Code 582
Flight Software Branch

CORE FLIGHT SYSTEM MEMORY DWELL BUILD 2.3.1.0

FLIGHT SOFTWARE BUILD VERIFICATON TEST REPORT

Flight Software Branch – Code 582

Version 1.0

SIGNATURES

Submitted by:

X WALTER MOLESKI

Digitally signed by WALTER MOLESKI
DN: c=US, o=U.S. Government, ou=NASA, ou=People, cn=WALTER MOLESKI, 0.9.2342.19200300.100.1.1=wmoleski
Date: 2017.07.13 15:03:27 -04'00'

Walt Moleski
cFS Flight Software Tester

Approved by:

X SUSANNE STREGE

Digitally signed by SUSANNE STREGE
Date: 2017.07.13 14:40:33 -04'00'

Susanne Strege
cFS Flight Software Product Development Lead

PLAN UPDATE HISTORY

Version	Date	Description	Affected Pages
Draft	7/27/2012	Draft for review	All
1.0	7/31/2012	Initial Release	All

TABLE OF CONTENTS

1 INTRODUCTION

1.1 DOCUMENT PURPOSE

This Test Report describes the test results from the Core Flight System (cFS) Memory Dwell (MD) Flight Software (FSW) Test Team build 2.3.1.0 verification testing. It is used to verify that the MD FSW has been tested in a manner that validates that it satisfies the functional and performance requirements defined within the cFS MD Requirements Document. This Test Report summarizes the FSW test history, the build verification process, the build test configuration, and the test execution and results.

1.2 APPLICABLE DOCUMENTS

Unless otherwise stated, these documents refer to the latest version.

Parent Documents (Mission and FSW)

- 582-2007-019 cFS Memory Dwell Requirements Document, Version 1.3
- 582-2008-012 cFS Deployment Guide, Version 3.1

Reference Documents

All of the references below can be found on the Code 582 internal website at http://fsw.gsfc.nasa.gov/

- 582-2003-001 FSB FSW Test Plan Template
- 582-2004-001 FSB FSW Test Description Template
- 582-2004-002 FSB FSW Test Scenario Template
- 582-2004-003 FSB FSW Test Procedure Template
- 582-2004-004 FSB FSW Test Execution Summary Template
- 582-2004-005 FSB Test Product Peer Review Form
- 582-2000-002 FSB FSW Unit Test Standard

1.3 DOCUMENT ORGANIZATION

Section 1 of this document presents some introductory material.

Section 2 provides a flight software overview and context along with the test history and testing overview.

Section 3 describes the build verification process including procedure development and execution and test products produced.

Section 4 describes the build test configuration which includes an overview of the testbed and the requirements verification matrix.

Section 5 describes the test execution and results by subsystem.

Appendix A - provides the Requirements Traceability Matrix

Appendix B - provides the Command, Telemetry, and Events Verification Matrix

1.4 DEFINITIONS

There were 3 verifications methods used during build verification testing. They were:

- Demonstration: Show compliance with system requirement by exhibiting the required capability (e.g. by demonstrating interactive capability, display capability, print capability, etc.
- Inspection: Show compliance with a system requirement by visual verification of the software (e.g. verifying preparation for delivery, proper interfacing)
- Analysis: Perform detailed analysis of code, generated data (both intermediate data and final output data), etc., to determine compliance with system requirements.

The fields in the Requirements Verification Matrix in Section 4.3 are defined as follows:

- Requirements Tested Passed: Requirement was fully tested in a build test procedure and passed all tests.
- Requirements Tested Failed: Requirement was fully tested in a build test procedure and failed one or more aspect of the testing.
- Requirements Tested Partially: Requirement was tested partially in a build test procedure. To be fully tested, the partially tested requirement is either tested additionally in one or more other test procedures within the same build **and/or** other aspects of the requirement must be tested in a later build, due to capabilities not present in the current build
- Total Tested: Total number of requirements fully tested in a build test procedure. Includes total passed and total failed, but does **not** include requirements tested partially, **unless** (included as a separate entry) testing in multiple procedures within the same build constitutes total testing of a particular requirement. Total Requirements Tested is computed this way in order to avoid multiple counting of individual requirements that are tested partially in more than one procedure.
- Deferred: Number of requirements that were planned to be tested in current build, but were not tested due to some FSW capability or necessary system component not being present.
- Total: Total Requirements Tested + Number of Requirements Deferred

In each software test section in Section 5 there is a table of DCR's. The state definitions are as follows:

- Opened: The DCR is currently being addressed
- Assigned: The DCR was accepted and the modification is being addressed
- InTest: The DCR was corrected and is currently in test
- Validated: The DCR was corrected and tested and have been validated, needs to have a CCB to close the DCR
- Closed: The DCR is closed and have been resolved and tested to satisfaction
- Closed with Defect: The DCR is closed and the defect is most likely assigned a differed DCR number associated with another subsystem.

2 OVERVIEW

2.1 FLIGHT DATA SYSTEM CONTEXT

Figure 2-1 illustrates the cFS system context. The cFE interfaces to five external systems: an Operating System (OS), a Hardware Platform (HP), an Operational Interface (OI), Applications (APP), and other cFE-based systems.

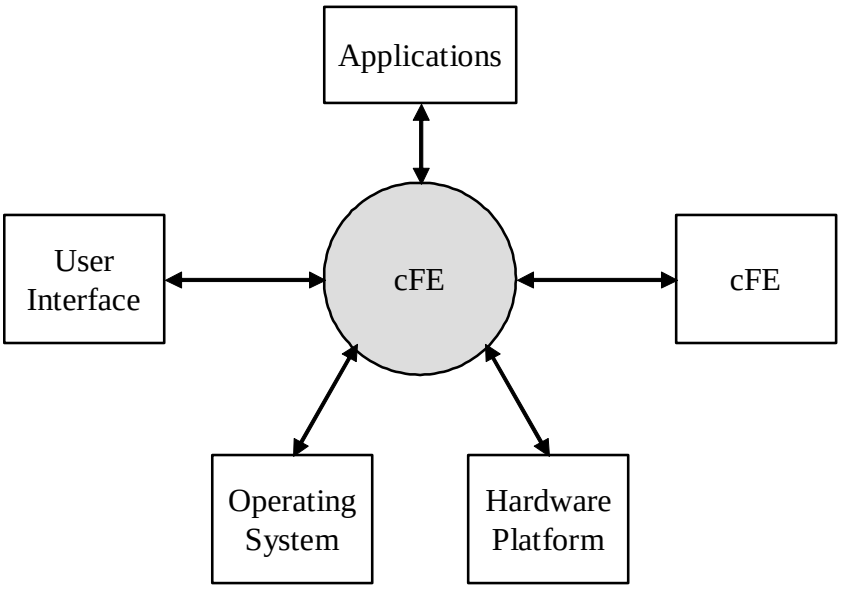

Figure 2-1 cFS System Context

The figure below shows major interfaces between the Memory Dwell task and other core Flight Executive (cFE) and Core Flight System (cFS) tasks. Note that although it isn't shown explicitly, all task-to-task communications are accomplished via the cFE Software Bus task.

Inputs to the Memory Dwell task include: 1) Wake-up calls from the Scheduler (SH) task which trigger dwell processing, 2) Housekeeping requests from the Scheduler (SH) task which trigger housekeeping data collection, 3) configuration commands from the Command Ingest (CI) task, and 4) updates to Memory Dwell Tables managed by the Table Services (TBL) task.

Outputs from the Memory Dwell task include: 1) Memory Dwell housekeeping messages sent to the Housekeeping (HK) task, 2) Dwell messages sent to the Data Storage (DS) task for storage, and metered to the ground by the Telemetry Output (TO) task, and 3) Event messages. Up-to-date values for Memory Dwell Table contents and for other state data that control the generation of dwell packets are maintained in the Critical Data Store (CDS). Upon processor reset or Memory Dwell Application Reset, these data are restored to the Memory Dwell task enabling the task to resume generation of Memory Dwell packets.

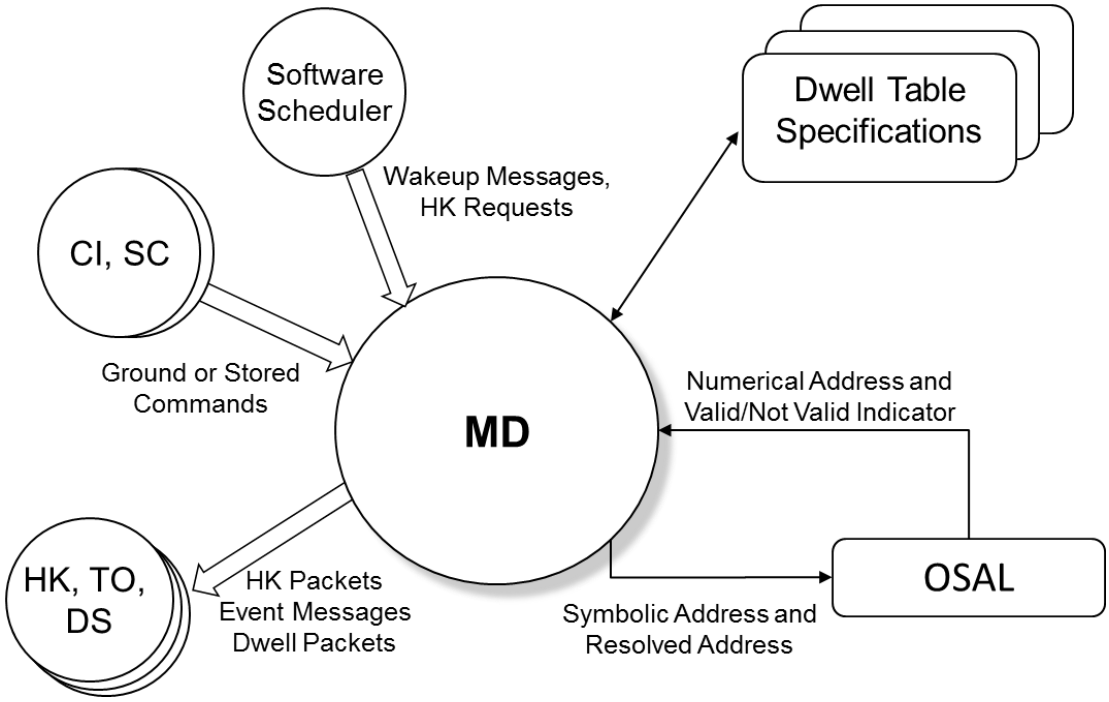

Figure 2-1 cFS MD Context

Memory Dwell makes use of the OSAL when interfacing to memory. Memory Dwell assumes that the OSAL will provide routines to access processor memory as well as memory that is not directly accessible (i.e. requires address translation). Address checking is performed using the OSAL. Any addresses specified outside of the valid address range will be considered invalid.

2.2 TEST HISTORY

MD 1.0.1.0 – Build Verification Testing completed 10/15/2009
MD 1.1.0.0 – Build Verification Testing completed 3/9/2009
MD 1.1.1.0 – Build Verification Testing completed 4/16/2009
MD 2.0.0.0 – Build Verification Testing completed 8/26/2009
MD 2.1.0.0 – Build Verification Testing completed 12/7/2009
MD 2.2.0.0 – Build Verification Testing completed 1/25/2012
MD 2.3.0.0 – Build Verification Testing completed 7/27/2012
MD 2.3.1.0 – Build Verification Testing completed 7/12/2017

2.3 TESTING OVERVIEW

The MD application was tested during Build Verification testing using the following:

- 1 test application: tst_md
- 7 main test procedures: md_ctrlcmds.prc, md_dwellproc.prc, md_gencmds.prc, md_initreset.prc, md_jamdwell.prc, md_signatures.prc, md_symtab.prc
- 1 test procedure that is called by the main procedures: md_deftables.prc
- Test require the Advanced Spacecraft Integration and System Test (ASIST) Ground Station

The tst_md test application is used to send schedule requests for the output of MD's housekeeping data to the MD application. This was useful when performing build verification testing since it provided great control over the sequence of steps. In addition, having the test application eliminated the need to modify the SCH_LAB application and rebuild. When deployed for a mission, the Scheduler Application would provide this request. In addition, the test application also creates an area of memory (256 bytes) with pre-set data that some of the MD tests use to verify that the correct data is coming out in the dwell packets. At startup the area is filled with static (1-byte) values, from 0 to 255. The data remains static unless the TST_MD_STARTDATA command described below is sent. TST_MD has the following ground commands:

- TST_MD_NOOP
  - This command that issues an event and increments the command processed counter.
- TST_MD_ResetCtrs
  - This command resets the command processed and command error counters to zero (0).
- TST_MD_StartData
  - This command starts incrementing the test data at a rate of once per second up to 255 at which time the data rolls over to 0.
- TST_MD_StopData
  - This command stops incrementing the test data.
- TST_MD_ResetData
  - This command initializes the test data to its initial values.

The MD 2.3.1.0 testing was performed using 2 different configurations. Each configuration required a separate compilation with changes to the MD configuration parameters. The configurations are:

- Normal: MD compiled out of the box with long-word alignment enforcement and signatures on.
- No Alignment: MD compiled with long-word alignment turned off.

These 7 main MD test procedures do the following:

Procedure	**Description**
md_gencmds	The purpose of this test is to verify that the Memory Dwell (MD) general commands function properly. MD Initialization and the MD_NOOP and MD_Reset commands will be tested as well as invalid commands.
md_ctrlcmds.prc	The purpose of this test is to verify that the Memory Dwell (MD) Start and Stop control commands function properly.
md_jamdwell	The purpose of this test is to verify that the Memory Dwell (MD) Jam Dwell command functions properly.
md_dwellproc	The purpose of this test is to verify that Memory Dwell (MD) dwell tables are processed properly.
md_initreset	The purpose of this test is to verify that the Memory Dwell (MD) application behaves correctly when it is initialized or reset, storing data in the CDS and restoring the data when a cFE Processor Reset or MD application reset is performed. NOTE: The steps necessary to corrupt the CDS requires manual intervention, therefore this test cannot be run automatically.
md_signatures	The purpose of this test is to verify that the Memory Dwell (MD) Set Dwell Table Signature command functions properly. Memory Dwell Table Signature support is optional and thus provided in a separate test. If the mission provides Memory Dwell Table Signature support, this test can be used to verify its functionality.
md_symtab	The purpose of this test is to verify the Memory Dwell (MD) Symbol Table functionality of the Core Flight System (cFS). Symbol Table support is optional and thus provided in a separate test. If the mission provides Symbol Table support, this test can be used to verify its functionality.

The test procedure described in the table below is called by the main test procedures.

Procedure	Description
md_deftables	The purpose of this proc is to setup and load the default set of dwell tables used by some of the tests

The cFS Deployment Guide contains the instruction for how to set up both the cFS Flight and Ground test environment. The testers use a cFS Test Account for each build test. This account runs ASIST and is setup to contain all the files needed to test the application. These files are extracted from MKS, the source repository tool. Included in these files are test utilities. These utilities can be located in 2 places depending upon whether they are "local" or "global" utilities. The local utilities are extracted into the working prc directory ($WORK/prc). The global utilities are pointed to by ASIST in the global area defined on the test system. Additional tools utilized by the test procedures are located in the $TOOLS directory. It is assumed that test procedures and the ASIST telemetry database used for testing is built using procedure and database templates

The following utilities were used during testing:

Name	Description
CFE_startup	Directive combines the "start_data_center", "open_tlm", and "open cmd <cpu>" ASIST startup commands.
create_tbl_file_from_cvt load_start_app	Procedure that creates a load file from the specified arguments and cvt Procedure to load and start a user application from the /s/opr/accounts/cfebx/apps/cpux directory.
load_table	Procedure that takes the specified file and transfers the file to the specified processor and then issues a TBL_LOAD command using the file.
tst_md (version 2.3.1.0)	Test application with 3 primary commands that the MD test procedures use to start/stop/reset the test data. They are: • StartDataCommand: starts the data in the data area incrementing once per second • StopDataCommand: stops the data in the data area from incrementing • ResetDataCommand: resets the data in the data area back to the default values
ut_pfindicate	Directive to print the pass fail status of a particular requirement number.
ut_runproc	Directive to formally run the procedure and capture the log file.
ut_sendcmd	Directive to send EVS commands Verifies command processed and command error counters.
ut_sendrawcmd	Send raw commands to the spacecraft. Verifies command processed and command error counters.
ut_setrequirements	A directive to set the status of the cFE requirements array.
ut_setupevents	Directive to look for multiple events and increment a value for each event to indicate receipt.
ut_tlmupdate	Procedure to wait for a specified telemetry point to update.
ut_tlmwait	Directive that waits for the specified telemetry condition to be met

2.4 VERSION INFORMATION

Item	Version
MD Requirements	1.3
MD Application	2.3.1.0
TST_MD Application	2.3.1.0
cFE	6.5.0.0
ASIST	20.2
VxWorks	6.9

3 BUILD VERIFICATION TEST PREPARATION

3.1 SCENERIO DEVELOPMENT

No new scenarios were developed for MD 2.3.1.0 Build Verification Test. All scenarios are stored on the MKS server, in cFS-Repository MD test-and-ground directory within the test-review-packages subdirectory in the Scenarios folder. It should be noted that as MD requirements evolve these scenarios are not updated to reflect any changes made.

3.2 PROCEDURE DEVELOPMENT AND EXECUTION

This build test was completed by running 7 test procedures. All test procedures were written using the STOL scripting language. The naming convention for files created by the test procedures was: scx_cpu<#>_<procedure name>_GMT.<ext>.

3.3 TEST PRODUCTS

Four log files were generated for every procedure that was run. They are defined as follows:

- Logs with the .loge extension list all events sent by the flight software
- Logs with the .logr extension list all requirements that passed validation by demonstration
- Logs with the .logp extension lists all prints that are generated by the test procedure
- Logs with the .logf extension lists everything from the other logs along with the steps in the test procedure
- Logs with the .logs extension lists the SFDU information (if applicable) contained in the full log.

A test summary reported is developed in MKS for each procedure by the tester after build testing is completed. All test products are maintained on MKS in the cFS-Repository MD test-and-ground directory.

The MD 2.3.1.0 test results contained 2 sets of test products for the md_dwellproc, md_jamdwell and md_symtab tests. One set enforcing long-word alignment and one set with this enforcement turned off.

4 BUILD VERIFICATION TEST EXECUTION

4.1 TESTBED OVERVIEW

MD FSW testing took place in the cFS FSW Development and Test Facility. A high level view of the cFS FSW Test Bed is shown in Figure 4-1. This facility is located in GSFC Building 23, Room N410. This facility consists of two ASIST workstations running ASIST version 9.7k and three MPC750 CPU boards running VxWorks 6.4. CPU1 is primarily used for development testing while CPU2 and CPU3 are used for build verification testing.

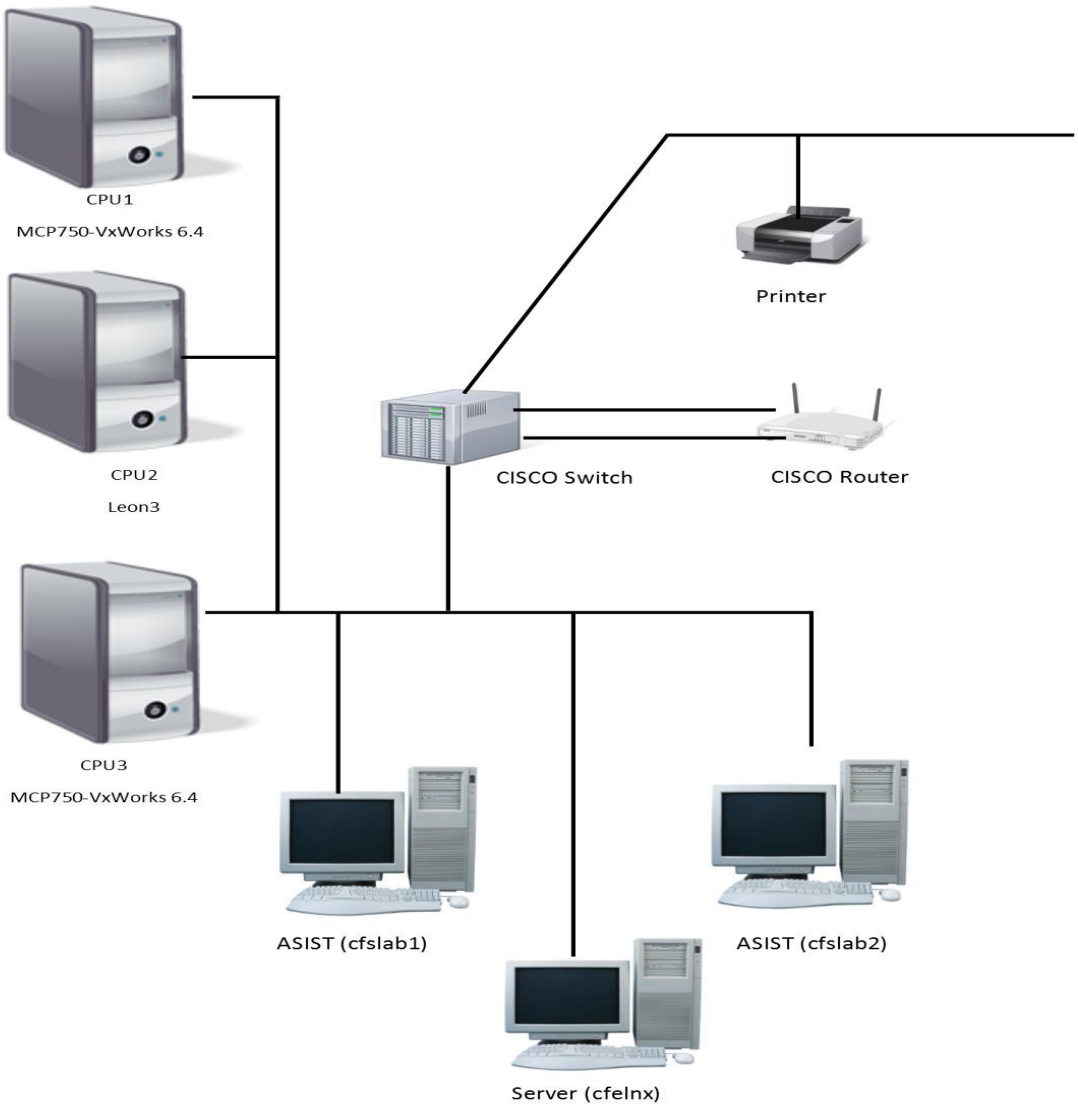

Figure 4-1 cFS FSW Development and Testing Facility

4.2 REQUIREMENTS VERIFICATION MATRIX

	Memory Dwell (MD)
Requirements Tested Passed	35
Requirements Tested Failed	0
Requirements Tested Partially	0
Total Tested	35
Deferred	0
Total	35

4.3 REQUIREMENTS PARTIALLY TESTED

No requirements were partially tested.

4.4 REQUIREMENTS/FUNCTIONALITY DEFERRED

No requirements were deferred to later build testing.

4.5 REQUIREMENTS/FUNCTIONALITY DEFERRED FOR MISSION TESTING

The following functionality was deferred to mission testing:

- RAM was the only physical memory type tested. EEPROM, Compact Flash, SSR not tested. EEPROM testing was done by simulating EEPROM in RAM.

5 BUILD VERFICIATON TEST RESULTS

5.1 OVERALL ASSESSMENT

During this build test of the MD Application the software behaved as expected with several problems still outstanding from previous testing. Below is a summary of the results:

- 35 requirements passed via demonstration
- 5 DCRs were verified.

5.2 PROCEDURE DESCRIPTION

Procedure	Description	Requirements tested
md_ctrlcmds	The purpose of this test is to verify that the Memory Dwell (MD) Start and Stop control commands function properly.	MD1003, MD1004, MD1005, MD2000, MD2002.2, MD2001, MD2001.1, MD4000, MD8000, MD9000, MD9001, MD9002
md_dwellproc	The purpose of this test is to verify that Memory Dwell (MD) dwell tables are processed properly.	MD1004, MD2000, MD2001, MD3000, MD3000.2, MD3000.3, MD3001, MD3002, MD3002.2, MD3002.5, MD4000, MD8000, MD9000, MD9001, MD9002
md_gencmds	The purpose of this test is to verify that the Memory Dwell (MD) general commands function properly. MD Initialization and the MD_NOOP and MD_ResetCtrs commands will be tested as well as invalid commands.	MD1000, MD1001, MD1002, MD1003, MD1004, MD1005, MD1006, MD8000, MD9000, MD9001, MD9002
md_initreset	The purpose of this test is to verify that the Memory Dwell (MD) application behaves correctly when it is initialized or reset, storing data in the CDS and restoring the data when a cFE Processor Reset or MD application reset is performed. NOTE: The steps necessary to corrupt the CDS requires manual intervention, therefore this test cannot be run automatically.	MD1004, MD2000, MD4000, MD8000, MD9000, MD9001, MD9002, MD9003, MD9004, MD9004.1
md_jamdwell	The purpose of this test is to verify that the Memory Dwell (MD) Jam Dwell command functions properly.	MD1003, MD1004, MD1005, MD2000, MD4000, MD4000.1, MD4000.2, MD4000.3, MD4000.5, MD8000, MD9000, MD9001, MD9002,
md_signatures	The purpose of this test is to verify that the Memory Dwell (MD) Set Dwell Table Signature command functions properly. Memory Dwell Table Signature support is optional and thus provided in a separate test. If the mission provides Memory Dwell Table Signature support, this test can be used to verify its functionality.	MD1002, MD1004, MD1005, MD2000, MD3001, MD3002, MD5000, MD5000.1, MD8000, MD9000, MD9001, MD9002, MD9003, MD9004

Procedure	Description	Requirements tested
md_symtab	The purpose of this test is to verify the Memory Dwell (MD) Symbol Table functionality of the Core Flight System (cFS). Symbol Table support is optional and thus provided in a separate test. If the mission provides Symbol Table support, this test can be used to verify its functionality.	MD1004, MD1005, MD2000, MD3000, MD3000.1, MD3001, MD3002, MD3002.2, MD3002.4, MD4000, MD4000.2, MD4000.4, MD8000, MD9000, MD9001, MD9002, MD9003, MD9004

5.3 ANALYSIS REQUIREMENTS VERIFICATION

No requirements were verified using analysis.

5.4 DCRS

No new DCRs were generated during MD 2.3.1.0 testing.

5.4.1 DCRs Verified

DCR	Description	Test Method	Test Approach
3699	The MD User's Guide does not contain information in the Commands Section	Inspection	The User's Guide was displayed in a browser and the Commands Section now contains information.
3716	MD table verify events and default table location need to be updated	Test Procedure	All MD test procedures perform table validation. The validation message implemented by this DCR was generated after each validation. The default table file location was as stated in the DCR as well.
4099	MD Event ID 48 Is Defined and Not Used	Inspection	The stated Event ID was removed from the MD fsw.
145924	MD – CFE_EVA_SendEvent Format Warnings	Demonstration	No compiler warnings were generated during the make process for MD.
146175	MD – Fix use of uint32 to Store/Reference a Memory Address	Inspection	The stated fix in this DCR was verified for each instance of a memory address in the MD fsw.

5.4.2 Outstanding DCRs

Trac ticket references are proceeded with a '#' character

DCR/Trac #	Description
#89	MD Table Configuration is Not Consistent with Other Applications: MD does not allow the option to save/not save tables in the CDS MD does not allow default address to be configurable MD does not use the CFE_TBL_Manage feature
4118	MD - Add Trick Simulation Support (JSC Request)

5.5 NOTES

It should be noted that integration testing is the ultimate verification of the MD applications performance in a system-like scenario.

APPENDIX A - RTTM

The MD Build 2.3.1.0 RTTM can be found on the MKS server, in cFS-Repository MD test-and-ground/results folder.

APPENDIX B - COMMAND, TELEMETRY, AND EVENTS VERIFICATION MATRIX

Command	Test Procedure(s)	Notes/Comments
MD_NOOP	md_gencmds	
MD_ResetCtrs	md_gencmds	
MD_StartDwell	md_ctrlcmds, md_dwellproc md_gencmds, md_initreset md_jamdwell, md_symtab md_signatures	
MD_StopDwell	md_ctrlcmds, md_dwellproc md_gencmds	
MD_JamDwell	md_ctrlcmds, md_dwellproc md_gencmds, md_initreset md_jamdwell, md_symtab	
MD_SetSignature	md_signatures	

Telemetry	Test Procedure(s)	Notes/Comments
MD_CMDEC	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_initreset, md_symtab md_signatures	
MD_CMDPC	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_initreset, md_symtab md_signatures	
MD_EnableMask	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_initreset, md_symtab md_signatures	
MD_AddrCnt[]	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_symtab, md_signatures	
MD_Rate[]	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_symtab, md_signatures	
MD_DataSize[]	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_symtab, md_signatures	
MD_DwPktOffset[]	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_symtab, md_signatures	
MD_DwTblEntry[]	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_symtab, md_signatures	
MD_CountDown[]	md_ctrlcmds, md_dwellproc md_gencmds, md_jamdwell md_symtab, md_signatures	
MD_DwlPkt1TableId	md_dwellproc, md_symtab	
MD_DwlPkt1NumAddresses	md_dwellproc, md_symtab	
MD_DwlPkt1DataSize	md_dwellproc, md_symtab	
MD_DwlPkt1Rate	md_dwellproc, md_symtab	
MD_DwlPkt1Signature	md_signatures	

Telemetry	Test Procedure(s)	Notes/Comments
MD_DwlPkt1DwellData[]	md_dwellproc, md_symtab	
MD_DwlPkt2TableId	md_dwellproc, md_symtab	
MD_DwlPkt2NumAddresses	md_dwellproc, md_symtab	
MD_DwlPkt2DataSize	md_dwellproc, md_symtab	
MD_DwlPkt2Rate	md_dwellproc, md_symtab	
MD_DwlPkt2Signature	md_signatures	
MD_DwlPkt2DwellData[]	md_dwellproc, md_symtab	
MD_DwlPkt3TableId	md_symtab	
MD_DwlPkt3NumAddresses	md_symtab	
MD_DwlPkt3DataSize	md_symtab	
MD_DwlPkt3Rate	md_symtab	
MD_DwlPkt3Signature	md_signatures	
MD_DwlPkt3DwellData[]	md_symtab	
MD_DwlPkt4TableId	md_dwellproc	
MD_DwlPkt4NumAddresses	md_dwellproc	
MD_DwlPkt4DataSize	md_dwellproc	
MD_DwlPkt4Rate	md_dwellproc	
MD_DwlPkt4Signature	md_signatures	
MD_DwlPkt4DwellData[]	md_dwellproc	

Id	Event Message	Test Procedure(s)	Notes/Comments
1	MD_INIT_INF_EID	md_ctrlcmds, md_dwellproc, md_gencmds, md_jamdwell, md_initreset, md_signatures, md_symtab	
2	MD_PIPE_ERR_EID		
3	MD_RECOVERED_TBL_VALID_INF_EID	md_initreset, md_signatures, md_symtab	
4	MD_RECOVERED_TBL_NOT_VALID_ERR_EID		
5	MD_DWELL_TBL_TOO_LARGE_CRIT_EID		
6	MD_TBL_REGISTER_CRIT_EID		
7	MD_TBL_INIT_INF_EID	md_ctrlcmds, md_dwellproc, md_gencmds, md_initreset, md_jamdwell, md_signatures, md_symtab	
10	MD_NOOP_INF_EID	md_gencmds	
11	MD_RESET_CNTRS_DBG_EID	md_gencmds	
12	MD_START_DWELL_INF_EID	md_ctrlcmds, md_dwellproc, md_jamdwell, md_initreset, md_signatures, md_symtab	
13	MD_STOP_DWELL_INF_EID	md_ctrlcmds, md_dwellproc	
14	MD_EMPTY_TBLMASK_ERR_EID	md_ctrlcmds, md_gencmds	
15	MD_MID_ERR_EID		
16	MD_CC_NOT_IN_TBL_ERR_EID	md_gencmds	
17	MD_CC_NOT_IN_LOOP_ERR_EID		
20	MD_TBL_STATUS_ERR_EID		
21	MD_CMD_LEN_ERR_EID	md_gencmds, md_signatures	
22	MD_MSG_LEN_ERR_EID		
30	MD_JAM_DWELL_INF_EID	md_ctrlcmds, md_jamdwell, md_initreset, md_symtab	

31	MD_JAM_NULL_DWELL_INF_EID	md_ctrlcmds, md_dwellproc, md_initreset	
32	MD_INVALID_JAM_TABLE_ERR_EID	md_jamdwell	
33	MD_INVALID_ENTRY_ARG_ERR_EID		
34	MD_INVALID_LEN_ARG_ERR_EID	md_jamdwell	
35	MD_CANT_RESOLVE_JAM_ADDR_ERR_EID	md_gencmds, md_jamdwell, md_symtab	
36	MD_INVALID_JAM_ADDR_ERR_EID	md_ctrlcmds, md_jamdwell, md_initreset, md_symtab	
37	MD_JAM_ADDR_NOT_32BIT_ERR_EID	md_jamdwell, md_symtab	
38	MD_JAM_ADDR_NOT_16BIT_ERR_EID	md_jamdwell, md_symtab	
39	MD_NO_TBL_COPY_ERR_EID		
40	MD_ZERO_RATE_TBL_ERR_EID		
41	MD_RESOLVE_ERR_EID	md_dwellproc, md_symtab	
42	MD_RANGE_ERR_EID	md_dwellproc, md_initreset, md_signatures, md_symtab	
43	MD_TBL_HAS_LEN_ERR_EID	md_dwellproc	
44	MD_TBL_ENA_FLAG_EID		
45	MD_TBL_ALIGN_ERR_EID	md_dwellproc, md_symtab	
46	MD_SET_SIGNATURE_INF_EID	md_signatures	
47	MD_INVALID_SIGNATURE_TABLE_ERR_EID		
48	MD_SIGNATURE_TOO_LONG_ERR_EID		
49	MD_INVALID_SIGNATURE_LENGTH_ERR_EID	md_signatures	
50	MD_TBL_SIG_LEN_ERR_EID		
51	MD_ZERO_RATE_CMD_INF_EID	md_ctrlcmds, md_dwellproc, md_jamdwell	