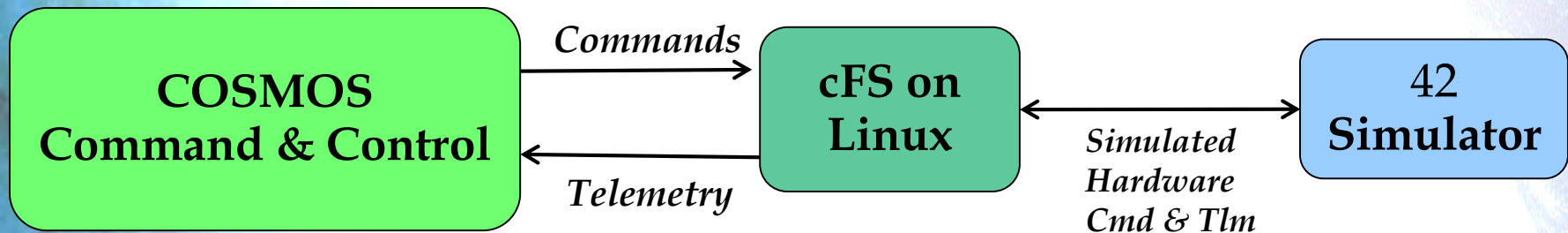


# OpenSatKit

## Tour

# Introduction

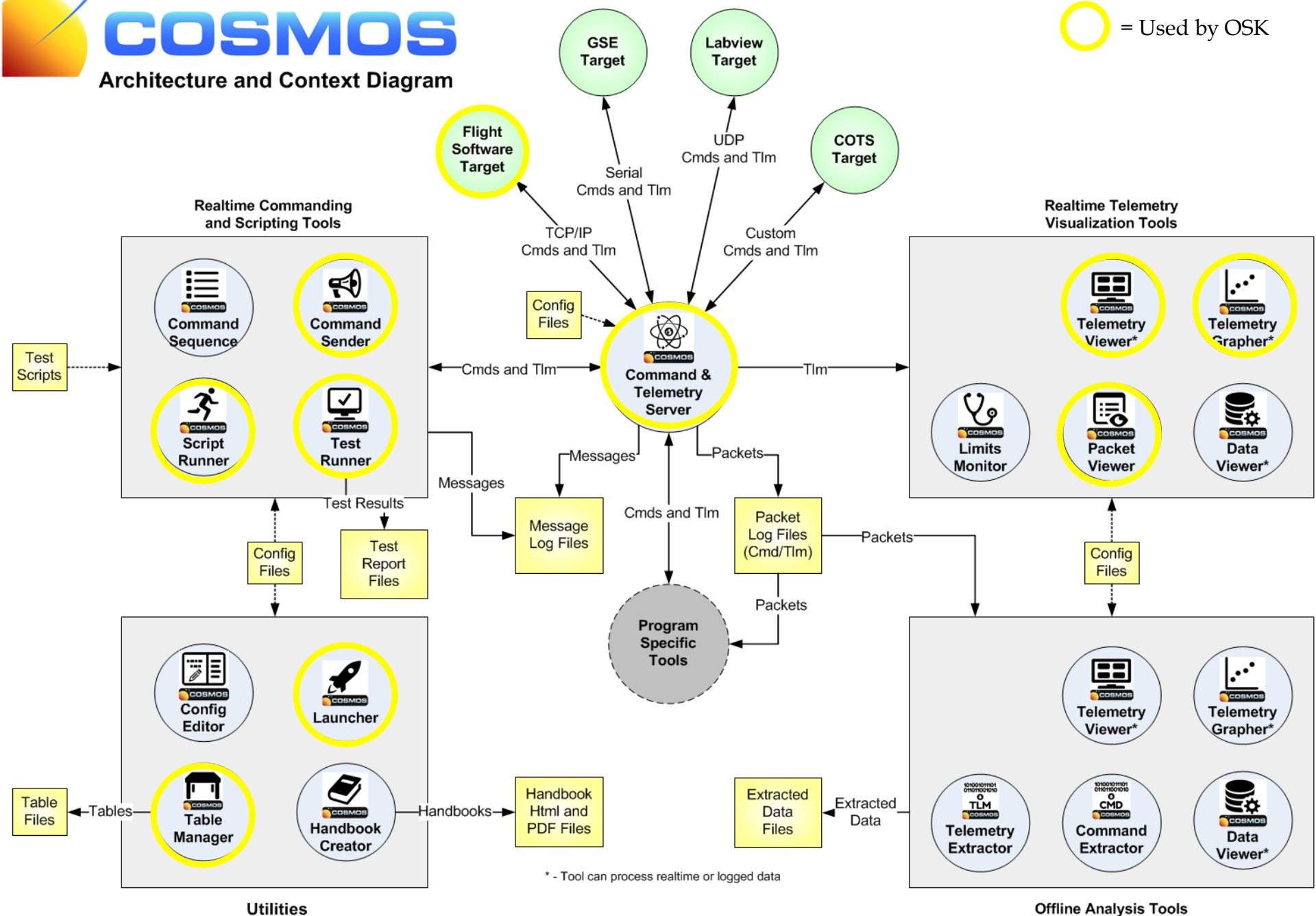
- The primary goal of OpenSatKit (OSK) is to provide a core Flight System (cFS) development and run time environment that can be used to learn about the cFS and to serve as a starting point for a new project
- In addition to the cFS itself, OSK uses two additional open source applications
  - Ball Aerospace's 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 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



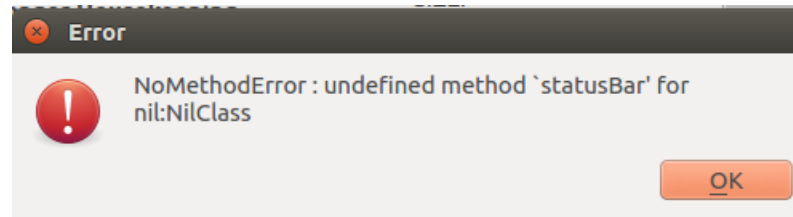
# COSMOS Configuration (2 of 2)

---

- **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.

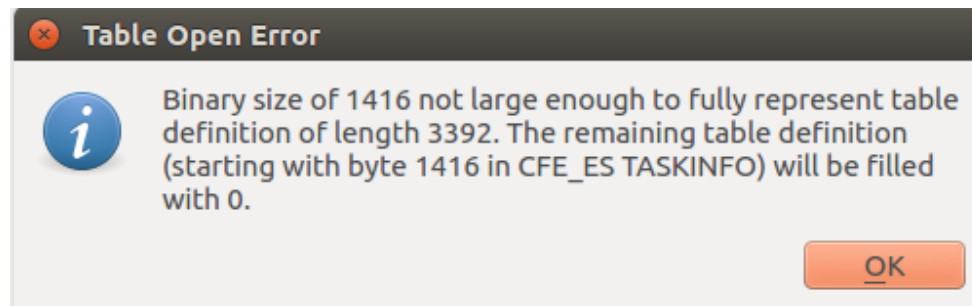


- The FSW terminal window may display start and stop “FlyWheel” messages
  - OSK is a non-realtime environment so the cFE time service is warning that’s it’s not operating within its real-time precision limits relative to a 1Hz timer
  - OSK is designed to help users learn functional features and only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

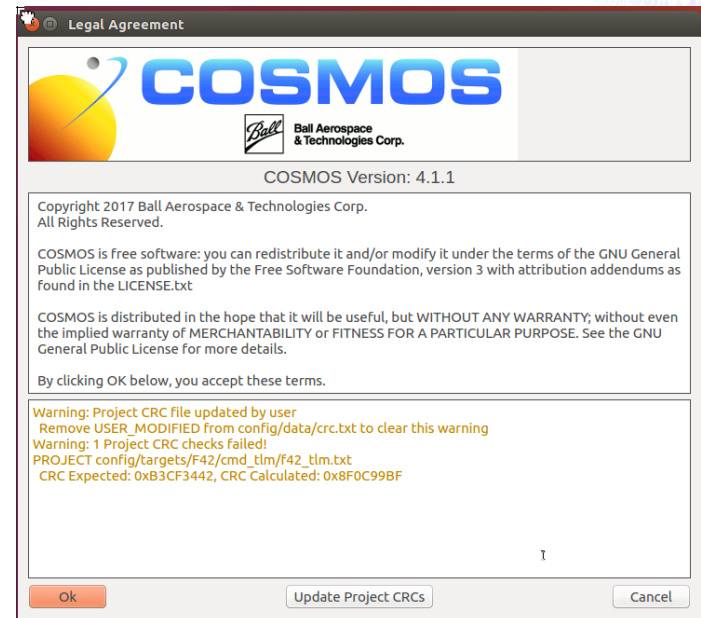


# Getting Started



# 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>



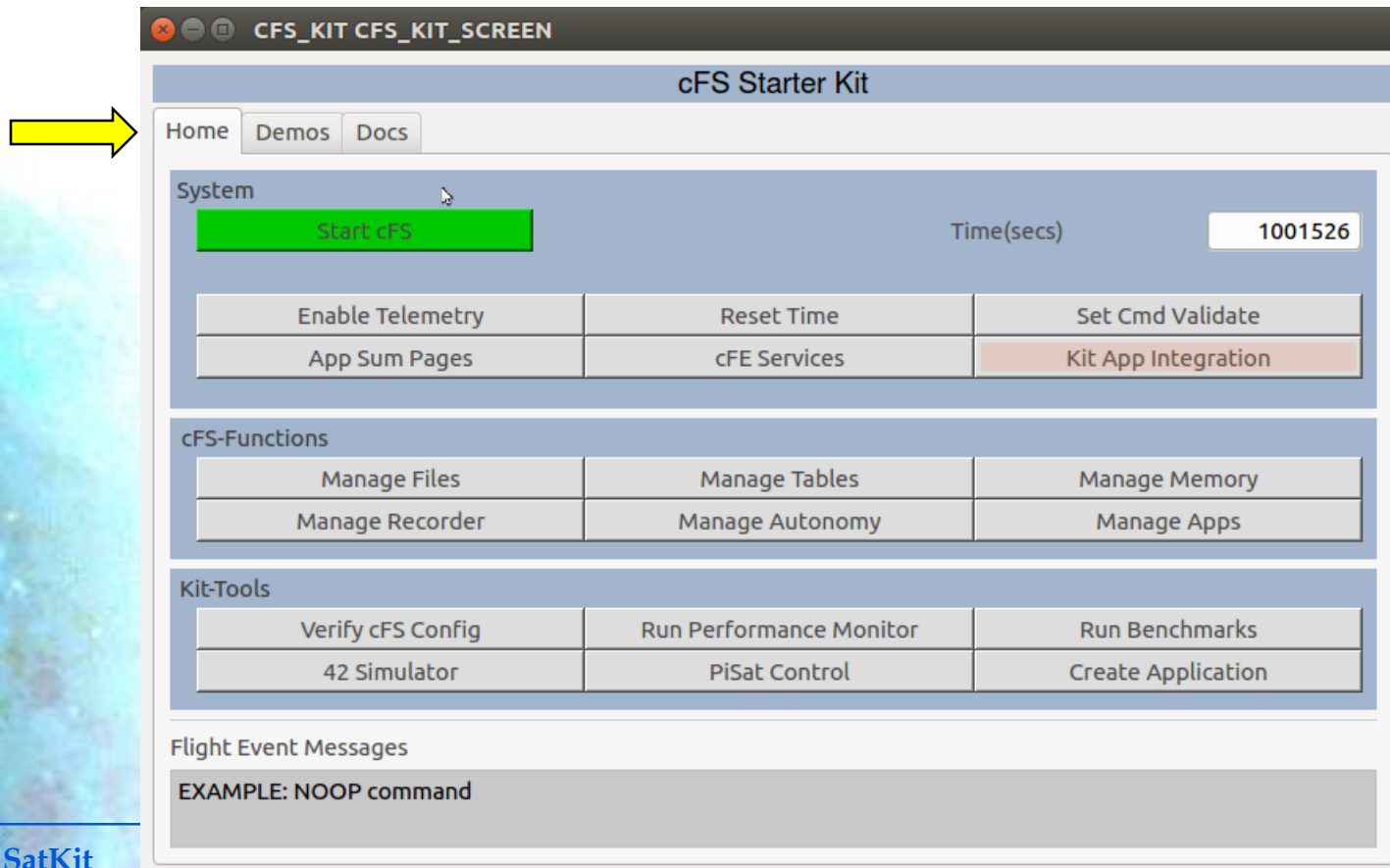
# Running OpenSatKit (2 of 2)



- 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.



The screenshot shows the 'cFS Starter Kit' window. A yellow arrow points to the 'Home' tab. The window contains the following sections:

- System**
  - A green 'Start cFS' button.
  - A 'Time(secs)' display showing '1001526'.
  - A table of system functions:
 

Enable Telemetry	Reset Time	Set Cmd Validate
App Sum Pages	cFE Services	Kit App Integration
- cFS-Functions**
  - A table of cFS functions:
 

Manage Files	Manage Tables	Manage Memory
Manage Recorder	Manage Autonomy	Manage Apps
- Kit-Tools**
  - A table of kit tools:
 

Verify cFS Config	Run Performance Monitor	Run Benchmarks
42 Simulator	PiSat Control	Create Application
- Flight Event Messages**
  - A text area showing 'EXAMPLE: NOOP command'.

# 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

Prototype Benchmark App

Create "Hello World" App

Screen to manage "PiSat"

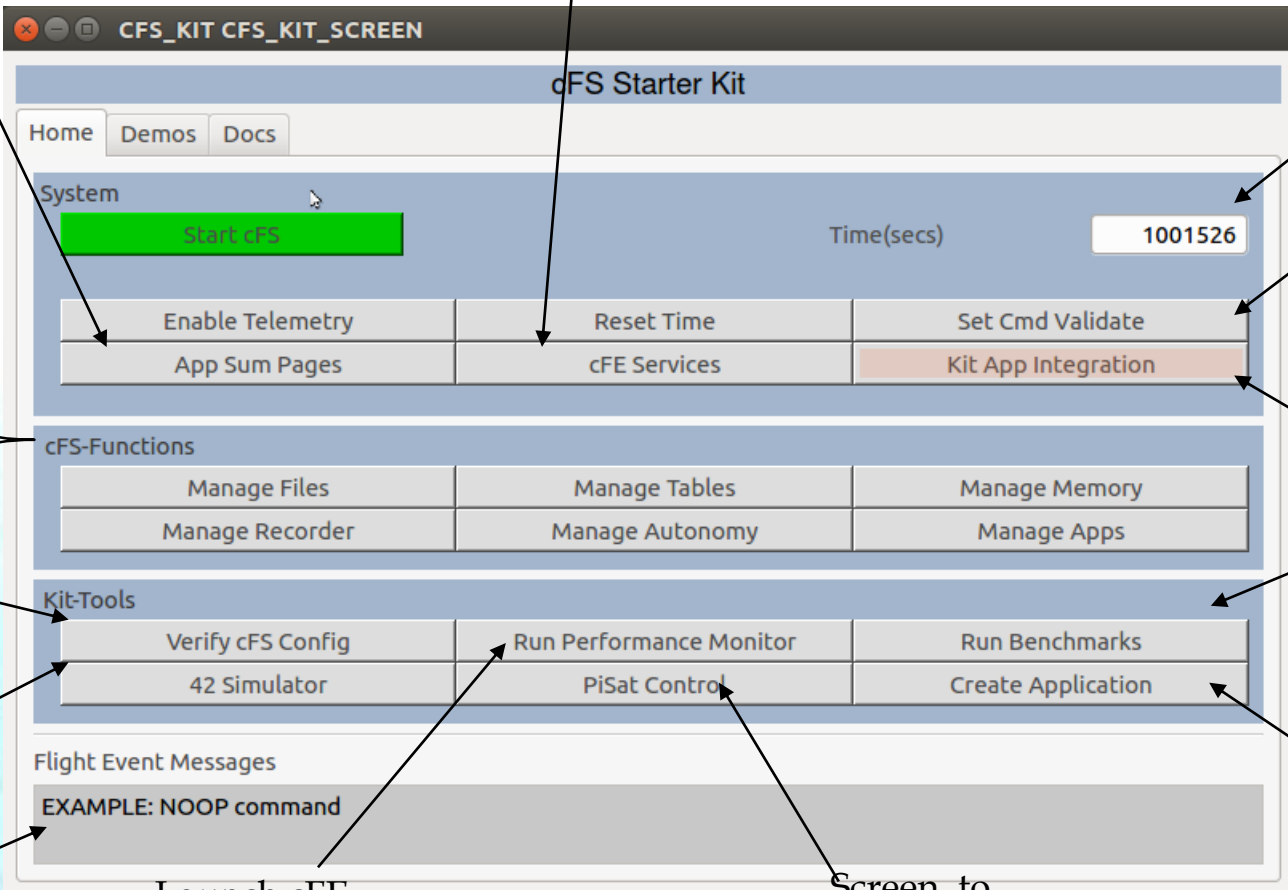
Launch cFE Perf Mon Tool

Display last FSW event message

Closed-loop control sim

Run test script

Launch screens to manage a functional goal



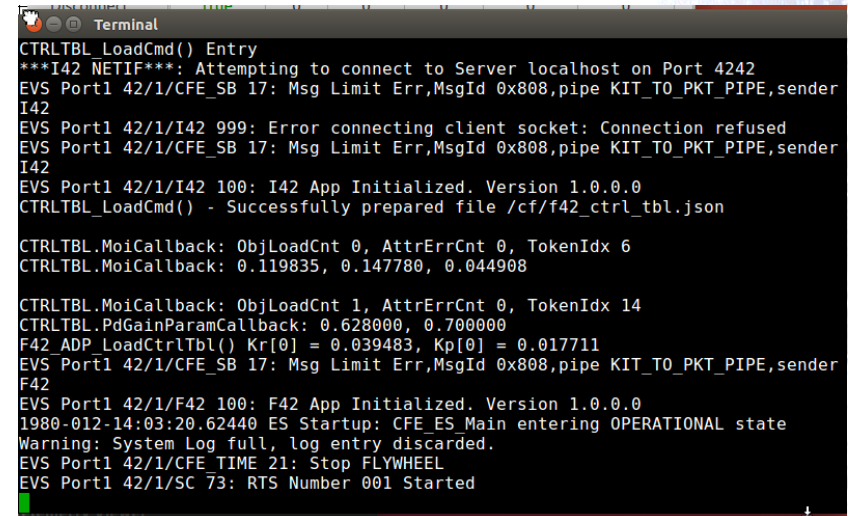
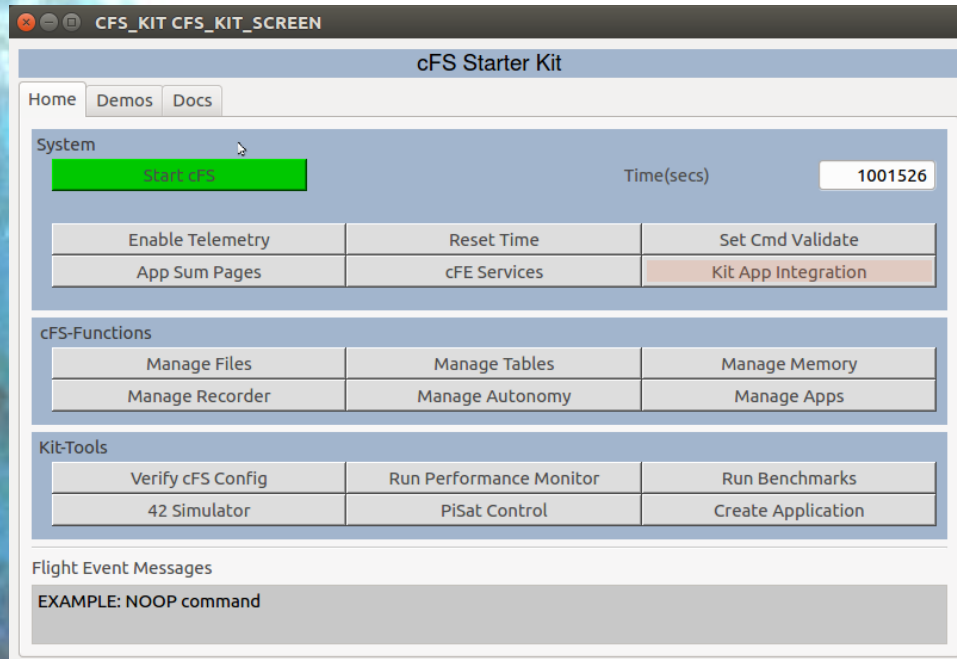
The screenshot shows the CFS Starter Kit interface with the following components and annotations:

- Header:** CFS\_KIT CFS\_KIT\_SCREEN
- Navigation:** Home, Demos, Docs
- System Section:**
  - Start cFS (Green button)
  - Time(secs): 1001526
  - Buttons: Enable Telemetry, Reset Time, Set Cmd Validate, App Sum Pages, cFE Services, Kit App Integration
- cFS-Functions Section:**
  - Buttons: Manage Files, Manage Tables, Manage Memory, Manage Recorder, Manage Autonomy, Manage Apps
- Kit-Tools Section:**
  - Buttons: Verify cFS Config, Run Performance Monitor, Run Benchmarks, 42 Simulator, PiSat Control, Create Application
- Flight Event Messages Section:**
  - EXAMPLE: NOOP command



# 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

Each app sends a housekeeping packet containing

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

CFS\_KIT APP\_CFS\_SUMMARY\_SCREEN

cFS Applications

App Name	Seq Cnt	Cmd Valid Cnt	Cmd Error Cnt
CFE_ES - cFE Executive 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

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

CFS\_KIT CFE\_SCREEN

Core Flight Executive

Executive Service

NOOP

Send Command

Perf Mon

Rst Type

0

Rst Subtype

0

# Proc Rsts

0

ErLog Entries

0

Heap Free

0

Heap Blk Free

0

Heap Max Blk

0

ErLog Index

0

SysLog Used

0

SysLog Size

0

SysLog Entries

0

SysLog Mode

0

Event Service

NOOP

Send Command

Send Count

0

Msg Format

SHORT

Msg Trunc

0

Port Mask

0

Log Ena

0

Log Mode

OVERWRITE

Log Full

FALSE

Log OvFl

0

Software Bus

NOOP

Send Command

No Sub

0

Send Err

0

Recv Err

0

Pipe OvrFlw

0

Msg Lim Err

0

Que Err

0

Mem In Use

0

UnUsed Mem

0

Create Pipe Err

0

Subscribe Err

0

Dup Subscribe

0

Table Service

NOOP

Send Command

Tbl Mgmt

# Tbl Reg

0

Load Pend

0

Free Buf

0

Last Update Sec

0

Valid Cnt

0

Valid Status

0

Valid Buf

INACTIVE

Valid Tbl

Valid Pass

0

Valid Fail

0

Valid Req

0

Valid CRC

0

Time Service

NOOP

Send Command

Clk State

0x0000

API State

VALID

Leap Secs

0

1Hz Adj Secs

0

Pkt Time

0

MET Secs

0

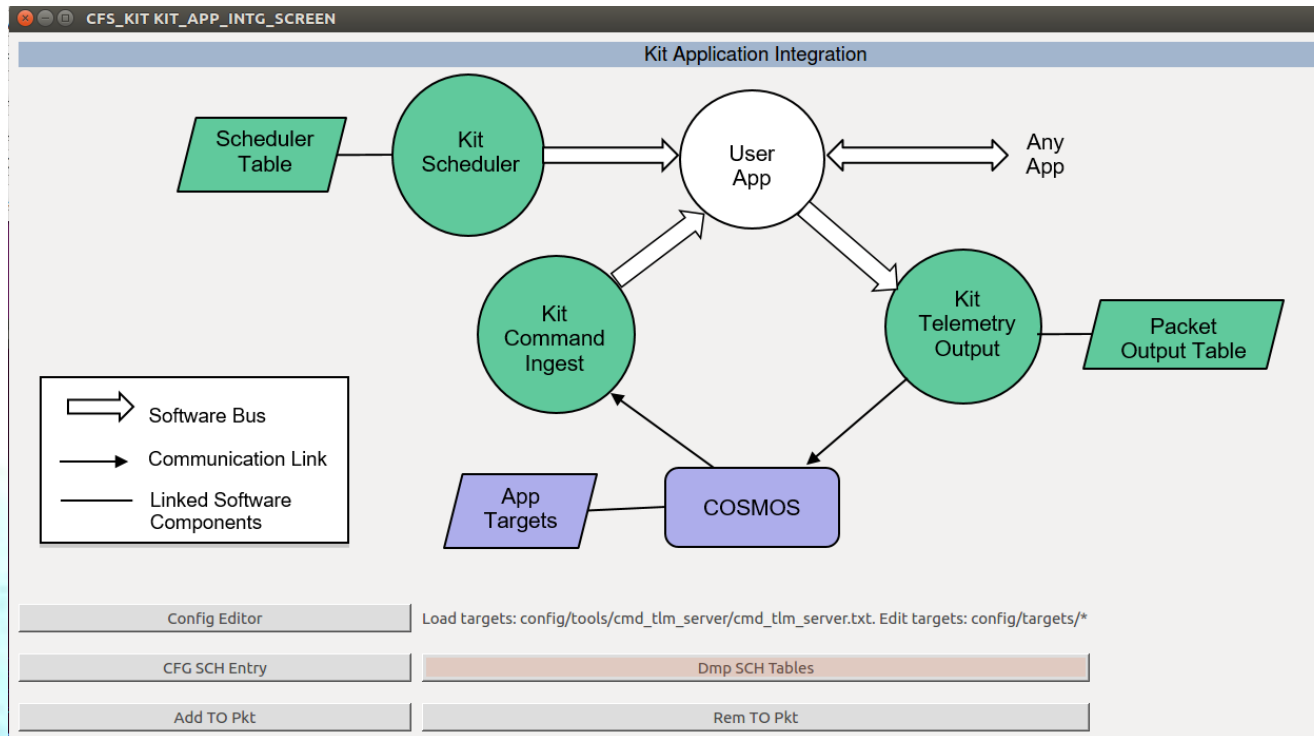
STCF Secs

0

Flight Event Messages

- 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 file location
  - Tables 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

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:

PACKETFILES:

FIRSTFILE:

FILE01\_NAME:

FILE02\_NAME:

FILE03\_NAME:

FILE04\_NAME:

FILE05\_NAME:

FILE06\_NAME:

FILE07\_NAME:

FILE08\_NAME:

FILE10\_NAME:

FILE11\_NAME:

FILE12\_NAME:

**File Transfer**

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

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

Ground Working Directory

Flight Working Directory

**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

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

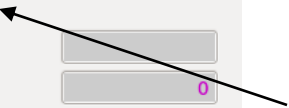
LAST\_FILE\_LOADED:

OWNER\_APP\_NAME:

File Transfer

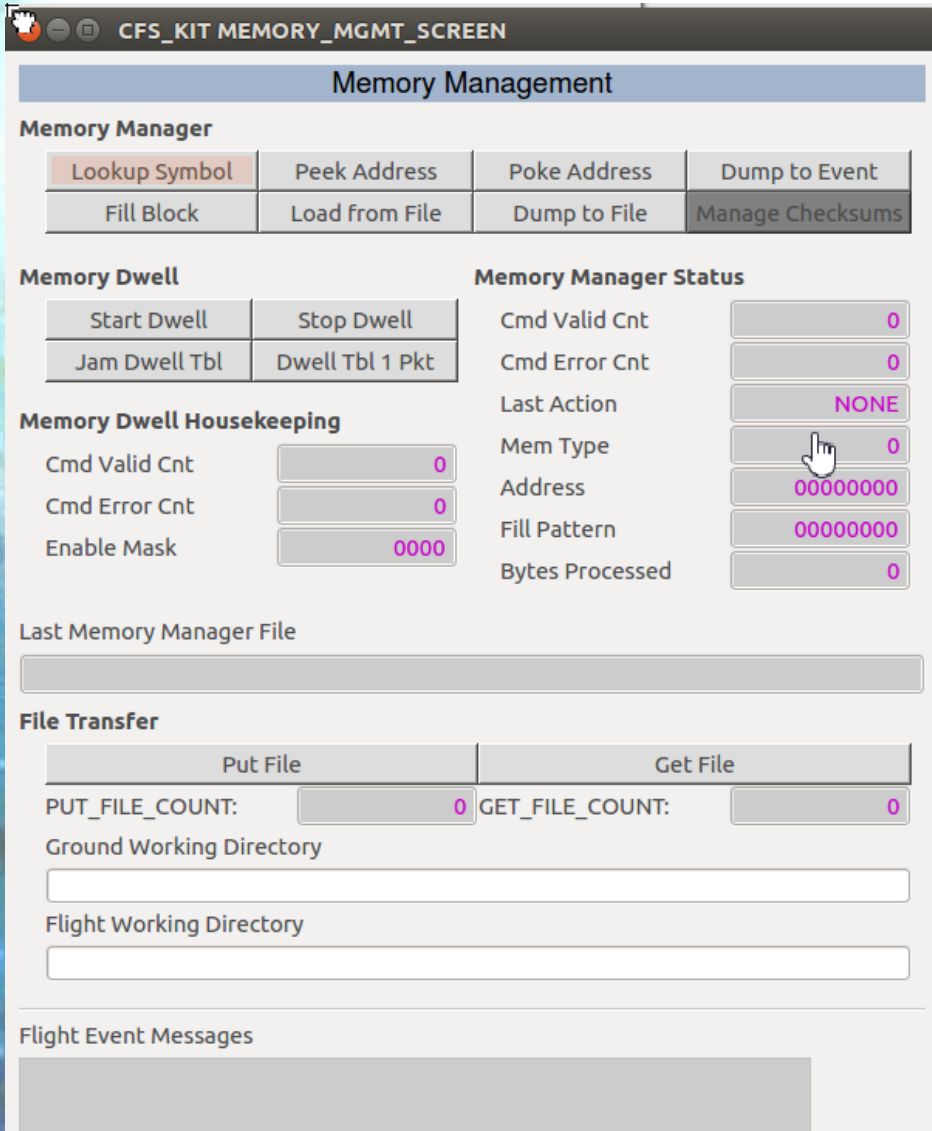
Put File	Get File
PUT_FILE_COUNT: <input type="text"/> 0	GET_FILE_COUNT: <input type="text"/> 0
Ground Working Directory	
<input type="text"/>	
Flight Working Directory	
<input type="text"/>	

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

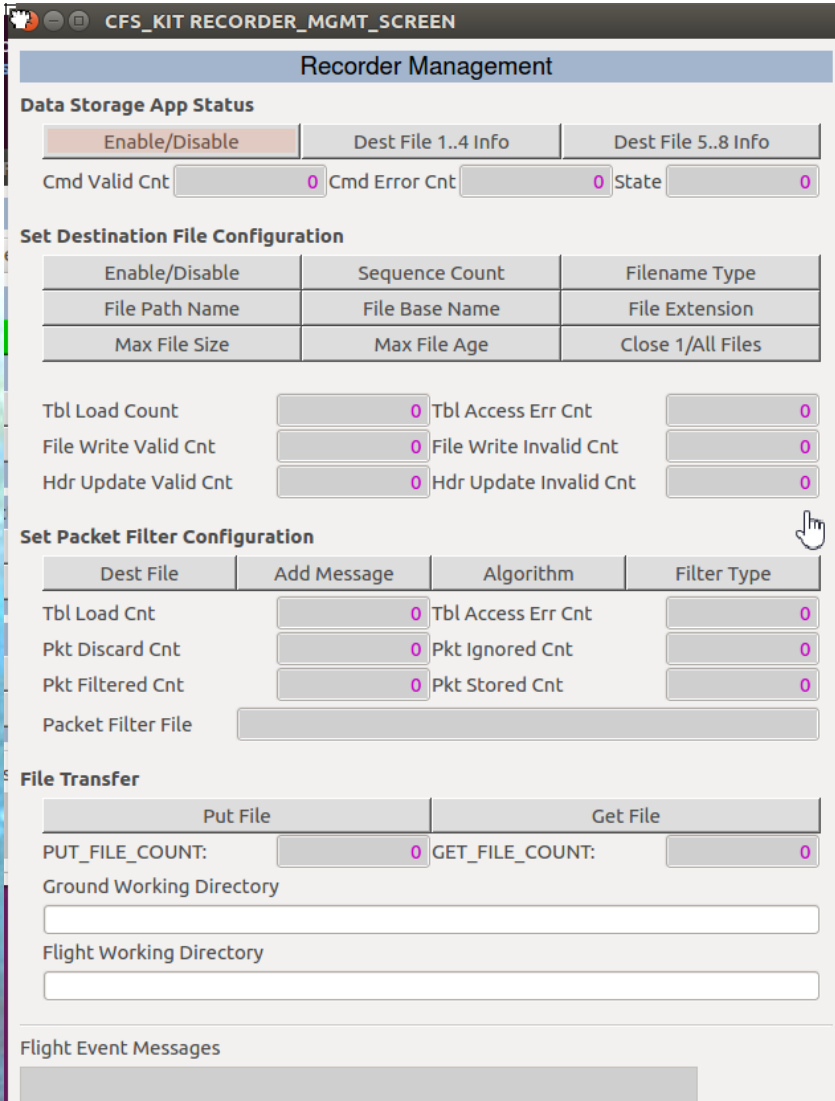
File Transfer

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



**CFS\_KIT RECORDER\_MGMT\_SCREEN**

## Recorder Management

### Data Storage App Status

Enable/Disable

Dest File 1..4 Info

Dest File 5..8 Info

Cmd Valid Cnt 
Cmd Error Cnt 
State

### 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 
Tbl Access Err Cnt 
File Write Valid Cnt 
File Write Invalid Cnt 
Hdr Update Valid Cnt 
Hdr Update Invalid Cnt

### Set Packet Filter Configuration

Dest File	Add Message	Algorithm	Filter Type
Tbl Load Cnt	<input type="text" value="0"/>	Tbl Access Err Cnt	<input type="text" value="0"/>
Pkt Discard Cnt	<input type="text" value="0"/>	Pkt Ignored Cnt	<input type="text" value="0"/>
Pkt Filtered Cnt	<input type="text" value="0"/>	Pkt Stored Cnt	<input type="text" value="0"/>

Packet Filter File


### File Transfer

Put File	Get File
PUT_FILE_COUNT: <input type="text" value="0"/>	GET_FILE_COUNT: <input type="text" value="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	000000
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


CFS\_KIT APP\_MGMT\_SCREEN

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	<input type="text" value="0"/>	Cmd Err Ctr	<input type="text" value="0"/>
Registered Apps	<input type="text" value="0"/>	Registered Tasks	<input type="text" value="0"/>

App Info

Name	<input type="text"/>	Entry Point	<input type="text"/>
Main Task Name	<input type="text"/>	Main Task ID	<input type="text" value="0"/>
APP ID	<input type="text" value="0"/>	Priority	<input type="text" value="0"/>
Type	<input type="text" value="0"/>	# Child Tasks	<input type="text" value="0"/>
File Name	<input type="text"/>	Exception	<input type="text" value="0"/>
Code Size	<input type="text" value="0"/>	Data Size	<input type="text" value="0"/>
BSS Size	<input type="text" value="0"/>	Stack Size	<input type="text" value="0"/>

File Transfer

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

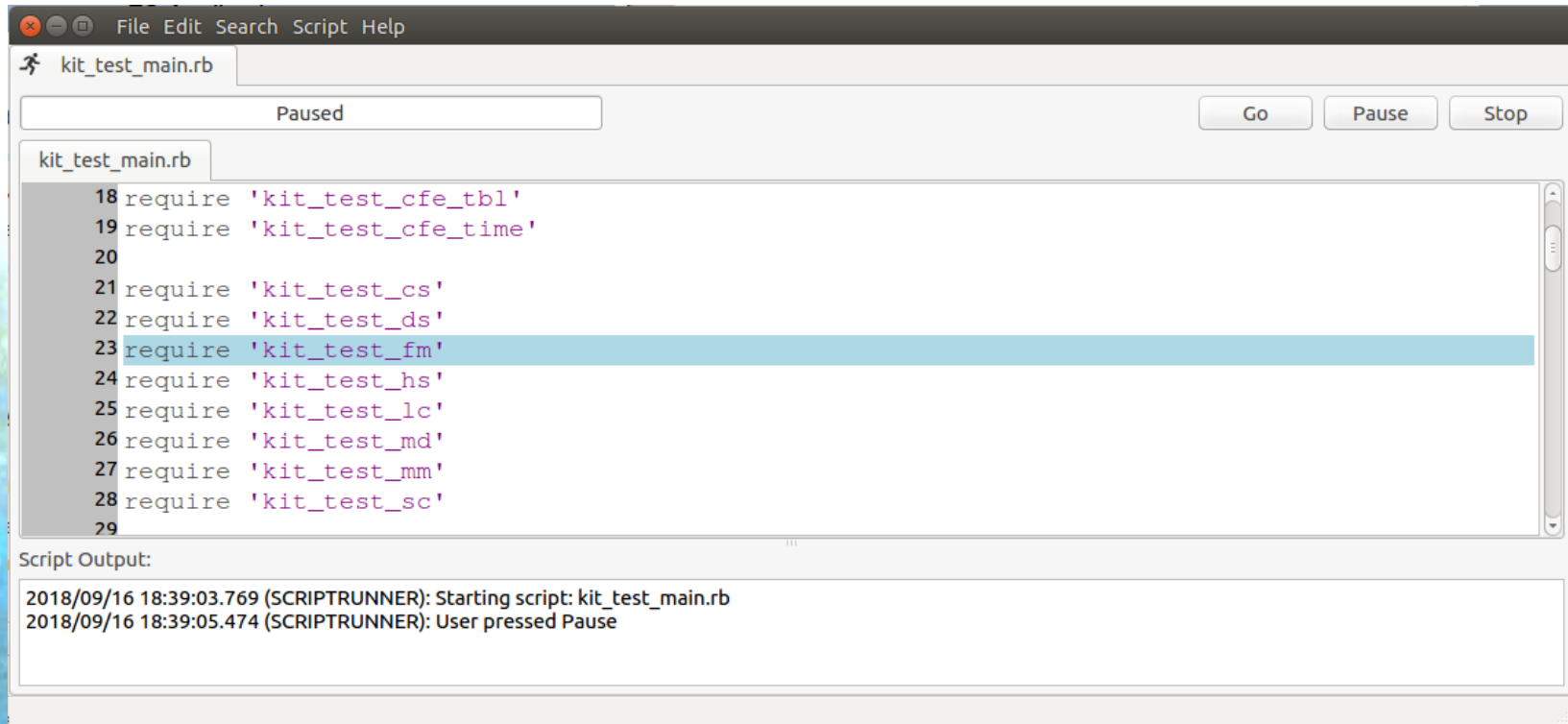
Flight Event Messages

- *<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>*



# Tools

# Tools: Verify cFS Configuration



The screenshot shows the OpenSatKit Script Runner window. The title bar includes 'File Edit Search Script Help'. The main window has a tab labeled 'kit\_test\_main.rb'. Below the tab is a 'Paused' status indicator and three buttons: 'Go', 'Pause', and 'Stop'. The script content is displayed in a text area, showing a series of 'require' statements for various test modules. Line 23, 'require 'kit\_test\_fm'', is highlighted. The 'Script Output' panel at the bottom shows the following log entries:

```
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
```

```
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
```

- 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>

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**

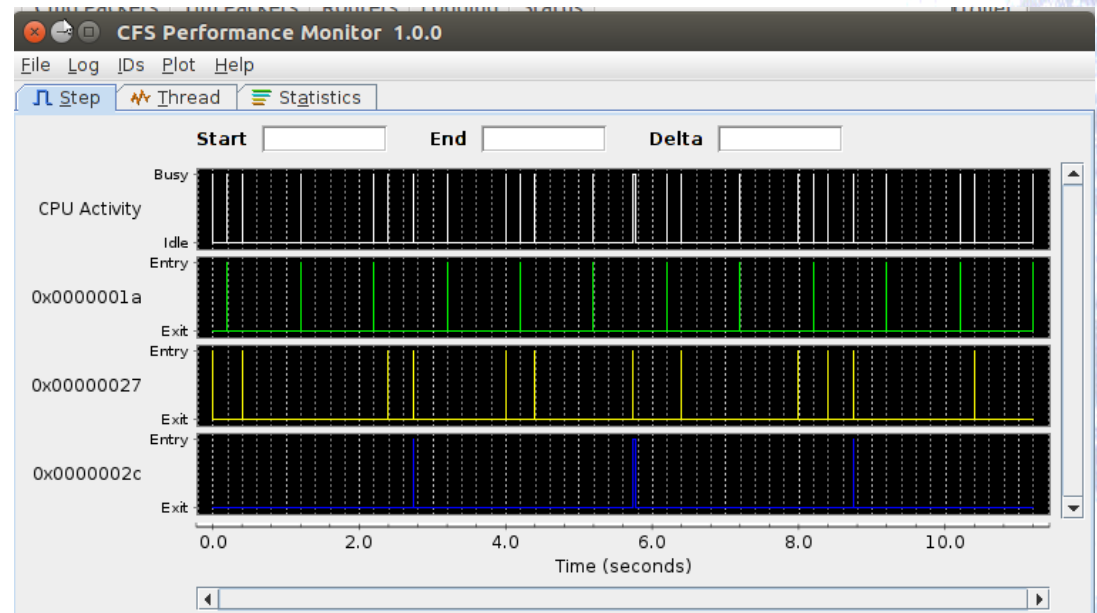
Put File	Get File
----------	----------

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

Ground Working Directory

Flight Working Directory

Flight Event Messages



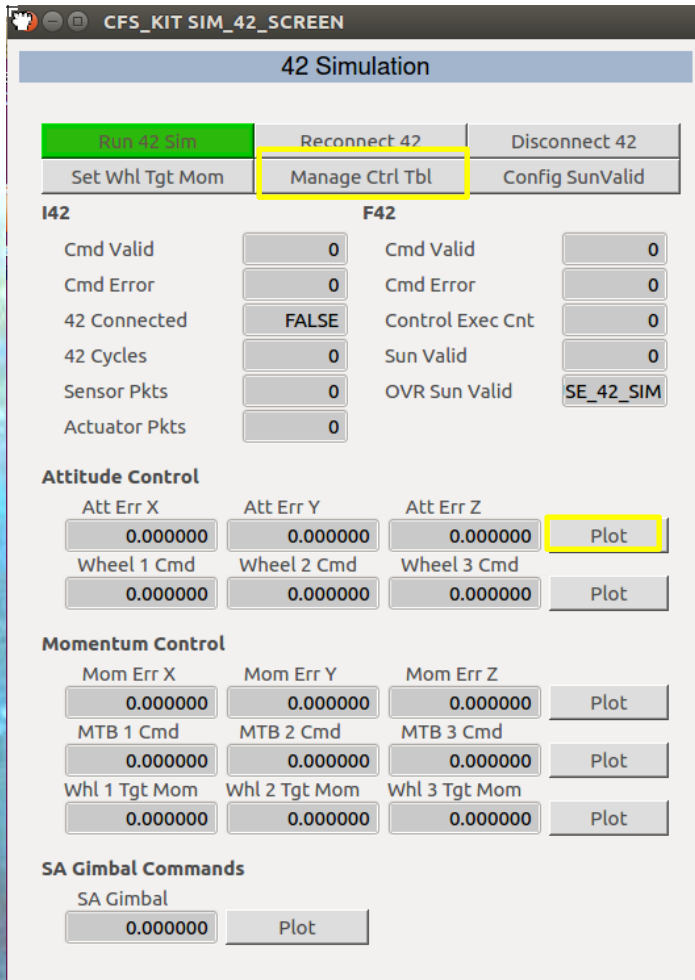
# Tools: Benchmarks

---

**Coming Soon...**



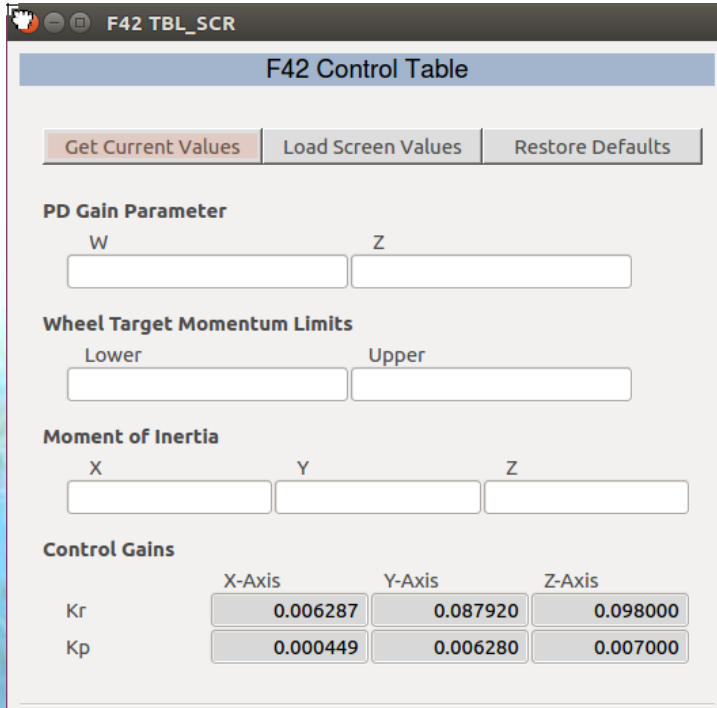
# Tools: Preparing 42 Simulation



The screenshot shows the '42 Simulation' control interface. At the top, there are three buttons: 'Run 42 Sim' (highlighted in green), 'Reconnect 42', 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', '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'. Below these are three sections: 'Attitude Control', 'Momentum Control', and 'SA Gimbal Commands'. Each of these sections contains numerical readouts for various error and command values, and 'Plot' buttons. The 'Attitude Control' section has a 'Plot' button highlighted with a yellow box. The 'Momentum Control' section has three 'Plot' buttons. The 'SA Gimbal Commands' section has 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

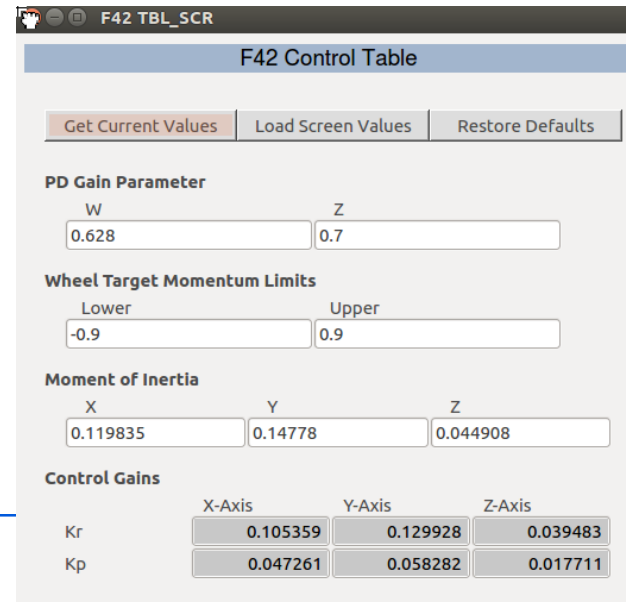
**Moment of Inertia**

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

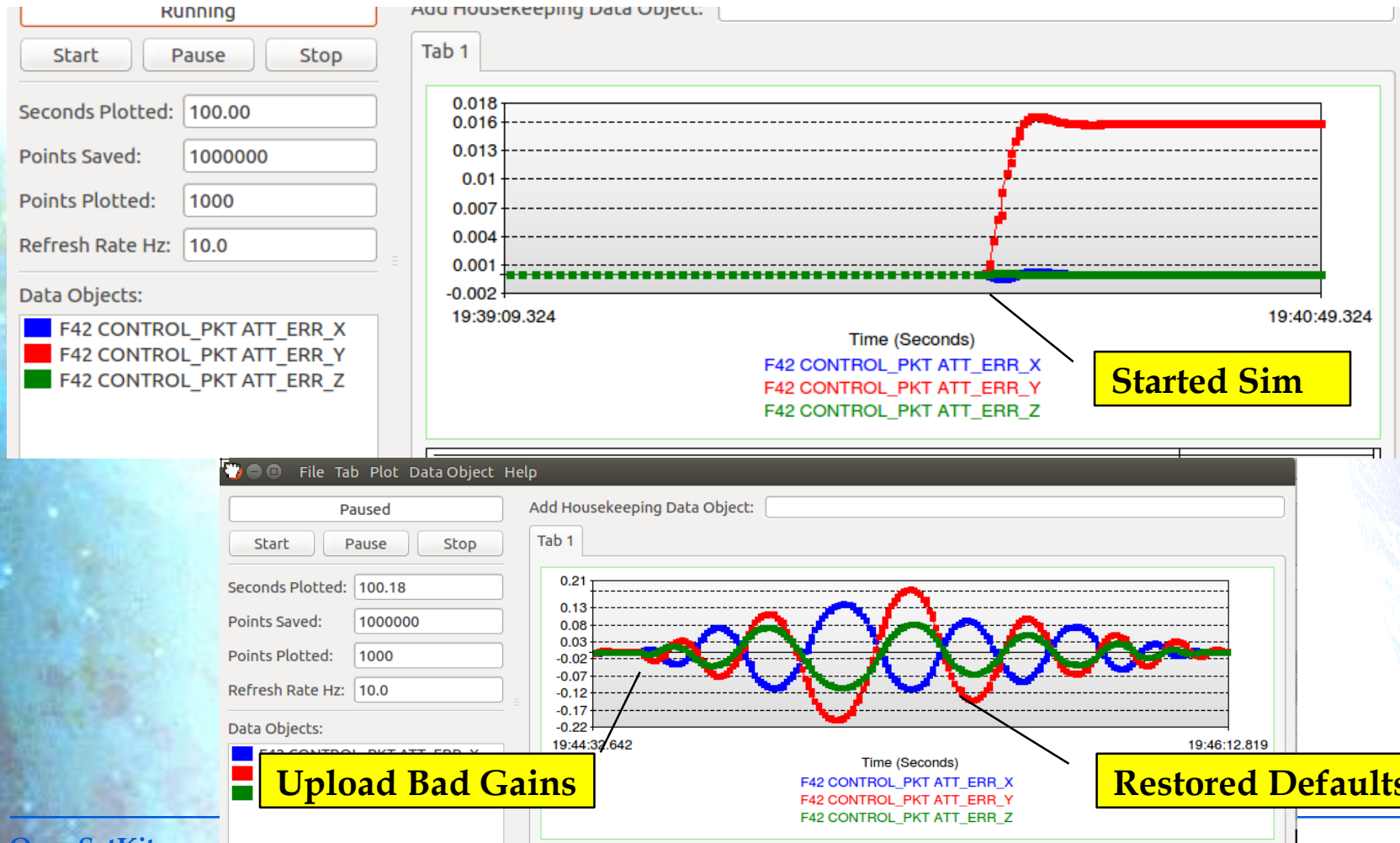
0.119835 0.14778 0.044908

**Control Gains**

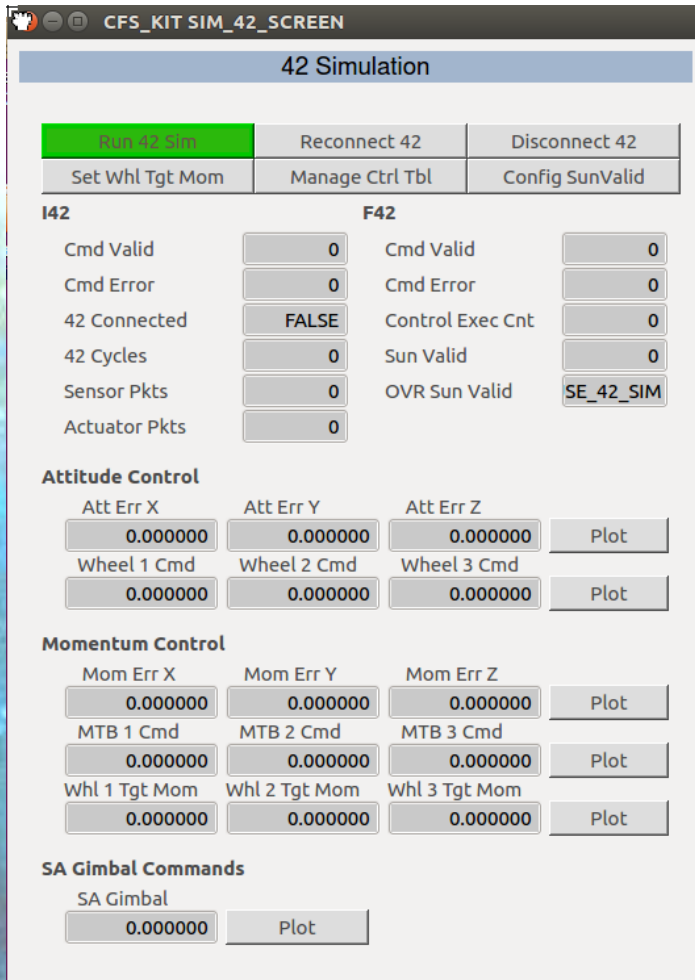
	X-Axis	Y-Axis	Z-Axis
Kr	0.105359	0.129928	0.039483
Kp	0.047261	0.058282	0.017711

# 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'. Each section contains several status indicators and numerical values, some with 'Plot' buttons. The 'Attitude Control' section includes '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). The 'Momentum Control' section includes '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). The 'SA Gimbal Commands' section includes 'SA Gimbal' (0.000000) and a 'Plot' button. The 'OVR Sun Valid' button is labeled 'SE\_42\_SIM'.

- 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

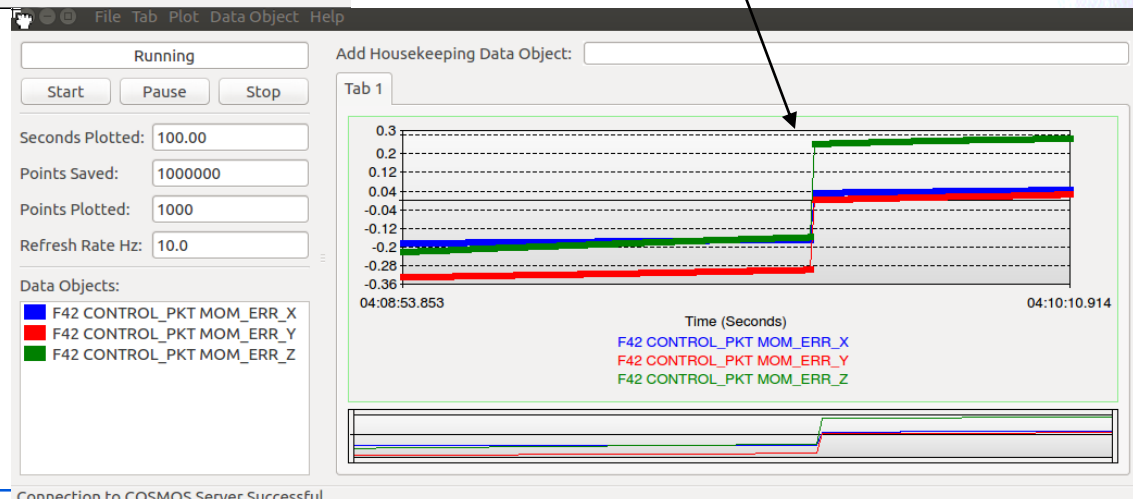
**Command Target Momentum**

Wheel 1	Wheel 2	Wheel 3

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

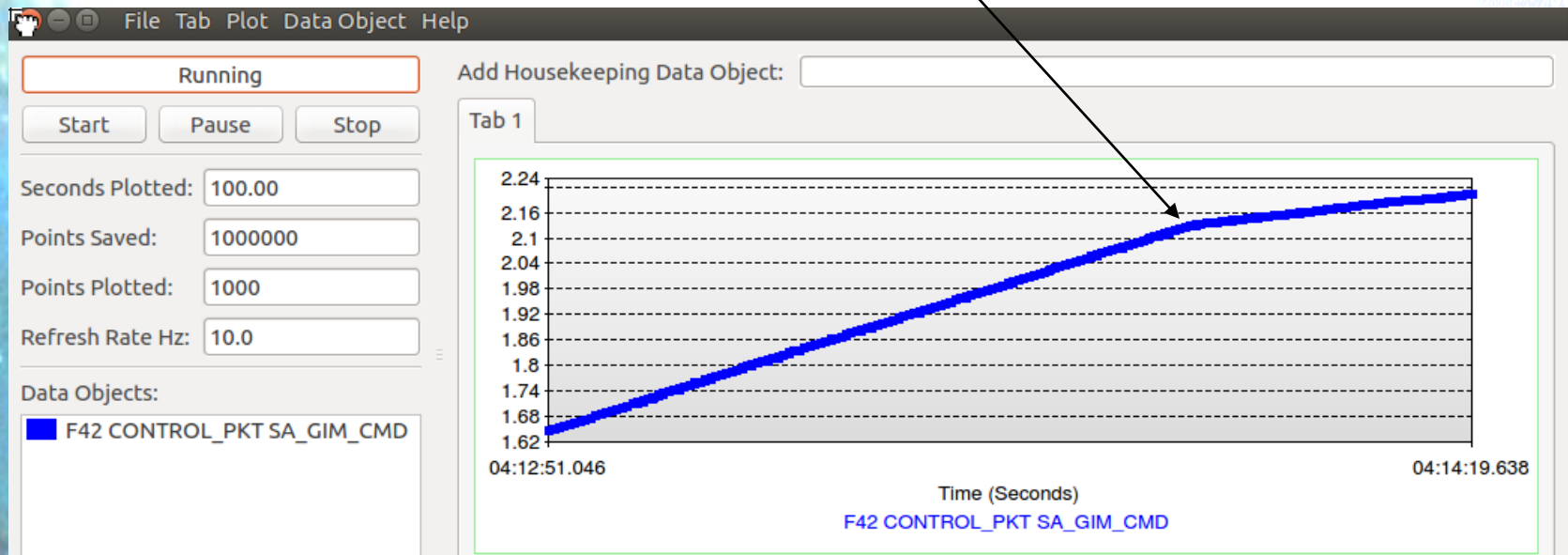
**Send** **Cancel**

- 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.



# Tools: PiSat Control

---

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

# 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

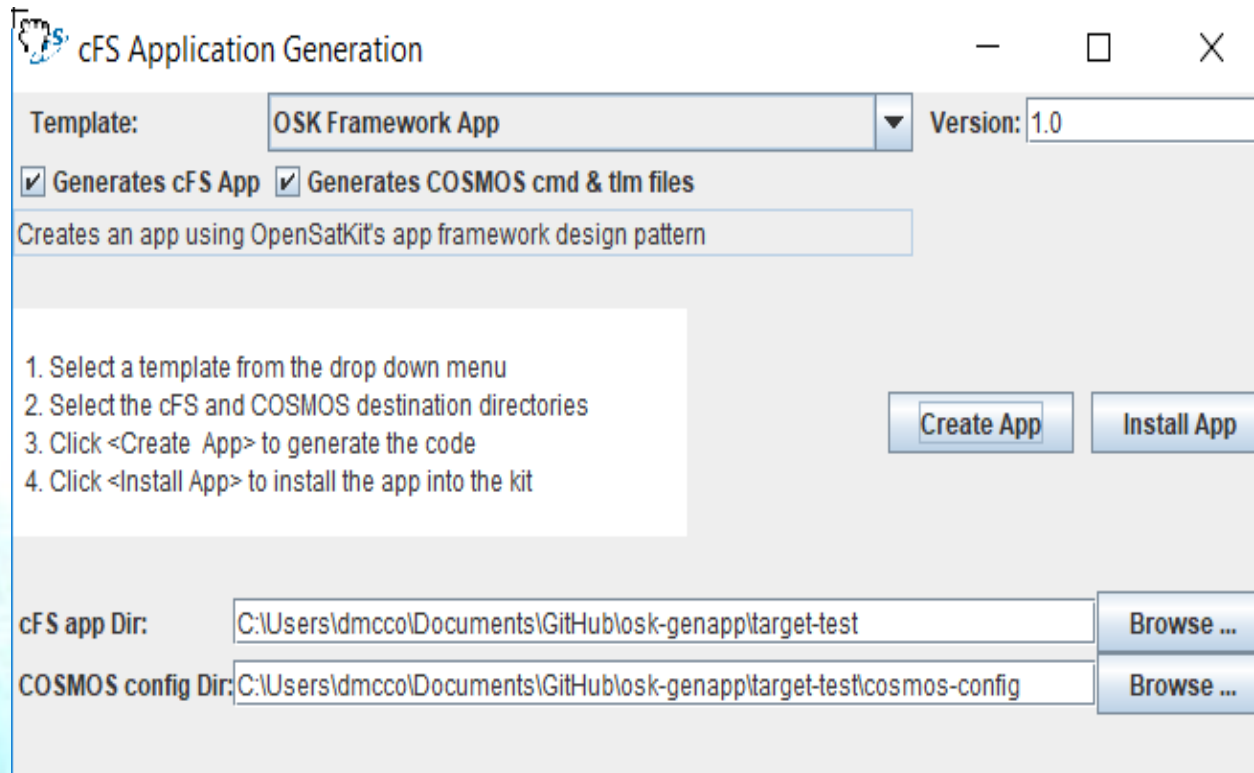
 Manually start cmd-tlm server from COSMOS launcher
- Start cFS

 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

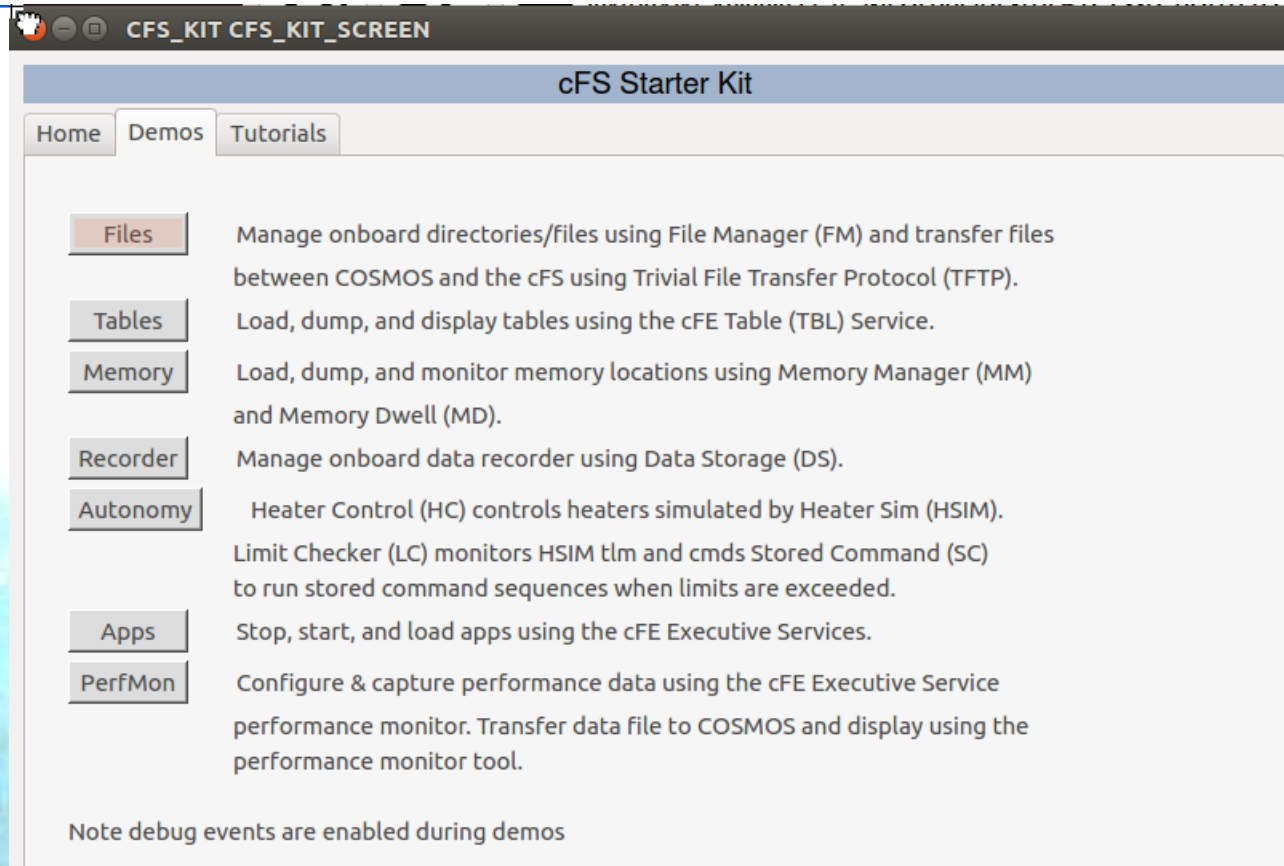


- Follow the instructions in the center of the dialogue. Create app generates the fsw source/make files, the cosmos target, and edits the COSMOS cmd-tlm-server config file.
- *<Install App>* has not been implemented. Follow the instructions on the previous slide

# Demos

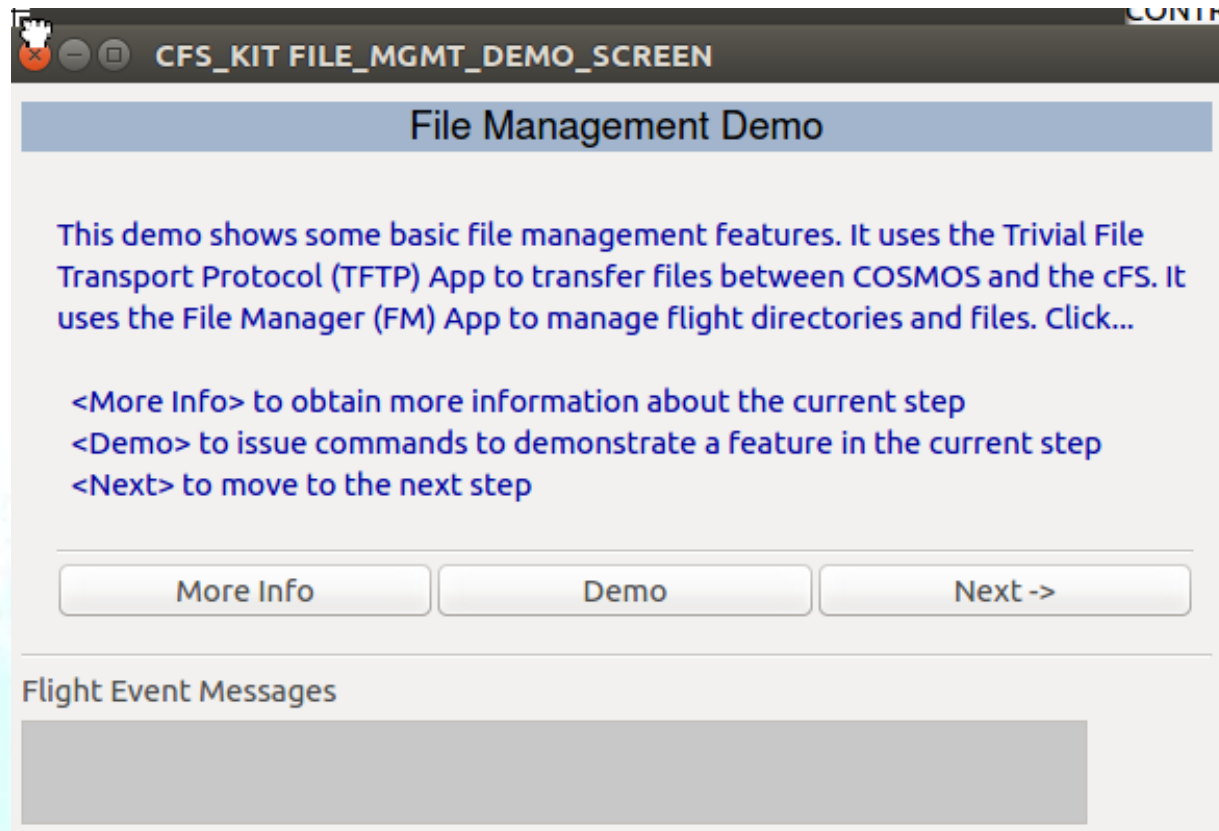


# Demo Tab

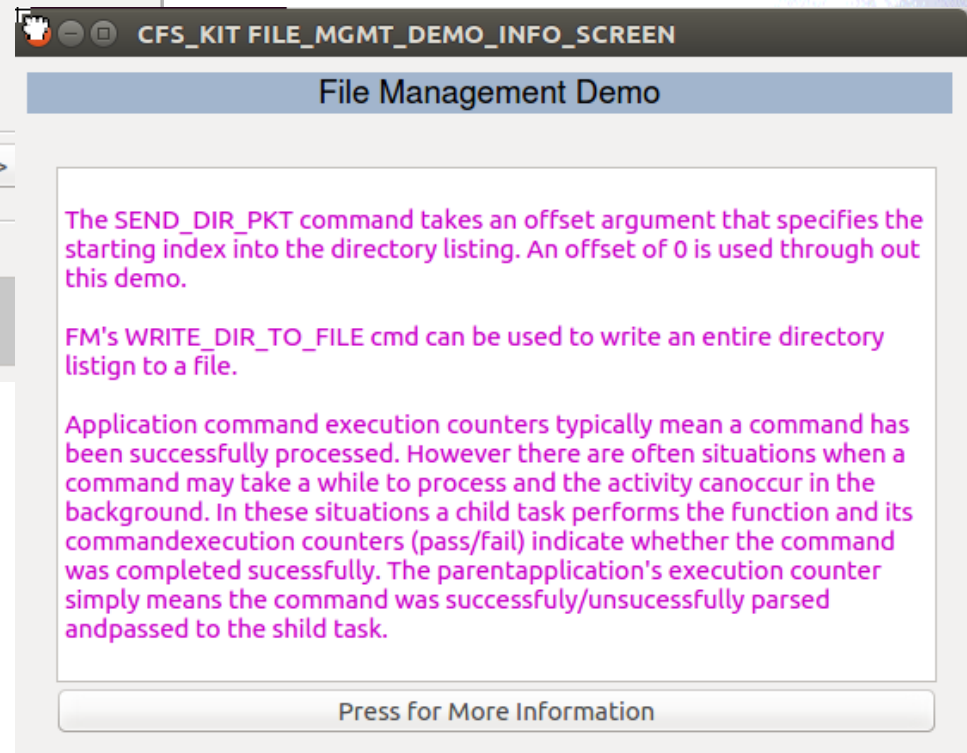
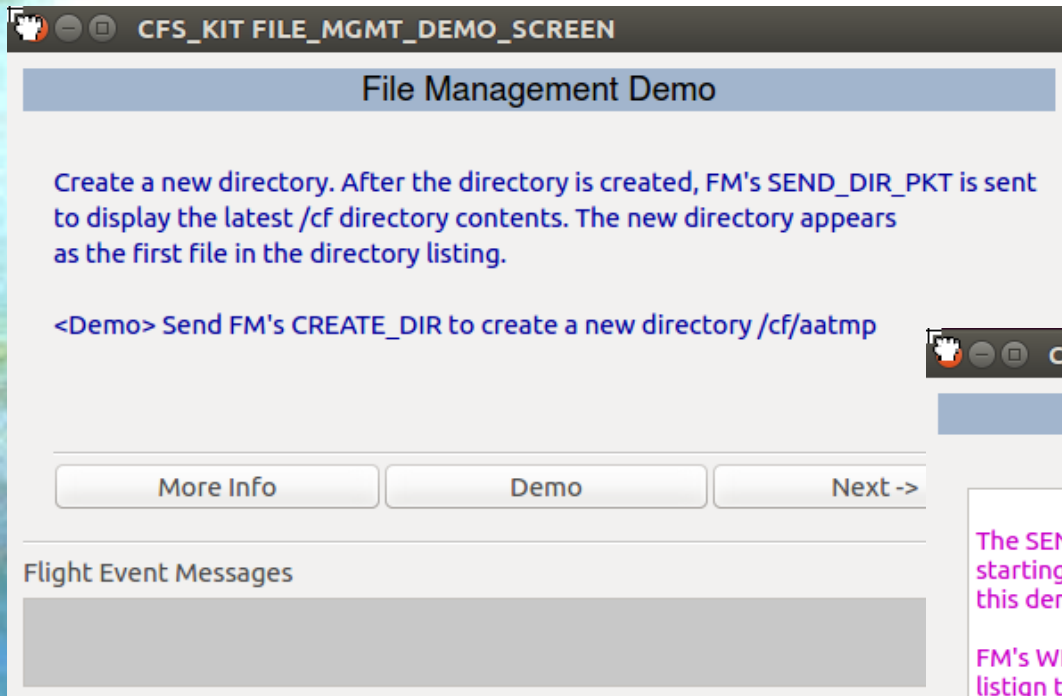


- The first six demos correspond to the six functional buttons on the Home tab
- Each demo follows a common user screen configuration that is described on the following slides

# Demo Structure – FM Example (1 of 2)



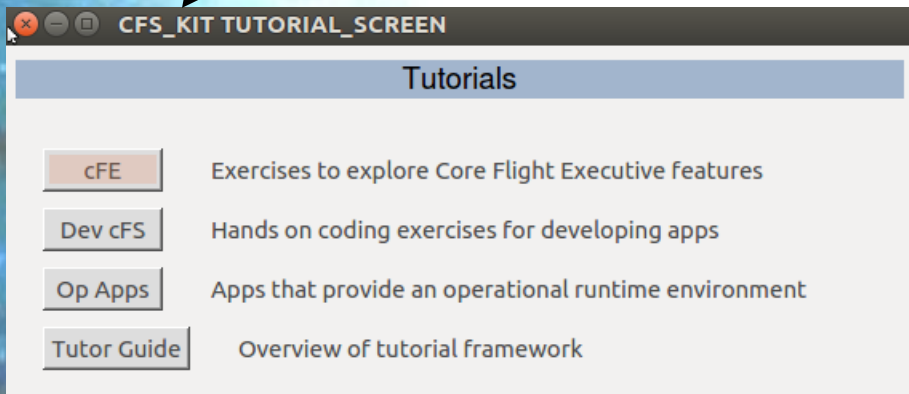
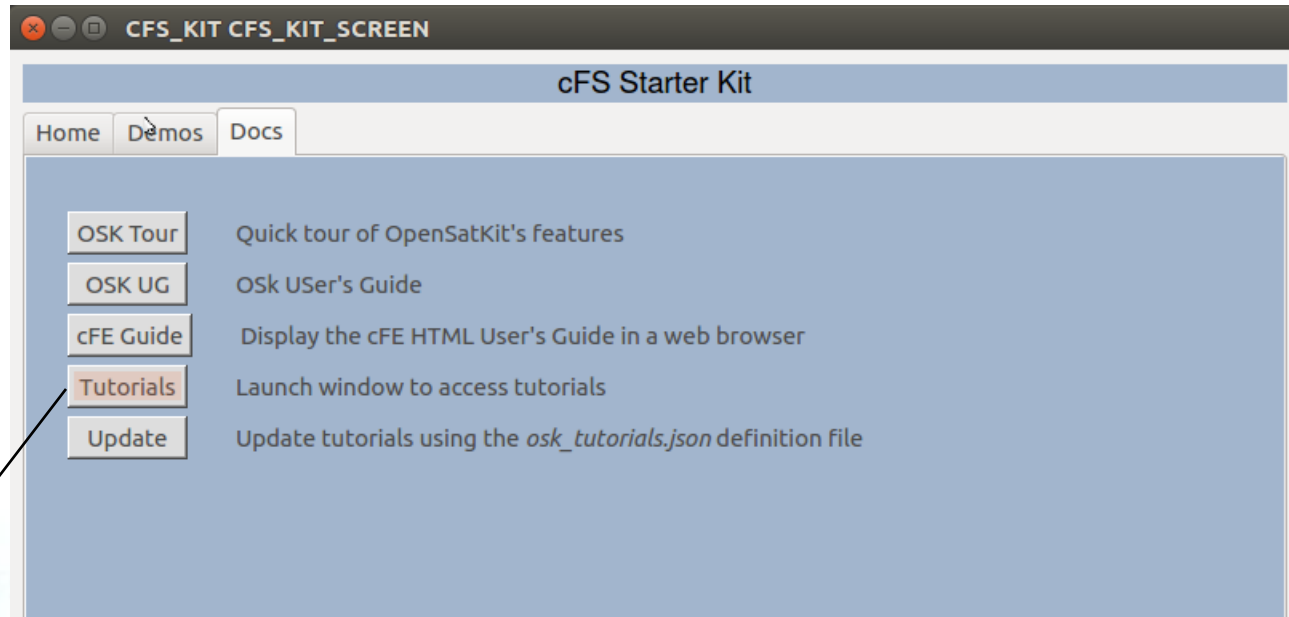
# Demo Structure – FM Example (2 of 2)



# Docs



# Docs Tab



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