DRAFT

Solar Probe Plus
EPI-Hi Instrument
Flight Software Development Plan

Document No. TBD

Version 03

Date: 2013-09-06

Authors: Andrew Davis

Custodian: Andrew Davis

Space Radiation Laboratory

California Institute of Technology

Pasadena, California

Signature Page

Prepared By:

Andrew Davis, Caltech Science Ops lead	date

Approved by:

TBD	date

Concurred by:

Rick Cook, SPP EPI-Hi systems engineer	date

CHANGE LOG

DATE	SECTIONS CHANGED	REASON FOR CHANGE (ECR)	REVISION
2013-02-11	All	--	Initial preliminary draft
2013-05-01	All	Respond to Project comments	Version 02
2013-09-06	2.1	Minor updates, new block diagram	Version 03 (Baseline)

Table of Contents

1. Overview

1.1. *Introduction*

Four microprocessors are dedicated to controlling, interfacing, and acquiring data from the three telescopes comprising the EPI-Hi instrument (Energetic Particle Instrument - High). This document defines the development plan for the flight software that will reside in these microprocessors.

Flight software for the EPI-Hi instrument microprocessors will be developed at the Caltech Space Radiation Laboratory (SRL). Previously, the Caltech group developed the flight software for the SIS and CRIS instruments on ACE, the SEP instrument suite on STEREO, the NuSTAR X-ray telescope mission, and for several balloon instruments.

1.2. *Document Conventions*

Parameters encompassed by brackets such as [25]: The parameter is To Be Confirmed (TBC) and may be changed as the mission is refined. To Be Determined items are highlighted in red, and labeled TBD. To Be Resolved items are highlighted in red, and labeled TBR.

1.3. *Applicable Documents*

Ref. 1	EPI-Hi Science Requirements Document
Ref. 2	ISIS Level 3 Requirements Document
Ref. 3	ISIS Level 4 Requirements Document
Ref. 4	EPI-Hi Flight Software Requirements Document
Ref. 5	SPP General Instrument Specification (GIS)
Ref. 6	P24 MISC Processor Manual
Ref. 7	EPI-Hi Flight Software Design Document
Ref. 8	EPI-Hi Inter-MISC command/data interface Document
Ref. 9	EPI-Hi Data Format Document
Ref. 10	EPI-Hi Flight Software Test Plan
Ref. 11	EPI-Hi Commanding and User Manual
Ref. 12	ISIS Risk Management Plan

Note: Official document IDs for these references are TBD

2. Host System and Interfaces

2.1. *System Overview*

The microprocessors used for the EPI-Hi instrument will be the FPGA-based P24 MISC (Minimal Instruction Set Computer), described below and in the P24 MISC Processor Manual (Ref. 6). The EPI-Hi MISCs are named:

- Data Processing Unit (DPU) MISC
- Low Energy Telescope 1 (LET1) MISC
- Low Energy Telescope 2 (LET2) MISC
- High Energy Telescope (HET) MISC

Data from the MISCs associated with the LET1, LET2 and HET telescopes (the peripheral MISCs) will be gathered by the DPU MISC and formatted for transmission to the spacecraft (per the SPP GIS, Ref. 5). Figure 2.1 shows a block diagram of the EPI-Hi instrument electronics system, showing internal and external interfaces.

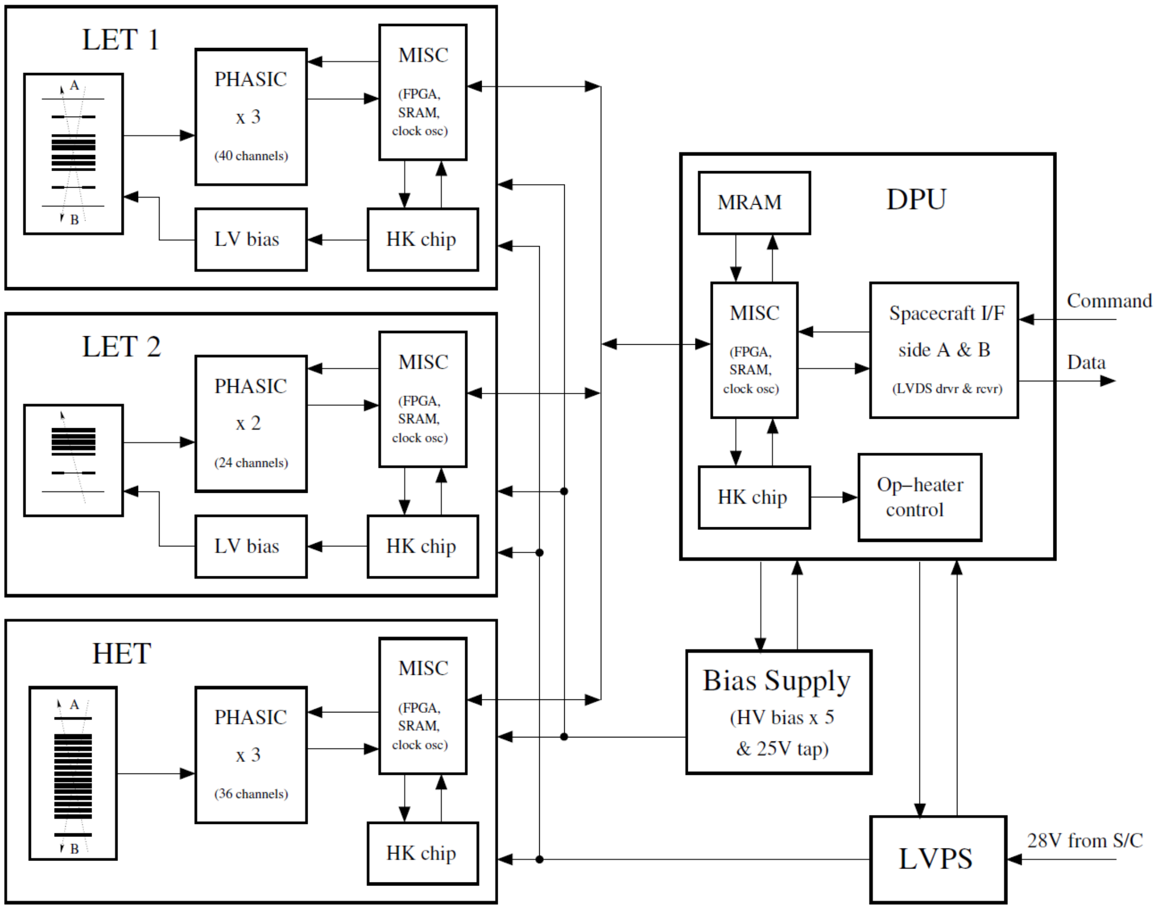

Figure 2.1 – EPI-Hi Instrument Electronics block diagram

2.2. *MISC Microprocessor*

The P24 MISC has a 24-bit CPU core with dual stack architecture intended to efficiently execute Forth-like instructions. The processor design is simple to allow implementation within field programmable gate arrays. The basic MISC processor design is in the public domain. For the EPI-Hi application, the MISC design is implemented in the ACTEL RTAX250SL FPGA, with Caltech SRL customizations developed for the STEREO mission. The MISC can be clocked at speeds up to 25MHz; we will use [14.7] MHz (TBC) for the EPI-Hi application. See the P24 MISC Processor Manual (Ref. 6) for more details.

2.2.1. **Memory Map**

The MISC is capable of addressing a flat memory page of 2^{24} = 16Mwords (each word is 24 bits). For the EPI-Hi application, the DPU MISC will have 128Kwords of SRAM, and each peripheral MISC will have 512Kwords of SRAM. The extra SRAM in the peripheral MISCs is provided for the large lookup tables used in the onboard particle identification software. Jump and call addresses are restricted to an 18 bit address range, so executable code needs to be within the lower 256Kwords of SRAM.

2.2.2. **Operating System**

A Forth operating system with an embedded optimizing forth compiler is implemented. This is the same operating system as used in the MISC systems developed for the STEREO and NuSTAR missions. Multi-tasking is implemented via a round-robin system inherited from the system implemented first in the Harris RTX2010 microprocessor used on ACE, and subsequently on the STEREO and NuSTAR missions

2.2.3. **Boot Loader and MRAM**

Included in the FPGA implementation of each MISC is a small boot loader program that boots the system either directly via Magnetoresistive Random-Access Memory (MRAM), or over a serial link. For EPI-Hi, the DPU MISC is equipped with 2M×8 of MRAM. The other MISCs in the system always boot over the serial link with the DPU. The MRAM in the DPU can be updated via the command interface, initiated by a command that specifies page number, followed by fixed length binary data block. The protocols and formats for MRAM uploads are the same as for Caltech’s NuSTAR instrument, and will be documented in the EPI-Hi Commanding and User Manual (Ref. 11). MRAM uploads will utilize the instrument commanding interface described in the SPP GIS (Ref. 5).

Upon power up, the DPU MISC boot loader automatically boots a binary image from page 0 of MRAM. After power up, a re-boot may be initiated independent of the status of software execution, by sending over the command line a sequence of six consecutive "Z" characters followed by a character whose three least significant bits are written into special "reset field" hardware register. The two least significant bits control the operation of the boot loader. The least significant bit selects between serial boot (value 0) and boot from MRAM (value 1). In the case of reboot from MRAM the boot process occurs in three steps:

1. The FPGA resident boot loader boots a secondary boot loader from MRAM
2. The secondary loader conditionally dumps memory, and then boots the FORTH system
3. The FORTH system loads the "user" program.

The conditional memory dump is controlled by the third least bit of the reset field. The reset field register is also writable from s/w allowing all the reboot options to be exercised by FORTH commands. In the case of a watchdog timer initiated reboot the reset field is cleared, except for the third bit which is set, resulting in a memory dump followed by a reboot from MRAM page 0.

For the peripheral MISCs, the option to boot from MRAM is not available. Upon power up, the FPGA boot loader in these MISCs automatically configures the system to boot via the serial interface with the DPU. Software in the DPU manages the transfer of boot images or memory-dump programs to the peripheral MISCs via the serial interface. This peripheral MISC serial-boot scheme is inherited directly from the NuSTAR mission. Re-boots of the peripheral MISCs are initiated via the same command-line sequence as described above for the DPU MISC.

2.2.4. **I/O Bus (G-Buss) Peripherals**

Currently, G-buss peripherals include an interrupt control and status register at address 0, supporting 15 prioritized interrupts. The interrupts are used to support serial I/O, the timer interrupt, and event interrupts.

2.3. ***External Interfaces***

2.3.1. **Interface between the EPI-Hi DPU and the spacecraft**

The DPU MISC will interface with the spacecraft via two serial interfaces: command and telemetry (see Figure 2.1). These interfaces are defined in the SPP GIS. Commands, telemetry data and command responses from the DPU will be in the form of CCSDS packets enclosed within Instrument Transfer Frames (ITFs). The ITF protocols for transferring commands, command responses, and telemetry data over these interfaces are also defined in the EPI-Hi Instrument/Spacecraft ICD. Instrument telemetry data formats, and instrument CCSDS data packet definitions will be defined in the EPI-Hi Instrument Science Data Format document (Ref. 9).

2.3.2. **Interface between the DPU MISC and the LET1, LET2, and HET MISCs**

There will be two RS422 serial interfaces between the DPU MISC and each of the three peripheral MISCs (LET1, LET1, and HET MISCs). The first interface will be bi-directional, for transferring boot-code, commands, and command responses. The second interface will be uni-directional, for transferring data from the peripheral MISCs to the DPU. These interfaces will be defined in the EPI-Hi Inter-MISC command/data interface Document (Ref. 8).

3. Flight Software Tasks

The flight software requirements will be described in detail in the EPI-Hi Instrument Flight Software Requirements document (Ref. 4), and the Flight Software Design document (Ref. 7). The top-level tasks that the EPI-Hi flight software will perform are listed in Tables 3.1 and 3.2 below.

DPU MISC Flight Software Task	**Heritage (Est.)**
Forth operating system and low-level I/O routines	95%
Power-on and initialization sequence management	60%
Peripheral MISC serial boot sequence management	80%
Housekeeping data collection	50% *
Inter-MISC communication management	90%
Setup and control of instrument HV, bias supply, heaters, and other electronics systems	50% *
Monitor status, and time-synchronization data from the spacecraft, and perform autonomous mode adjustments as needed	10%
Management of software uploads and MRAM burns	75%
Formatting and transfer of science & housekeeping data and command responses to the Spacecraft	50% *
Monitor heartbeat from peripheral MISCs, and perform autonomous diagnostics/reboot as needed	10%
* general scheme is inherited, specifics are unique	

Table 3.1: DPU MISC Flight Software Tasks

LET1, LET2, and HET MISC Flight Software Tasks – each MISC performs a similar set of functions:	**Heritage (Est.)**
Forth operating system and low-level I/O routines	95%
Science data acquisition	50% *
Science data processing and reduction (particle ID)	20%
Housekeeping data acquisition	50% *
Processing of status, time-synchronization, and command data from the EPI-Hi DPU	50% *
Monitor key counting rates and adjust the telescope operating condition to optimize data quality	80%

Monitor temperatures and adjust detector gain/offset settings to compensate for temperature variations	0%
Formatting and transfer of science & housekeeping data and command responses to the EPI-Hi DPU	50% *
Setup and control of any instrument HV, bias supply, heaters, etc. not controlled by the EPI-Hi DPU.	50% *

Table 3.2: LET1, LET2, HET MISC Flight Software Tasks

3.1. ***Heritage and Reuse***

Previously, the Caltech group developed the flight software for the following space missions: SIS and CRIS instruments on ACE, the SEP instrument suite on STEREO, and, most recently, the NuSTAR mission. Some software modules developed for these projects will be reused almost unchanged for EPI-Hi. Some others will be reused with varying degrees of modification. Preliminary estimates of the degree of heritage for the various software tasks are listed in Tables 3.1 and 3.2.

4. Software Development Plan

"Top-down" and "Bottom-up" approaches to software development will run in parallel for the DPU and peripheral MISCs. The Top-down approach can be divided into four phases:

1. Requirements definition and analysis
2. Design
3. Implementation
4. System testing and acceptance testing

The activities occurring during each of these phases are detailed below. During the requirements definition and design phases, the following Bottom-up activities will occur:

1. Gain familiarity with MISC processor, using small test routines
2. Gather together a suitable set of tools for MISC software development, including MISC simulator, serial communication software, version control software, etc.
3. Verify Forth system on MISC with standard software test suite
4. Prototype onboard data processing algorithms

NOTE: for the EPI-Hi application, most of these bottom-up activities have already taken place, given the direct heritage from the STEREO and NuSTAR projects.

By the time the implementation phase of the Top-down approach begins, the Bottom-up approach will have resulted in a stable hardware platform and operating system, and software development tools adequate for implementing the software design.

4.1. *Top-down Software Development Phases*

Although the development phases listed below can be thought of as dividing the software development period into consecutive non-overlapping time periods, activities associated with one phase may be performed in other phases, e.g. most design activity will occur during the design phase, but some preliminary design work may be performed during the requirements definition phase.

4.1.1. **Requirements Definition and Analysis Phase**

During this phase, Caltech will develop a set of science requirements for EPI-Hi. These requirements will be recorded in the EPI-Hi Science Requirements Document (Ref. 4). At the same time, Caltech will also develop preliminary designs of the detectors and the front-end electronics.

Using the science requirements, the preliminary instrument and electronics designs, the SPP GIS, the ISIS Level 3 and Level 4 Requirements, and consultations with other EPI-Hi team members, the Caltech lead engineer and software developer will derive a set of software requirements and interface specifications for each MISC. These requirements and specifications will define what data flows into and out of each MISC, the operations each MISC will perform on the data, and the interactions that will occur between the DPU MISC and the peripheral MISCs. The specifications will also define the relevant properties of the host system, such as memory requirements, I/O peripherals, safing and reliability requirements, etc. All these requirements will be defined in the EPI-Hi Flight Software Requirements document (Ref. 4).

4.1.2. **Design Phase**

During this phase, software developers will define the software architecture that will meet the software requirements. The requirements will be sorted into subsystems and all internal and external interfaces will be defined. The design phase will result in a set of design specifications that will include the following:

- Functional design diagrams
- Detailed descriptions of all inputs, outputs, and data formats
- Data processing algorithms
- Instrument inter-MISC communications protocols
- P24 MISC processor manual
- Command lists and protocols
- Other TBD design specs

The P24 MISC processor manual already exists. The other design specs will be documented in the EPI-Hi Software Design and Data Format documents (Ref. 7 & Ref. 9).

4.1.3. **Implementation Phase**

In this phase, developers will build software from the design specifications. The software will be written in Forth and assembly language. Assembly language will be used when optimization is required for performance reasons. Coding guidelines and standards used on the STEREO and NuSTAR mission will be carried over to this project, given that the same lead engineer and software developer are also carried over to this project.

4.1.4. System Testing and Acceptance Phase

End-to-end sensor, electronics, and software functional tests will be done using radiation sources, built-in self-test routines and test procedures, and instrument calibrations at energetic particle accelerator facilities.
In addition, simulation data from Monte Carlo models of the EPI-Hi telescopes will be used to test the EPI-Hi onboard processing software. The software acceptance test plan is described in section 4.3.2 below.
Software walkthroughs will occur at software peer reviews, and reports and status of action items will be presented at instrument PDR, CDR, and other Project reviews (see Section 5.2).

4.2. *Development Environment and Equipment Needed*

Several MISC development boards are already available for use from previous missions. The Forth system running on the MISC development boards can be controlled via a serial link to a PC. Code can be uploaded via this serial link and thus development/test of software can easily take place directly on the MISC development boards. Early in the design phase, MISC development boards specific to EPI-Hi will be designed and manufactured in-house. These boards will feature the same SRAM, memory-management, boot-sequence, and serial-IO features as will be used for EPI-Hi EM units flight hardware.
Early builds of the instrument EGSE that will interface to these development boards and EM-units will also be delivered early in the design phase.

4.3. *Product Assurance*

The following is based on the approach taken on STEREO and NuSTAR for flight software version control, configuration control and flight software integrity assurance. This same approach will be taken for the EPI-Hi flight software development at Caltech.

In addition, Southwest Research Institute (SwRI) will provide QA support for EPI-Hi software development.

4.3.1. Software Version Control/Backup Plan

- Only one person will be authorized to load flight software into flight MISCs – the lead electrical engineer.
- Version control and software archiving will be achieved via the 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.
- Beginning in the implementation phase, a Software Development Log will be maintained by the lead engineer containing software change requests, the date of any change, a description of the change, and the reason for the change.
- One software developer will be working under the lead engineer. The developer will be required to use the SVN system to maintain version control for his/her portion of the code.
- At the end of each day of software work, the lead engineer and software developer will ensure that all software modules are checked-in to the appropriate location in the SVN repository.
- At the end of each day of software work, the lead engineer will also backup the current flight software directories on his computer to a different computer and/or to removable media.
- Periodically, the software developer will deliver his/her portion of the code to the lead engineer, who will incorporate it into the current flight software build.
- There will never be more than one person working on the same piece of code. If a change is required, the responsible developer will make the change and re-deliver new code to the lead engineer.
- During the MRAM burn-in process, checksums will be calculated and recorded. During boot of a flight MISC, the MRAMs will be read and same checksum will be calculated and typed out. Formal checkout procedures will include recording these checksums and verifying the proper values.
- All new software will be tested first on MISC development boards and EM units prior to being loaded into flight hardware.
- After the beginning of integration and test with flight hardware, all flight software changes will be under configuration management, per the procedures described in Section 4.3.3 below.

4.3.2. **Software Acceptance Test Plan**

The EPI-Hi Flight Software Test Plan document (Ref. 10) 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 (Ref. 1 - Ref. 4) 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.

4.3.3. Software Configuration Management

All problems and change requests will be documented and tracked in the Software Development Log by the lead engineer (see Section 4.3.1). Given such a small development team, a change request tool such as JIRA is an unnecessary overhead.

After the beginning of integration and test with flight hardware, all flight software changes will be approved by a Configuration Control Board (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 build/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.

4.3.4. Risk Assessment

As much as possible, the software utilizes elements flown on previous Caltech instruments and the core software (the Forth OS) is unchanged and flight-proven. Risk-mitigation efforts therefore focus largely on the EPI-Hi-specific elements. Risks will be managed in accordance with ISIS Risk Management Plan (Ref. 12).

4.3.5. Software Maintenance after Delivery

After the EPI-Hi instrument is delivered, the ETUs, together with the supporting GSE, will be maintained at Caltech. Any problems can be recreated and diagnosed in this environment. Flight software changes will be run through an acceptance test on the ETU and CCB review prior to loading into the flight hardware. Every effort will be used to keep the flight software programmers available (though probably on another program) to make any necessary changes through the life of the program. Software documentation will be adequate to allow another programmer to understand and make changes to the software if necessary.

5. Management Plan

5.1. *Build Plan*

Code will be developed using many incremental builds. The Forth environment is extremely modular in nature, with complex software structures being built from many smaller, simpler structures that are independently developed and tested. However, the following software development milestones have been identified in the instrument schedule, and these apply to each of the following modules: DPU, LET1, LET2, and HET.

➢ Build 1 – Support Initial Logic Board. Includes Forth OS, and low-level electronics and serial interfaces. Preliminary data processing routines.
➢ Build 2 – Support EM unit integration. Commands and telemetry processing.
➢ Build 3 – Support accelerator calibration of EPI-Hi telescopes.
➢ Build 4 – Support flight unit integration. Final data processing routines.
➢ Final Build – Final Flight software.

5.2. *Software Reviews*

Reviews by Project personnel of the software requirements and development will occur several times during the development and implementation phases. At these reviews, metrics including CPU margin, RAM and MRAM margin, and open software issues will be presented. Action items from these reviews will be captured and the status of these action items will be presented at later reviews.

- A Flight Software Requirements review will be held with the EPI-Hi team and ISIS and SPP Project personnel attending to ensure that the documented requirements are complete and clear. This review will occur at the end of the requirements definition and analysis phase.
- At Instrument PDR and CDR, the flight software development and design will be reviewed.
- A flight software walkthrough will be held with ISIS and SPP Project personnel a few months before the acceptance tests are scheduled.
- A software acceptance review will be held after the acceptance tests.

5.3. *Documentation Deliverables*

The EPI-Hi Flight Software will be documented as follows:

Document	**Delivery Schedule**
P24 MISC Processor Manual (Ref. 6)	Instrument PDR
EPI-Hi Instrument Flight Software Requirements Document (4)	Instrument PDR
EPI-Hi Instrument Flight Software Design Document (Ref. 7)	Instrument CDR
EPI-Hi Data Format Document (Ref. 9)	Beginning of I&T
EPI-Hi Flight Software Test Plan (Ref. 10)	Instrument CDR
Software Development Log	Beginning of I&T
EPI-Hi Commanding and User Manual (Ref. 11)	Beginning of I&T
Source code comments	Beginning of I&T

5.4. *Staff and Schedule*

Lead engineer and software developer: Rick Cook, Caltech SRL
Software developer: Andrew Davis, Caltech SRL.

The Flight Software schedule is maintained in the Instrument Schedule (a separate Microsoft Project document).

Software maintenance following delivery of the EPI-Hi instrument to the spacecraft will be on an as-needed basis

Appendix A Acronyms

CCB	Change Control Board
CCSDS	Consultative Committee for Space Data Systems
CDR	Critical Design Review
EPI-Hi	Energetic Particle Instrument - High Energy
DPU	Data Processing Unit
EM	Engineering Model
ETU	Engineering Test Unit
FPGA	Field Programmable Gate Array
GIS	General Interface Specification
GSE	Ground Support Equipment
HET	High Energy Telescope
HV	High Voltage
ICD	Interface Control Document
ISIS	Integrated Science Investigation of the Sun
IT	Information Technology
I&T	Integration and Test
ITF	Instrument Transfer Frame
LET	Low Energy Telescope
MISC	Minimal Instruction Set Computer
MRAM	Magnetoresistive Random-Access Memory
OS	Operating System
PDR	Preliminary Design Review
QA	Quality Assurance
SPP	Solar Probe Plus
SRAM	Static Random-Access Memory
SVN	Apache Subversion version control system
VDD	Version Description Document
VLSI	Very Large Scale Integration