Open
STEM ware

cFS Basecamp
Hello World Coding Lessons

Basecamp

Version 2.7

July 2025

Introduction

- **These slides provide guidance for doing the Hello Object 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

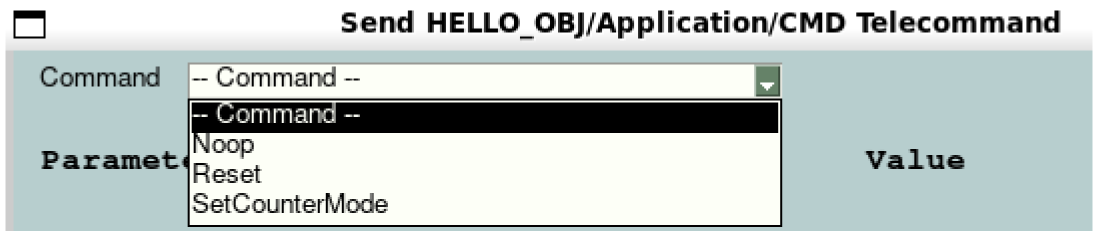
- **The Hello Object app template adds an example object to the Hello World application**
  - The coding exercises introduce developer's to the Basecamp's object-based design strategy

- **Prerequisites**
  - Completed Hello World coding tutorial and met its prerequisites

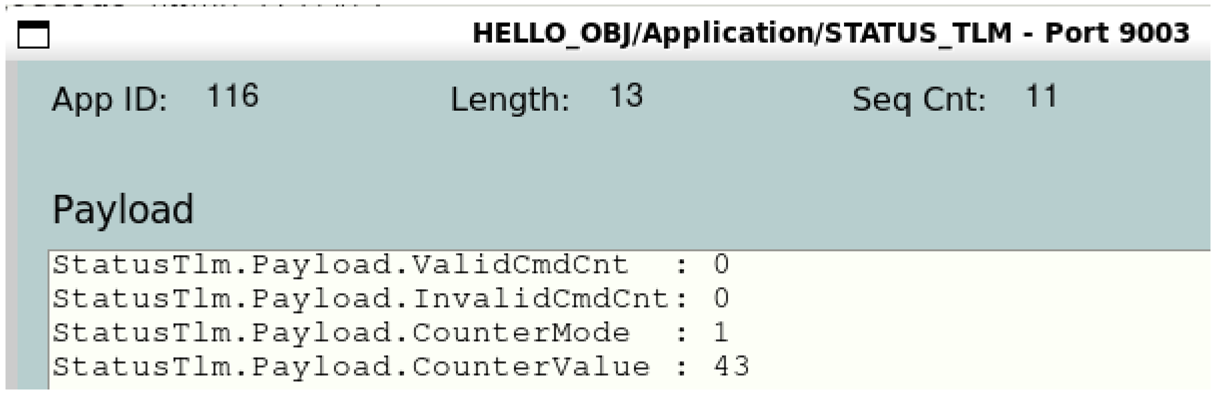
Hello Object Operations

Commands

- Retain Hello World's Noop and Reset commands
- New Set Counter Mode command



Send HELLO_OBJ/Application/CMD Telecommand



HELLO_OBJ/Application/STATUS_TLM - Port 9003

App ID: 116 Length: 13 Seq Cnt: 11

Payload

```
StatusTlm.Payload.ValidCmdCnt   : 0
StatusTlm.Payload.InvalidCmdCnt: 0
StatusTlm.Payload.CounterMode   : 1
StatusTlm.Payload.CounterValue  : 43
```

Telemetry

- Retain Hello World command valid/invalid counters
- New Counter Mode data point
  - A code exercise changes the EDS definition so this will be a descriptive string
- New Counter Value data point
  - The counter updated at 1Hz and the status telemetry is sent every 4 seconds

Open STEM ware

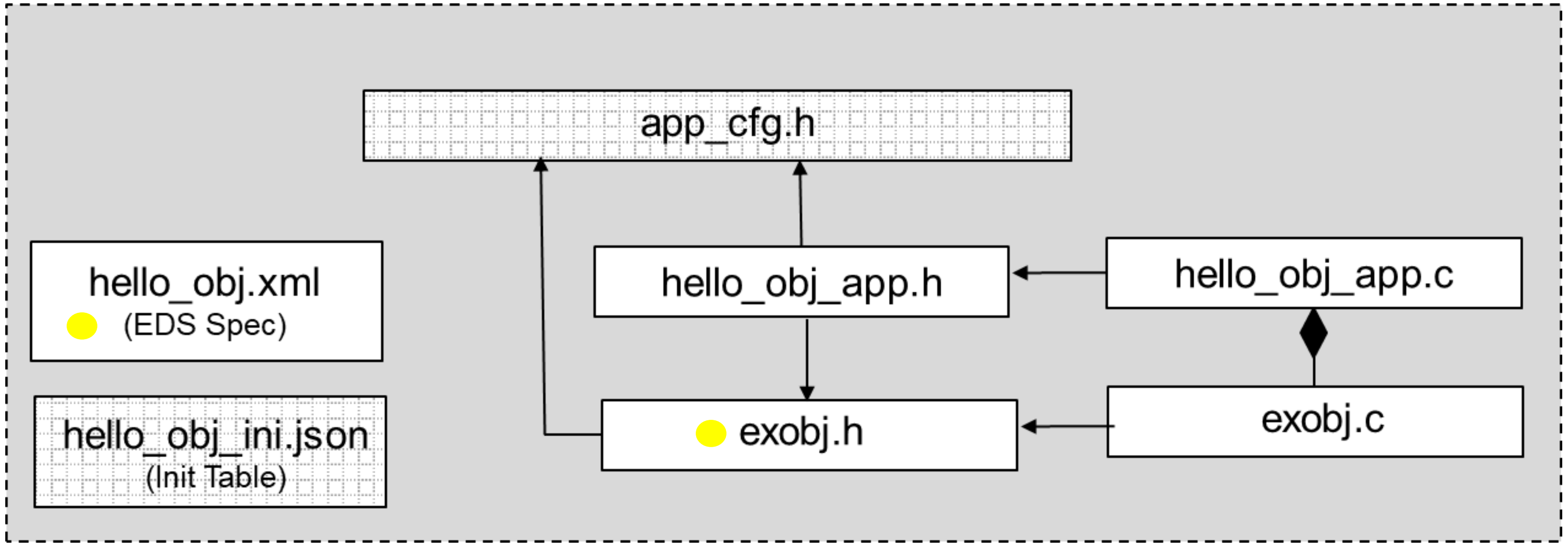
Lesson 1 – Add EDS Enum Type to Status Telemetry (1 of 2)

Basecamp

Objectives

- Increase knowledge of Electronic Data Sheets and how Basecamp apps use the code generated from the EDS

● The following files are modified in this lesson



Lesson 1 – Add EDS Enum Type to Status Telemetry (2 of 2)

hello_obj.xml

- CounterMode enumeration definition

```
<EnumeratedDataType name="CounterMode" shortDescription="" >
  <IntegerDataEncoding sizeInBits="16" encoding="unsigned" />
  <EnumerationList>
    <Enumeration label="Increment" value="1" shortDescription
    <Enumeration label="Decrement" value="2" shortDescription
  </EnumerationList>
</EnumeratedDataType>
```

Zero is avoided unless the enumeration is used as an index into array

- The EDS toolchain created a file named hello_obj_eds_typedefs.h
- The Basecamp app convention is to include the EDS typedefs in app_cfg.h
  - Basecamp app designs are object-based, however EDS specs define app level interfaces that may include definitions from multiple objects so adhering to type definitions encapsulated within an object is not always achievable

exobj.h

- The EDS toolchain creates the following variable for the enumeration definition: *HELLO_OBJ_CounterMode_Enum_t*
  - *HELLO_OBJ* is the package name
  - *CounterMode* is the enumerated data type name
  - *Enum_t* is a naming convention used by the EDS toolchain

Lesson 1 – Build New cFS Target

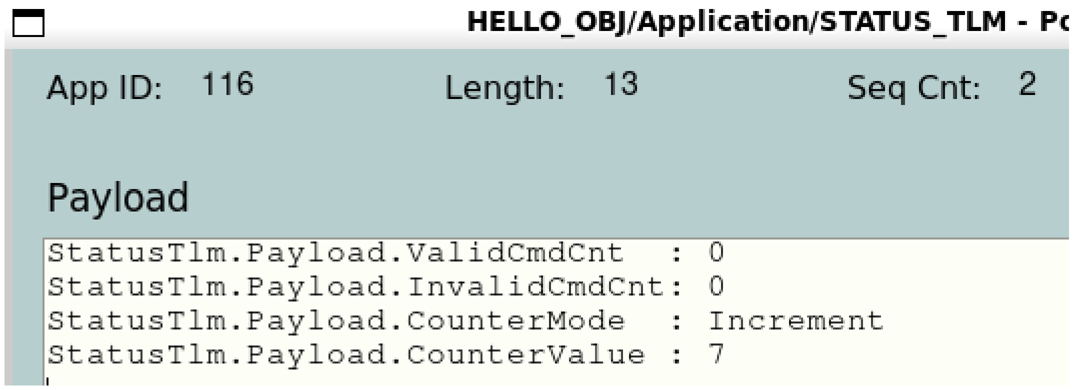
1. **Use the main screen's cFS Build button to build the target**
   - Only existing files changed, so no need to perform a Build New

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 verify the new enum type in telemetry**

Lesson 1 – Verify New EDS Enum Type in Status Telemetry

1. **After starting the cFS open the HELLO_OBJ status telemetry page**
   – The default mode should be set to Increment



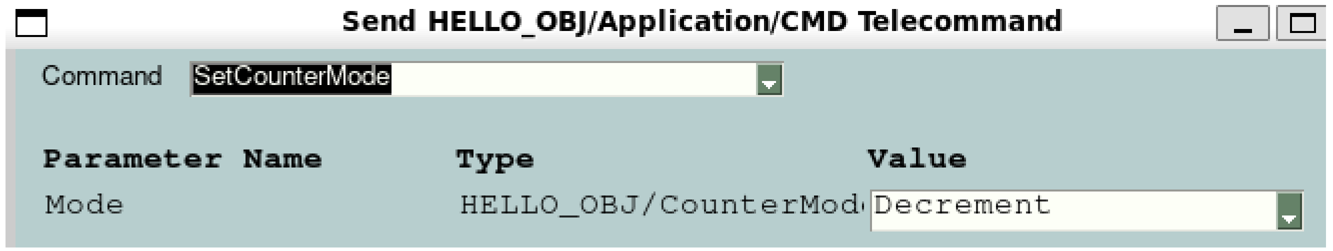
HELLO_OBJ/Application/STATUS_TLM - Pc

App ID: 116 Length: 13 Seq Cnt: 2

Payload

```
StatusTlm.Payload.ValidCmdCnt   : 0
StatusTlm.Payload.InvalidCmdCnt: 0
StatusTlm.Payload.CounterMode   : Increment
StatusTlm.Payload.CounterValue : 7
```

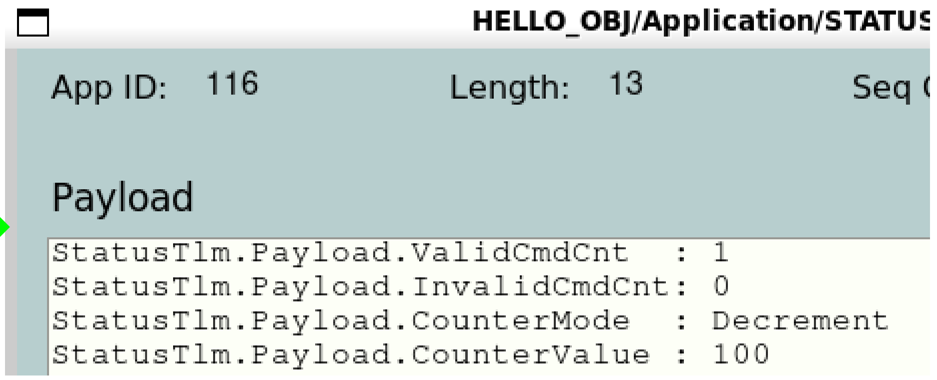
2. **Issue a SetCounterMode decrement command**



Send HELLO_OBJ/Application/CMD Telecommand

Command: SetCounterMode

Parameter Name	Type	Value
Mode	HELLO_OBJ/CounterMod	Decrement



HELLO_OBJ/Application/STATUS

App ID: 116 Length: 13 Seq (

Payload

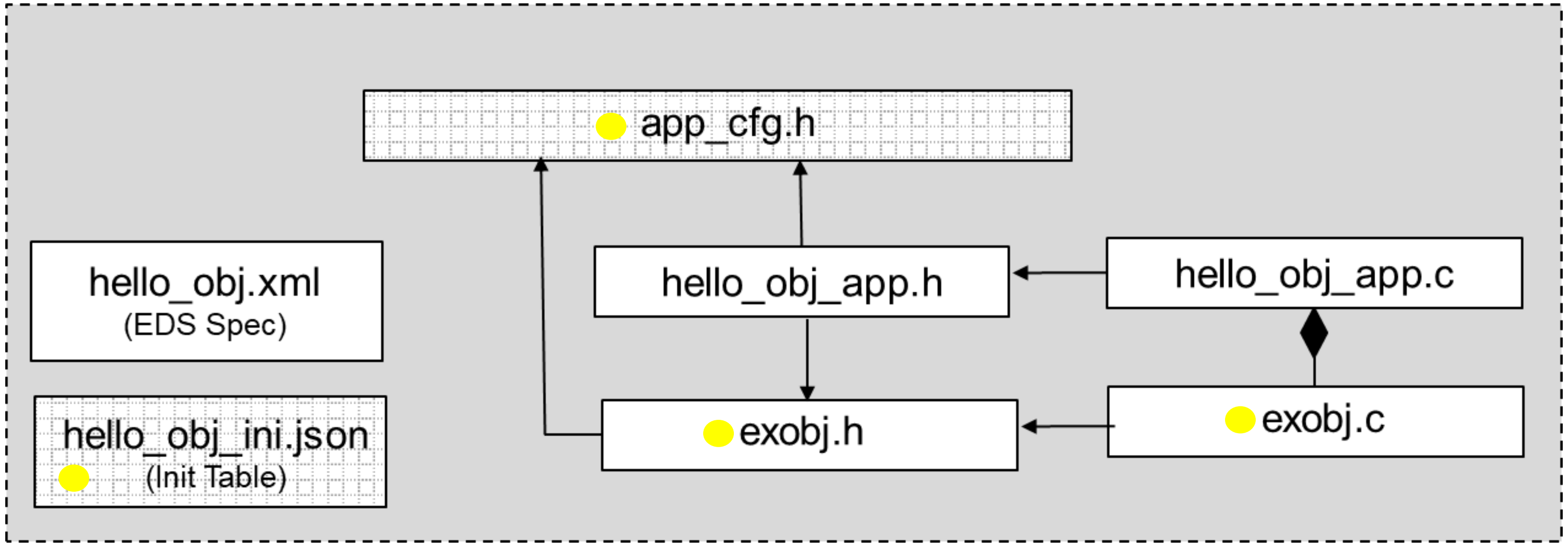
```
StatusTlm.Payload.ValidCmdCnt   : 1
StatusTlm.Payload.InvalidCmdCnt: 0
StatusTlm.Payload.CounterMode   : Decrement
StatusTlm.Payload.CounterValue : 100
```

Lesson 2 – Move Limits to JSON Init File (1 of 2)

Objectives

- Learn app_cfg.h architectural role
- Learn object-based strategy and role of encapsulation
- Build upon Hello World JSON init table lesson and learn init file app level management and relationship to objects

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



Lesson 2 – Move Limits to JSON Init File (2 of 2)

cpu1_hello_obj_ini.json

- Defining limits in the init file means the limits can be changed between app restarts without the need for recompilation
- If you need the ability to change limit changes during runtime without restarting the app then either a command is needed or the limits can be defined in a table

app_cfg.h

- Add the limit parameter declarations

exobj.h

- Remove the hardcoded limits and add class variables to store the limit values

exobj.c

- Initialize the limits in the example object's constructor.
- If an init parameter is needed once and remains unchanged during runtime then it is used in the app's main init function or an object's constructor and it doesn't need to be stored in a variable. Otherwise it should be stored in a variable.
- Because of the example app's encapsulation, the main app files did not need modification for this lesson

Lesson 2 – Build New cFS Target

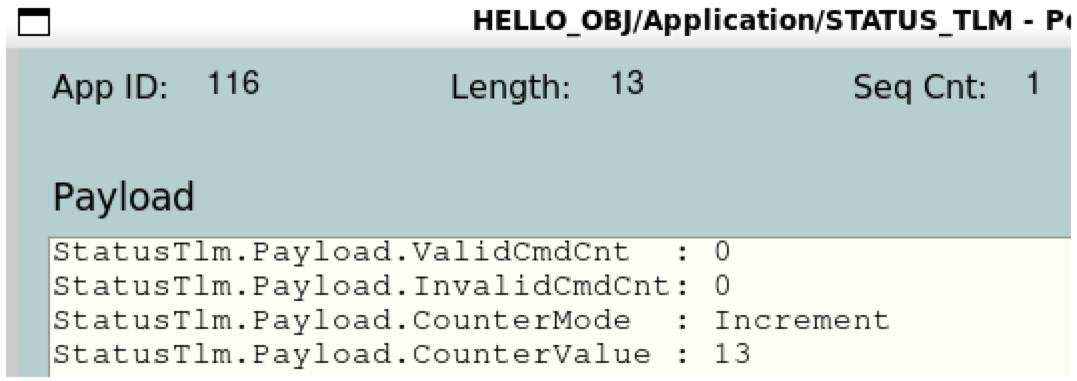
1. **Use the main screen's cFS Build button to build the target**
   – Only existing files changed, so no need to perform a Build New

2. **Since the EDS was not modified, the GUI does not need to be restarted**

3. **The following slides describe how to use the limits in the init JSON file**

Lesson 2 – Verify Limits in JSON Init File

1. **After the app starts observe new range taking effect**
   - Here is the first packet after HELLO_OBJ starts (sequence count equals1) and the *Counter Value* is 13 which is slightly more than the low limit of 10
   - This is reasonable because the counter is incremented at 1Hz and the status telemetry is sent every 4 seconds



```
StatusTlm.Payload.ValidCmdCnt   : 0
StatusTlm.Payload.InvalidCmdCnt: 0
StatusTlm.Payload.CounterMode   : Increment
StatusTlm.Payload.CounterValue  : 13
```

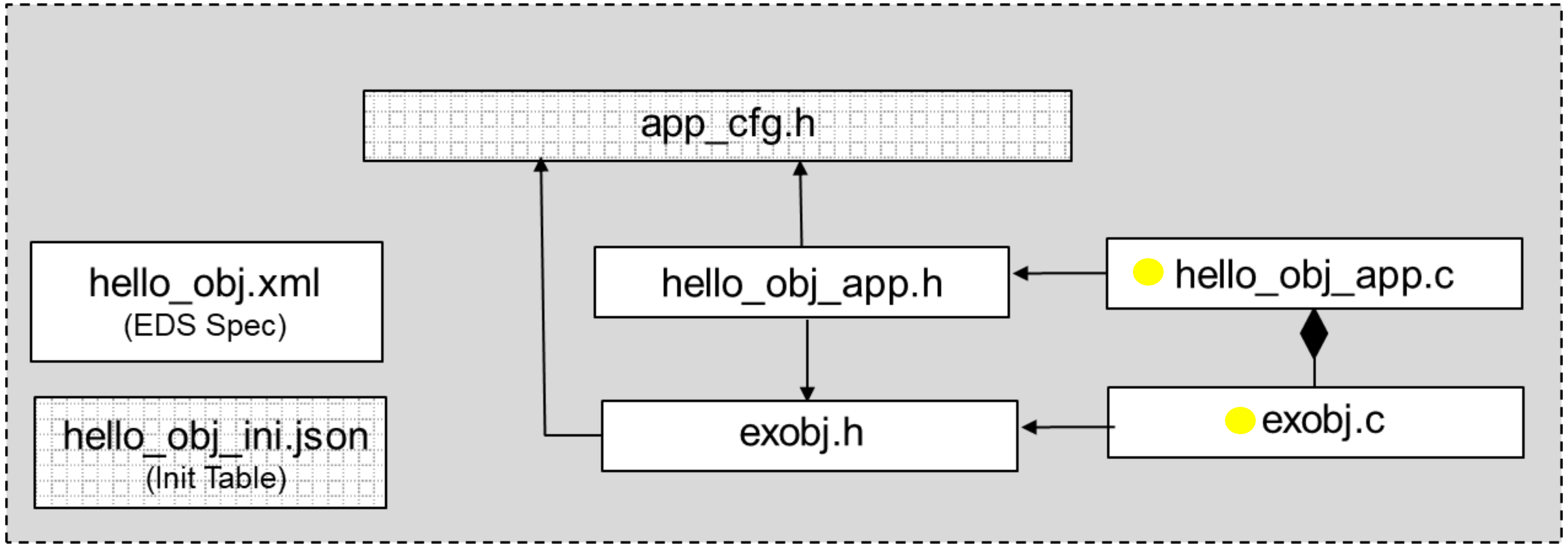
2. **Change HELLO_OBJ's init table limits and restart the app to observe the different limits taking effect**
   - See the Demo App lesson in the Basecamp Introduction tutorial for guidance on how to perform these steps

Lesson 3 – Add Event Message Filter (1 of 2)

Objectives

- Introduce event types and event filters
- Use an object with a periodic function that is triggered by a Scheduler (SCH) app message
- Deeper dive into app architecture, resources, roles and responsibilities

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



Lesson 3 – Add Event Message Filter (2 of 2)

hello_obj.c

- The event message filter array is owned by the app's main object
- The event message IDs are defined by the objects owned by the app
- The cFE Event Service defines several macros such as CFE_EVS_FIRST_8_STOP which is used in this lesson
- It's up to the user whether to add the call to CFE_EVS_ResetAllFilters() as part of the app's reset command. The event services provides a commands to reset individual app event filters

exobj.c

- The Execute()'s debug event message is changed to an INFORMATION type which causes it to be continuously sent until the filter limit is reached
- In general a type of DEBUG makes sense for this kind of event message and it was changed in this lesson to demonstrate event message filters and to demonstrate how event types can be used

Lesson 3 – 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

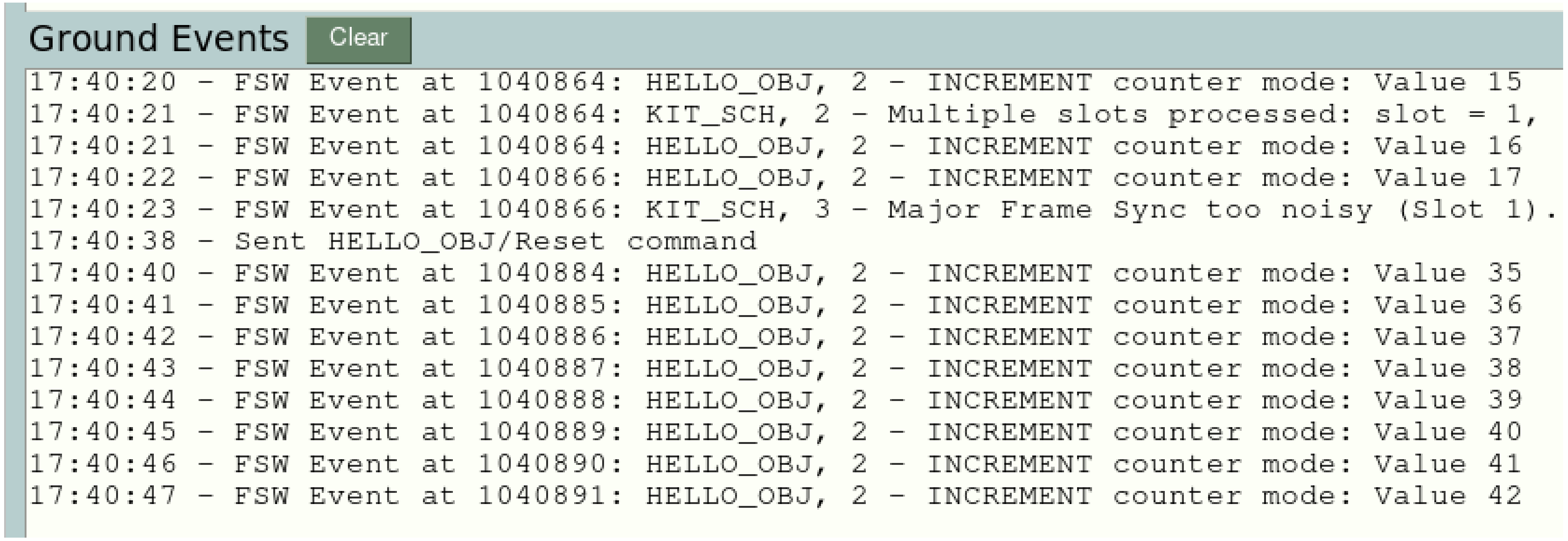
2. **Since the EDS was not modified, the GUI does not need to be restarted**

3. **The following slides describe how to verify the event message filter**

Lesson 3 – Verify Event Message Filter

1. **Observe 8 INFORMATION event messages are sent when the app first starts running and then the filter limit is reached so the messages are no longer output**

2. **When you issue an app reset command you should see 8 more messages get sent because the event filter is reset**



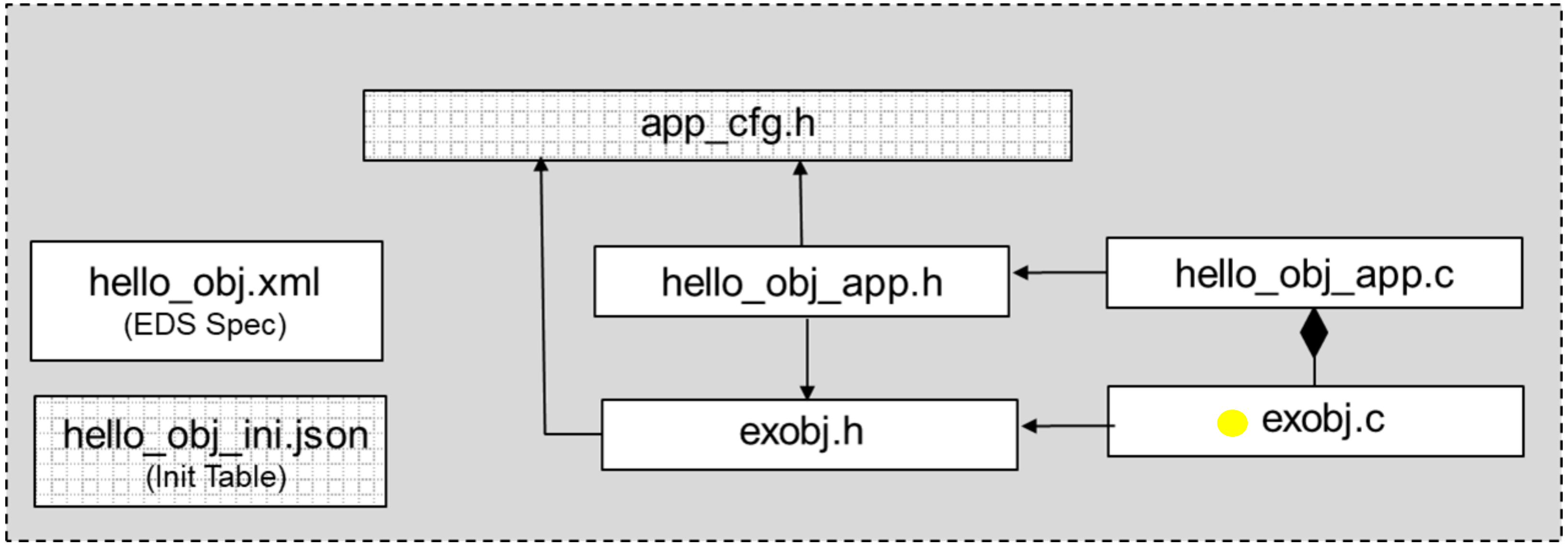
```
17:40:20 - FSW Event at 1040864: HELLO_OBJ, 2 - INCREMENT counter mode: Value 15
17:40:21 - FSW Event at 1040864: KIT_SCH, 2 - Multiple slots processed: slot = 1,
17:40:21 - FSW Event at 1040864: HELLO_OBJ, 2 - INCREMENT counter mode: Value 16
17:40:22 - FSW Event at 1040866: HELLO_OBJ, 2 - INCREMENT counter mode: Value 17
17:40:23 - FSW Event at 1040866: KIT_SCH, 3 - Major Frame Sync too noisy (Slot 1).
17:40:38 - Sent HELLO_OBJ/Reset command
17:40:40 - FSW Event at 1040884: HELLO_OBJ, 2 - INCREMENT counter mode: Value 35
17:40:41 - FSW Event at 1040885: HELLO_OBJ, 2 - INCREMENT counter mode: Value 36
17:40:42 - FSW Event at 1040886: HELLO_OBJ, 2 - INCREMENT counter mode: Value 37
17:40:43 - FSW Event at 1040887: HELLO_OBJ, 2 - INCREMENT counter mode: Value 38
17:40:44 - FSW Event at 1040888: HELLO_OBJ, 2 - INCREMENT counter mode: Value 39
17:40:45 - FSW Event at 1040889: HELLO_OBJ, 2 - INCREMENT counter mode: Value 40
17:40:46 - FSW Event at 1040890: HELLO_OBJ, 2 - INCREMENT counter mode: Value 41
17:40:47 - FSW Event at 1040891: HELLO_OBJ, 2 - INCREMENT counter mode: Value 42
```

Lesson 4 – Add Object Reset Functionality (1 of 2)

Objectives

- Show how objects can have functional requirements for an App's Reset command
- Demonstrate how event message filters can be useful for debugging periodic events

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



Lesson 4 – Add Object Reset Functionality (2 of 2)

exobj.c

- The Basecamp interpretation of the “standard” Reset app command functionality is broader than the cFS convention
- The cFS convention is to focus on counters and this grew out of the need to reset the valid/invalid command counters
- Basecamp apps reset state and it’s up to the mission/developer as to what makes sense
- Be careful not to make it reset too much when it would be beneficial to have individual commands to reset specific functionality

- EXOBJ_Execute() runs at 1Hz so having the debug event filtered allows the developer to enable debug events to prove the function is running as expected and the events stop so telemetry is not cluttered
- Ultimately the event strategy should match the debugging scenario needs

Lesson 4 – Build New cFS Target

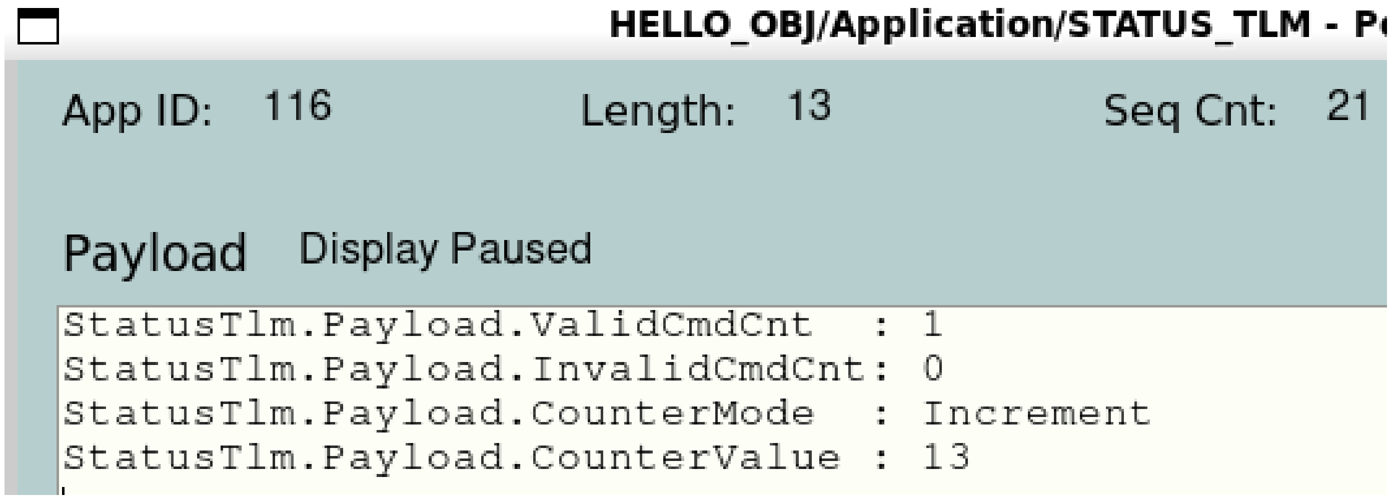
1. **Use the main screen's cFS Build button to build the target**
   – Only existing files changed, so no need to perform a Build New

2. **Since the EDS was not modified, the GUI does not need to be restarted**

3. **The following slides describe how to verify the object reset functionality**

Lesson 4 – Verify Object Reset Functionality (1 of 2)

1. **Reset the app and observe the counter go to the low limit**
2. **Change the counter mode to decrement, wait and then reset to show it is reset to the high limit (not shown)**



HELLO_OBJ/Application/STATUS_TLM - P

App ID: 116　　Length: 13　　Seq Cnt: 21

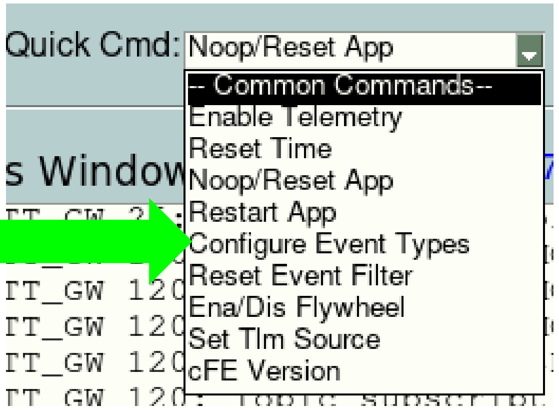
Payload　Display Paused

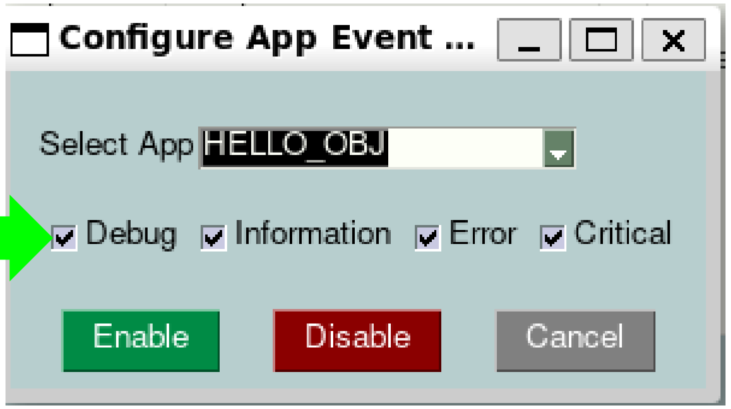
```
StatusTlm.Payload.ValidCmdCnt   : 1
StatusTlm.Payload.InvalidCmdCnt: 0
StatusTlm.Payload.CounterMode   : Increment
StatusTlm.Payload.CounterValue : 13
```

- First telemetry message after reset command
- Display paused
- Counter value of 13 makes sense because near the low limit of 10 and telemetry is output every 4 seconds

Lesson 4 – Verify Object Reset Functionality (2 of 2)

3. **Enable debug messages and see the execute debug events**
4. **Reset app and show the filter is reset and applies to any event types (not shown)**



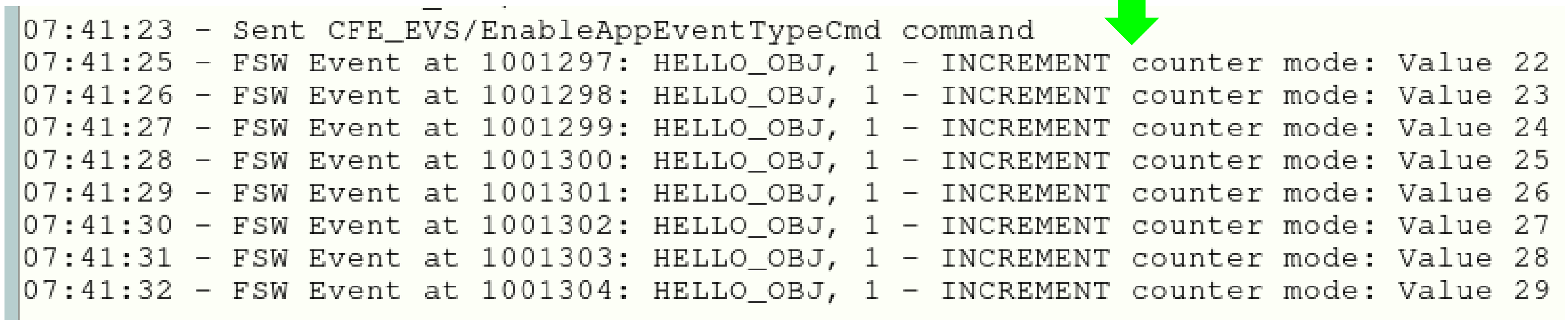


Select App HELLO_OBJ

[✓] Debug [✓] Information [✓] Error [✓] Critical

Use Configure Event command to enable HELLO_OBJ's debug events

First 8 debug events are output



```
07:41:23 - Sent CFE_EVS/EnableAppEventTypeCmd command
07:41:25 - FSW Event at 1001297: HELLO_OBJ, 1 - INCREMENT counter mode: Value 22
07:41:26 - FSW Event at 1001298: HELLO_OBJ, 1 - INCREMENT counter mode: Value 23
07:41:27 - FSW Event at 1001299: HELLO_OBJ, 1 - INCREMENT counter mode: Value 24
07:41:28 - FSW Event at 1001300: HELLO_OBJ, 1 - INCREMENT counter mode: Value 25
07:41:29 - FSW Event at 1001301: HELLO_OBJ, 1 - INCREMENT counter mode: Value 26
07:41:30 - FSW Event at 1001302: HELLO_OBJ, 1 - INCREMENT counter mode: Value 27
07:41:31 - FSW Event at 1001303: HELLO_OBJ, 1 - INCREMENT counter mode: Value 28
07:41:32 - FSW Event at 1001304: HELLO_OBJ, 1 - INCREMENT counter mode: Value 29
```