PowerSat Custom F’ Components

Nayana Tiwari
SURP

CAL POLY

Purpose

This document serves to describe the input, output, and internals of each custom component as defined by accompanying documents that show the components and define the components.The connections between all components including F Prime provided components can be viewed using the Topology GitHub repository.

Revision History

Rev.	Date	Reason for Revision	Author
1.0	3/31/22	Creation and presentation to JPL	N. Tiwari
2.0	7/21/22	SURP redefinition. Consistent with PS-07-BBM_006 and _007	N. Tiwari
2.1	8/11/22	Edits from JPL review	N. Tiwari

Definitions

PowerSat Nomenclature

- Battery Management System (BMS)
- High Powered Board (HIPO)
- On Board Computer (OBC)
- Electrical Power System (EPS)
- Attitude Determination and Control System (ADCS)
- Spacecraft Bus/Avionics Stack: BMS, OBC, EPS, ADCS (not HIPO)
- Spacecraft (S/C)

FSW Nomenclature

- **FSW**
  - flight software, all software files on the spacecraft: Core, ADCS, Comms, and Payload SW.
- **Core SW** PF0
  - the files related to the health and status of the spacecraft bus
  - collects data from power and temperature sensors, monitors battery status
  - synchronizes the clock and checks flight images in memory for bit flips due to radiation
- **ADCS SW** PF2 PF3
  - the files related to collecting telemetry form gyroscope, accelerometer, magnetometer, and GPS
- **Comms SW** PF4
  - the files needed to uplink commands and downlink data (files, beacon, and telemetry)
  - process commands received from ground
- **Payload SW**
  - the files needed to manage the PowerSat specific payloads: Solar Array from DcubeD and accompanying PowerSat HIPO

Slide 6

PF0 Is there a more detailed list (or else) of functions considered?
Pauline Faure, 2022-07-22T16:12:50.798

NT0 0 its just the combination of the following four things. added the picture for clarity
Nayana Tiwari, 2022-07-25T15:56:51.105

PF1 What specifically? > Batt voltage, temperatures throughout s/c, ?
Pauline Faure, 2022-07-22T16:13:28.379

PF2 We need to define what telemetry means in our case too
Pauline Faure, 2022-07-22T16:13:53.583

PF3 add: (acceleration and angular velocity)
Pauline Faure, 2022-07-22T16:14:25.601

PF4 here too, data should be defined. For example, should downlink of TM be included or is TM included in data?
Pauline Faure, 2022-07-22T16:15:11.510

F Prime Definitions

- **Topology**
  - The graph of interconnected components and ports developed to produce a full functioning software architecture.
- **Component**
  - Each component is a discrete piece of the overall architecture and should have no code dependency on any other components.
    - **Passive**
      - A passive component has no thread and cannot support asynchronous commands or port invocations.
    - **Active**
      - A component has a thread of execution and a queue.
    - **Queued**
      - A queued component has a queue but no thread of execution. It can handle asynchronous commands and port invocations.
- **Port**
  - The components are connected through ports which are defined prior to creation of the component. Ports have a type and a kind. A port's type is defined by the port invocations' data type. A port's kind defines its directionality (in/out) and can be guarded, and synchronous or asynchronous.
    - **Guarded**
      - The port is limited to a single invocation at a time.
    - **Synchronous**
      - The call directly invokes functions without using a queue.
    - **Asynchronous**
      - The call is placed in a queue and dispatched on a thread.

From Caitlin's work, for examples and diagrams please reference PS-07-BBM_005

Slide 7

PF0 Examples for each, or where possible, could help further the understanding of what a component is
Pauline Faure, 2022-07-22T16:16:34.141

PF1 Here would help to have a diagram to accompany the definition (can move that whole part to another slide)
Pauline Faure, 2022-07-22T16:17:24.957

NT2 I'll add this to the powersat fsw definitions document so it is easy to reference
Nayana Tiwari, 2022-07-25T16:01:59.256

NT3

F Prime Definitions

- **Event (EVR)**
  - A type of downlinked data that represents a single event in the system. Events represent the history of the system and are defined per-component.
- **(Telemetry) Channel**
  - A single value read and downlinked. Channels represent the current system state.
- **Command**
  - Uplinked data items that instruct the system to perform an action. Commands are defined per-component.

PF0

PF2

Additional definitions especially of F' provided components can be found here.

From Caitlin's work, for examples and diagrams please reference PS-07-BBM_005

Slide 8

PF0 What is an event in the system? Here too, adding example could help for the understanding
Pauline Faure, 2022-07-22T16:18:26.164

PF1 For example...
Pauline Faure, 2022-07-22T16:19:06.558

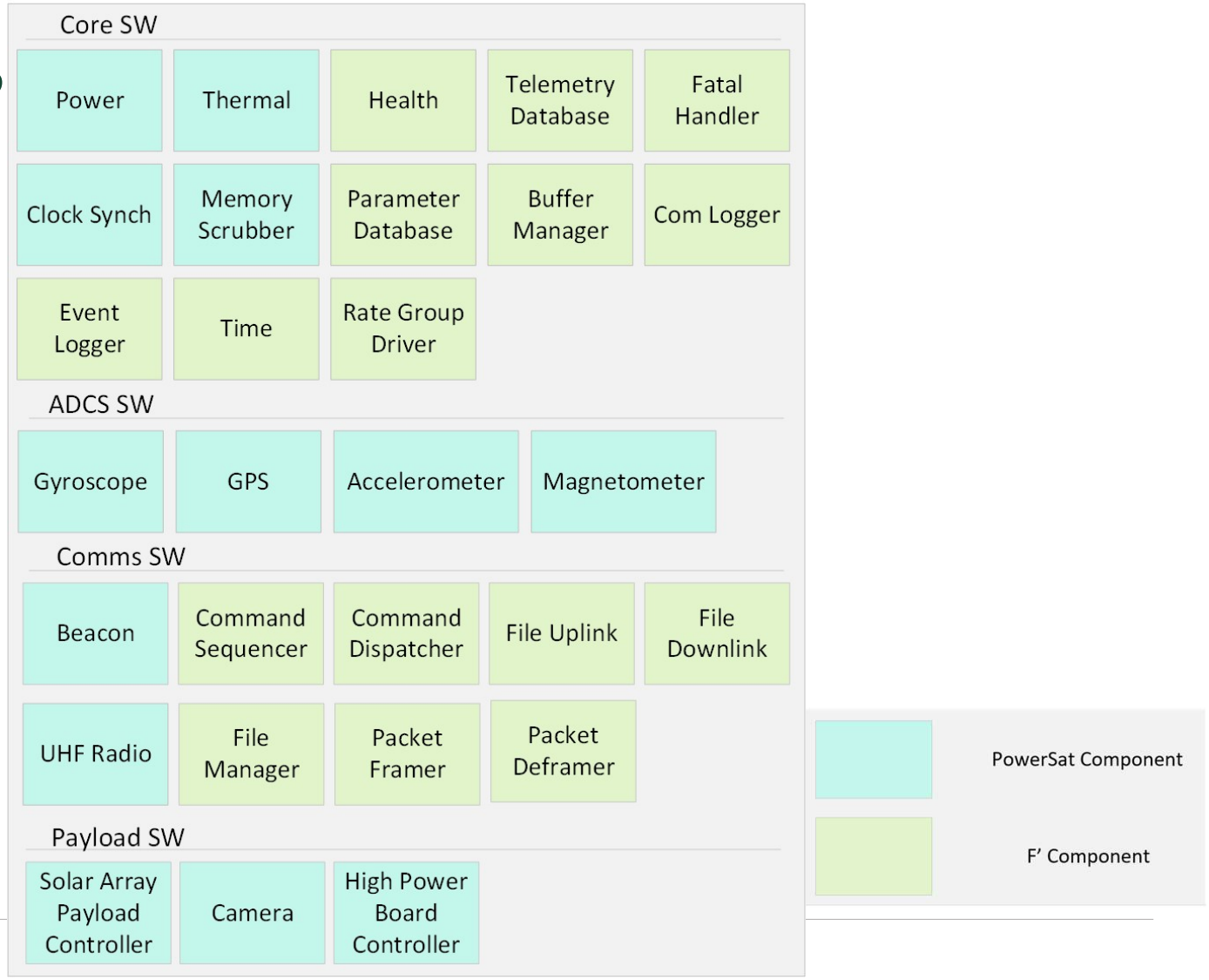
PF2 Might be useful to specify whether one command can be used for several components, that one component can have several commands (yes, the grammar of the sentence indicates it, but there is value in writing it clearly too). Likewise for event too
Pauline Faure, 2022-07-22T16:20:43.080

NT3 I'll add this to the powersat fsw definitions document so it is easy to reference
Nayana Tiwari, 2022-07-25T16:02:08.329

Components

Components

- Component visualization similar to Mars Helicopter visual from JPL



Custom Components

Core SW:

1. Power
2. Thermal
3. Clock Synch
4. Memory Scrubber

ADCS SW:

5. Accelerometer
6. Gyroscope
7. Magnetometer
8. GPS

Comms SW:

9. Beacon
10. UHF Radio

Payload SW:

11. Solar Array Payload
12. Camera
13. HIPO

Slide 11

PF0 When ppt first approved version is done, let's indicate the slide number to which each of those is defined in more details

Pauline Faure, 2022-07-22T16:22:11.009

Core SW Components

1. Power: Functionality

- Collects power data from all spacecraft bus power sensors, collects battery status information
- Must interface with the microcontroller on the BMS board
- Must interface with all power sensors on the spacecraft bus (not HIPO)
- Must record all data and timestamp
- Must report faults to Event Logger and Fatal Handler

1. Power: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **Core Active Rate Group**
- Command to check the power subsystem asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**
- Nominal bounds received from **Parameter Database**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

1. Power: Internals

- Connection to all INA219 power sensors on spacecraft bus. Able to read from and convert reading to sensible units. Reads all power sensors in succession when the Rate Group is scheduled.
- Connection to BMS microcontroller. Requires register access or a UART-reported status packet to receive sensor data and battery information.
- Regular telemetry, command, and event connections.
- Additional events to support the reporting of battery faults

2. Thermal: Functionality

- Collects temperature data from spacecraft bus
- TBD: Check temperatures against nominal expectations. Activates heaters if needed.
  - Waiting on thermal subsystem team to decide on heater placement if needed

2. Thermal: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **Core Active Rate Group**
- Command to check the thermal subsystem asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**
- Nominal bounds received from **Parameter Database**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

2. Thermal: Internals

- Connection to all temperature sensors on spacecraft bus. Able to read from and convert reading to sensible units. Reads all temperature sensors in succession when the Rate Group is scheduled.
- Regular telemetry, command, and event connections.
- Additional events to support the reporting of temperature faults

3. Clock Synch: Functionality

- Gets time from GPS
- Re-calibrates the Real Time Clock (RTC) on the OBC to the GPS

3. Clock Synch: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **Core Active Rate Group**
- Command to synchronize the clock received asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**
- New calibrated time received from **GPS**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Command response and registration to **Command Dispatcher**

3. Clock Synch: Internals

- Must receive time from GPS (could also pull from the telemetry database)
- - Use the offset between the Time input port and the telemetry database to offset the GOS recorded time if needed (depends on required precision)
- Must then use the GPS time to update the RTC on the OBC
- Again consider needed precision and the delay of this process
- May need to modify given Time component to create the synchronization

4. Memory Scrubber: Functionality

- Performs a checksum on all copies of FSW
- Overwrites images that have been deemed corrupted

4. Memory Scrubber: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **Core Active Rate Group**
- Command to run the memory scrubber or stop scrubbing asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

4. Memory Scrubber: Internals

- Must read OBC flash memory for FSW images
- Must be able to overwrite FSW images (may require a Buffer Manager connection)
- Must be able to report faults in overwriting memory to Event Logger/Fatal Handler. If the S/C is reset with a faulty overwrite, the mission may be in danger.

ADCS SW Components

5. Accelerometer: Functionality

- Collects acceleration data. Possibly from multiple accelerometers.
- Must record all data and timestamp
- Must report to Event Logger

5. Accelerometer: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **ADCS Active Rate Group**
- Command to check the accelerometers asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

5. Accelerometer: Internals

- Connection to accelerometer(s) and necessary driver functions implemented. Able to read from and convert reading to sensible units. Reads all accelerometer(s) in succession when the Rate Group is scheduled.
- Regular telemetry, command, and event connections.

6. Gyroscope: Functionality

- Collects gyroscope data. Possibly from multiple gyroscopes.
- Must record all data and timestamp
- Must report to Event Logger

6. Gyroscope: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **ADCS Active Rate Group**
- Command to check the gyroscopes asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

6. Gyroscope: Internals

- Connection to gyroscope(s) and necessary driver functions implemented. Able to read from and convert reading to sensible units. Reads all gyroscope(s) in succession when the Rate Group is scheduled.
- Regular telemetry, command, and event connections.

7. Magnetometer: Functionality

- Collects magnetometer data. Possibly from multiple magnetometer(s).
- Must record all data and timestamp
- Must report to Event Logger

7. Magnetometer: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **ADCS Active Rate Group**
- Command to check the magnetometers asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

7. Magnetometer: Internals

- Connection to magnetometer(s) and necessary driver functions implemented. Able to read from and convert reading to sensible units. Reads all magnetometer(s) in succession when the Rate Group is scheduled.
- Regular telemetry, command, and event connections.

8. GPS: Functionality

- Collects GPS data frame
- Must record all data from frame and timestamp
- Must report to Event Logger

8. GPS: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **ADCS Active Rate Group**
- Command to startup GPS and read a frame asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**

Output Ports:

- Time output from frame to **Clock Synch**
- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

8. GPS: Internals

- Has a startup time of 20-30ish seconds. Must be integrated into a state machine.
- Frame is read from GPS. Necessary information is extracted and stored in telemetry database. [TODO reference ADCS document]
- Regular telemetry, command, and event connections.

Comms

9. Beacon: Functionality

- Gets data from telemetry database based on predetermined format
- Compresses data as needed to form a small enough packet (typically below 230 bytes)
- Sends packet to packet framer

9. Beacon: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **Comms Active Rate Group**
- Command to check the power subsystem asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**
- Telemetry input from **Telemetry Database**

Output Ports:

- Packet sent to **Packet Framer**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

9. Beacon: Internals

- Requests data from telemetry database
- Ensures data is somewhat up to date
- Performs some compression to minimize packet size. As needed.
- Perhaps requires a connection to buffer manager however the packet should not be that large.

10. UHF Radio: Functionality

- Uses radio chip's SPI interface to send/receive data
- Takes in a framed packet and sends over radio
- Receives data from radio and sends to deframer

10. UHF Radio: Ports

- Active Component to handle RX interrupts.

Input Ports:

- Scheduling from **Comms Active Rate Group**
- Commands to control the UHF radio asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**
- Framed packet to downlink from **Packet Framer**

Output Ports:

- Buffer allocation and deallocation with **Buffer Manager**
- Uplinked packet to **Packet Deframer**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

10. UHF Radio: Internals

- Mostly a driver to interface with the UHF radio chip
- Decisions to be made by systems engineering about when to RX/TX, how the interrupt pin will be used, etc.
- Likely will take the form of a state machine. Comms has mentioned an internal state machine in the UHF chip docs. [TODO add comms documentation]
- Regular telemetry, command, and event connections.

Payload

11. Solar Array : Functionality

- Collects sensor data from DCubeD array from an ADC or digital from DCubeD.
- Saves sensor data.
- Number of sensor, type of sensors, type of serial unclear as of now. Waiting on an ICD

11. Solar Array: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **Payload Active Rate Group**
- Command to check the accelerometers asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Command response and registration to **Command Dispatcher**

11. Solar Array: Internals

- Will need sensor drivers.

- Sensor data must be collected when the rate group is scheduled. Must be converted into sensical units and stored in telemetry.

- Regular telemetry, command, and event connections.

12. Camera: Functionality

- Takes pictures on 3 axes (3 cameras) to capture solar array deployment
- Provide image compression

12. Camera: Ports

- Passivecomponent that is commanded

Input Ports:

- Command to take images in succession asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**

Output Ports:

- Telemetry output (such as number of images taken) to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**
- Possible connection to **Buffer Manager** for image saving

12. Camera: Internals

- Cameras must be active during solar array deployment. Images must be taken within a minimal threshold to impose the axes into one almost 360 degree photo.
- Regular telemetry, command, and event connections.
- Likely needs some fault response

13. HIPO: Functionality

- Collects sensor data
- Stores sensor data
- Controls duty cycle of high powered LED

13. HIPO: Ports

- Passivecomponent connected to an Active Rate Group and commands

Input Ports:

- Scheduling from **Payload Active Rate Group**
- Command to check the accelerometers asynchronously from **Command Dispatcher**
- Time input for timestamping from **Time**
- Duty cycle and nominal bounds input from **Parameter Database**

Output Ports:

- Telemetry output to **Telemetry Database**
- Nominal event output to **Event Logger**
- Fault event output to **Event Logger** (then connects to **Fatal Handler**)
- Command response and registration to **Command Dispatcher**

13. HIPO: Internals

- Connection to sensors. Converts data to sensical units and stores in database. Requires low level drivers for sensors.
- Control LED duty cycle. Unclear whether the S/C will control this autonomously or if it will be a ground command. If autonomous, requires knowledge of pointing/location/time to avoid LED usage during the night.
- Regular telemetry, command, and event connections.
- Likely needs some fault response