# cFS Basecamp
# Hello World Coding Lessons


**Version 1.9**

**October 2023**

# Introduction

- **Objectives**
  - Provide documentation for the Hello World coding tutorials
  - Basecamp
- **Intended Audience**
  - Software developers that want to develop cFS applications
- **Prerequisites**
  - Basic understanding of flight software context, the cFS architecture, and the cFS Application Developer's Guide
  - Working knowledge of the C programming language


STEM
Open ware
cFS
Hello Object Functionality and Operations

# Hello Object Functionality

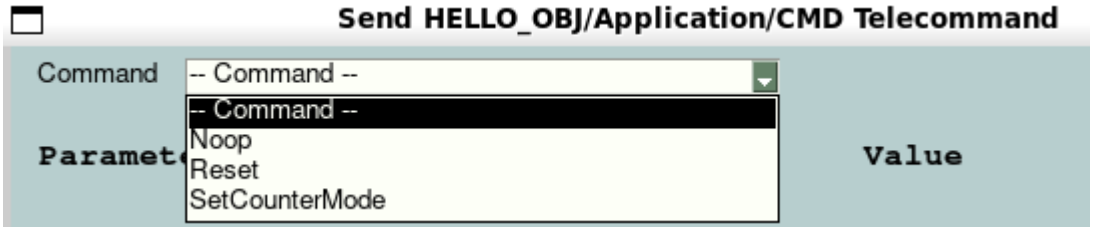
- **The Hello Obect app adds an example object to the Hello World app**
  - The Hello World coding exercise additions are not part of the Hello World app baseline

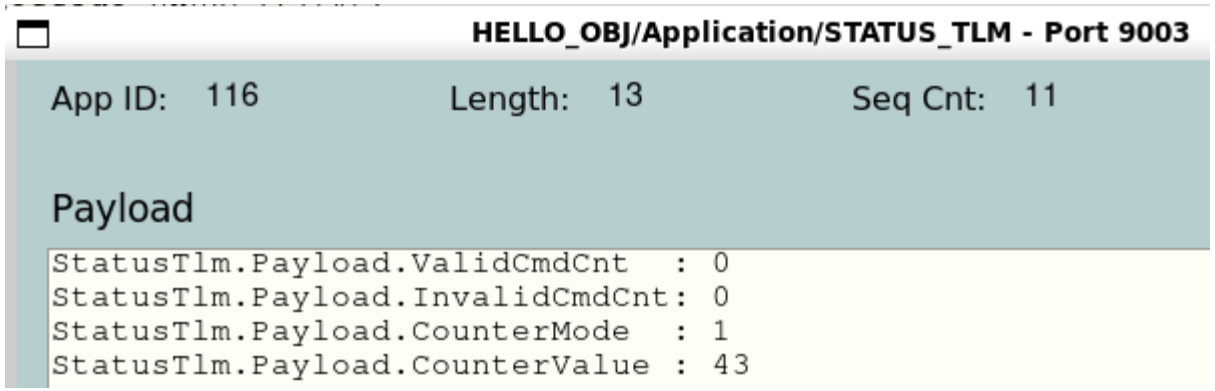
- **The example obect performs the following functions**
  - Provides an up/down counter that can either be in an increment or decrement mode
  - Provides a command to set the counter mode
  - Defines lower and upper counter limits
  - The counters ‘wrap around’ using the limits
    - In increment mode when the upper limit is reached the counter value is set to the lower limit
    - In decrement mode when the lower limit is reached the counter value is set to the upper limit
  - The counter runs at 1Hz
  - The counter defaults to increment mode starting at the low limit
  - The current counter value and counter mode are in the status telemetry message

# Hello Object Operations

## Commands

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



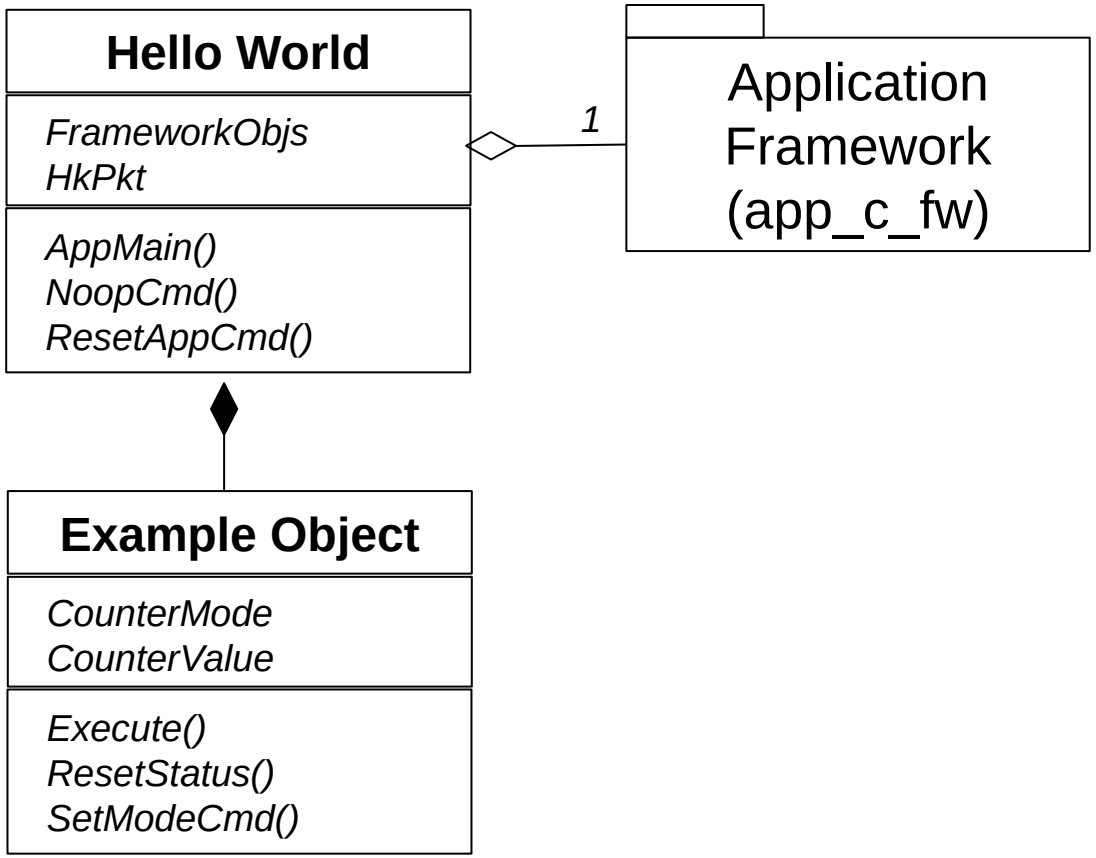




## Telemetry

- 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

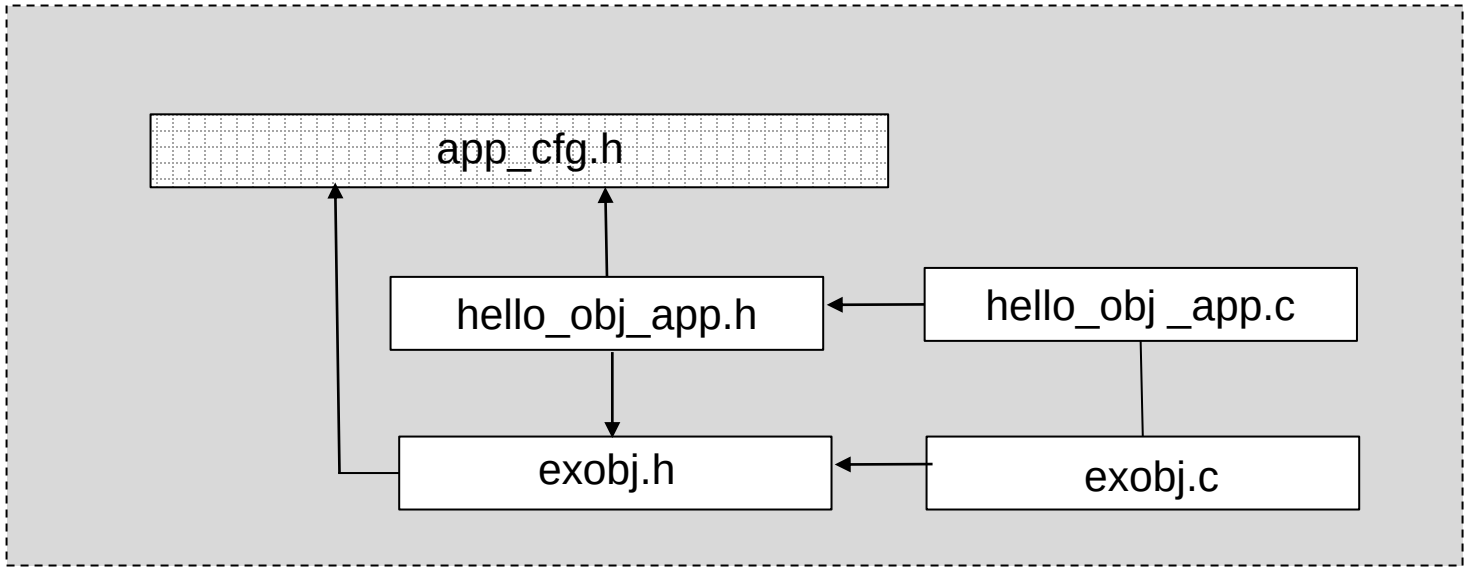

STEM
Open ware
cFS
Hello Object Design

# Hello Object Design



# Hello Object Source Files

## App Source Files




- app_cfg.h has additional ‘standard’ includes that are not shown, see App Dev Guide for details
- Hello_obj includes exobj.h so it can declare an instance of EXOBJ in its class data

```
typedef struct
{

    /*
    ** App Framework
    */

    INITBL_Class_t  IniTbl;
    CMDMGR_Class_t  CmdMgr;

    /*
    ** Telemetry Packets
    */

    HELLO_OBJ_StatusTlm_t  StatusTlm;

    /*
    ** HELLO_OBJ State & Contained Objects
    */

    uint32            PerfId;
    CFE_SB_PipeId_t   CmdPipe;
    CFE_SB_MsgId_t    CmdMid;
    CFE_SB_MsgId_t    ExecuteMid;
    CFE_SB_MsgId_t    SendStatusMid;

    EXOBJ_Class_t  ExObj;

} HELLO_OBJ_Class_t;
```

# Application Run Loop Messaging Example

**Suspend execution until a message arrives on app's pipe**

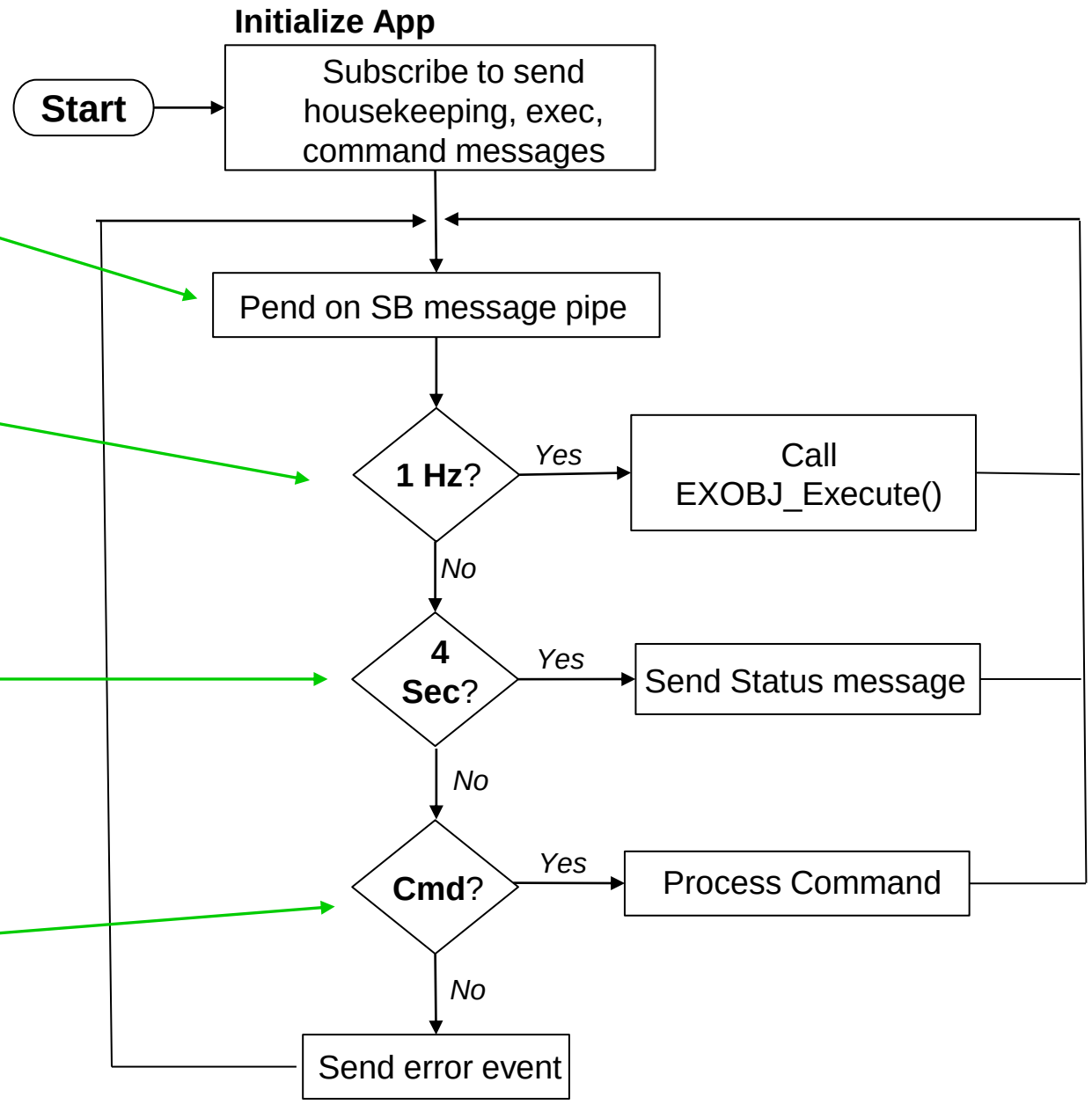
**Periodic *1Hz* message from SCH app**

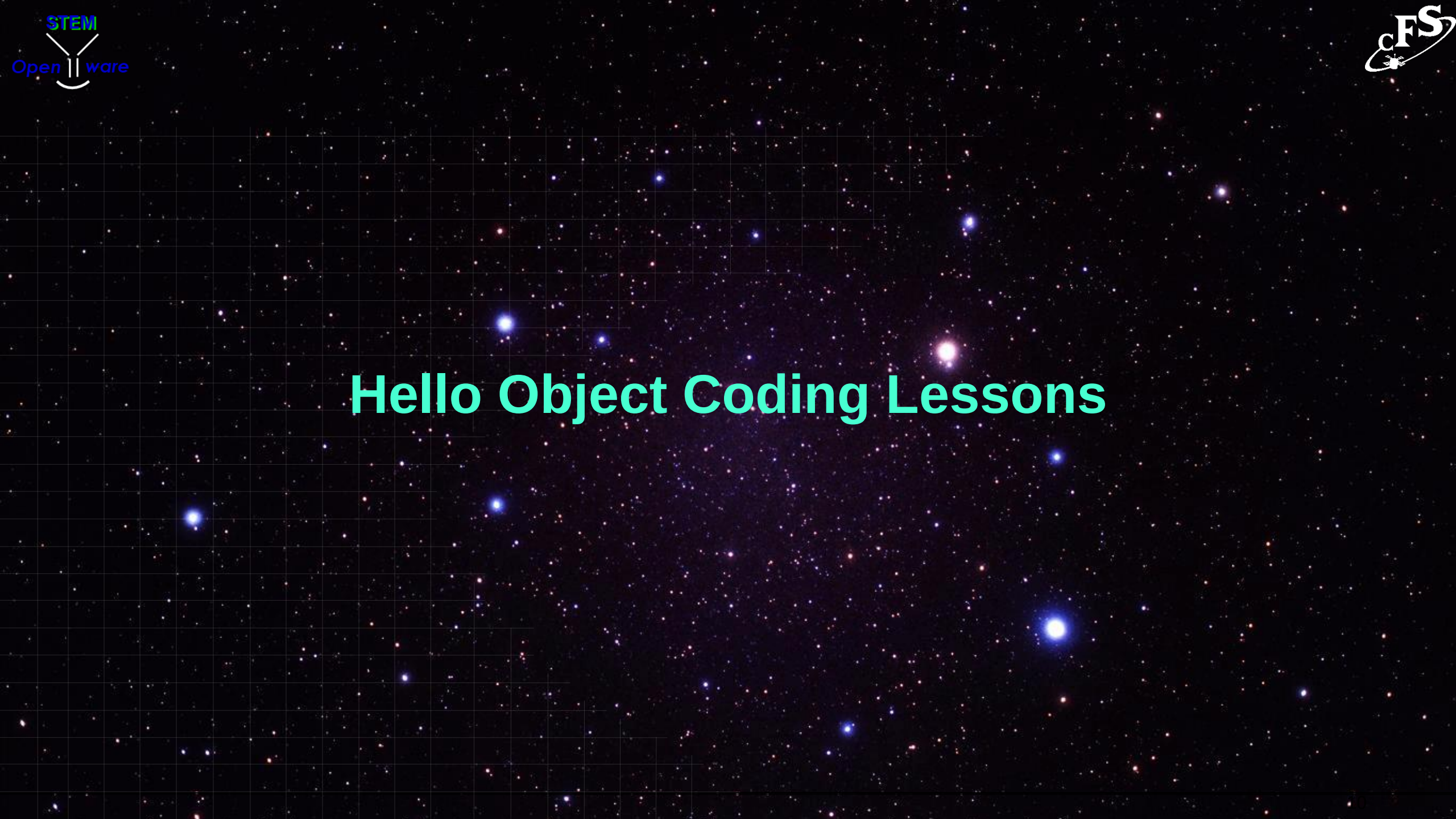
**Periodic *4 second* message from SCH app**

- Send status telemetry message
- "Housekeeping cycle" convenient time to perform non-critical functions

**Process commands**

- Commands can originate from ground or other onboard apps




STEM
Open ware
cFS
Hello Object Coding Lessons

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

## Objectives

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

## Design/Code

- CounterMode enumeration definition

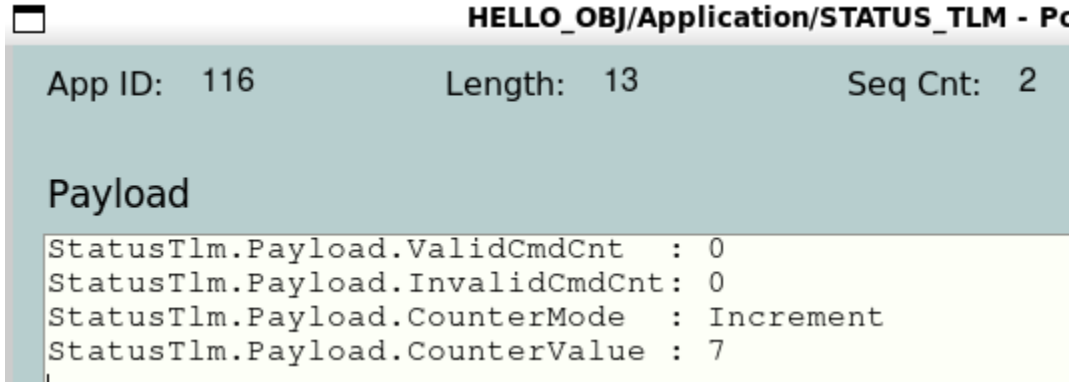
```
<EnumeratedDataType name="CounterMode" shortDescription="" >
  <IntegerDataEncoding sizeInBits="16" encoding="unsigned" />
  <EnumerationList>
    <Enumeration label="Increment" value="1" shortDescription
    <Enumeration label="Decrement" value="2" shortDescription
  </EnumerationList>
</EnumeratedDataType>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

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

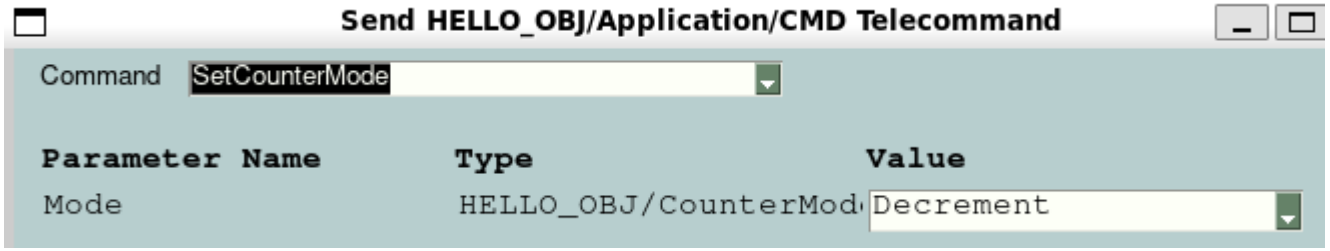
- The EDS toolchain created a file named hello_obj_eds_typedefs.h
- The Basecamp app convention is to include the EDS typedefs in app_cfg.h
  - Basecamp app designs are object-based, however EDS specs define app level interfaces that may include definitions from multiple objects so adhering to type definitions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

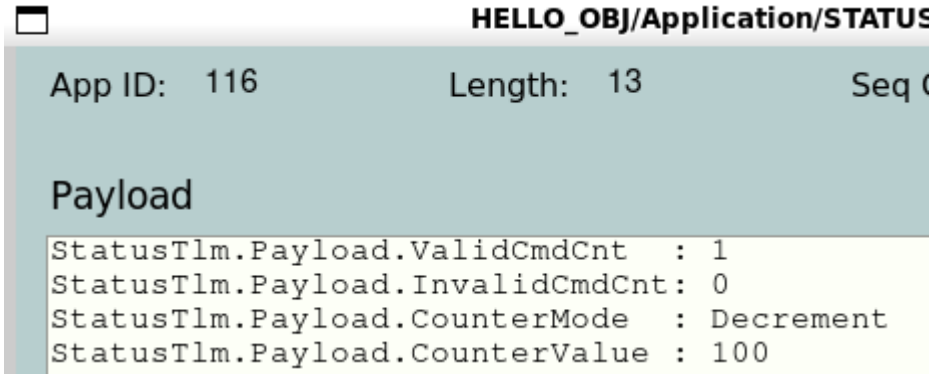
## Verification

- The Python GUI must be restarted after the new cFS Target is built so the enumerated type will be used in the telemetry message
- After starting the cFS open the HELLO_OBJ status telemetry page
  - The default mode should be Increment




- Issue a SetCounterMode decrement command



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

## Objectives

- Reinforce Hello World JSON init
- App_cfg.h architectural role
- Object based encapsulation
- Ini file app level management and relationship to objects

## Design

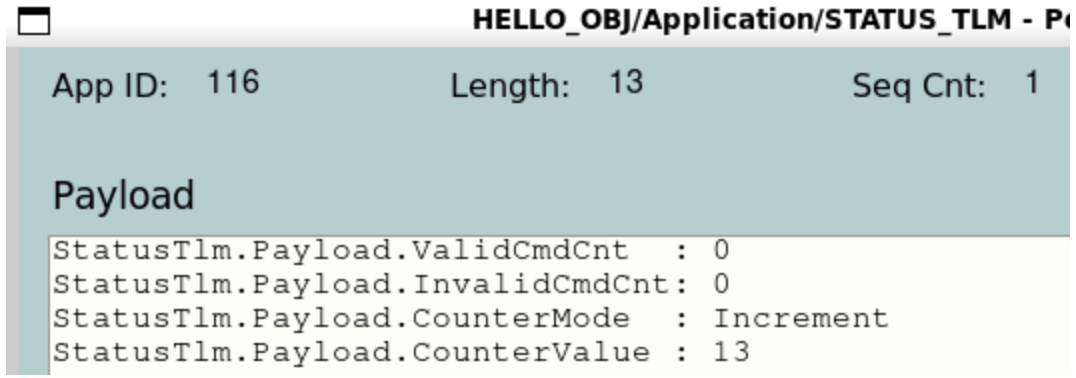
- **TBD**

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



## Verification

1. After app starts observe new range taking effect
   - Here is the first packet after HEELO_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




2. Change HELLO_OBJ's init table limits and restart the app to observe the different limts 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 periocid function that is triggered by SCH message
- Deeper dive into app architecture, resources, roles and responsibilties

## Design

- **TBD**

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

## Verification

- Observe the 8 execute event messages
- Reset filter by restting the app

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

**Objectives**

- Show how objects can have functional requirements for the App's Reset command
- Show another usage of the Execute message as debug so when you enable it it can help but doesn't flood

**Design**

- **TBD**

- **Objectives**
- **Verify**
  - Reset app to see it go to the low limit
  - Change mode to decrement, wait and then reset to show it is reset to the high limit
  - Enable dubug messages and see the execute debug events
  - Reset app and show the filter is reset and applies to any event types

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

## Verification

- Reset app to observe it go to the low limit
- Change mode to decrement, wait and then reset to show it is reset to the high limit
- Enable dubug messages and see the execute debug events
- Reset app and show the filter is reset and applies to any event types