DraMS Overview

Simulators

Several applications have been created to simulate various parts of the system, and these simulators can be put together in various ways. Portions can even be swapped out for actual hardware when necessary.

The major components (applications) are:

- CFSIM
- DramsDevSim
- DramsDevSimClient
- HiDra
- LanderSim

CFSIM

The ColdFire Simulator (CFSIM) is a software emulator that can run a Flight Software (FSW) image or a bootloader image. It has been built to simulate the MOMA Command, Control & Data Handling (CCDH) board. DraMS is using the MOMA board for initial development of the FSW until a DraMS board is available. An ancillary application (qterm) is also included that can send/receive data from the Debug UART through a terminal-like interface.

The CFSIM has a Spacewire (CMD/TLM) interface (TCP ports 27014 and 27015) and can connect to "peripherals" (TCP ports 4000-4003). There is also a Debug UART that is connected to a named pipe ("/dev/uart"). These peripherals are currently mapped as follows:

- LASER (port 4000)
- GC (port 4001) (NOTE: Used for the VH in DramsDevSim)
- MAIF (port 4002)
- SEB (port 4003) (NOTE: Used for the CTL in DramsDevSim)

Git repo

DramsDevSim

This application is a collection of simulators for the LEU (Laser), Valve/Heaters (VH), Pressure/Temp (PT), and MS (Mass Spectrometer) instrument boards. These can be connected via a serial or TCP connection to either the CFSIM or to an actual C&DH. These simulators can receive commands and generate telemetry like the actual hardware.

- Valve/Heater (VHSim)
- Laser (LaserSim)
- Pressure/Temperature (PTSim) - Planned
- ITMS (MSSim) - Planned
- MMC (MMCSim) - Planned

Git repo

DramsDevSimClient

This applications was created to test the various simulators included in the DramsDevSim. Users can select from one of the "client" types: Laser (LEU), Valve/Heater (VH), and Mass Spec (CTL) and send commands directly to the simulators. The generated packets can also be useful for creating tests for the various instruments.

Git repo

HiDra (HI DRAgonfly)

This application is an early version of the DramsGSe that provides basic communication with the CCDH. It connects directly to the Spacewire ports (27014/27015) and can send and receive data directly to the CCDH. This application was built for development of the FSW and is used mostly by the FSW developer (Micah Johnson) or for FSW testing (Raj Roy).

Git repo

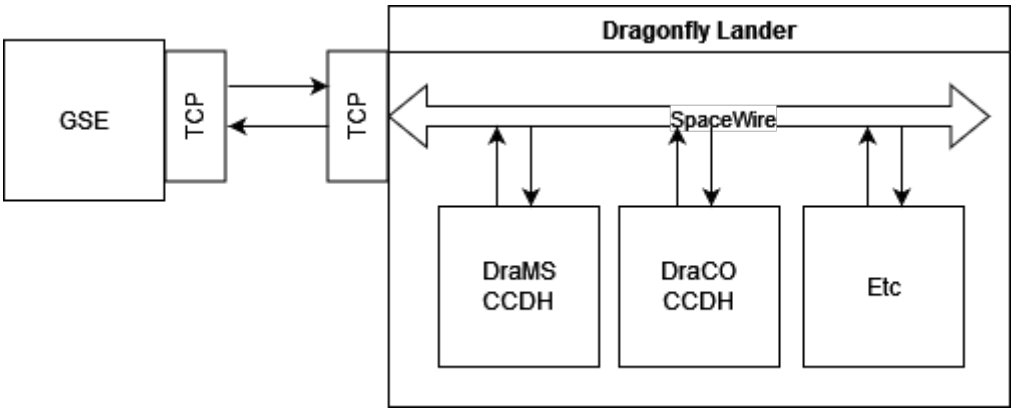
LanderSim

The LanderSim receives ITF packets from the ground (DramsGSE) and routes them to the Spacewire Bus (and this to the Drams C&DH). Spacewire telemetry packets received from the instrument(s) (i.e., DraMS CCDH) are sent to the ground as ITF packets. The LanderSim also generates a once-per-second Time & Status packet that is sent to all instruments.

Git repo

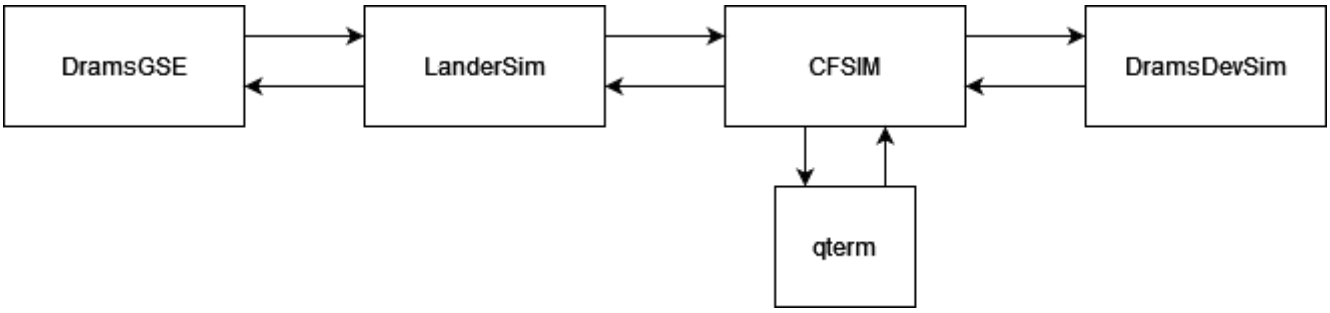
Logical Layout

The simulators above can be set up in several different configurations, and paired with actual hardware where possible. The basic layout is shown below. The GSE will communicate with the Dragonfly Lander via a TCP socket connection, which will in turn pass packets along to the internal instruments (DRAMS, DRACO, etc) via Spacewire.

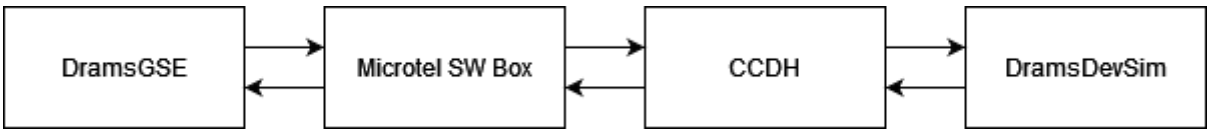


The SpaceWire connections can be simulated in software or replaced with a dedicated SpaceWire device (either the "Microtel box" or the "STAR-Dundee Brick").

Full System Layout (All software)



Hybrid System Layout (Hardware and software)

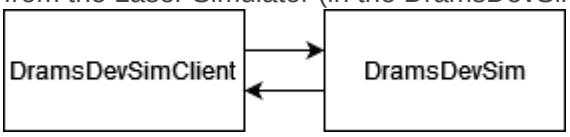


NOTE: the "Microtel SW Box" and the CCDH are both physical hardware.

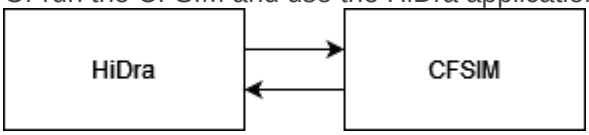
Subsystem Setups

Smaller configurations are possible for testing individual components.

For example, one can run the LaserClient application to send command packets and receive telemetry directly from the Laser Simulator (in the DramsDevSim):



Or run the CFSIM and use the HiDra application to send/receive packets directly:



Commanding

- Needs to support sending FSW file system commands
- Needs to support loading and running BASIC scripts

Questions:

- How should commands be defined?

Telemetry

- dramsgse should store the received telemetry into test directory archives (TID)
- Telemetry mnemonics will be defined via an Excel sheet and then exported to CSV text files
- The most recent telemetry menmonic values (CVT) should be accessible from BASIC

Interfaces

- What main interface does dramsgse need to support?
- Does dramsgse need to support any custom interfaces?

Viewing Data

- dramsdataview
  - Provides custom views of the data

Revision #13
Created 21 March 2022 22:05:18 by Bradley Tse
Updated 25 March 2024 12:33:46 by Michael Burkhart