# cFS Basecamp
# Hello World Coding Lessons

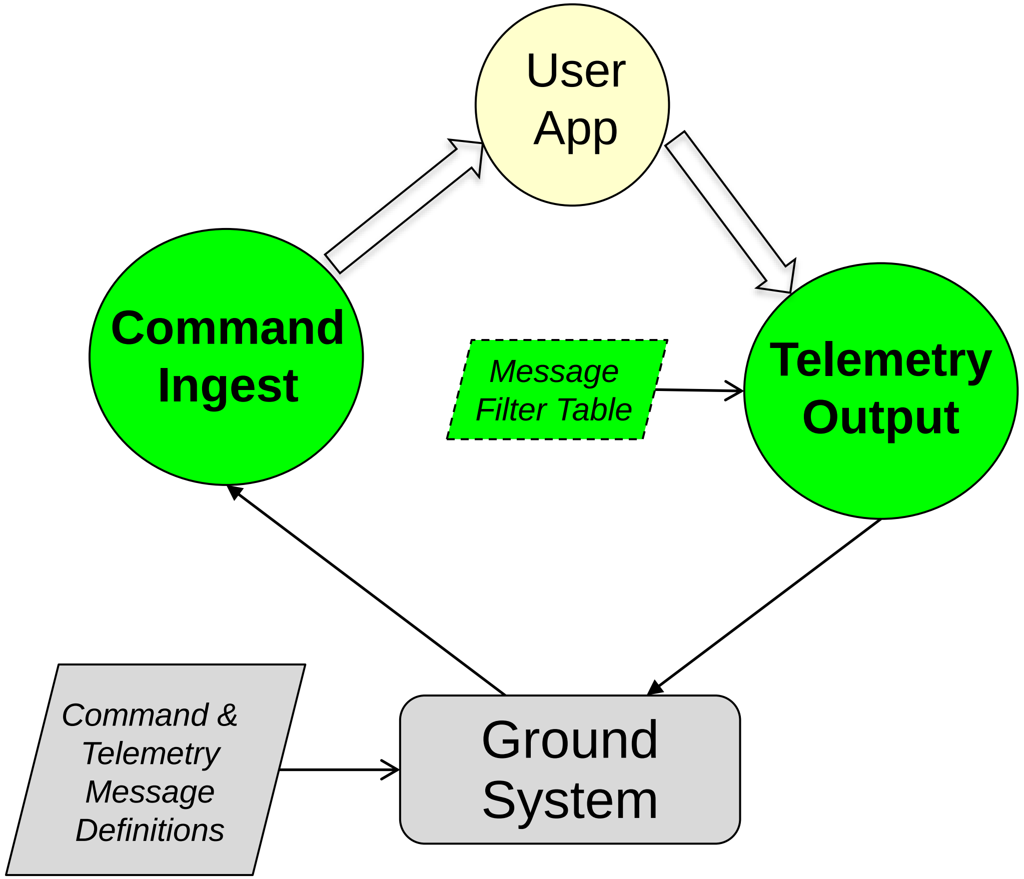
Version 1.8

August 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

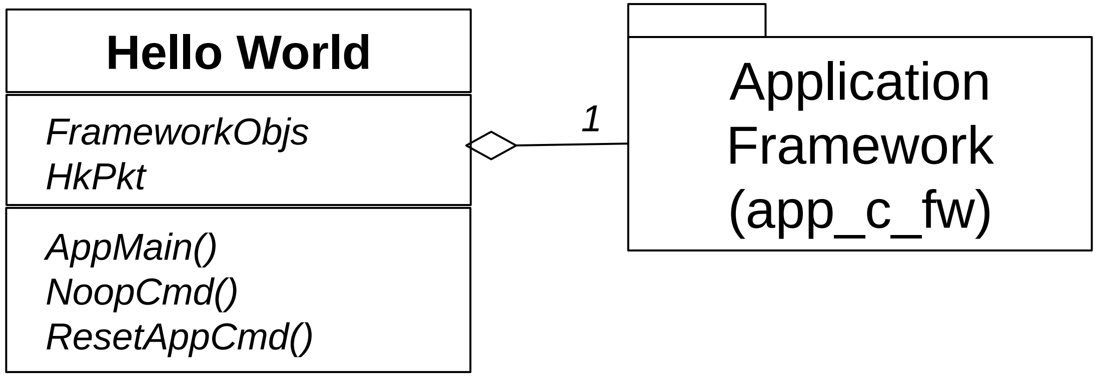
# App Context Diagram



- **Command Ingest (CI) App**
  - Receives commands from an external source, typically the ground system, and sends them on the software bus
- **Telemetry Output (TO) App**
  - Receives telemetry packets from a the software bus and sends them to an external source, typically the ground system
  - Optional *Filter Table* that provides parameters to algorithms that select which messages should be output on the external communications link

- **Different versions of CI and TO used on different platforms**
  - cFE delivered with ‘lab’ versions that use UDP for the external comm
  - JSC released versions that use a configurable I/O library for a different external comm links
  - OSK versions use UDP and a JSON filter table
  - ITAR-restricted flight versions typically used inflight

# Hello World Object Design



# Application Run Loop Messaging Example

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

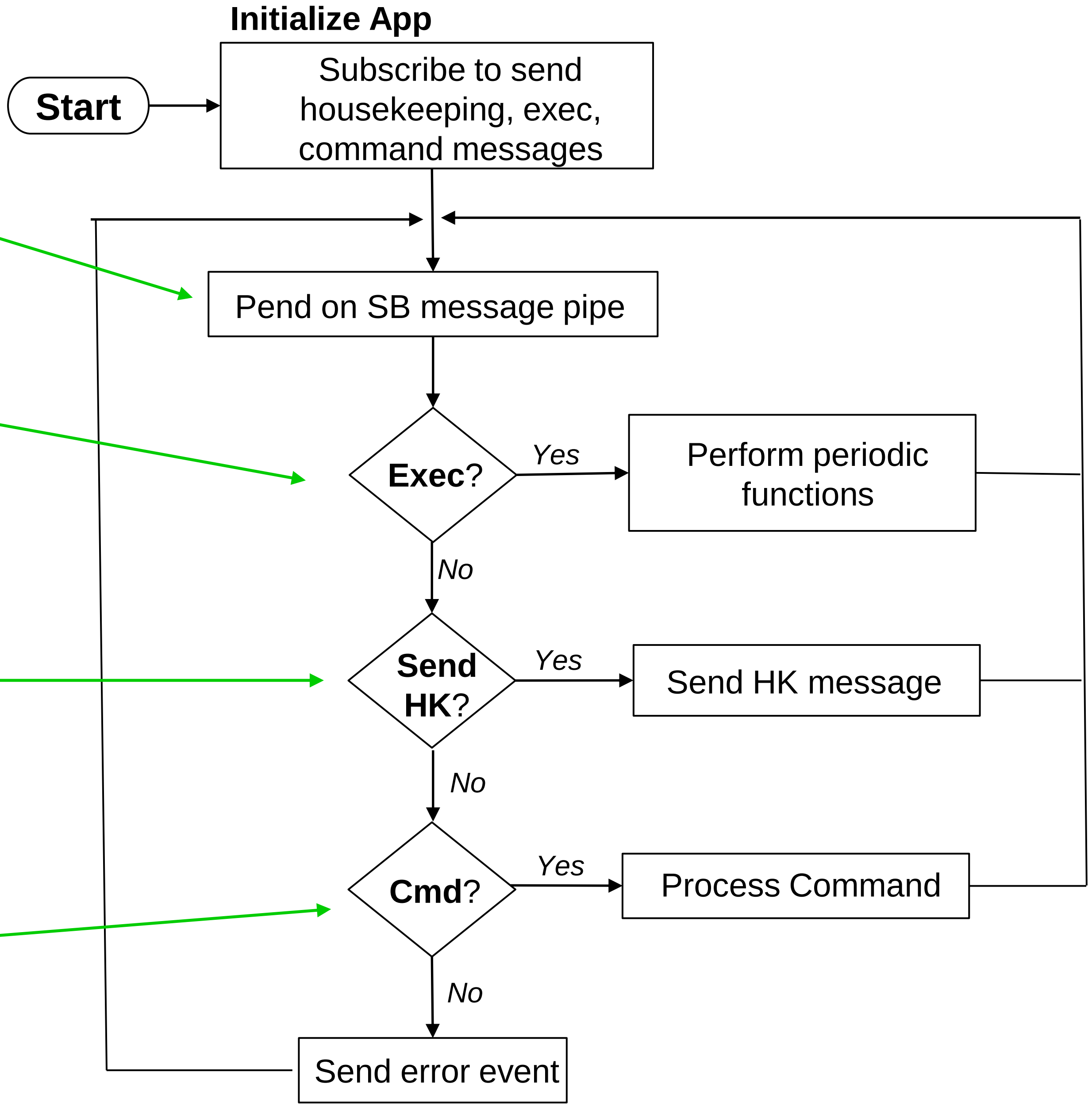
**Periodic *execute* message from SCH app**

**Periodic *send housekeeping* message from SCH app**

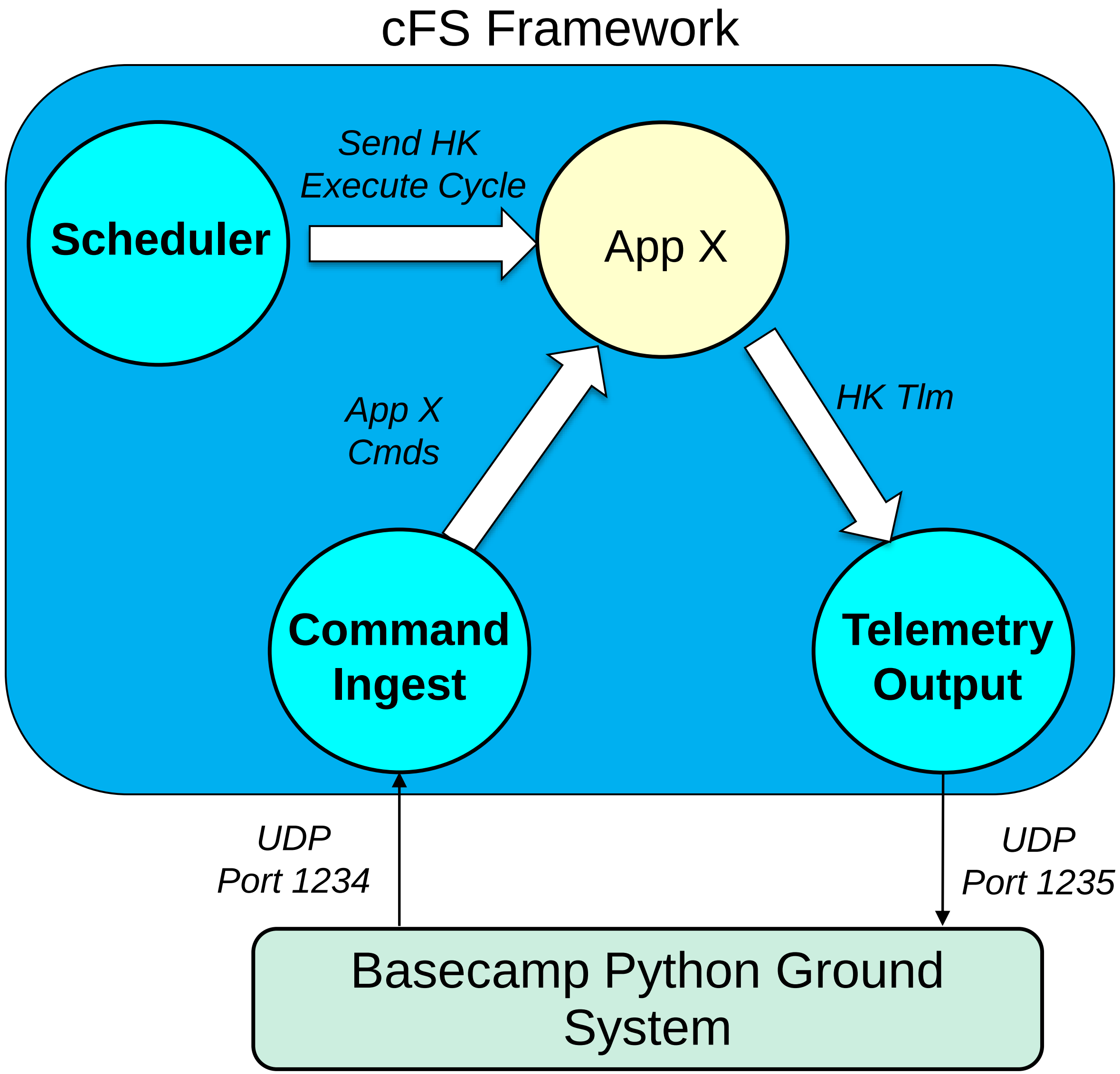
- Typically, on the order of seconds
- "Housekeeping cycle" convenient time to perform non-critical functions

**Process commands**

- Commands can originate from ground or other onboard apps



# cFS Application Runtime Environment



**A core set of apps are required to provide a runtime environment**

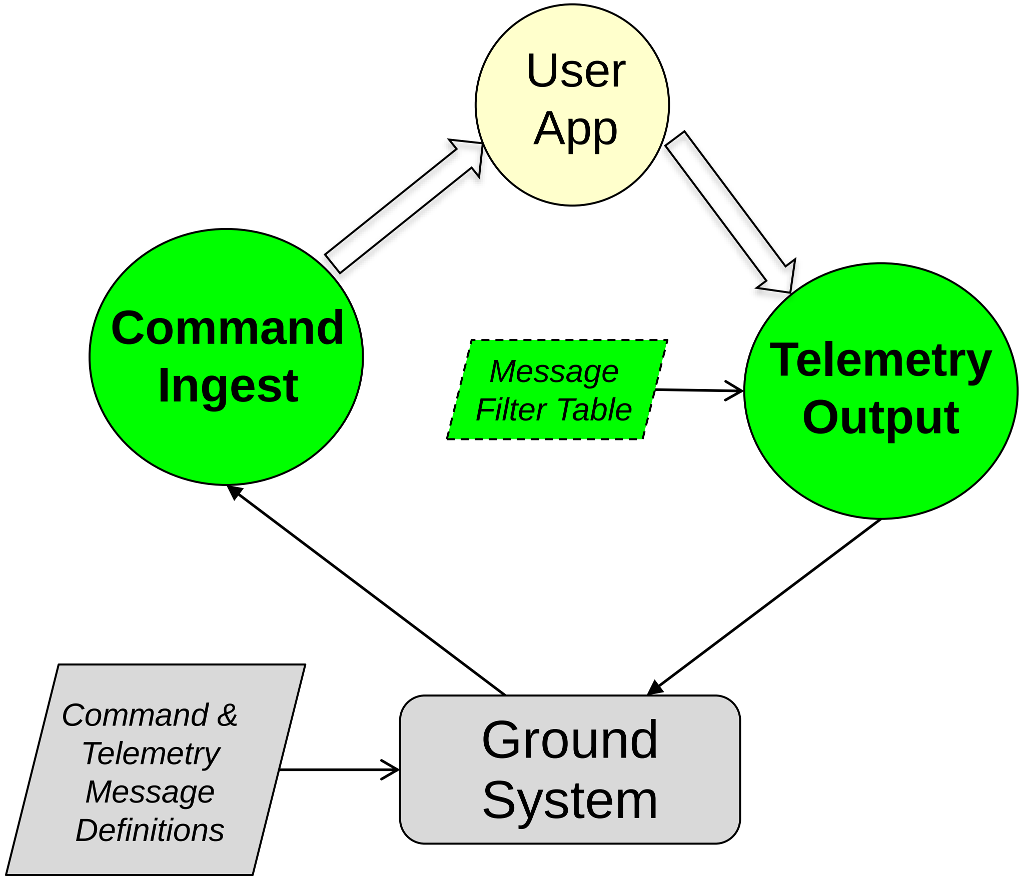
- Different app implementations can provide customized solutions for different platforms
- File management & transfer not shown

**Scheduler (SCH)** sends messages at fixed time intervals to signal apps to perform a particular function

**Command Ingest (CI)** receives commands from an external source and publishes them on the SB

**Telemetry Out (TO)** receives messages from the SB and sends them to an external destination

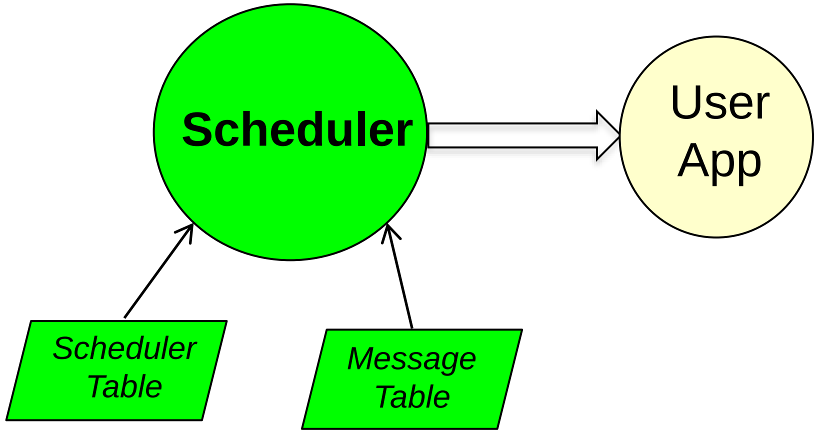
# Command & Telemetry Context



- **Command Ingest (CI) App**
  - Receives commands from an external source, typically the ground system, and sends them on the software bus
- **Telemetry Output (TO) App**
  - Receives telemetry packets from a the software bus and sends them to an external source, typically the ground system
  - Optional *Filter Table* that provides parameters to algorithms that select which messages should be output on the external communications link

- **Different versions of CI and TO used on different platforms**
  - cFE delivered with ‘lab’ versions that use UDP for the external comm
  - JSC released versions that use a configurable I/O library for a different external comm links
  - OSK versions use UDP and a JSON filter table
  - ITAR-restricted flight versions typically used inflight

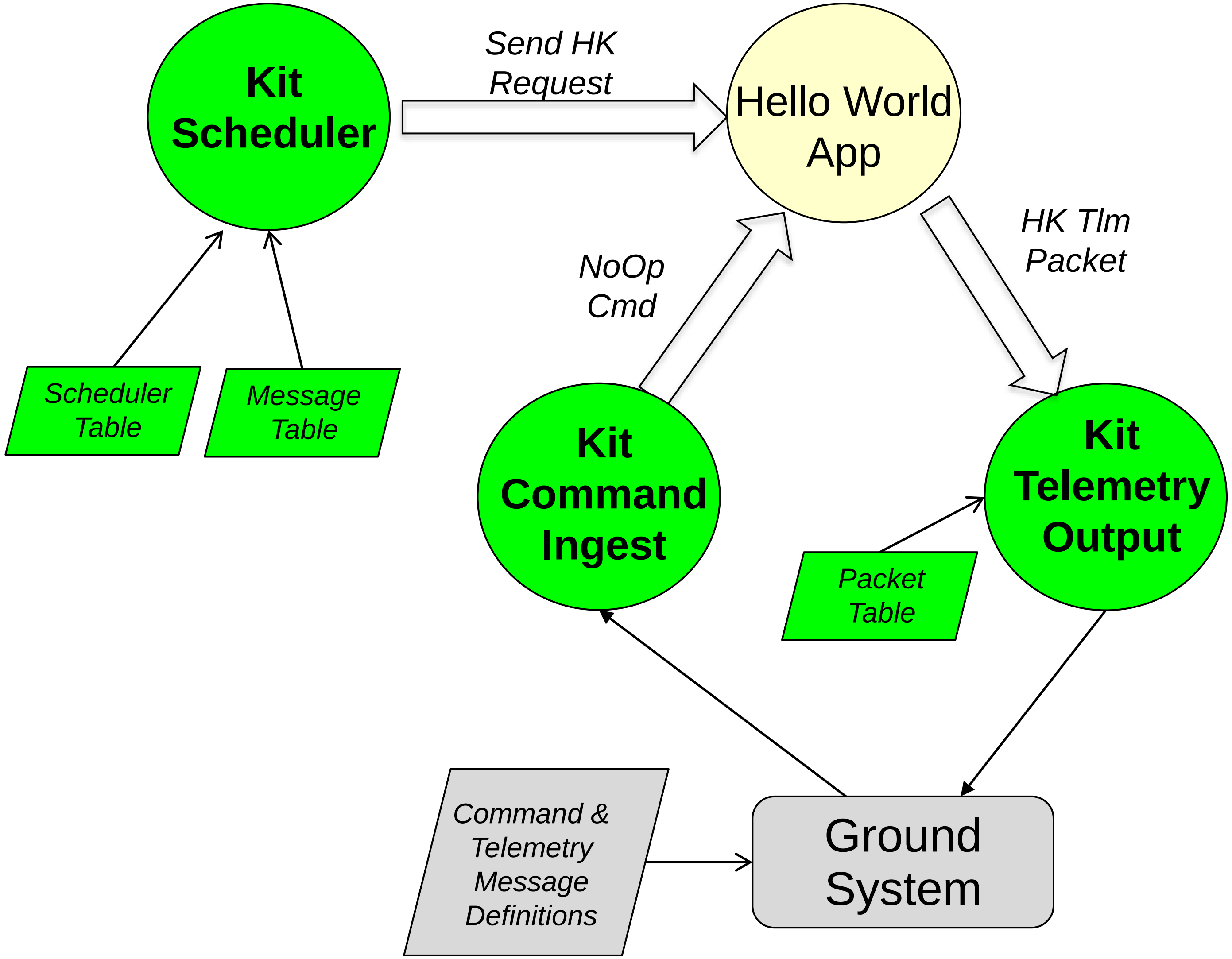
# Application Scheduling Context



- **Scheduler (SCH) App**
  - Synchronizes execution with clock’s 1Hz signal
  - Sends software bus messages defined in the *Message Table* at time intervals defined in the *Scheduler Table*

- **Application Control Flow Options**
  - Pend indefinitely on a *SB Pipe* with subscriptions to messages from the Scheduler
    - This is a common way to synchronize the execution of most of the apps on a single processor
    - Many apps send periodic “Housekeeping” status packets in response to a “Housekeeping Request message from Scheduler
  - Pend indefinitely on a message from another app
    - Often used when an application is part of a data processing pipeline
  - Pend with a timeout
    - Used in situation with loose timing requirements and system synchronization Is not required
    - The SB timeout mechanism uses the local oscillator so the wakeup time may drift relative to  the 1Hz

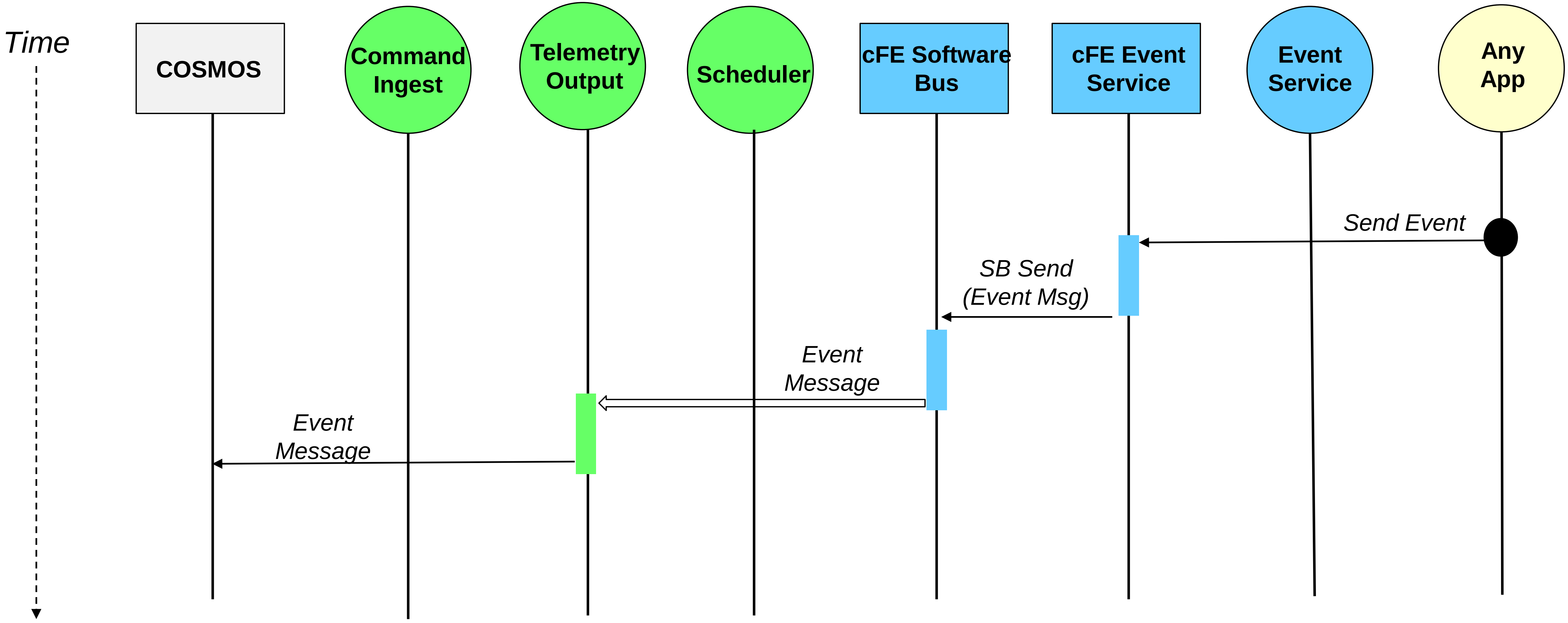
# Hello World App Runtime Environment



## Context of ”Hello World” app created in the next section

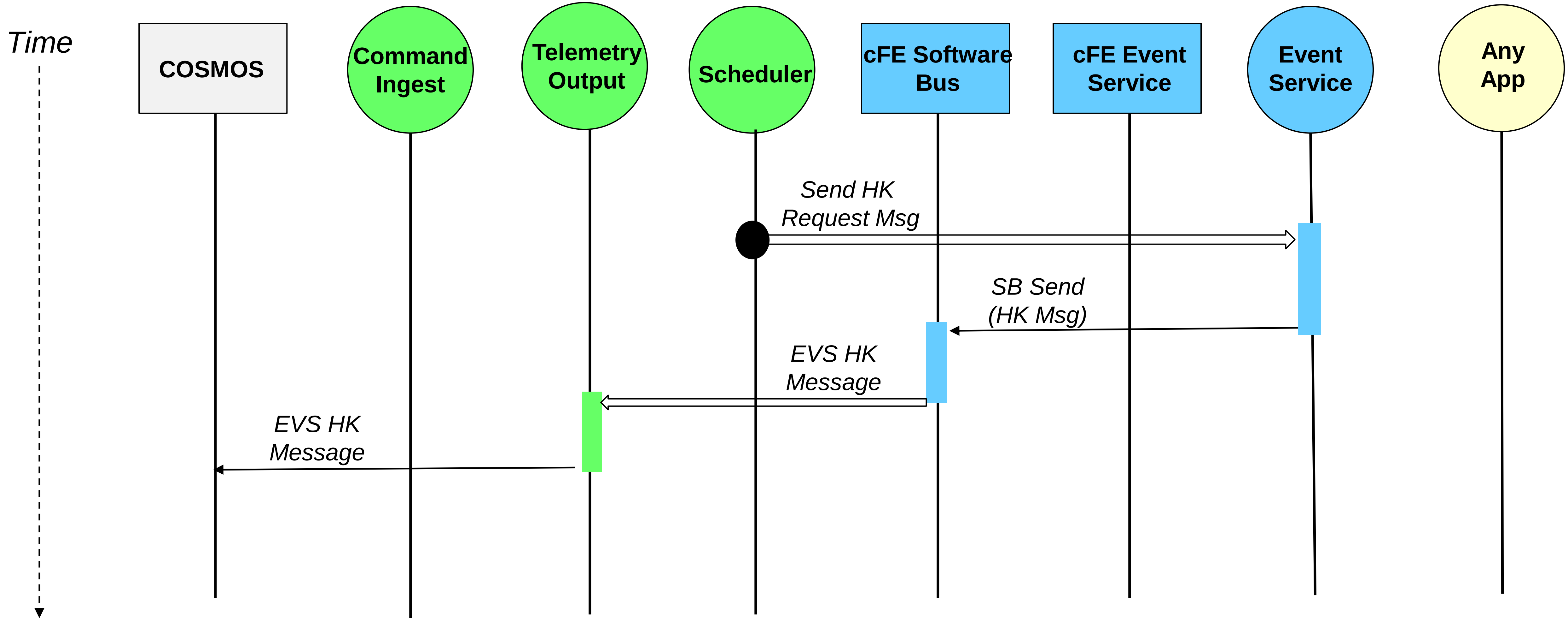
- Every 3 seconds Scheduler sends a “Send Housekeeping Telemetry Request”
  - HK telemetry includes valid and invalid command counters
- When user sends a “No Operation” command from the ground system Hello World responds with
  - An event message that contains the app’s version number
  - Increments the command valid counter

# App Send Event Sequence Diagram



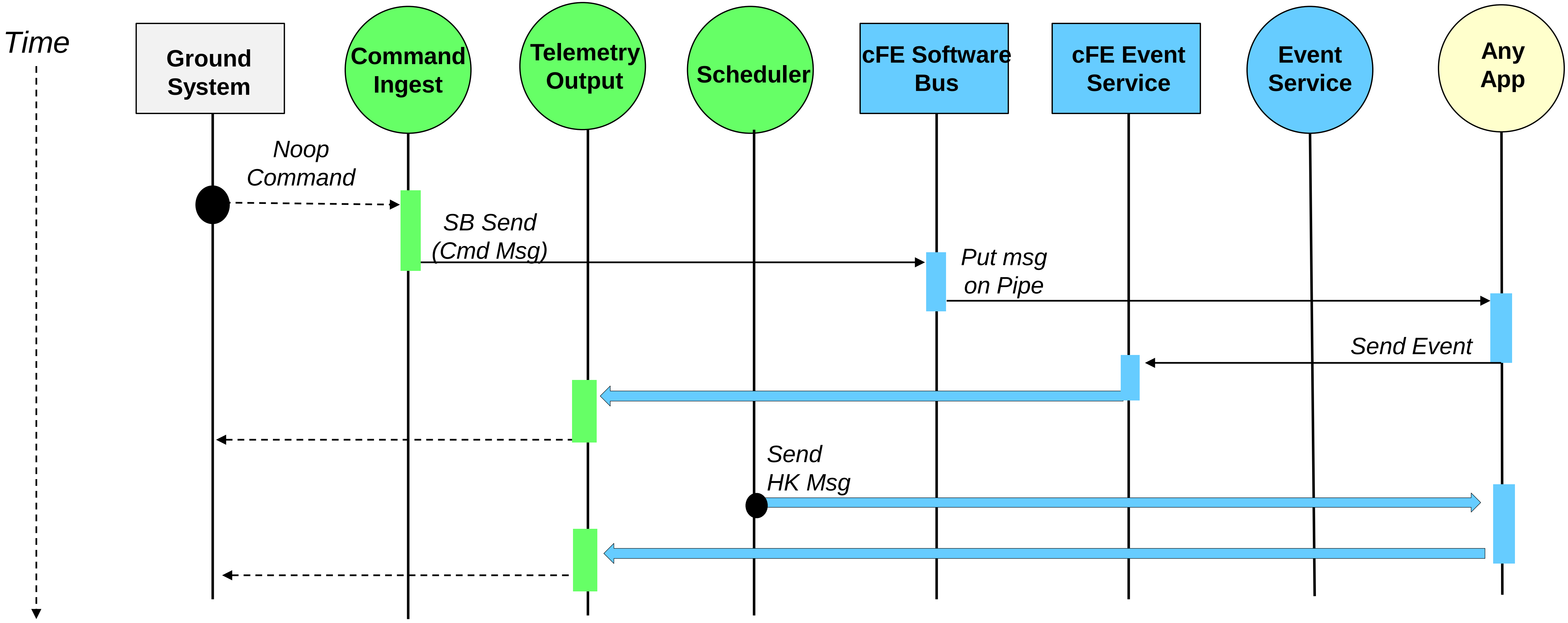
● = Initial event

# Event Service App Sends Housekeeping Telemetry



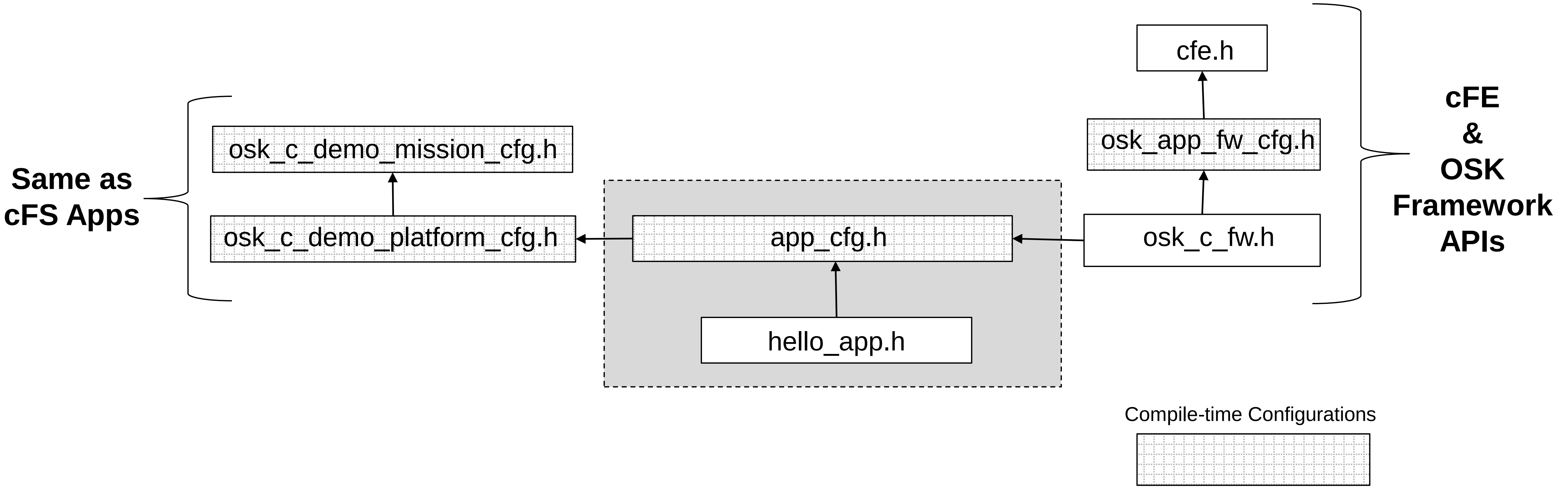
● = Initial event

# No Operation (Noop) Command Sequence



● Start Event - - - ▸ Network Comm ⇨ Software Bus Message → Library Call

# Object Composition Model – Header Files Inclusion Tree


Same as
cFS Apps
osk_c_demo_mission_cfg.h
osk_c_demo_platform_cfg.h
app_cfg.h
hello_app.h
cfe.h
osk_app_fw_cfg.h
osk_c_fw.h
cFE
&
OSK
Framework
APIs
Compile-time Configurations