Open STEM ware

"NASA World" NASA Style App Coding Lessons

Basecamp

Basecamp Version 2.7

July 2025

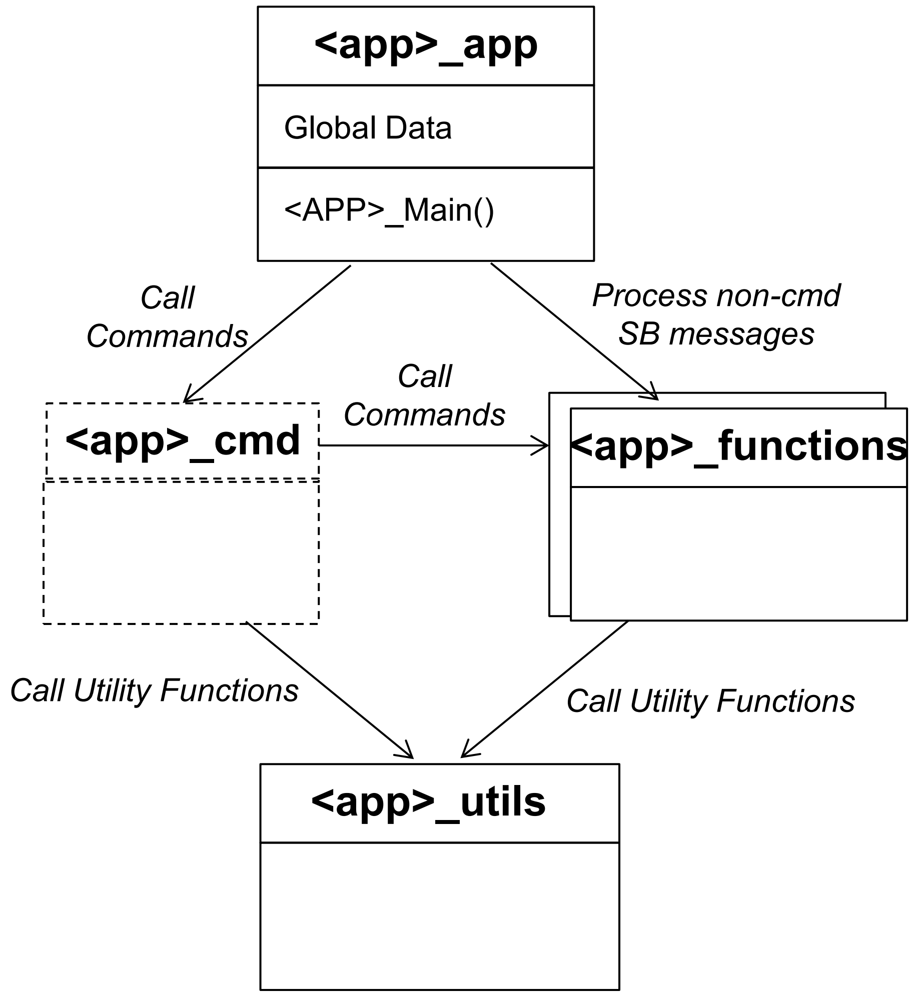
Introduction

- **These slides supplement the *NASA World* (i.e. "Hello World") coding tutorials**
  - If you change the *NASA World* default name use a name of equal or shorter length to avoid exceeding file and table name length limits
- **The goal is to teach the NASA app design style and essential app components**
  - See the *cFE Application Devloper's Guide* for details **https://github.com/nasa/cFE/blob/main/docs/cFE%20Application%20Developers%20Guide.md#table-of-contents**
- **Tutorial approach:**
  1. Generate an app using the NASA World app template to create a minimal cFS application
  2. The application design follows the NASA app design conventions described in the next few slides
  3. The initial app is a pared down version of NASA's Sample App https://github.com/nasa/sample_app
  4. The coding exercises in this and subsequent tutorials introduce developers to the different app components that result in an app that includes most of the features of the Sample App.
- **Prerequisites**
  - Working knowledge of the C programming language
  - Familiarity with Basecamp's GUI operations covered by the built-in introduction tutorial
  - Basic understanding of flight software context, the cFS architecture, and the cFS Application Developer's Guide

NASA App Design Style (1 of 3)

- **The NASA app designs don’t follow a rigid design pattern but they do have similar design structures**
- **The main app file defines a global data structure that is accessed by functions that can reside in any of the app’s source files**
  - These apps were designed when onboard memory and processor speeds were significantly constrained
  - Sharing global memory reduces memory footprints and avoids excessive memory copying
  - Global memory can also simplify in-orbit patches
- **The main app file contains**
  - The app’s entry point called by Executive Services and the app’s initialization function that registers with cFE services
  - The app’s main loop of execution
  - Housekeeping telemetry generation
  - The no operation and reset counter commands
  - Apps are less consistent once beyond these basic functions
- **Functions typically contained in files outside of main are invoked by commands and software bus messages**
- **Note the cFS Framework does <u>not</u> dictate a particular app design strategy**
  - Basecamp apps use an object-based design implemented in C
  - There are examples of C++ apps within the cFS community

NASA App Design Style (2 of 3)



- Global memory can be read and written by any module (not shown)
- Not all apps have an <app>_cmd file and command dispatching is performed in the main file

NASA App Design Style (3 of 3)

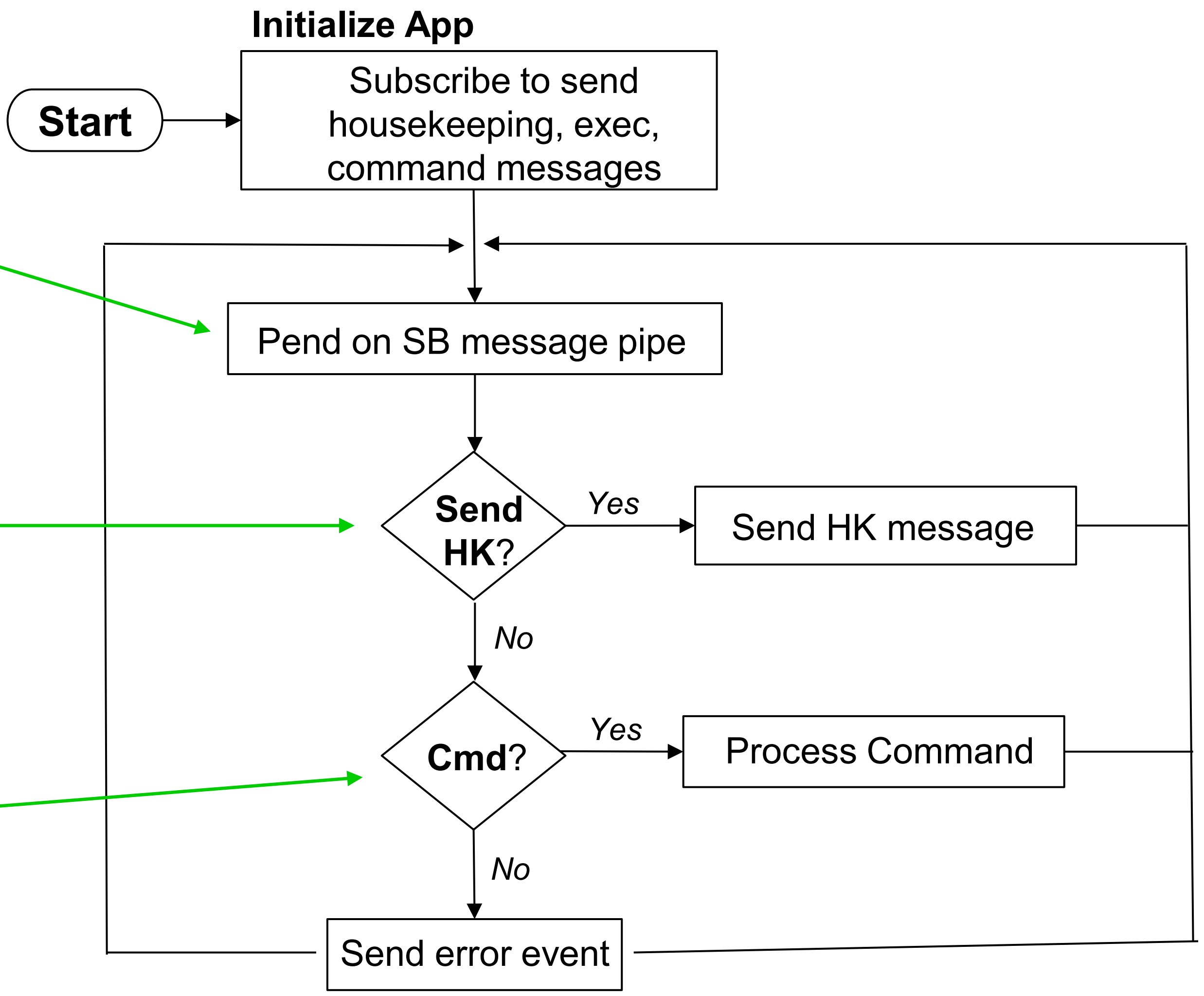
Suspend execution until a message arrives on app's pipe

Periodic *request housekeeping* message from SCH app

- Typically, on the order of seconds
- "Housekeeping cycle" convenient time to perform non-critical app functions

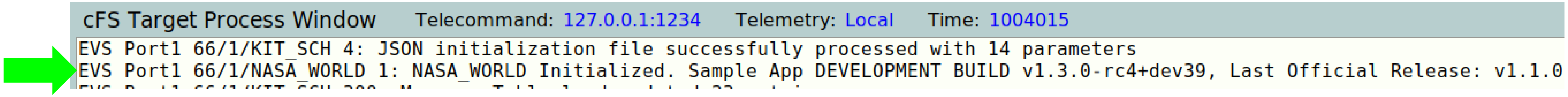
Process commands

- Commands can originate from ground or other onboard apps



NASA World Operations (1 of 3)

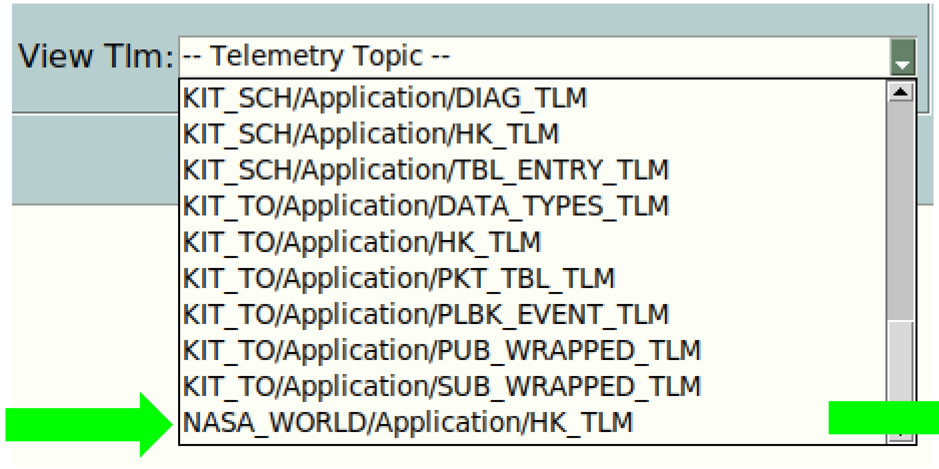
- **After generating *NASA World*, start the cFS target using the cFS <Start> button**
  - Scroll up in the cFS Target Process Window and you should see the following event message indicating the NASA World app successfully started



```
EVS Port1 66/1/KIT_SCH 4: JSON initialization file successfully processed with 14 parameters
EVS Port1 66/1/NASA_WORLD 1: NASA_WORLD Initialized. Sample App DEVELOPMENT BUILD v1.3.0-rc4+dev39, Last Official Release: v1.1.0
```

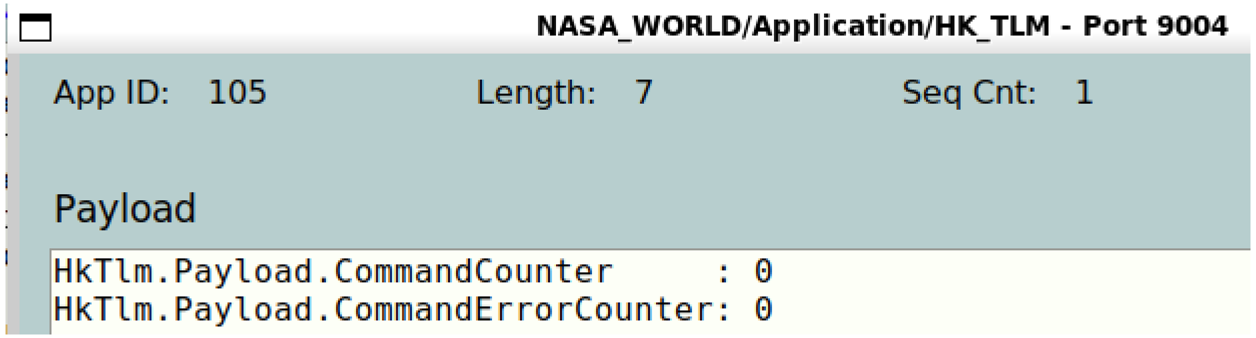
NASA World Operations (2 of 3)

- **Open the NASA World status telemetry message**
  - It only contains the valid and invalid command counters



View Tlm: -- Telemetry Topic --

```
KIT_SCH/Application/DIAG_TLM
KIT_SCH/Application/HK_TLM
KIT_SCH/Application/TBL_ENTRY_TLM
KIT_TO/Application/DATA_TYPES_TLM
KIT_TO/Application/HK_TLM
KIT_TO/Application/PKT_TBL_TLM
KIT_TO/Application/PLBK_EVENT_TLM
KIT_TO/Application/PUB_WRAPPED_TLM
KIT_TO/Application/SUB_WRAPPED_TLM
NASA_WORLD/Application/HK_TLM
```



NASA_WORLD/Application/HK_TLM - Port 9004

App ID: 105 Length: 7 Seq Cnt: 1

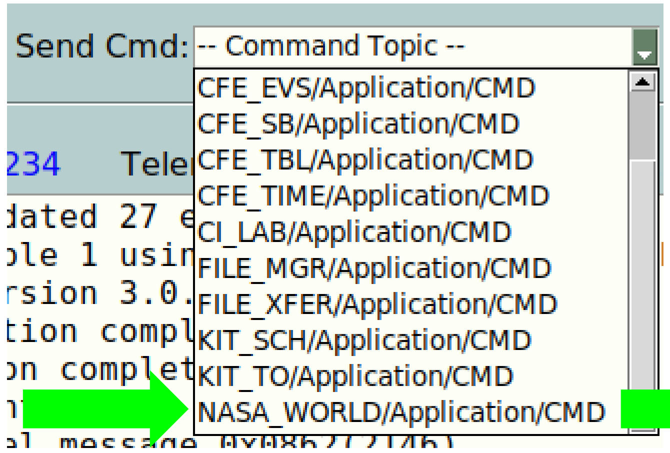
Payload

```
HkTlm.Payload.CommandCounter     : 0
HkTlm.Payload.CommandErrorCounter: 0
```

- **Status telemetry is sent every 4 seconds**
  - The app subscribes to receive the Scheduler App’s 4 sec message BC_SCH_4_SEC_TOPICID

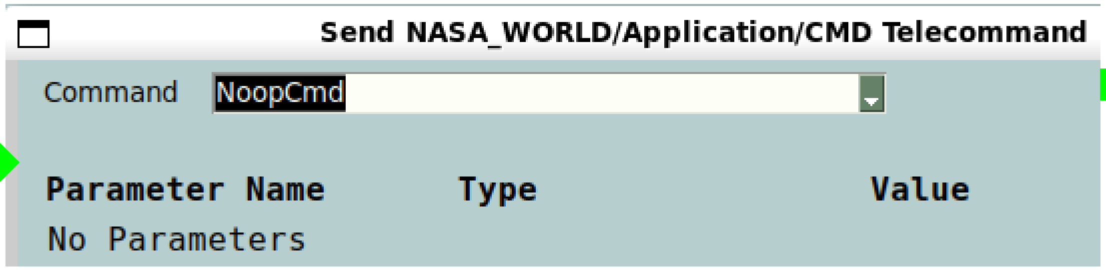
NASA World Operations (3 of 3)

- **Issue multiple NASA World Noop commands**



Send Cmd: -- Command Topic --

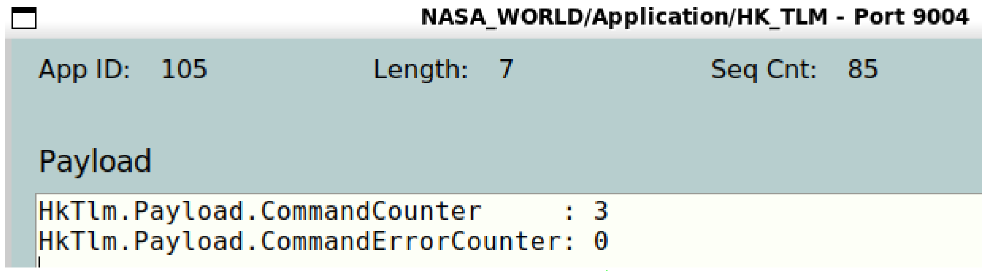
CFE_EVS/Application/CMD
CFE_SB/Application/CMD
CFE_TBL/Application/CMD
CFE_TIME/Application/CMD
CI_LAB/Application/CMD
FILE_MGR/Application/CMD
FILE_XFER/Application/CMD
KIT_SCH/Application/CMD
KIT_TO/Application/CMD
NASA_WORLD/Application/CMD



Send NASA_WORLD/Application/CMD Telecommand

Command NoopCmd

Parameter Name	Type	Value
No Parameters		



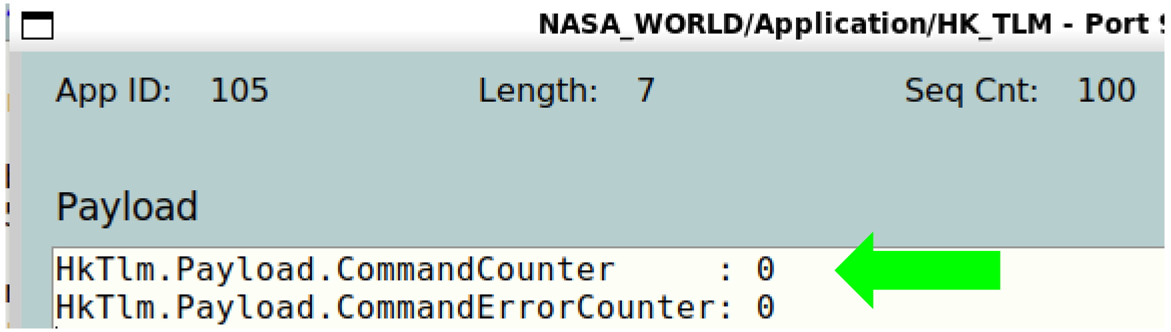
NASA_WORLD/Application/HK_TLM - Port 9004

App ID: 105 Length: 7 Seq Cnt: 85

Payload

```
HkTlm.Payload.CommandCounter     : 3
HkTlm.Payload.CommandErrorCounter: 0
```

- **Issue a NASA World Reset App command to clear the command counters**



NASA_WORLD/Application/HK_TLM - Port

App ID: 105 Length: 7 Seq Cnt: 100

Payload

```
HkTlm.Payload.CommandCounter     : 0
HkTlm.Payload.CommandErrorCounter: 0
```

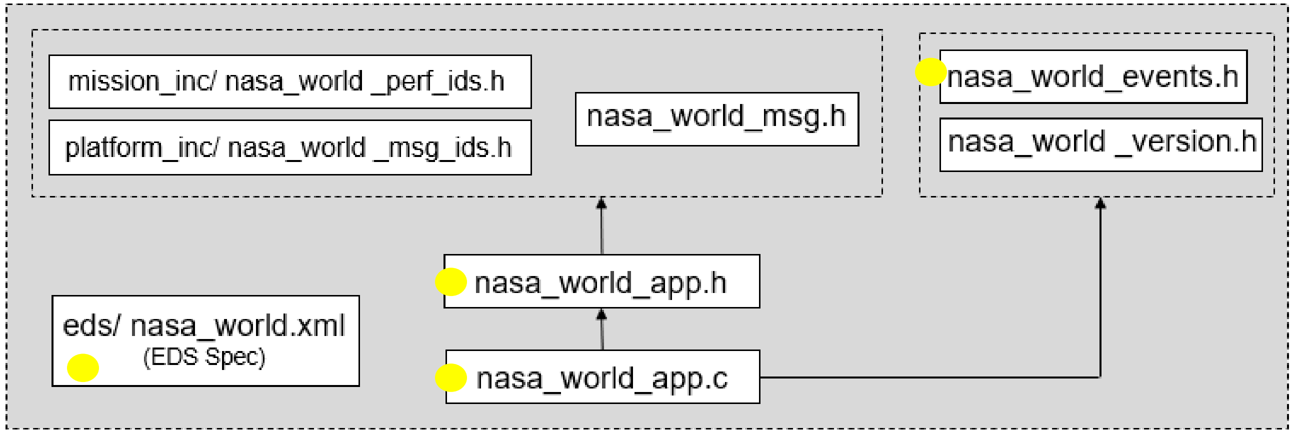
Lesson 1 – Add a Command (1 of 2)

Objectives

- Learn how to define commands using Electronic Data Sheets
- Learn how to dispatch a command using its function code

The following files are modified in this lesson

App Source Files



The new command is added to the main app C file to minimize the scope of changed files. In practice, the command function could be in a separate file that is related to the command's function.

Lesson 1 – Add a Command (2 of 2)

nasa_world.xml

- The new command is defined in two parts
- The *ExampleParamCmd_Payload* defines the command parameter
  - BASETYPES is an EDS package defined in the cFE EDS specs

nasa_world_msg.h

- As noted in the file prologue, this file is not modified because the headers generated from the EDS replace it

nasa_world_events.h

- The new command requires a new event message identifier and by convention the macro names end in _EID

nasa_world_app.h

- The new command function protype is added

nasa_world_app.c

- A call to the new command function is added to the NASA_WORLD_ProcessGroundCommand()'s

Lesson 1 – Build New cFS Target

1. **Use the main screen's cFS Build button to build the target**
   - Only existing files changed, so no need to perform a Build New

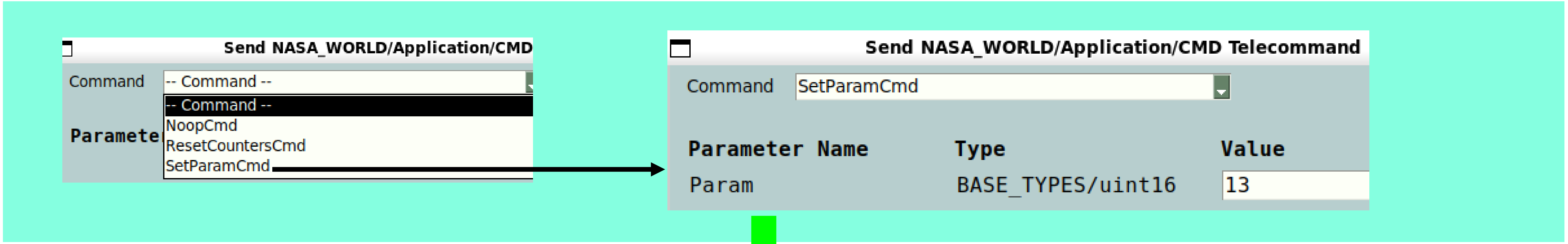
2. **Since the EDS was modified, the GUI must be restarted so the new EDS library with the new command is used**

3. **The following slides describe how to use the new command**

Lesson 1 – New Command Operations

Verify the new code by sending the new command, observing the event message and valid command counter

Telecommand

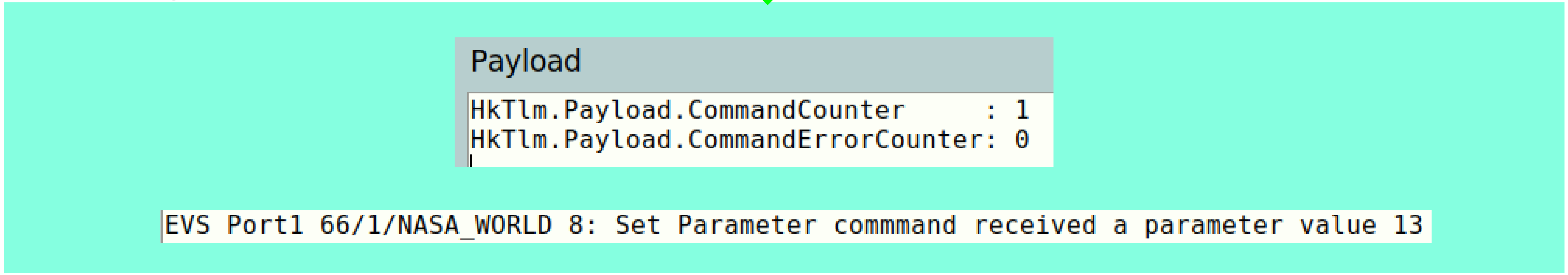


Send NASA_WORLD/Application/CMD Telecommand

Command: SetParamCmd

Parameter Name	Type	Value
Param	BASE_TYPES/uint16	13

Telemetry



```
HkTlm.Payload.CommandCounter      : 1
HkTlm.Payload.CommandErrorCounter: 0
```

```
EVS Port1 66/1/NASA_WORLD 8: Set Parameter commmand received a parameter value 13
```

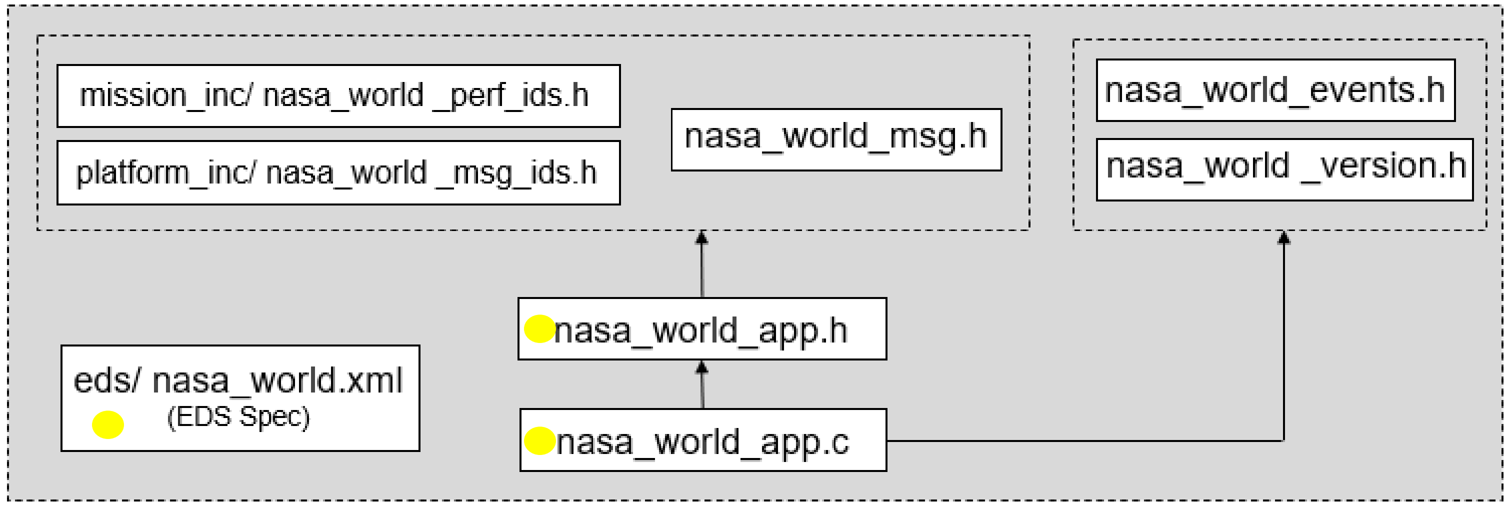
Lesson 2 – Add Telemetry Data Point (1 of 2)

Objectives

- Learn how to define telemetry messages using Electronic Data Sheets
- Introduce the concept of an app ”Housekeeping Cycle”

● **The following files are modified in this lesson**

App Source Files



Lesson 2 – Add Telemetry Data Point (2 of 2)

nasa_world.xml

- Since the *HkTlm_Payload* container type already exists, this change only requires a new <*EntryList*> entry

nasa_world _app.h

- A new variable needs to be added to save the command parameter so it can be sent in telemetry

nasa_world _app.c

- The new command parameter variable needs to be
  - Initialized in the app initialization function
  - Set in the set command parameter function
  - Copied to the housekeeping packet

Notes

- The *NASA_WORLD_HkTlm_t* structure is generated by the EDS toolchain
- The app's execution period when it sends its housekeeping telemetry is often referred to as the "housekeeping cycle"
- Apps often perform other low frequency activities in this housekeeping cycle such as table validation as you'll see in the *NASA Table* tutorial

Lesson 2 – Build New cFS Target

1. **Use the main screen's cFS Build button to build the target**
   – Only existing files changed, so no need to perform a Build New

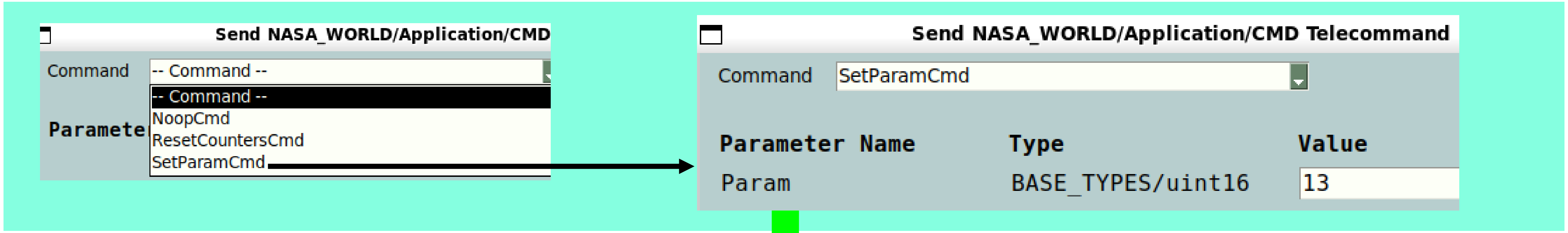
2. **Since the EDS was modified, the GUI must be restarted so the new EDS library with the new telemetry definition is used**

3. **The following slides describe how to observe the new telemetry data**

Lesson 2 – New Telemetry Operations

Verify the new code by sending the set parameter command and observing the telemetry is updated with the commanded value

Telecommand

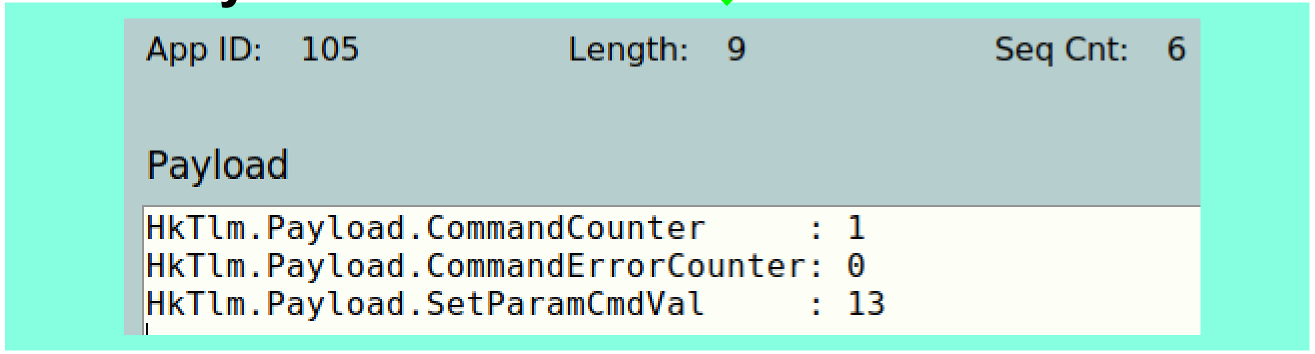


Send NASA_WORLD/Application/CMD Telecommand

Command: SetParamCmd

Parameter Name	Type	Value
Param	BASE_TYPES/uint16	13

Telemetry



App ID: 105   Length: 9   Seq Cnt: 6

Payload

```
HkTlm.Payload.CommandCounter      : 1
HkTlm.Payload.CommandErrorCounter: 0
HkTlm.Payload.SetParamCmdVal      : 13
```