OSK Application Development Tutorial

Developing apps using the OSK C Framework

v3.1

Introduction

- **Tutorial Guide Objectives**
  - Describe how the tutorial is structured and how a user works through the lessons
  - Describe OSK Demo App's (OSK_C_DEMO) role in the tutorial
- **Tutorial Objectives**
  - Teach how to develop cFS apps using the OSK C Framework
- **Tutorial Prerequisites**
  - Familiarity with the cFS design
  - Working knowledge of the C programming language

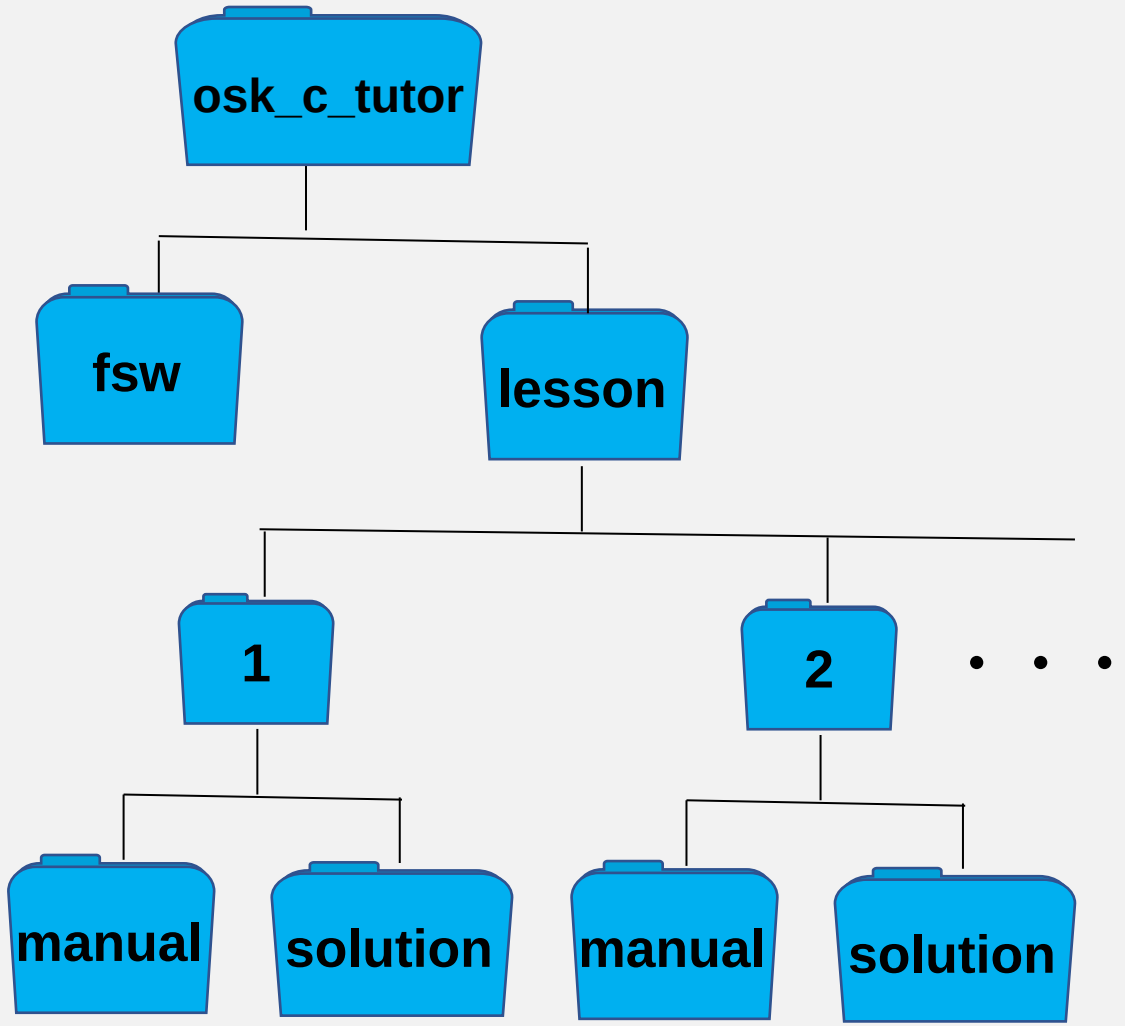
Tutorial Approach (1 of 2)

- **Use a series of self-guided lessons that instruct a user to transform a "Hello World" app into the OSK_C_DEMO app**
- **OSK_C_DEMO is a fully functioning app that is delivered as part of the OSK R&D sandbox target**
  - This guide includes a high-level description of osk_c_demo
  - The OSK Application Developer's Guide provides design and implementation details
- **The tutorial app is developed in cfs/apps/osk_c_tutor so the demo app in cfs/apps/osk_c_demo remains intact**
- **The osk_c_tutor app uses the same file names and messages IDs as the osk_c_demo app**
  - Since the tutorial is developed within the cfsat target there aren't any conflicts with osk_c_demo in the sandbox target

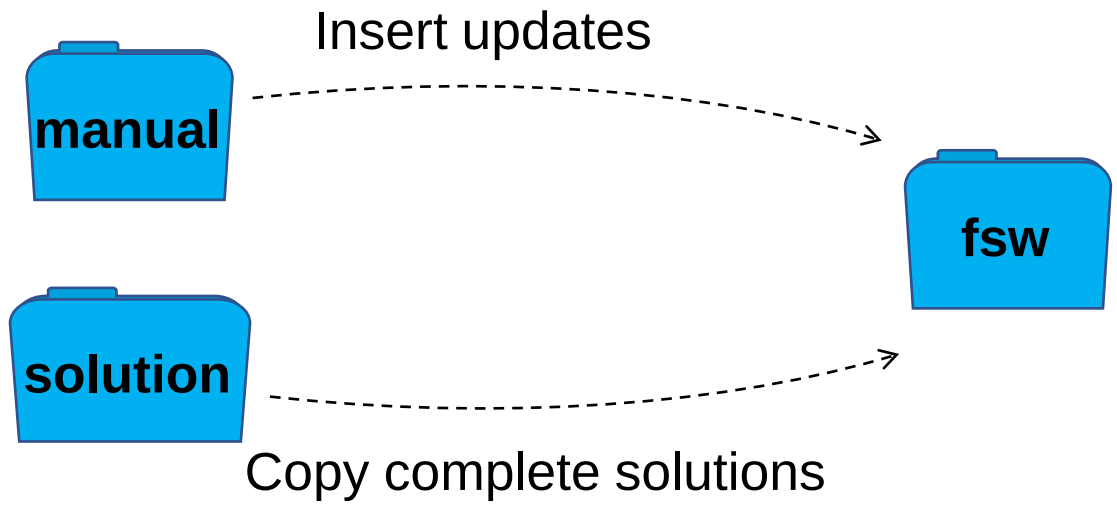
Tutorial Approach (2 of 2)

- **The first lesson creates all cmake, JSON table, and COSMOS artifacts**
- **Only osk_c_tutor/fsw/ source code needs to be updated for each of the following lessons**
  - Allows users to focus on developing OSK Framework-based apps
  - osk_c_tutor/fsw/ contains seven source files (see osk_c_demo section)
  - Not every source file needs to be updated in each lesson
- **The lessons are designed to help illustrate the benefits of the object-based framework**
  - Lesson notes highlight which files are updated in each lesson based on the lesson's objectives showing loose coupling between objects with well defined interface points
- **Lessons designed so each incremental version of the app can be built and run**
  - The app revision is incremented in each lesson to help verify the lesson was properly built and loaded
  - The *demo_ops_screen* can be used to interact with each incremental instantiation even though all of the app features are not available in each step
  - The test and ops script will not successfully run until the last lesson is completed

Tutor app

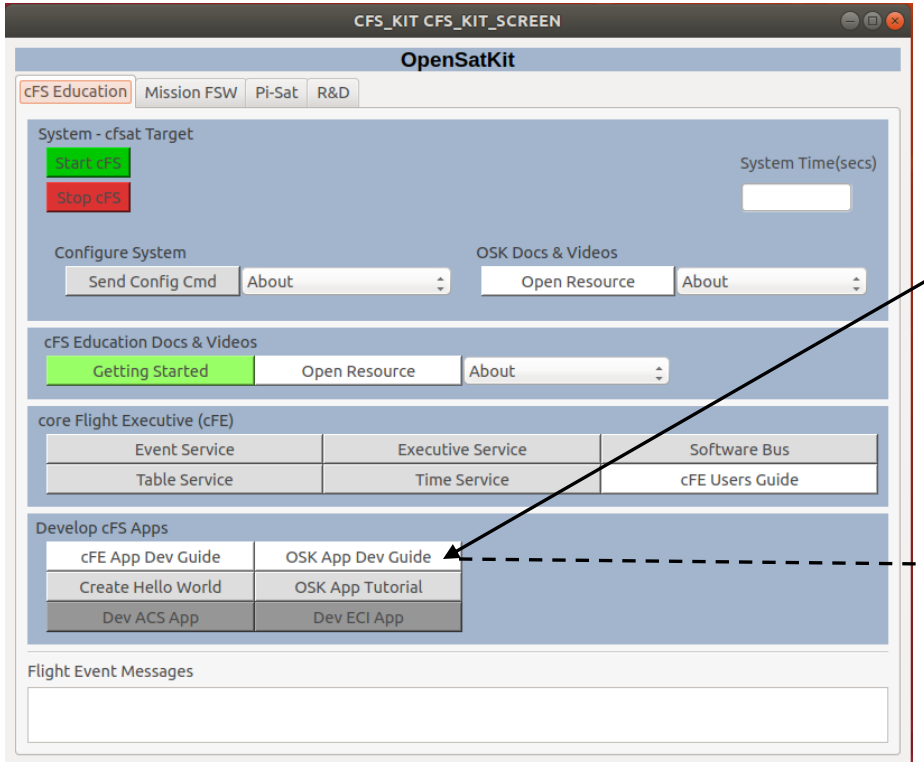


During each lesson a user can manually insert updates to the source files in the fsw directory or copy complete solutions

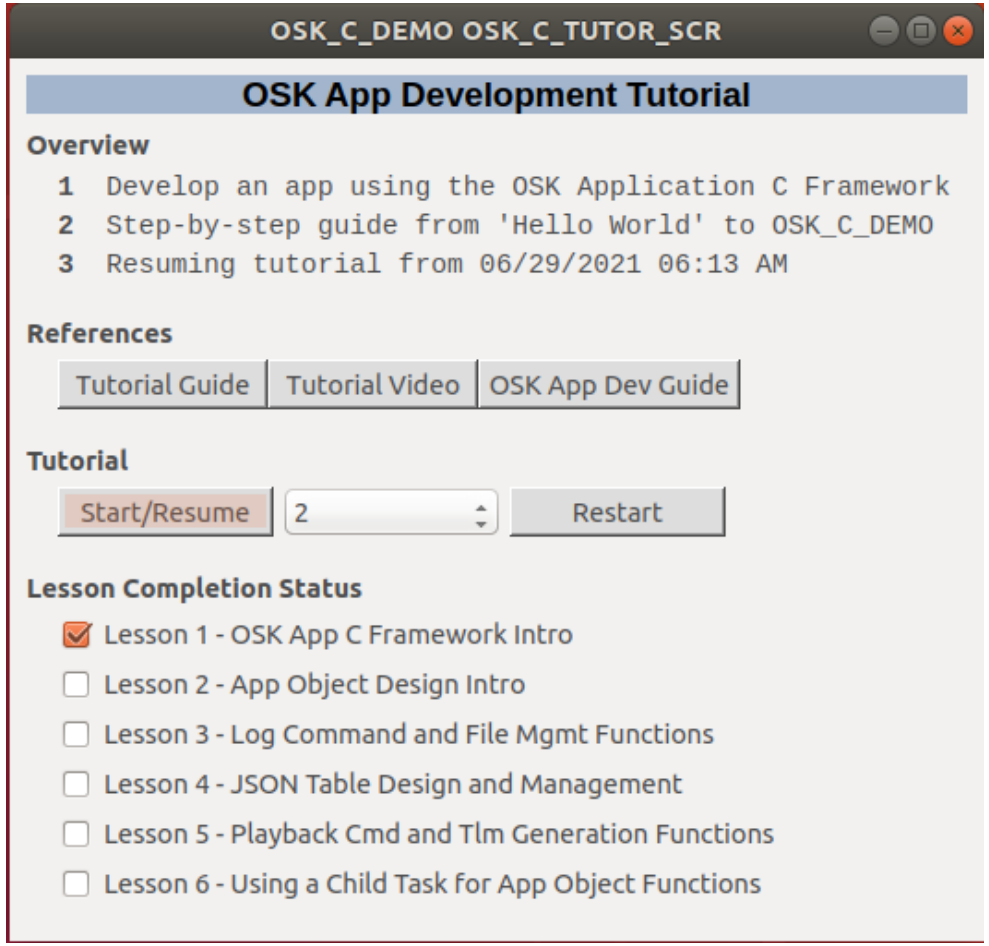


Manual insertion is recommended to get the most benefit from the tutorial

Launching the Tutorial



On the cFS Eng Edu tab launch the tutorial by selecting OSK App Tutorial

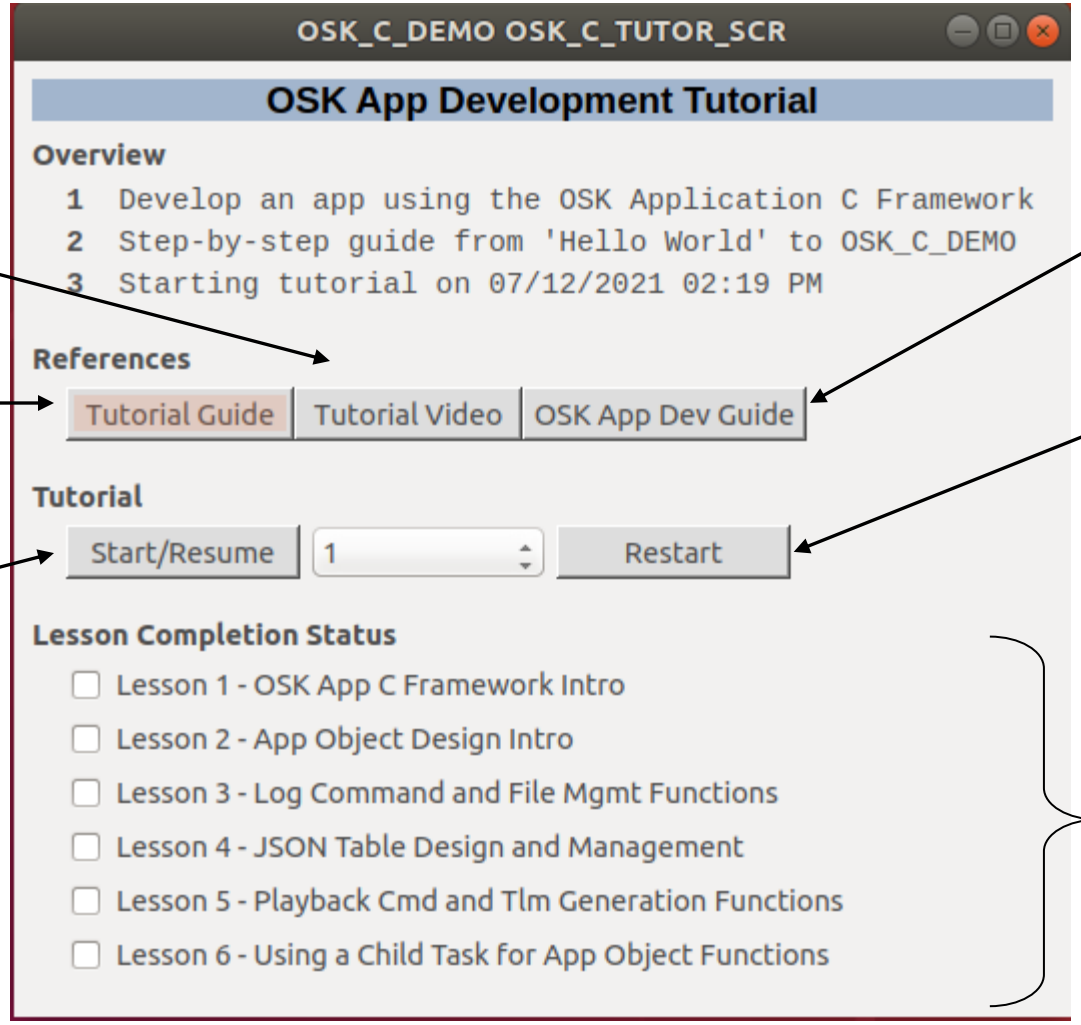


Tutorial Screen Overview

YouTube video demonstrating how to use the tutorial

Opens this guide you're currently reading

Start or resume a lesson. Drop-down menu lists all lesson up to the current lesson so you can go backwards



OKS App Developer's Guide

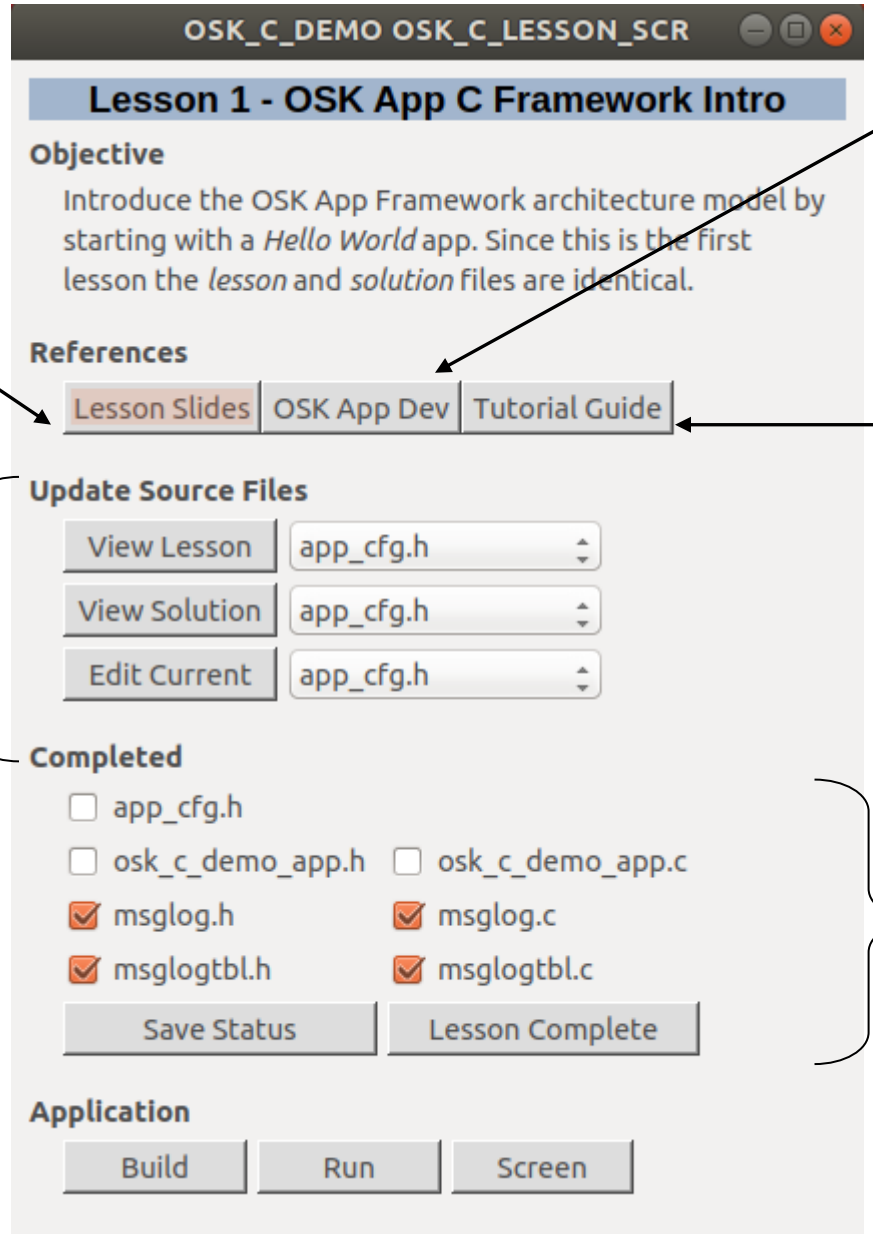
Removes the files from the osk_c_tutor/fsw directory, clears the lesson status, and starts with lesson 1

Tracks completed lessons

Lesson Screen Overview

Opens a set of slides specific to the lesson

OKS App Developer's Guide



Opens this guide you're currently reading

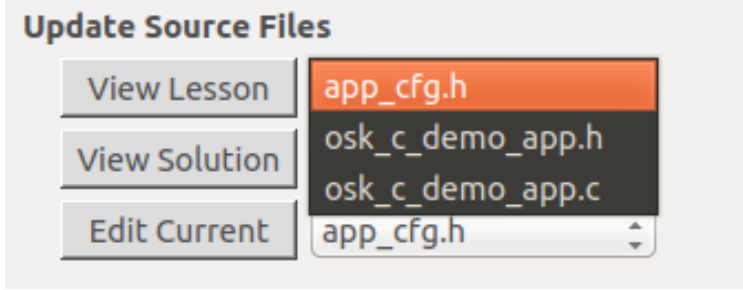
See next slides

See next slides

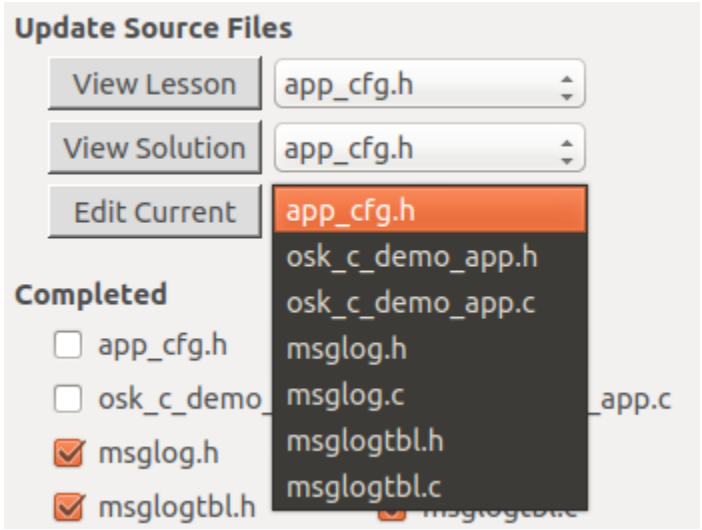
Manually track file update status

- Some files marked as complete if they don't need to be updated in the lesson
- Lesson complete checks the lesson complete box on the tutorial main screen and advances to the next lesson

1. **Use *View Lesson* or *View Solution* to open a file in a text window**
   - View Lesson provides instructions for making updates to the source file
   - View Solution provides a complete source file that can be cut and pasted into the
   - Only the files that need to be updated for the lesson appear in the drop-down menu



2. **Use *Edit Current* to open an osk_c_tutor/fsw/* source file in the COSMOS text editor that needs to be updated for the lesson**
   - All of the files are available for editing, but you should only need to edit the files in the lesson if everything is progressing normally
   - You can use any text editor you like, just be sure to edit the source file in osk_c_tutor/fsw/*

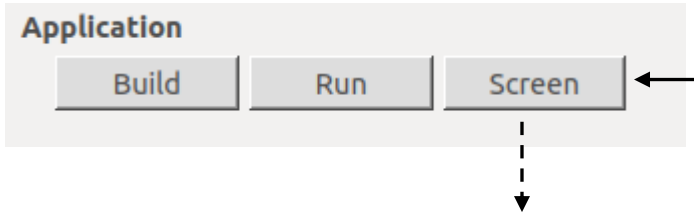


Edit, Build, Run, & Test Lessons (2 of 3)

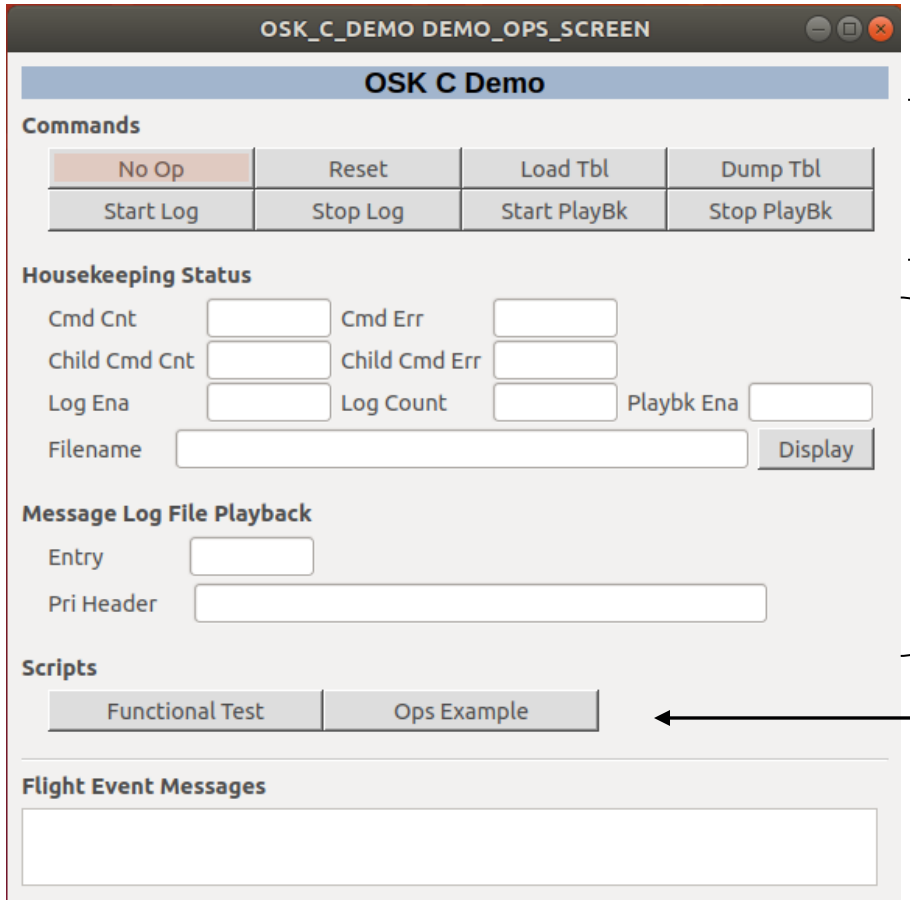
3. **Select the Build button after all the lesson updates are complete**
   - This opens a new terminal window and automatically invokes the build process
   - If the build fails, use the compiler errors to diagnose the errors and edit the osk_c_tutor/fsw source files that need correction

4. **After the lesson successfully builds, select the Run button to run the new target**
   - The cFS target is restarted because the loading the new app dynamically does not currently work on Linux

Edit, Build, Run, & Test Lessons (3 of 3)



5. **Select the Screen button to show the DEMO_OPS_SCREEN**



Access all commands to interactively test each lesson
- Functionality increases with each lesson
- Some commands not available in early lessons

Use telemetry to verify commands
- Functionality increases with each lesson
- Some telemetry not available in early lessons

Scripts can only be run after the final lesson is complete

Lesson Outline

Lesson	Framework	Description
1	Arch, initbl, cmdmgr	Initial “hello world” app with Noop & Reset commands and HK telemetry
2	Arch, cmdmgr	Add a ‘skeleton’ message log object with command stubs
3	fileutil	Add message log file functionality with hardcoded parameters
4	tblmgr	Add a table and define message log file parameters
5	Arch, fileutil	Add playback functionality
6	childmgr	Manage file logging and playback functions within child task context

Refer to the following apps for examples using framework components not covered in this tutorial

- OSK_C_PROTO : State Reporter
- KIT_TO: Packet Utilities


OSK_C_DEMO App

OSK_C_DEMO App's Tutorial Role

- **Is a fully functioning cFS app that is delivered as part of OSK's Research & Development (R&D) Sandbox target**
- **Uses many of OSK's C app framework (OSK_C_FW) library features**
- **Serves as the end-goal for the app development tutorial**
  - The tutorial starts with a "hello world" app and each step adds more features/functions that ultimately create OSK_C_DEMO

OSK_C_DEMO Overview

- **Upon command start logging the primary header of the command-specified message ID**
  - The header is written as hexadecimal text
  - Logging stops when a table-defined number of entries have been written or when the user issues a command to stop logging
- **Upon command playback in telemetry the contents of the message log file**
  - One header is contained in each playback telemetry message
  - A table-defined value specifies the delay between telemetry messages
  - The playback loops through the message log file until a stop playback or start new log command is received

OSK_C_DEMO Table

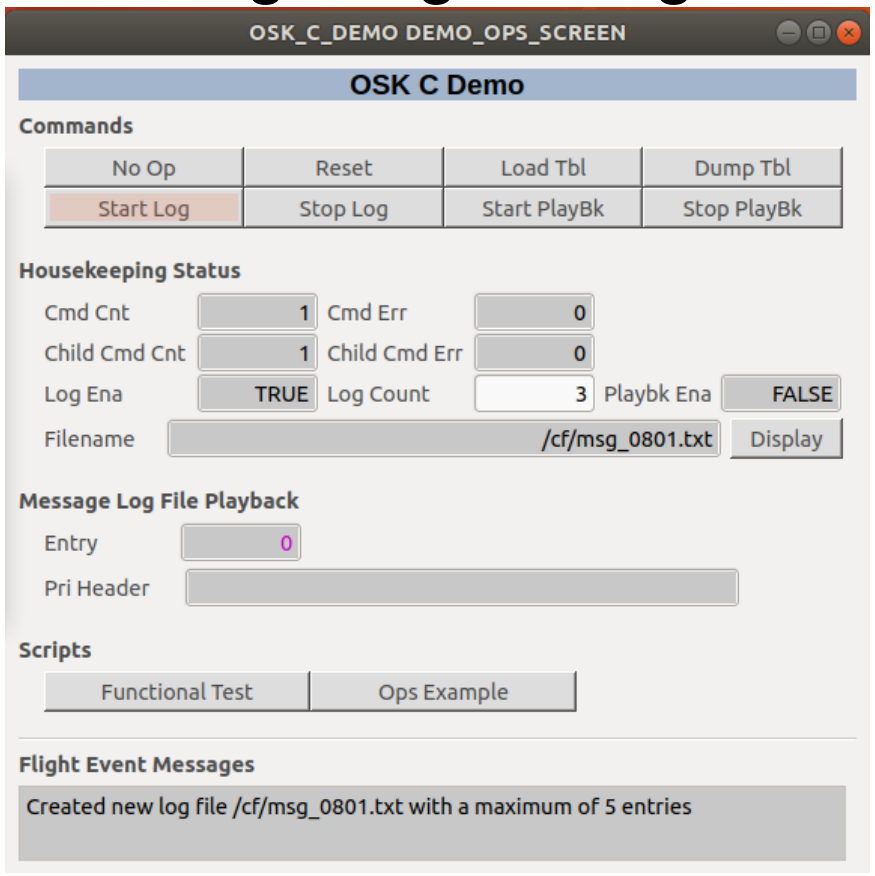
osk_c_demo.json

```
{
    "app-name": "OSK_C_DEMO",
    "tbl-name": "Message Log",
    "description": "Define parameters for demo message logger",
    "file": {
        "path-base-name": "/cf/msg_",
        "extension":      ".txt",
        "entry-cnt":      5
    },
    "playbk-delay": 3
}
```

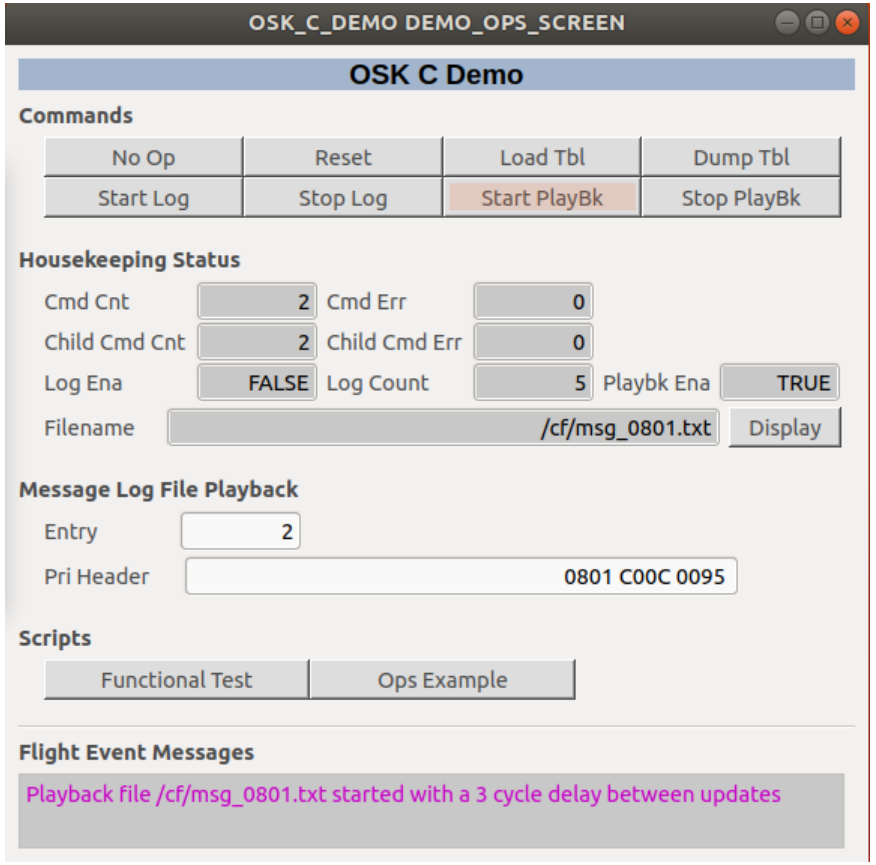
- **Message log file name created by concatenating *"path-base-filename"*, command-specified message ID, and *"extension"***
  - e.g. Sending the OSK_C_DEMO start log command ith a parameter of 0x0801 (cFE EVS housekeeping telemetry message) results in a log filename of "msg_0801.txt"
- ***"entry-cnt"* defines maximum number of message log file entries**
- ***"playbk-delay"* defines number of OSK_C_DEMO execution cycles between playback telemetry messages**

OSK_C_DEMO Ops Screen

Message Log in Progress



Log File Playback in Progress



- cFE event service housekeeping message (ID = 0x0801) logged
- A child task performs logging and playback
- *"Display"* button transfers log file to ground and displays it in a text window

Source Files Updated in the Tutorial