**<MKSID-5088>**



core Flight System (cFS) Simulink Interface Layer (SIL)

Requirements Document

Version X.X

**March 1, 2018**

<MKSID-5091>

1.0       Introduction

1.1        Document Purpose

The core Flight Software System (cFS) Simulink Interface Layer (SIL) has been developed by

* the Flight Software Branch (FSB) of the Software Engineering Division (SED) at NASA’s Goddard Space Flight Center,
* the Attitude Control Systems (ACS) Engineering Branch of the Mission Engineering and Systems Analysis (MESA) Division at NASA’s Goddard Space Flight Center,
* NASA’s Ames Research Center,
* and Mathworks.

The purpose of this requirements specification is to define the requirements to be satisfied by the cFS Simulink Interface Layer.  This application has been developed for re-use.  For this reason, several nomenclatures are used in this document to identify configurations for a mission.

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

1.2        Document Scope

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

1.3        Document Organization

This document is organized into three additional sections and several appendices.

Section 2 gives the Simulink Wrapper context.

Section 3 documents the Simulink Wrapper system design decisions and constraints.

Section 4 contains the Simulink Wrapper subsystem requirements

Section 5 contains the Simulink Wrapper detailed functional and performance requirements.

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

1.4        Relevant Documents

1.4.1     Parent Documents

   
1.4.2     Reference Documents

cFE Application Developer’s Guide   582-2007-001

cFE User’s Guide

2.0       **cFS Simulink Interface Layer Application Context**

Matlab/Simulink is an analysis platform which includes simulation capabilities which allows modeling the behavior of dynamic systems. Simulink is a graphical programming language in which higher level functionality is built by the sequential application of algorithms (graphically represented by blocks) on data (graphically represented by signals/lines). Simulink also leverages Matlab’s programming language and integrated analysis tools.

Both Matlab and Simulink feature the capability of generating C code from their respective languages. The generation of code from existing analysis models allows reuse of those models/algorithms as flight software. The code generated by Matlab/Simulink has documented structure and interfaces which can be used to call that code from external applications.

In the most basic sense, the cFS Simulink Interface Layer maps the interfaces generated by the Simulink code generation process (autocode) into cFS APIs so that the autocode can be executed in the CFS environment and take advantage of the standard CFS APIs for message passing, table management, fault reporting, event generation, and scheduling. The SIL does this by extending the autocode process to produce an additional header file, containing a definition of the interface to the generated code. Because the interface definition header is generated by the code generation process itself, no manual configuration of the SIL is necessary for the software developer/integrator, reducing the chance of errors and speeding development and code iteration.

The SIL code, when compiled with the autocode and interface header file, forms a cFS application and handles registering the autocode with the appropriate cFS API’s. This includes subscribing to messages which are input to the autocode, initializing messages which are output by the autocode, registering parameter tables with cFS’s table services, subscribing to the cFS scheduler, and registering itself with the cFS HK application.

In addition, the SIL contains a library of Simulink blocks which allow accessing additional CFS interfaces if needed. These blocks permit the issuing of cFS event messages, reporting status flags to the Limit Checker (LC) application, and retrieving the cFS time from within the model.

It should also be noted that the code and interface definition header do not necessarily need to be generated by the Simulink code generation process. The SIL is agnostic to the source of the interface definition so long as it complies with the requirements enumerated in this document, which enables the use of the SIL as a tool for integrating existing/legacy code into the cFS architecture.

Figure 1.0 shows the context diagram for the cFS Simulink Wrapper.  During initialization, SIL subscribes to messages from other applications as defined in the autocode. The Scheduler Application (SCH) sends periodic commands to SIL as defined in the SCH Schedule Table. Ground commands come from the Command Ingest task (CI). Messages are routed to SIL by the cFE SB Application. SIL learns of ground updates to the SIL tables through the cFE Table Services application.  Messages generated by SIL are routed to Housekeeping (HK) and Data Store (DS) (as long as the applications subscribe to them).  
   
SIL optionally generate a dump-only tables containing the entire state of the autocode.

    
 

**SIL Wrapper Code**

**Simulink**

**Autocode**

Bus Out

Event Signals

Bus In

cFE Software Bus

Event Processing

Other Apps / Mission Ops

cFE Event Services

Mission Ops

- Queue Cmd Message

- Provide Seq#

- Error Check Message

SCH

Execute

House Keeping

Table Management

cFE Table Services

Mission Ops

cFE Time Services

cFS Time

Fetch Time

Fault Flags

**SIL cFS Application**

Figure 1.0 - cFS SIL Context

2.1        Assumptions

The following list summarizes the assumptions made by the cFS Simulink Interface Layer:

* cFE API and OSAL are being used
* A command is sent to SIL to schedule the execution of the application (from Scheduler Application)
* The code being wrapped has generated a compliant interface definition header file to specify its particular interface needs

3.0       Design Specifications

The cFS Simulink Interface Layer requirements and design are based on an evolutionary history while being utilized on several projects. The SIL was originally developed by NASA’s Ames Research Center for use on the LADEE mission and has since been used by Goddard Space Flight Center on the NICER, GEDI, and PACE projects.

CFG

3.1        Design Constraints

The SIL depends on an interface definition header file to specify the interfaces of the code being wrapped. This header file must be generated accurately for the SIL to map the interfaces correctly.

If the header file is generated using Simulink, care must be taken to follow the implementation guide provided with the Simulink library to ensure to code generation process operates correctly.

   
4.0       Subsystem Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | ReqID | Text | Rational | Heritage Reference |
| N/A | N/A | N/A | The SIL, as a wrapper, does not have specific subsystem requirements, rather it enables other code to perform its responsibilities. |  |

<MKSID-5339>

5.0       Detailed Requirements

The following are the detailed requirements for the Simulink Interface Layer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | ReqID | Text | Rational | Heritage Reference |
|  |  | 5.1        Basic Requirements  The following requirements are basic requirements of cFS Simulink Interface Layer.  Some of them are included here to avoid repeating these requirements for each applicable requirement. |  |  |
|  | SIL-1001 | If the message ID of a received command, request, or wake-up message is not recognized, SIL shall reject the command, request, or wake-up message. | Provides extra layer of error checking/verbosity for mis-routed messages. |  |
|  | SIL-1010 | If the message ID of a received data telemetry message is not recognized, the app shall output an error event message and disregard the message data. | Provides extra layer of error checking/verbosity for mis-routed messages. |  |
|  | SIL-1020 | If a received command, request, or wake-up message’s specified length is not equal to the expected message length, SIL shall reject the command, request, or wake-up message. | Basic command verification in the event of SEU or memory corruption. Incorrect message sizes could result in buffer overruns. |  |
|  | SIL-1030 | If a received data telemetry message’s specified length is not equal to the expected length, SIL shall issue an error event message and disregard the message data. | Incorrect message sizes could result in buffer overruns. |  |
|  | SIL-1040 | If SIL accepts a command as valid, SIL shall execute the command, increment the Command Execution Counter and issue an event message. | Operators require feedback on command execution.  Note that this only applies to “ground commands” (i.e. does not include requests that come from the scheduler) |  |
|  | SIL-1050 | If SIL rejects a command, SIL shall abort the command execution, increment the Command Rejected Counter and issue an error event message. | Operators require feedback on command execution |  |
|  |  | 5.2        Table Requirements  The following requirements document the SIL’s definition and registration of tables with the cFS table services |  |  |
|  | TBL-1001 | SIL shall register up to <MISSION\_DEFINED, 10> parameter tables with cFE table services, each with a mission-specified  1) Table Name  2) Table size, in bytes | Supports managing software parameters through cFS table services. |  |
|  | TBL-1002 | On Power-On Reset or Processor Reset, SIL shall load each parameter table with initial data values from a mission-specified file (complete with file name and directory path). |  |  |
|  | TBL-1010 | For each parameter table, SIL shall validate each table update with a corresponding table validation callback function, if that function is specified by the mission in SIL-compatible software | Allows validation of table before activation. |  |
|  | TBL-1011 | For each parameter table, if the mission does not specify a corresponding table validation callback function in SIL-compatible software, SIL shall consider all table updates as valid. |  |  |
|  | TBL-1020 | SIL shall registered all cFE parameter table as double-buffered and loadable. | Loadable supports updating parameters in flight. Double-buffer allows validation of table before activation. |  |
|  |  | 5.3        Software Bus Requirements  The following requirements document the SIL’s definition and registration of messages with the cFS software bus application |  |  |
|  | SB-1001 | SIL shall receive command, request, and wakeup messages with mission-specified App Ids and command function codes. Including:   1. “No-Op” command (Default FcnCode 0) 2. “Reset Housekeeping Counter” command (Default FcnCode 1) 3. "Configure FDC" command (if FDC reporting is enabled) (Default FcnCode 2) 4. "Clear FDC" command (if FDC reporting is enabled) (Default FcnCode 3) 5. Housekeeping request messages 6. Application wakeup messages 7. Table management request messages | Required for interactions with SB and separating Cmds and Tlm. |  |
|  | SB-1010 | SIL shall receive data telemetry messages with <MISSION\_DEFINED> App Ids. | Required for interactions with SB and separating Cmds and Tlm. |  |
|  |  |  |  |  |
|  |  | 5.4        Wake-up Requirements  The following requirements document the SIL’s actions on receipt of a wake-up. |  |  |
|  | WAKE-1001 | When SIL receives a valid application wake-up request message, SIL shall poll for subscribed data telemetry messages. | Gathers inputs for wrapped code. |  |
|  | WAKE-1010 | After polling for subscribed data telemetry messages, the SIL shall call a mission-specified "step" function defined in the SIL-compatible software. | Calls the wrapped code. |  |
|  | WAKE-1020 | After executing the “step” function, SIL shall place each mission-specified output message onto the SB if the message’s mission-specified criteria for output are met as determined by the SIL-compatible software. | Outputs results for wrapped code. |  |
|  | WAKE-1025 | If a message-specified output message does not have a mission-specified criteria for output, the SIL shall always place that message onto the software bus after executing the “step” function. |  |  |
|  | WAKE-1030 | If FDC reporting is enabled, SIL shall output an FDC state telemetry message onto the SB after executing the “step” function. | Outputs results from wrapped code. |  |
|  |  | 5.5        Command Requirements  The following requirements document the SIL’s command handling. Note that some general commanding requirements are presented in the General section. |  |  |
|  | CMD-1001 | Upon receipt of a "No-Op" command, SIL shall output an informational event message and increment the command execution counter. | Allows testing command interfaces. |  |
|  | CMD-1010 | Upon receipt of a "Reset Housekeeping Counter" command, SIL shall reset the values of the command execution counter, the command rejection counter, and the data telemetry input counters to their initialization values. | Important for testing and on-orbit in order to start with a “clean slate” |  |
|  | CMD-1020 | Upon receipt of a “Table management” request, the app shall manage the selected command-specified parameter table using cFE table services. | Needed to allow updating code parameters in flight and for testing. |  |
|  | CMD-1030 | If FDC reporting is enabled, upon receipt of a "Configure FDC" command, SIL shall validate the parameters and set the command-specified FDC IDs to the command-specified state (enabled/disabled). | Supports enabling/disabling FDC flags to support different mission and test configurations.  Note: If SIL-Compatible software uses that function code, will be accepted, otherwize will will be rejected as having invalid function code. |  |
|  | CMD-1040 | If FDC reporting is enabled, upon receipt of a "Clear FDC" command, SIL shall validate the parameters and shall clear the command-specified FDC IDs’ latch status. | Important for testing in order to start with a “clean slate” and fault recovery  Note: If SIL-Compatible software uses that function code, will be accepted, otherwize will will be rejected as having invalid function code. |  |
|  | CMD-1050 | For all other commands, if the command function code is within a mission-specified range of values, SIL shall store the command message in a SIL-defined command queue. | Allows autocode to process any commands not destined for the wrapper code. |  |
|  | CMD-1051 | SIL shall buffer up to <MISSION\_DEFINED, 25> messages at any given time on the command queue. | Wrapped code can only execute one command per cycle, so commands must be buffered so as not to lose any. |  |
|  | CMD-1052 | If the command queue is full, SIL shall disregard any additional command messages received, increment the command rejection counter, and output an error event message. | Buffers must be finite size, so communicate an overflow to operators. |  |
|  | CMD-1053 | After executing the “step” function, SIL shall remove the oldest message on the command queue, increment the command execution counter, and store the message data for use by the SIL-compatible software during the next execution of “step” function. | SIL-Compatible software is only capable of receiving one command per cycle. |  |
|  | CMD-1060 | If SIL receives a command message with an command function code outside a <MISSION\_DEFINED> range of valid values, SIL shall increment the command rejection counter, output an error event message, then disregard the message | Handle unrecognized command function code. |  |
|  |  | 5.6 Time Requirements  The following requirements document the SIL’s interface with time services. |  |  |
|  | TIME-1001 | If time-fetching is <MISSION\_DEFINED> as enabled, SIL shall check system time via cFE Time Services upon receipt of a valid application wake-up message and pass the time data to the SIL-compatible software. | Enables Sim-compatible software usage of FSW time within algorithms. |  |
|  |  | 5.7        Event Requirements  The following requirements document the SIL’s definition and registration of events with the cFS event services application |  |  |
|  |  | After executing the application’s “step” function, SIL small execute cFE event services for a list of <MISSION\_DEFINED> Event Ids specified in SIL-compatible software if a corresponding <MISSION\_DEFINED> criteria for each event is met as determined by SIL-compatible software |  |  |
|  |  | For each Event ID which meets its corresponding <MISSION\_DEFINED> criteria, SIL shall execute cFE event services consistent with its <MISSION\_DEFINED> Event services parameters that are specified in SIL-compatible software including:  1) Event Type  2) Event Mask  3) Event Message String  4) Event Message Data Points (Up to 5) |  |  |
|  |  |  |  |  |
|  |  | 5.8        FDC Requirements  The following are the requirements associated with cFS Simulink Interface Layer reporting the fault status of the wrapped code. |  |  |
|  | FDC-1001 | SIL shall enable FDC reporting if a <MISSION\_DEFINED> parameter is set in SIL-compatible software. |  |  |
|  | FDC-1010 | If FDC reporting is enabled, SIL shall report on up to <MISSION\_DEFINED, 80> FDC IDs. |  |  |
|  |  |  |  |  |
|  | FDC-1030 | If FDC Reporting is enabled, SIL shall update the FDC State and Latch Status for each valid FDC ID after executing the application’s “step” function based upon whether each FDC ID’s <MISSION\_DEFINED> fault-detection criteria is met as determined in SIL-compatible software. |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | 5.9        Status Reporting  The following are the requirements associated with cFS Simulink Interface Layer reporting its status to the HK application. |  |  |
|  | STAT-1001 | Upon receipt of a valid housekeeping request, SIL shall place an application housekeeping telemetry message onto the SB which contains the following housekeeping data:  1) Command Execution Counter  2) Command Rejection Counter  3) FDC Enabled/Disabled Status (If FDC-Reporting Enabled)  4) FDC Latch Status (If FDC-Reporting Enabled)  5) Counter of each mission-specified data telemetry message received | Prevents HK watchdog from flagging fault, reports interface status for operators |  |
|  |  | 5.10        Initialization Requirements  The following are the requirements associated with cFS Simulink Interface Layer on an Application reset, cFE Processor Reset, or a cFE Power-on Reset |  |  |
|  | INIT-1001 | Upon a Power-On Reset or Processor Reset, SIL shall initialize the following housekeeping data:   1. Command Executed Counter to 0 2. Command Rejected Counter to 0 3. FDC Enabled Status to Disabled for all FDC IDs (If FDC-Reporting Enabled) 4. FDC Latch Status to UnLatched for all FDC IDs (If FDC-Reporting Enabled) 5. Data Telemetry Message Counters to 0 | Need to initialize values to a default state on cFE Power-on reset |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | INIT-1012 | Upon Power-On Reset or Processor Reset, if there is an error loading default table values to a parameter table, SIL shall output an error event message, fail initialization, and exit the cFS-SIL application. | Handle fault condition. |  |
|  | INIT-1013 | Upon Power-On Reset or Processor Reset, If SIL-compatible software specifies more FDC IDs than the maximum-allowed number of FDC IDs, SIL shall output an error event message, fail initialization, and exit the cFS-SIL application. | Handle Fault condition |  |
|  | INIT-1020 | Upon Power-On Reset or Processor Reset, if SIL does not fail initialization, SIL shall produce an event message indicating successful completion of initialization. | Indicates for operators when application has entered nominal operations. |  |
|  |  |  |  |  |

<MKSID-5353>

Appendix A Terminology

<MKSID-5361>

 <MKSID-5367>

This appendix contains the list of terminology for the CFS Simulink Wrapper used in this document

* **cFS –** Core Flight System, software framework for running modular applications
* **SIL –** Simulink Interface Layer, prior name of SIL
* **LC –** Limit Checker Application, cFS application for monitoring telemetry values and issuing actions in response
* **SB –** Software Bus,
* **FDC** – Fault Detection and Correction,
* **EVS –** Event Services, cFS application

<MKSID-5383>

<MKSID-5385>

<MKSID-5387>

<MKSID-5391>

<MKSID-5393>

<MKSID-5395>

<MKSID-5397>

<MKSID-5399>

<MKSID-5401>

<MKSID-5403>

<MKSID-5405>

<MKSID-5407>

<MKSID-5409>

Appendix B Future Ideas

1. Support for multirate model

* Current workaround is to separate rate groups into separate models/CFE apps
* Could potentially be supported via CFE Child Task API, but may be complicated
* If not officially supported, perhaps prepare ‘official’ recommendation and/or examples on how to implement these models
* APL request

2. Support for performance monitoring of sections of model

* Would be a block, placed into an atomic unit of the model (ie model reference, atomic subsystem)
* In simulation, would do nothing (or maybe run tic toc, or maybe report to performance analyzer?)
* In codegen, would code performance entry/exit API calls at the subsystem boundary

3. Support for autogenerated table validation functions

* Utilize matlab coder to generate c code for a matlab function which implements checks on table parameters

4. Support for CFS interfaces from within Matlab blocks

* Supports alternative modeling paradigms by providing functions for events, FDC flags, and retrieving time
* Allows issuing events within Matlab-based logic

5. Support a method of logging FDC/Event signals

* In current implementation separate logging of these signals are required because blocks do not function in simulation mode

6. Direct generation of app from code generation

* Advantages: Simplifies integration of code since no separate code or header files are required… everything is generated
* Disadvantages: Code generation process must now be aware of CFS API and updated to match rather than using header file as interface between generated and hand code

7. Generate model harnesses into unit tests?

8. Autogenerate supporting files (cmake, etc)?

9. Maybe track usage of variant models and make visible in generated code<MKSID-5411>

<MKSID-5415>

<MKSID-5417>

<MKSID-5427>

<MKSID-5431>

<MKSID-5433>