- **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

- **cFS apps can create one of more child tasks to perform functions that run in an execution thread separate from the parent app**

- **Child resource are owned by the parent app**

- **Parent and child share memory address space**

  - Use semaphores to prevent simultaneous memory access conflicts

- **Convention is to create child tasks when the parent app initializes**

  - Established for realtime systems that have strict timing requirements

  - System initialization timing is usually less stringent and dynamic resource management is minimized when the system is operational

  - See the child task 'Use Cases' and examples to help guide your decision

- **Use Cases**

  - Low priority back ground: FM and CS

  - High priority, typically short

    - Interface apps: bc42_intf, gpio_demo (sort of), mqtt_gw

- Low priority back ground: FM and CS

- Periodic tasks that run

- High priority, typically short

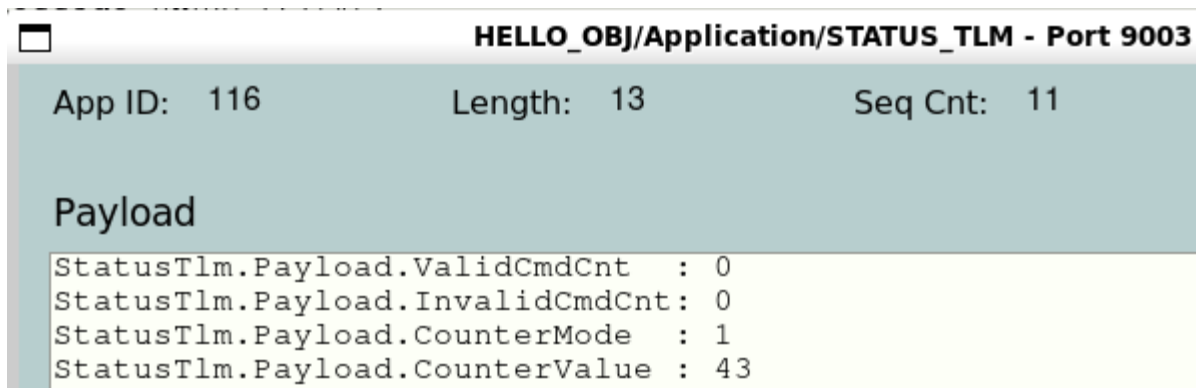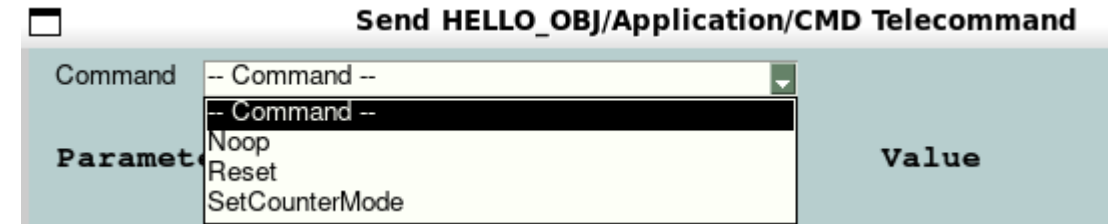 - Interface apps: bc42_intf, gpio_demo (sort of), mqtt_gw

Hello Child Functionality and Operations

- **Same functionality as the Hello Object app**

  – The EDS-defined Counter Mode type is in the status telemetry message (retained from coding lesson)

  – The EXOBJ_Execute() event message is defined as a DEBUG event and an event filter allows the first 8 events to be published (retained from coding lesson)

  – The App reset command resets the event filter. EXOBJ does not have any reset behavior.

  – The counter limits are defined in a new parameter

- **Additional functionality**

  – There's no additional functionality

  – The only change is how the functionality is implemented that is explained in the design section

Commands

- Retain Hello World's Noop and Reset commands

- New Set Counter Mode command

Send HELLO_OBJ/Application/CMD Telecommand

Command: -- Command --
-- Command --
Noop
Reset
SetCounterMode

Paramet...   Value

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
```
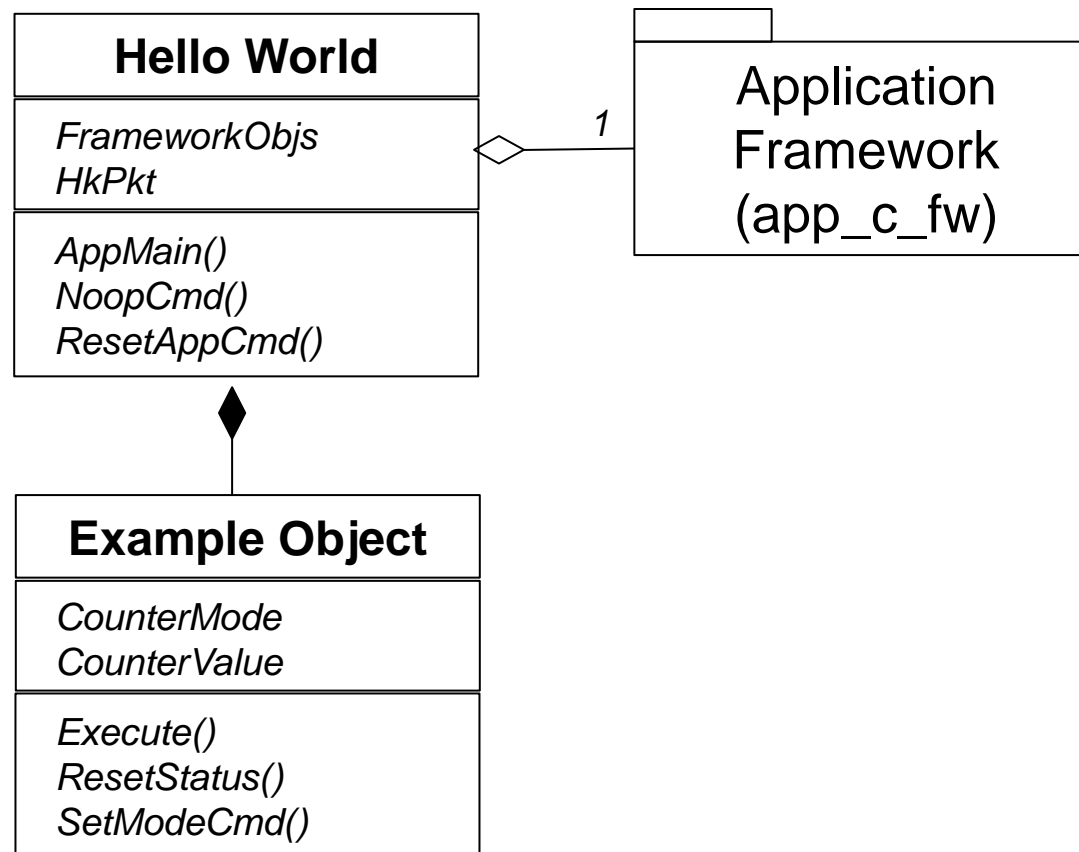
Telemetry

- 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
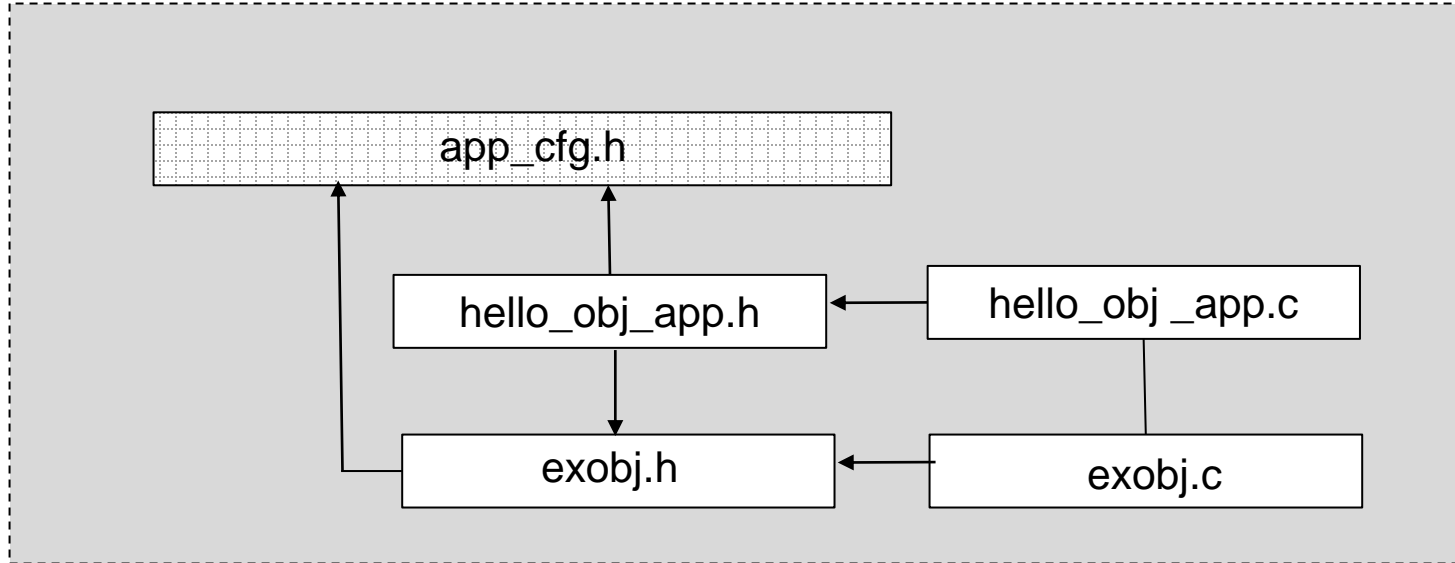
Hello Child Design

- **App Design**
  - Add ChildMgr framework to app
  - Add child task init table parameters
  - Constructor child task
  - NASA app sexamples

Hello World

FrameworkObjs
HkPkt

AppMain()
NoopCmd()
ResetAppCmd()

1

Application
Framework
(app_c_fw)

Example Object

CounterMode
CounterValue

Execute()
ResetStatus()
SetModeCmd()

App Source Files


```
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;
```

- 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

Initialize App

Start → Subscribe to send housekeeping, exec, command messages

Suspend execution until a message arrives on app's pipe

Pend on SB message pipe

Periodic *1Hz* message from SCH app

1 Hz? —*Yes*→ Call EXOBJ_Execute()
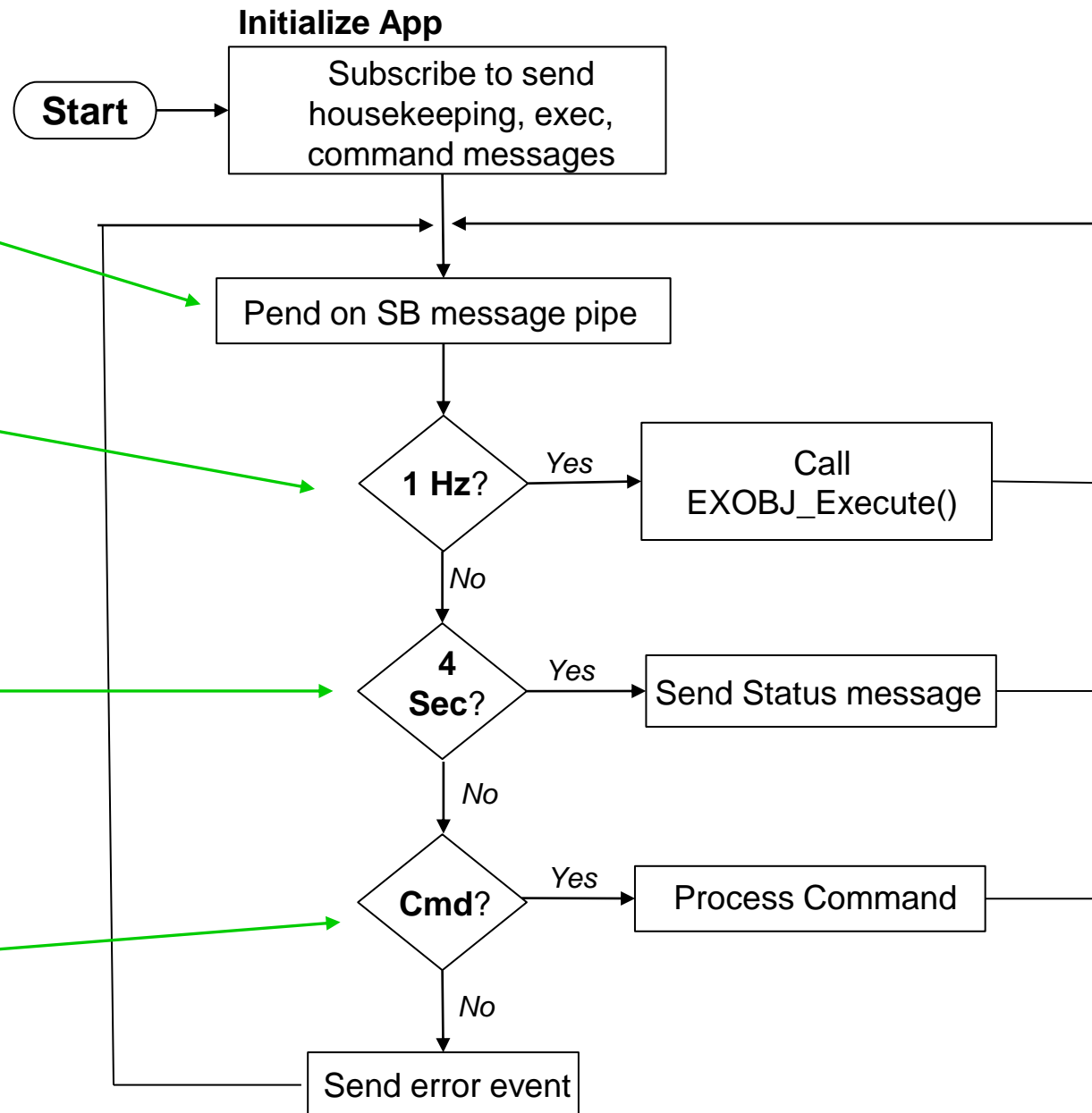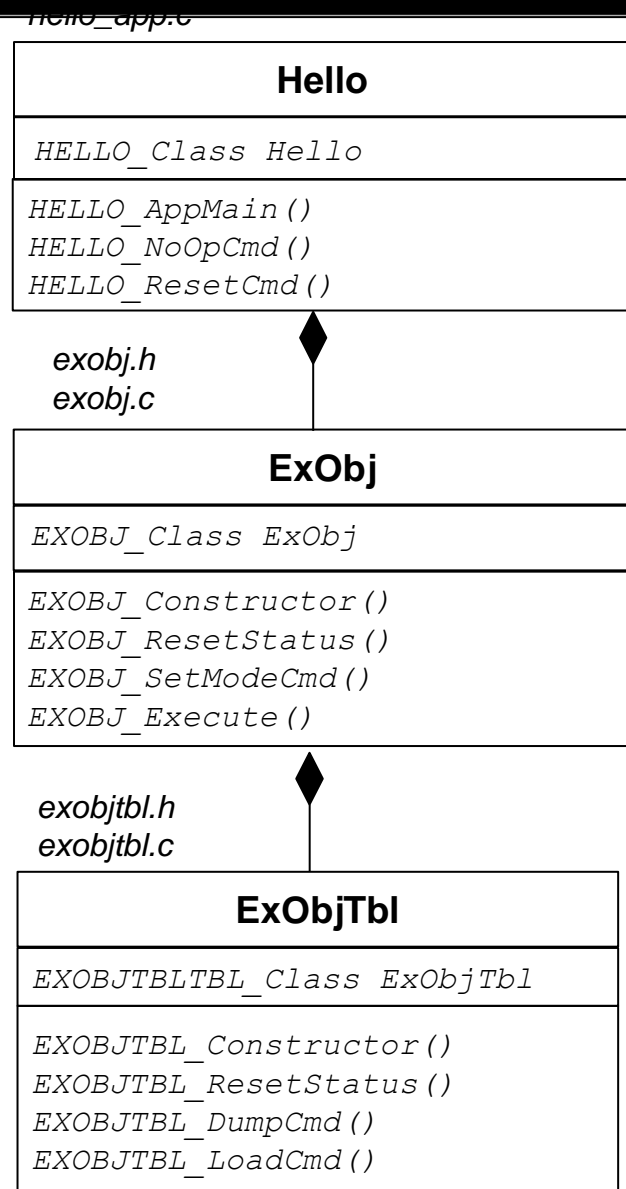
No

Periodic *4 second* message from SCH app
- Send status telemetry message
- "Housekeeping cycle" convenient time to perform non-critical functions

4 Sec? —*Yes*→ Send Status message

No

Process commands
- Commands can originate from ground or other onboard apps

Cmd? —*Yes*→ Process Command

No

Send error event

Lesson 1 Design Highlights

Hello_app.c

```
          Hello
─────────────────────────
HELLO_Class Hello
─────────────────────────
HELLO_AppMain()
HELLO_NoOpCmd()
HELLO_ResetCmd()
```

exobj.h
exobj.c

```
          ExObj
─────────────────────────
EXOBJ_Class ExObj
─────────────────────────
EXOBJ_Constructor()
EXOBJ_ResetStatus()
EXOBJ_SetModeCmd()
EXOBJ_Execute()
```

exobjtbl.h
exobjtbl.c

```
         ExObjTbl
─────────────────────────
EXOBJTBLTBL_Class ExObjTbl
─────────────────────────
EXOBJTBL_Constructor()
EXOBJTBL_ResetStatus()
EXOBJTBL_DumpCmd()
EXOBJTBL_LoadCmd()
```

- **The App C Framework is an object-based design written in C**
- **Apps are constructed as an aggregation of objects**
  – Hello contains one Example Object (ExObj)
  – ExObj contains one Example Object Table (ExObjTbl)
  – The object hierarchy can be as wide or deep as needed
- **The key roles of the main app are to**
  – Read the app's JSON initialization configuration file
  – Initialize contained objects and register their commands
  – Manage the main control loop
- **Contained objects implement the 'business logic'**
  – ExObj increments a counter during each execution cycle
  – ExObj's Set Mode command supports increment and decrement
  – ExObjTbl defines the counter's lower and upper limits

STEM

Open ware

Basecamp

hello_app.h

```
 97 typedef struct
 98 {
 99
100     /*
101     ** App Framework
102     */
103
104     INITBL_Class_t     IniTbl;
105     CMDMGR_Class_t     CmdMgr;
106     TBLMGR_Class_t     TblMgr;
107
108     /*
109     ** Command Packets
110     */
111
112
113     /*
114     ** Telemetry Packets
115     */
116
117     HELLO_HkPkt_t  HkPkt;
118
119     /*
120     ** HELLO State & Contained Objects
121     */
122
123     CFE_SB_PipeId_t  CmdPipe;
124     CFE_SB_MsgId_t   CmdMid;
125     CFE_SB_MsgId_t   ExecuteMid;
126     CFE_SB_MsgId_t   SendHkMid;
127     uint32           PerfId;
128
129     EXOBJ_Class_t  ExObj;
130
131 } HELLO_Class_t;
```

- Use a variation of the 'singleton" design pattern
  - Object constructors passed reference to owner's storage
  - `void EXOBJ_Constructor(EXOBJ_Class_t* ExObjPtr, const INITBL_Class_t* IniTbl);`
  - Contained objects store reference a static variable so subsequent function (or method) calls don't require a pointer to be passed

exobjtbl.h

```
73 typedef struct
74 {
75
76     /*
77     ** Table parameter data
78     */
79
80     EXOBJTBL_Data_t Data;
81
82     /*
83     ** Standard CJSON table data
84     */
85
86     const char*  AppName;
87     bool         Loaded;     /* Has
88     uint8        LastLoadStatus;
89     uint16       LastLoadCnt;
90
91     size_t       JsonObjCnt;
92     char         JsonBuf[EXOBJTBL_
93     size_t       JsonFileLen;
94
95 } EXOBJTBL_Class_t;
```

exobj.h

```
81 typedef struct
82 {
83
84     /*
85     ** State Data
86     */
87
88     EXOBJ_CounterModeType_t  CounterMode;
89     uint16  CounterValue;
90
91     /*
92     ** Contained Objects
93     */
94
95     EXOBJTBL_Class_t  Tbl;
96
97 } EXOBJ_Class_t;
```

Add Event Messages to ExObj Execute()

```
  ├─cisat_defs/
  │  └─cpu1_hello_tbl.json
  └─cfs-apps/
    └─hello/
      ├─eds/
      ├─fsw/src/
      │  └─exobj.c
```

- **Open** *./cfs-apps/hello/fsw/src/exobj.c in a text editor*
- **Event message function call**
  - `CFE_EVE_SendEvent(Event ID, Event Type, Format String, Values)`
- **Add events as show below**
  1. Send event when increment mode reaches its high limit
  2. Send event when decrement mode reaches its low limit

```
128 void EXOBJ_Execute(void)
129 {
130
131    if (ExObj->CounterMode == EXOBJ_CounterModeType_INCREMENT)
132    {
133       if (ExObj->CounterValue < ExObj->Tbl.Data.HighLimit)
134       {
135          ExObj->CounterValue++;
136       }
137       else
138       {
139          ExObj->CounterValue = ExObj->Tbl.Data.LowLimit;
140       }
141    } /* End if increment */
142    else
143    {
144       if (ExObj->CounterValue > ExObj->Tbl.Data.LowLimit)
145       {
146          ExObj->CounterValue--;
147       }
148       else
149       {
150          ExObj->CounterValue = ExObj->Tbl.Data.HighLimit;
151       }
152    } /* End if decrement */
153
154    CFE_EVS_SendEvent (EXOBJ_EXECUTE_EID, CFE_EVS_EventType_DEBUG,
155                        "%s counter mode: Value %d",
156                        CounterModeStr(ExObj->CounterMode), ExObj->CounterValue);
157
158
159 } /* End EXOBJ_Execute() */
```

```
131    if (ExObj->CounterMode == EXOBJ_CounterModeType_INCREMENT)
132    {
133       if (ExObj->CounterValue < ExObj->Tbl.Data.HighLimit)
134       {
135          ExObj->CounterValue++;
136       }
137       else
138       {
139          ExObj->CounterValue = ExObj->Tbl.Data.LowLimit;
140          CFE_EVS_SendEvent (EXOBJ_EXECUTE_EID, CFE_EVS_EventType_INFORMATION,
141                        "%s counter mode: Value reached high limit %d, resetting to low limit %d",
142                        CounterModeStr(ExObj->CounterMode),
143                        ExObj->Tbl.Data.HighLimit,
144                        ExObj->Tbl.Data.LowLimit);
145       }
146    } /* End if increment */
147    else
148    {
149       if (ExObj->CounterValue > ExObj->Tbl.Data.LowLimit)
150       {
151          ExObj->CounterValue--;
152       }
153       else
154       {
155          ExObj->CounterValue = ExObj->Tbl.Data.HighLimit;
156          CFE_EVS_SendEvent (EXOBJ_EXECUTE_EID, CFE_EVS_EventType_INFORMATION,
157                        "%s counter mode: Value reached low limit %d, resetting to high limit %d",
158                        CounterModeStr(ExObj->CounterMode),
159                        ExObj->Tbl.Data.LowLimit,
160                        ExObj->Tbl.Data.HighLimit);
161       }
162    } /* End if decrement */
```

1

2

- **Open** *./cfe-eds-framework/cfsat_defs/cpu1_hello_tbl.json in a text editor*

- **Change low limit from 0 to 50**

- **Change high limit from 100 to 60**

```
cfsat
├─cfe-eds-framework/
│  ├─cfsat_defs/
│  │  └─cpu1_hello_tbl.json
```

```
1 {
2    "app-name": "HELLO",
3    "tbl-name": "Limits",
4    "description": "Example table",
5    "low-limit":  0,
6    "high-limit": 100
7 }
```

1

```
1 {
2    "app-name": "HELLO",
3    "tbl-name": "Limits",
4    "description": "Example table",
5    "low-limit":  50,
6    "high-limit": 60
7 }
```

```
cfsat
├─cfe-eds-framework/
  └─build/exe/
          └─cpu1/
              └─cf/
```

1. **Change directory to *./cfe-eds-framework/***

   - `Make install`

2. **Change directory to *./cfe-eds-framework/build/exe/cpu1***

   - `./core-cpu1`

3. **From a different terminal change directory to *./cfsat/gnd-sys/app***

   - `. ./setvars.sh`

   - `Python3 cfsat.py`

   - `cFSAT: cFS Config -> Enable Telemetry`

4. **Try hello's Set Mode Command to verify your changes behave as expected**

Ground & Flight Events `Clear`

```
07:10:42 - cFSAT version 0.1.0 initialized with mission samplemission, target cpu1 on 02/06/2022
07:10:42 - cFSAT target host 127.0.0.1, command port 1234, telemetry port 1235
07:10:48 - TO_LAB EnableOutputCmd command sent
07:10:49 - CFE_EVS AddEventFilterCmd command sent
07:10:49 - CFE_EVS AddEventFilterCmd command sent
07:10:50 - FSW Event at 1001153: TO_LAB_APP, 2 - TO telemetry output enabled for IP 127.0.0.1
07:10:52 - FSW Event at 1001156: CFE_TIME, 2 - Start FLYWHEEL
07:10:54 - FSW Event at 1001158: CFE_TIME, 2 - Stop FLYWHEEL
07:11:25 - Created telemetry screen for HELLO/Application/HK_TLM
07:11:42 - FSW Event at 1001206: HELLO, 2 - INCREMENT counter mode: Value reached high limit 60, resetting to low limit 50
07:11:53 - FSW Event at 1001217: HELLO, 2 - INCREMENT counter mode: Value reached high limit 60, resetting to low limit 50
07:12:05 - FSW Event at 1001228: HELLO, 2 - INCREMENT counter mode: Value reached high limit 60, resetting to low limit 50
07:12:16 - FSW Event at 1001239: HELLO, 2 - INCREMENT counter mode: Value reached high limit 60, resetting to low limit 50
```

Hello Object Coding Lessons

Objectives

- Increase knowledge of EDS

- Show app_cfg.h EDS include

Design

- **TBD**

Verification

- Open telemetry page and see default mode

- Issue mode command to decrement

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**

Verification

- After app starts observe new range taking effect

- Change limits and restart app

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**

Verification

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

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

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