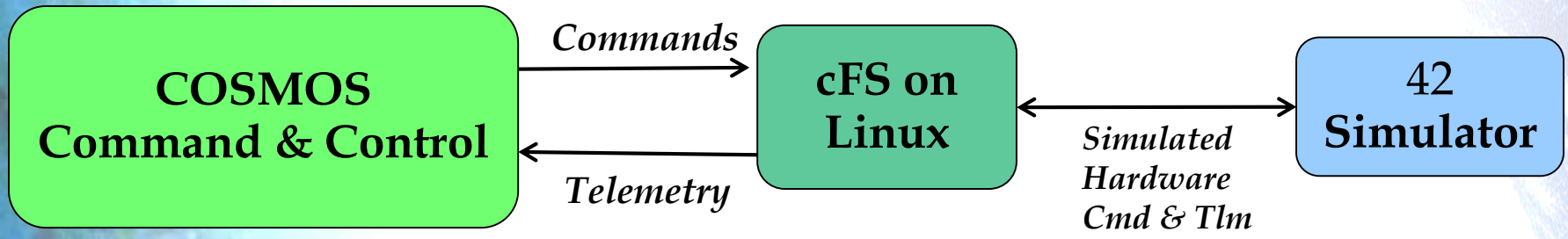
OpenSatKit

Tour

Introduction

- The primary goal of OpenSatKit (OSK) is to provide a core Flight System (cFS) development and run time environment that can be used to learn about the cFS and to serve as a starting point for a new project
- 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

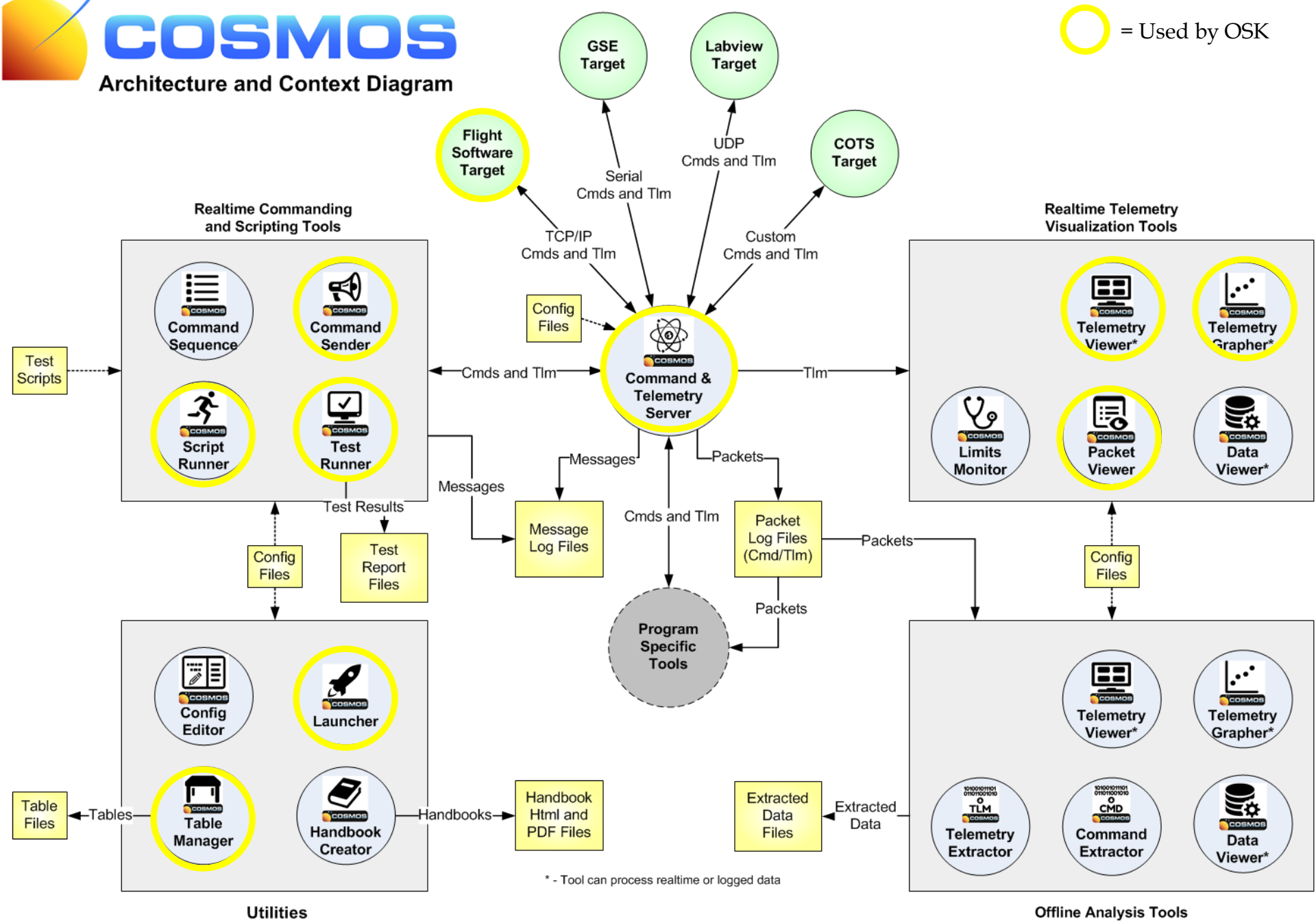


COSMOS Overview

- Ball Aerospace's COSMOS provides a set of tools each with a Graphical User Interface (GUI) that runs as a separate Linux process
- The next slide shows the top-level COSMOS architecture with each tool used by OSK highlighted with a yellow circle
- OSK implements extensive COSMOS configurations and customizations so COSMOS can serve as the primary OSK user interface
- After the architectural slide each COSMOS tool used by OSK is briefly introduced
- The cFS runs as a separate Linux process in a terminal window
- NASA's 42 Simulator is a spacecraft and environmental dynamic simulator
  - Runs as a separate process as part of a demo
  - A FSW control app "F42" implements a control algorithm

COSMOS

Architecture and Context Diagram



COSMOS Tools (1 of 3)

- **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.

COSMOS Tools (2 of 3)

- **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*

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

COSMOS Tools (3 of 3)

- **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.*

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

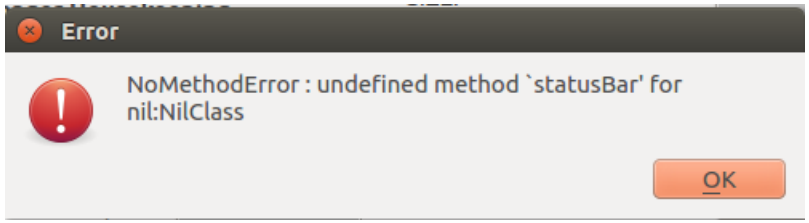
- **OSK scripts in *OpenSatKit/cosmos/lib* extend COSMOS scripting API**
  - API documentation is under development. See code for details

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.**

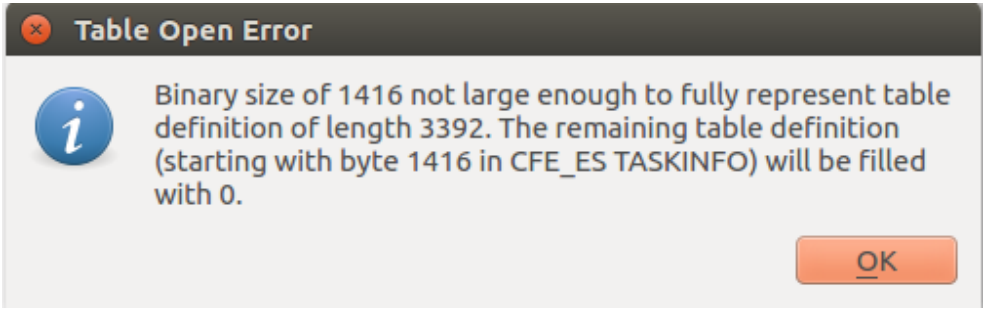


- **The FSW terminal window may display start and stop "FlyWheel" messages**
  - OSK is a non-realtime environment so the cFE time service is warning that's it's not operating within its real-time precision limits relative to a 1Hz timer
  - OSK is designed to help users learn functional features and 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)

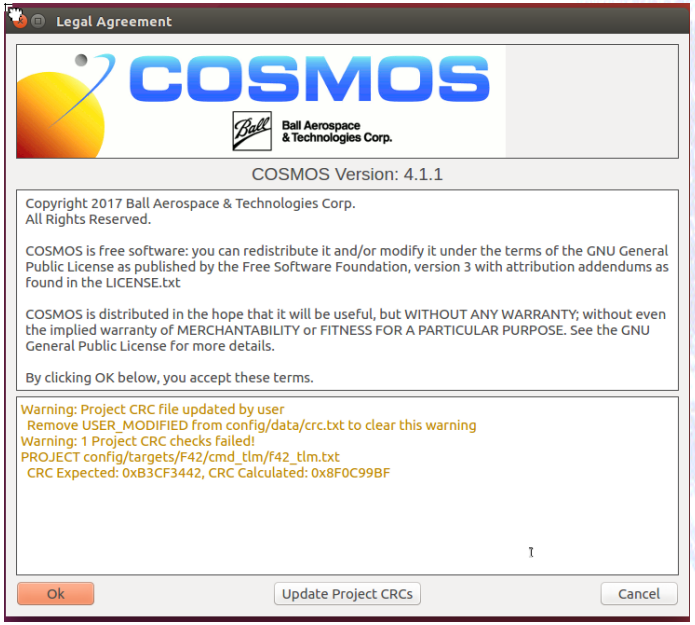
- Some cFS binary files are variable length. The Table Manager definition files support fixed length files, therefore you may see an error dialog stating the file doesn't contain all of the records. This message is from cFE Executive Service Task Information file.



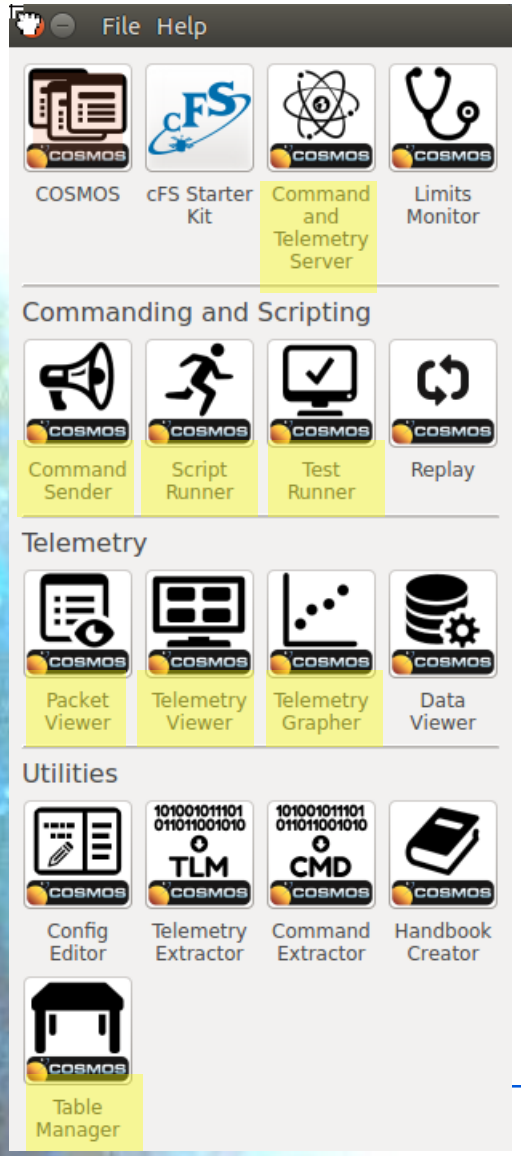
Getting Started

Running OpenSatKit (1 of 2)

- Open a terminal window (Ctrl-Alt-t)
- Change directory to cosmos
  - [~] cd OpenSatKit/cosmos
- Start COSMOS
  - [~/OpenSatKit/cosmos]ruby Launcher
  - You'll see a screen similar to below. Select <OK>



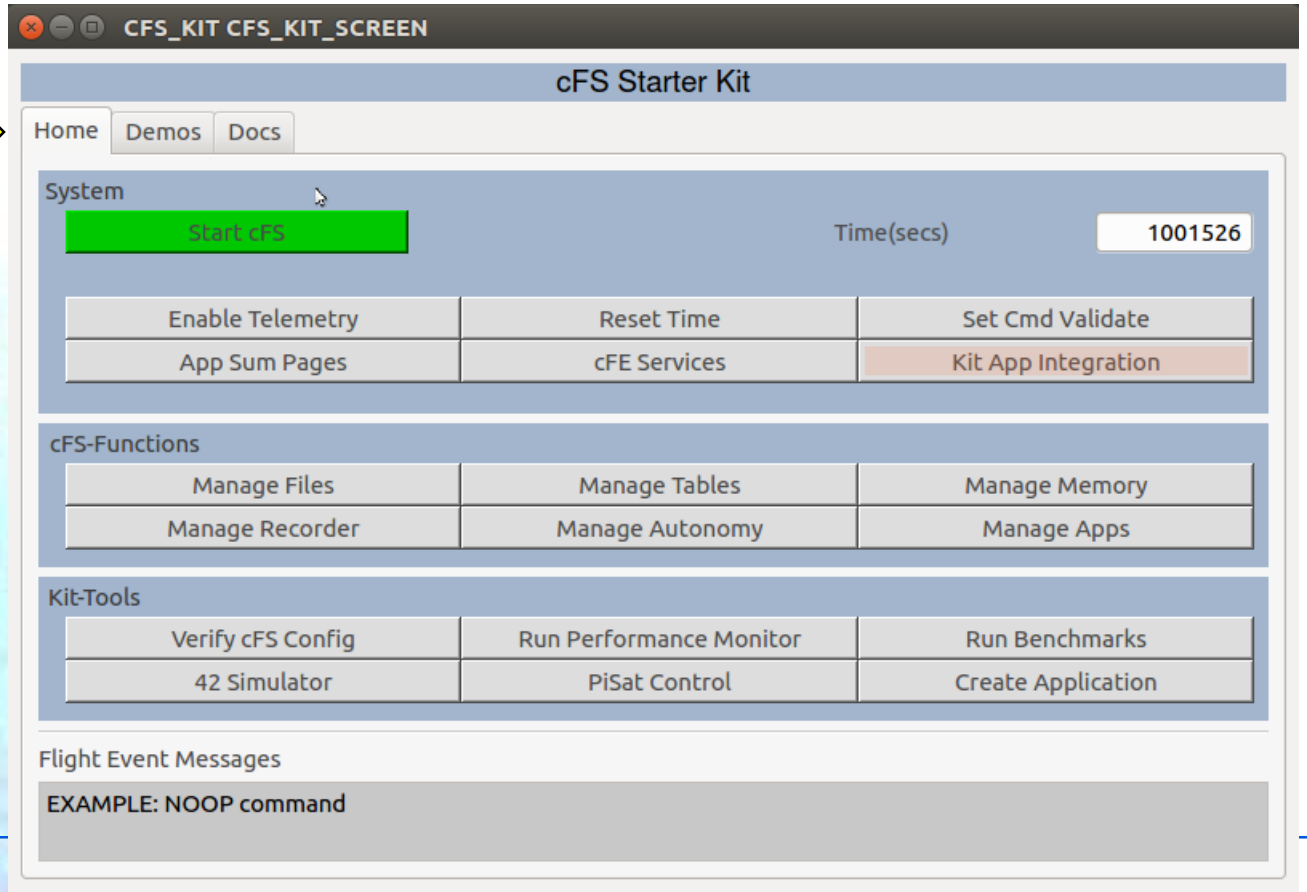
Running OpenSatKit (2 of 2)



- **Select "cFS Starter Kit" with a single click**
  - This launches COSMOS's Command and Telemetry Server and Telemetry Viewer
  - You can minimize them, but don't close them
- **The main kit windows is on the next slide**
- **Shaded tool titles indicate the COSMOS tools used by OSK**

Main OSK Window

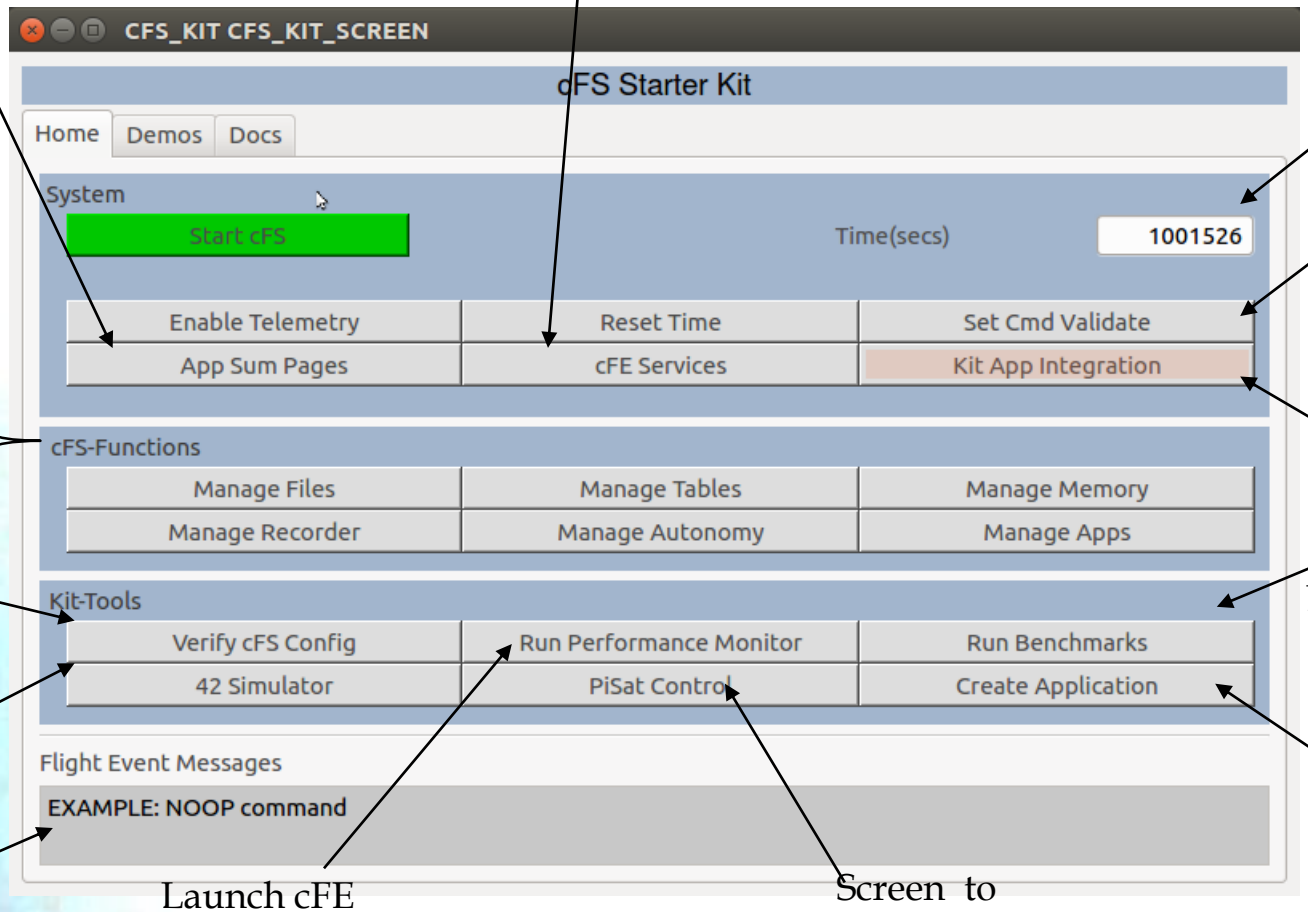
- Three tabs ***Home***, ***Demos***, and ***Docs*** provide the top-level organization
- ***Home*** contains primary interactive user interface for the user to engage with the cFS
- ***Demos*** are predefined screens and scripts that illustrate cFS features and tool function
- ***Docs*** contains links to documentation and tutorials. Tutorials are instructional in nature, more interactive than demos, and can be extended by the user.



OSK Home Tab

Pages summarizing all apps in the kit

Page summarizing cFE service telemetry and drop down menus to all cFE commands



Current FSW time

Enable/Disable OSK's cmd counter validation

Add App To Kit

Launch screens to manage a functional goal

Prototype Benchmark App

Run test script

Closed-loop control sim

Create "Hello World" App

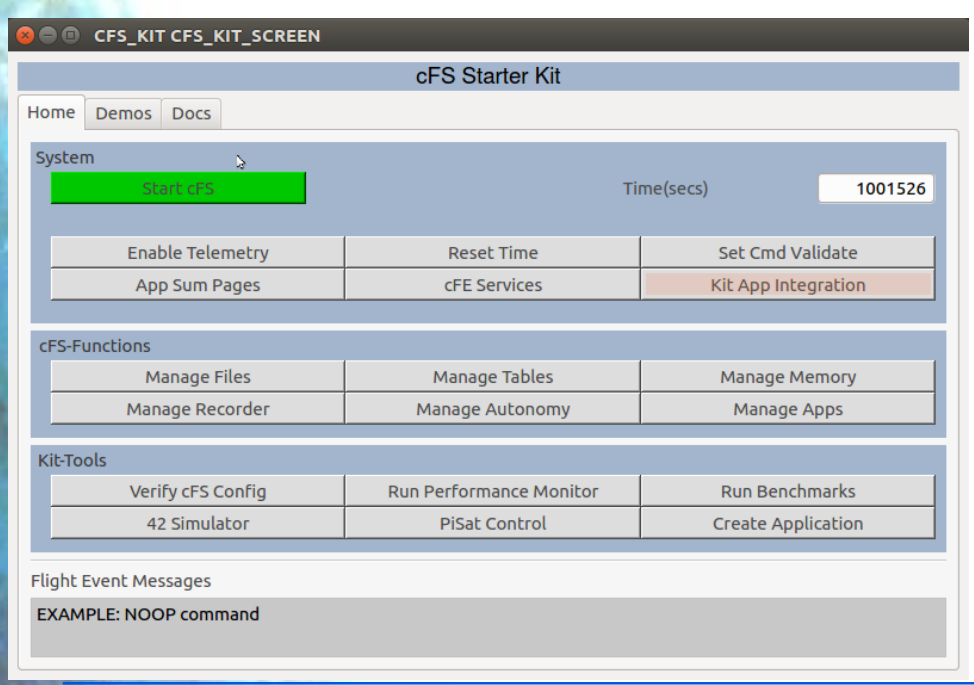
Display last FSW event message

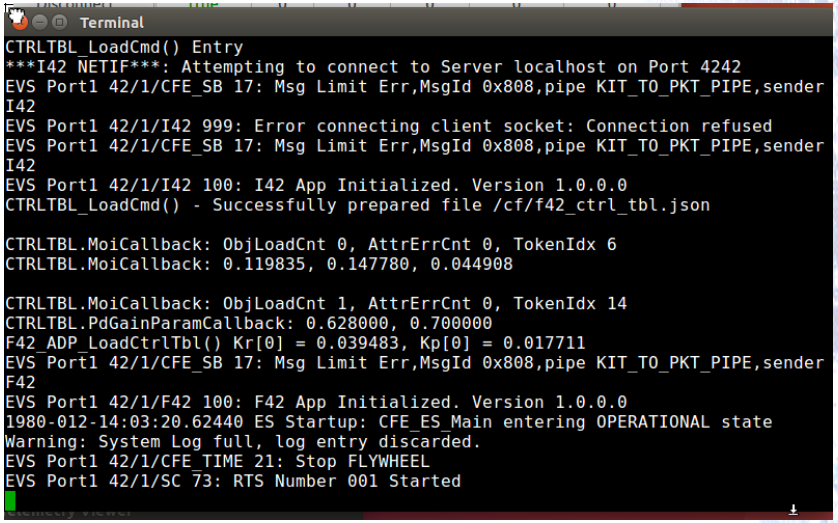
Launch cFE Perf Mon Tool

Screen to manage "PiSat"

Start the Flight Software (FSW)

- **Click <Start cFS> to run the FSW**
  - A new terminal window is created
  - 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 <Enable Telemetry>





App Summary Pages

CFS_KIT APP_CFS_SUMMARY_SCREEN

cFS Applications

App Name	Seq Cnt	Cmd Valid Cnt	Cmd Error Cnt
CFE_ES - cFE Exceutive Service	51966	0	0
CFE_EVS - cFE Event Service	51966	0	0
CFE_SB - cFE Software Bus	51966	0	0
CFE_TBL - cFE Table Service	51966	0	0
CFE_TIME - cFE Time Service	51966	0	0
CS - Checksum	51403	0	0
DS - Data Storage	51403	0	0
FM - File Manager	51403	0	0
HS - Health & Safety	51403	0	0
LC - Limit Checker	51403	0	0
MD - Memory Dwell	51403	0	0
MM - Memory Manager	51403	0	0
SC - Stored Command	51403	1	0

CFE_ES | Noop | Reset Counters

Each app sends a housekeeping packet containing

- Incrementing sequence counter
- Count of valid and invalid commands received

CFS_KIT APP_KIT_SUMMARY_SCREEN

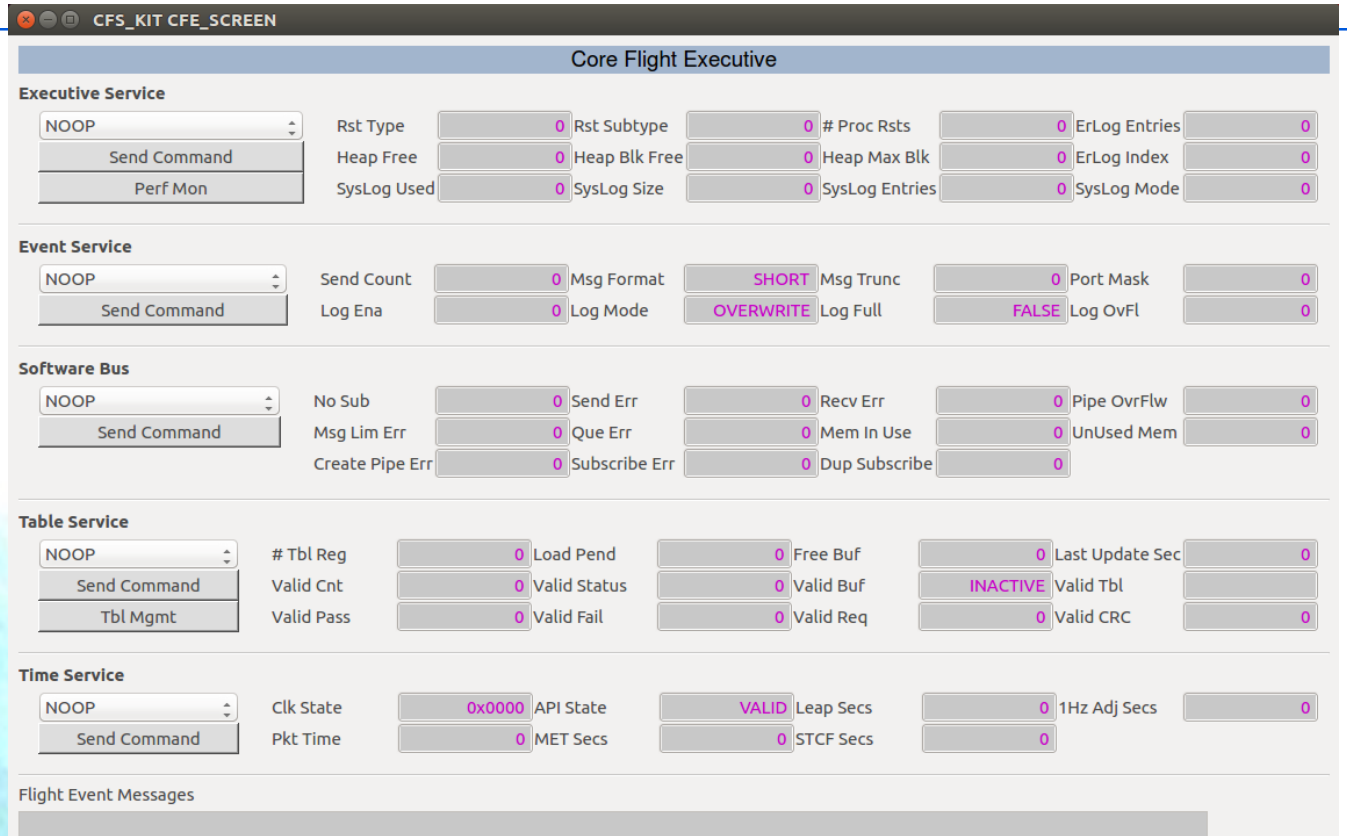
Kit Applications

App Name	Seq Cnt	Cmd Valid Cnt	Cmd Error Cnt
BM - Benchmark	51408	0	0
F42 - 42 Simulator FSW Controller	52911	0	0
HC - Heater Control	51408	0	0
HSIM - Heater Simulation	51408	0	0
I42 - 42 Simulator Interface	52911	1	0
KIT_CI - Command Ingest	51408	0	0
KIT_SCH - Scheduler	51408	0	0
KIT_TO - Telemetry Output	51408	1	0
TFTP - Trivial File Transfer Protocol	54224	0	0

BM | Noop | Reset Counters

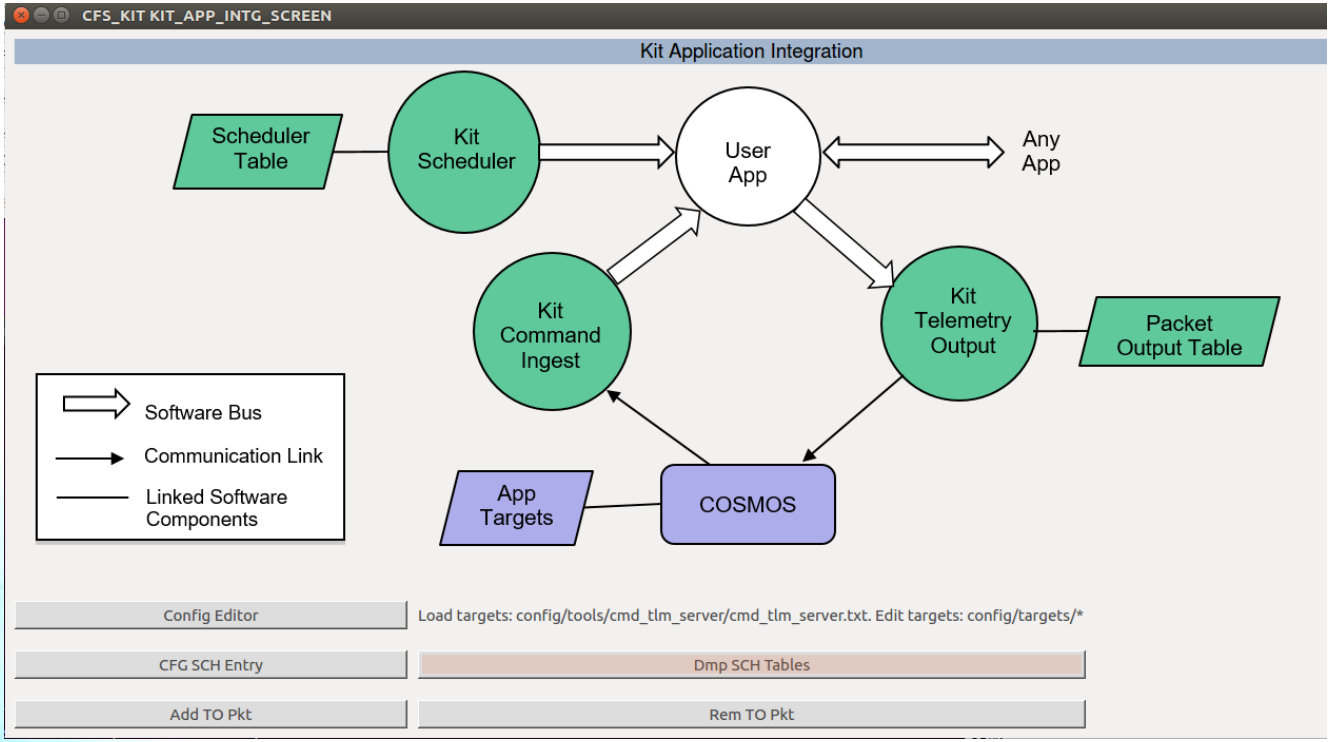
Convenient drop down let's you send *Noop* and *Reset Counter* commands to each app

cFE Service Page



- **Easy access to all cFE app service commands**
  - Select command from drop down and then click <Send Command> to be prompted for command parameters
- **Key Housekeeping telemetry points displayed for each service**

Kit App Integration



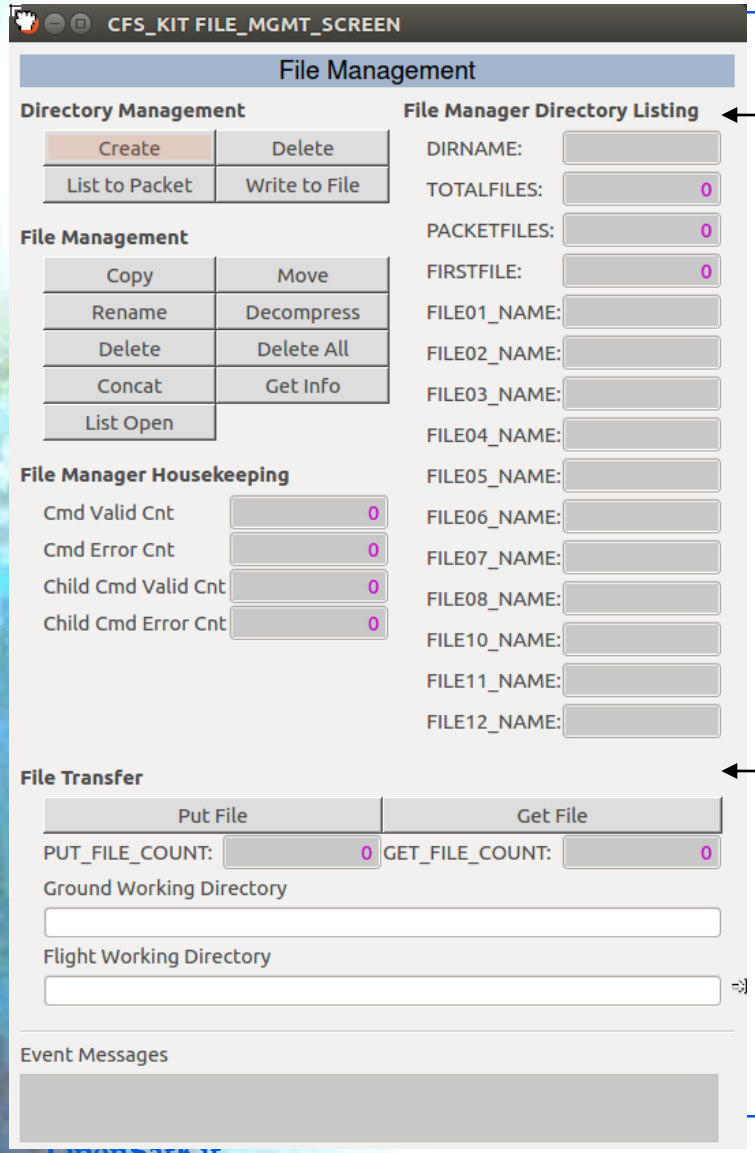
- **Goal is to provide easy access to COSMOS, KIT_TO, and KIT_SCH to integrate a new app**

Manage cFS Functions

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 fiel 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**

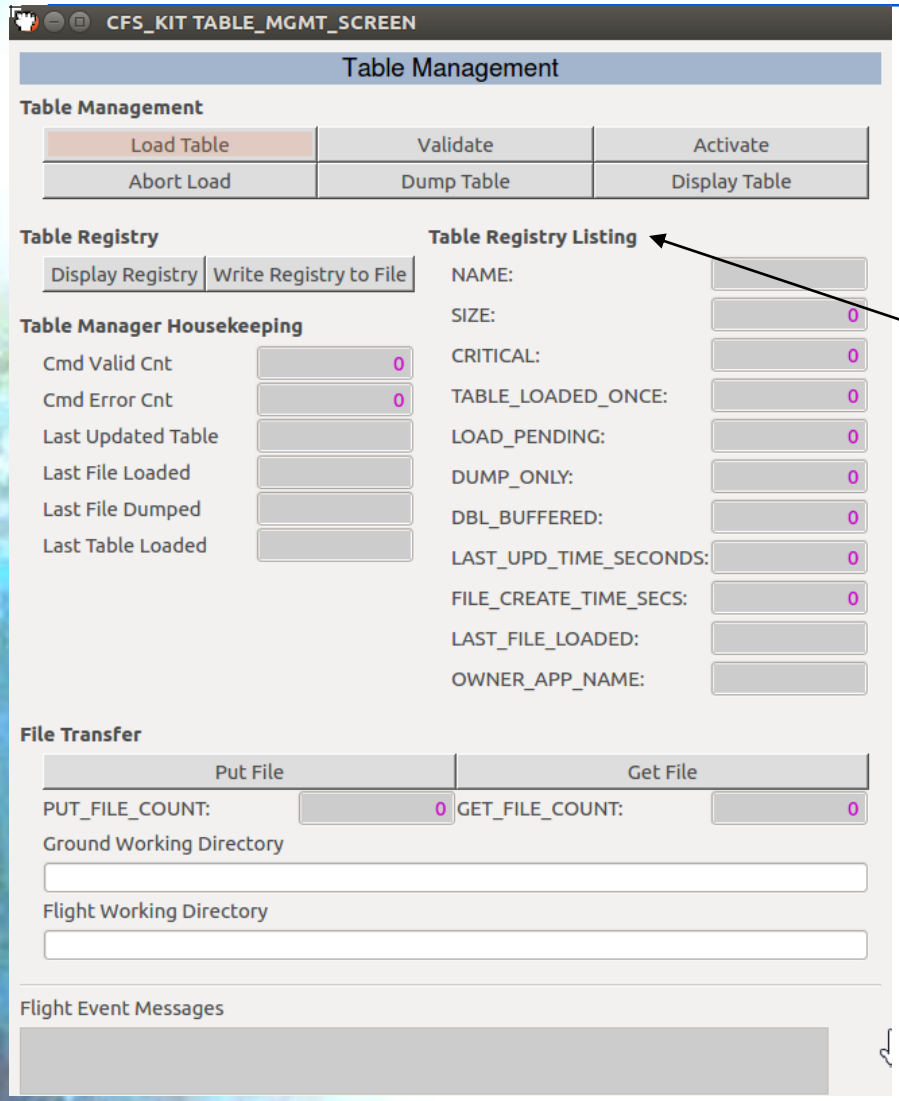
File Management



- <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

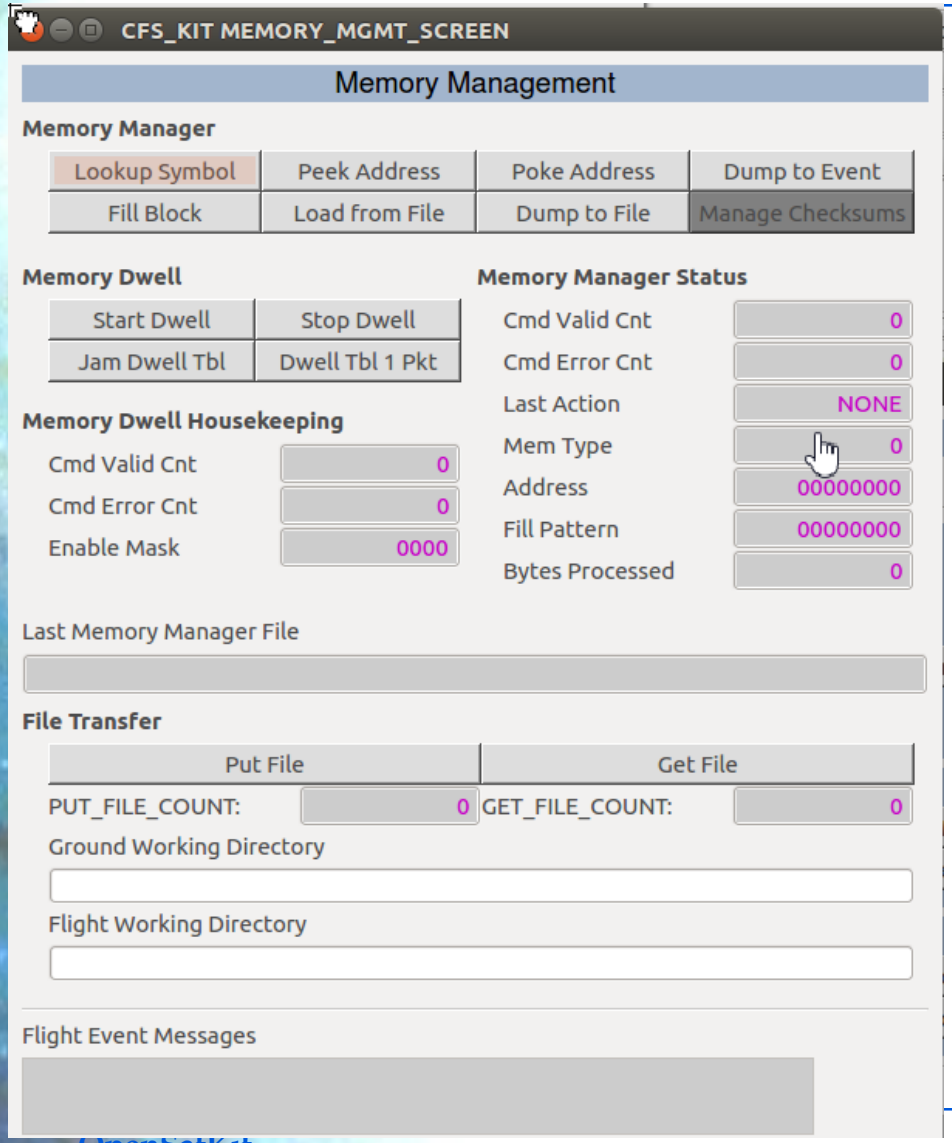


- Load a new FSW table
  <Put File> transfers file from ground to flight
  <Load Table> into table buffer
  <Validate> table via app validation function
  <Activate> new table

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

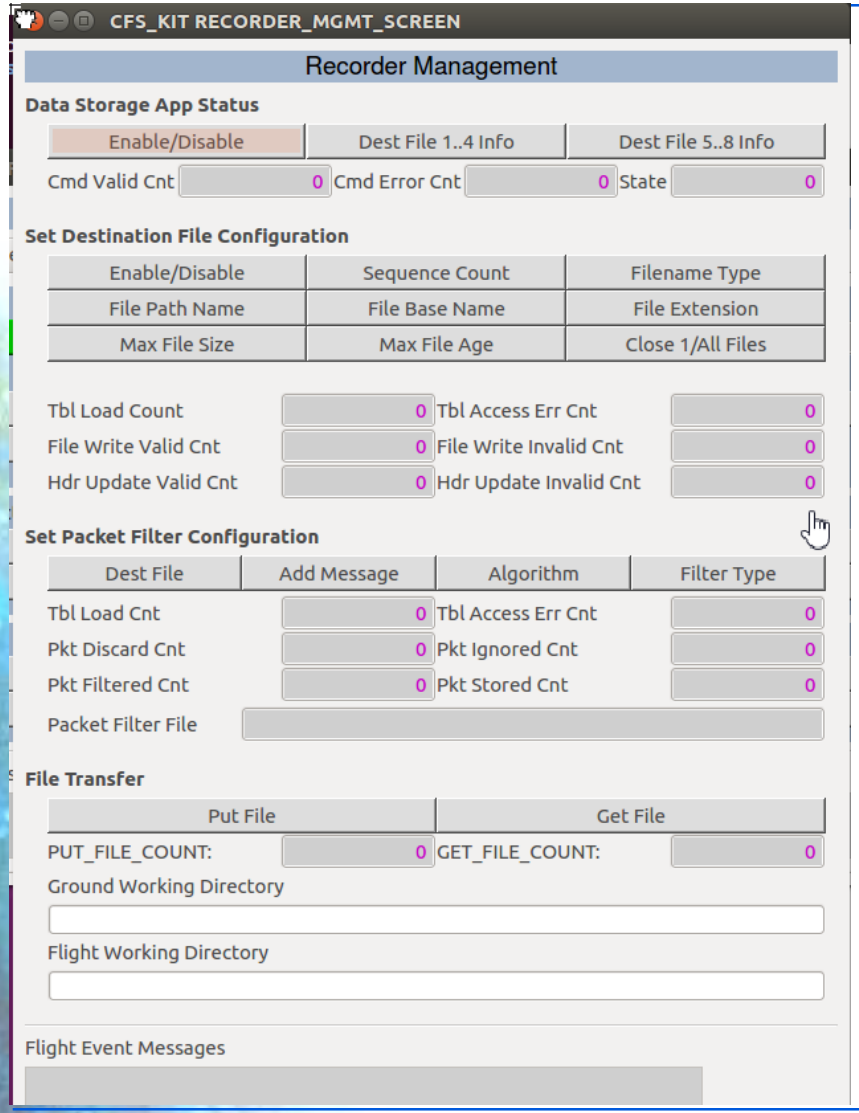
- Dump and display FSW table
  <Dump Table> to onboard file
  <Get File> transfers file from flight to ground
  <Display Table> launches COSMOS Table Manager to view file. Requires binary file definition.

Memory Management

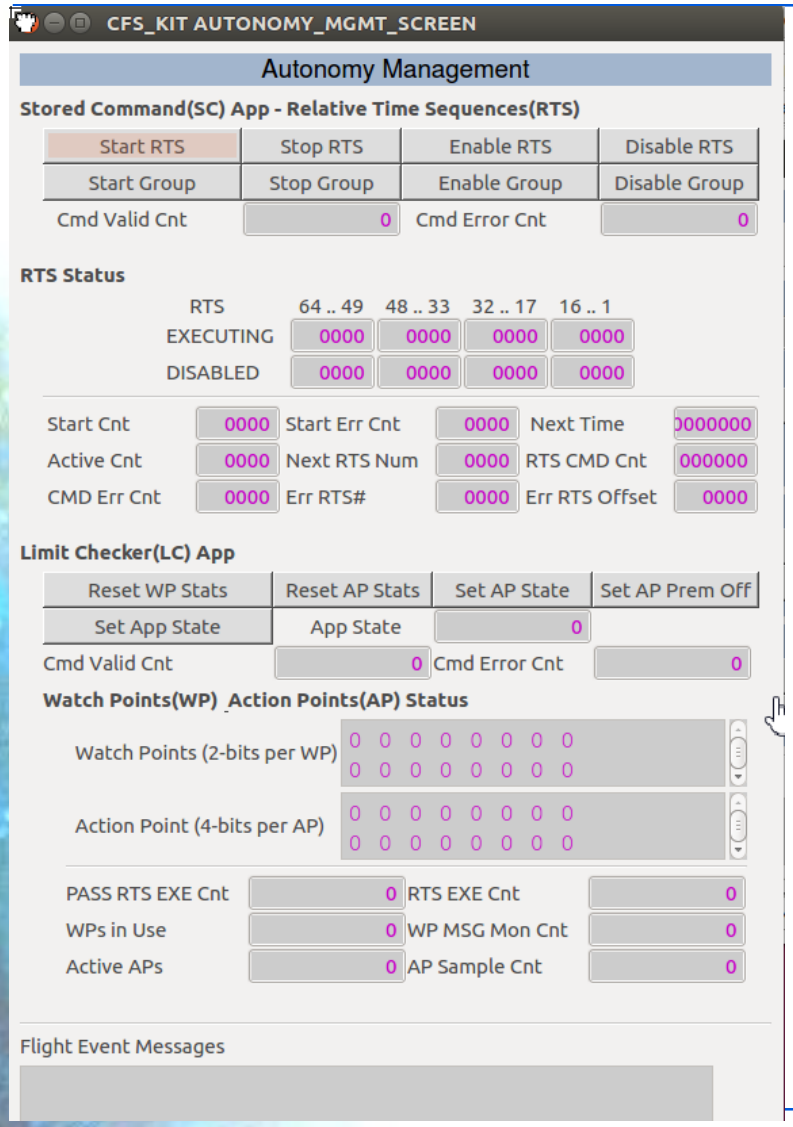


- Memory Manager (MM) and Memory Dwell (MD) apps are typically used for inflight maintenance.
- MM commands allow direct access to any memory location
- MD generates telemetry packets that contain the contents of table-specified memory locations
  - Only 1 dwell table telemetry packet is defined
  - *<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

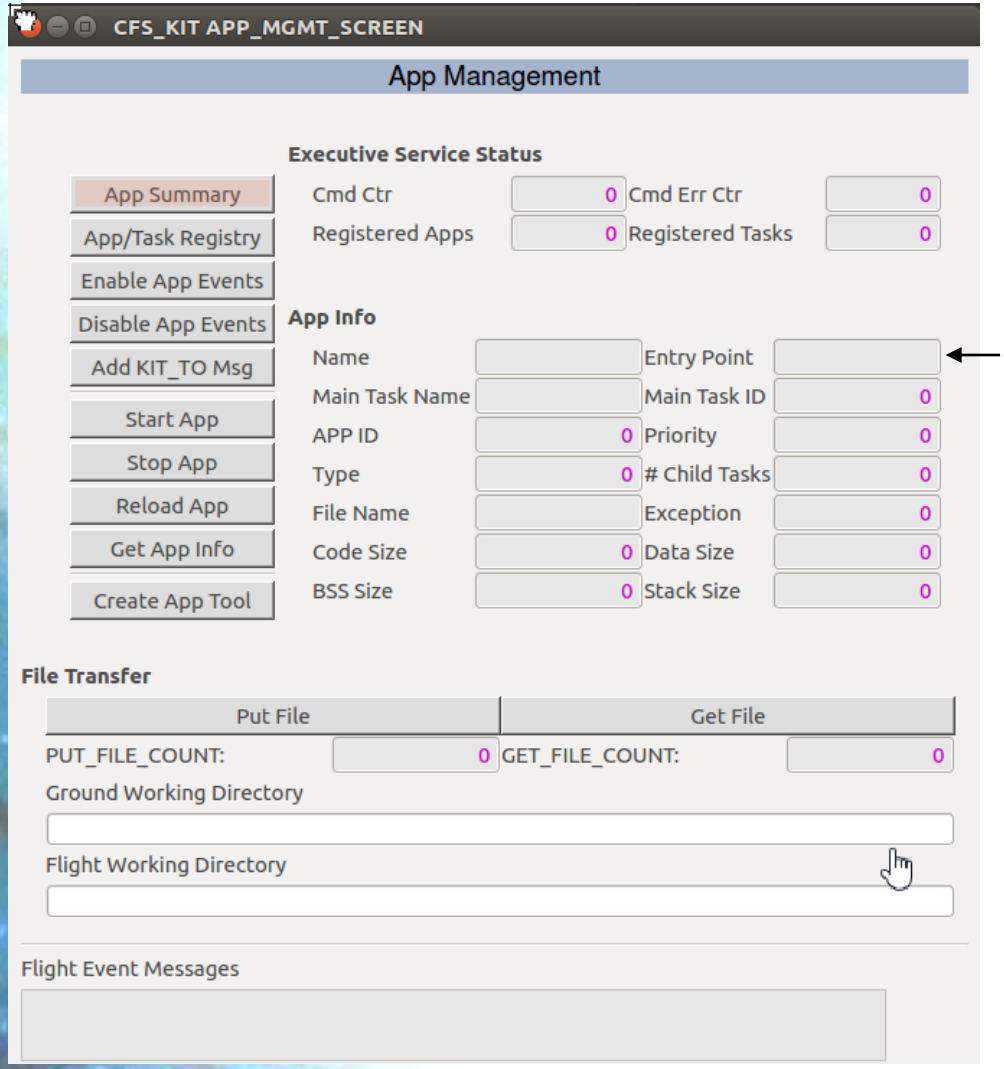
Recorder Management



Autonomy Management



Application Management



- *<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>*

Tools

Tools: Verify cFS Configuration

File Edit Search Script Help

kit_test_main.rb

Paused | Go | Pause | Stop

kit_test_main.rb

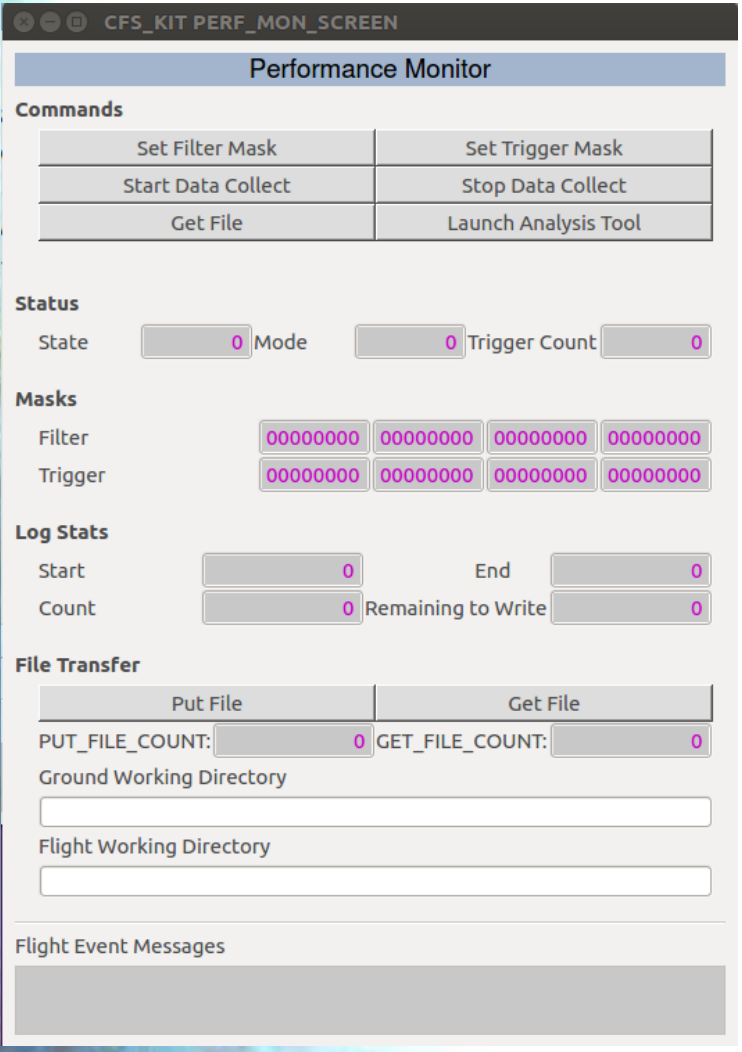
```
18 require 'kit_test_cfe_tbl'
19 require 'kit_test_cfe_time'
20
21 require 'kit_test_cs'
22 require 'kit_test_ds'
23 require 'kit_test_fm'
24 require 'kit_test_hs'
25 require 'kit_test_lc'
26 require 'kit_test_md'
27 require 'kit_test_mm'
28 require 'kit_test_sc'
29
```

Script Output:

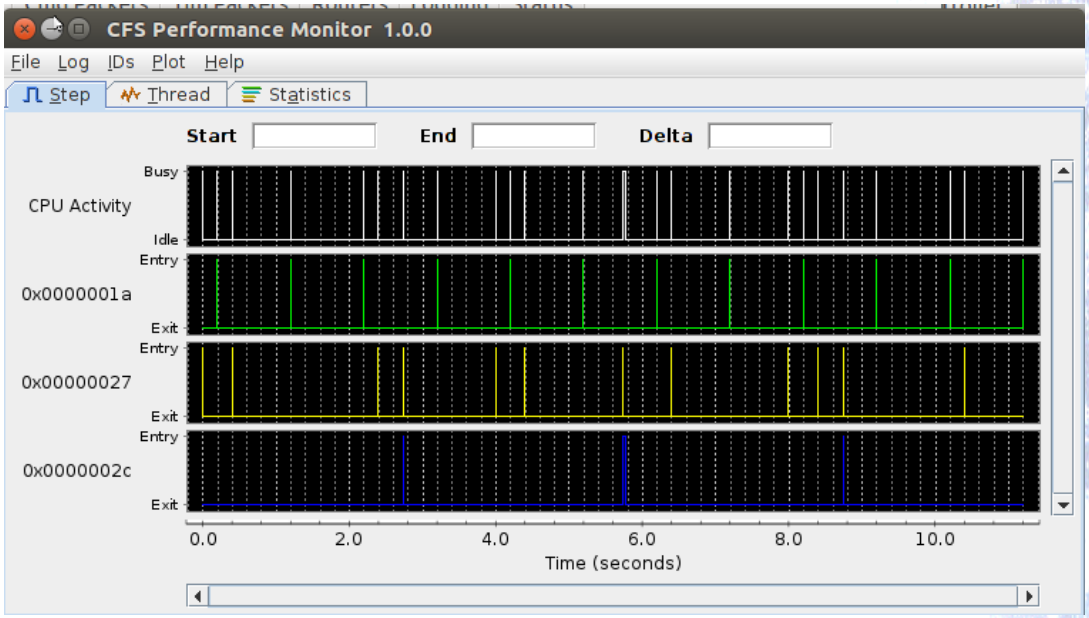
```
2018/09/16 18:39:03.769 (SCRIPTRUNNER): Starting script: kit_test_main.rb
2018/09/16 18:39:05.474 (SCRIPTRUNNER): User pressed Pause
```

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

Tools: Performance Monitor



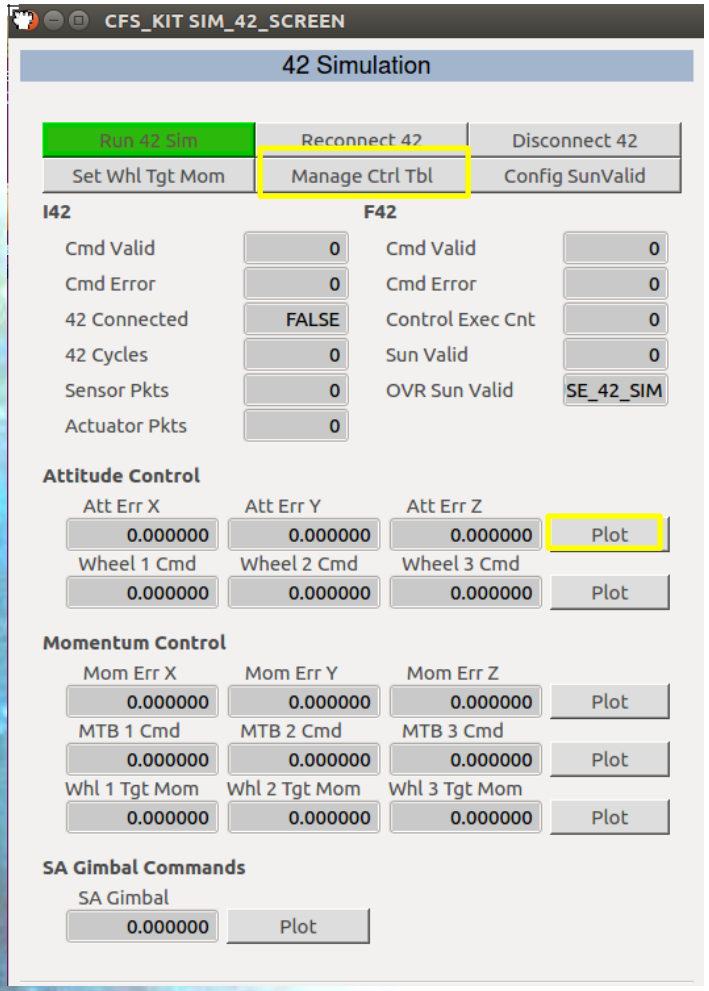
- Capture FSW performance data using screen
- Download file and <Launch Analysis Tool>



Tools: Benchmarks

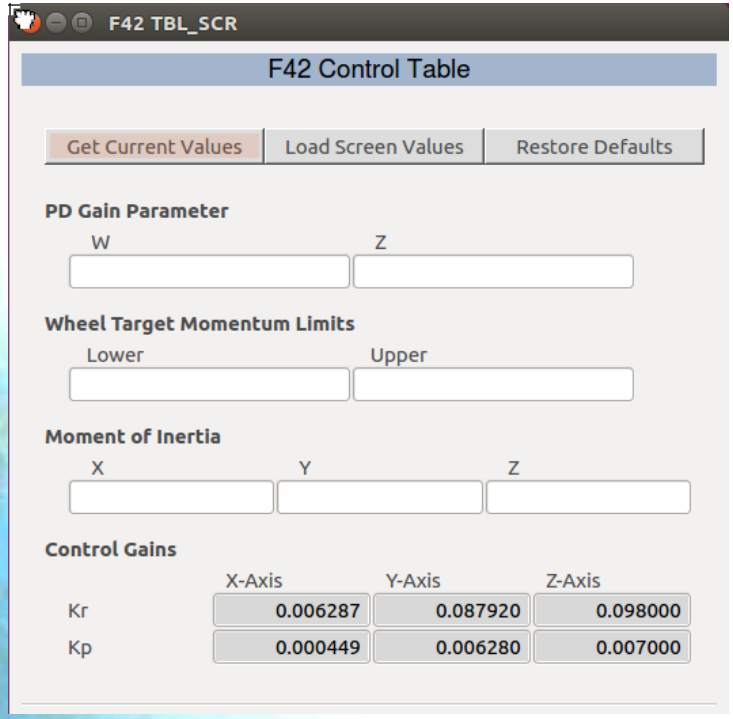
Coming Soon…

Tools: Preparing 42 Simulation

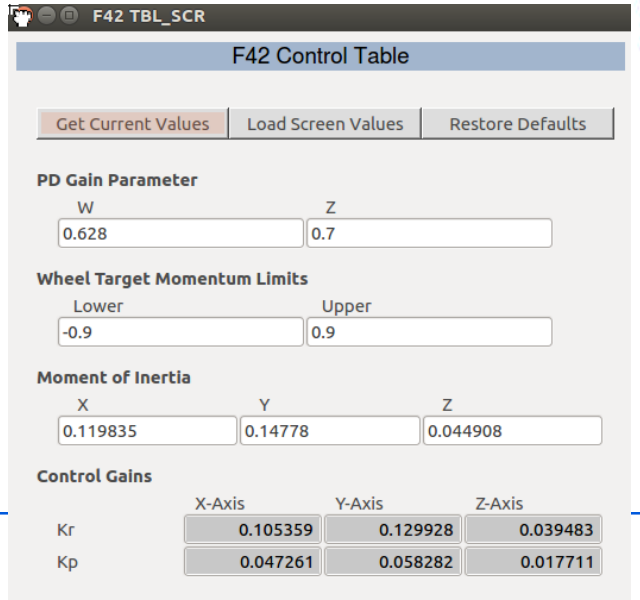


- From the kit main page on the previous slide select <42 Simulator> and the screen to the left will appear.
- The 2nd row of buttons allow you to change the behavior of the control algorithms running in the FSW and are described on the next slides
- Before running the sim you will open some additional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

42 Sim: Manage Control Table

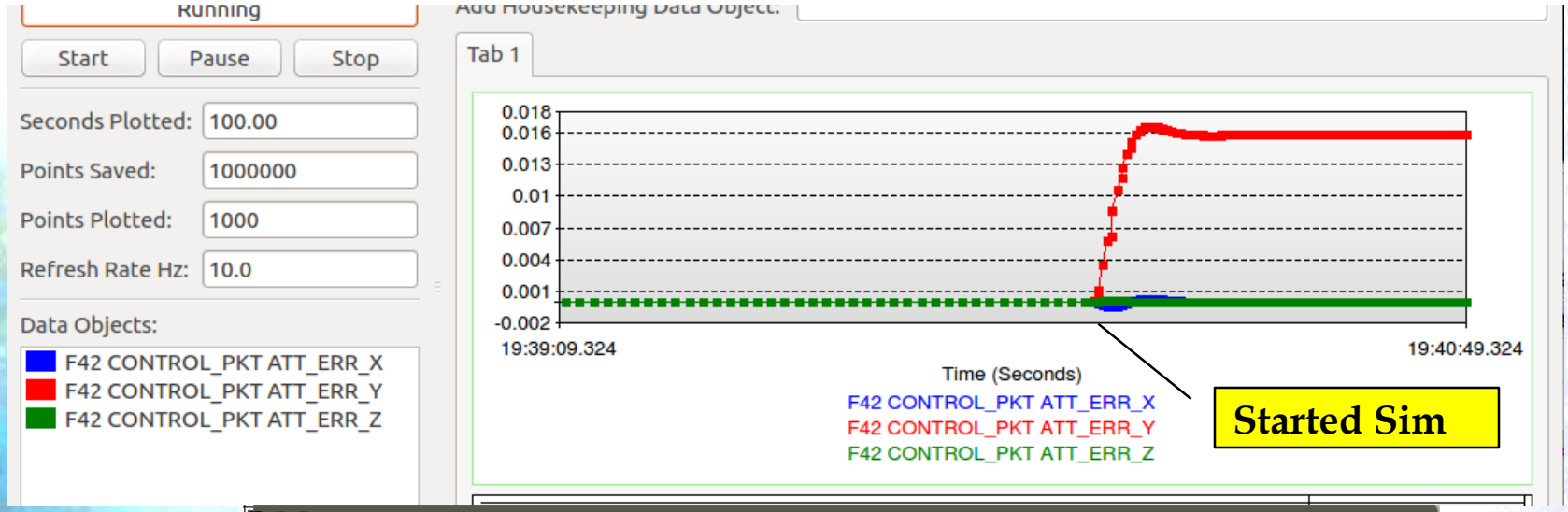


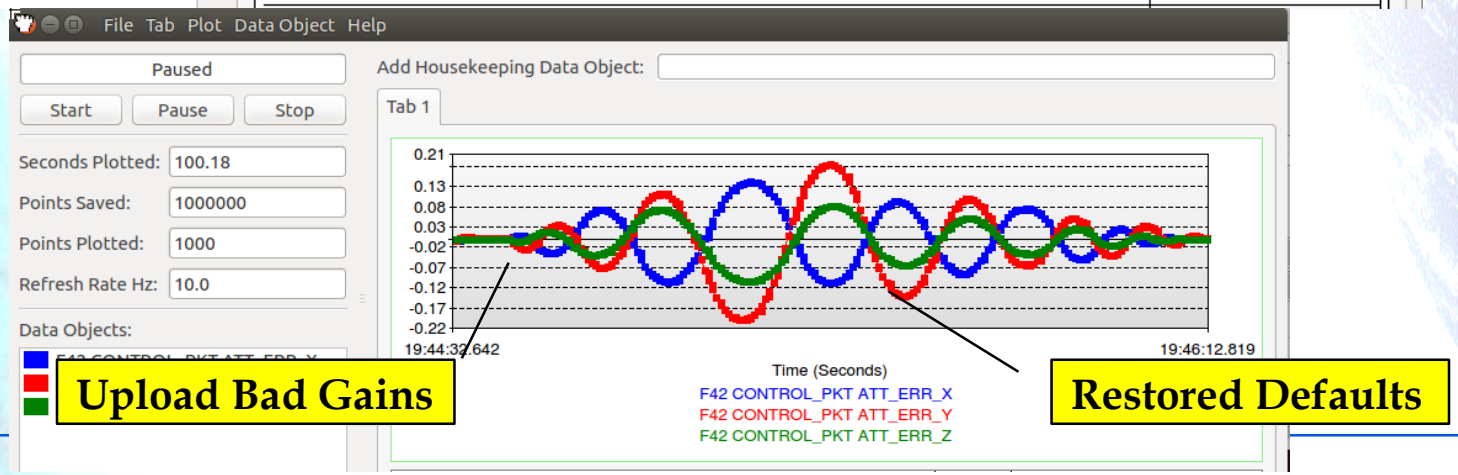
- Selecting *<Manage Control Table>* on the 42 Sim screen produces the screen to the left.
- Select *<Get Current Values>* and it will populate the screen with the current control table values. This takes a little time because it is transferring a file from flight to ground
- Edit the screen as desired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



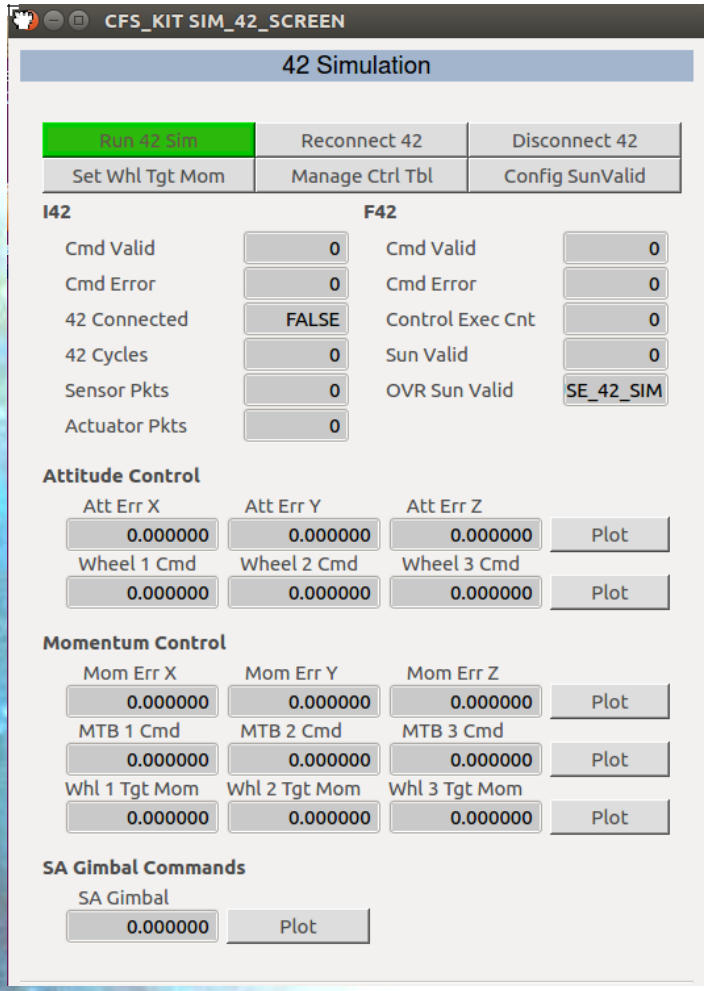
42 Sim: Plot Attitude Errors

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





42 Sim: Starting the Simulation

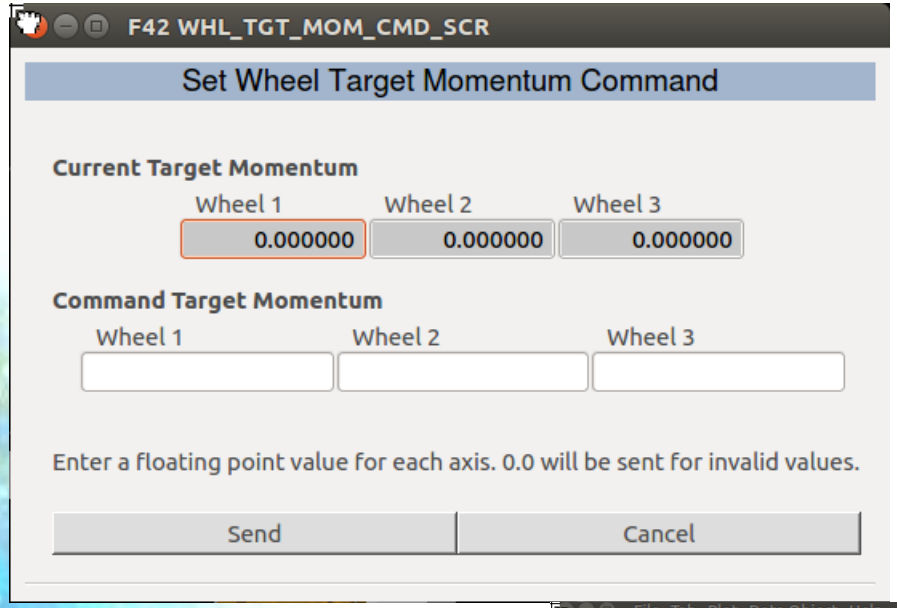


- Select <***Run 42 Sim***> which will start the 42 simulator in a new terminal window.
- The 42 configuration files used in the simulation are located in directory ***OpenSatKit/42/OSK***
- The simulation takes a while to initialize

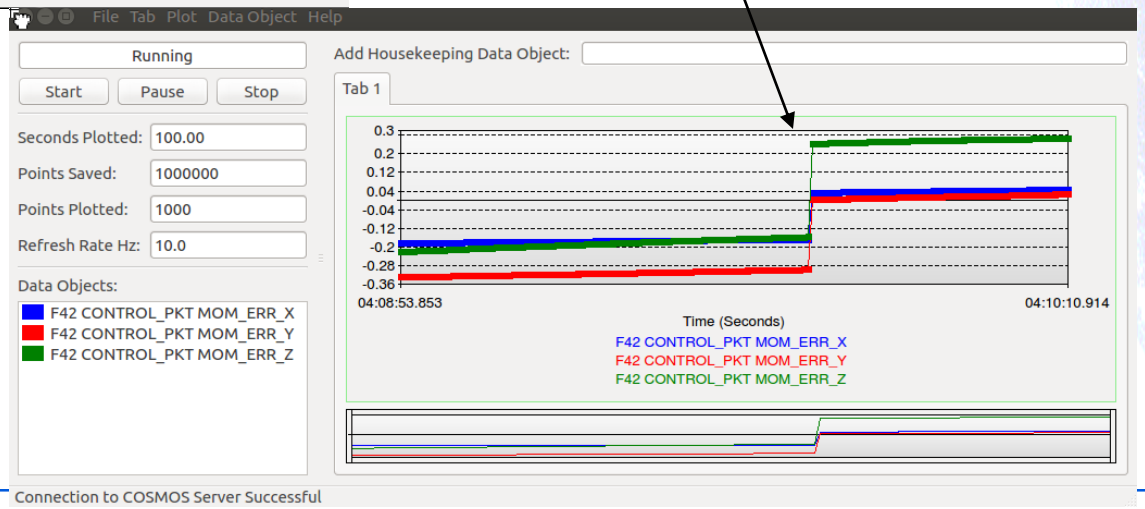
42 Sim: Additional Configuration Options

- 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

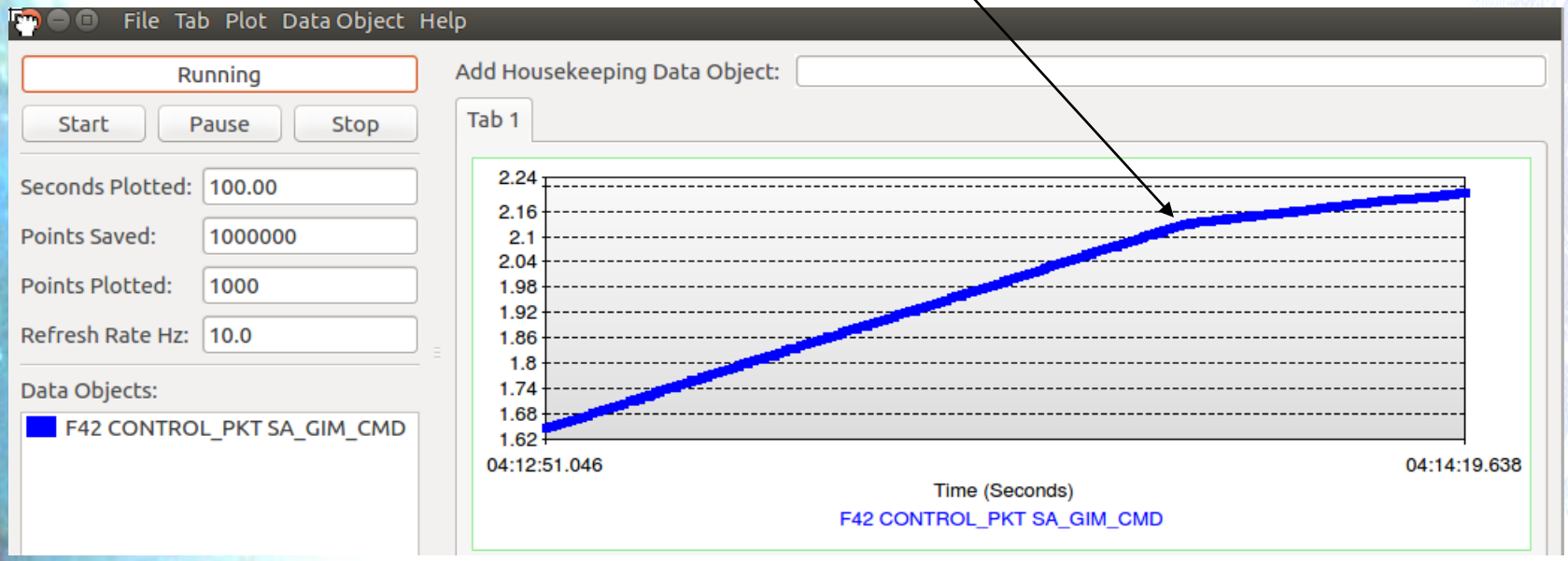


- The controller allows a non-zero (default) momentum to be stored in the wheels
- Enter new values and click <*Send*> to change the values
- The plot below shows a jump in momentum errors when new targets were selected



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

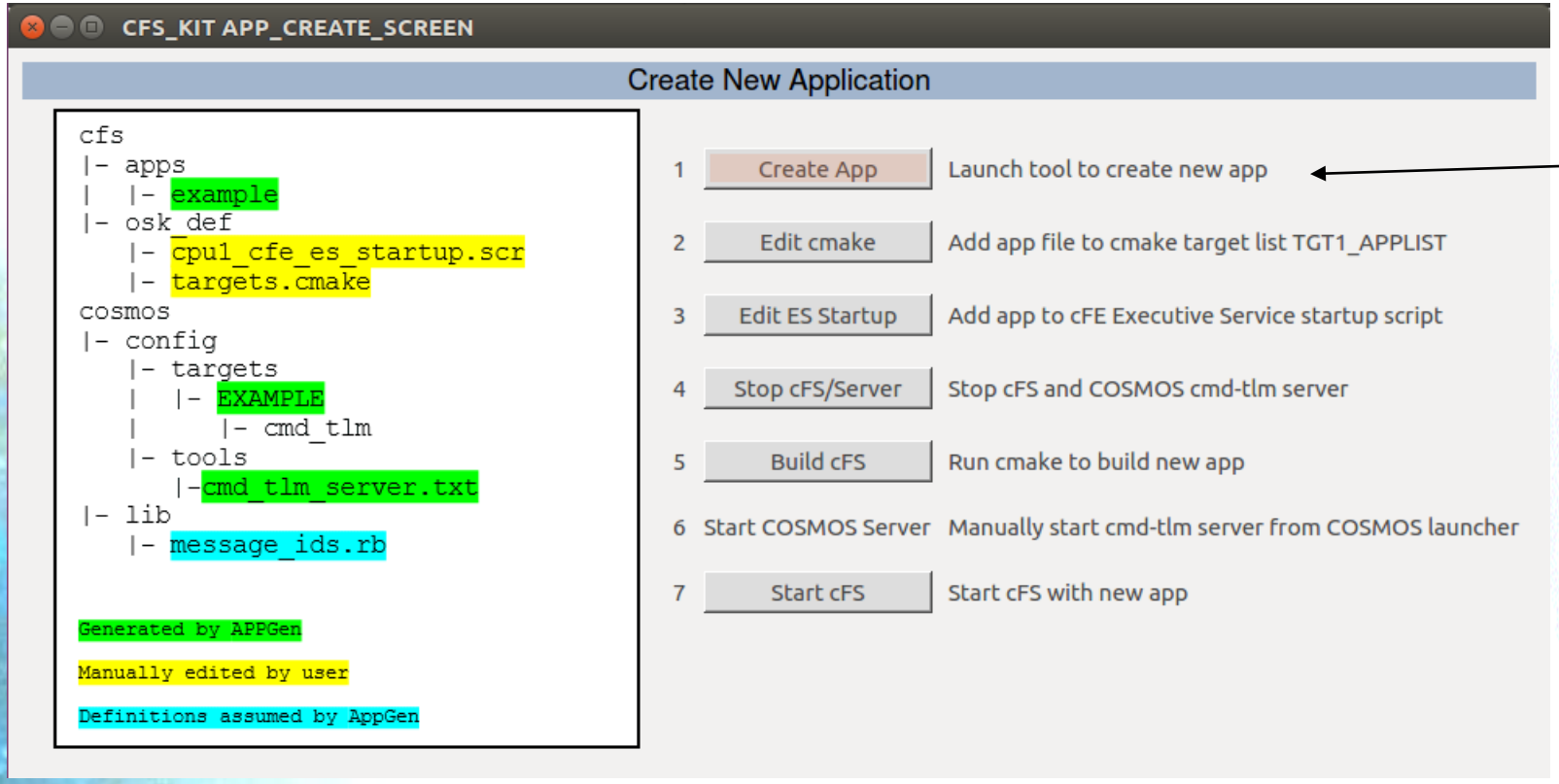
2. To terminate the flight software click on the terminal window with the FSW messages and then enter ctrl-c

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.

Tools: PiSat Control

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

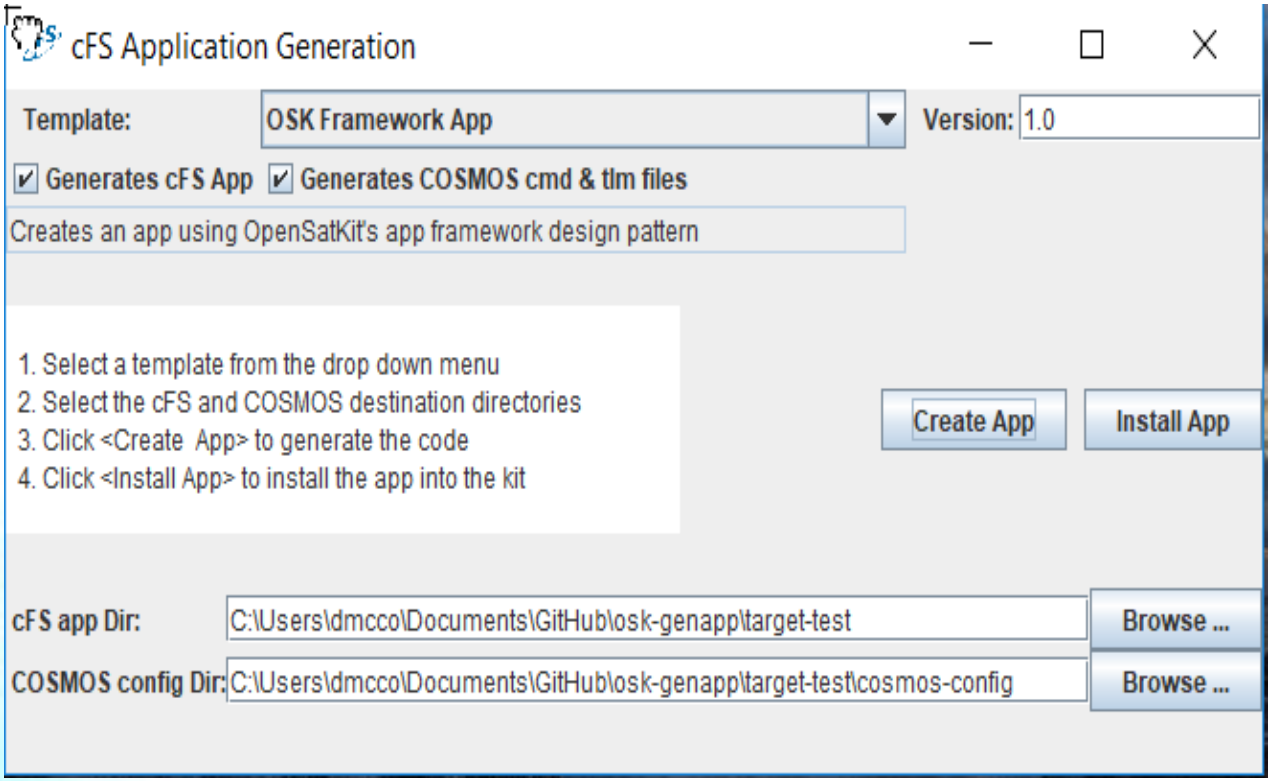
Tools: Create Application



See next slide

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

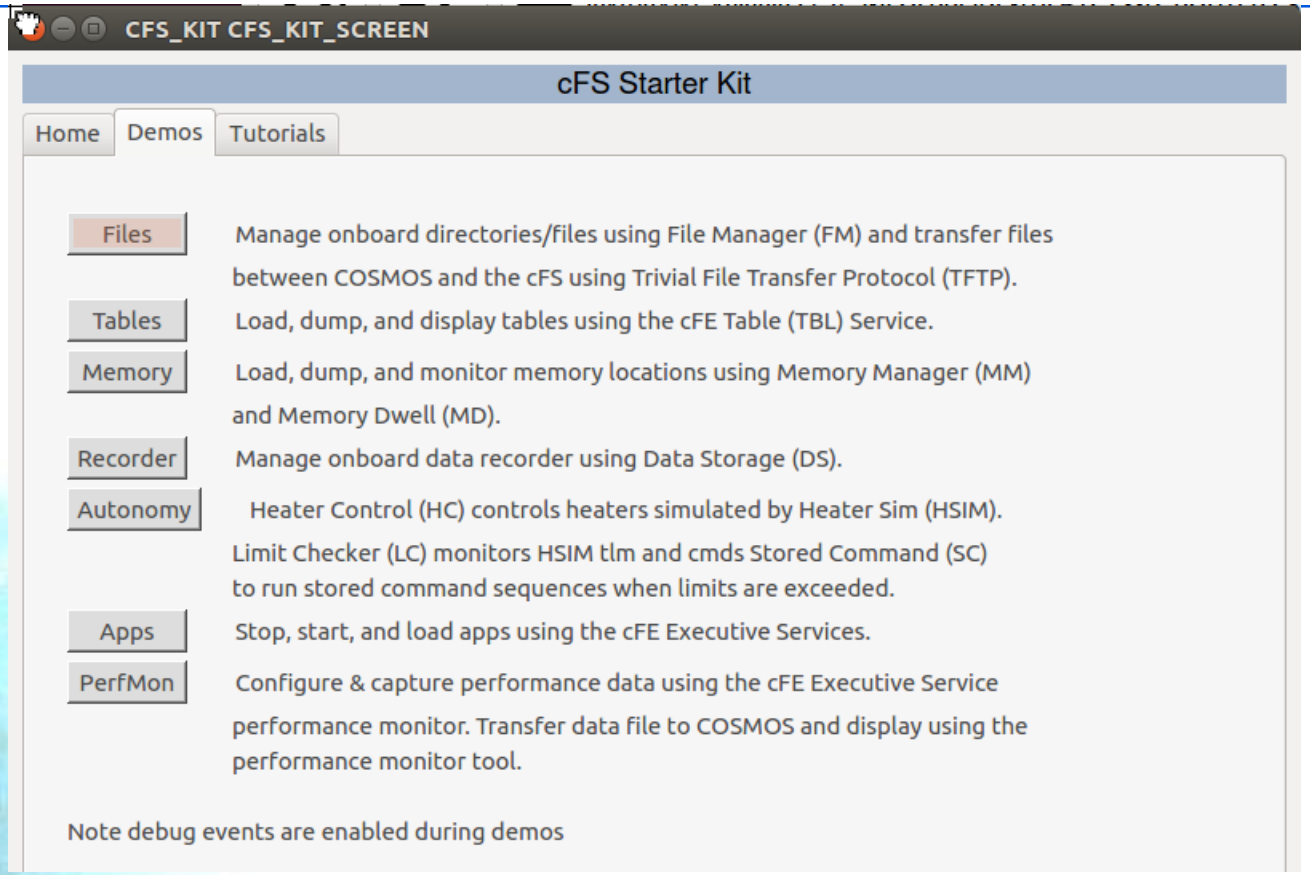
Tools: Create Application



- 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

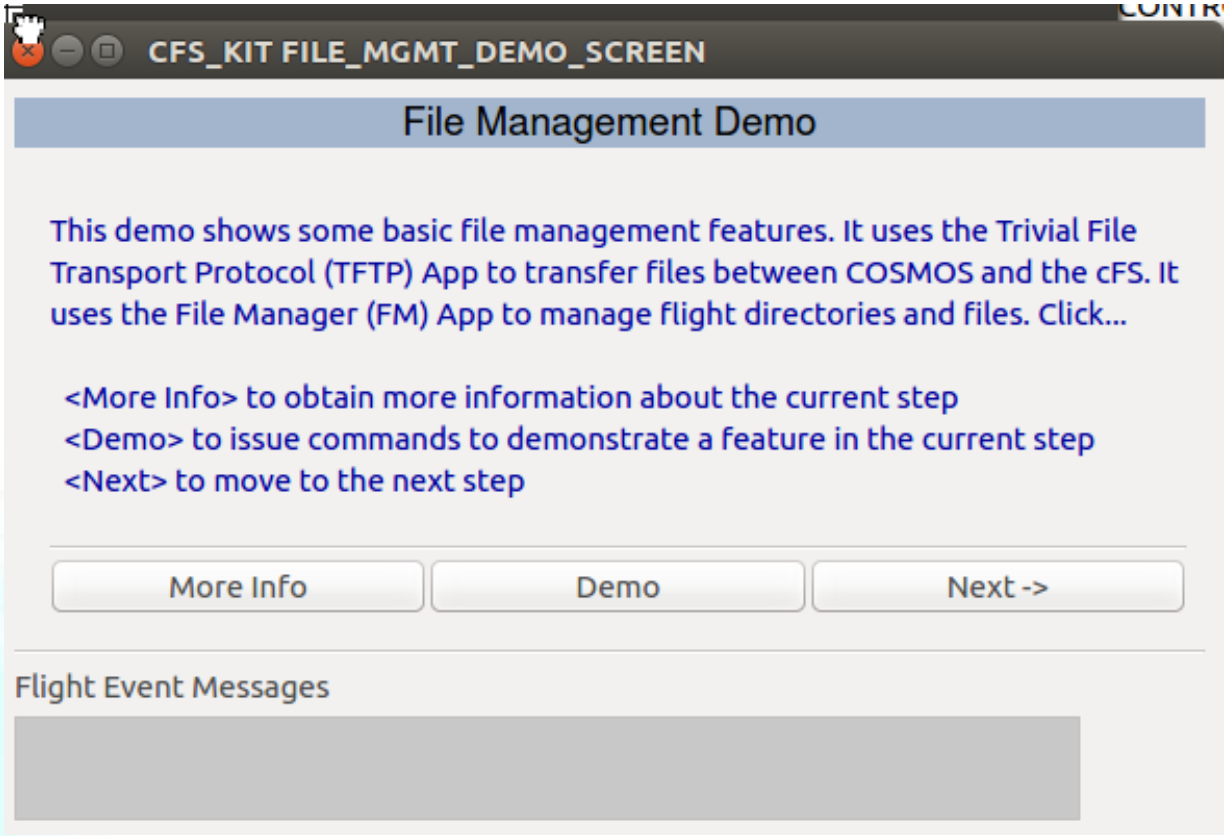
Demos

Demo Tab

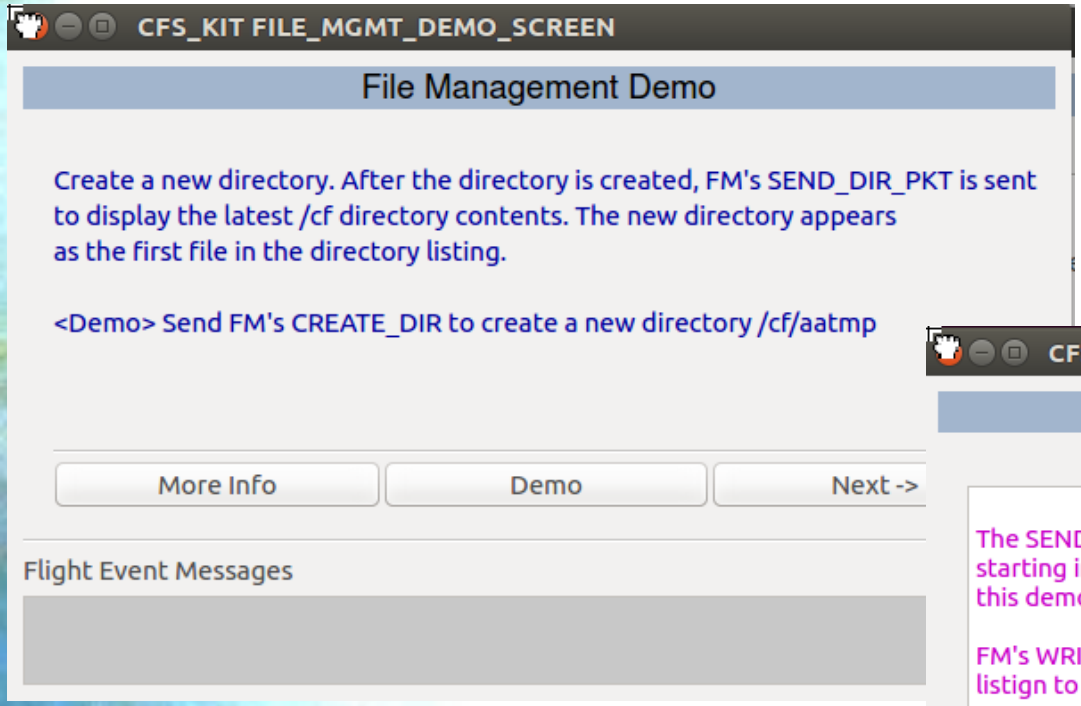


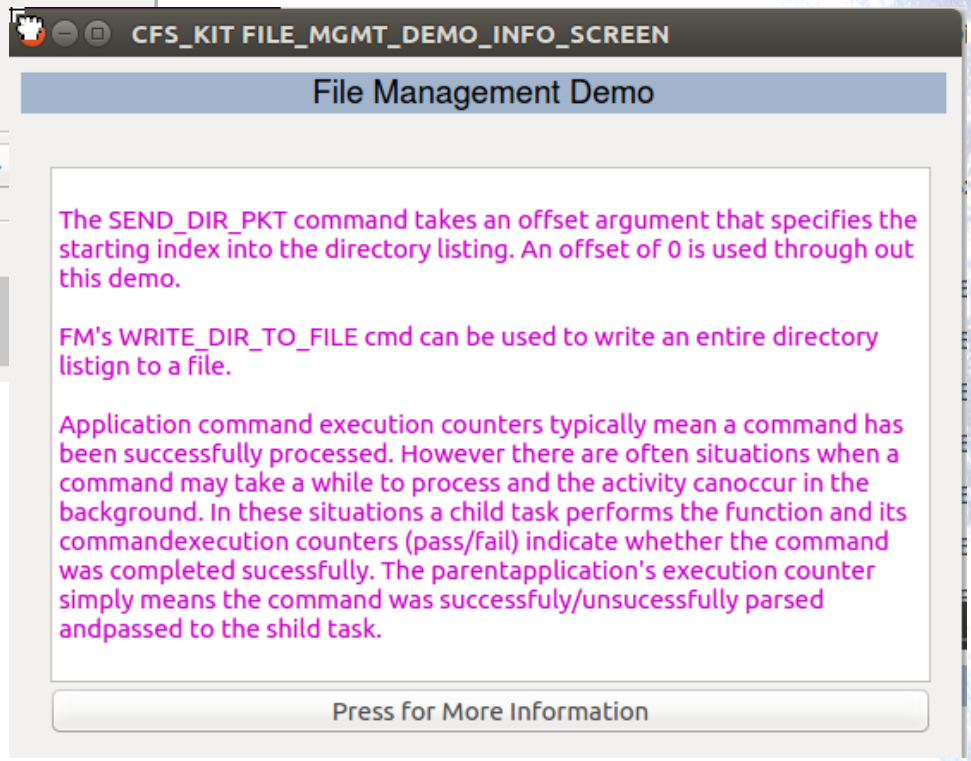
- **The first six demos correspond to the six functional buttons on the Home tab**
- **Each demo follows a common user screen configuration that is described on the following slides**

Demo Structure – FM Example (1 of 2)



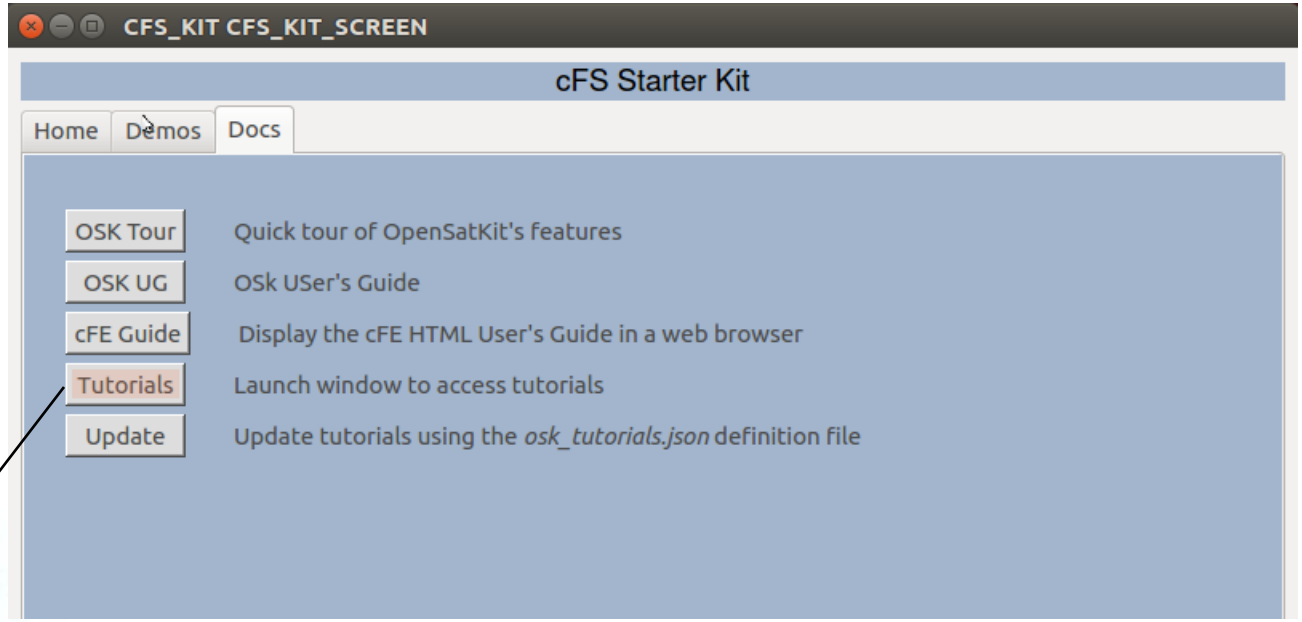
Demo Structure – FM Example (2 of 2)

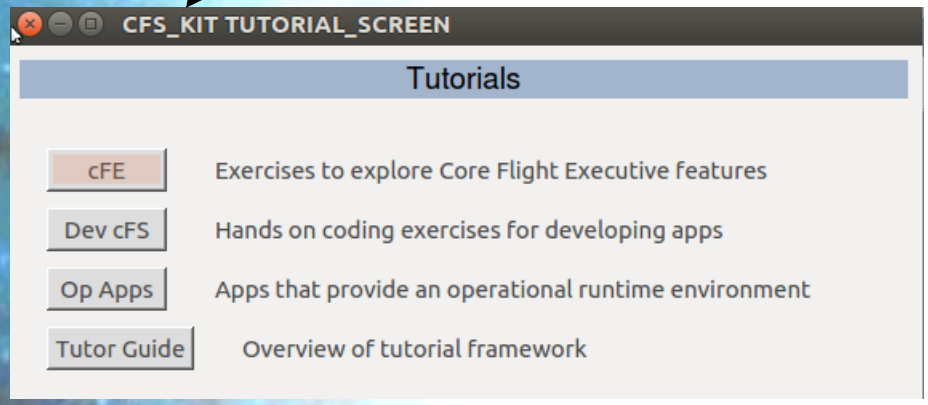




Docs

Docs Tab





- **Add your own tutorials**
- **Instructions in <Tutor Guide>**