What's New in F´?

Michael Starch
NASA Jet Propulsion Laboratory
2023-08-07

Jet Propulsion Laboratory
California Institute of Technology

What is F´: A Product Line for Flight Software

Software Architecture

- Decomposes an embedded software system into discrete components with well defined interfaces that communicate over ports interconnected in a topology
- Architectural features include: rapid development, portability, high performance, component reusability, software system analyzability and testability

C++ Framework

- Adheres to the F′ architecture and is used to construct applications
- Provides basic features such as message queues, threading and an OS abstraction layer

Component Library

- Provides a growing collection of generic components for common capabilities that can be incorporated without modification into new software projects
- Examples include command dispatch, event logging, and memory management

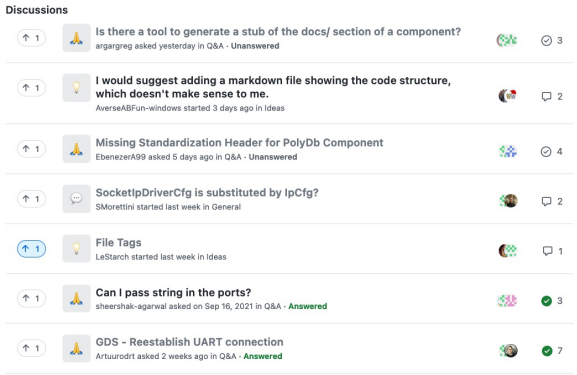
Suite of Tools

- Includes tools for specifying components and their connections and automatically generating a partial implementation from the specification
- Also includes tools for testing software at the unit and integration levels including a lightweight Ground Data System

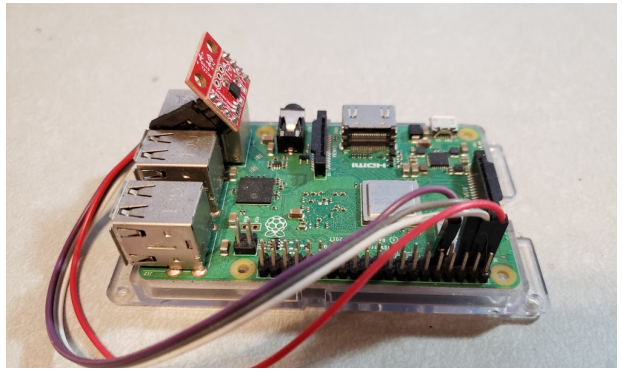
F´ is a product line enabling efficient development of flight software.

F´ and Open Source

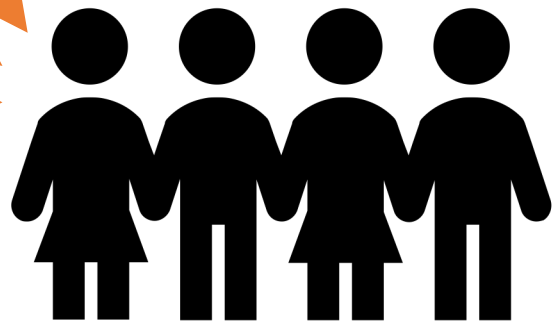
Documentation

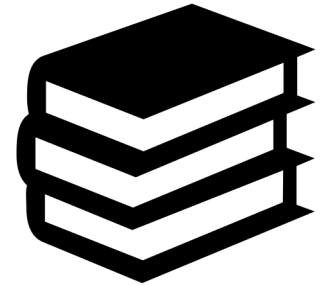


Support and Issues

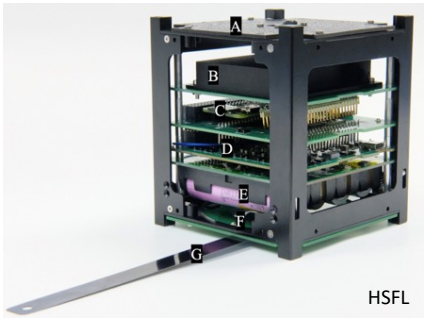


References / Examples





Useful Libraries

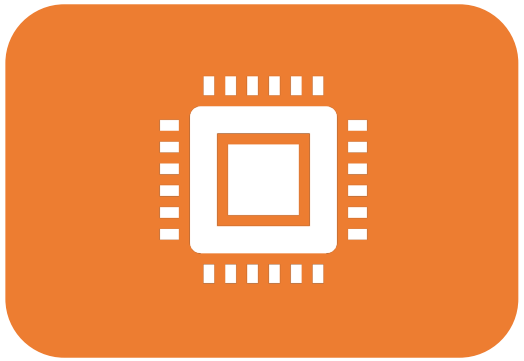


New Use Cases

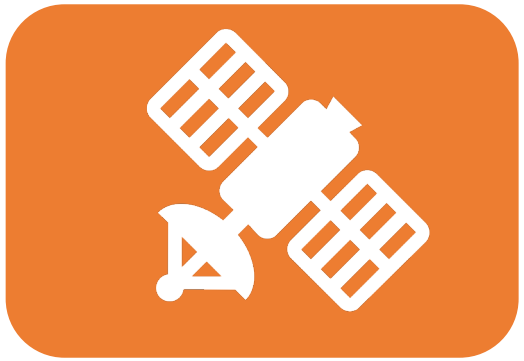
F´ is Open-Source software with a userbase that has grown from simple consumers into creators providing new use cases, useful libraries, support, bug reports, and bug fixes.

What's New: The F´ Product Line Roadmap

Maintenance



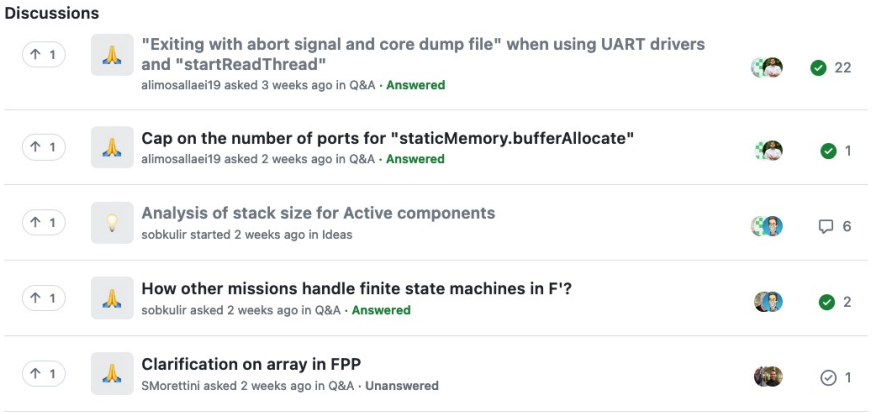
Baremetal Support

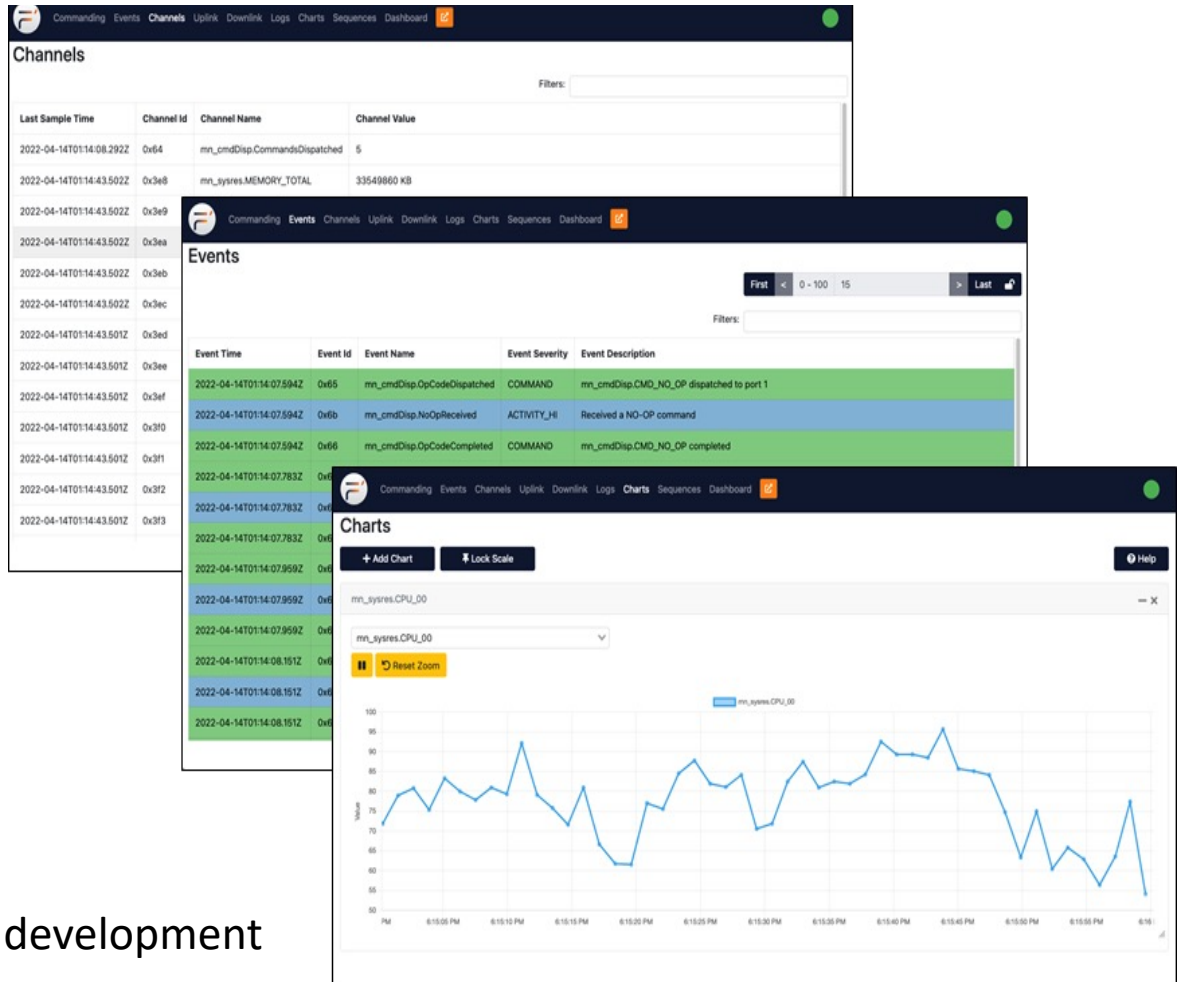


Class-B Standards

Cybersecurity

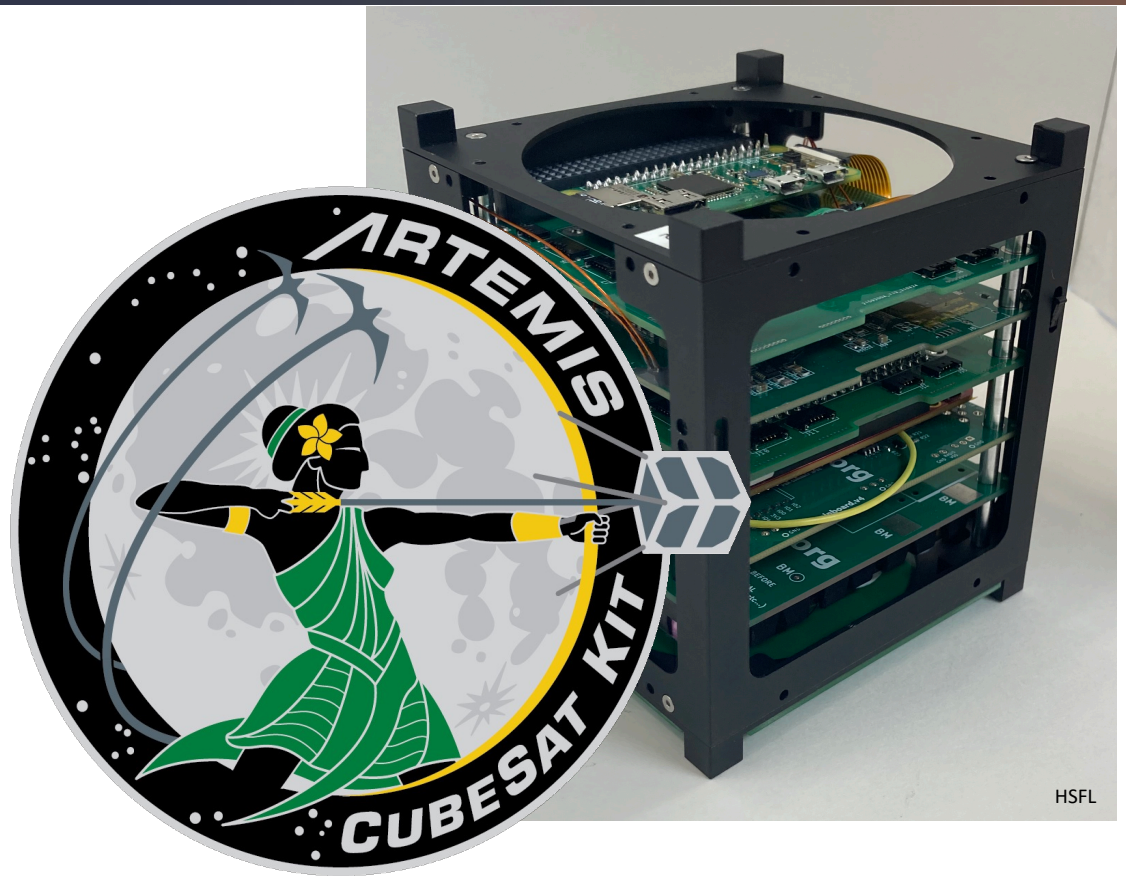
Roadmap: Maintenance, Tooling, and Support





F´ will continue to maintain the framework and tools that enable development and provide support to users.

Roadmap: Baremetal Best Practices



F´ will provide guidance and recommendations to users deploying F´ on systems without operating systems.

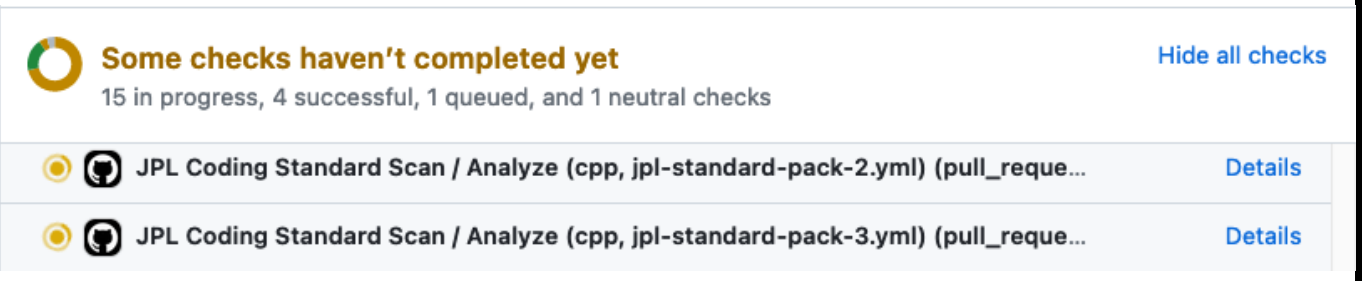
```
=======================================
       Deployment: Ref
=======================================
-----------------------------
    Size for F' Components
-----------------------------
.bss (Bytes)    Component
16              Ref::cycleLock
224             Ref::fatalHandler
224             Ref::linuxTime
224             Ref::textLogger
480             Ref::fatalAdapter
608             Ref::rateGroupDriverComp
1000            Ref::downlink
1144            Ref::fileUplinkBufferManager
1296            Ref::comm
1328            Ref::SG1
1328            Ref::SG2
1328            Ref::SG3
1328            Ref::SG4
1328            Ref::SG5
1336            Ref::blockDrv
1432            Ref::recvBuffComp
1448            Ref::fileManager
1448            Ref::pingRcvr
1544            Ref::fileUplink
1568            Ref::typeDemo
1616            Ref::systemResources
1928            Ref::sendBuffComp
2208            Ref::eventLogger
2480            Ref::rateGroup1Comp
2480            Ref::rateGroup2Comp
2480            Ref::rateGroup3Comp
2768            Ref::cmdSeq
3104            Ref::fileDownlink
3304            Ref::uplink
5464            Ref::prmDb
9336            Ref::staticMemory
11992           Ref::cmdDisp
14744           Ref::health
19776           Ref::tlmSend

--------------------------------
    Minimum F' Configurations
--------------------------------
=== Number of Telemetry Channel Hash Slots (config/TlmChanImplCfg.hpp:45)
        - TLMCHAN_NUM_TLM_HASH_SLOTS = 21
=== Number of Telemetry Channel Buckets (config/TlmChanImplCfg.hpp:50)
        - TLMCHAN_HASH_BUCKETS = 86
=== Number of Commands (config/CommandDispatcherImplCfg.hpp:14)
        - CMD_DISPATCHER_DISPATCH_TABLE_SIZE = 80

-----------------------------
    Size for Linux
-----------------------------
   text    data     bss     dec     hex filename
1026253   23608  116600 1166461  11cc7d /home/echee/fprime-projects/fprime/Ref/build-artifacts/Linux/Ref/bin/Ref
```

Roadmap: Class-B Software Enhancements

	Function / Data Usage			
17	Are declarations grouped into global and static/local?	Yes	Yes	
18	Are all variables locally defined unless global or specific visibility is required?	Yes	Yes	
19	Are global declarations uniquely named and namespaced?	Yes	Yes	
20	Is the code free of any literals that are not properly documented in the form of macros or static constants (i.e. magic numbers)?	Yes	Yes	
21	Have hardware specific code/data been sufficiently documented?	n/a	n/a	No hardware-specific computation
22	Is the precision of floating point numbers sufficient to ensure accuracy and floating-point equivalence is evaluated with margin (i.e. safe floating-point usage), and is not used in a loop control?	n/a	n/a	No floating-point computation
23	Are all variables initialized before use?	Yes	Yes	
24	Have all non-atomic data elements been protected from corruption?	Yes	Yes	
25	Have all function input arguments been validated prior to use?	Yes	Yes	
26	Have unused arguments to functions been documented?	Yes	Yes	
27	Verify that all commands are acknowledged success/failure	n/a	n/a	No commands
28	Verify that every EVR includes protection from cyclic generation. For persistent error conditions, the pattern should be a throttled EVR, and the total count of errors should be reported in EHA.	n/a	n/a	No events
29	Verify EVR arguments match format strings. E.g. no U8/U16s passed to %d.	n/a	n/a	No events
30	Verify all counter EHA channels pushed with initial values at module init or task preamble.	n/a	n/a	No telemetry channels



Svc::Deframer (Passive Component)

1. Introduction

`Svc::Deframer` is a passive component. It accepts as input a sequence of byte buffers, which typically come from a ground data system via a byte stream driver. It interprets the concatenated data of the buffers as a sequence of uplink frames. The uplink frames need not be aligned on the buffer boundaries, and each frame may span one or more buffers. `Deframer` extracts the frames from the sequence of buffers. For each complete frame *F* received, `Deframer` validates *F* and extracts a data packet from *F*. It sends the data packet to another component in the service layer, e.g., an instance of `Svc::CommandDispatcher`, `Svc::FileUplink`, or `Svc::GenericHub`.

F´ will formalize the built-in quality by meeting NASA Class-B flight software standards.

Roadmap: Cybersecurity

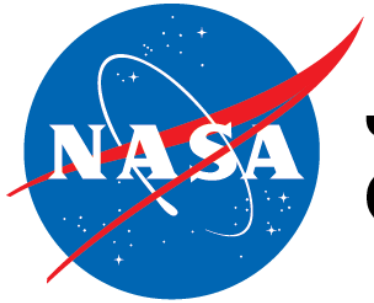
Bill of Materials

Uplink Encryption

Randomized Opcodes

Cybersecurity Standard Operating Procedure

F´ will bring safe cybersecurity practices to the forefront of the product line's development.



Jet Propulsion Laboratory
California Institute of Technology