

# cFS Basecamp Hello World Coding Lessons



Version 2.7  
July 2025

- These slides provide guidance for doing the Hello World coding tutorial exercises
- The “Hello App Designs” section in Basecamp’s *Application Developer’s Guide* provides design information for all of Hello App coding tutorials
  - Having all of the design information in one place makes the developer’s guide flow better
  - It should be used in conjunction with this guide
- The Hello World app template creates a minimal cFS application
  - The application design follows the Basecamp app design conventions as described in the developer’s guide
  - The coding exercises introduce developers to the different app components
- 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 and Basecamp’s Application Developer's Guide

- After generating Hello World, start the cFS target using the cFS <Start> button
  - Scroll up in the cFS Target Process Window and you should see the following two event messages that indicate the Hello World app successfully started

cFS Target Process Window    Telecommand: 127.0.0.1:1234    Telemetry: Local    Time: 1001055

```

EVS Port1 66/1/PL_SIM 100: PL_SIM App initialized. version 1.0.0
EVS Port1 66/1/PL_MGR 4: JSON initialization file successfully processed with 11 parameters
EVS Port1 66/1/PL_MGR 100: PL_MGR App Initialized. Version 1.0.0
EVS Port1 66/1/HELLO 4: JSON initialization file successfully processed with 7 parameters
EVS Port1 66/1/HELLO 100: HELLO App Initialized. Version 1.0.0
EVS Port1 66/1/KIT_SCH 300: Message Table load updated 23 entries
  
```

- Use the File Browser to download hello\_ini.json and open it in the text editor
  - Note the TOPICID values are populated when “make topicids” is executed

```

{
  "title": "Hello initialization file",
  "description": ["Define runtime configurations"],
  "config": {
    "APP_CFE_NAME": "HELLO",
    "APP_PERF_ID": 127,

    "APP_CMD_PIPE_DEPTH": 5,
    "APP_CMD_PIPE_NAME": "HELLO_CMD",

    "HELLO_CMD_TOPICID": 6247,
    "BC_SCH_4_SEC_TOPICID": 6228,
    "HELLO_HK_TLM_TOPICID": 2163
  }
}
  
```

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

The diagram illustrates the process of selecting a specific telemetry message. On the left, a 'View Tlm' window shows a list of available messages. A green arrow points to the 'HELLO/Application/STATUS\_TLM' entry at the bottom of the list. A second green arrow points from this entry to a detailed view of the message on the right.

**View Tlm: CFE\_ES/Application/HK\_TLM**

- KIT\_TO/Application/DATA\_TYPES\_TLM
- KIT\_TO/Application/PKT\_TBL\_TLM
- 1001027 KIT\_TO/Application/PLBK\_EVENT\_TLM
- KIT\_TO/Application/SUB\_WRAPPED\_TLM
- KIT\_TO/Application/PUB\_WRAPPED\_TLM
- APP\_C\_DEMO/Application/STATUS\_TLM
- APP\_C\_DEMO/Application/BIN\_PLAYBK\_TLM
- PL\_MGR/Application/STATUS\_TLM
- PL\_SIM/Application/STATUS\_TLM
- HELLO/Application/STATUS\_TLM

**HELLO/Application/STATUS\_TLM - Port 9004**

App ID: 115      Length: 9      Seq Cnt: 18

**Payload**

```
StatusTlm.Payload.ValidCmdCnt : 0
StatusTlm.Payload.InvalidCmdCnt: 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
  - BC\_SCH\_4\_SEC\_TOPICID is a parameter in the JSON init file so it can be referenced by the app

- **Issue multiple Hello World Noop commands**

- Note the *Quick Cmd* menu includes a “Noop/Reset App” entry that may be more convenient than *Send Cmd*

The screenshot shows the 'Send Cmd' dialog on the left with a list of command topics. A green arrow points from 'HELLO/Application/CMD' in the list to the 'Send HELLO/Application/CMD Telecommand' window in the center. This window has 'NoopCmd' selected in the 'Command' dropdown. A second green arrow points from the 'Payload' section of this window to a third window on the right titled 'HELLO/Applicati'. This third window shows 'App ID: 115' and 'Length: 9'. The 'Payload' section contains two lines: 'StatusTlm.Payload.ValidCmdCnt : 3' and 'StatusTlm.Payload.InvalidCmdCnt : 0'. A green arrow points to the value '3' in the first line.

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

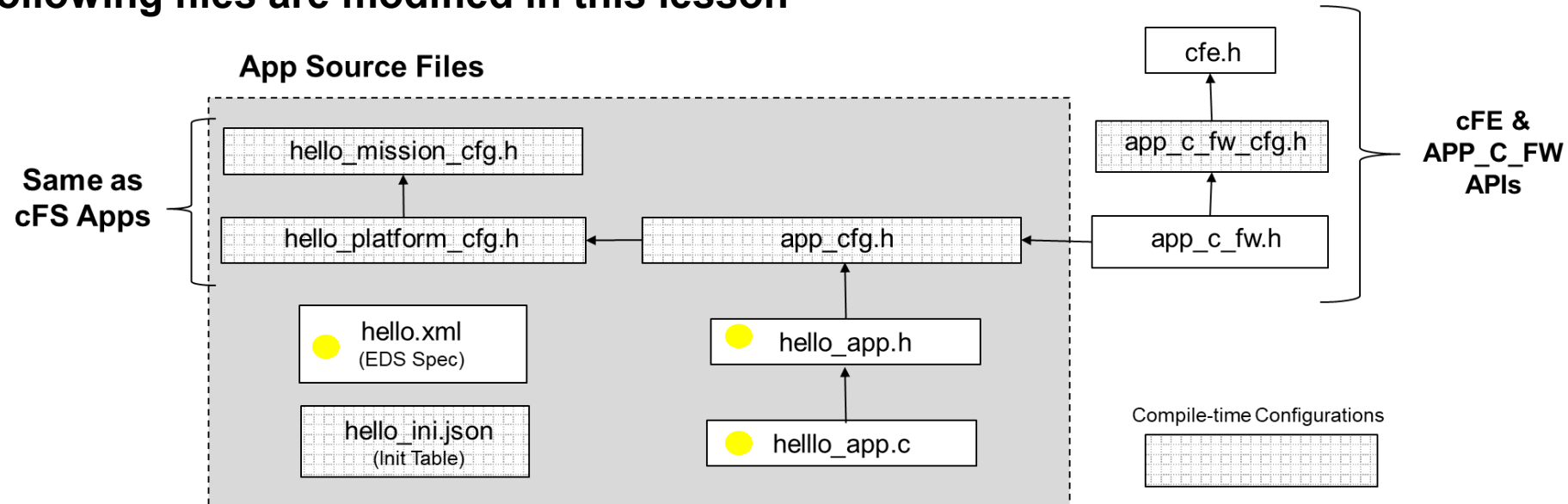
- Note Basecamp’s convention is to have the reset command clear the command counters and then the counter increments to 1 because the reset command was successful

The screenshot shows the 'HELLO/Applicati' window. The 'App ID' is 115 and the 'Length' is 9. The 'Payload' section shows 'StatusTlm.Payload.ValidCmdCnt : 1' and 'StatusTlm.Payload.InvalidCmdCnt : 0'. A green arrow points to the value '1' in the first line.

## Objectives

- Learn how to define commands using Electronic Data Sheets
- Learn how to use APP\_C\_FW's Command Manager utility for registering and dispatching commands

## ● The following files are modified in this lesson



Typically, new commands are not added to the main app file because in Basecamp's object-based design strategy new commands would be part of an object owned by the main app. It is being done here to teach the objectives while keeping the app structure simple.

## hello.xml

- The new command is defined in two parts
- The *SetParam\_CmdPayload* defines the command parameter
  - The Developer's Guide explains the naming convention
  - BASETYPES is an EDS package defined in the cFE EDS specs
- *APP\_C\_FW/APP\_BASE\_CC* is the starting Command Code for apps
  - The Noop and Reset commands are in every app so the framework defines their command codes

## hello\_app.h

- The new command requires a new event message identifier and by convention the macro names end in `_EID`
- `XXX_SetParamCmd()` definition must follow the *APP\_C\_FW* Command Manager API requirements

## hello\_app.c

- The `const *CmdPayload` definition in `XXX*CmdPayload()` is a coding idiom that makes accessing command parameters easy and consistent
- `CMDMGR_RegisterFunc()` registers the command processing function which is invoked because of the call to `CMDMGR_DispatchFunc()` in `ProcessCommands()`

- 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 definition is used**
- 3. The following slides describe how to use the new command**

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

## Telecommand

Send HELLO/Application/CMD

Command

-- Command --

-- Command --

Parameter

Noop

Reset

SetParam

Send HELLO/Application/CMD Telecommand

Command

SetParam

Parameter Name	Type	Value
Param	BASE_TYPES/uint16	13

## Telemetry

### Payload

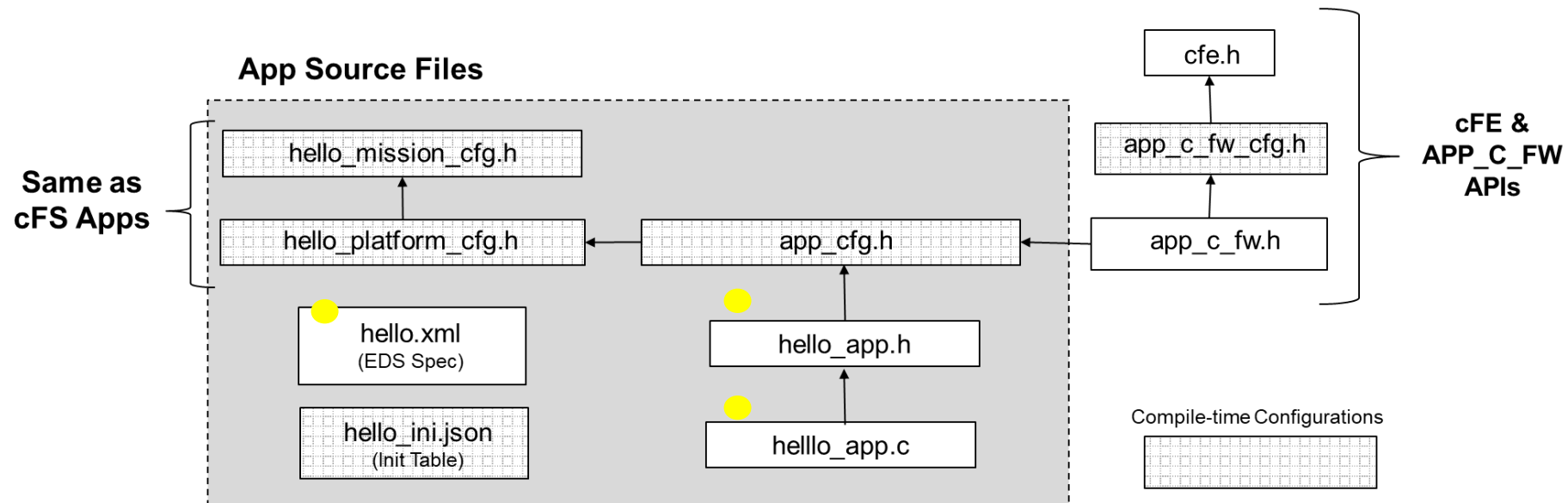
```
StatusTlm.Payload.ValidCmdCnt : 1
StatusTlm.Payload.InvalidCmdCnt: 0
```

```
EVS Port1 66/1/HELLO 104: Set Parameter command received a parameter value 13
```

## Objectives

- Learn how to define telemetry messages using Electronic Data Sheets
- Introduce how telemetry messages are managed in a Basecamp app

- The following files are modified in this lesson



## hello.xml

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

## hello\_app.h

- A new variable needs to be added to save the command parameter so it can be sent in telemetry
- This variable is added to the hello world app's object data
- As mentioned in lesson one, this new command would typically be part of an object owned by the main app and it is being done here to keep the exercise simple

## hello\_app.c

- The *XXX\_SetParamCmd()* function needs to be modified to save the parameter value
- The *SendStatusTlm()* needs to be modified to copy the command parameter into the telemetry packet

- 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**

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

## Telecommand

Send HELLO/Application/CMD

Command

-- Command --

Parameter

Noop

Reset

SetParam

Send HELLO/Application/CMD Telecommand

Command

SetParam

Parameter Name	Type	Value
Param	BASE_TYPES/uint16	13

## Telemetry

HELLO/Application/STATUS\_TLM - Port 9003

App ID: 115

Length: 11

Seq Cnt: 9

Payload

StatusTlm.Payload.ValidCmdCnt : 1

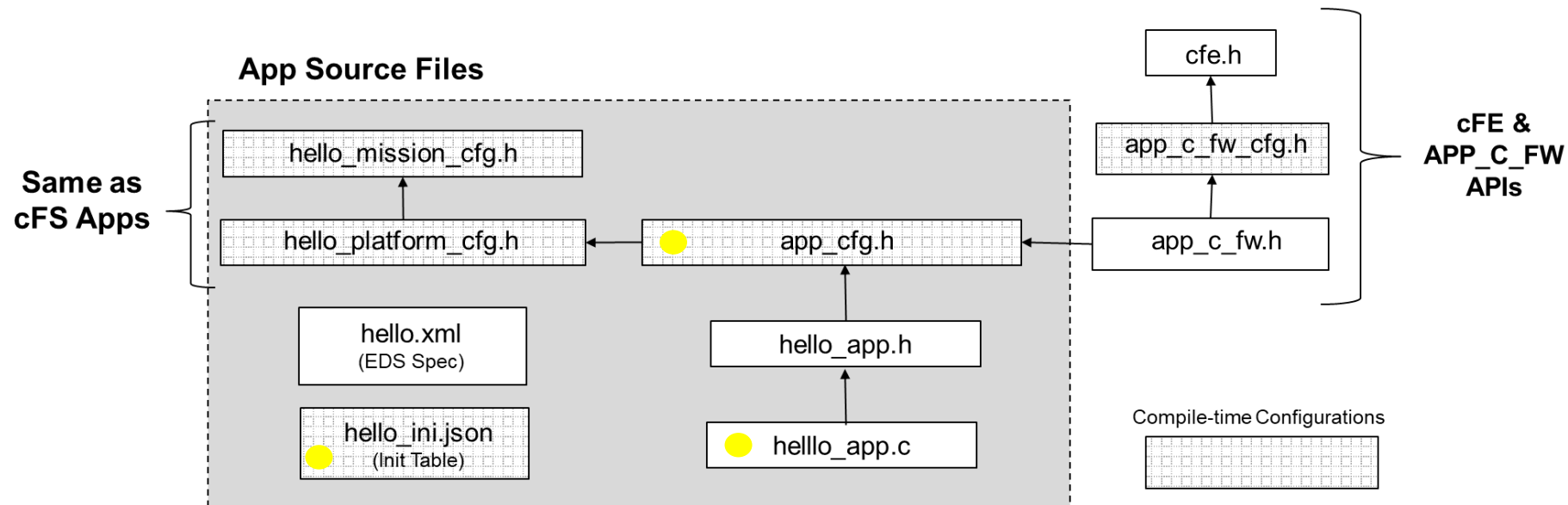
StatusTlm.Payload.InvalidCmdCnt : 0

StatusTlm.Payload.SetParamCmdVal : 13

## Objectives

- Introduce Basecamp's app parameter initialization JSON files
- Learn how init parameters are defined and accessed

- The following files are modified in this lesson



## cpu1\_hello\_wrld.ini

- Changing the APP\_CMD\_PIPE\_DEPTH parameter name does not effect the how the parameter is used
- The value of 7 is chosen because it is not used by any other apps and will be used as part of the verification
- During the target build process this file is copied from basecamp\_defs to the build/exe/cpu1/cf and the 'cpu1\_' prefix is removed from the filename

## app\_cfg.h

- This JSON initialization file parameter names are defined in this file, block comment has detailed instructions
- Initialization parameters can either by of type string or integer

## hello\_app.c

- The initialization parameters values are retrieved using either the INITBL\_GetIntConfig() or INITBL\_GetStrConfig() functions
- APP\_CMD\_PIPE\_MAX is used in the CFE\_SB\_CreatePipe() call and defines the maximum number of packets that can be on the pipe

- 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 use the new telemetry data point**

1. The Software Bus Stats telemetry message will be used to indirectly verify the initialization parameter
2. Open SB's STATS\_TLM using the "View Tlm" drop down menu
3. Issue a CFE\_SB *SendSbStatsCmd* which causes SB to send a single STATS\_TLM message.
4. Scroll down until you find an app with a MaxQueueDepth of 7 which implies this is the Hello World app.
5. You can change the parameter value again and follow these steps to verify you find an app with the expected value.

## Telecommand

Send Cmd: -- Command Topic --

0.0.1:1234

ced tabl

d. Versi

TT clien

TT clien

TT clien

for top

-- Command Topic --

CFE\_ES/Application/CMD

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

Send CFE\_SB/Application/CMD Telecommand

Command **SendSbStatsCmd**

Parameter Name	Type	Value
No Parameters		

## Telemetry

View Tlm: -- Telemetry Topic --

1001019

\_to\_pkt

emo/sc/

-- Telemetry Topic --

CFE\_ES/Application/HK\_TLM

CFE\_ES/Application/APP\_TLM

CFE\_ES/Application/MEMSTATS\_TLM

CFE\_EVS/Application/HK\_TLM

CFE\_EVS/Application/SHORT\_EVENT\_MSG

CFE\_EVS/Application/LONG\_EVENT\_MSG

CFE\_SB/Application/HK\_TLM

CFE\_SB/Application/STATS\_TLM

CFE\_SB/Application/ALLSUBS\_TLM

```
StatsTlm.Payload.PipeDepthStats[6].PipeId.BaseType : 34996231
StatsTlm.Payload.PipeDepthStats[6].MaxQueueDepth : 7
StatsTlm.Payload.PipeDepthStats[6].CurrentQueueDepth : 0
StatsTlm.Payload.PipeDepthStats[6].PeakQueueDepth : 1
```