HuskySat-1: CDH Requirements

Definitions:

* CE: compute element, e.g. an MSP430, MSP432, or Raspberry Pi
* CS: core software – shared software that is common to all or most CE’s on the spacecraft
* CECS: compute element + core software

**NOTE**: These requirements pertain to the general compute hardware and software environment on board the spacecraft, and all functional and performance targets are generic and apply UNLESS overridden at the subsystem level (e.g. state estimation subsystem module has considerably stricter timing-determination goals than the rest of the s/c). Overrides will be specified in the individual subsystem requirements.

1. CECS must be capable of best-case operation under all **power conditions**.
   1. CECS’s – including all required peripherals – must be able to run properly at the nominal voltage of **3.3V.**
      1. Properly is defined as a) maintaining nominal MSP43x clock speeds (**8Mhz** for 430, **48Mhz** for 432) and b) not incurring any brown-out power-up reset events on the MSP chips, assuming nominal noise levels on the power rail.
   2. CECS’s must at all times **retain sufficient (but ONLY sufficient) information to rebuild state in the event of unexpected/unplanned “sudden”** (= no warning, no time to store state) power loss.
      1. Examples include some (though not all) parameters uploaded from the ground to set/change configuration on-orbit
      2. MET counts (see time requirement category below
      3. Each subsystem will have its own definition for what is “sufficient state”
   3. CECS must **not retain any state that would disallow the use of manual/intentional power cycling** as a means to return the CE to a “known good” state. In other words, the total state of the subsystem needs to be rebuilt from observations plus a minimal state of previous state information that is saved (e.g. in NVRAM).
2. CECS must **properly handle time-based data and functionality** – including calculating and storing time, measuring events occurring over time, etc. – with minimal error (which varies depending on specific use case)
   1. Each subsystem should have its main timer source(s) configured – both electrically in terms of supporting oscillators/xtals and in software, in terms of clock peripheral settings – such that clock jitter is less than **+/-5 microseconds**
   2. All subsystems must be able to keep track of elapsed time to within a certain delta of all other subsystem time-keeping mechanisms, where delta must not exceed a drift of **+/-1 millisecond/second**
   3. All subsystems must have **access to two externally-tracked time sources at all possible times**: true time and MET (mission-elapsed time):
      1. True time (TT) will be sourced by the GPS sensor, and will have an accuracy of **+/-10 microseconds** (note that GPS itself is more accurate than this, but delays in the information propagation architecture (most notably CAN) degrade this somewhat)
      2. Mission-elapsed time (MET) will be tracked primarily by EPS, as the longest-lived subsystem CE. Proper state must be stored in a non-volatile location to ensure that EPS can rebuild this counter throughout power events, with a loss of no more than **+/-1 second/24h**
      3. In the event of a CE reboot, CS will initially start without either time source. his must be rectified – i.e. TT and MET need to be determined – within
         1. 30s for TT
         2. 2s for MET
   4. Internal timers used
   5. GPS clock and propagating that throughout s/c?
   6. MET?
   7. Stable enough throughout temp range
   8. Milliseconds vs microseconds?
3. CECS must allow reliable, timely exchange of information, both between devices within a module/subsystem, as well as throughout the s/c.
   1. CAN bus stuff …
   2. I2C bus stuff
   3. UART stuff …
   4. Noisy environments – PPT craziness?
4. CECS must be built in a manner that allows for maximun reliability
   1. Toolset freezes
   2. Debug backchannel
   3. TESTABILITY
5. CECS must work cleanly together with the larger system that forms the Ground Segment.
   1. COSMOS-y stuff here
   2. Integration stuff/flat sat as well?
6. Others:
   1. Other hardware (bsp hardware redirection)
   2. Power consumption?