



cFS Basecamp Hello Child Coding Lessons



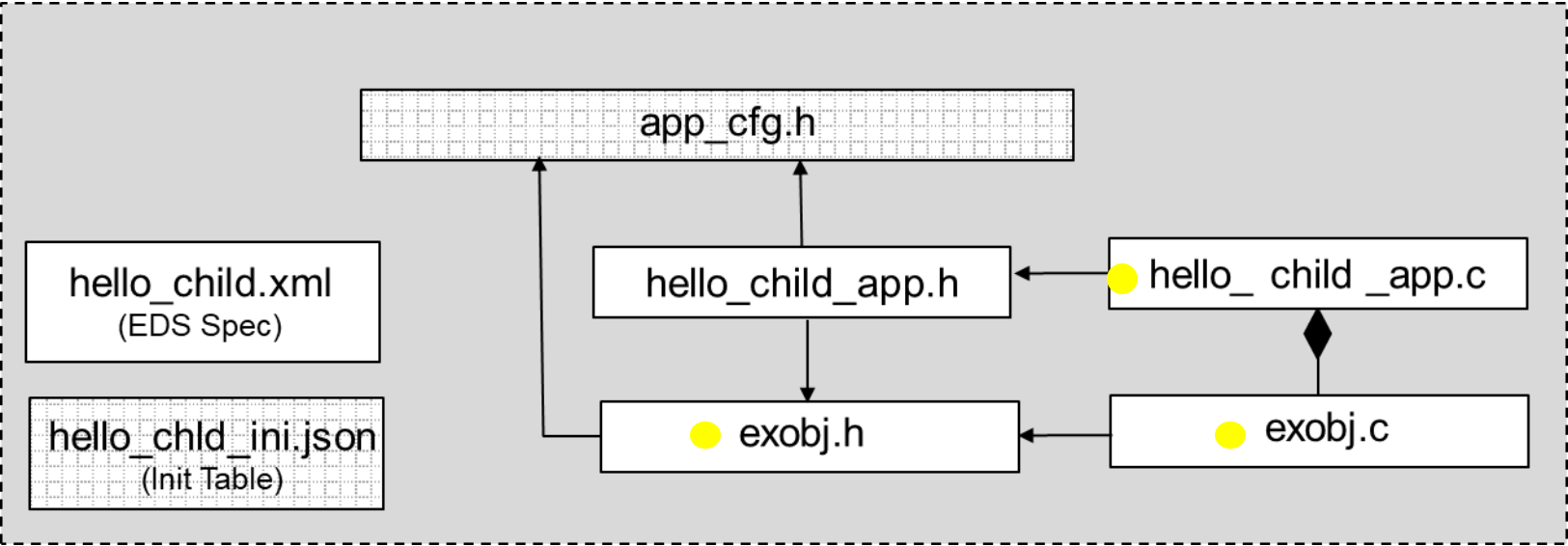
Version 2.7
July 2025

- These slides provide guidance for doing the Hello Child 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 Child app template includes the Hello Object app and some of the lesson functionality
 - The EDS-defined Counter Mode type is in the status telemetry message (retained from coding lesson)
 - The EXOBJ_Execute() event message is defined as an INFORMATION event and an event filter allows the first 16 events to be published (similar to hello object coding lesson)
 - The counter limits are defined in the JSON init file
 - The App reset command resets the event filter. EXOBJ does not have any reset behavior
 - A new command allows the child task execution delay to be modified
 - The coding exercises introduce developer’s to mutex and counting semaphores
- Prerequisites
 - Completed Hello Object coding tutorial and met its prerequisites

Objectives

- Learn how to create and use a Mutex Semaphore to control access to a resource that is shared between two threads

● The following files are modified in this lesson



exobj.h

- Add a new variable called `ChildDataSemaphore` to hold the Mutex Semaphore identifier

exobj.c

- The Mutex Semaphore is created in the constructor by calling `OS_MutSemCreate()`
- The stack data structure and global function `EXOBJ_StackPop()` already exists.
- This lesson adds
 - A new static `StackPush()` function that includes calls to `OS_MutSemTake()` and `OS_MutSemGive()`
 - Calls to `OS_MutSemTake()` and `OS_MutSemGive()` in `EXOBJ_StackPop()`
 - A call to `StackPush()` in `ManageCounter()`
- These changes cause the child task to push a new stack counter entry for each execution

hello_child_app.c

- Add a new static function `ProcessExObjStack()` that pops all data counter stack entries and sends an event message containing the contents of the entry
 - The event message is filtered
- Call the new `ProcessExObjStack()` function from `ProcessComamnds()` when the 1Hz message is received

- 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 mutex semaphore**

1. The default configuration should result in EXOBJ pushing counter entries and the main app task popping the entries

- The events are filtered so you should see the first 16

```
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[21.76395]: INCREMENT counter mode: Value 1. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:03:21.76395]: 1
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[23.76467]: INCREMENT counter mode: Value 2. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:03:23.76467]: 2
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[25.76594]: INCREMENT counter mode: Value 3. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:03:25.76594]: 3
```

2. Use the HELLO_CHILD SetChildDelay command to experiment with different timing relationships between the two tasks

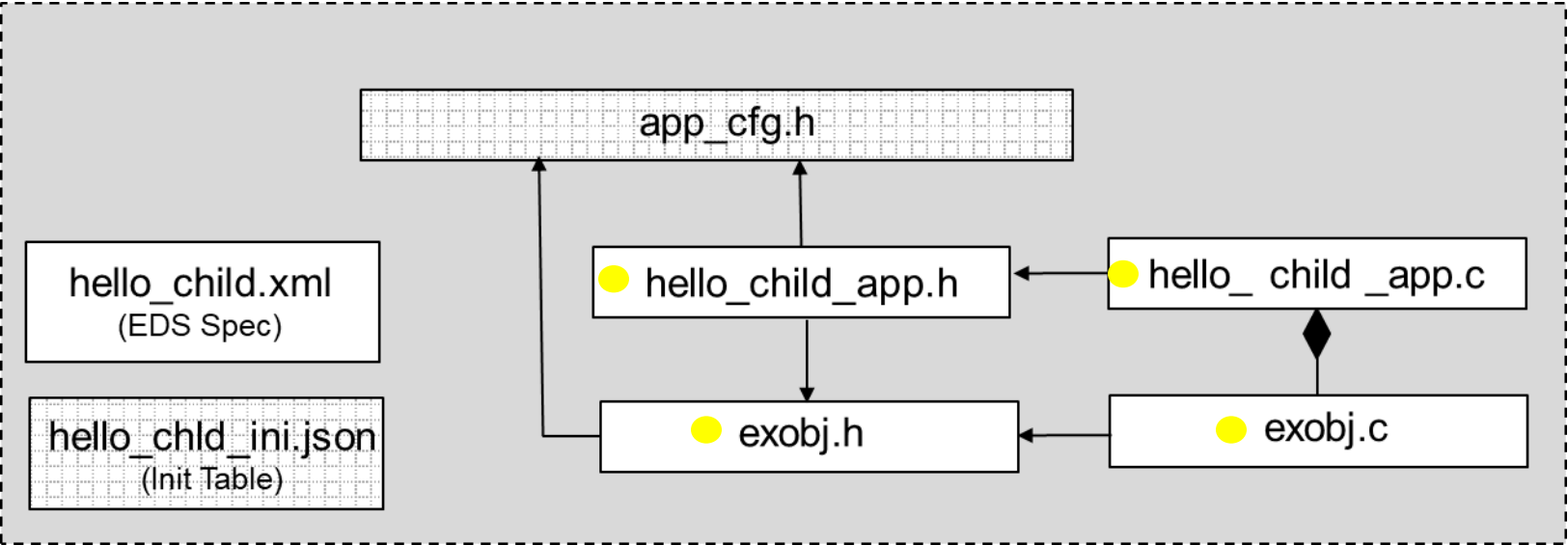
- The child delay units are in milliseconds
- Here’s an example with a 200ms delay

```
EVS Port1 66/1/HELLO_CHILD 120: Child task loop delay changed from 1000 to 200
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[17.39351]: INCREMENT counter mode: Value 66. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[17.79395]: INCREMENT counter mode: Value 67. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:11:17.79395]: 67
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:11:17.39351]: 66
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[18.19416]: INCREMENT counter mode: Value 68. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[18.59461]: INCREMENT counter mode: Value 69. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[18.99503]: INCREMENT counter mode: Value 70. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:11:18.99503]: 70
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:11:18.59461]: 69
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:11:18.19416]: 68
```

Objectives

- Learn how to create and use a Counting Semaphore to control the execution of a child task

● The following files are modified in this lesson



exobj.h

- Add a new variable called `ChildExecSemaphore` to hold the Counting Semaphore identifier
- Since `EXOBJ` doesn't own the counting semaphore a new parameter is added to `EXOBJ_Constructor()` so the semaphore ID can be passed to `EXOBJ`

exobj.c

- The constructor is updated to save the counting semaphore ID
- In `EXOBJ_ChildTask()`'s the task delay statement is replaced with a call to `OS_CountSemTake()` because it will control when the child task executes

hello_child_app.h

- Add a new variable called `ChildExecSemaphore` to hold the Counting Semaphore identifier

hello_child_app.c

- In `InitApp()` create the Counting Semaphore by calling `OS_CountSemCreate()`
- In `ProcessCommands()`
 - In the 1Hz logic replace the call to `ProcessExObjStack()` with `OS_CountSemGive()` which will cause the child task to execute
 - A call to `ProcessExObjStack()` is added to the 4second logic
- This child task execution logic isn't very practical, its purpose is to illustrate counting semaphores

- 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 counting semaphore**

1. The default configuration should result in EXOBJ pushing counter entries at 1Hz and the main app task popping the entries every 2 seconds
- The events are filtered so you should see the first 16
 - During startup the scheduler app timing may be erratic. You should see the events settle into the following pattern:

```
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[00.00594]: INCREMENT counter mode: Value 6. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[01.00680]: INCREMENT counter mode: Value 7. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[02.00760]: INCREMENT counter mode: Value 8. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[03.00757]: INCREMENT counter mode: Value 9. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:03.00757]: 9
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:02.00760]: 8
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:01.00680]: 7
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:00.00594]: 6
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[04.00854]: INCREMENT counter mode: Value 10. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[05.00906]: INCREMENT counter mode: Value 11. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[06.00915]: INCREMENT counter mode: Value 12. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 122: EXOBJ Manage Counter[07.00968]: INCREMENT counter mode: Value 13. Limits: Low=0, High=99
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:07.00968]: 13
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:06.00915]: 12
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:05.00906]: 11
EVS Port1 66/1/HELLO_CHILD 104: EXOBJ StackPop [1980-012-14:18:04.00854]: 10
```