NASA GSFC FLIGHT SOFTWARE SYSTEMS BRANCH

FSW VERSION DESCRIPTION DOCUMENT

CFS LC APPLICATION

BUILD: LC 2.2.0

RELEASE DATE: 9/28/2021

1.0 FSW VERSION DESCRIPTION

1.1 PURPOSE AND SUMMARY

The purpose of this build is to continue to refine the cFS Limit Checker (LC) application product. This build provides various bug fixes and enhancements, but does not include any new functionality. The primary purpose of this release is to ensure compatibility between the LC application and cFS Caelum.

This document serves as the notification of the Build 2.2.0 release of the cFS LC application.

1.2 NEW/CHANGED FUNCTIONALITY IN THIS VERSION

Table 1.2-1 identifies the DCRs that have been implemented in this FSW version. For each DCR the "Key" column shows the corresponding DCR in the GSFC cFS tracking system.

Table 1.2-1 – DCRs Implemented in this Version

Key	Summary	Description
GSFCCFS-1086	'(uint32) &RTSRequest' cast in LC_ExecuteRTS does not work in 64-bit	The function 'void LC_ExecuteRTS(uint16 RTSId)' does not work under 64-bit build due to the casting of the RTSRequest pointer into a uint32. Removal of this casting (3 times within the function) allows the unit tests 'LC_SampleSingleAP_Test_ActiveRequestRTS' and 'LC_ExecuteRTS_Test' to pass instead of segfaulting. It is unknown how this affects functionality in 64-bit of the program outside of unit testing.
GSFCCFS-1093	Migrate LC unit tests to distributed UT Assert	
GSFCCFS-1094	LC Doxygen docs out of date, missing LCX updates	The Doxygen documentation for LC does not include the LCX updates. There is a PowerPoint presentation about the updates, but the Doxygen files continue to mention 'LC_ACTION_NOT_MEASURED' instead of 'LC_ACTION_STALE' which came from the LCX update.
GSFCCFS-1096	LC_SbInit sets LC_OperData.CmdPipe to 0, even though LC_OperData already did a memset to 0	LC_SbInit which is (ONLY!) called from LC_AppInit sets LC_OperData.CmdPipe = 0; However, this is not necessary because LC_AppInit called CFE_PSP_MemSet(&LC_OperData, 0, sizeof(LC_OperData_t)); Therefore, the LC_OperData.CmdPipe is already 0 because there are no changes to it in between the two lines of code.
GSFCCFS-1105	LC_CreateDefinitionTables may call table register twice, but same error results if both calls fail	LC_CreateDefinitionTables may call table register twice for both WDT and ADT; however it only does it if the first call fails AND LC_OperData.HaveActiveCDS == TRUE, then it calls it a second time which will produce an error if it also fails. However, when LC_OperData.HaveActiveCDS == FALSE it produces the same error with only 1 call.

		Should these be different error IDs?
GSFCCFS-1183	LC has static code analysis findings	In analysis done on 7/10/2020, CodeSonar flagged the following findings.
GSFCCFS-1260	Update LC to use new cFE Message Module	
GSFCCFS-1353	LC Comments need updates	LC comments need to be updated with the correct references to #defines and API calls. Specific instances are listed below, but there may be others: --lc_platform_cfg.h ----In comment for LC_RTS_REQ_MID refer to CFE_PLATFORM_SB_HIGHEST_VALID_MSGID instead of CFE_SB_HIGHEST_VALID_MSGID ----In comment for LC_RTS_REQ_CC, update API call to replace CFE_SB_GetCmdCode
GSFCCFS-1354	LC Defines Should be moved to appropriate files and documented	Some #defines in LC don't appear to be in the correct header files, and many need improved documentation. All comments should be scrubbed and specific suggestions of #defines to move are listed below. lc_msgids: Suggest moving LC_ALL_ACTIONPOINTS and LC_ALL_WATCHPOINTS lc_custom.c: Suggest moving the LC_RTSRequest_t to either the mission_cfg.h file or the msg.h file
GSFCCFS-1356	Ensure that LC files contain the correct includes	In the lc_platform_cfg.h file LC states are referenced, but the header file they are defined in is not included. States: LC_STATE_ACTIVE LC_STATE_PASSIVE LC_STATE_DISABLED LC_STATE_FROM_CDS
GSFCCFS-1357	Ensure the LC comments state appropriate assumptions	LC_SampleAPs function assumes that start and end indexes were validated by the calling function, and that if startIndex > endIndex, the function will (intentionally) silently no-op Block comment in LC_SampleSingleAP should be included in doxygen documentation. Functions POP_RPN_DATA and PUSH_RPN_DATA In LC_AppPipe, assumption tat all other messages should be monitor packets should be added to the doxygen. In LC_HousekeepingReq, the block comment about

		CDS behavior at line 492 should also be present in doxygen documentation and the users guide.
GSFCCFS-1359	LC Initialize all variables	Ensure that all LC variables are initialized before usage. Use static code analysis to confirm.
GSFCCFS-1360	LC: Ensure correct bounds checking on all array indicies	Several places to check: --APNumber in LC_SampleSingleAP --APNumber in LC_EvaluateRPN --TableIndex in LC_SetAPStateCmd --TableIndex in LC_SetAPPermOffCmd --CmdPtr->ApNumber in LC_ResetAPStatsCmd --StartIndex, EndIndex in LC_ResetResultsAP --CmdPtr->WPNumber in LC_ResetWPStatsCmd --StartIndex, EndIndex in LC_ResetResultsWP --WatchpointCount in LC_AddWatchpoint --WatchIndex in LC_ProcessWP
GSFCCFS-1361	LC: Replace strncpy with snprintf where possible	Strncpy is used in several places to create event messages. Snprintf can be used instead.
GSFCCFS-1362	LC formatting cleanup	LC has inconsistent indentation and spacing.
GSFCCFS-1363	LC should not pend forever on the software bus	It is a recommended best practice to not use CFE_SB_PEND_FOREVER when waiting on a software bus message
GSFCCFS-1364	LC: Consider reducing functions to one exit point where feasible	JSC recommends having a single exit point where possible. Some LC functions would require significant refactoring. Others (like LC_EvsInit) can be easily reduced to a single exit point.
GSFCCFS-1365	Remove redunant initialization of LC_OperData.CmdPipe	LC_OperData.CmdPipe doesn't need to be set to zero in LC_SbInit because it is initialized with an earlier memset of the entire LC_OperData struct.
GSFCCFS-1366	Condense LC_CreateDefinitionTables	If we initialize OptionFlags to CFE_TBL_OPT_DEFAULT, the else block can be removed.
GSFCCFS-1367	Clarify preprocessor condition in LC_CreateTaskCDS	In LC_CreateTaskCDS, there's a preprocessor condition "#if LC_STATE_WHEN_CDS_RESTORED != LC_STATE_FROM_CDS" Need to verify (1) why this is needed and (2) why it needs to be a pre-processor condition. At minimum, clarifying comments should be added.
GSFCCFS-1369	LC: Add reserved EID 0	Consider adding to lc_events.h /** ** \event 'Reserved event ID - not to be used' */ #define LC_RESERVED_EID 0
GSFCCFS-1371	LC: Ensure that the verify.h file is consistent with the platform_cfg.h file	In lc_verify.h, LC_MAX_WATCHPOINTS is checked against 65520, while the lc_platform_cfg.h file says it must not exceed CFE_TBL_MAX_SNGL_TABLE_SIZE In lc_verify.h LC_MAX_ACTIONPOINTS is

		checked against 65535, while the lc_platform_cfg.h file says it must not exceed CFE_TBL_MAX_SNGL_TABLE_SIZE
GSFCCFS-1372	LC: Replace hard-coded values with #defines	There are many hard-coded constants in LC_GetSizedWPData that should be replaced with #defines. There are several in LC_Uint32IsNAN as well
GSFCCFS-1373	LC: Structures should be passed as pointers to functions	There are a couple places in LC where structs are not passed as pointers when used as function arguments. This should be changed.
GSFCCFS-1480	LC does not build with eval-cert3	
GSFCCFS-1582	LC doxygen config file should be renamed for clarity	The filename "lc_config.txt" suggests that this a configuration file for the app itself as opposed to a configuration file for doxygen.
GSFCCFS-1591	LC should use const for function arguments where possible	
GSFCCFS-1618	LC event messages do not allow for extended message IDs	Events that print out a messageID value use the 0x04X format specifier, which does not work for longer message IDs.
GSFCCFS-1638	LC does not work with extended headers	

1.3 MISSING PLANNED FEATURES AND KNOWN PROBLEMS

Table 1.3-1 identifies currently open DCRs that are not addressed in this build. Any workarounds that may apply are identified.

Information on currently open DCRs is available at:

https://etdjira.gsfc.nasa.gov/projects/GSFCCFS/issues

Note that this is a restricted website that requires a server account. Additional DCRs may have been submitted after preparation of this VDD. A cFS LC DCR report containing a listing of open DCRs is available upon request for customers who do not have access to the restricted server. Please contact the cFS Program Team, cfs-program@nasa.onmicrosoft.com.

Table 1.3-1 – Currently open DCRs

Key	Summary	Description
GSFCCFS-744	LC Transitions Active Action Points to Passive When Application is in Passive Mode	During a project rehearsal there were several APs that were commanded "active" while the LC application state was in "passive" mode. Before operations could command the application state to "active" mode, some of the APs that were activated and had "tripped" causing the AP to transition back to passive mode. The purpose of changing a "tripped" APs

		state from active to passive is to prevent an RTS from getting initiated more than once. In "passive" mode, LC performs all limit tests as in "active" mode, but no stored command sequences are invoked as the result of AP failures. Having the AP's state transition while the application is in passive mode will make enabling APs with a low threshold while LC is in passive mode very difficult. The rational for this design feature (LRO heritage) needs to be clearly understood and documented. The LC user's guides (both doxygen and word/pdf) do not make this design feature clear. If no rational exists this design feature should be removed from LC. Note: The reporting project did not make any modifications to LC as a result of this behavior.
GSFCCFS-753	LC - improve events, generate debug events	LC's (complex) processing of watchpoints and actionpoints is opaque unless you use a code debugger. It would be very helpful to generate debug events. Also LC's event definitions are that the events are of a particular type (such as LC_ART_REGISTER_ERR_EID), whereas EventType should be a separate parameter and the event #defines be generalized such as LC_ART_REGISTER_EID so that debug messages can be produced using the same EID's.
GSFCCFS-768	LC: support 64-bit types	LC currently does not support 64-bit integers or floats (doubles).
GSFCCFS-769	LC - more deterministic behavior	Currently LC will process messages when they are received, which is fine, generally. Also, currently, LC uses a single pipe for commands and for watchpoint telemetry. LC doesn't process action points until it receives a command message. But in deterministic environments it may be better to have LC be commanded to read telemetry as well as be commanded to process action points. This will necessitate having a separate message pipe for commanding (which the main loop would block on) and a telemetry pipe (which would accumulate telemetry until the command to read.) The default LC behavior should remain the same (process messages as they arrive.) This should be a compile-time option, or perhaps run-time command-able.

GSFCCFS-770	LC - platform-endian bytes	LC should support numerical telemetry data types that are platform-endian. While it would be possible to get the same effect with an #if macro block, it would make the table very hard to read.
GSFCCFS-772	LC: Add requirement for handling byte-swapped inputs	One of our instruments is producing byte-swapped values in their housekeeping data and they want us to monitor some values using LC. I looked at the LC code and it appears to handle byte-swapped inputs. However, the feature isn't mentioned in the User's Guide or the LC requirements. Walt said he never tested that feature because it wasn't in the requirements.
GSFCCFS-1061	Limit Checker build tests were not up to date with the latest RDL	The build verification tests failed due to a change made to the FtoTValue and TtoFValue in the Watchpoint Results Table. Updates to these procedures are needed.
GSFCCFS-1075	LC Sets the TtoFValue when it transitions from STALE to FALSE	While testing LC against cFE 6.6, the lc_noaction test procedure failed because values were contained in the TtoFValue when they were not expected. The transition from STALE to FALSE seemed to set these values.
GSFCCFS-1099	LC_SbInit casts every Status to unsigned int in every event message	LC_SbInit is casting the Status in every event send to an unsigned int. The cast appears unnecessary, CFE_EVS_SendEvent takes a variable length of arguments because it does a formatting of the string with the values sent AND the format in the send events is %08X (print 8 characters in upper case hex).
GSFCCFS-1102	LC_TableInit helper functions should be moved to their own file	LC_TableInit has several helper functions that it calls while running. These calls have many different variations and effects upon what happens in LC_TableInit. This makes it very difficult to unit test because there is too much variation in the helper functions to attempt to keep adequate track of what may or may not be happening. If these helper functions were moved to their own .c file they could be wrapped and stubbed for testing LC_TableInit. This would simplify the unit testing process.
GSFCCFS-1103	Should results tables be critical tables instead of stored to CDS	The Watchpoint Results Table and Actionpoint Results Table are currently saved to the CDS on the housekeeping request interval (see functions LC_UpdateTaskCDS and LC_HousekeepingReq). Should these tables be critical tables instead?

GSFCCFS-1104	LC_TableInit has an if/else if without an else clause and its behavior is undefined	So at the end of the LC_TableInit function there is an odd branching
		<pseudo coded here>
		if (LC_CDS_ENABLED)
		{
		if (LC_CDS_RESTORED)
		{
		restored event

		}
		else if (LC_CDS_UPDATED)
		{
		default event
		}
		// nothing else here !!!
		}
		else
		{
		CDS disabled event
		}
		
		So, the question is: Can we have a scenario where LC_CDS_ENABLED is TRUE, but both LC_CDS_RESTORED and LC_CDS_UPDATED are FALSE? And IF SO: What is the desired behavior here?? This is a situation where the lack of an else clause on an "else if" most definitively is cause for concern. If that scenario cannot exist, then it would seem an "else if" is not required. Unfortunately, due to the convoluted nature of this function and those that it calls, it is would be difficult to determine if the above scenario is a possibility.

GSFCCFS-1119	Suggest reversing order in which AP/WP telemetry is stored	LC builds AP/WP status/results starting with most significant bits first. i.e. AP 1 state in most significant 2 bits, then AP 1 results, then AP 0 state, and AP 0 results in least significant bits. When doing an array of bit fields, would be nice to have AP 0 in the most significant bits.
GSFCCFS-1358	Refactor LC_SampleAPs	The function LC_SampleAPs can be refactored - the "if(StartIndex == EndIndex)" condition can be removed.
GSFCCFS-1368	Refactor LC_SampleAPReq and LC_HousekeepingReq	The functions LC_SampleAPReq and LC_Housekeeping Req could be refactored to use a loop instead of the multiple switch statements. The final if condition in LC_SampleAPReq could also be refactored to reduce the nesting.
GSFCCFS-1370	LC: Consider making states into enums	In lc_msgdefs.h, LC states could be made into enums
GSFCCFS-1731	LC has duplicate conditions leading to untestable branches	LC_ValidateWDT has duplicate conditions in its switch statement for both DataType and OperatorID. This leads to branches that cannot be covered by unit testing.

2.0 DELIVERED PRODUCTS

Table 2-1 identifies the locations of FSW products relevant to this FSW Build. The version or date of the Build and where the product can be located are provided. Changes from a previous VDD are identified.

Table 2-1 – Delivered Products and their Locations

Software Element	Changed with this Version?	New Version or Date	Location
Source Code of this FSW Build	Yes	2.2.0	https://github.com/nasa/LC
Doxygen Documentation	Yes	N/A	https://github.com/nasa/LC
Unit Test Data	Yes	2.2.0	https://github.com/nasa/LC
FSW Make Files	Yes	2.2.0	https://github.com/nasa/LC

3.0 INSTALLATION PROCEDURES

In order to build and install the LC application, it must be added to the cFE CMake build system. This is done by modifying the TGTX_APPLIST in the cFE targets.cmake file. This is shown in the trivial example below.

```
SET(TGT1_NAME cpu1)
SET(TGT1_APPLIST lc)
SET(TGT1_FILELIST cfe_es_startup.scr)
```

After LC is added to the targets.cmake file, it is built and installed using the standard cFE CMake build instructions. These instructions are available in cFE CMake documentation:

https://github.com/nasa/cFE/blob/main/cmake/README.md

4.0 CONFIGURATION SUMMARY AND VERSION IDENTIFICATION

This software can be found in the LC GitHub repository (https://github.com/nasa/LC) under the tag “2.2.0”.

Verification of the version can be done by sending an LC NOOP command that produces an event message containing the version information. In addition, the initialization event message generated during the application startup provides the version information.

ACRONYMS

ACS	Attitude Control System
C&DH	Command and Data Handling
cFS	Core Flight System
CM	Configuration Management
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
DCR	Discrepancy/Change Request
ETU	Engineering Test Unit
FSB	Flight Software Branch
FSW	Flight Software
GSFC	Goddard Space Flight Center
I&T	Integration & Test
JSC	Johnson Space Center
LC	Limit Checker
POSIX	Portable Operating System Interface
RTOS	Real-Time Operating System
SMP	Symmetric Multiprocessing
T&C	Telemetry and Command
TBD	To Be Determined
URL	Universal Resource Locator
VDD	Version Description Document