OpenSatKit
Mission FSW Guide
Quick Start

V3.0

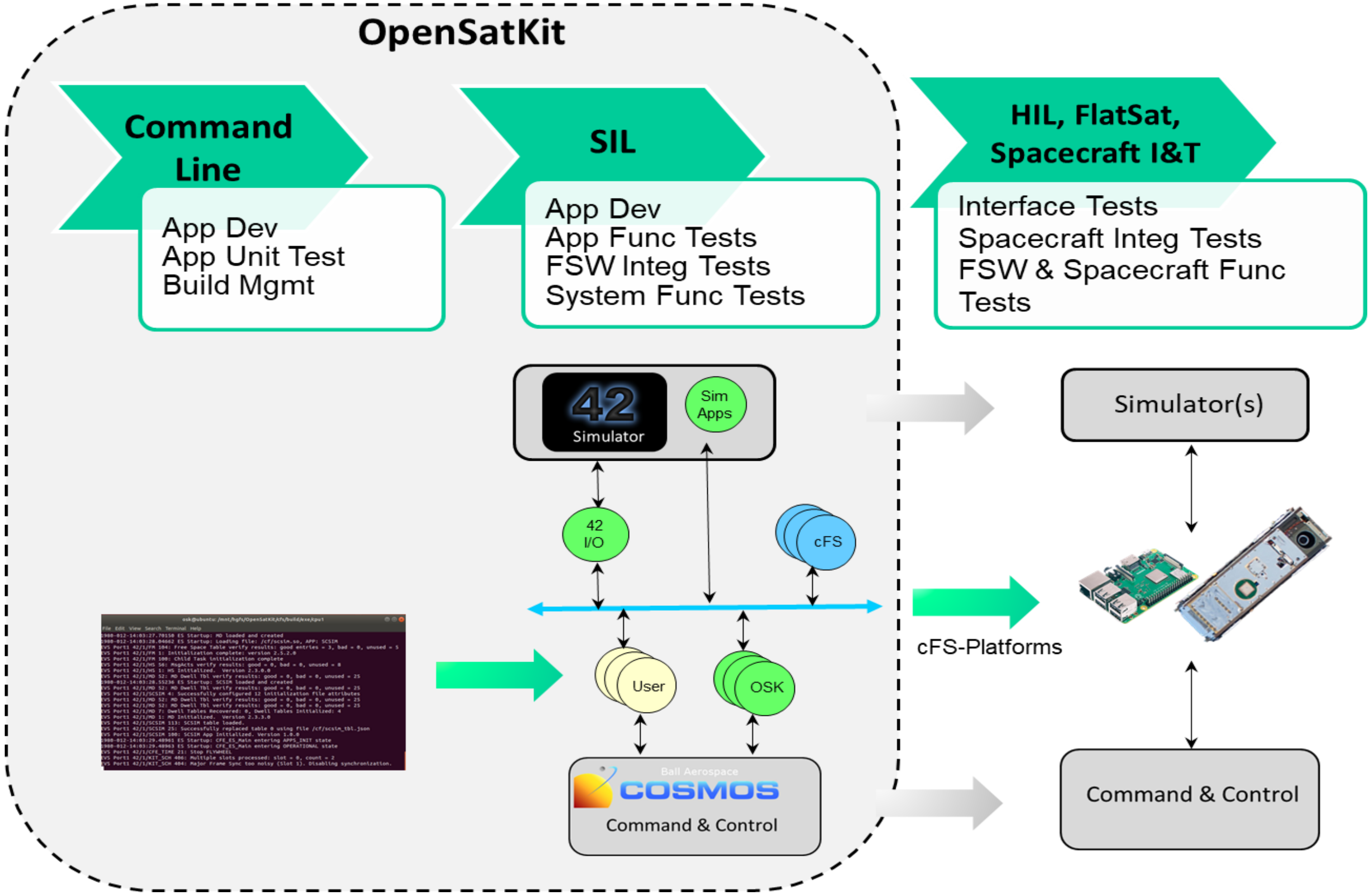
May 2021

Table of Contents

1. System Engineering Introduction
2. Apply System Engineering Development Processes
3. Apply System Engineering V&V Processes

Introduction (1 of 2)

Spacecraft FSW development phases



```
osk@ubuntu: /mnt/hgfs/OpenSatKit/cfs/build/exe/cpu1
File  Edit  View  Search  Terminal  Help
1980-012-14:03:27.70150 ES Startup: MD loaded and created
1980-012-14:03:28.04662 ES Startup: Loading file: /cf/scsim.so, APP: SCSIM
EVS Port1 42/1/FM 104: Free Space Table verify results: good entries = 3, bad = 0, unused = 5
EVS Port1 42/1/FM 1: Initialization complete: version 2.5.2.0
EVS Port1 42/1/FM 100: Child Task initialization complete
EVS Port1 42/1/HS 56: MsgActs verify results: good = 0, bad = 0, unused = 8
EVS Port1 42/1/HS 1: HS Initialized.  Version 2.3.0.0
EVS Port1 42/1/MD 52: MD Dwell Tbl verify results: good = 0, bad = 0, unused = 25
1980-012-14:03:28.55236 ES Startup: SCSIM loaded and created
EVS Port1 42/1/MD 52: MD Dwell Tbl verify results: good = 0, bad = 0, unused = 25
EVS Port1 42/1/SCSIM 4: Successfully configured 12 initialization file attributes
EVS Port1 42/1/MD 52: MD Dwell Tbl verify results: good = 0, bad = 0, unused = 25
EVS Port1 42/1/MD 52: MD Dwell Tbl verify results: good = 0, bad = 0, unused = 25
EVS Port1 42/1/MD 7: Dwell Tables Recovered: 0, Dwell Tables Initialized: 4
EVS Port1 42/1/MD 1: MD Initialized.  Version 2.3.3.0
EVS Port1 42/1/SCSIM 113: SCSIM table loaded.
EVS Port1 42/1/SCSIM 25: Successfully replaced table 0 using file /cf/scsim_tbl.json
EVS Port1 42/1/SCSIM 100: SCSIM App Initialized. Version 1.0.0
1980-012-14:03:29.48961 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:29.48963 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/KIT_SCH 406: Multiple slots processed: slot = 0, count = 2
EVS Port1 42/1/KIT_SCH 404: Major Frame Sync too noisy (Slot 1). Disabling synchronization.
```

Introduction (2 of 2)

- OSK supports integrating and testing cFS community apps, OSK apps, and user mission-specific apps into a functional FSW system that runs within OSK's software-in-the-loop (SIL) environment

- Nothing precludes OSK from being used in later mission lifecycle phases, however, creating the hardware in-the-loop (HIL) interfaces, developing simulators, and migrating ground system artifacts (if COSMOS is not used) are not covered by OSK.

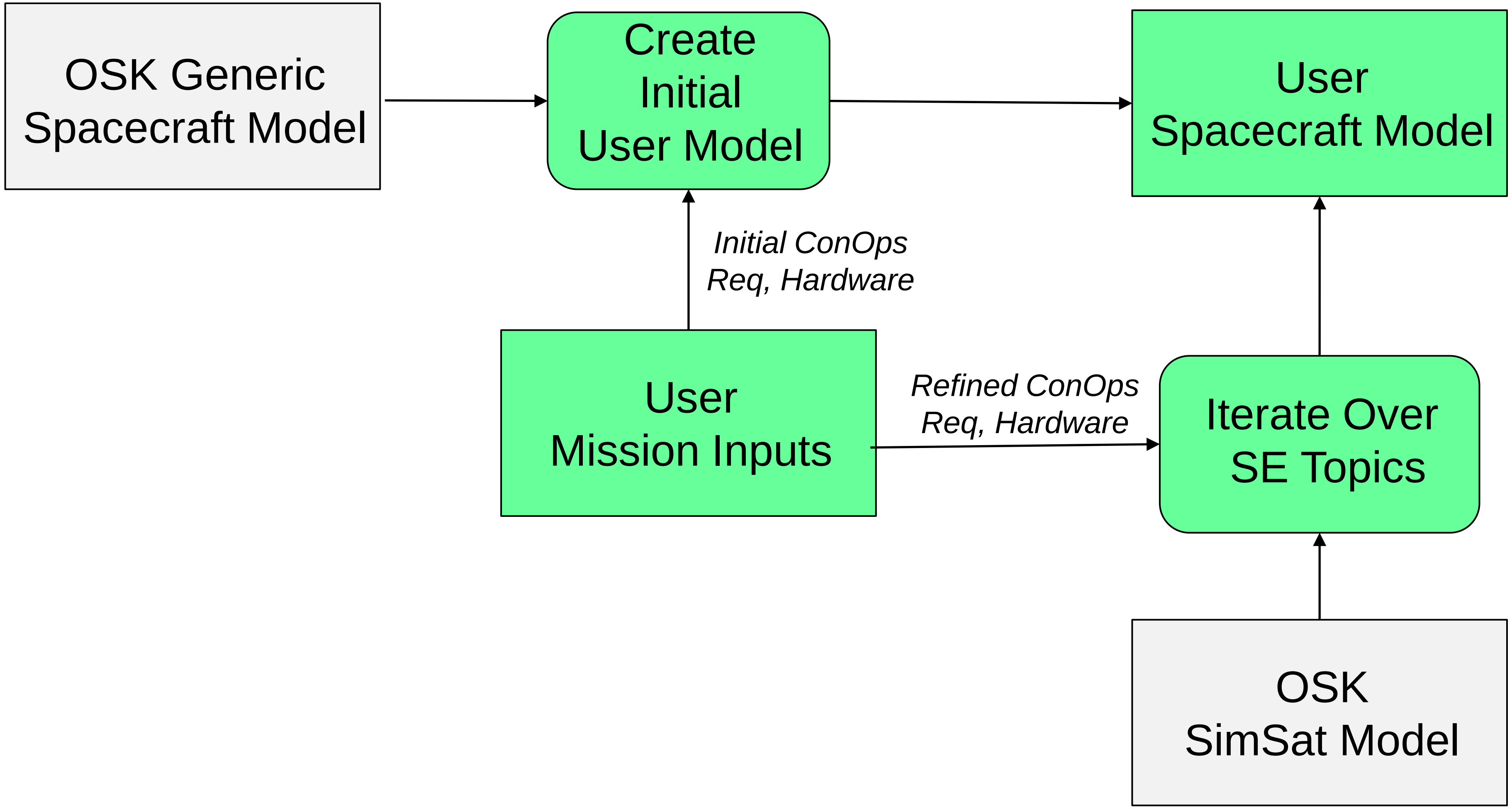
- These efforts are represented by the gray arrows. The green arrow pointing to the processor card is not within the OSK boundaries because porting the cFE to a hardware platform is not directly covered by OSK, however, a cFS community platform list https://github.com/OpenSatKit/cfs-platform-list is maintained by the OSK project and provides links to cFS porting resources.

- OSK is not required to develop cFS apps, however, note the following
  - You will eventually need a ground interface and test script environment
  - You can leave OSK's SimSat environment and develop new apps in a new mission or target or use OSK's Sandbox target

Mission FSW Systems Engineering Steps

1. **Create an initial app model from OSK's generic spacecraft model using mission concepts of operations, mission requirements, and the spacecraft hardware architecture often in the form of a block diagram**
   - Initial goal is to create a "good enough" architecture based on the maturity of the information at hand
   - Designing FSW is a very iterative process with top-down and bottom-up technical and non-technical forces at work
   - Trades are often made throughout the requirements analysis and spacecraft design phases that impact FSW. These forces are both technical and non-technical concerns may also influence decisions that impact the technical design
2. **Analyze OSK's SimSat mission app model to understand what capabilities exist within OSK**
3. **Work through system engineering topics and app groups to design new apps and understand how to configure cFS community apps**
4. **Work through spacecraft lifecycle to determine the need for**
   - Different versions of apps for different test environments
   - Simulation apps that can serve
5. **Determine how you want to use OSK in the spacecraft lifecycle**
   - Create OSK mission target
   - Plan migration to PIL and other environments
6. **If OSK will be used in a verification role then develop test artifacts as needed**
   - FSW validation should occur in a high-fidelity test environment

FSW System Engineering Process



OSK Generic Spacecraft Model

Create Initial User Model

User Spacecraft Model

Initial ConOps Req, Hardware

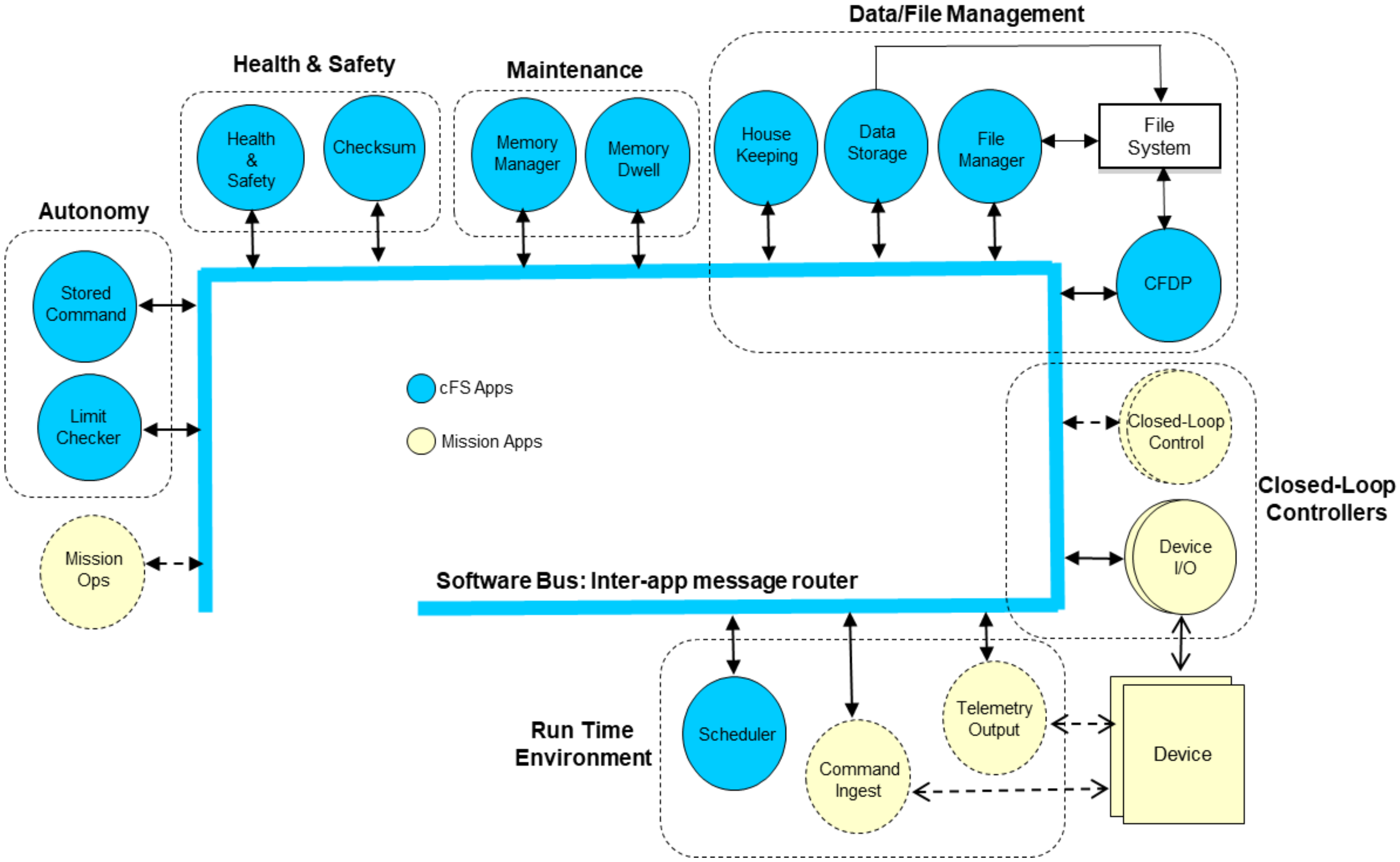
User Mission Inputs

Refined ConOps Req, Hardware

Iterate Over SE Topics

OSK SimSat Model

OSK Generic Spacecraft Model



OSK Generic Spacecraft Model

- **Show how generic model can be tailored for different missions. First focus on CubeSats, larger mission can extrapolate.**
- **Three main tailoring areas**
  - Device I/O
  - Closed loop control needs
  - Mission ops
- **Examples**
  - COTS vs inhouse ADCS
  - Payload control (closed vs open loop) and data management
  - Need for a mission manager app or ACS mode manager type app coupled with autonomy app group to achieve con ops
- **In order to work through the steps an example user mission is needed to show how to create a user model and then create a plan for how to migrate from SimSat to the user model needs**

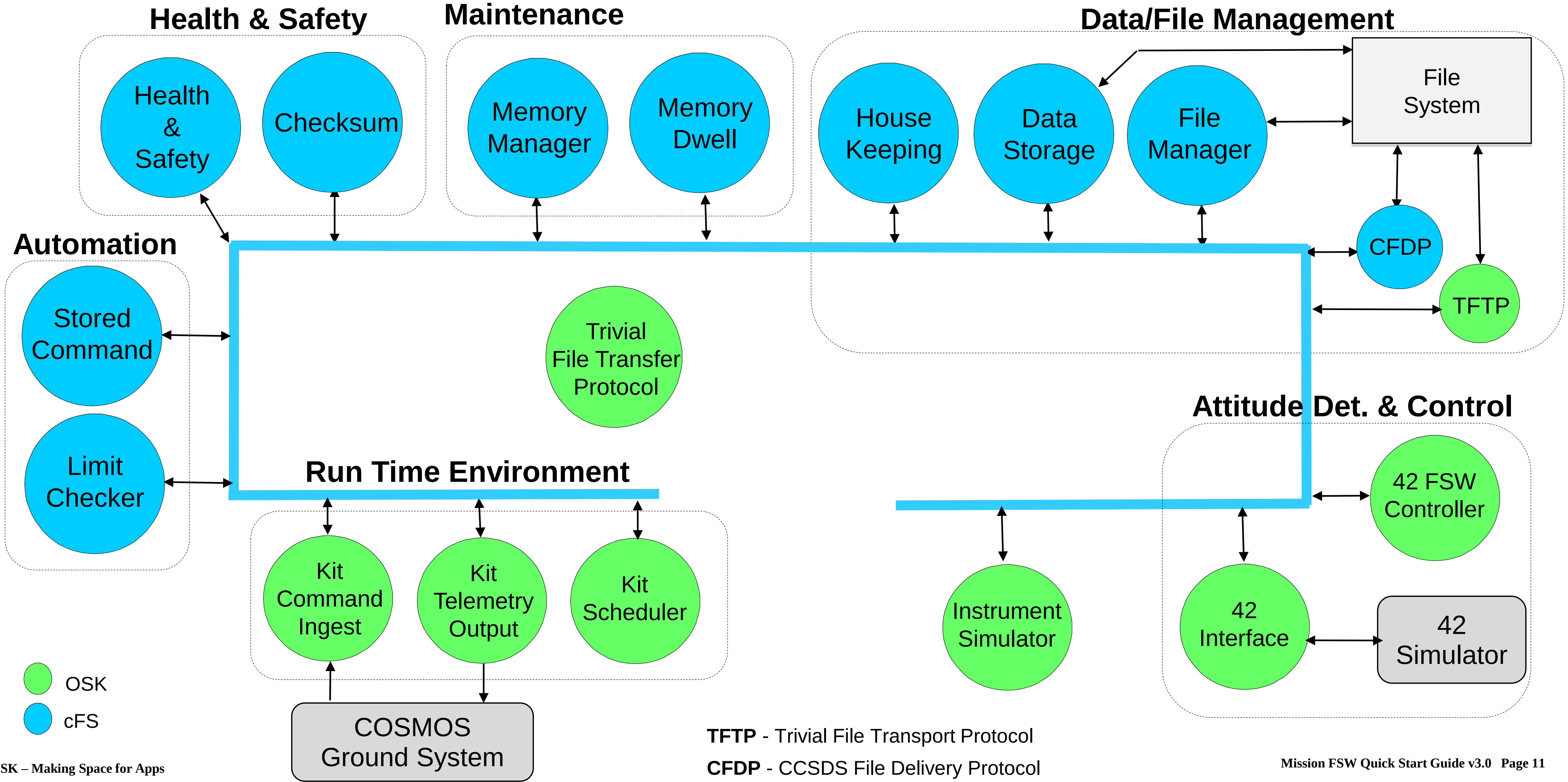
System Engineering Topics

- **System Engineering topics do not have a one-to-one correlation with app groups**
- **ConOps, mission requirements, and a hardware block diagram are required inputs**
  - This is an iterative process so reapply these inputs
  - Spacecraft lifecycle can be considered a process input and iteratively examined to determine what’s needed
- **Goal is to manage complexity by working through topics**
  - Topics are not 100% orthogonal (non-overlapping)
  - Create a ConOps and show how it impacts different apps groups
- **SE Topics**
  1. Device I/O
  2. Close-loop control
  3. Mission Ops & autonomy
  4. Data-File management
  5. Runtime Environment
  6. FDIR
  7. Interface & control apps
  8. Maintenance
  9. Time
  10. Parameters and configuration
- **V&V Topics**
  1. Unit tests
  2. App functional tests
  3. Integration tests
  4. System and ops tests

Simple Satellite (SimSat)

- **Default OSK app configuration is for a fictitious satellite called SimpleSat (SimSat)**
  - The cFS can be used for many different types of embedded systems. A spacecraft was chosen due to the increased usage of the cFS on CubeSats
- **SimSat provides a reference mission to provide context to**
  - Illustrate what applications are required and how they are configured and integrated as a system to meet the requirements
  - Demonstrate an example integration test script
  - Demonstrate an operational script
- **This does not include**
  - Porting SimSat to a new platform
  - Integrating hardware devices
- **SimSat is a**
  - Low Earth Orbit (LEO) satellite with one nadir-pointing science instrument
  - The instrument has
    - A detector that produces 10 bytes of data per second
    - A power the following sequence: Apply power, wait for instrument initialization (~20s), and command to enable science
  - The science team requires

SimSat Applications (1 of 4)



TFTP - Trivial File Transport Protocol

CFDP - CCSDS File Delivery Protocol

SimSat Applications (2 of 4)

- **The previous slide shows a cFS "bubble" chart where each app is a bubble and they communicate via messages on the software bus.**
  - The blue cFS apps are reusable open source apps that are available on https://github.com/nasa/xx where 'xx' is the abbreviated app name
  - The green OSK apps were written specifically for OSK
  - The external COSMOS and 42 interfaces use UDP and TCP respectively
- **Apps are designed to perform a dedicated function with clear interfaces and they operate in groups to achieve higher level mission objectives**

- **Runtime Environment Apps**
  - **Kit Command Ingest (KIT_CI)** receives CCSDS command packets from COSMOS and sends them on the Software Bus
  - **Kit Telemetry Output (KIT_TO)** reads CCSDS telemetry packets from the Software Bus and sends them to COSMOS
  - **Kit Scheduler (KIT_SCH)** contains tables that define when to send messages on the Software Bus
    - Apps can use these messages to perform synchronous activities, e.g. sending their housekeeping status packet

SimSat Applications (3 of 4)

- **Data/File Management**
  - **File Manager (FM)** provides a ground interface for performing common directory and file operations
  - **Data Storage (DS)** reads packets from the software bus and writes them to files according to table-defined
  - **Housekeeping (HK)** creates new telemetry packets from pieces of other telemetry packets. The new packets are written to the SB and can be stored and/or telemetered.
  - **Trivial File Transfer Protocol (TFTP)** transfers files between the flight and ground COSMOS. There's an open source CCSDS File Delivery Protocol (CFDP) app that will be added in a future release.

- **Autonomy**
  - **Limit Checker (LC)** monitors one or more telemetry values and start stored command relative time sequences (RTSs) in response to limit violations
  - **Stored Command (SC)** Provides services to execute preloaded, table-defined command sequences at predetermined absolute or relative time intervals

SimSat Applications (4 of 4)

- **Attitude Determination and Control Apps**
  - **42 Interface (I42)** manages a TCP/IP connection to 42 and transfers actuators/sensor packets to/from 42
  - **42 FSW (F42)** Implements the “ThreeAxisFsw” attitude control algorithm defined in 42
- **Maintenance**
  - **Memory Dwell (MD)** creates telemetry packets containing contents of memory location specified in dwell tables
  - **Memory Manager (MM)** provides read/write access to memory
- **Health & Safety**
  - **Checksum (CS)** monitors checksums across table-defined static code/data regions and reports errors
  - **Health & Safety (HS)** monitors table-defined application check-in and event messages and reporting errors and/or starting a RTS to address the issue

Apply the System Engineering Development Processes

UNDER CONSTRUCTION

System Engineering Topics

- **SE Topics**
  1. Device I/O
  2. Close-loop control
  3. Mission Ops & autonomy
  4. Data-File management
  5. Runtime Environment
  6. FDIR
  7. Interface & control apps
  8. Maintenance
  9. Time
  10. Parameters and configuration
- **V&V Topics**
  1. Unit tests
  2. App functional tests
  3. Integration tests
  4. System and ops tests

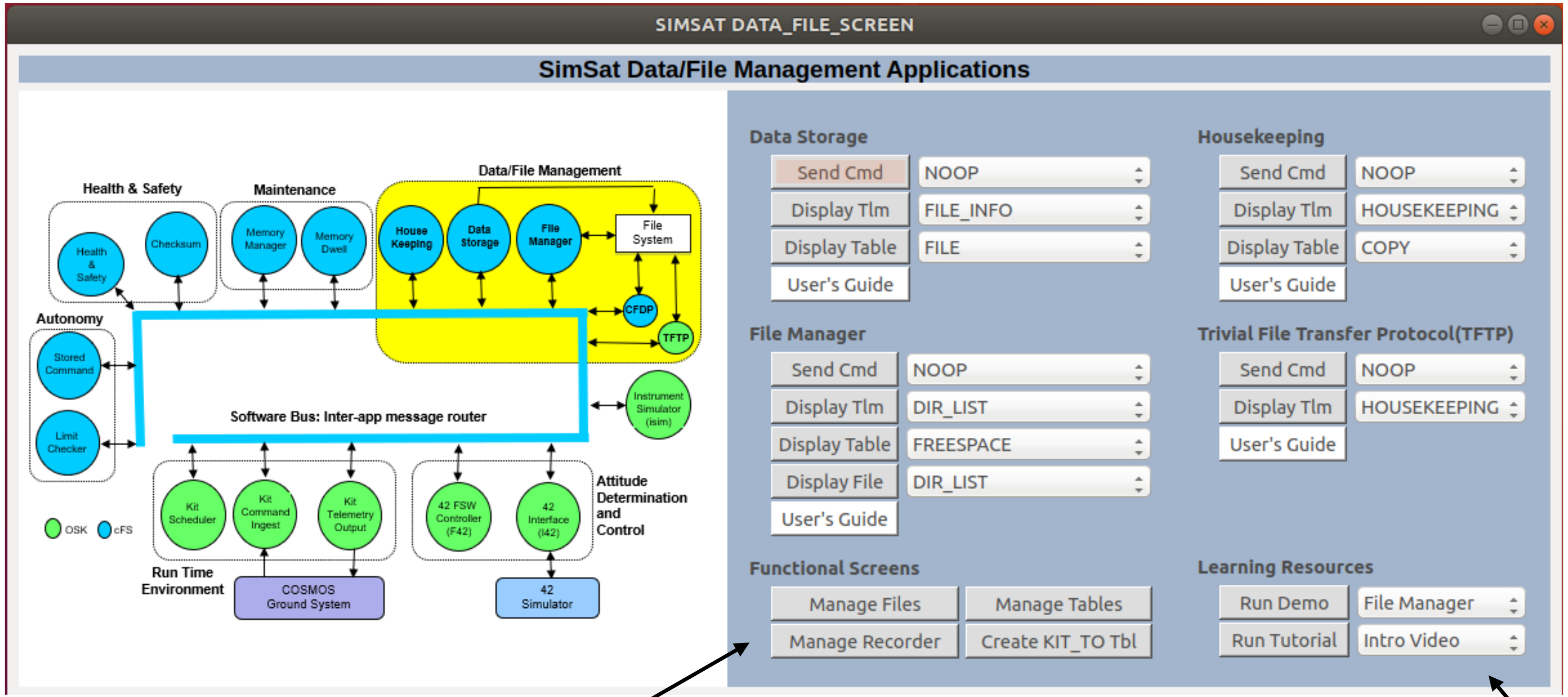
1. What data must be downlinked to meet mission goals?
2. What are the contact frequencies, durations, and data rates?
3. What parts of operations need to be automated?
4. What level of fault detection, isolation, and recovery (FDIR) is necessary?

- Bottom-up

1. What processor card is being used and is a realtime operating system required?
2. What device interfaces does the FSW need to manage and how are they connected?
3. Make versus buy decisions. Some top-down decisions impact bottom-up design.

SimSat App Screens

Each functional application group screen uses the following layout



Complete interface to each app

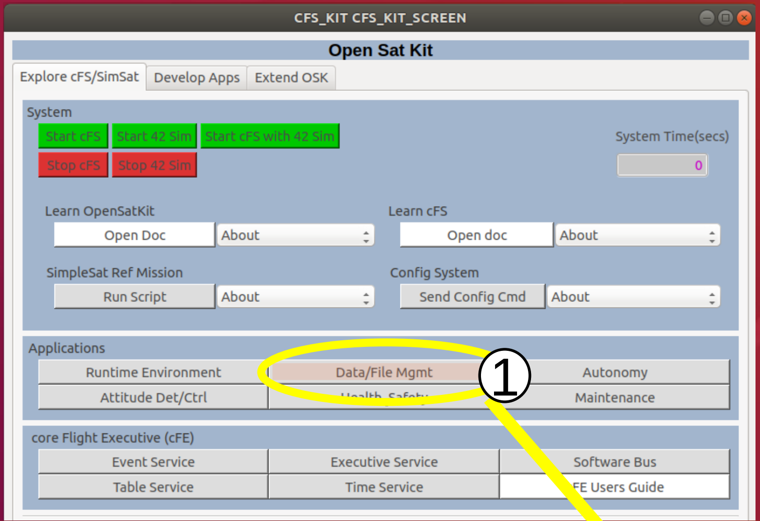
- All commands
- All telemetry packets
- "Display Table" – Dump, transfer and display table in COSMOS Table Manager
- "Display File" – Issue app's command to create a file, then transfer and display binary file in COSMOS Table Manager

Functional screens combine commands and telemetry from one or more apps that work together to perform a related tasks.

Launch videos, demos (pre-defined screen sequences) and tutorials (slides and/or scripts)

App Interactive Resources (1 of 2)

1. Launch Data/File Management Screen from OSK main screen
2. Access FM commands, telemetry, tables, files and users guide.



CFS_KIT CFS_KIT_SCREEN

Open Sat Kit

Explore cFS/SimSat | Develop Apps | Extend OSK

System

Start cFS | Start 42 Sim | Start cFS with 42 Sim

Stop cFS | Stop 42 Sim

System Time(secs): 0

Learn OpenSatKit: Open Doc | About

Learn cFS: Open doc | About

SimpleSat Ref Mission: Run Script | About

Config System: Send Config Cmd | About

Applications

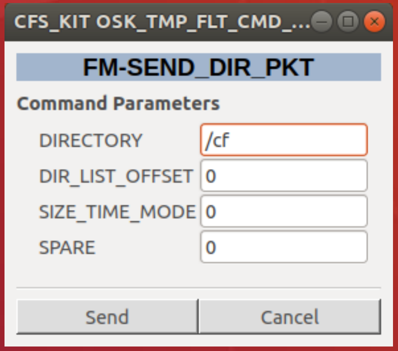
Runtime Environment | Data/File Mgmt | Autonomy

Attitude Det/Ctrl | Health Safety | Maintenance

core Flight Executive (cFE)

Event Service | Executive Service | Software Bus

Table Service | Time Service | cFE Users Guide



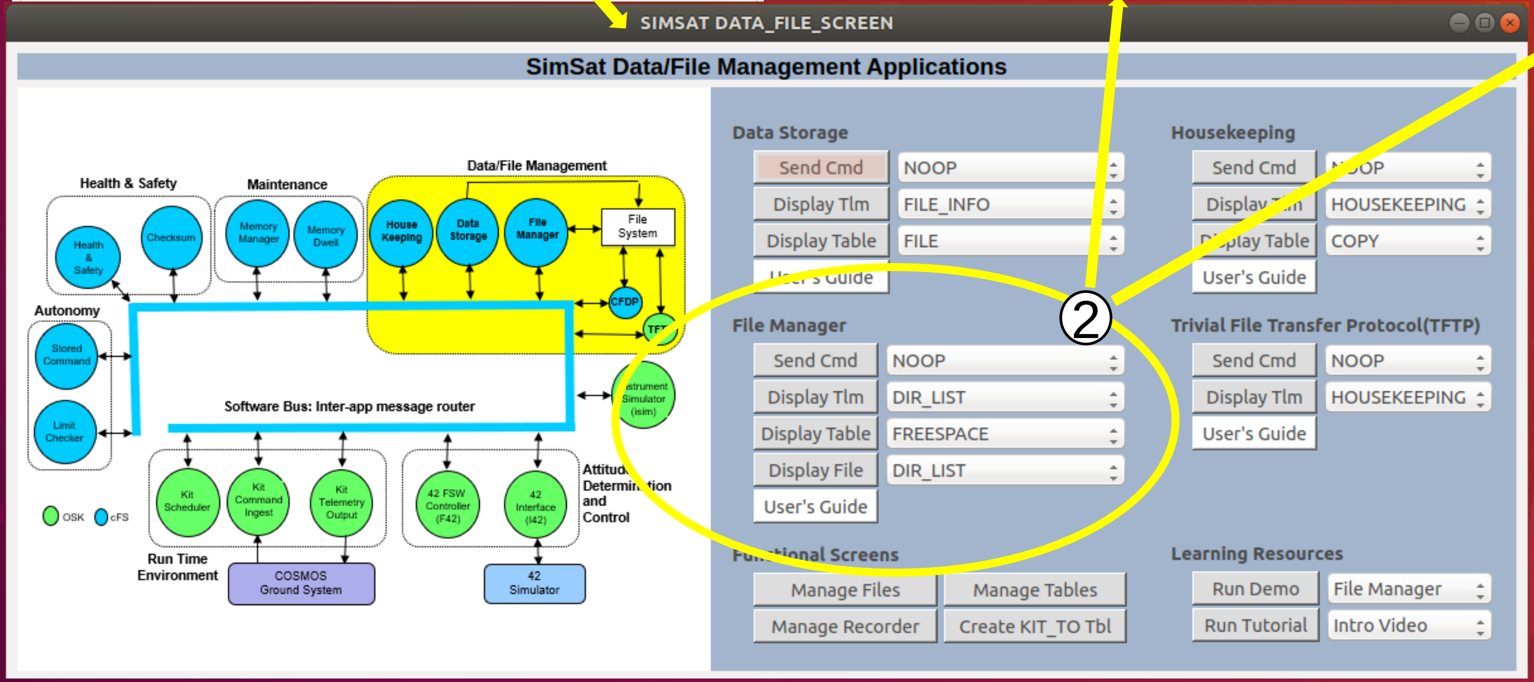
CFS_KIT OSK_TMP_FLT_CMD_...

FM-SEND_DIR_PKT

Command Parameters

DIRECTORY	/cf
DIR_LIST_OFFSET	0
SIZE_TIME_MODE	0
SPARE	0

Send | Cancel



SIMSAT DATA_FILE_SCREEN

SimSat Data/File Management Applications

Data Storage

Send Cmd: NOOP

Display Tlm: FILE_INFO

Display Table: FILE

User's Guide

Housekeeping

Send Cmd: NOOP

Display Tlm: HOUSEKEEPING

Display Table: COPY

User's Guide

File Manager

Send Cmd: NOOP

Display Tlm: DIR_LIST

Display Table: FREESPACE

Display File: DIR_LIST

User's Guide

Trivial File Transfer Protocol(TFTP)

Send Cmd: NOOP

Display Tlm: HOUSEKEEPING

User's Guide

Functional Screens

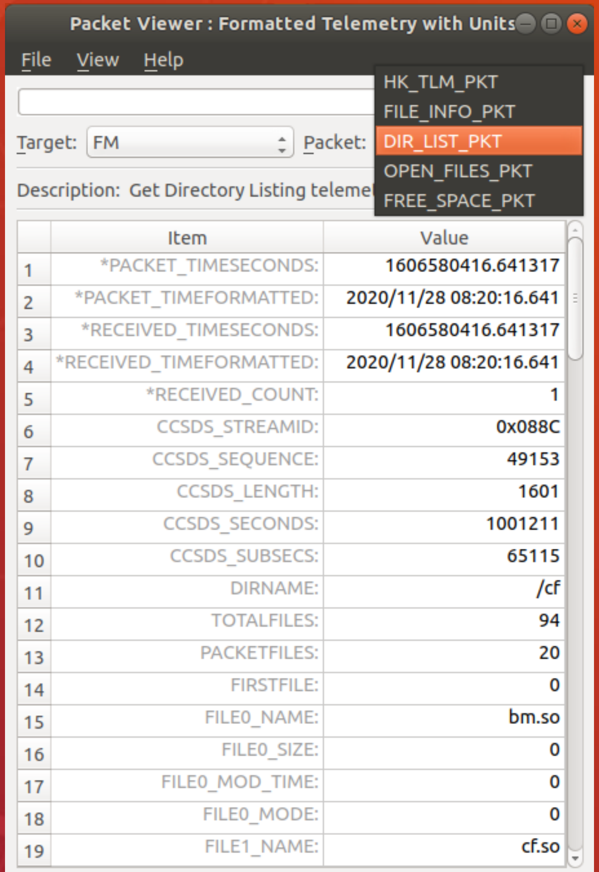
Manage Files | Manage Tables

Manage Recorder | Create KIT_TO Tbl

Learning Resources

Run Demo: File Manager

Run Tutorial: Intro Video



Packet Viewer : Formatted Telemetry with Units

File View Help

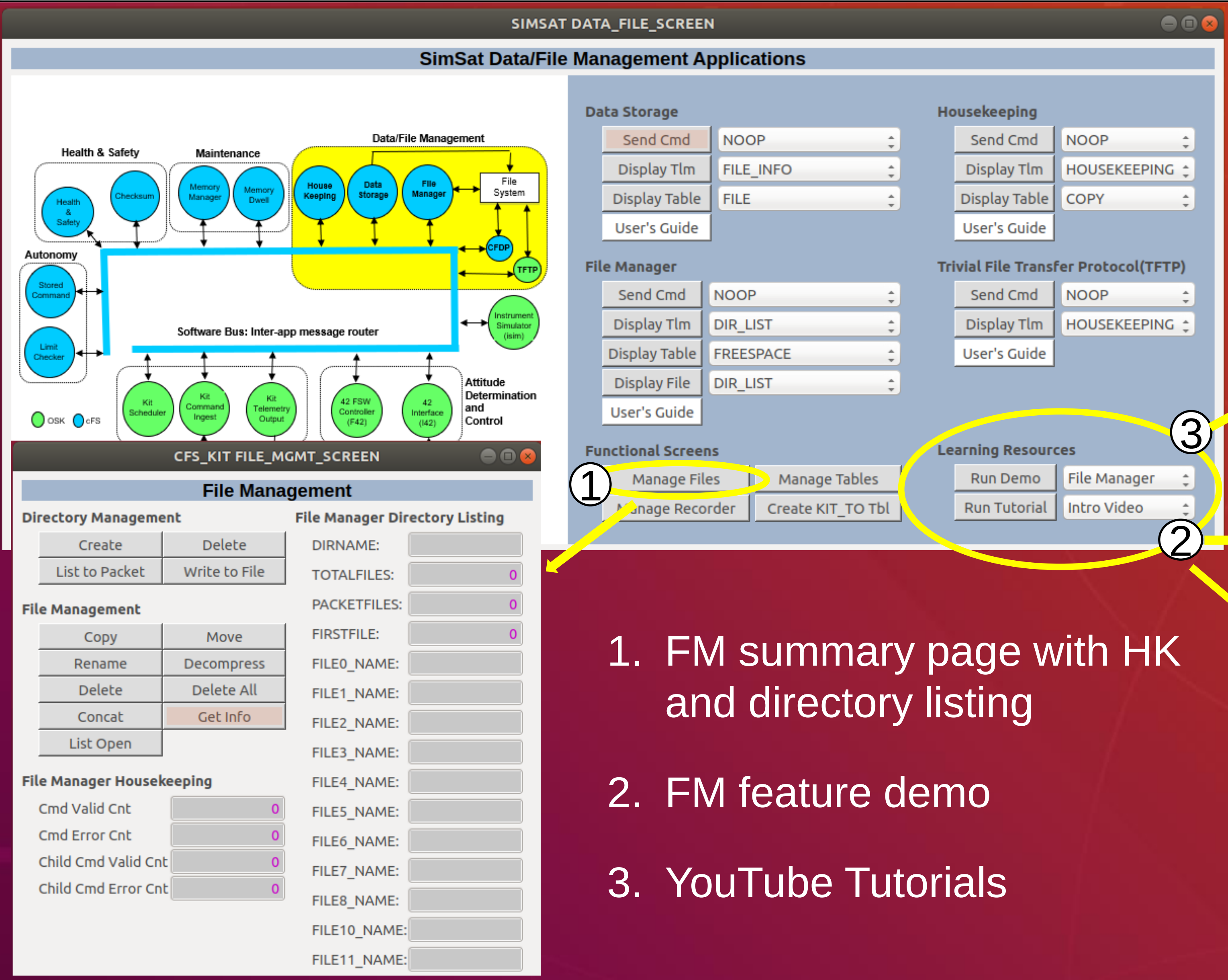
Target: FM Packet: DIR_LIST_PKT

HK_TLM_PKT
FILE_INFO_PKT
DIR_LIST_PKT
OPEN_FILES_PKT
FREE_SPACE_PKT

Description: Get Directory Listing teleme

	Item	Value
1	*PACKET_TIMESECONDS:	1606580416.641317
2	*PACKET_TIMEFORMATTED:	2020/11/28 08:20:16.641
3	*RECEIVED_TIMESECONDS:	1606580416.641317
4	*RECEIVED_TIMEFORMATTED:	2020/11/28 08:20:16.641
5	*RECEIVED_COUNT:	1
6	CCSDS_STREAMID:	0x088C
7	CCSDS_SEQUENCE:	49153
8	CCSDS_LENGTH:	1601
9	CCSDS_SECONDS:	1001211
10	CCSDS_SUBSECS:	65115
11	DIRNAME:	/cf
12	TOTALFILES:	94
13	PACKETFILES:	20
14	FIRSTFILE:	0
15	FILE0_NAME:	bm.so
16	FILE0_SIZE:	0
17	FILE0_MOD_TIME:	0
18	FILE0_MODE:	0
19	FILE1_NAME:	cf.so

App Interactive Resources (2 of 2)



SIMSAT DATA_FILE_SCREEN

SimSat Data/File Management Applications

Data Storage

Send Cmd | NOOP
Display Tlm | FILE_INFO
Display Table | FILE
User's Guide

Housekeeping

Send Cmd | NOOP
Display Tlm | HOUSEKEEPING
Display Table | COPY
User's Guide

File Manager

Send Cmd | NOOP
Display Tlm | DIR_LIST
Display Table | FREESPACE
Display File | DIR_LIST
User's Guide

Trivial File Transfer Protocol(TFTP)

Send Cmd | NOOP
Display Tlm | HOUSEKEEPING
User's Guide

Functional Screens

Manage Files | Manage Tables
Manage Recorder | Create KIT_TO Tbl

Learning Resources

Run Demo | File Manager
Run Tutorial | Intro Video

CFS_KIT FILE_MGMT_SCREEN

File Management

Directory Management

Create | Delete
List to Packet | Write to File

File Management

Copy | Move
Rename | Decompress
Delete | Delete All
Concat | Get Info
List Open

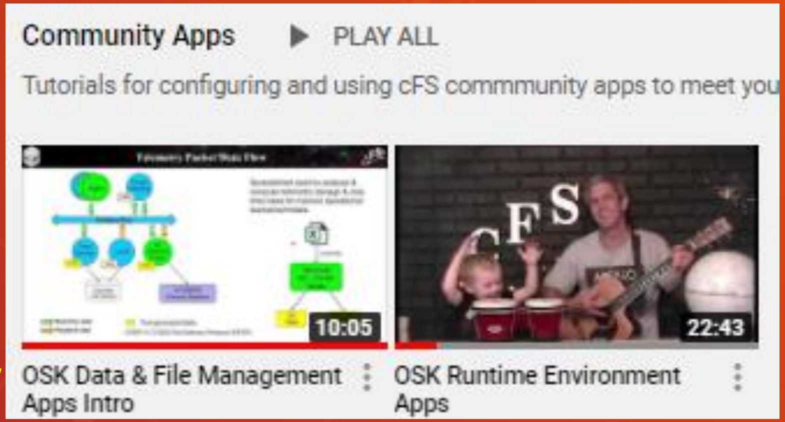
File Manager Housekeeping

Cmd Valid Cnt: 0
Cmd Error Cnt: 0
Child Cmd Valid Cnt: 0
Child Cmd Error Cnt: 0

File Manager Directory Listing

DIRNAME:
TOTALFILES: 0
PACKETFILES: 0
FIRSTFILE: 0
FILE0_NAME:
FILE1_NAME:
FILE2_NAME:
FILE3_NAME:
FILE4_NAME:
FILE5_NAME:
FILE6_NAME:
FILE7_NAME:
FILE8_NAME:
FILE10_NAME:
FILE11_NAME:

1. FM summary page with HK and directory listing
2. FM feature demo
3. YouTube Tutorials

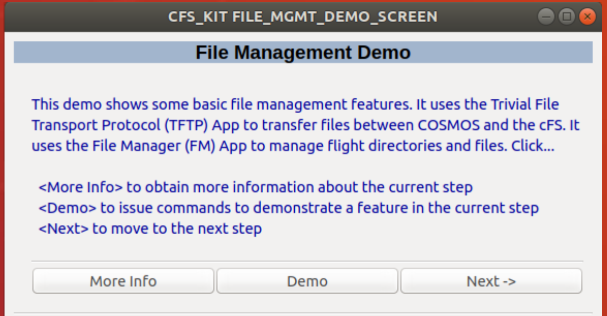


Community Apps ▶ PLAY ALL

Tutorials for configuring and using cFS commmunity apps to meet you

OSK Data & File Management Apps Intro — 10:05

OSK Runtime Environment Apps — 22:43



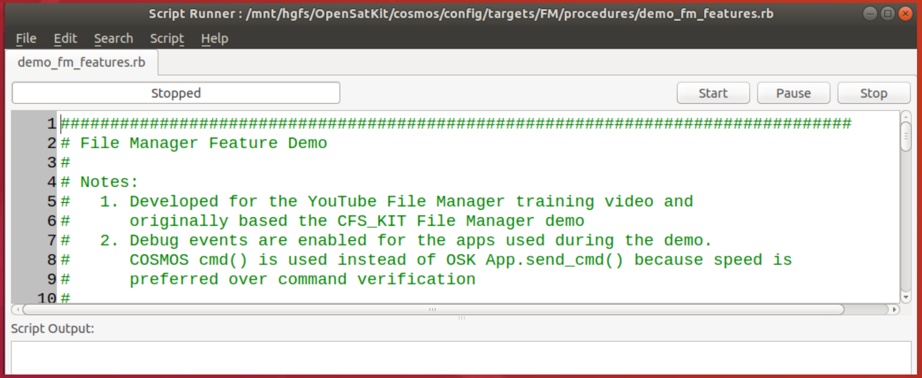
CFS_KIT FILE_MGMT_DEMO_SCREEN

File Management Demo

This demo shows some basic file management features. It uses the Trivial File Transfer Protocol (TFTP) App to transfer files between COSMOS and the cFS. It uses the File Manager (FM) App to manage flight directories and files. Click...

<More Info> to obtain more information about the current step
<Demo> to issue commands to demonstrate a feature in the current step
<Next> to move to the next step

More Info | Demo | Next ->



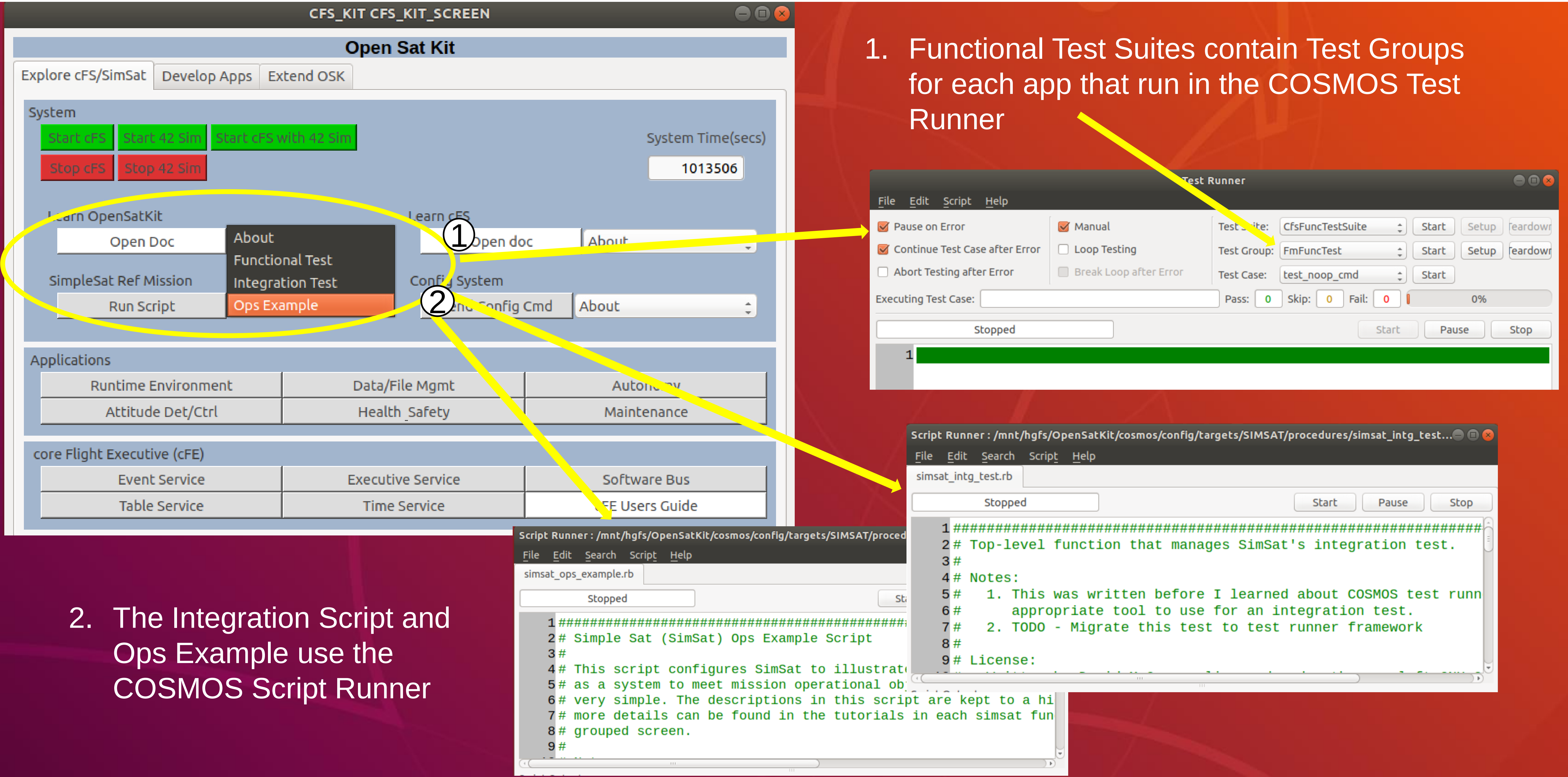
Script Runner : /mnt/hgfs/OpenSatKit/cosmos/config/targets/FM/procedures/demo_fm_features.rb

Stopped

```
################################################################################
# File Manager Feature Demo
#
# Notes:
#   1. Developed for the YouTube File Manager training video and
#      originally based the CFS_KIT File Manager demo
#   2. Debug events are enabled for the apps used during the demo.
#      COSMOS cmd() is used instead of OSK App.send_cmd() because speed is
#      preferred over command verification
#
```

Script Output:

App System Scripting Resources



```
#####################################################################
# Top-level function that manages SimSat's integration test.
#
# Notes:
#   1. This was written before I learned about COSMOS test runn
#      appropriate tool to use for an integration test.
#   2. TODO - Migrate this test to test runner framework
#
# License:
```

```
##############################################
# Simple Sat (SimSat) Ops Example Script
#
# This script configures SimSat to illustrate
# as a system to meet mission operational ob
# very simple. The descriptions in this script are kept to a hi
# more details can be found in the tutorials in each simsat fun
# grouped screen.
#
```

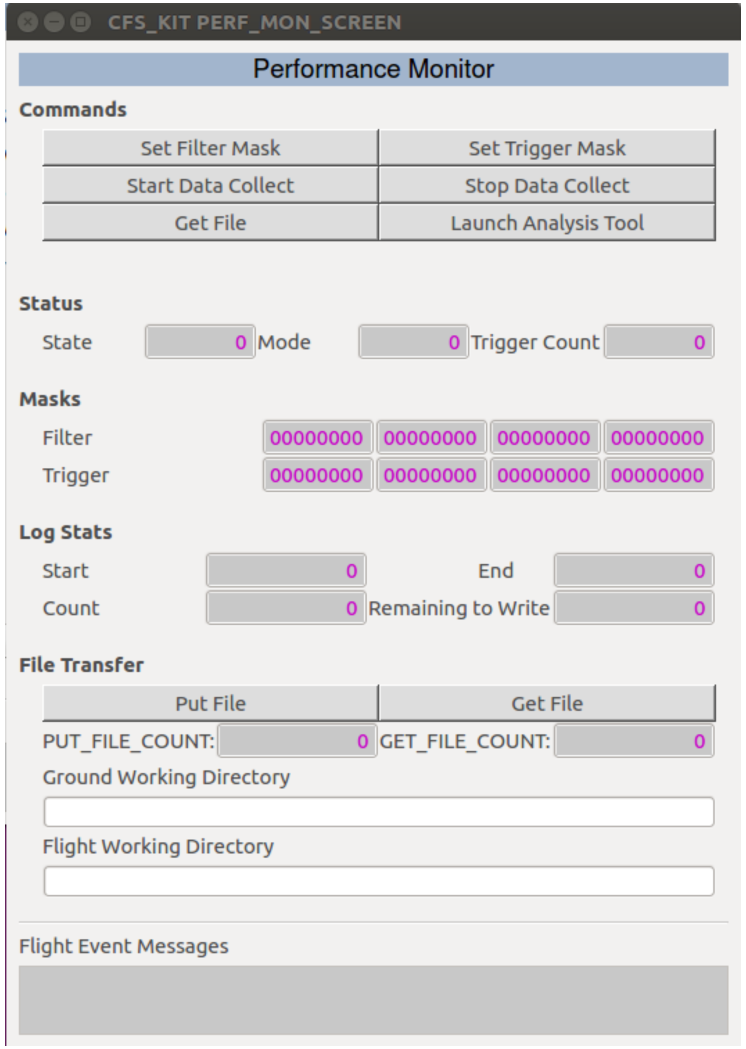
1. Functional Test Suites contain Test Groups for each app that run in the COSMOS Test Runner

2. The Integration Script and Ops Example use the COSMOS Script Runner

Tune, Verify, & Validate

UNDER CONSTRUCTION

Performance Monitor Tool



CFS_KIT PERF_MON_SCREEN

Performance Monitor

Commands

Set Filter Mask | Set Trigger Mask
Start Data Collect | Stop Data Collect
Get File | Launch Analysis Tool

Status

State: 0 Mode: 0 Trigger Count: 0

Masks

Filter: 00000000 00000000 00000000 00000000

Trigger: 00000000 00000000 00000000 00000000

Log Stats

Start: 0 End: 0

Count: 0 Remaining to Write: 0

File Transfer

Put File | Get File

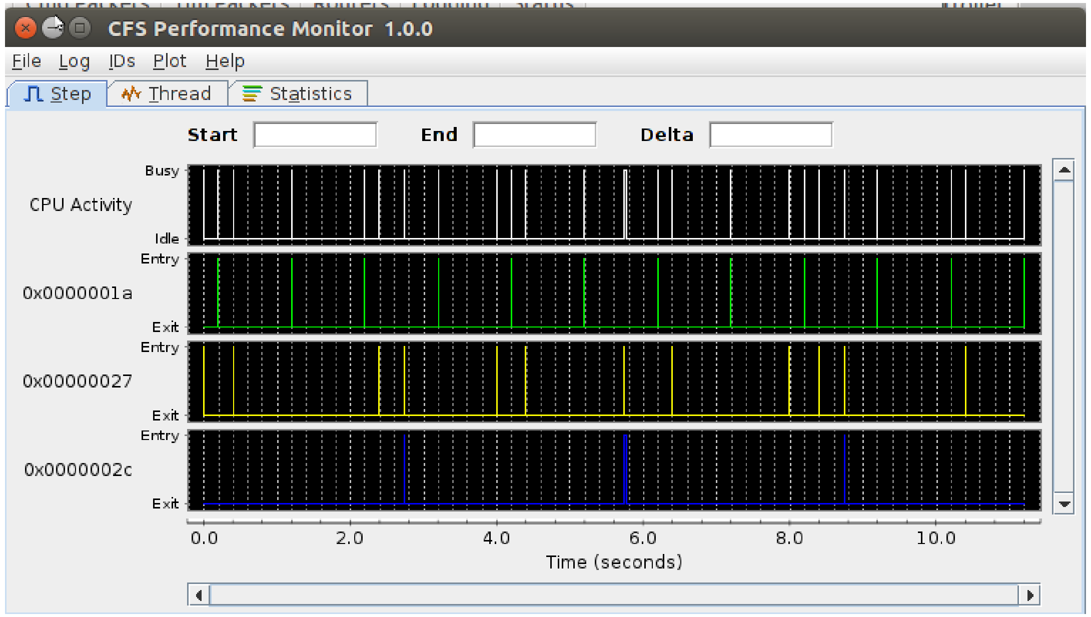
PUT_FILE_COUNT: 0 GET_FILE_COUNT: 0

Ground Working Directory

Flight Working Directory

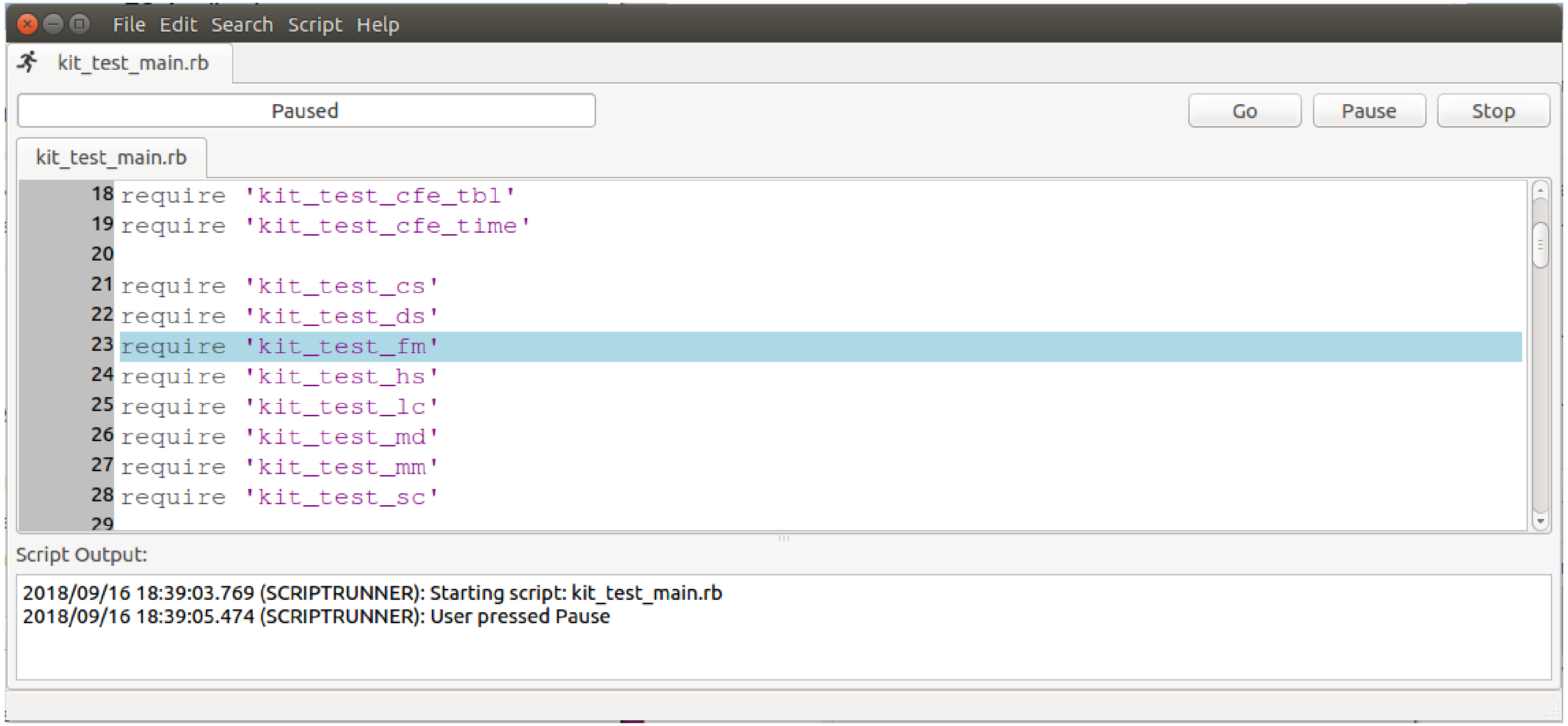
Flight Event Messages

- Capture FSW performance data using screen
- Download file and <Launch Analysis Tool>



CFS Performance Monitor 1.0.0

SimSat Integration Script



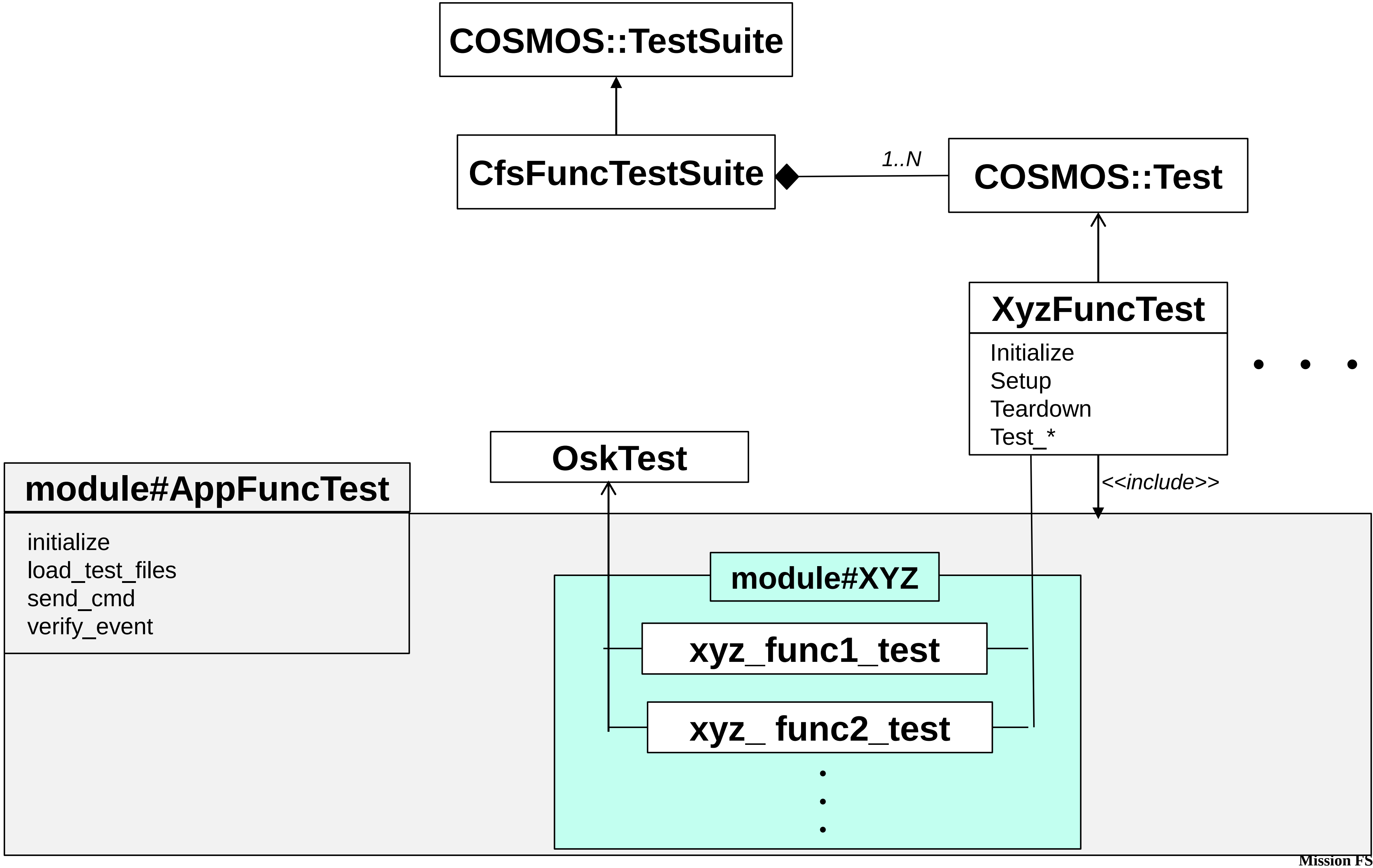
```
require 'kit_test_cfe_tbl'
require 'kit_test_cfe_time'

require 'kit_test_cs'
require 'kit_test_ds'
require 'kit_test_fm'
require 'kit_test_hs'
require 'kit_test_lc'
require 'kit_test_md'
require 'kit_test_mm'
require 'kit_test_sc'
```

```
2018/09/16 18:39:03.769 (SCRIPTRUNNER): Starting script: kit_test_main.rb
2018/09/16 18:39:05.474 (SCRIPTRUNNER): User pressed Pause
```

- **Runs test script using Script Runner**
- **Issues Noop command to every application and verifies telemetry response**

Functional Test Scripts



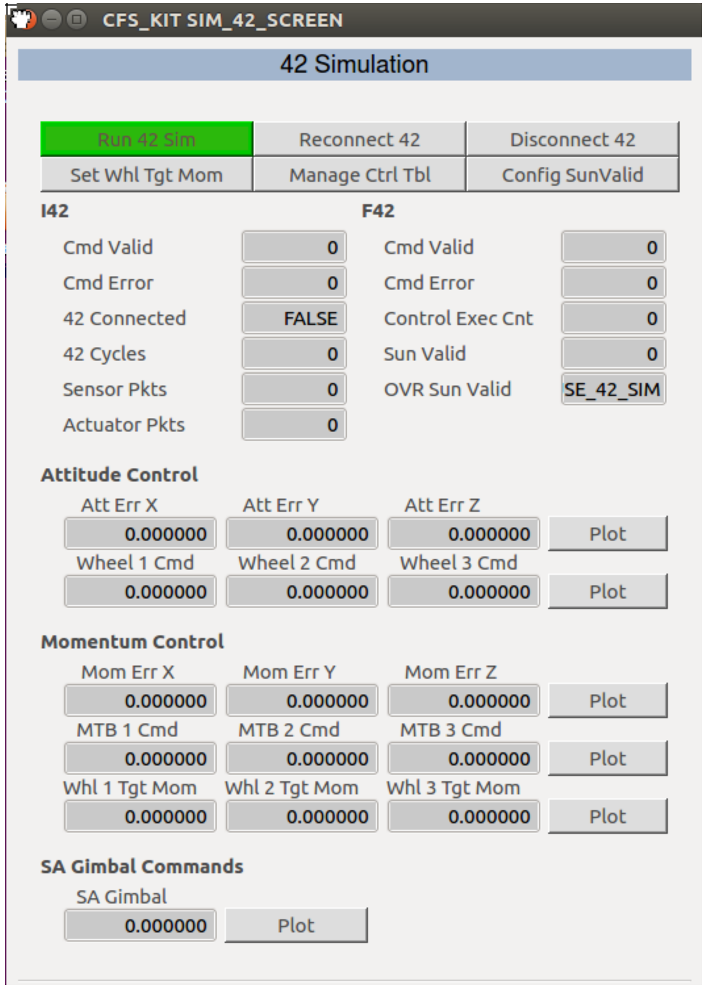
SimSat Operation Script Example

Running with 42 Simulator

UNDER CONSTRUCTION Needs updates since v2.4

Merge with apply systems engineering processes

Starting the Simulation



42 Simulation

I42

Cmd Valid 0
Cmd Error 0
42 Connected FALSE
42 Cycles 0
Sensor Pkts 0
Actuator Pkts 0

F42

Cmd Valid 0
Cmd Error 0
Control Exec Cnt 0
Sun Valid 0
OVR Sun Valid SE_42_SIM

Attitude Control

Att Err X	Att Err Y	Att Err Z
0.000000	0.000000	0.000000
Wheel 1 Cmd	Wheel 2 Cmd	Wheel 3 Cmd
0.000000	0.000000	0.000000

Momentum Control

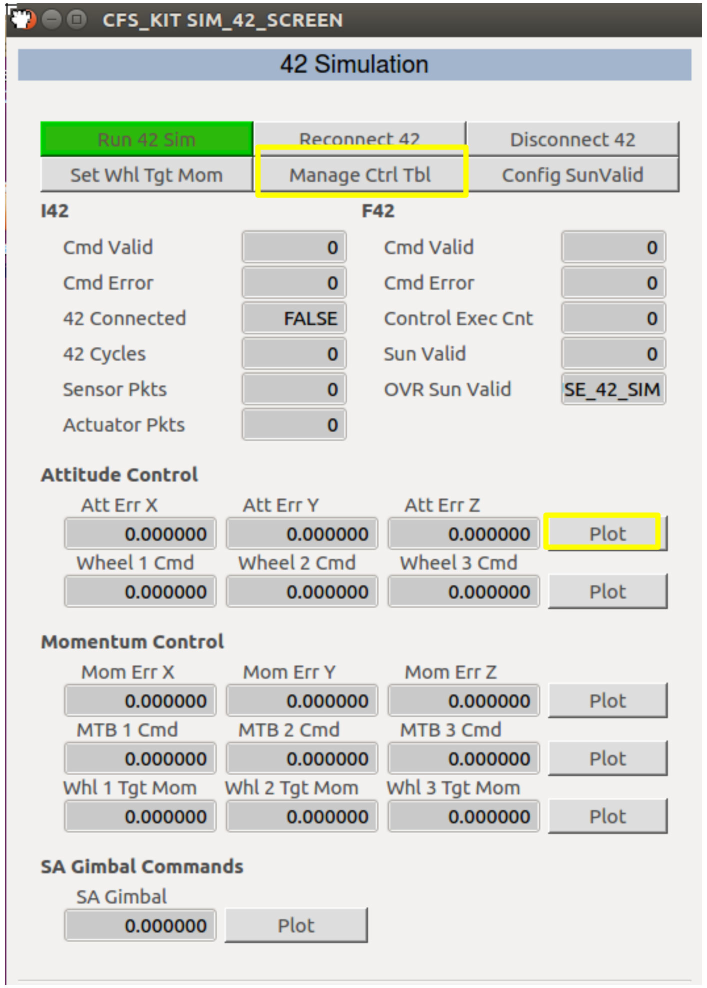
Mom Err X	Mom Err Y	Mom Err Z
0.000000	0.000000	0.000000
MTB 1 Cmd	MTB 2 Cmd	MTB 3 Cmd
0.000000	0.000000	0.000000
Whl 1 Tgt Mom	Whl 2 Tgt Mom	Whl 3 Tgt Mom
0.000000	0.000000	0.000000

SA Gimbal Commands

SA Gimbal 0.000000

- Select <*Run 42 Sim*> which will start the 42 simulator in a new terminal window.
- The 42 configuration files used in the simulation are located in directory *OpenSatKit/42/OSK*
- The simulation takes a while to initialize

Preparing 42 Simulation



42 Simulation

I42		F42	
Cmd Valid	0	Cmd Valid	0
Cmd Error	0	Cmd Error	0
42 Connected	FALSE	Control Exec Cnt	0
42 Cycles	0	Sun Valid	0
Sensor Pkts	0	OVR Sun Valid	SE_42_SIM
Actuator Pkts	0		

Attitude Control

Att Err X	Att Err Y	Att Err Z
0.000000	0.000000	0.000000
Wheel 1 Cmd	Wheel 2 Cmd	Wheel 3 Cmd
0.000000	0.000000	0.000000

Momentum Control

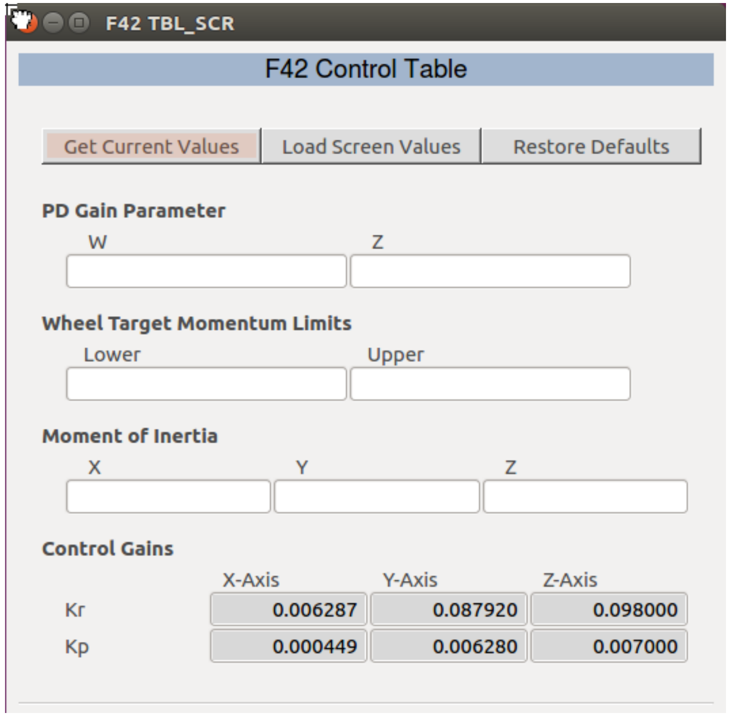
Mom Err X	Mom Err Y	Mom Err Z
0.000000	0.000000	0.000000
MTB 1 Cmd	MTB 2 Cmd	MTB 3 Cmd
0.000000	0.000000	0.000000
Whl 1 Tgt Mom	Whl 2 Tgt Mom	Whl 3 Tgt Mom
0.000000	0.000000	0.000000

SA Gimbal Commands

SA Gimbal: 0.000000

- From the kit main page on the previous slide select <42 Simulator> and the screen to the left will appear.
- The 2nd row of buttons allow you to change the behavior of the control algorithms running in the FSW and are described on the next slides
- Before running the sim you will open some additional windows that will be used for your class exercise
  - Manage Control Table
  - Plot Attitude Errors

Managing Control Table



F42 Control Table

PD Gain Parameter

W	Z

Wheel Target Momentum Limits

Lower	Upper

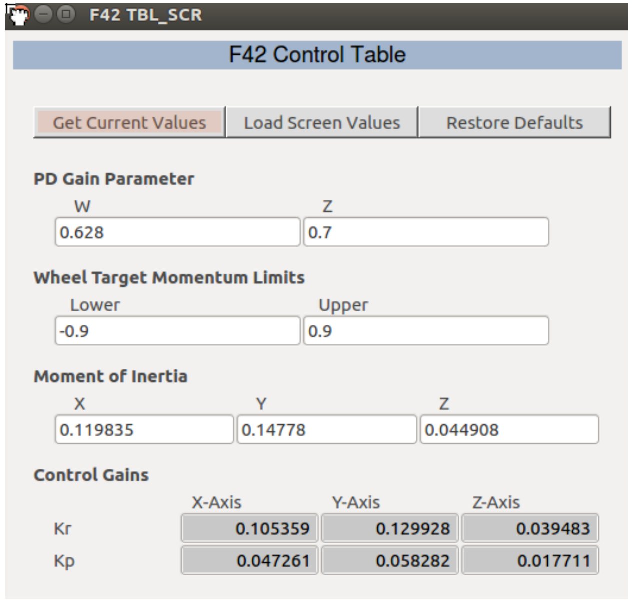
Moment of Inertia

X	Y	Z

Control Gains

	X-Axis	Y-Axis	Z-Axis
Kr	0.006287	0.087920	0.098000
Kp	0.000449	0.006280	0.007000

- Selecting <*Manage Control Table*> on the 42 Sim screen produces the screen to the left.
- Select <*Get Current Values*> and it will populate the screen with the current control table values. This takes a little time because it is transferring a file from flight to ground
- Edit the screen as desired and click <*Load Screen Values*> to replace the current control table values
- The defaults can be restored by clicking <*Restore Defaults*>



F42 Control Table

PD Gain Parameter

W	Z
0.628	0.7

Wheel Target Momentum Limits

Lower	Upper
-0.9	0.9

Moment of Inertia

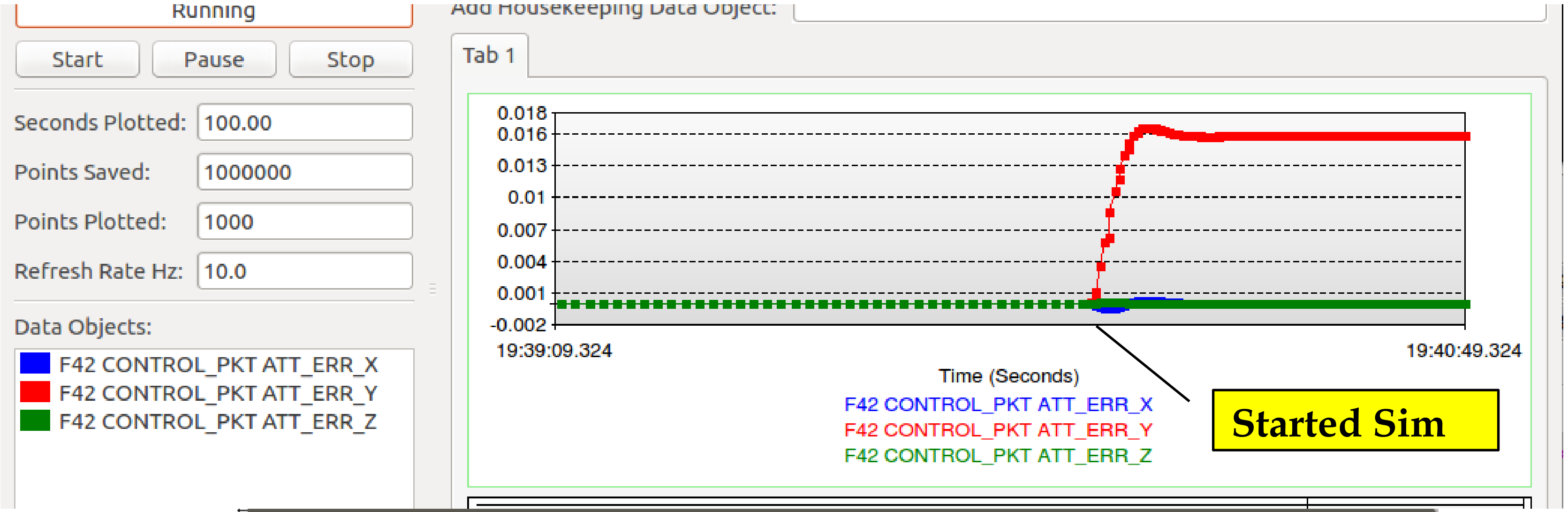
X	Y	Z
0.119835	0.14778	0.044908

Control Gains

	X-Axis	Y-Axis	Z-Axis
Kr	0.105359	0.129928	0.039483
Kp	0.047261	0.058282	0.017711

Plot Attitude Errors

- Selecting <Plot> button next to the attitude errors produces the screen below



Running

Seconds Plotted: 100.00

Points Saved: 1000000

Points Plotted: 1000

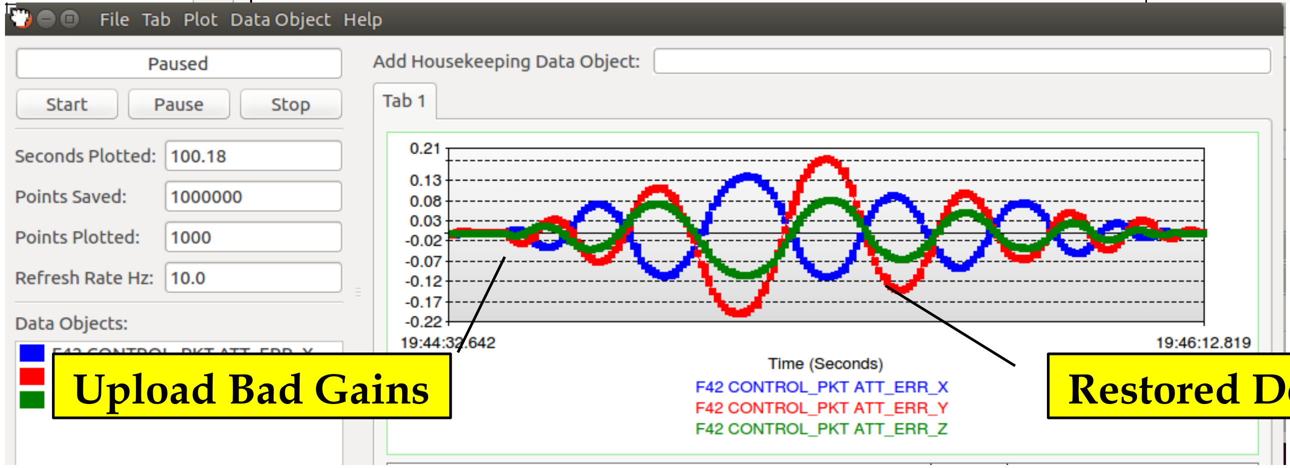
Refresh Rate Hz: 10.0

Data Objects:

- 🟦 F42 CONTROL_PKT ATT_ERR_X
- 🟥 F42 CONTROL_PKT ATT_ERR_Y
- 🟩 F42 CONTROL_PKT ATT_ERR_Z

Start
Pause
Stop
Add Housekeeping Data Object:
Tab 1
0.018
0.016
0.013
0.01
0.007
0.004
0.001
-0.002
19:39:09.324
19:40:49.324
Time (Seconds)
F42 CONTROL_PKT ATT_ERR_X
F42 CONTROL_PKT ATT_ERR_Y
F42 CONTROL_PKT ATT_ERR_Z

Started Sim



Paused

Seconds Plotted: 100.18

Points Saved: 1000000

Points Plotted: 1000

Refresh Rate Hz: 10.0

Data Objects:

Upload Bad Gains

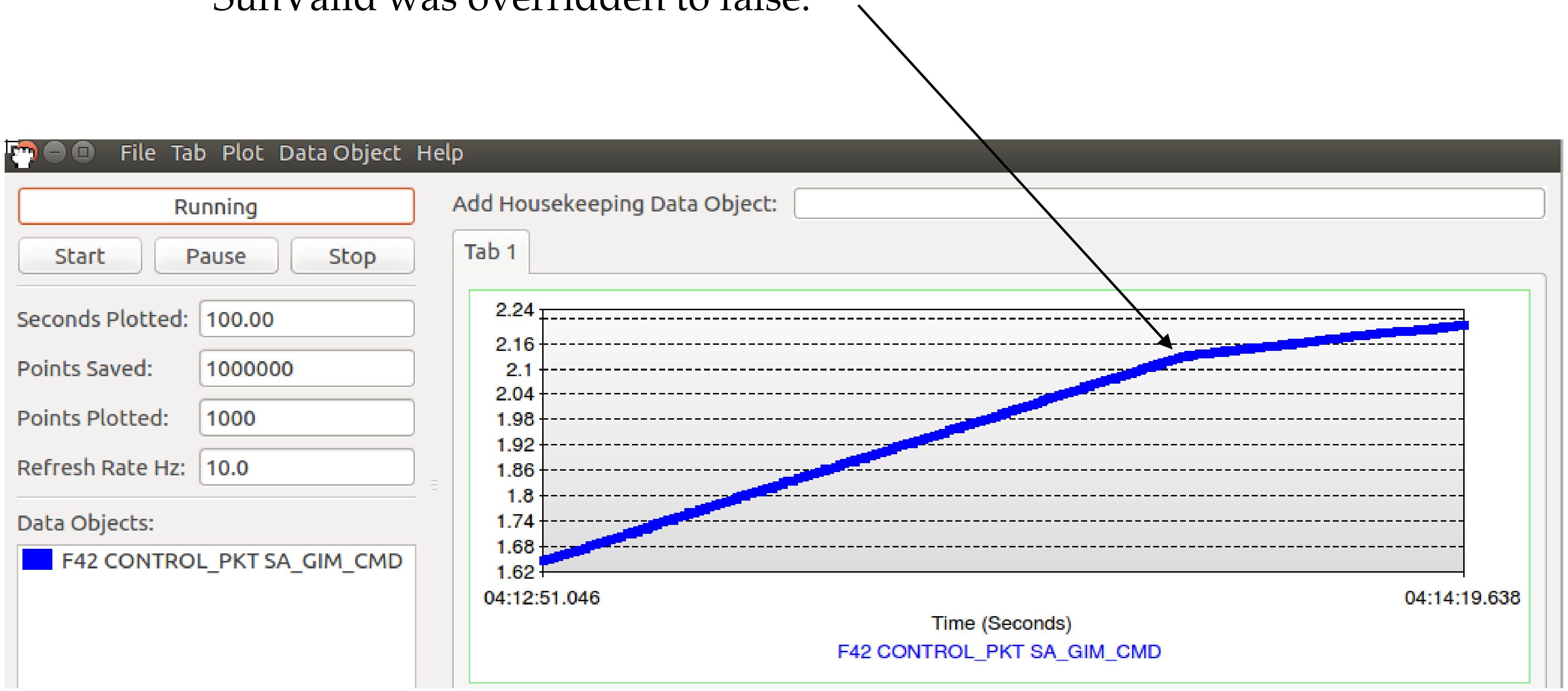
Restored Defaults

Additional Configuration Options

- **The kit includes two additional configuration options that can be manipulated**
  1. Wheel target Momentum
  2. Sun Valid Configuration

Configure Sun Valid

- Selecting <*Config SunValid*> to override the current sun valid flag
- The plot below shows gimbal command
  - The linear portion had a valid sun and the bend occurred when the SunValid was overridden to false.



Add Housekeeping Data Object:

Seconds Plotted: 100.00

Points Saved: 1000000

Points Plotted: 1000

Refresh Rate Hz: 10.0

Data Objects:

🟦 F42 CONTROL_PKT SA_GIM_CMD

Sim Termination

1. Click ***<Disconnect 42>*** to end a 42 simulation that is running with the FSW

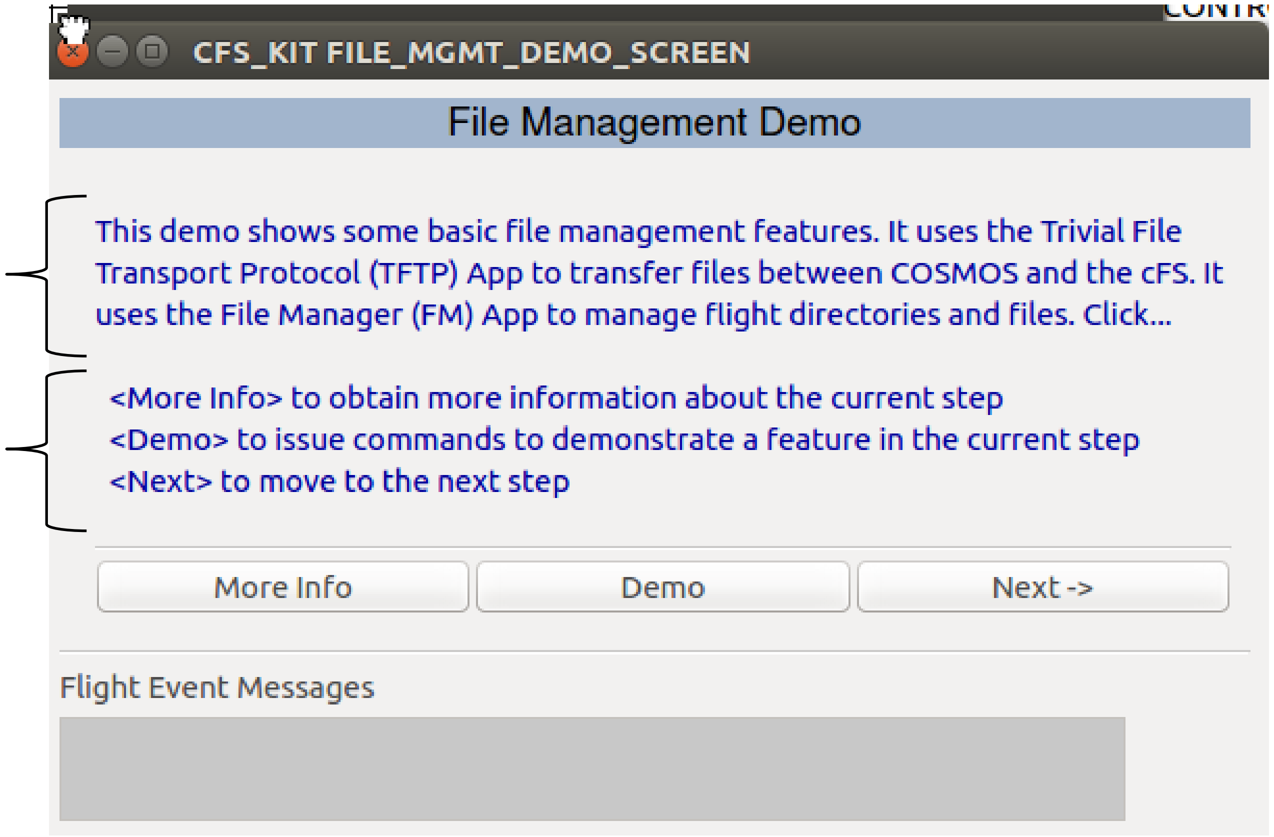
2. To terminate the flight software click on the terminal window with the FSW messages and then enter ctrl-c

3. Each of the cosmos windows will need to be closed individually. If you close the COSMOS TlmViewer window first it prompt you to close all of the telemetry screens at once.

Demos

Screen-based Demos: FM Example (1 of 2)

Each demo follows a common user screen configuration



Description of current step

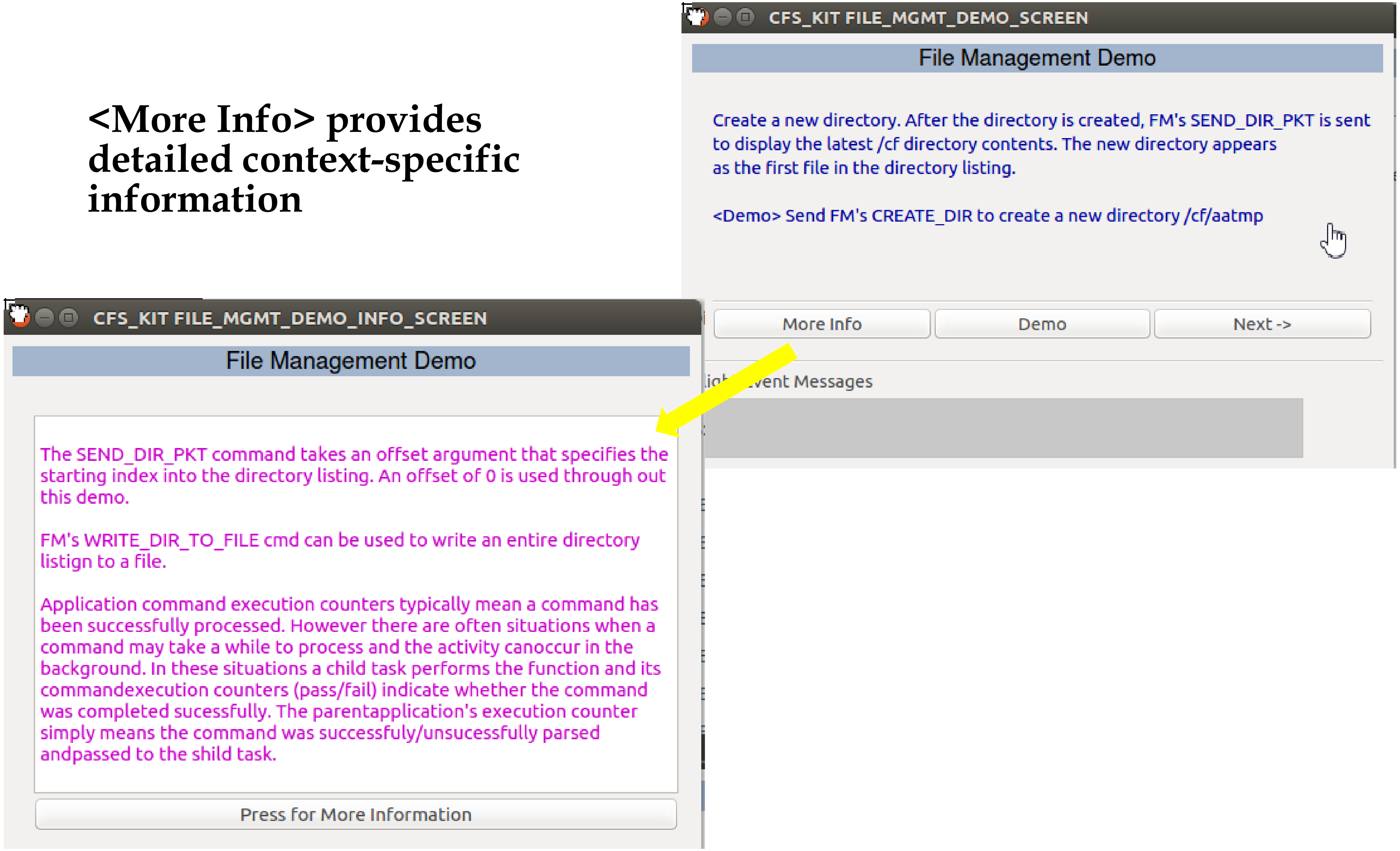
This demo shows some basic file management features. It uses the Trivial File Transport Protocol (TFTP) App to transfer files between COSMOS and the cFS. It uses the File Manager (FM) App to manage flight directories and files. Click...

Button usage description

<More Info> to obtain more information about the current step
<Demo> to issue commands to demonstrate a feature in the current step
<Next> to move to the next step

Screen-based Demos: FM Example (2 of 2)

<More Info> provides detailed context-specific information



CFS_KIT FILE_MGMT_DEMO_SCREEN
File Management Demo
More Info
Demo
Next ->
Lig vent Messages
CFS_KIT FILE_MGMT_DEMO_INFO_SCREEN
File Management Demo
Press for More Information

Create a new directory. After the directory is created, FM's SEND_DIR_PKT is sent to display the latest /cf directory contents. The new directory appears as the first file in the directory listing.

<Demo> Send FM's CREATE_DIR to create a new directory /cf/aatmp

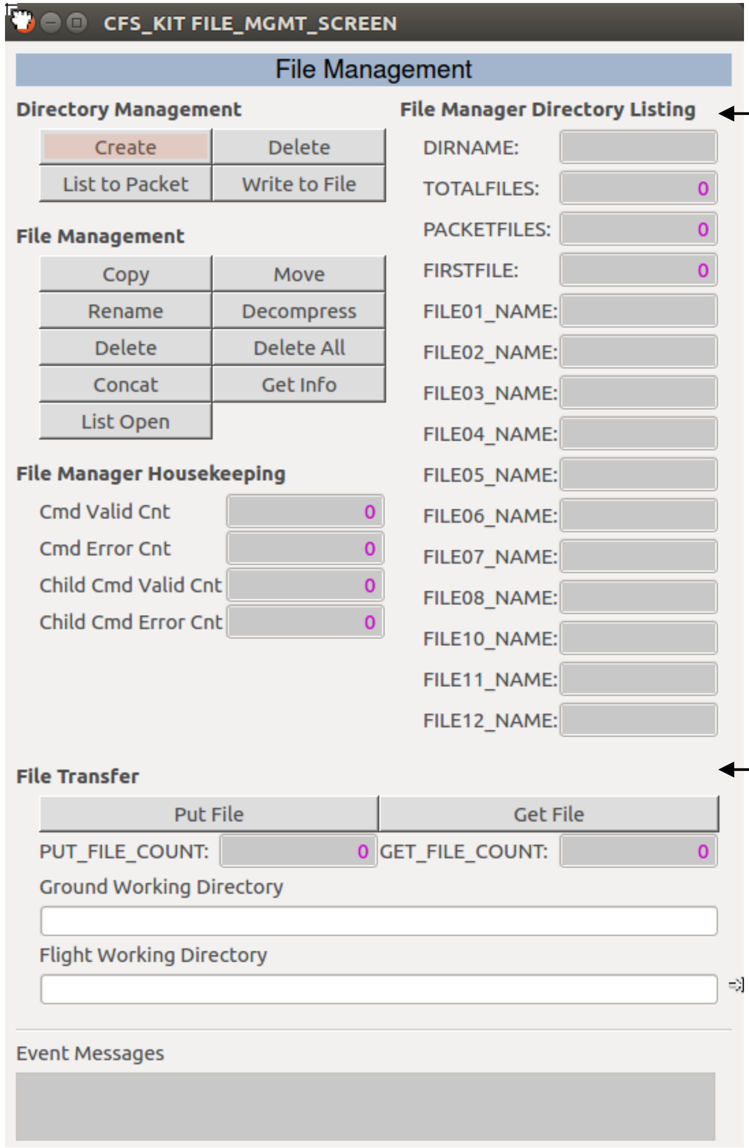
The SEND_DIR_PKT command takes an offset argument that specifies the starting index into the directory listing. An offset of 0 is used through out this demo.

FM's WRITE_DIR_TO_FILE cmd can be used to write an entire directory listign to a file.

Application command execution counters typically mean a command has been successfully processed. However there are often situations when a command may take a while to process and the activity canoccur in the background. In these situations a child task performs the function and its commandexecution counters (pass/fail) indicate whether the command was completed sucessfully. The parentapplication's execution counter simply means the command was successfuly/unsucessfully parsed andpassed to the shild task.

Application Functional Screens

File Management



Cmd Valid Cnt: 0
Cmd Error Cnt: 0
Child Cmd Valid Cnt: 0
Child Cmd Error Cnt: 0

DIRNAME:
TOTALFILES: 0
PACKETFILES: 0
FIRSTFILE: 0
FILE01_NAME:
FILE02_NAME:
FILE03_NAME:
FILE04_NAME:
FILE05_NAME:
FILE06_NAME:
FILE07_NAME:
FILE08_NAME:
FILE10_NAME:
FILE11_NAME:
FILE12_NAME:

PUT_FILE_COUNT: 0
GET_FILE_COUNT: 0
Ground Working Directory
Flight Working Directory

- <List to Packet> commands File Manage (FM)
  - To send a directory listing
  - The command uses a directory listing alphabetical “offset” to determine which file to start with in the listing
- OSK uses the verbs ***list*** and ***send*** to indicate information is sent in a telemetry packet.
- ***Write*** is used when information is written to a file

- <List to Packet> commands File Manage (FM)
  - To send a directory listing
  - The command uses a directory listing alphabetical “offset” to determine which file to start with in the listing

Table Management

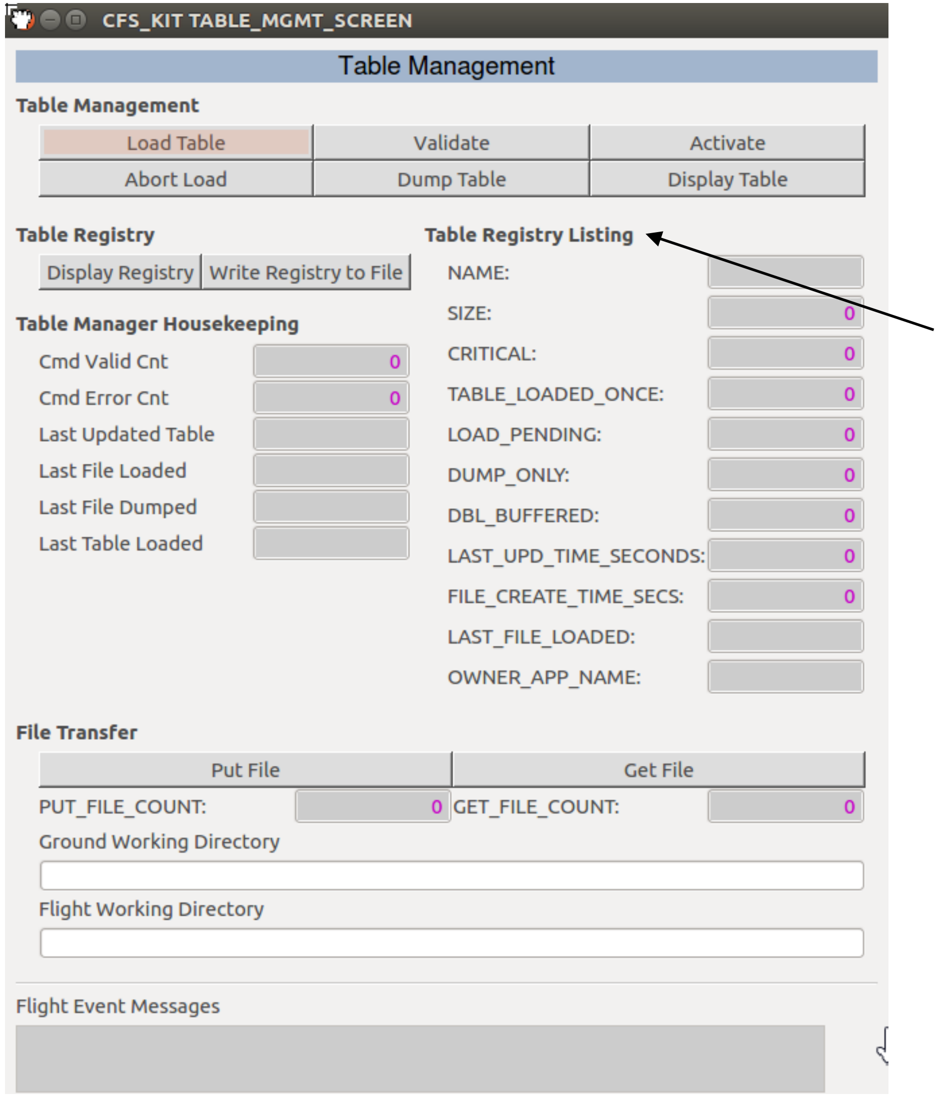


Table Management

Table Management

Table Registry

Table Manager Housekeeping

Cmd Valid Cnt	0
Cmd Error Cnt	0
Last Updated Table	
Last File Loaded	
Last File Dumped	
Last Table Loaded	

Table Registry Listing

NAME:	
SIZE:	0
CRITICAL:	0
TABLE_LOADED_ONCE:	0
LOAD_PENDING:	0
DUMP_ONLY:	0
DBL_BUFFERED:	0
LAST_UPD_TIME_SECONDS:	0
FILE_CREATE_TIME_SECS:	0
LAST_FILE_LOADED:	
OWNER_APP_NAME:	

File Transfer

PUT_FILE_COUNT:	0	GET_FILE_COUNT:	0

Ground Working Directory

Flight Working Directory

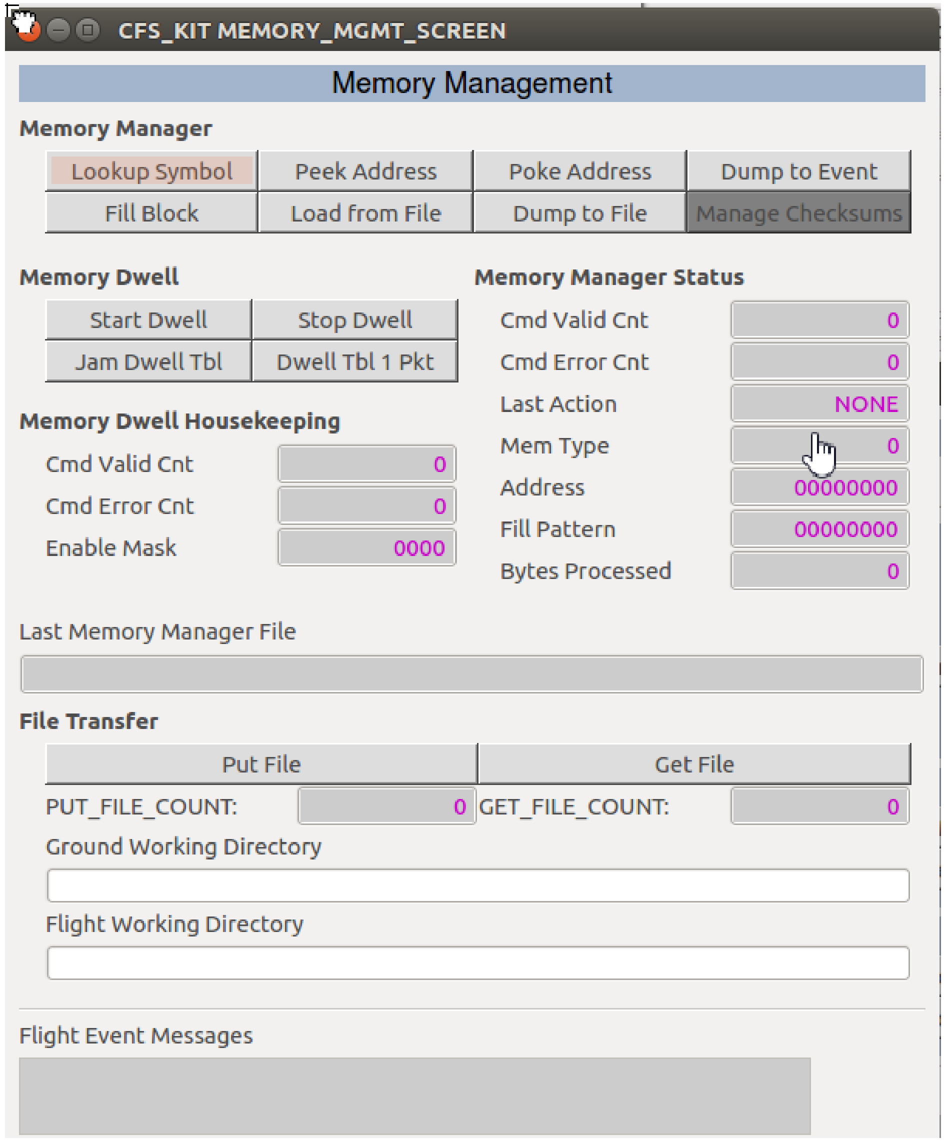
Flight Event Messages

- Load a new FSW table
  <***Put File***> transfers file from ground to flight
  <***Load Table***> into table buffer
  <***Validate***> table via app validation function
  <***Activate***> new table

- <***Display Registry***> sends a table's registry information in a telemetry packet

- Dump and display FSW table
  <***Dump Table***> to onboard file
  <***Get File***> transfers file from flight to ground
  <***Display Table***> launches COSMOS Table Manager to view file. Requires binary file definition.

Memory Management



Memory Management

Memory Manager

Memory Dwell

Memory Dwell Housekeeping

Cmd Valid Cnt	0
Cmd Error Cnt	0
Enable Mask	0000

Memory Manager Status

Cmd Valid Cnt	0
Cmd Error Cnt	0
Last Action	NONE
Mem Type	0
Address	00000000
Fill Pattern	00000000
Bytes Processed	0

Last Memory Manager File

File Transfer

PUT_FILE_COUNT: 0 GET_FILE_COUNT: 0

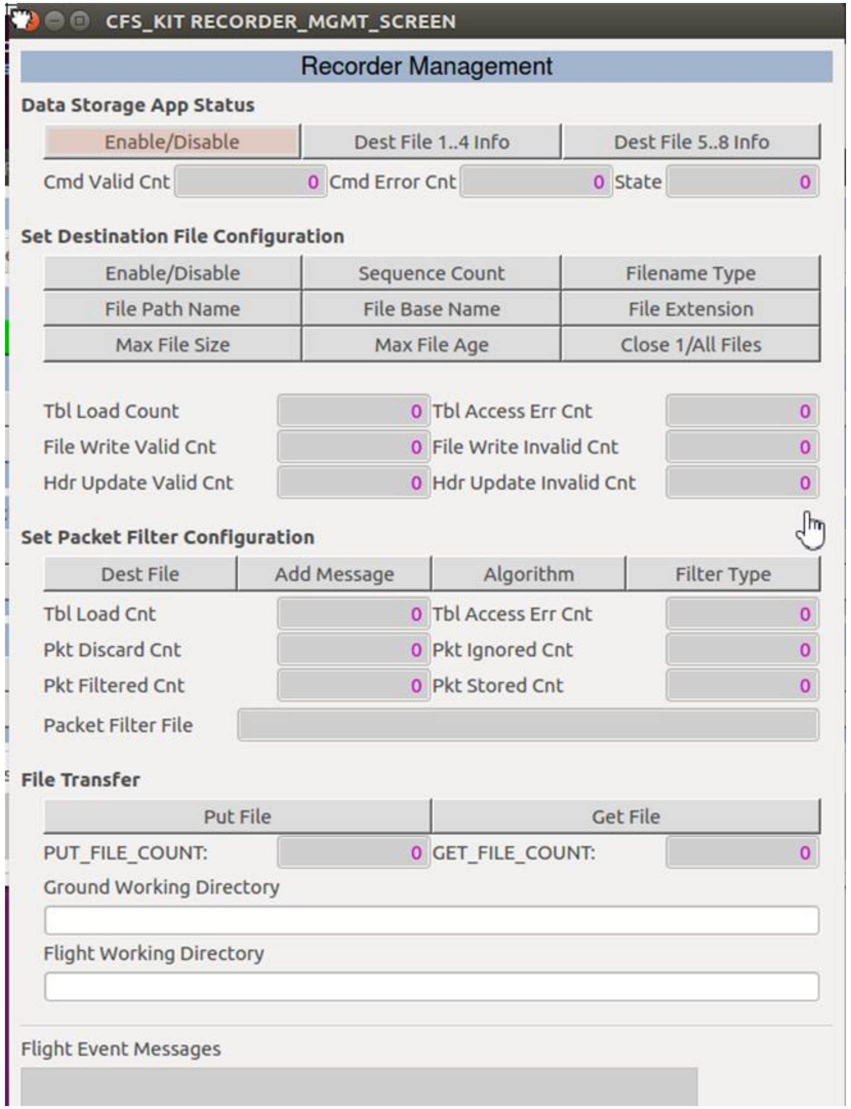
Ground Working Directory

Flight Working Directory

Flight Event Messages

- Memory Manager (MM) and Memory Dwell (MD) apps are typically used for inflight maintenance.
- MM commands allow direct access to any memory location
- MD generates telemetry packets that contain the contents of table-specified memory locations
  - Only 1 dwell table telemetry packet is defined
  - <*Jam Dwell Table*> allows the dwell table to be loaded without using the table load service
- The FSW can easily be corrupted using memory manager
- The memory management demo is a good place to start since it demonstrates MM and MD using safe memory locations

Recorder Management



CFS_KIT RECORDER_MGMT_SCREEN

Recorder Management

Data Storage App Status

Cmd Valid Cnt 0 Cmd Error Cnt 0 State 0

Set Destination File Configuration

Tbl Load Count 0 Tbl Access Err Cnt 0

File Write Valid Cnt 0 File Write Invalid Cnt 0

Hdr Update Valid Cnt 0 Hdr Update Invalid Cnt 0

Set Packet Filter Configuration

Tbl Load Cnt 0 Tbl Access Err Cnt 0

Pkt Discard Cnt 0 Pkt Ignored Cnt 0

Pkt Filtered Cnt 0 Pkt Stored Cnt 0

Packet Filter File

File Transfer

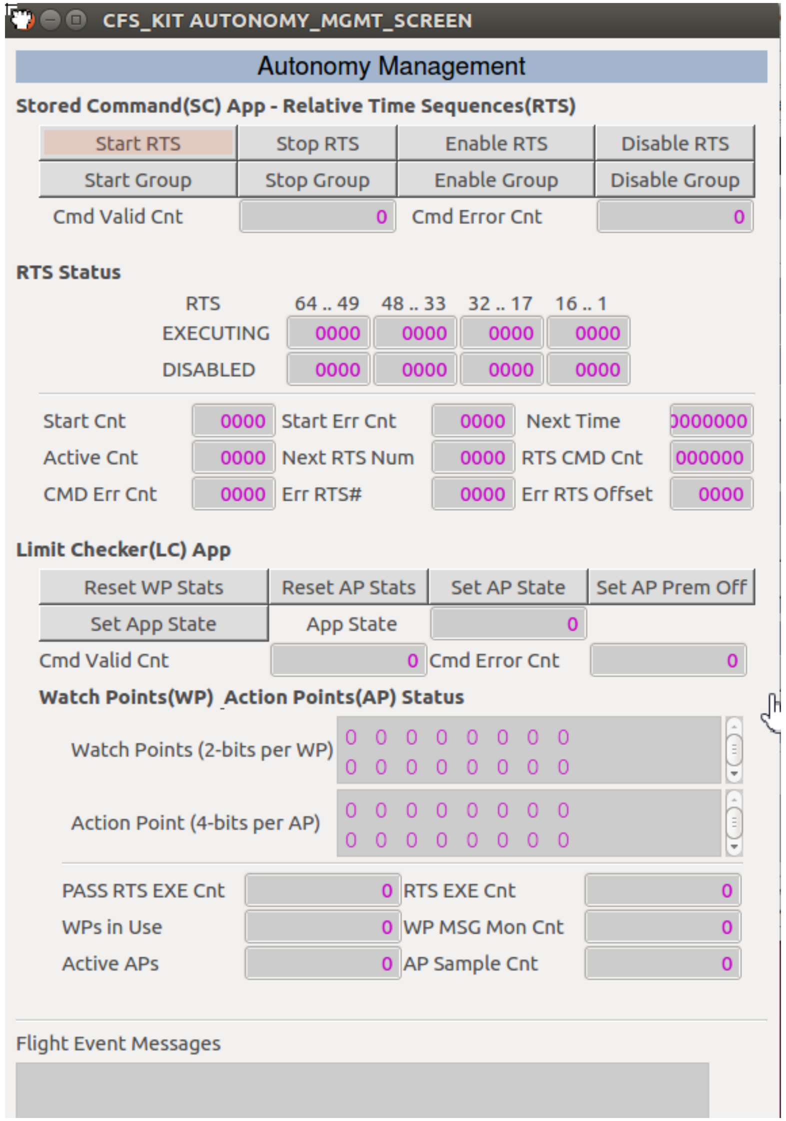
PUT_FILE_COUNT: 0 GET_FILE_COUNT: 0

Ground Working Directory

Flight Working Directory

Flight Event Messages

Autonomy Management



Autonomy Management

Stored Command(SC) App - Relative Time Sequences(RTS)

Cmd Valid Cnt 0 Cmd Error Cnt 0

RTS Status

RTS	64 .. 49	48 .. 33	32 .. 17	16 .. 1
EXECUTING	0000	0000	0000	0000
DISABLED	0000	0000	0000	0000

Start Cnt	0000	Start Err Cnt	0000	Next Time	0000000
Active Cnt	0000	Next RTS Num	0000	RTS CMD Cnt	000000
CMD Err Cnt	0000	Err RTS#	0000	Err RTS Offset	0000

Limit Checker(LC) App

App State 0

Cmd Valid Cnt 0 Cmd Error Cnt 0

Watch Points(WP) Action Points(AP) Status

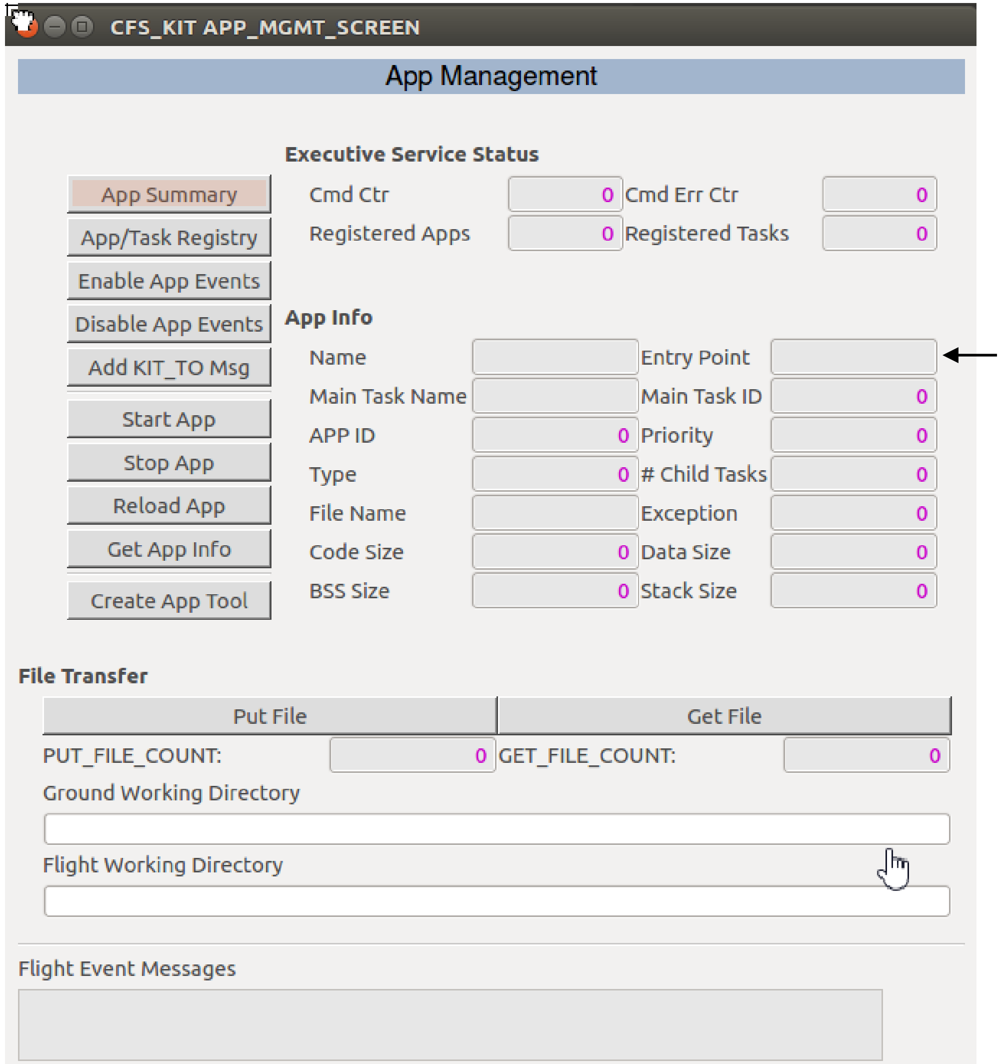
Watch Points (2-bits per WP)
```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

Action Point (4-bits per AP)
```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

PASS RTS EXE Cnt	0	RTS EXE Cnt	0
WPs in Use	0	WP MSG Mon Cnt	0
Active APs	0	AP Sample Cnt	0

Flight Event Messages

Application Management



Executive Service Status

Cmd Ctr	0	Cmd Err Ctr	0
Registered Apps	0	Registered Tasks	0

App Info

Name		Entry Point	
Main Task Name		Main Task ID	0
APP ID	0	Priority	0
Type	0	# Child Tasks	0
File Name		Exception	0
Code Size	0	Data Size	0
BSS Size	0	Stack Size	0

File Transfer

PUT_FILE_COUNT:	0	GET_FILE_COUNT:	0

Ground Working Directory

Flight Working Directory

Flight Event Messages

- *<Get App Info>* commands cFE executive services to send a telemetry packet with the command-specified app

- *<App/Task Registry>* commands cFE executive services to write app or task information to a file that can be transferred to ground via a *<Get File>*