

B.2 RTS Definition Tables

Documentation of RTS Definition Tables is located at (mission defined).

B.3 SC Configuration Table

Documentation of SC Configuration Table is located at (mission defined).

This page intentionally blank

Appendix C Document Notes

C.1 Mission-Specific Conventions

- *This document presents selected information that should be removed when tailoring this document for a mission in this italic dark orange Times Roman font.*
- *Command and Telemetry mnemonics are mission-specific. This document as delivered has "suggested" names that may or may not be used by the mission when the MOT creates the ground system database. In particular, the suggested names start with $sc_$cpu which indicate a global setting for spacecraft and processor selection. This has meaning if the mission has multiple spacecraft, each with a copy of cFE/CFS apps being executed, and/or multiple processors per spacecraft, each with a copy of cFE/CFS apps being executed. Most missions have neither and they do not prepend a $sc or $cpu selection to the front of the command name. However it is common for missions to differentiate the spacecraft subsystem commands from instrument commands by prepending a couple of characters (eg. pw for power system) to all the command and telemetry names for that subsystem.*
- *The nomenclature of command and telemetry mnemonics is highly mission-specific, so this document does not attempt to include the actual command and telemetry database names in advance. For example, for MMS the telemetry mnemonics are defined in a Record Definition Language (RDL) file for ASIST, and they will not exactly match the mnemonics in this Guide.*

C.2 Updating This Document

This section is for anyone updating this Guide in the future.

When tailoring this document for a particular mission, remove text appearing in this italic dark orange Times Roman font.

- *Review figures to be sure there is no conflict with mission configurations. Edit figures with MS Visio if necessary.*
- *Add Mission Defined values in Appendix A.*
- *Add the location of mission-specific documentation in Appendix B.*
- *Regenerate the table of contents (TOC); add Appendix page prefixes to TOC as needed.*

This Guide is formatted using Microsoft Word styles. When adding new sections to the Guide, assign paragraphs to the styles shown in the table below. Center tables and figures horizontally on the page. Use 15% grayscale in new table headings.

This document is set up for double-sided printing. Insert a blank page before chapters if the page is an even number, so that chapters begin on the right hand side in printouts.

Table of Contents, Figures, and Tables can be updated automatically, but the letter prefix for Appendix pages must be added manually after update. To update all figure and table references in the document, when using the PC version of Word, select all, then choose F9.

Table 173 Internal Document Styles

Type	Style to be Used	Justification
Chapter titles, subtitles, and subsections.	Heading 1 through 6	Left
First level bullets	"List Bullet 1" style.	Left
Second level bullets	"List Bullet 2" style	Left
Numbered lists	"Style List Enum 0"	Left
Names of code modules	Code	N/A
All text not otherwise tagged	"Body Text"	Full justification
Tables & Figures		
First row of tables	Table Header	Center
All other cells of tables	Table Cells	Left
Figure captions	Caption Figure	Center
Table captions	Caption Table	Center

C.3 Providing Feedback about this Document

For CM reasons, if you find any item in this Guide that is in error, or want to be informed of any updates, please email feedback to the cFE/CFS PDL for validation and routing. As of the date of publication, the cFE/CFS PDL is Susan Strege (susanne.L.strege@nasa.gov).

Besides corrections of any errors, the CFS team is interested in your ideas on improvements that help understanding, such as, perhaps, improvements to flow charts to show better how SC juggles incoming commands with other program responsibilities, for example, or any other area.