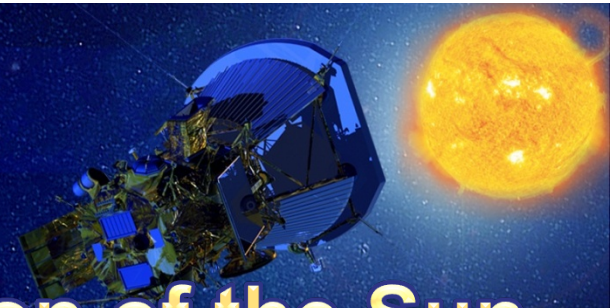
Solar Probe Plus

A NASA Mission to Touch the Sun



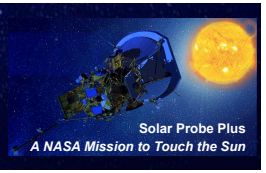
Integrated Science Investigation of the Sun
Energetic Particles

Preliminary Design Review
05 – 06 NOV 2013

EPI-Hi Flight Software

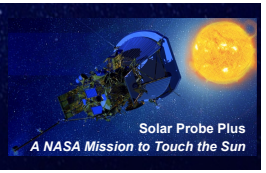
Andrew Davis, Rick Cook

Outline



- Software requirements, software design and functionality, software documentation and unit testing as well as plans for software integration, testing and verification at the instrument level
- Plans for software maintenance for the duration of the mission
- Description of the software development environment and plans for preserving the development environment
- Maturity of the design, heritage, what's new/different from previous missions
- Projected CPU, SRAM, MRAM margins
- Backup slides on Boot sequence

Software Requirements



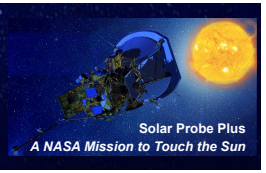
- The Flight Software Requirements Document details the flow-down from higher-level requirements, and from GIS and SIS ICDs
- These requirements are quite similar to those for our work on recent NuSTAR and STEREO missions
- The same modular flight processor and software architecture that we used for NuSTAR and STEREO will meet the EPI-Hi requirements
- The requirements are mapped into a set of modular software tasks, implemented on the EPI-Hi flight processors

Documentation Status

Document	Status/due date
Flight Software Requirements	Baseline draft Delivered
Flight Software Development Plan	Baseline draft Delivered
Flight Software Design Document	Prelim draft delivered at PDR
Flight Software Test Plan	Due by CDR
Test verification matrix	Due by CDR
Instrument Telemetry Data Format Document	Due by CDR
P24 MISC Processor Manual	Delivered
PHASIC User Manual	Delivered

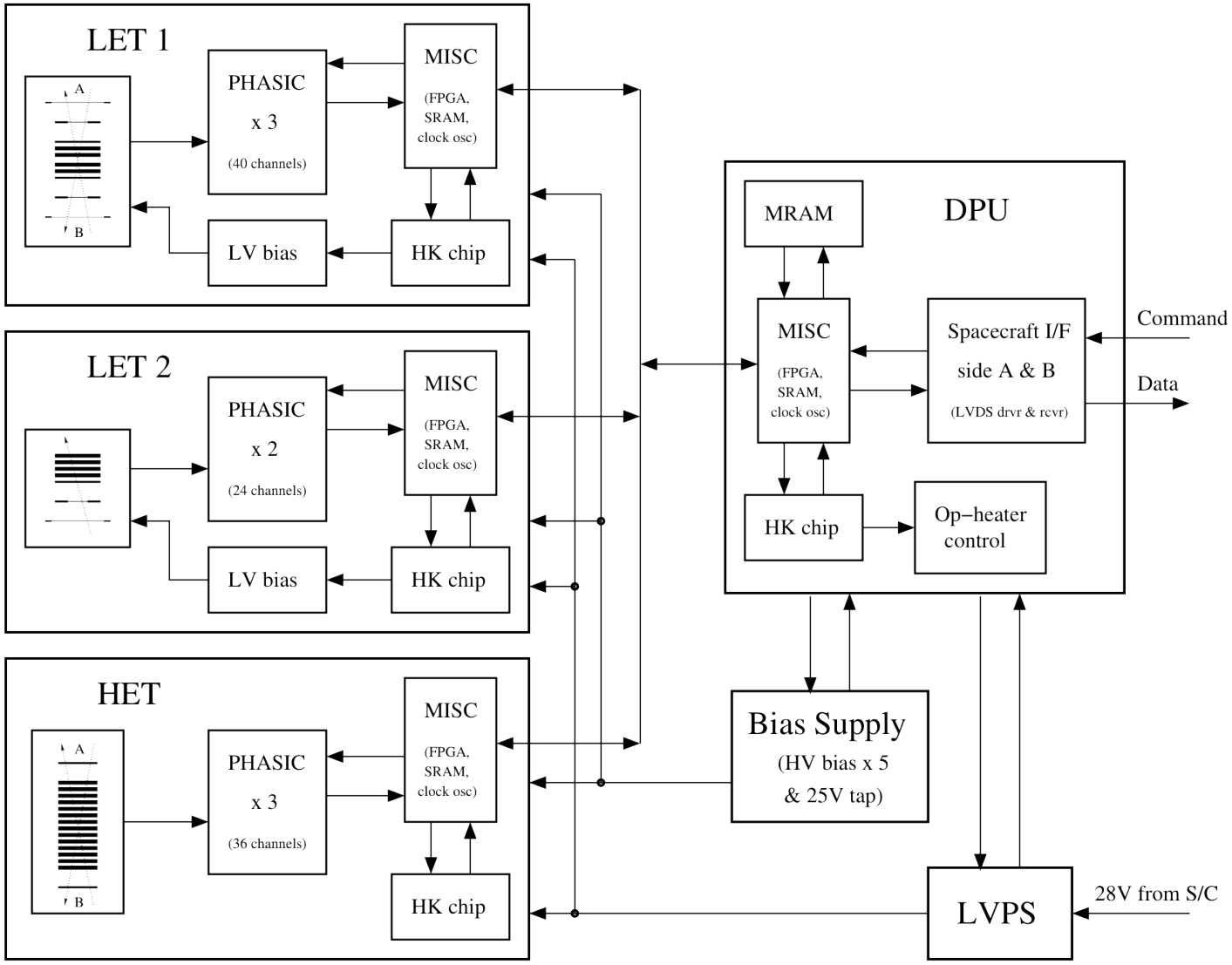
These docs are responsive to the GI ICD, the SI ICD, the PAIP, L4 reqs…

Processor and Software Architecture

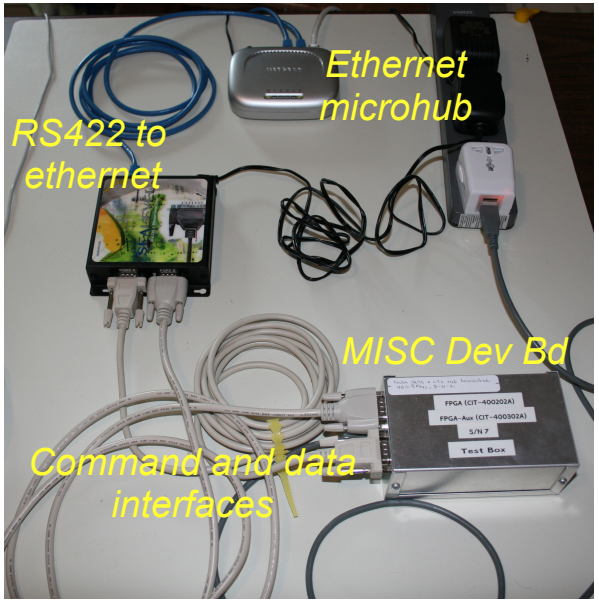


- Software distributed across 4 Minimal Instruction Set Computers (MISCs)
- 1 MISC in DPU, 1 each in LET1, LET2, HET
- Custom FPGA-embedded micro-controller
- Design implemented in Actel RTAX250SL
- Architecture used in STEREO and NuSTAR space missions
- 24-bit CPU, up to 512 Kwords SRAM, Forth operating system, can boot from MRAM or serial interface
- MISC design and FORTH operating system stable since 2002
- Only DPU MISC has MRAM (2M×8).
- DPU can boot from MRAM, or via serial uplink from ground
- The three “satellite” MISCs communicate with DPU MISC via RS-422 serial links, 460.8kbps
- Satellite MISCs booted via serial link by DPU MISC
- Satellite MISCs use boot images from DPU MRAM

Processor and Software Architecture



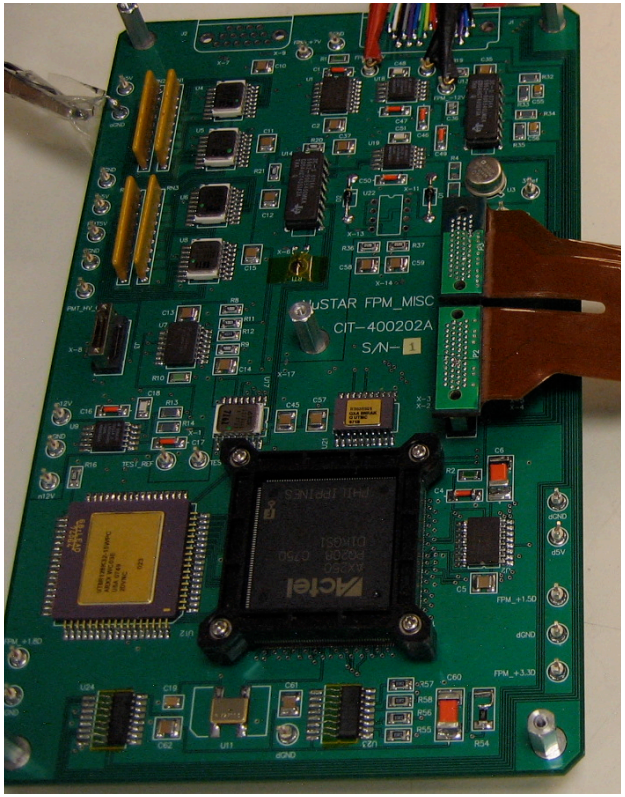
Development Environment



Interface to MISC Development Board

- Initial testing of new code takes place on MISC development boards
- Forth system on development boards accessed via serial link to PC
- New code is easily uploaded via the serial link
- Testing of code that accesses hardware is done on EM units
- Low-level MISC I/O routines are stable, and in everyday lab use, where MISCs are used to acquire test data and control test runs
- The same central MISC / multiple-satellite MISC architecture was used for NuSTAR and STEREO
- EM units are preserved after launch, and used for testing/verification of software uploads and command sequences

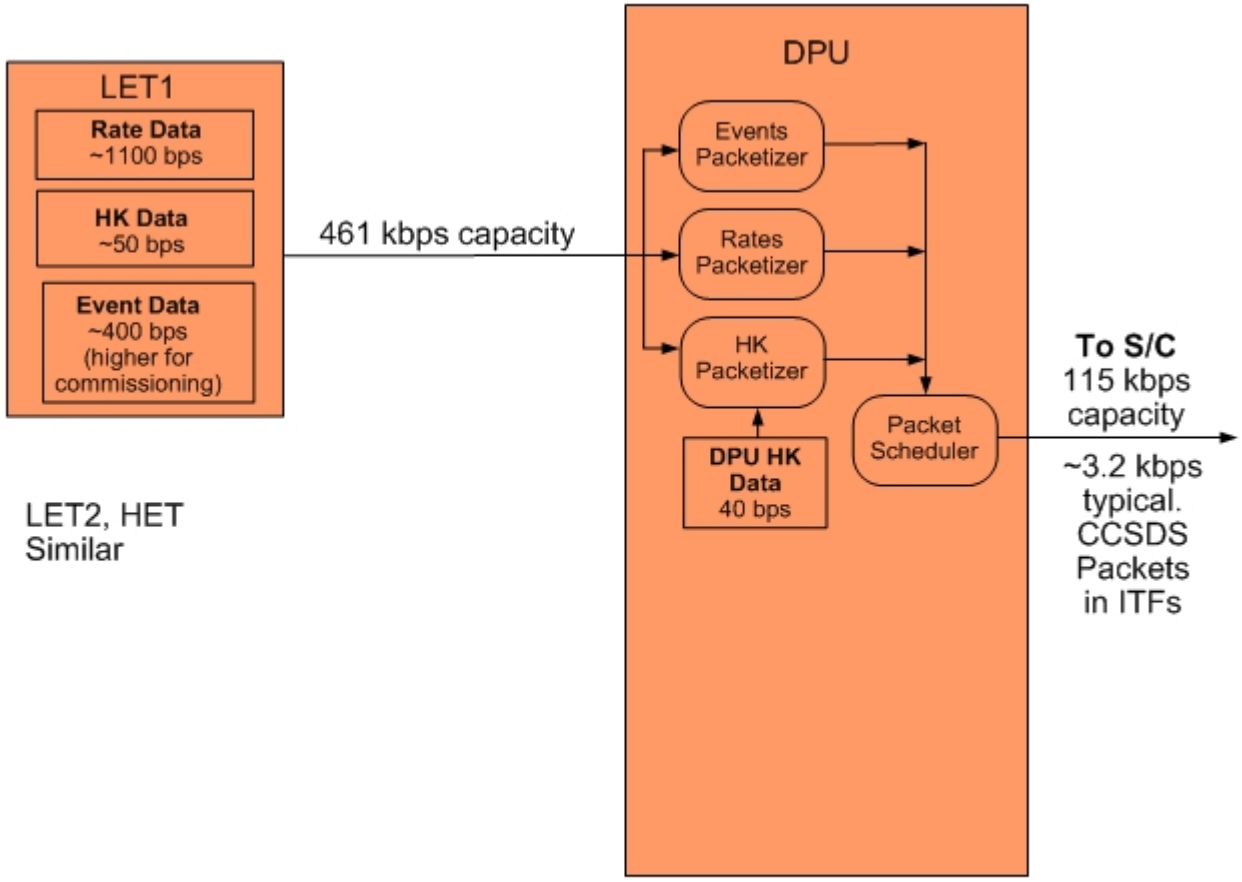
Development Environment



NuSTAR MISC EM board

- Initial testing of new code takes place on MISC development boards
- Forth system on development boards accessed via serial link to PC
- New code is easily uploaded via the serial link
- Testing of code that accesses hardware is done on EM units
- Low-level MISC I/O routines are in everyday lab use, where MISCs are used to acquire test data and control test runs
- The same central MISC / multiple satellite MISC architecture was used for NuSTAR and STEREO
- EM units are preserved after launch, and used for testing/verification of software uploads and command sequences

Instrument Data Flow

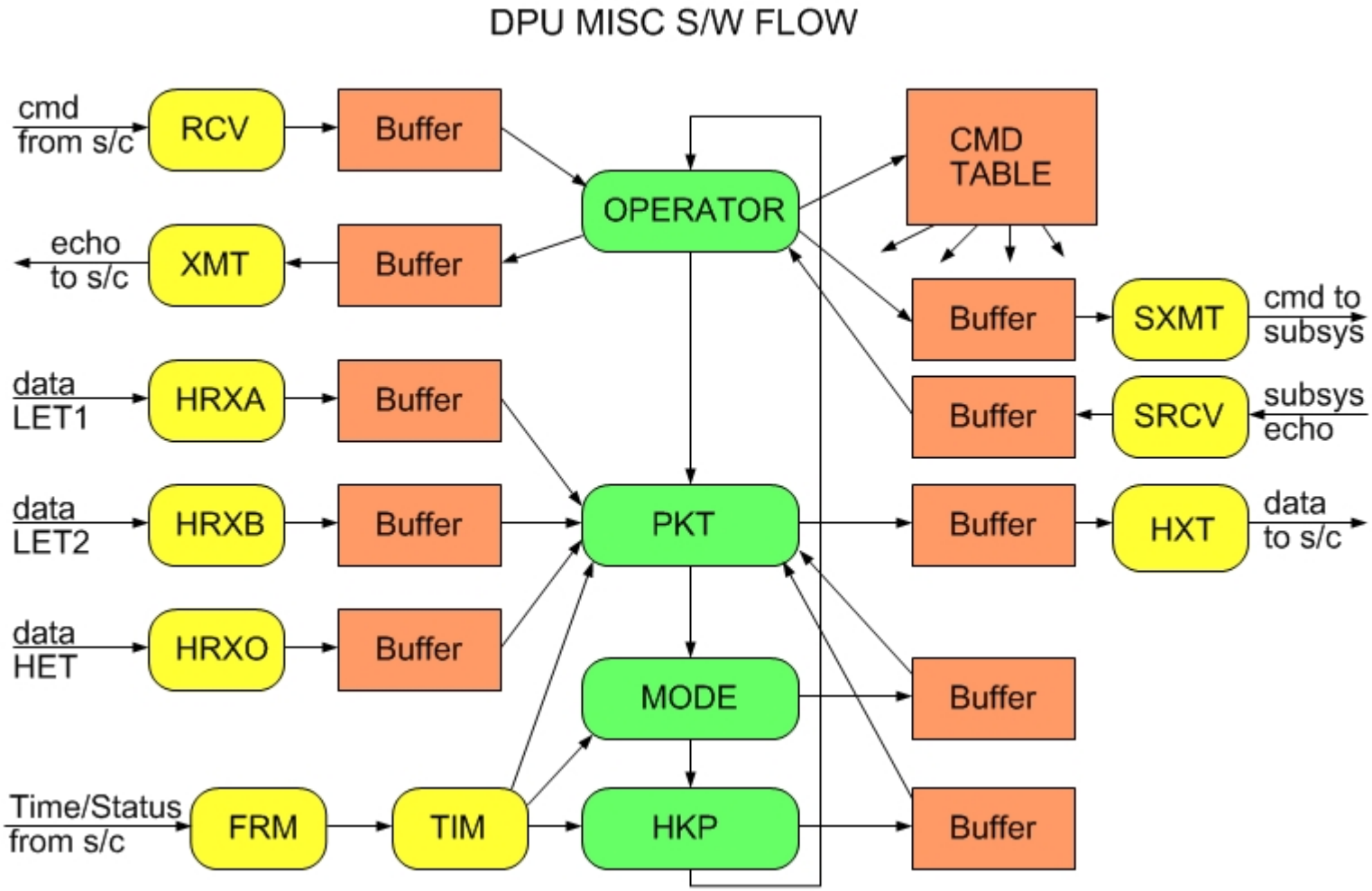


Note: Typical data rates are preliminary, and subject to change based on our SSR allocation

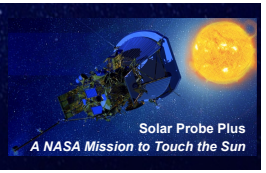
DPU Software Tasks

- Forth operating system and low-level I/O routines
- Power-on and initialization sequence management
- Satellite MISC serial boot sequence management
- Housekeeping data collection
- Inter-MISC communication management, and data acquisition/buffering from satellite MISCs
- Setup and control of instrument LVPS, bias supply, op heaters
- Receive/Monitor status and time-synchronization data from the spacecraft, and perform autonomous mode adjustments as needed
- Management of software uploads and MRAM “burns”
- Formatting/packetizing and transfer of science & housekeeping data and command responses to the Spacecraft
- Monitor heartbeat from peripheral MISCs, and perform autonomous diagnostics/reboot as needed
- Manage volume of instrument data transferred to SSRs on S/C as function of time

Software Flow – DPU MISC

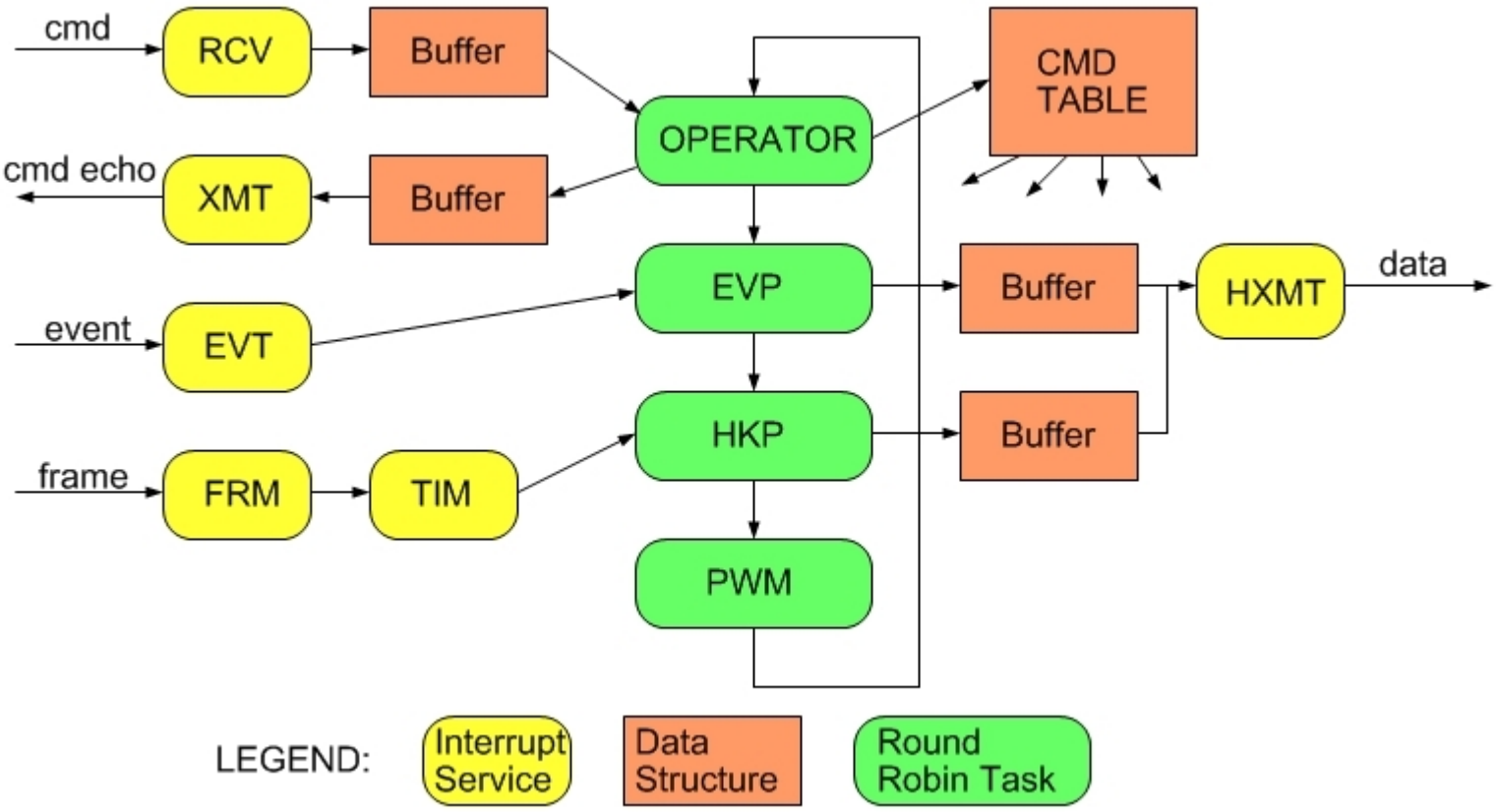


Satellite MISC Software Tasks

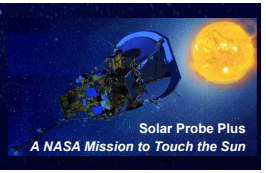


- Forth operating system and low-level I/O routines
- Science data acquisition
- Science data processing and reduction (particle ID)
- Housekeeping data acquisition
- Processing of status, time-synchronization, and command data from the EPI-Hi DPU
- Monitor key counting rates and adjust the telescope operating condition to optimize data quality
- Monitor temperatures and adjust detector gain/offset settings to compensate for temperature variations
- Formatting and transfer of science & housekeeping data and command responses to the EPI-Hi DPU
- Setup and control of PHASICs and HK chips, and any instrument HV, bias supply, heaters, etc. not controlled by the EPI-Hi DPU.
- Dynamic adjustment of detector thresholds during periods of high particle intensity

Software Flow – Detector Module MISC



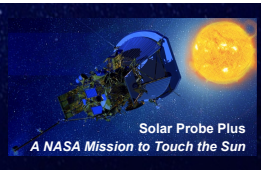
Integration, Testing and Verification



The EPI-Hi Flight Software Test Plan document will define this plan in detail. This plan will include a software requirements verification matrix.

- Software tests will start at the module level. As the code builds up, system-level testing will start, and the majority of the test time will be targeted at the system level.
- A Test Readiness Review will precede formal acceptance testing. SPP Project and ISIS personnel will oversee/participate in this review.
- Formal acceptance tests will be performed on the EPI-Hi instrument, including the flight software, during the EPI-Hi integration and test phase.
- During environmental tests and SPP I&T, more experience will be gained with the flight software, real sensor data, and with controlling the instrument via the SOC-MOC interfaces. Flight software changes may be expected as a result. Any changes in the flight software at this stage will result in CCB review, and a repeat of these acceptance tests.
- The acceptance tests for the EPI-Hi flight software will be designed to verify each of the software functional requirements as called out in the requirements documents and also the functions provided by the Forth operating system, the I/O API, and the multitasking software running on the MISC processors.
- A software requirements verification matrix will be created and maintained by the EPI-Hi team to aid in verifying that the software meets the requirements. The matrix will be available for review by the SPP Project. A preliminary software requirements verification matrix will be presented at CDR and a final version will be presented at the Software Acceptance Review, following successful completion of the acceptance tests.
- A Software Acceptance Review will follow the successful completion of the acceptance tests, and the EPI-Hi team will present a test report. SPP Project will oversee/participate in this review.

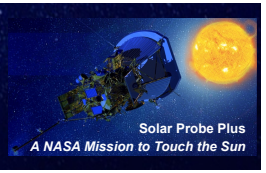
Software Maintenance, Configuration Management



Configuration Management Plan is contained in Software Development Plan

- Version control and software archiving: achieved via Subversion version control system (SVN) (http://subversion.apache.org/). The subversion server is hosted and maintained by Caltech Space Radiation Lab IT personnel. It is access-controlled for security, and is backed-up regularly.
- The lead engineer will maintain the structure of the SVN repository, including separate directories for each of the MISC's flight software, separate directories for different software builds, etc.
- All problems and change requests will be documented/tracked in the Software Development Log by the lead engineer. Given such a small dev team, a change request tool such as JIRA is an unnecessary overhead.
- After the beginning of I&T with flight hardware, all flight software changes will be approved by a CCB prior to being loaded into flight hardware. Also after this point, Version Description Documents (VDDs) will accompany all Software releases. The VDD will contain the functionality of the release, a list of closed software problem reports, a list of any liens/workarounds, installation instructions, and a list of deliverable source code files.
- The CCB membership will consist of the EPI-Hi lead engineer and software developer, and the Epi-Hi Project Manager, plus any other ISIS or SPP Project engineers that the QA team at SWRI deems necessary.

Software Heritage



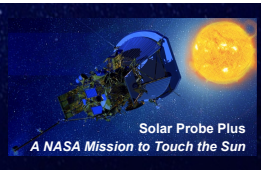
The flight software inherits some major components from previous projects:

- The Forth kernel and real-time multi-tasking S/W are the same as for the SEP instrument suite on STEREO, the NuSTAR mission, and the many MISC processors used ubiquitously in ground test equipment at Caltech
- The inter-MISC communication S/W is the same as for STEREO & NuSTAR
- The software for packetizing instrument data into CCSDS packets and scheduling the transfer of data packets to the spacecraft is the same as for STEREO & NuSTAR
- The software for processing instrument commands, routing commands to detector module MISCs, and packetizing command responses is the same as for STEREO & NuSTAR, except for the unpacking of the commands from the telecommand ITF.
- The onboard analysis and binning of particle event data by the detector module MISCs has significant heritage from STEREO

Software Heritage – What's Different?

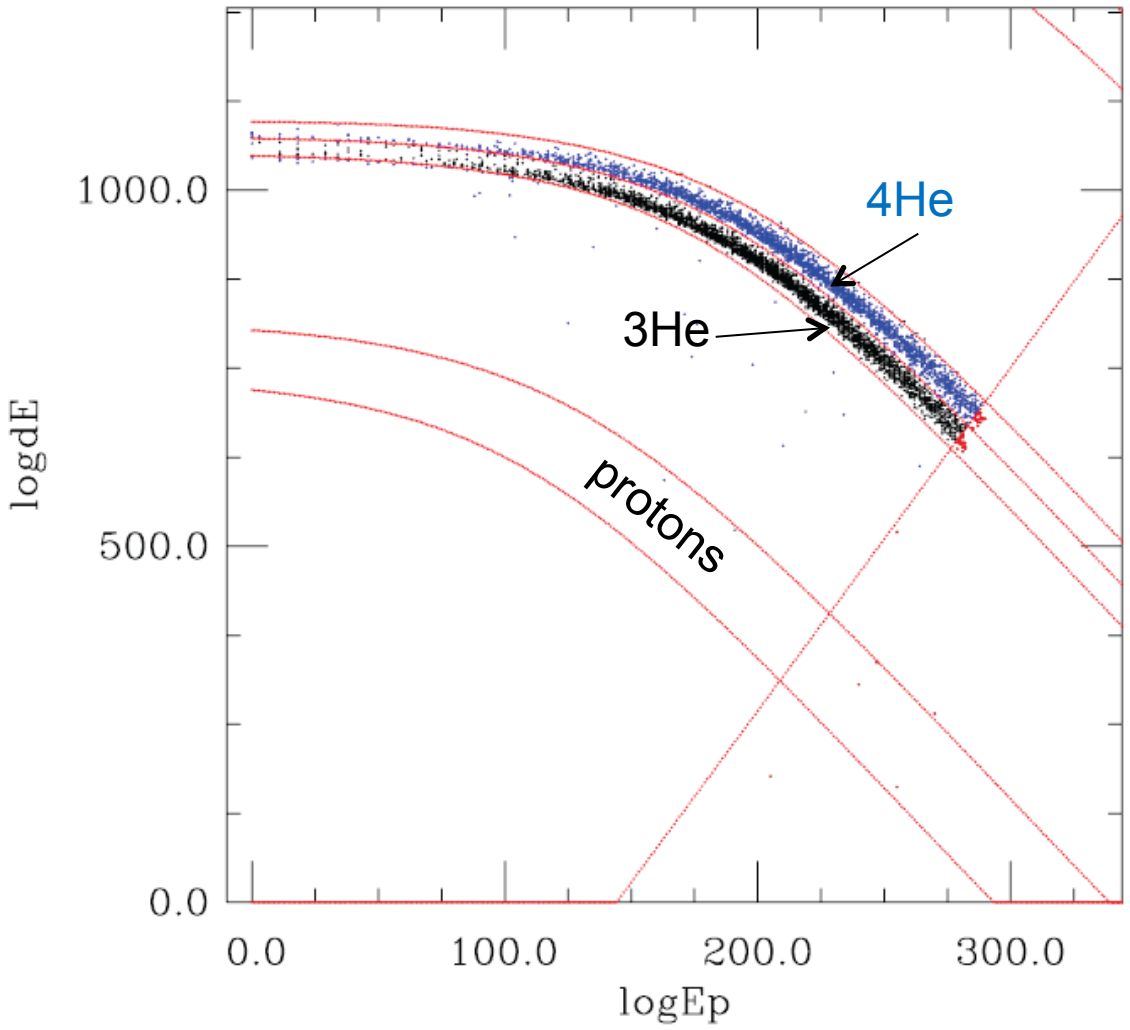
- The instrument boot sequence is quite different, given the lack of hardware reset line and the increased autonomy requirements – see separate boot sequence presentation.
- Each MISC FSW shall provide a capability to execute command sequences, i.e. macros. This is actually inherent in the Forth OS, but we need to write some utility code to make the implementations of timed command sequences simple/friendly
- The DPU MISC FSW needs to manage a section of MRAM as a "patch file" archive. Need subroutines to transfer between SRAM cmd buffer and the MRAM patch file, and routines to read and execute the patch file. This capability needed to support autonomous reconfiguration to known op state after power-on/reset.
- The DPU shall monitor the telemetry of the detector module MISCs, and perform an SRAM dump and reboot upon detection of a fault.
- Need to understand interaction between EPI-Hi autonomy rules and S/C autonomy rules for all possible fault scenarios

Software Heritage – What's Different? (2)



- Management of the volume of EPI-Hi instrument data telemetered to the Spacecraft as a function of time, based on the Solid-State-Recorder "fill-level" information in the Spacecraft status message, and other commandable parameters
- Extraction of Forth instrument commands from the telecommand ITFs. In previous missions, the S/C unpacked our commands from ITFs/packets for us.
- Although the onboard analysis and binning of particle event data by the detector module MISCs has significant heritage from STEREO, there are new challenges:
  - We want to do a better job on 3He/4He than on STEREO
  - The coincidence equations that operate on the signals generated by each incoming particle event are more complex than on STEREO

Onboard Particle ID – He Isotope Separation



- Output from a LET1 Monte Carlo simulation of 3He and 4He events was used as input to the latest version of our onboard particle ID algorithm, implemented on a MISC test board
- Onboard algorithm maps events into a 2D dE-Eprime space, where boundary-curves define different regions for different species
- Code currently processes events at ~ 9600 events/ second, ~4x faster than max incoming rate

Predicted memory, CPU Margins

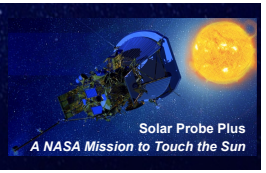
- **LET1 MISC (LET2, HET similar)**
  - SRAM 512kwords available (512k x 24 bits)
    - Basic OS, HK, I/O code + tables 20kwords
    - Event processing, particle ID 210kwords
    - Rates counters, double-buffered 10kwords
    - Event priority buffers 25kwords

    ⟶ SRAM usage ≈ 265/512 = 52%
  - MIPs 12 60% used (Maximum, dominated by event processing during intense SEP events)

- **DPU MISC:**
  - SRAM 128kwords available (128k x 24 bits)
    - Basic OS, HK, I/O code + tables 20kwords
    - Detector module data buffers, & rate accum, double-buffered 30kwords
    - CCSDS Packet buffers 20kwords

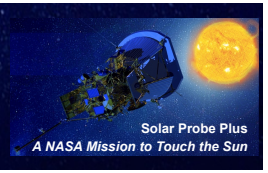
    ⟶ SRAM usage ≈ 70/128 = 54%
  - MRAM 2M x 8 bits ~70% used (2 sets of boot images)
  - MIPs 12 20% used

Pre-PDR Peer Review Summary



An EPI-Hi Flight Software Peer review was held on October 9th. Summary of comments from reviewers:

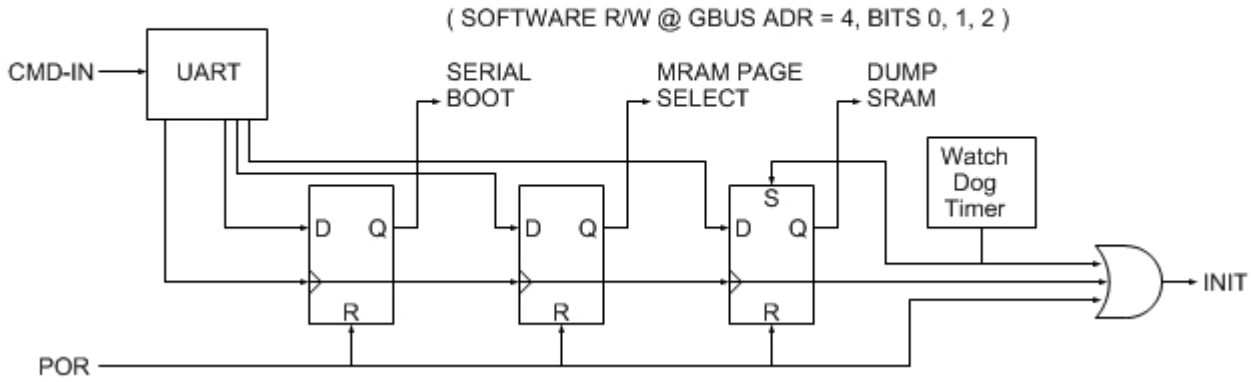
- Need to review NASA software standards docs listed in PAIP, ensure we are in compliance
- Other reviewer comments are TBD



Backup Slides

Boot Logic

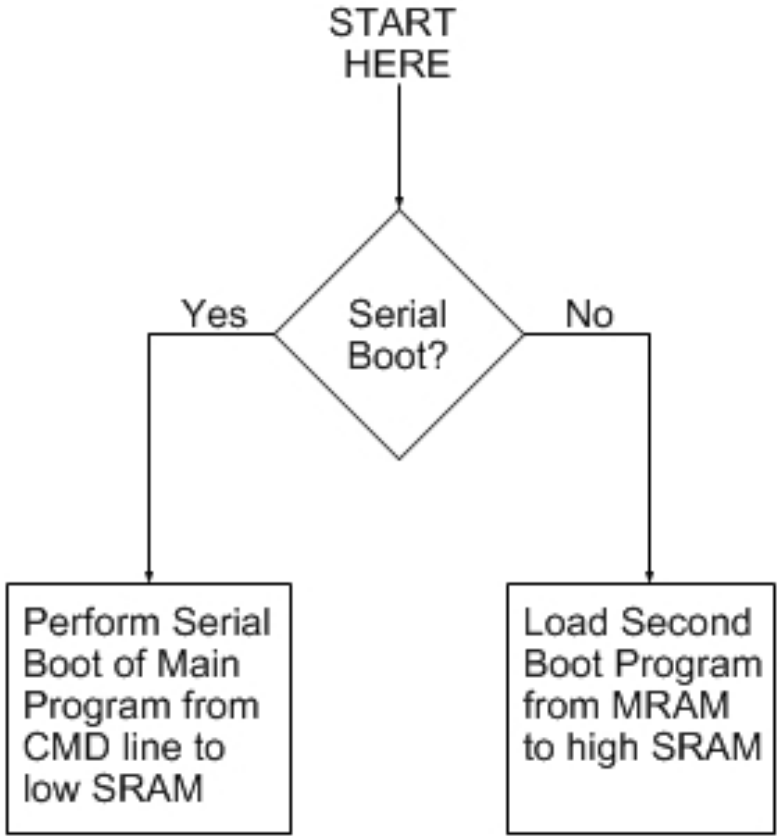
DPU MISC Reset Circuitry Block Diagram



- Supports three ways to initiate boot: Power On Reset (POR), CMD-IN and Watch Dog Timer
- “Reset Bit Field” is writable by CMD-IN and R/W by software
- “Serial Boot” bit determines if boot is to be performed serially via CMD-IN or from MRAM
- “MRAM Page Select” bit determined which page (0 or 1) of MRAM is source of boot images.
- “Dump SRAM” bit determines if SRAM is dumped prior to boot. This bit is set by the watch dog timeout.

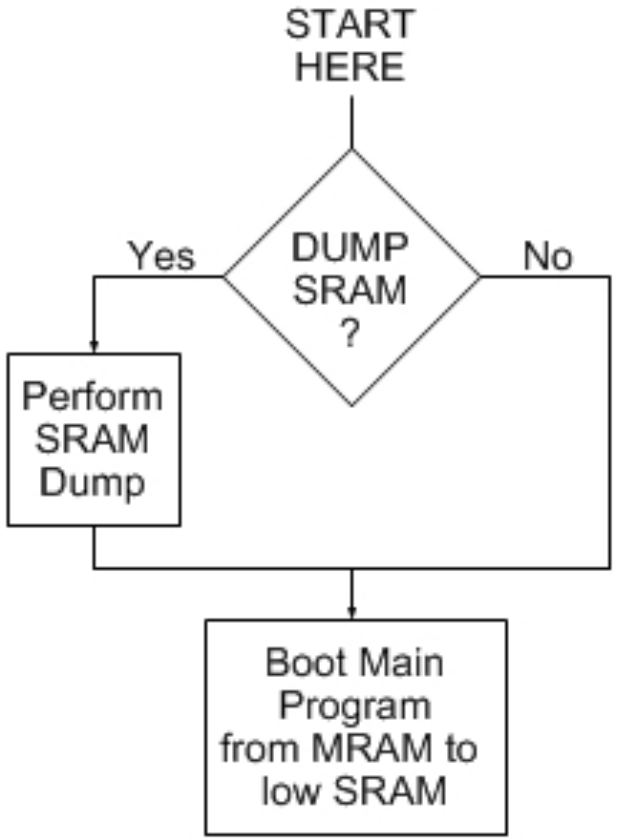
FPGA Resident Boot Program

- Permanently coded as part of FPGA design.
- Simple and small: <64 – 24 bit words
- Either performs serial boot of Main Program into low SRAM via CMD-IN
- Or loads second boot program from MRAM into high SRAM



Second Boot Program

- Optionally performs SRAM dump prior to booting Main Program from MRAM to low SRAM.
- SRAM dump is somewhat complex, does not fit into FPGA resident boot program; hence need for second boot program.



Main Program

- Reads and parses ITF from S/C.
- Determines if in “autonomy mode”.
- If so, boots peripheral MISCs, executes MRAM Patch File, applies HV bias, begins science operations in mode specified in S/C status message.
- If not, just awaits further commands; i.e. during commissioning.