



# OpenSatKit (OSK)

## Quick Start Guide

# Introduction

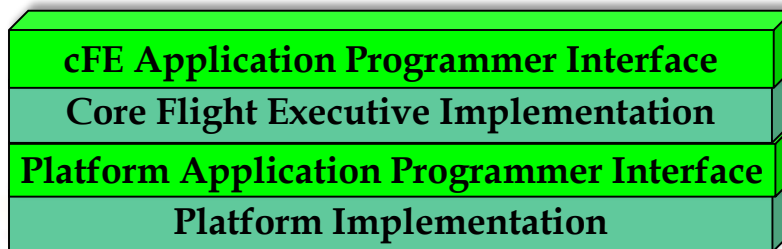
---

- **The primary goals of OpenSatKit (OSK) are to**
  - Provide a core Flight System (cFS) training environment
  - Provide a cFS application development environment
  - Serve as a starting point for a new cFS-based project
- **The cFS is an open architecture that is designed to be ported and extended**
  - These attributes add end-user deployment/configuration complexity
  - OSK provides fully functional cFS system deployed on Linux, however...
- **OSK introduces additional complexity because it integrates two additional powerful software packages, COSMOS and the 42 Simulator, that have their own learning curve.**

**There is a lot of high quality free software functionality so the rewards are high if you can persist through the learning curve!**

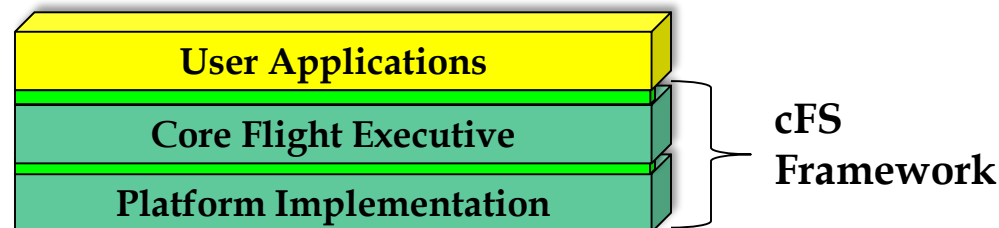
# cFS Introduction

- A NASA multi-center configuration controlled open source flight software framework



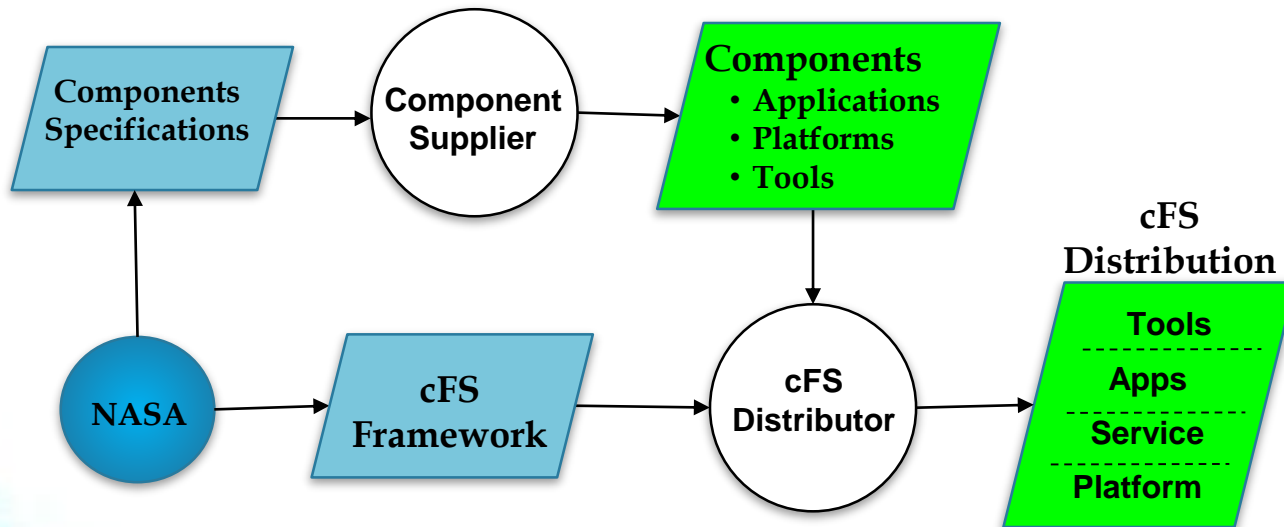
- Layered architecture with international standards-based interfaces
- Provides development tools and runtime environment for user applications
- Reusable NASA Class A/B lifecycle artifacts: requirements, design, code, tests, and documents

- The framework is ported to a platform and augmented with applications to create Core Flight System (cFS) distributions



- A worldwide community from government, industry, and academia

# cFS Product Model

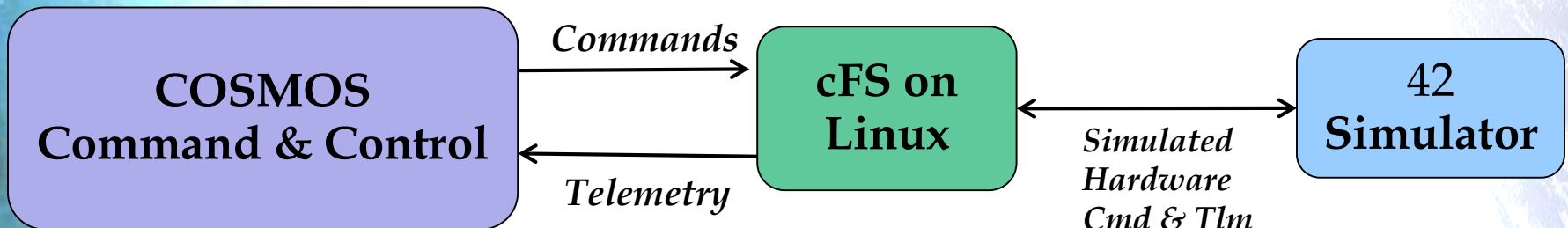


- A NASA multi-center configuration control board (CCB) manages releases of the open source cFS Framework and component specifications
- Community members (regardless of affiliation)
  - Supply applications, platforms, and tools
  - Create cFS distributions - OSK is a distribution



# OSK Architecture

- In addition to the cFS itself, OSK uses two additional open source applications
  - Ball Aerospace's COSMOS command and control platform for embedded systems
  - NASA Goddard's 42 dynamic simulator
- Each open source package is contained in its own OpenSatKit subdirectory



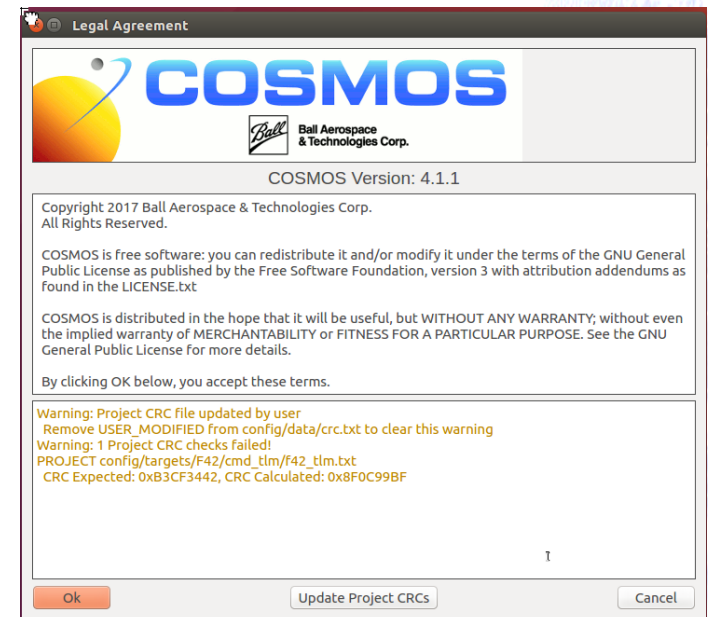
# Approach

---

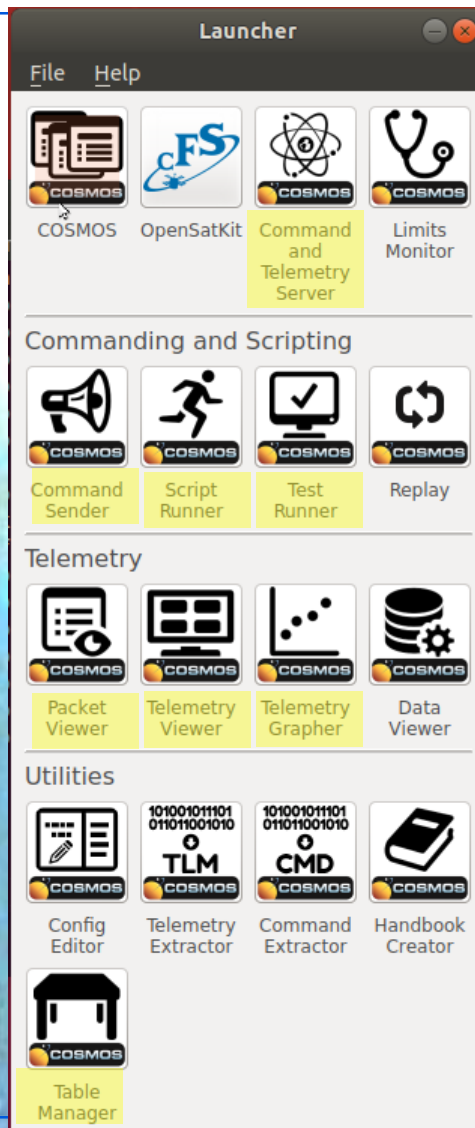
- **OSK comes with the cFS pre-configured for a fictitious satellite called SimpleSat (SimSat).**
  - The cFS can be used for many different types of embedded systems. A spacecraft was chosen due to the increased usage of the cFS on CubeSats
- **OSK implements extensive COSMOS configurations and customizations so COSMOS can serve as the primary OSK user interface**
- **OSK is arranged with the following user progression in mind**
  1. **Learn the cFS using SimSat to provide a context and working examples**
  2. **Manage and develop applications within the Linux desktop environment**
    - a. Add apps by creating new apps or importing from the app library
    - b. Configure runtime app suite
  3. **Extend OSK**
    - a. Deploy the cFS to a target system
      - i. Run benchmarks
      - ii. Use OSK as a ground system for a remote system
    - b. **Advanced application development and extensions**
      - i. External Code Interface (ECI), ROS2 bridge, etc.

# Running OpenSatKit (1 of 2)

- Open a terminal window (Ctrl-Alt-t)
- Navigate to the base directory where you installed OSK
  - “~/” is used to indicate the OSK base directory so “~/cfs” is equivalent to “/home/user/OpenSatKit-master/cfs” if OpenSatKit was installed in the home directory for an account named “user”
- Change directory to cosmos
  - cd ~/cosmos
- Start COSMOS
  - [~/cosmos]ruby Launcher
  - You’ll see a screen similar to the right.
    - Select <OK>
    - This creates the “Launcher” screen shown on the next slide



# Running OpenSatKit (2 of 2)



- Each tool on the COSMOS “Launcher” runs as a separate Linux process with a Graphical User Interface (GUI)
- Shaded **tool titles** indicate the COSMOS tools used by OSK
  - You do not have to invoke these tools directly
  - OSK screens launch COSMOS tools as they are needed to perform a task
  - A backup slide shows a COSMOS architectural view with the data flows between tools
- Select “OpenSatKit” with a single click
  - This launches COSMOS’s Command and Telemetry Server, Telemetry Viewer, and displays OSK’s main window
  - You can minimize the COSMOS tools, but don’t close them
- A picture of OSK’s main window follows 2 slides that briefly describe each COSMOS tool



# COSMOS Tool Summary (1 of 2)

---

- **Launcher**

- Provides a graphical interface for launching each of the tools that make up the COSMOS system
- *Custom OSK ICON "cFS Starter Kit" launches OSK's main page*

- **Command and Telemetry Server**

- Connects COSMOS to targets for real-time commanding and telemetry processing.
- All real-time COSMOS tools communicate with targets through the Command and Telemetry Server ensuring that all communications are logged.
- Localhost 127.0.0.1 used as cFS connection Targets created

- **Telemetry Viewer**

- Provides a way to organize telemetry points into custom "screens" that allow for the creation of unique and organized views of telemetry data.

- **Command Sender**

- Individually send any FSW command using GUI form
- Raw data files can be used to inject faults
- *OSK provides custom menus for common cFS commands*

- **Packet Viewer**

- View any telemetry packet with no extra configuration necessary
- *OSK provides custom telemetry screens functionally organized*

# COSMOS Tool Summary (2 of 2)

---

- **Telemetry Grapher**

- Real-time or offline graphing of any FSW telemetry point
- *OSK provides convenient access through some of its custom screens*

- **Table Manager**

- Edit and display binary files
- *OSK provides definitions for most of the cFE binary files and a limited number of cFS application binary files*

- **Script Runner**

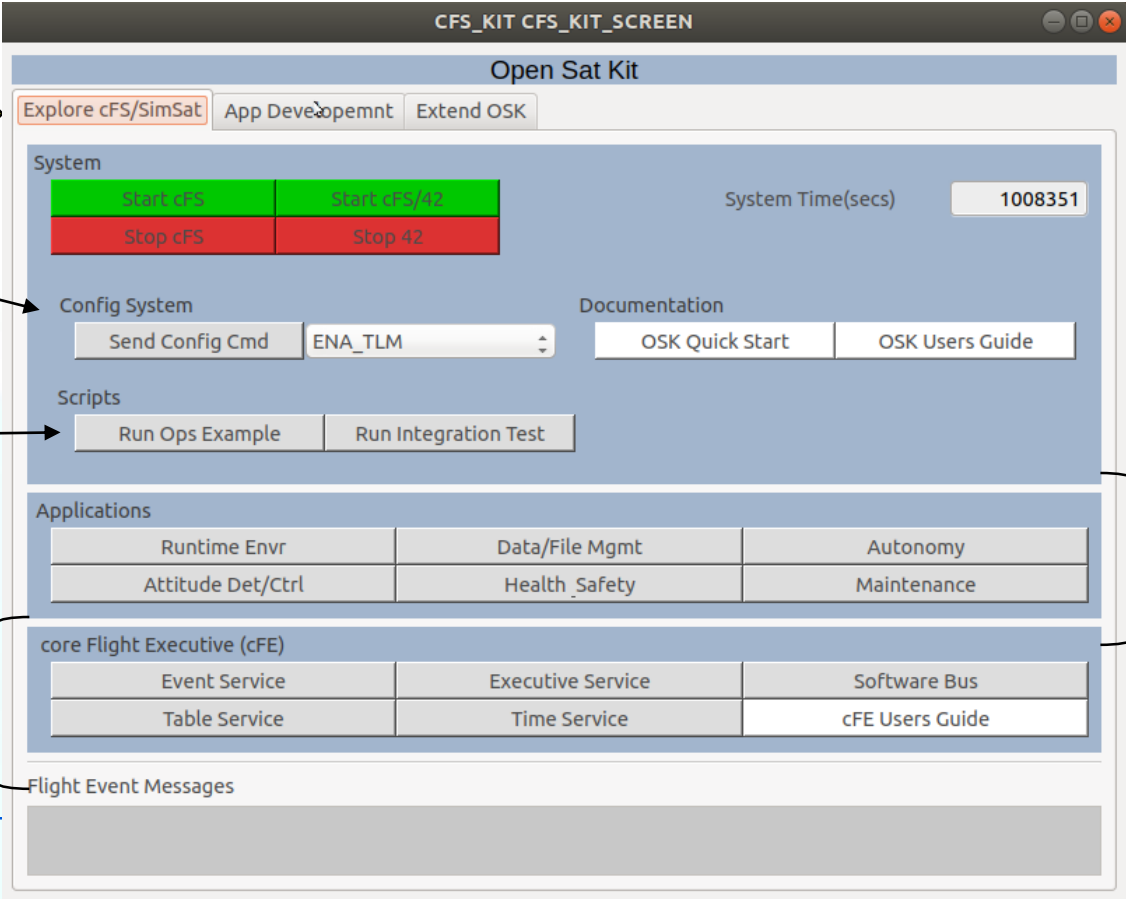
- Develop and execute test procedures using Ruby Scripts and COSMOS APIs
- *OSK provides additional APIs for functions like file transfer and binary file management*

- **Test Runner**

- Test framework for organizing, executing, and verifying test scripts
- *Currently OSK only includes some prototype scripts. The goal is to provide a complete test suite that can be extended by the user.*

# Main OSK Window

- Three tabs *Explore cFS/SimSat*, *Manage Apps*, and *Extend OSK* provide the top-level organization
- *Explore cFS/SimSat* allows the user to learn the cFS using SimSat
- *Manage Apps* provides tools for adding, removing, and creating apps
- *Extend OSK* is in its infancy, but it's goal is to allow the user to bridge the cFS to other systems and control remote devices



**Tabs** → Explore cFS/SimSat | App Development | Extend OSK

**A few high level system-oriented commands** →

Start cFS	Start cFS/42	System Time(secs) 1008351
Stop cFS	Stop 42	

**Launch scripts in script Runner** →

Config System: Send Config Cmd | ENA\_TLM | Documentation: OSK Quick Start | OSK Users Guide

Scripts: Run Ops Example | Run Integration Test

**Explore 5 cFE services** →

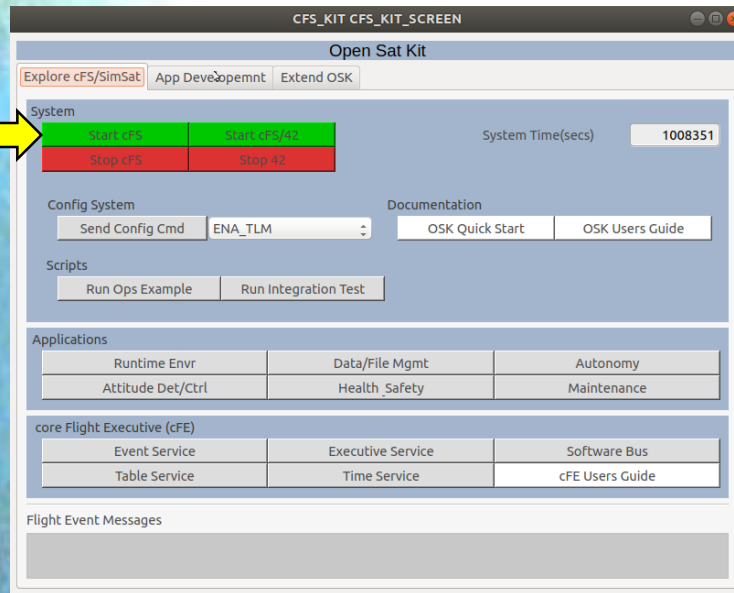
Applications		
Runtime Envr	Data/File Mgmt	Autonomy
Attitude Det/Ctrl	Health_Safety	Maintenance
core Flight Executive (cFE)		
Event Service	Executive Service	Software Bus
Table Service	Time Service	cFE Users Guide

**Explore apps arranged in functional groups** →

Flight Event Messages

# Start the Flight Software (FSW)

- Click <Start cFS> to run the FSW. <Start cFS/42> is used later.
  - A new terminal window is created for the Linux process running the cFS
  - Enter “osk” when prompted for a password.
- In a few seconds the time box should turn white time with advancing
  - If time doesn’t advance click <Send Config Cmd> “ENA\_TLM”



```

Terminal
CTRLTBL LoadCmd() Entry
***I42 NETIF***: Attempting to connect to Server localhost on Port 4242
EVS Port1 42/1/CFE_SB 17: Msg Limit Err,MsgId 0x808,pipe KIT_TO_PKT_PIPE,sender I42
EVS Port1 42/1/I42 999: Error connecting client socket: Connection refused
EVS Port1 42/1/CFE_SB 17: Msg Limit Err,MsgId 0x808,pipe KIT_TO_PKT_PIPE,sender I42
EVS Port1 42/1/I42 100: I42 App Initialized. Version 1.0.0.0
CTRLTBL LoadCmd() - Successfully prepared file /cf/f42_ctrl_tbl.json

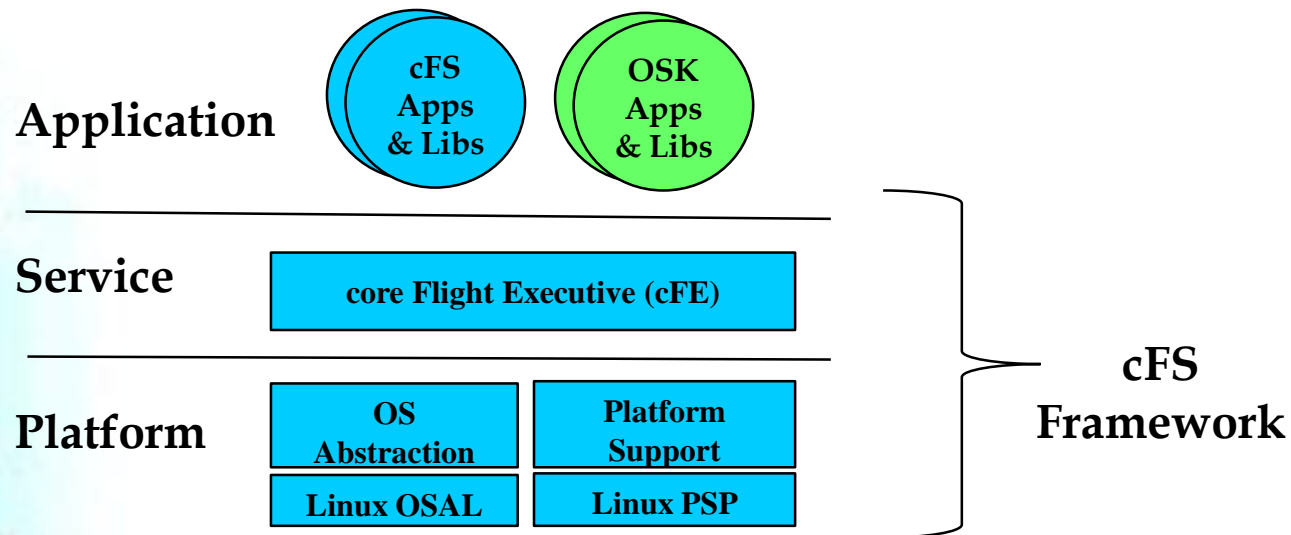
CTRLTBL.MoiCallback: ObjLoadCnt 0, AttrErrCnt 0, TokenIdx 6
CTRLTBL.MoiCallback: 0.119835, 0.147780, 0.044908

CTRLTBL.MoiCallback: ObjLoadCnt 1, AttrErrCnt 0, TokenIdx 14
CTRLTBL.PdGainParamCallback: 0.628000, 0.700000
F42 ADP LoadCtrlTbl() Kr[0] = 0.039483, Kp[0] = 0.017711
EVS Port1 42/1/CFE_SB 17: Msg Limit Err,MsgId 0x808,pipe KIT_TO_PKT_PIPE,sender F42
EVS Port1 42/1/F42 100: F42 App Initialized. Version 1.0.0.0
1980-012-14:03:20.62440 ES Startup: CFE ES Main entering OPERATIONAL state
Warning: System Log full, log entry discarded.
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/SC 73: RTS Number 001 Started
  
```



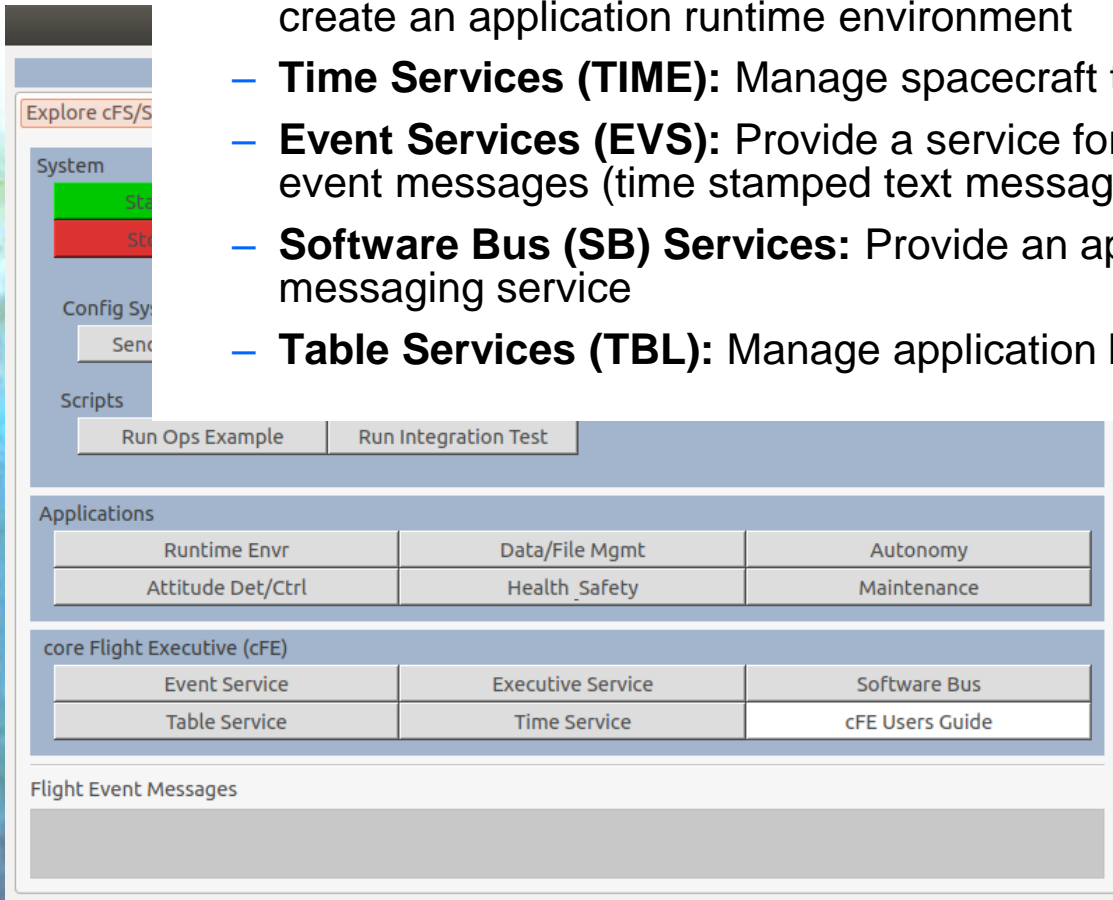
# What Just Happened?

- The <Start cFS> button invoked a ruby script that created a new terminal window executing the “cFS Framework”
- The cFS Framework is the bottom two layers of the 3-tiered cFS architecture. It is a portable application runtime environment that uses a startup script (cfe\_es\_startup.scr) to determine which apps to load during initialization. OSK’s startup script is configured for SimSat.



# Core Flight Executive (cFE)

- The cFE has 5 services
  - **Executive Services (ES):** Manage the embedded software system and create an application runtime environment
  - **Time Services (TIME):** Manage spacecraft time
  - **Event Services (EVS):** Provide a service for sending, filtering, and logging event messages (time stamped text messages).
  - **Software Bus (SB) Services:** Provide an application publish/subscribe messaging service
  - **Table Services (TBL):** Manage application binary file table images



The screenshot shows the OpenSatKit web interface. On the left is a sidebar with navigation links: 'Explore cFS/S...', 'System', 'Config Sy...', 'Scripts', 'Run Ops Example', and 'Run Integration Test'. The main content area is titled 'Applications' and contains a table of services. Below this is a section for 'core Flight Executive (cFE)' with another table of services. At the bottom is a section for 'Flight Event Messages'.

Applications		
Runtime Envr	Data/File Mgmt	Autonomy
Attitude Det/Ctrl	Health_Safety	Maintenance

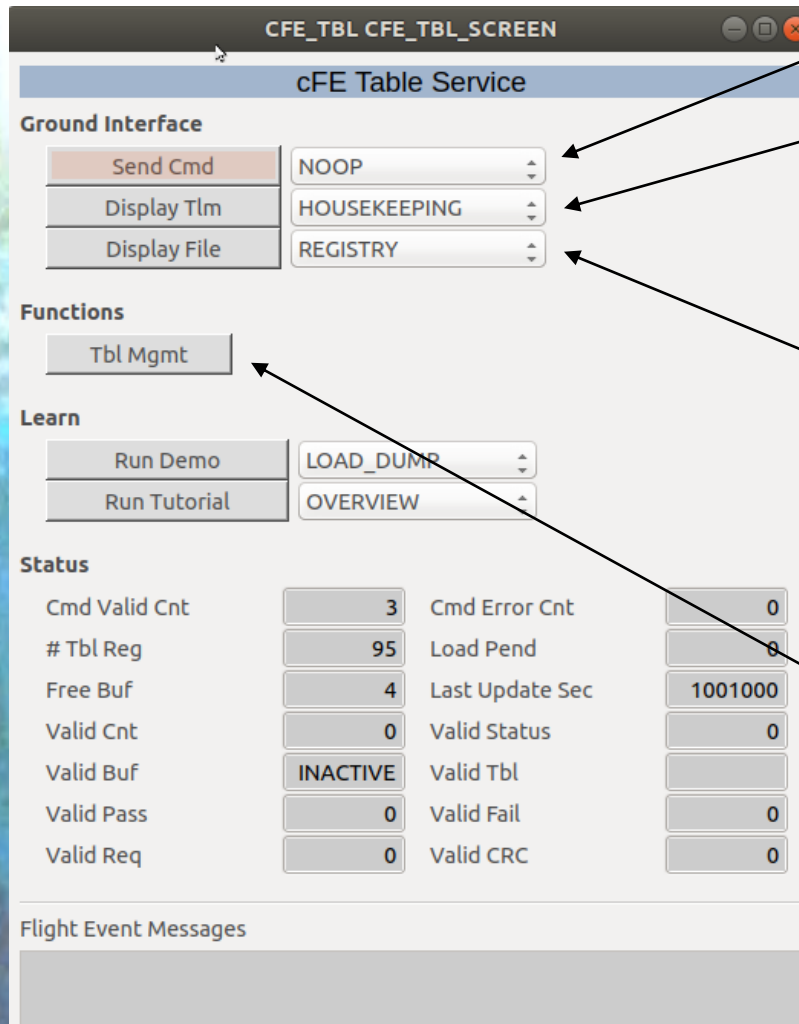
core Flight Executive (cFE)		
Event Service	Executive Service	Software Bus
Table Service	Time Service	cFE Users Guide

One button/screen  
for each service

cFE HTML User's  
Guide

# cFE Service Screen (1 of 2)

- Table Service screen shown. All cFE screens have the same layout but may not have every component/button



The screenshot shows the 'cFE Table Service' window. It has a title bar 'CFE\_TBL CFE\_TBL\_SCREEN'. The main content is divided into several sections: 'Ground Interface' with buttons 'Send Cmd', 'Display Tlm', and 'Display File'; 'Functions' with a button 'Tbl Mgmt'; 'Learn' with buttons 'Run Demo' and 'Run Tutorial'; and 'Status' with two columns of data. At the bottom is a 'Flight Event Messages' section.

**Ground Interface**

Send Cmd	NOOP
Display Tlm	HOUSEKEEPING
Display File	REGISTRY

**Functions**

Tbl Mgmt
----------

**Learn**

Run Demo	LOAD_DUMP
Run Tutorial	OVERVIEW

**Status**

Cmd Valid Cnt	3	Cmd Error Cnt	0
# Tbl Reg	95	Load Pend	0
Free Buf	4	Last Update Sec	1001000
Valid Cnt	0	Valid Status	0
Valid Buf	INACTIVE	Valid Tbl	
Valid Pass	0	Valid Fail	0
Valid Req	0	Valid CRC	0

**Flight Event Messages**

Select and send commands

Display a telemetry packet using COSMOS's Packet Viewer.

- Telemetry packets can be generated in response to a command
- E.g. Telemeter the registration information for a single table

Display a binary file using COSMOS's Table Manager

- Binary files can be generated in response to a command.
- E.g. Dump the entire table registry to a file

Display a screen that simplifies user interaction with a service

# cFE Service Screen (2 of 2)

CFE\_TBL CFE\_TBL\_SCREEN

cFE Table Service

Ground Interface

Send Cmd

NOOP

Display Tlm

HOUSEKEEPING

Display File

REGISTRY

Functions

Tbl Mgmt

Learn

Run Demo

LOAD\_DUMP

Run Tutorial

OVERVIEW

Status

Cmd Valid Cnt	3	Cmd Error Cnt	0
# Tbl Reg	95	Load Pend	0
Free Buf	4	Last Update Sec	1001000
Valid Cnt	0	Valid Status	0
Valid Buf	INACTIVE	Valid Tbl	
Valid Pass	0	Valid Fail	0
Valid Req	0	Valid CRC	0

Flight Event Messages

## Select and run a demo

- Demos are a sequence of interactive screens that step the user through a task

## Select and run a tutorial

- Tutorial are typically, but not limited to a set of slides coupled with a ruby script for exercises

## Each service generates a periodic “housekeeping” telemetry packet every few seconds

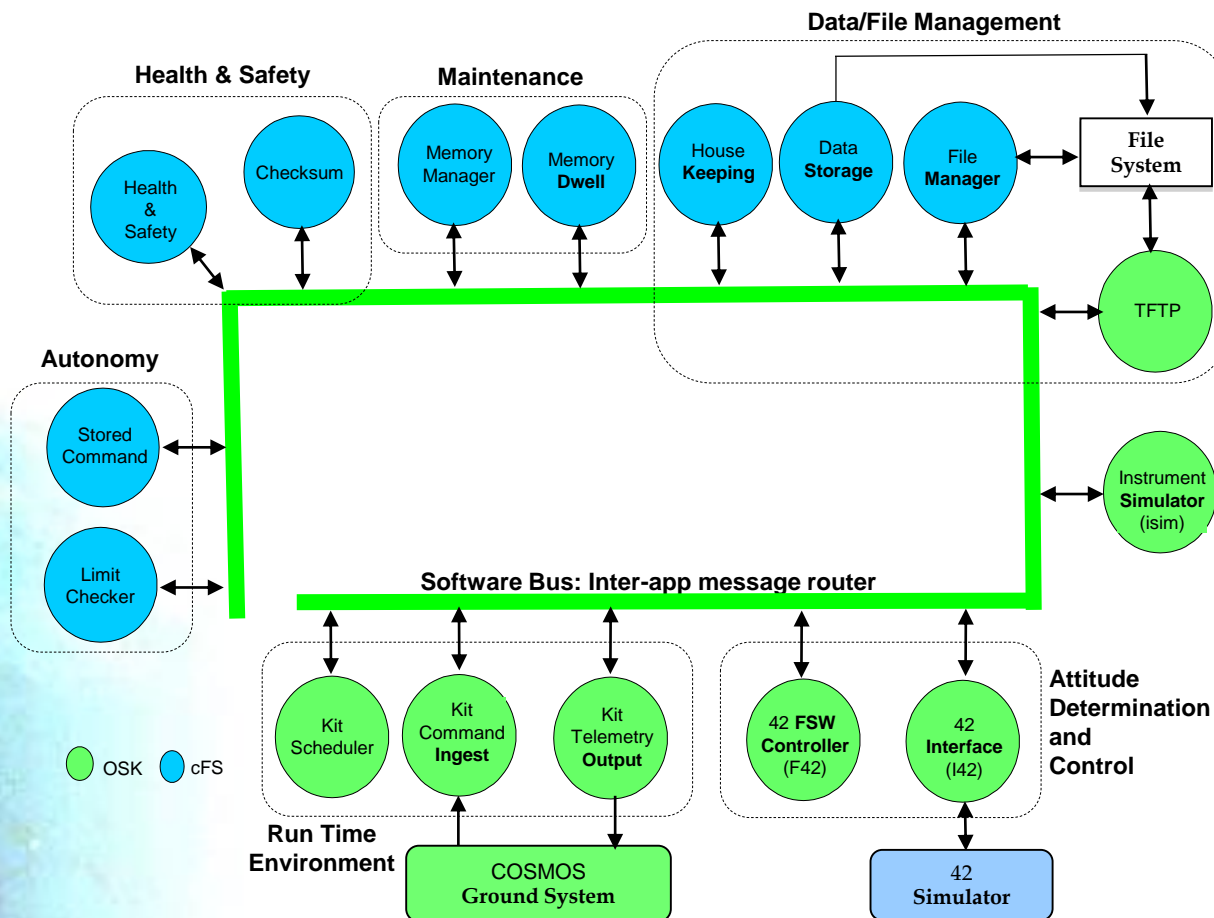
- The ‘Status’ section displays a portion of the housekeeping packet
- The entire packet can be displayed using the <Display Tlm> button in the Ground Interface section



# Simple Satellite (SimSat)

- **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
    - A 1Hz auxiliary spacecraft data containing time, attitude, orbit data, and instrument status
    - Start science during a ground contact. Can be automated but ops prefers to monitor instrument health.
  - Ground contact resources/schedule are preplanned
    - Implies autonomous operations can be loaded on board using stored commands
  - FSW must autonomously monitor instrument health and power off the instrument in the event of a fault

# SimSat Applications (1 of 3)



# SimSat Applications (2 of 3)

---

- **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 (2 of 3)

---

- **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 (3 of 3)

---

- **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**

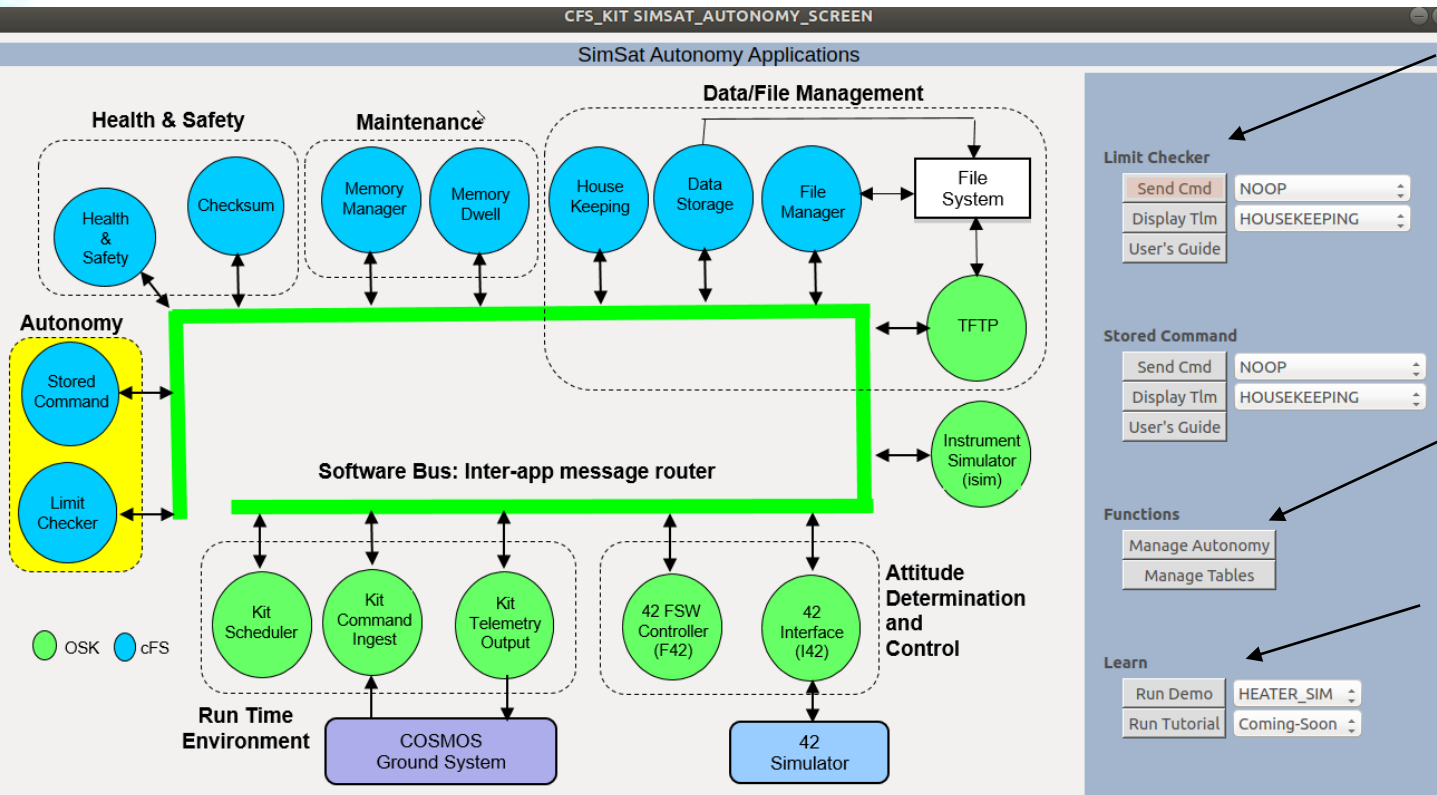
- **Memory Dwell (MD)** creates telemetry packets containing contents of memory location specified in dwell tables
- **Memory Manager (MM)** provides read/write access to memory

- **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

# SimSat Application Screens

- Each functional application group screen uses the following layout



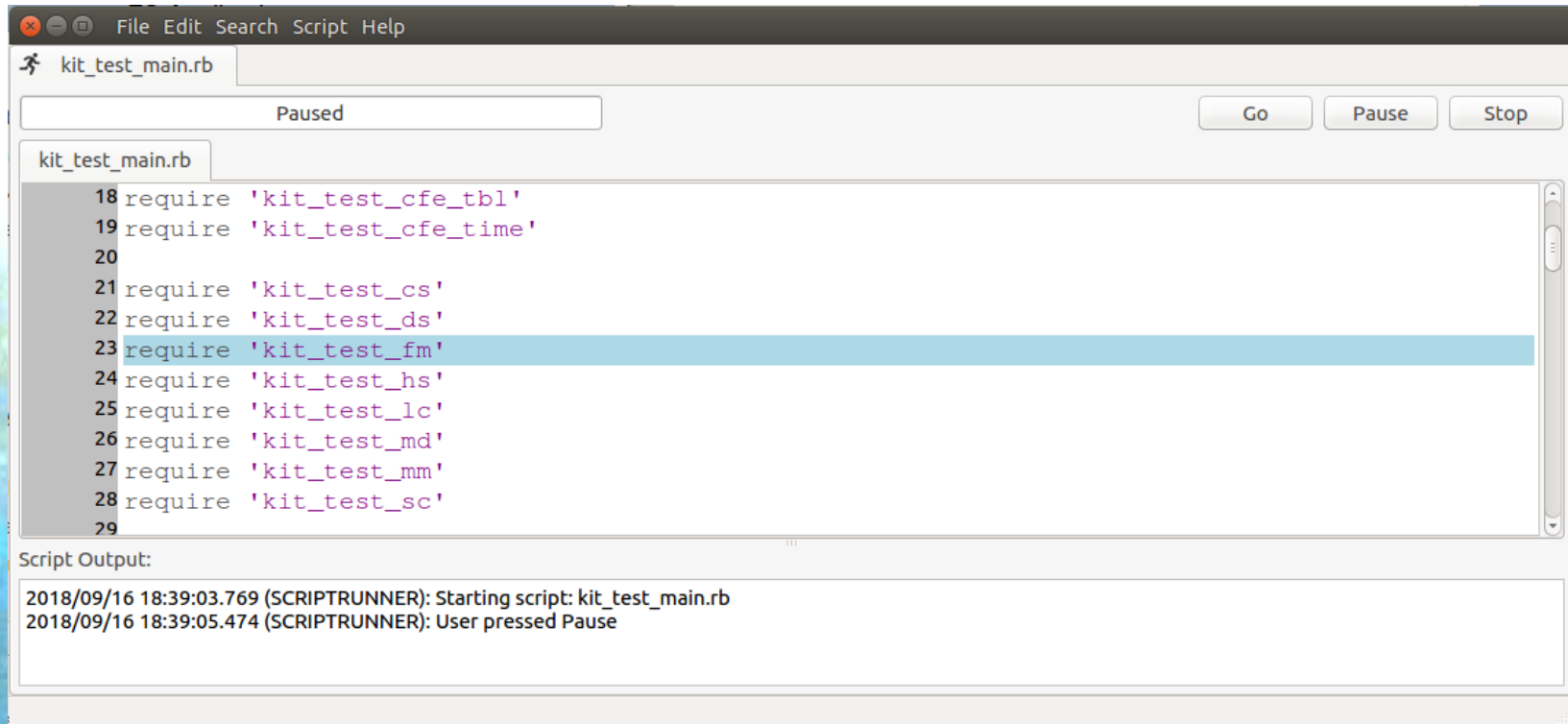
Quick access to each app's commands, telemetry, and user's guide

- Binary Tables & Files coming soon

Convenient functional screens for most common commands and relevant status. Listed at end of slide deck.

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

# SimSat Integration Script



```
File Edit Search Script Help
kit_test_main.rb
Paused
Go Pause Stop

kit_test_main.rb
18 require 'kit_test_cfe_tbl'
19 require 'kit_test_cfe_time'
20
21 require 'kit_test_cs'
22 require 'kit_test_ds'
23 require 'kit_test_fm'
24 require 'kit_test_hs'
25 require 'kit_test_lc'
26 require 'kit_test_md'
27 require 'kit_test_mm'
28 require 'kit_test_sc'
29

Script Output:
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**

# SimSat Operational Script

---

- Integration Scripts
- Operational Scripts



---

# Configuration and Convention Notes

# COSMOS Configuration (1 of 2)

---

- **COSMOS Target (*OpenSatKit/cosmos/config/targets*)**
  - Architectural component, typically on an embedded system, that COSMOS can send commands to and receive telemetry from
  - For each target users can define command packets, telemetry packets, screens, and Ruby scripts.
  - Each FSW application is defined as a target
  - OSK defines a virtual target *CFS\_KIT* to serve as the User's primary interface
- **OSK scripts in *OpenSatKit/cosmos/lib* extend COSMOS scripting API**
  - API documentation is under development. See code for details

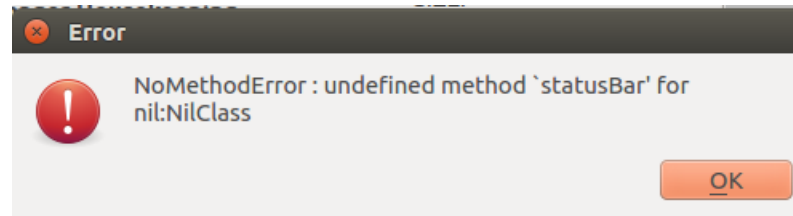
# COSMOS Configuration (2 of 2)

---

- **OSK specific directories defined in *OpenSatKit/cosmos/cfs\_kit***
  - */docs*: cFE and OSK documentation
  - */file\_server*: Default location for file transferred to/from FSW
    - */table* subdirectory contains table files
    - COSMOS Table Manager file formats defined in */cosmos/config/tools/TableManager*
  - */tools*: cFE and OSK standalone tools
  - */tutorials*: Tutorial files

# Minor Inconveniences (1 of 2)

- OSK is a work in progress with a few known issues that you can ignore
- If you cancel an OSK dialogue you may see the follow COSMOS error dialogue.



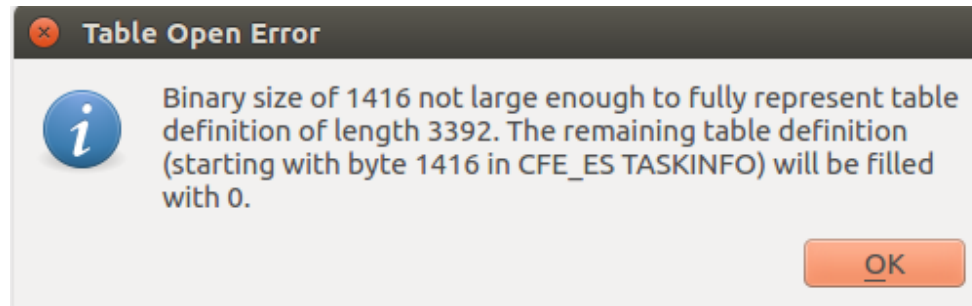
- The FSW terminal window may display start and stop “FlyWheel” messages
  - OSK is a non-realtime environment so the cFE time service is warning that’s it’s not operating within its real-time precision limits relative to a 1Hz timer
  - OSK is designed to help users learn functional features and only requires reasonable timing performance in order for the scheduler to execute its schedule correctly

```
EVS Port1 42/1/CFE_TIME 20: Start FLYWHEEL  
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
```



# Minor Inconveniences (2 of 2)

- Some cFS binary files are variable length. The Table Manager definition files support fixed length files, therefore you may see an error dialog stating the file doesn't contain all of the records. This message is from cFE Executive Service Task Information file.



# OSK Conventions

- Most cFE services have commands that can generate a telemetry as part of the response or write information to a file
  - The verbs *list* and *send* indicate information is sent in a telemetry packet.
  - *Write* is used when information is written to a file
- The FSW directory `/cf` (compact flash) is used as the default location for onboard file creation and flight-ground file transfers
  - This is mapped to `OpenSatKit/cfs/build/exe/cpu1/cf`
- `OpenSatKit/cosmos/cfs_kit/file_server` is used as the default ground file location
  - Table are located in the *tables* subdirectory
- OSK often uses `osk_tmp_bin.dat` as a standard temporary binary file name to avoid clutter
- OSK does not “cheat” when working with ground and flight tables
  - Files are transferred between flight and ground locations and not accessed via shared locations within the VM

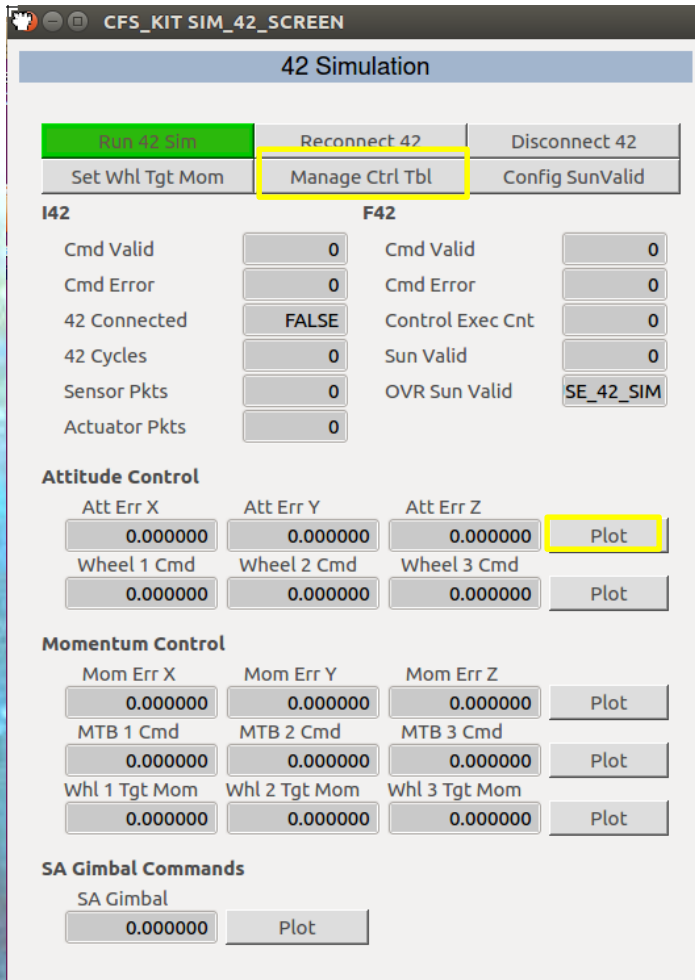


---

Running SimSat  
with 42

Needs 2.0 Updates

# Tools: Preparing 42 Simulation

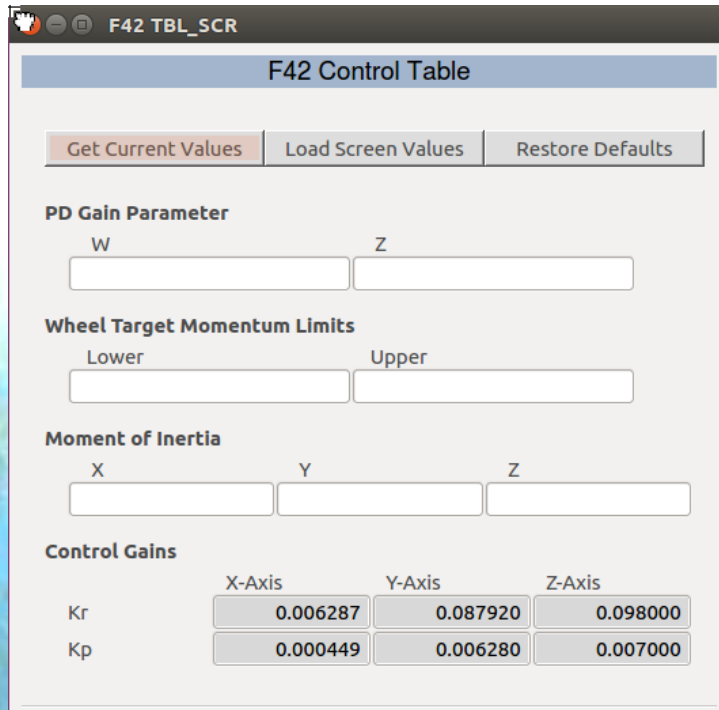


The screenshot shows the '42 Simulation' control interface. At the top, there are three buttons: 'Run 42 Sim' (highlighted in green), 'Reconnect 42' (highlighted with a yellow box), and 'Disconnect 42'. Below these are three more buttons: 'Set Whl Tgt Mom', 'Manage Ctrl Tbl' (highlighted with a yellow box), and 'Config SunValid'. The interface is divided into two main sections: 'I42' and 'F42'. The 'I42' section contains status indicators for 'Cmd Valid', 'Cmd Error', '42 Connected' (set to FALSE), '42 Cycles', 'Sensor Pkts', and 'Actuator Pkts'. The 'F42' section contains status indicators for 'Cmd Valid', 'Cmd Error', 'Control Exec Cnt', 'Sun Valid', and 'OVR Sun Valid' (set to SE\_42\_SIM). Below these are three control sections: 'Attitude Control' with 'Att Err X', 'Att Err Y', and 'Att Err Z' (all set to 0.000000) and a 'Plot' button; 'Momentum Control' with 'Mom Err X', 'Mom Err Y', and 'Mom Err Z' (all set to 0.000000) and a 'Plot' button; and 'SA Gimbal Commands' with 'SA Gimbal' (set to 0.000000) and a 'Plot' button.

- 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



# 42 Sim: Manage Control Table



**F42 Control Table**

Get Current Values Load Screen Values Restore Defaults

**PD Gain Parameter**

W Z

**Wheel Target Momentum Limits**

Lower Upper

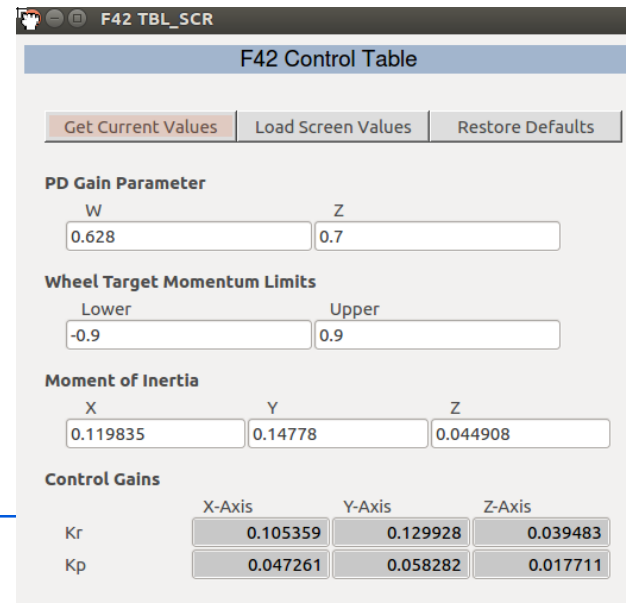
**Moment of Inertia**

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**

Get Current Values Load Screen Values Restore Defaults

**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

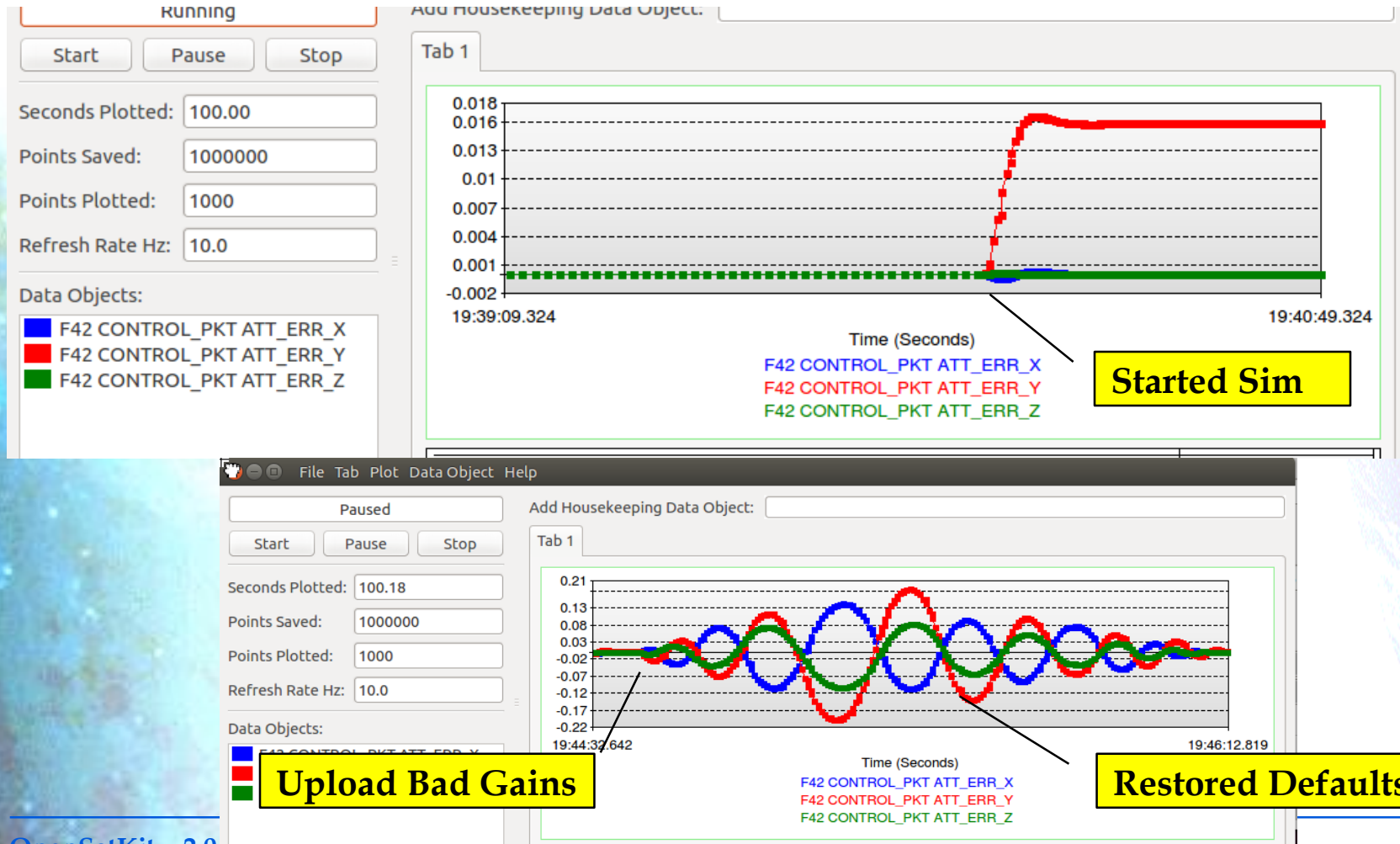
0.119835 0.14778 0.044908

**Control Gains**

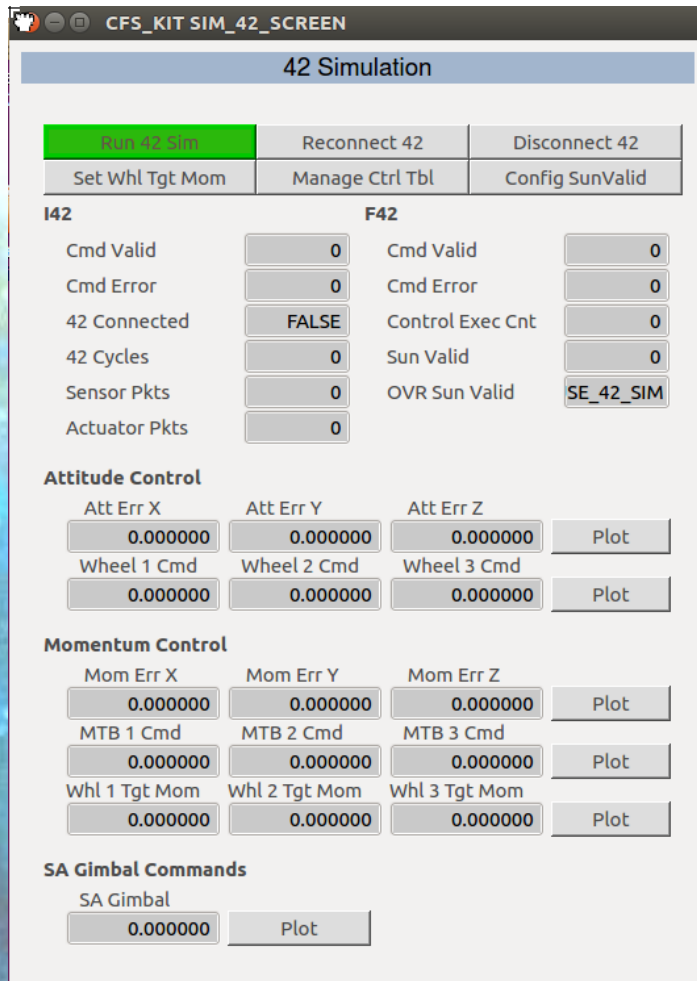
	X-Axis	Y-Axis	Z-Axis
Kr	0.105359	0.129928	0.039483
Kp	0.047261	0.058282	0.017711

# 42 Sim: Plot Attitude Errors

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



# 42 Sim: Starting the Simulation



The screenshot shows the '42 Simulation' control interface. At the top, there are three buttons: 'Run 42 Sim' (highlighted in green), 'Reconnect 42', and 'Disconnect 42'. Below these are three more buttons: 'Set Whl Tgt Mom', 'Manage Ctrl Tbl', and 'Config SunValid'. The interface is divided into two main sections: 'I42' and 'F42'. The 'I42' section contains a table of status indicators: 'Cmd Valid' (0), 'Cmd Error' (0), '42 Connected' (FALSE), '42 Cycles' (0), 'Sensor Pkts' (0), and 'Actuator Pkts' (0). The 'F42' section contains a table of status indicators: 'Cmd Valid' (0), 'Cmd Error' (0), 'Control Exec Cnt' (0), 'Sun Valid' (0), and 'OVR Sun Valid' (SE\_42\_SIM). Below these are three sections: 'Attitude Control' with 'Att Err X', 'Att Err Y', and 'Att Err Z' (all 0.000000) and 'Wheel 1 Cmd', 'Wheel 2 Cmd', and 'Wheel 3 Cmd' (all 0.000000); 'Momentum Control' with 'Mom Err X', 'Mom Err Y', and 'Mom Err Z' (all 0.000000) and 'MTB 1 Cmd', 'MTB 2 Cmd', and 'MTB 3 Cmd' (all 0.000000); and 'SA Gimbal Commands' with 'SA Gimbal' (0.000000). Each error value has a 'Plot' button next to it.

- 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

# 42 Sim: Additional Configuration Options

---

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



# 42 Sim: Set Wheel Target Momentum

**F42 WHL\_TGT\_MOM\_CMD\_SCR**

**Set Wheel Target Momentum Command**

**Current Target Momentum**

Wheel 1	Wheel 2	Wheel 3
0.000000	0.000000	0.000000

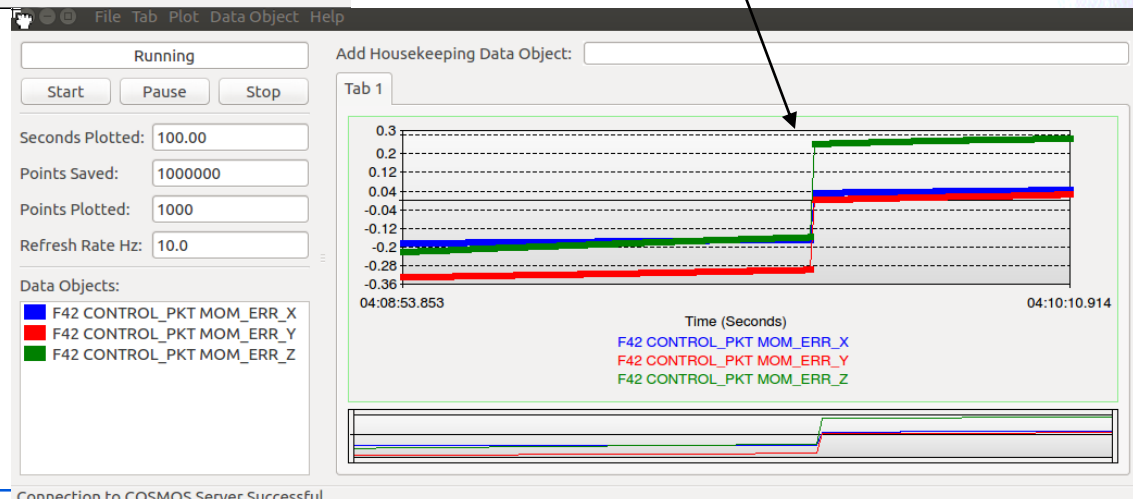
**Command Target Momentum**

Wheel 1	Wheel 2	Wheel 3

Enter a floating point value for each axis. 0.0 will be sent for invalid values.

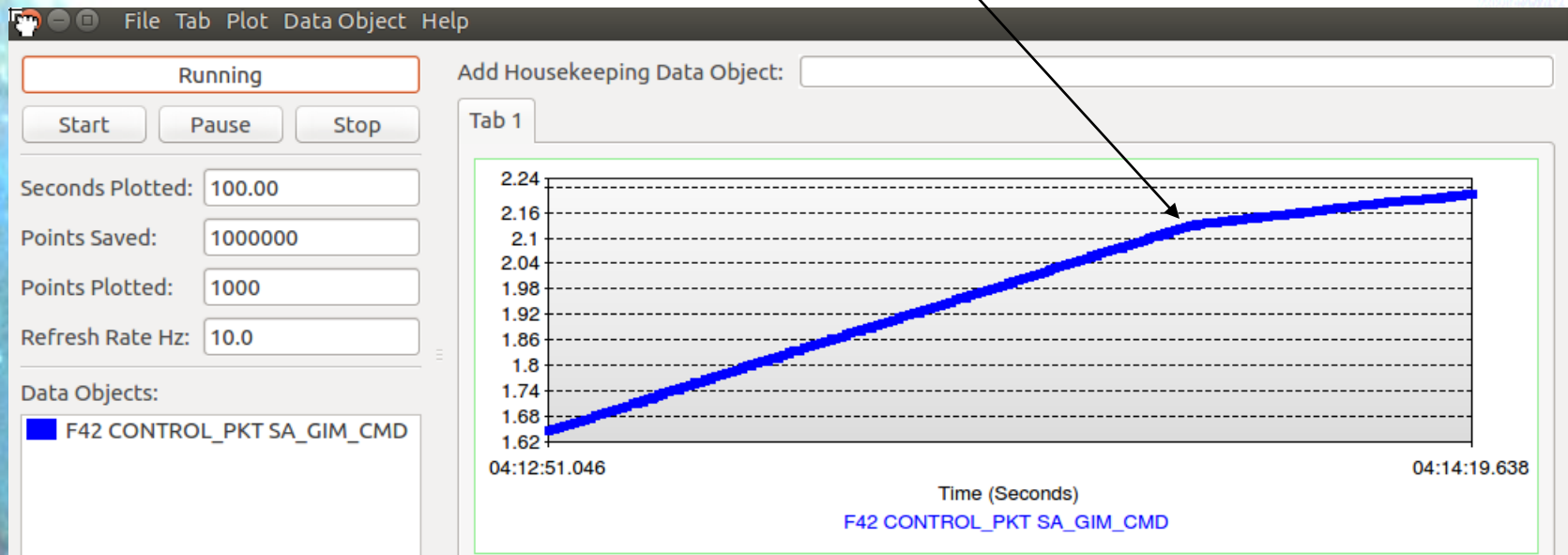
**Send** **Cancel**

- The controller allows a non-zero (default) momentum to be stored in the wheels
- Enter new values and click **<Send>** to change the values
- The plot below shows a jump in momentum errors when new targets were selected



# 42 Sim: Configure SunValid

- 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.



# 42 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.

---

**Manage Applications**

**Needs 2.0 Updates**



# Tools: Create Application

CFS\_KIT APP\_CREATE\_SCREEN

Create New Application

```

cfs
|- apps
| |- example
|- osk def
| |- cpul_cfe_es_startup.scr
| |- targets.cmake
cosmos
|- config
| |- targets
| | |- EXAMPLE
| |   |- cmd_tlm
|- tools
| |- cmd_tlm_server.txt
|- lib
| |- message_ids.rb

Generated by APPGen
Manually edited by user
Definitions assumed by AppGen

```

- Create App

 Launch tool to create new app
- Edit cmake

 Add app file to cmake target list TGT1\_APPLIST
- Edit ES Startup

 Add app to cFE Executive Service startup script
- Stop cFS/Server

 Stop cFS and COSMOS cmd-tlm server
- Build cFS

 Run cmake to build new app
- Start COSMOS Server

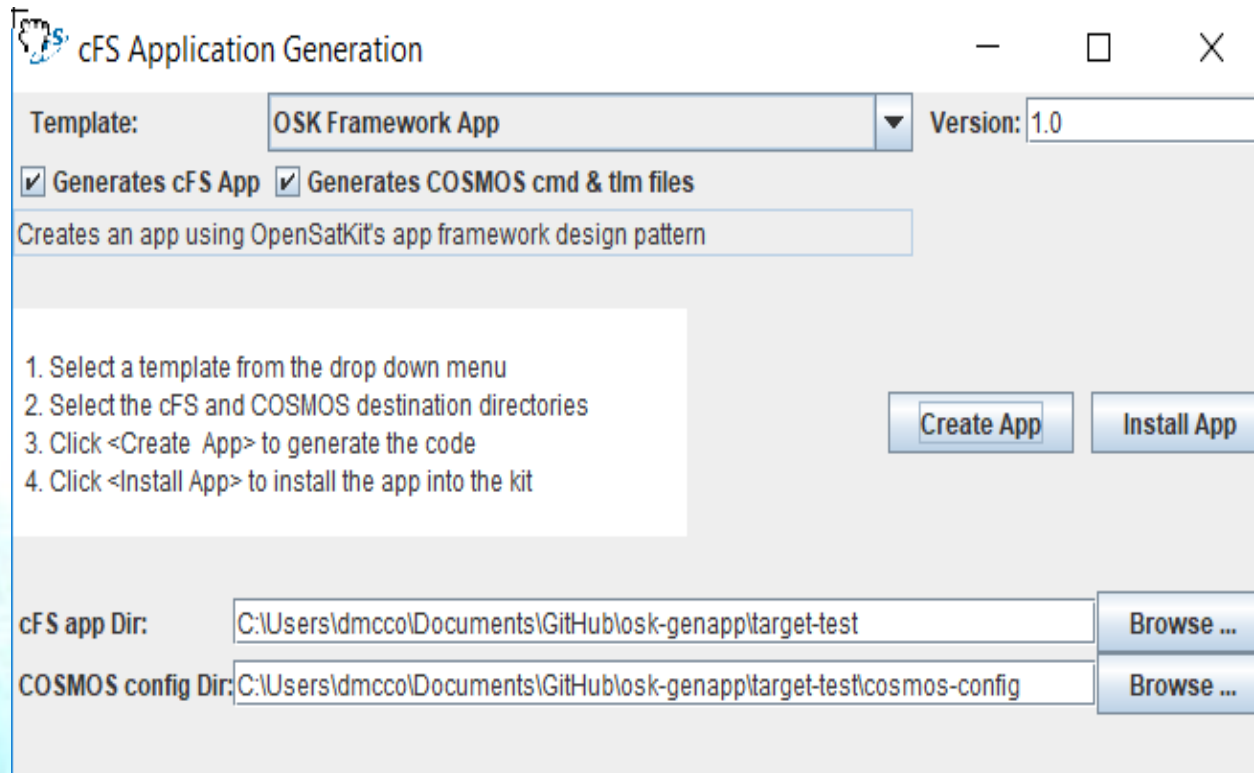
 Manually start cmd-tlm server from COSMOS launcher
- Start cFS

 Start cFS with new app

See next slide

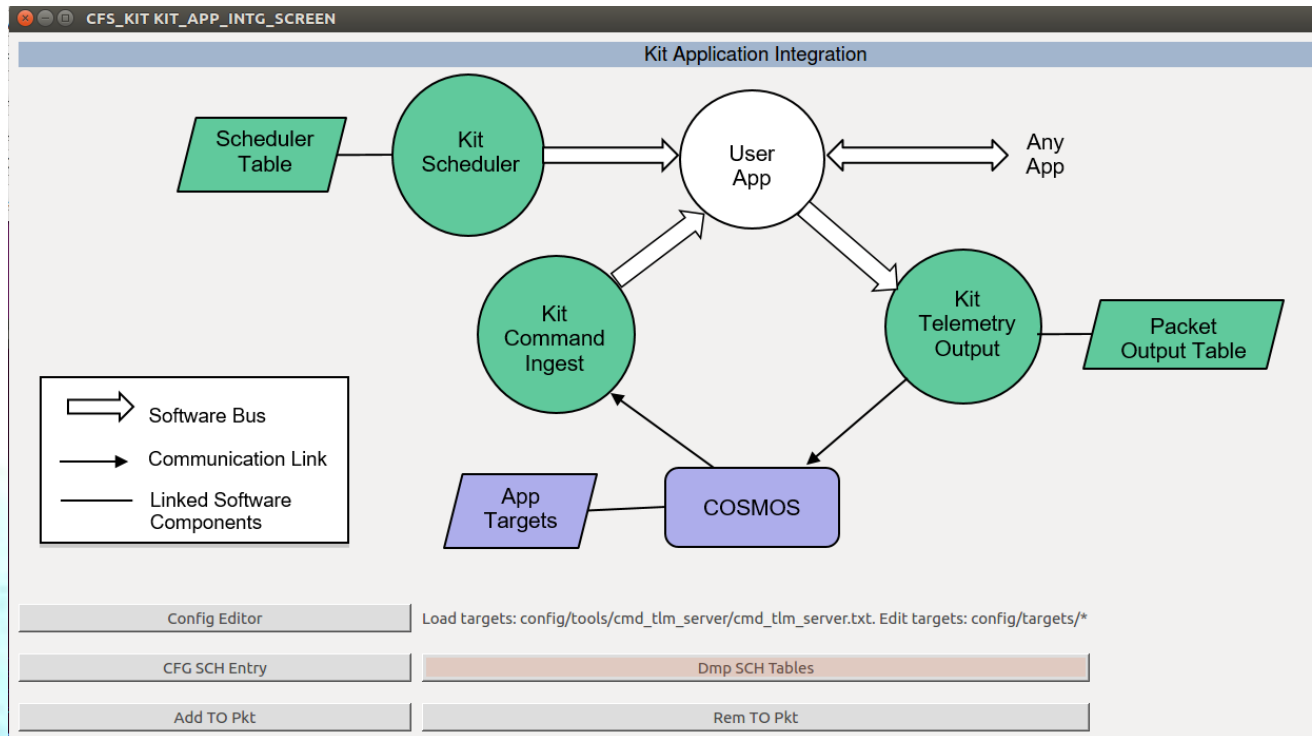
- Seven quick steps and a new app is created and integrated into the kit

# Tools: Create Application



- Follow the instructions in the center of the dialogue. Create app generates the fsw source/make files, the cosmos target, and edits the COSMOS cmd-tlm-server config file.
- *<Install App>* has not been implemented. Follow the instructions on the previous slide

# Kit App Integration



- Goal is to provide easy access to COSMOS, KIT\_TO, and KIT\_SCH to integrate a new app

# Extending OSK



# Tools: Benchmarks

---

**Coming Soon...**

# Tools: Performance Monitor

- Capture FSW performance data using screen
- Download file and <Launch Analysis 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  Mode  Trigger Count

**Masks**

Filter

Trigger

**Log Stats**

Start  End

Count  Remaining to Write

**File Transfer**

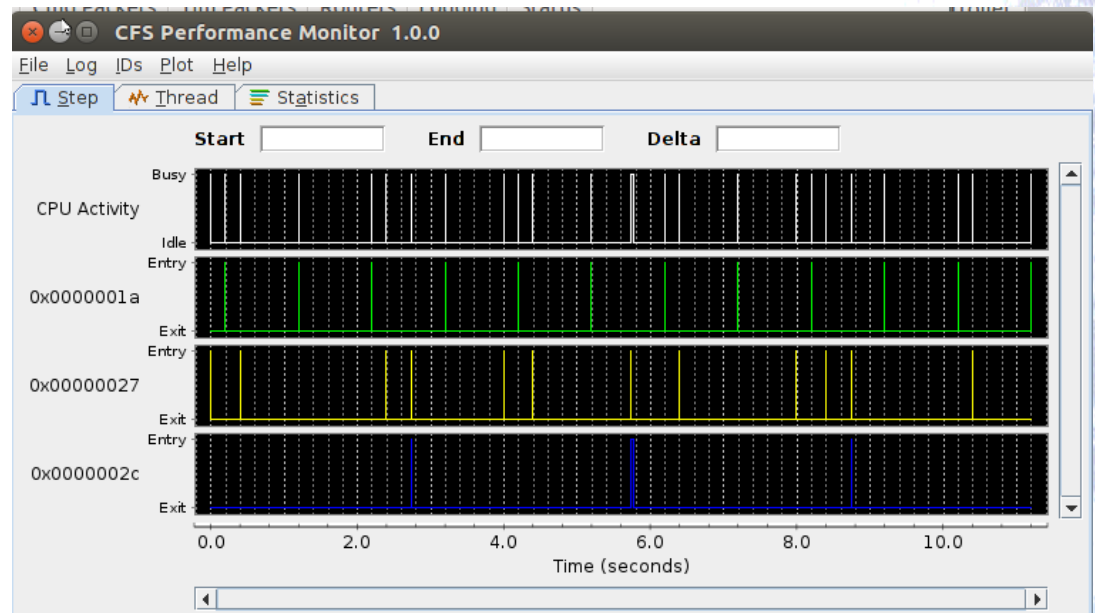
Put File	Get File
----------	----------

PUT\_FILE\_COUNT:  GET\_FILE\_COUNT:

Ground Working Directory

Flight Working Directory

Flight Event Messages



# Tools: PiSat Control

---

- This requires a PiSat which is currently not in the public domain

# Demos

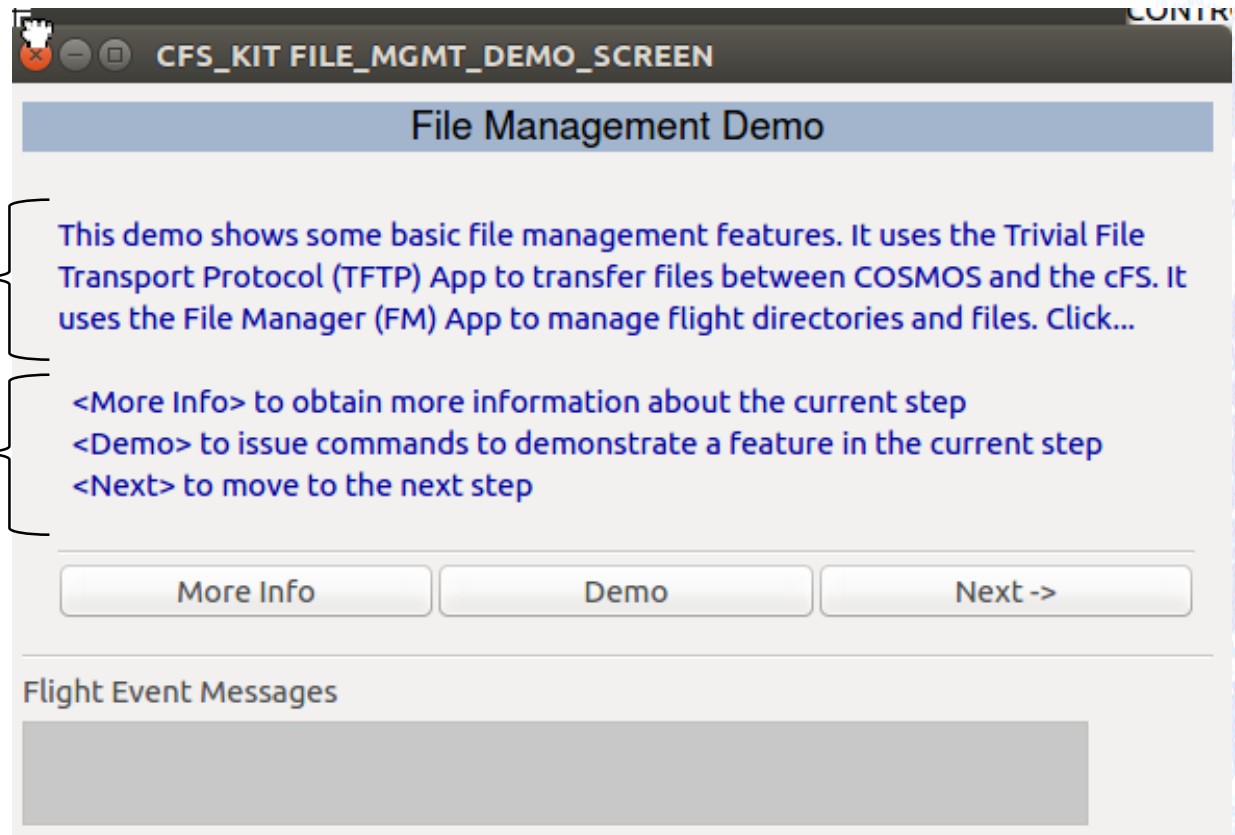


# Demo Structure – FM Example (1 of 2)

- Each demo follows a common user screen configuration

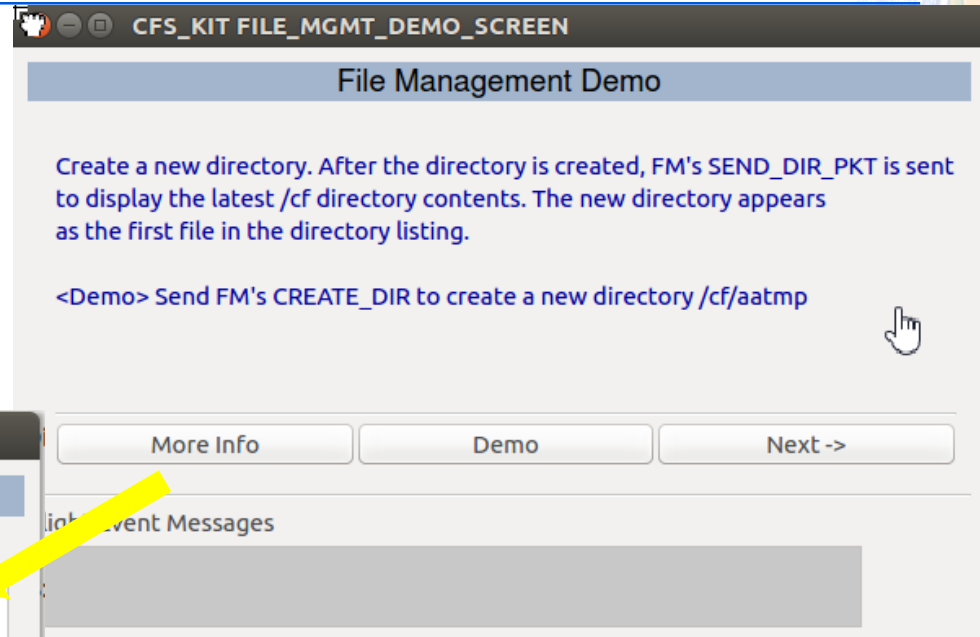
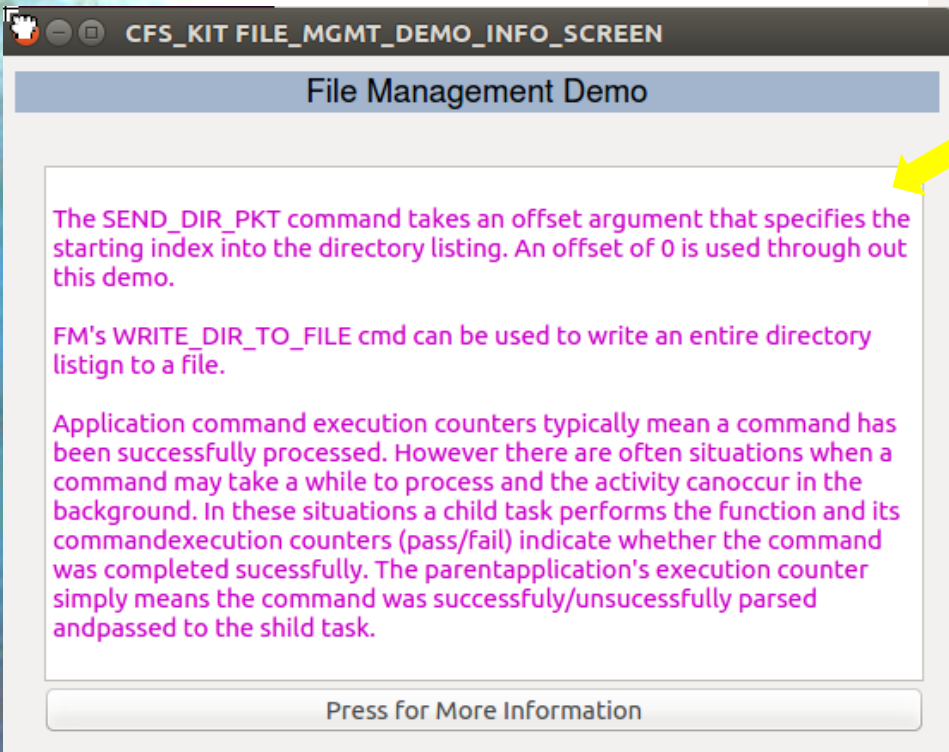
Description of  
current step

Button usage  
description



# Demo Structure – FM Example (2 of 2)

**<More Info> provides detailed context-specific information**



# Application Functional Screens

# File Management

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 Transfer	
Put File	Get File
PUT_FILE_COUNT: 0	GET_FILE_COUNT: 0
Ground Working Directory	
<input type="text"/>	
Flight Working Directory	
<input type="text"/>	

Event Messages

- <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

CFS\_KIT TABLE\_MGMT\_SCREEN

### Table Management

Load Table	Validate	Activate
Abort Load	Dump Table	Display Table

#### Table Registry

Display Registry Write Registry to File

#### 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:

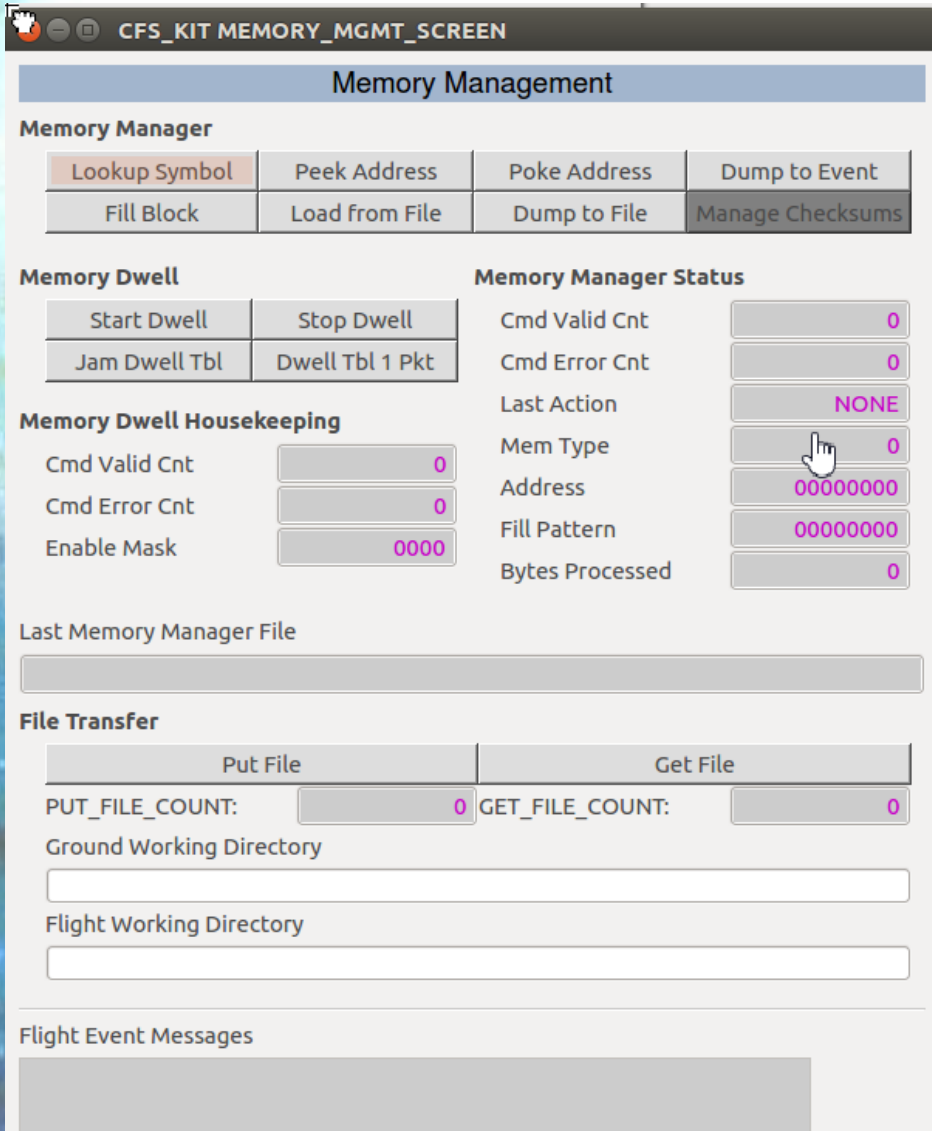
#### File Transfer

Put File	Get File
PUT_FILE_COUNT: <input type="text"/> 0	GET_FILE_COUNT: <input type="text"/> 0
Ground Working Directory <input type="text"/>	
Flight Working Directory <input type="text"/>	

#### 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

Lookup Symbol	Peek Address	Poke Address	Dump to Event
Fill Block	Load from File	Dump to File	Manage Checksums

Memory Dwell

Start Dwell	Stop Dwell
Jam Dwell Tbl	Dwell Tbl 1 Pkt

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

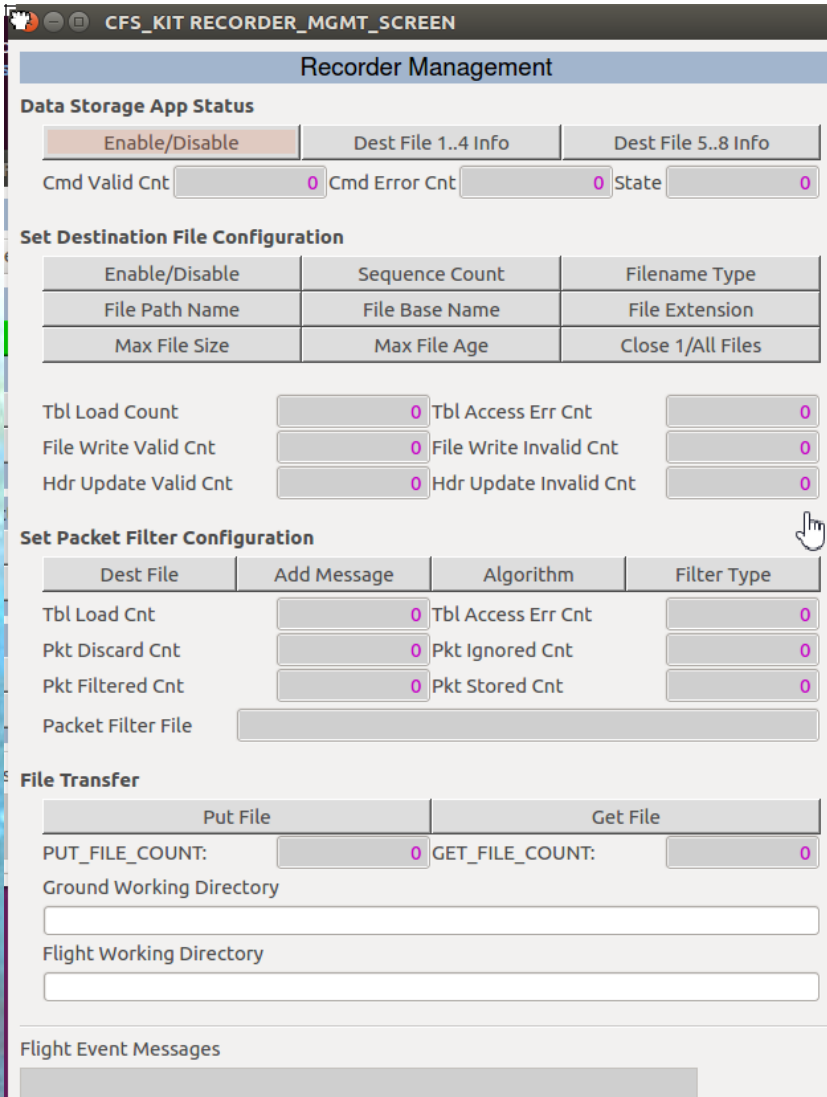
File Transfer

Put File	Get File
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



Recorder Management

Data Storage App Status

Enable/Disable

Dest File 1..4 Info

Dest File 5..8 Info

Cmd Valid Cnt

0

Cmd Error Cnt

0

State

0

Set Destination File Configuration

Enable/Disable	Sequence Count	Filename Type
File Path Name	File Base Name	File Extension
Max File Size	Max File Age	Close 1/All Files

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

Dest File	Add Message	Algorithm	Filter Type
-----------	-------------	-----------	-------------

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	Get File
----------	----------

PUT\_FILE\_COUNT:

0

GET\_FILE\_COUNT:


0

Ground Working Directory

Flight Working Directory

Flight Event Messages

# Autonomy Management


CFS\_KIT AUTONOMY\_MGMT\_SCREEN

## Autonomy Management

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

Start RTS	Stop RTS	Enable RTS	Disable RTS
Start Group	Stop Group	Enable Group	Disable Group
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	0000000
CMD Err Cnt	0000	Err RTS#	0000	Err RTS Offset	0000

### Limit Checker(LC) App

Reset WP Stats	Reset AP Stats	Set AP State	Set AP Prem Off
Set App State	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



App Management

App Summary

App/Task Registry

Enable App Events

Disable App Events

Add KIT\_TO Msg

Start App

Stop App

Reload App

Get App Info

Create App Tool

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

Get File

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>*

# COSMOS Extras

