Solar Probe Plus EPI-Hi
Flight Software Design Document

Version 0.1
PRELIMINARY DRAFT

Date: 2013/10/03

Authors: Andrew Davis
Custodian: Andrew Davis

Space Radiation Laboratory
California Institute of Technology
Pasadena, California

Signature Page

Prepared By:

Andrew Davis 10/11/2013
Caltech Software Engineer

Approved by:

_____ _____
Rick Cook, Caltech Sr Software Engineer date

_____ _____
A Very Important Person date

CHANGE LOG

DATE	SECTIONS CHANGED	REASON FOR CHANGE (ECR)	REVISION
2013/10/03	All	NA	Initial draft

Table of Contents

1 Introduction

1.1 Purpose and Scope

This Software Design Document (SDD) establishes the software design for the Solar Probe Plus (SPP) EPI-Hi instrument Flight Software (FSW) . The developer of the instrument FSW is the Space Radiation Laboratory at Caltech.

1.2 Relationship to Other Documents

This document is directly responsive to the SPP EPI-Hi Flight Software Requirements Document (Ref. 1), And the SPP Epi-Hi Flight Software Development Plan (Ref. 2). Those documents supercede this document in case of a conflict.

Other documents referred to by this document:

Ref. 3 SPP General Instrument Specification (GIS)

Ref. 4 ISIS Specific Instrument ICD (SI ICD)

Ref. 5 EPI-Hi Data Format Document

Ref. 6 P24 MISC Processor Manual

Ref. 7 EPI-Hi Instrument Flight Software Test Plan

Ref. 8 EPI-Hi Instrument Flight Software Verification Matrix

Ref. 9 EPI-Hi Event Processing and Particle Identification Scheme

1.3 Key Definitions

To Be Confirmed (TBC): an attribute or parameter value believed to be known with reasonable confidence and stated, but subject to further minor (5-10%) refinement.

To Be Resolved or Reviewed (TBR): an attribute or parameter value that is used in the requirement statement as a reasonable placeholder, but known to be in dispute until a disposition can be reached by the affected parties. May also be designated by the use of brackets [ ] around the attribute.

To Be Determined (TBD): an attribute or parameter value unknown at the time of writing.

1.4 Document Layout

An overview of the instrument design and operations concept is given in Section 2 – skip this if you are already familiar with it. Exposition of the FSW design is in Section 3, and the software development environment is described in Section 4. Requirements traceability and software verification are described in Refs 5 and 6.

2 Instrument Overview

2.1 Instrument Design

The SPP EPI-Hi instrument consists of three detector telescopes and associated electronics, designed to measure the intensity, composition, and angular distribution of protons and heavy ions in the energy range ~1MeV/nucleon to ≥50 MeV/nucleon, and electrons in the energy range ~0.5 MeV to ≥3 MeV. The telescopes are

- Low Energy Telescope 1 (LET1): Double-ended.
- Low Energy Telescope 2 (LET2): Similar to LET1, only single-ended.
- High Energy Telescope (HET): Double-ended.

All sensor elements in the telescopes are ion-implanted silicon solid state detectors. Multiple telescopes provide large energy range and greater sky coverage. Sensor elements are segmented to provide angular sectoring and adjustable geometrical factor.

The telescopes have significant heritage from numerous energetic particle instruments developed by the Caltech Space Radiation Laboratory (SRL) over the past 40 years. Direct predecessors are the LET and HET telescopes on board the STEREO spacecraft. Key differences between the SPP and EPI-Hi telescopes are

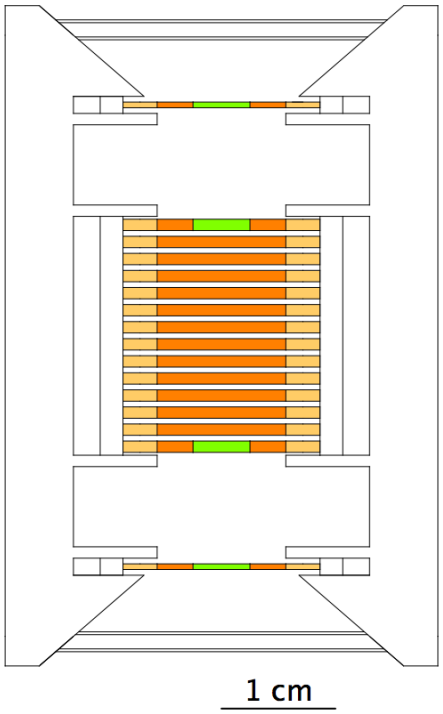

Figure 1*: HET telescope. Colored regions: active Si detector segments.*

- Thinner detectors and windows to reduce energy threshold
- Compact telescope designs to reduce saturation at high particle intensities and backgrounds at low intensities

Figure 1 shows a schematic of the HET telescope. The segmented silicon detectors are of circular cross-section, and are mounted in a stack. Energetic particles may enter the stack from either end. Signals from each active detector segment are individually read-out by the electronics. The LET1 and LET2 telescopes have a conceptually-similar design to HET, but are optimized for measuring lower-energy particles. LET2 is single-ended, since one end is shielded by the spacecraft.

The electronics subsystem is shown as a block diagram in Figure 2. The DPU provides system-level control of the EPI-Hi instrument, and provides the interface to the spacecraft. Operational control radiates out from the DPU to each of the detector modules, the Bias supply, and the LVPS. The DPU receives commands from the spacecraft and executes the commands. In some cases,

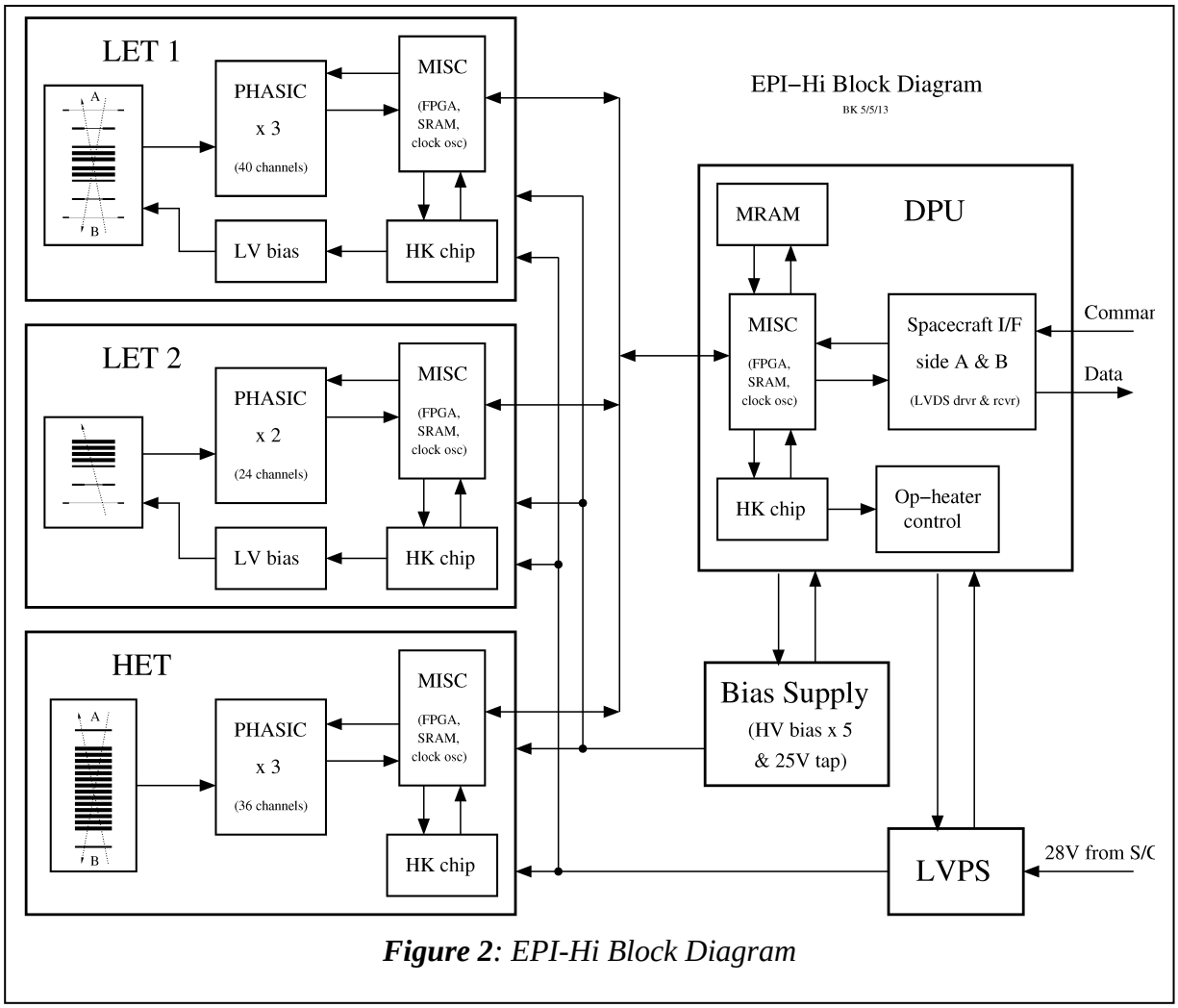

***Figure 2**: EPI-Hi Block Diagram*

command execution by the DPU consists of routing the command to the appropriate detector module.

The DPU and the three detector modules (LET1, LET2, and LET3) are each endowed with a custom FPGA-embedded Minimal Instruction Set Computer (MISC) processor. These processors are capable of booting either from Magnetoresistive Random-Access Memory (MRAM), or via a serial link. For SPP, only the DPU MISC boots from MRAM. The satellite MISCs in the detector modules each boot via serial link under the control of the DPU, using boot images stored in the DPU MRAM.

The software running in the DPU MISC orchestrates the operation of the instrument and controls the command/data interfaces with the spacecraft, while the software in each of the detector MISCs controls detector operation, data acquisition, data processing, manages the command/data interfaces with the DPU.

2.2 Instrument Operations Overview

During normal operations, the instrument runs autonomously, with instrument commanding required only for updating operating parameters if/when needed. Nominally, the instrument changes its operating mode autonomously in response to status data received once per second from the spacecraft. Status data include solar distance, orbit inbound/outbound flag, startup mode, and amount of space left on the SSRs. Details of the S/C status message may be found in the GIS (Ref. 3). Software in the DPU MISC monitors the S/C status message and fans mode-change commands to the detector modules as needed.

The start of the S/C status message also serves a virtual 1PPS "frame sync" pulse, since a hardware frame-sync from the S/C is not available (see the GIS, Ref. 3). The S/C status message also contains the Mission Elapsed Time (MET) at the next 1PPS. The DPU uses the virtual 1PPS and the MET data for time synchronization, and provides the 1PPS to the detector modules so that they are also time-synced.

2.2.1 Operating Modes

The instrument has only two normal operating modes:

- Spacecraft- Sun Distance R<0.25 AU (Normal Science Mode)
- Spacecraft- Sun Distance: 0.25< R<0.76 AU (Low-rate Science Mode)

These two modes differ only in the data transmission rate from the DPU to the spacecraft. The data collection rate and the operating mode of the three detector modules remains the same in these two modes. In both modes, the detector modules transmit the same data products at the same rate to the DPU. The DPU decides, based on the mode, what data are to be transmitted to the spacecraft to be stored on the SSR. Essentially, each of the detector modules has only one normal operating mode.

At least three special operational modes are currently envisioned. Several more may be defined in the future. The three currently-envisioned modes are:

- Software upload mode
- Calibration Science mode
- Safe mode

During software upload mode, data acquisition functions and some non-essential functions will be halted.

Calibration Science mode is a mode during which statistically significant samples of Pulse Height Analysis (PHA) event data will be accumulated and returned to validate onboard assignments of species, energy, and incidence angle, and for assessing instrument backgrounds. This mode will be activated outside 0.25 AU by command one or more times early in the mission when sufficiently high intensities of solar energetic particles are present for calibration purposes. This mode, which will be used for a few days after activation of the mode, will

require the return of a data volume significantly greater than that provided by the Low-rate Science Mode. The details of implementing this mode are TBD.

Safe mode is a mode where the instrument sits in a “quiet” state, awaiting commands from the ground, performing no data acquisition functions. This mode is reserved for instrument I&T, commissioning, and instances where the flight software encounters some irrecoverable fault (TBD).

2.2.2 Data Collection Cycles

Each detector module (LET1, LET2, HET) collects data from the detector electronics in three forms:

- Housekeeping data (HK) : temperatures, voltages, status, etc., read out at regular time intervals
- Hardware counters: counts read out from various hardware/firmware counters at regular time intervals
- Event data: whenever an incoming energetic particle satisfies the coincidence requirements programmed into the front-end electronics, an “Event” is triggered. The front-end electronics read out the triggered detector elements and generates a data block of detector signal amplitudes and addresses. Each block of event data is stuffed into a FIFO by the front-end electronics. The detector module FSW reads event data blocks out of this FIFO for analysis and possible insertion into the telemetered detector data.

There is not enough bandwidth to telemeter the raw data blocks generated by every event to the ground. Therefore, the event data blocks must be analyzed onboard, and only a selected subset of events will have their full data telemetered, for diagnostic purposes.

The analysis of each block of event data by the FSW in each detector module results in the incrementation of one or more software-maintained counters. The FSW is designed to determine the species, energy, and trajectory of the particle that generated the event, and to bin the result into appropriate species, energy, and angular bins (see Ref. 9). A counter is associated with each of these bins. The content of these software counters constitutes the primary science data product.

Each detector module has a 1-second data collection cycle that is tied to the 1PPS time pulse received from the DPU. During second N, the HK and all the counters from second N-1 are transferred to the DPU. Also, a prioritized selection of event data blocks is transferred to the DPU. Except for command responses and some items that may be multiplexed in the housekeeping data, the detector module telemetry data repeats the same format each second.

The DPU is responsible for formatting all the data received from the detector modules into CCSDS packets an d ITFs, per the EPI-Hi Data Format document (Ref. 5). It is also responsible for accumulating longer-term rates (10-second, 1-minute, 1-hour, …TBD) from the 1-second rates delivered by the detector modules. Based on the operating mode (normal or low-rate) and the S/C status data, the DPU will select the appropriate subset of data to be packetized and

transferred to the spacecraft for storage on the SSRs. Thus, during low-rate mode the 1-second counts would not be telemetered, and only a very restricted subset of event data would be transferred. Details on this will be found in the Data Format Document (Ref. 5).

2.2.3 Electronics Test Pulser Cycle

This is an autonomous activity that runs continuously and cycles the detector module electronics test pulsers through a series of amplitude steps. The test pulser settings are contained in onboard tables, and the sequencing is controlled by software counters and the 1Hz spacecraft timing pulse. Data from events that are triggered by the test pulses are folded into the standard science telemetry, and the test pulses do not influence the instrument mode of operation.

2.3 Instrument Commissioning Activities

This activity will checkout the instrument electronics and prepares the instrument electrically to take data.

Prior to this activity the following pre-conditions must be met:

- Details TBD, but basically the spacecraft must have successfully completed its own initial commissioning activities.

Once the preconditions have been met the first of the 2 Instrument commissioning sub-activities, Instrument Electronics Power On and Initial Checkout, can be performed.

2.3.1 Instrument Electronics Power-On and Initial Checkout Activity

The spacecraft will apply operational power to the Instrument upon receipt of a ground command, per the SI ICD (Ref. 4) . Once powered on, the DPU flight software will boot and from then on the DPU will control the initialization of the other instrument subsystems. The instrument power-on and initialization steps are as follows:

- TBD – details to be provided by Rick Cook prior to PDR.

2.4 Instrument Commanding

Details of the instrument commanding interface and protocols are contained in the GIS and SI ICDs (Refs. 3 and 4).

Instrument commands and software uploads are transmitted to the spacecraft in CCSDS telecommand packets encapsulated inside ITFs, per the GIS.

There are no requirements on the spacecraft to understand or act upon the contents of an instrument telecommand packet. There are no requirements on the spacecraft to wait either a minimum or maximum time before sending commands. In other words, all requirements for command timing are levied on the ground.

3 Flight Software Design

The instrument FSW is distributed across the four instrument MISCs: one in the DPU, one in each of the detector modules. Thus, the instrument FSW can be divided into four MISC-level modules.

The top-level FSW design is presented from an operational perspective and attempts to do the following:

- Identify flow of control and the events that can affect the processing of data; Present the instrument-level control flow and data-flow diagrams.
- Examine the data flowing in and out of the system; Present the instrument-level and MISC-level data flow diagrams.
- Provide high-level descriptions of the MISC-level modules of the FSW and how those modules map to the system requirements.
- Describe the hardware interfaces controlled by the MISC-level FSW modules. The interfaces are dealt with at a high-level. The intent is to describe the type of hardware control and status that is provided by the FSW.

3.1 Instrument-Level Control Flow

The instrument-level control flow diagram is shown in Figure 1. The DPU FSW has system-level control of the instrument, and provides the interface to the spacecraft. Control radiates out from the DPU to the FSW in each of the detector modules. Each detector module controls and reads data from its own set of detectors and housekeeping (HK) sensors. The DPU receives commands from the spacecraft and either executes the commands, or routes them to the appropriate detector module. The DPU FSW is responsible for orchestrating the boot process for the other MISCs after power-up or reset. The DPU FSW has direct control over the Bias Supply and the LVPS. Each MISC-level FSW module also has several other tasks, which are enumerated and described below.

FIGURE TBD

Figure 1 – Instrument Level Control Flow Diagram

3.2 Instrument-Level Data Flow

The instrument-level data flow diagram is shown in Figure 2.

Raw data, including housekeeping data, flow from the detector modules to the DPU via RS-422 serial interfaces. These interfaces run at 461 kbps, but are not typically used to full capacity. The DPU also gathers HK data from various sensors and subsystems under its control.

Once per second, The DPU FSW aggregates the data received from all these sources during the previous second, and formats the data into CCSDS packets and ITFs per the Instrument Science Data Format document. During the following second, these packets are transmitted to the spacecraft.

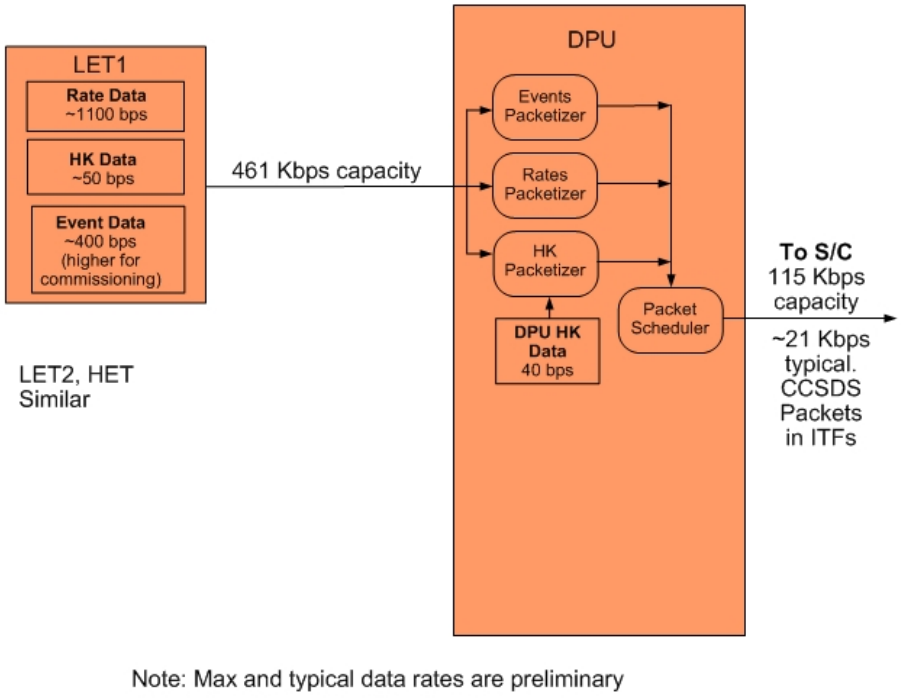

Figure 2 – Instrument Level Data Flow Diagram – Typical data rates and interface capacities shown

3.3 Design Elements Common to FSW in all Modules

3.3.1 MISC Architecture

The microprocessor used in the DPU and detector modules is the P24 MISC (Minimal Instruction Set Computer), designed at Caltech with the aid of Dr. C. H. Ting. The design derives from earlier MISC implementations developed by Chuck Moore (MuP21; see http://www.ultratechnology.com/mup21.html) and C. H. Ting (P8 and P16) and is simple enough to fit within a field-programmable gate array (FPGA), yet powerful enough to provide the needed on-board event analysis capability. For NuSTAR, Both the P24 design is implemented in the ACTEL RTAX250 FPGA.

The MISC employs a RISC-like instruction set with four 6-bit instructions packed into a 24-bit word. Instructions are executed consecutively after a word is fetched from memory. The most significant bit of each instruction designates an I/O buss (gbus) operation when set. For I/O buss instructions the second most significant bit specifies a write when set, read when cleared, while the remaining four bits specify the I/O buss address. For non-I/O buss instructions the most significant bit is cleared and the remaining 5 bits specify 32 possible instructions, 31 of which are implemented.

Following is a list of distinctive features of the P24 MISC:

- 24-bit address and data busses
- 6-bit RISC-like CPU instructions
- 4-deep instruction cache
- 256-deep data stack
- 256-deep return stack
- Current implementation runs at 15 MHz (at least 25 MHz capability), with 1 wait-state for SRAM fetch or store.

The MISC has the following registers, all 24 bits wide:

A	Address Register, supplying address for memory read and write
I	Instruction Latch, holding instructions to be executed
P	Program Counter, pointing to the next program word in memory
R	Top of Return Stack
S	Top of Data stack
T	Accumulator for ALU

The return stack is used to preserve return addresses on subroutine calls. The data stack is used to pass parameters among the nested subroutine calls. With these two stacks in the CPU hardware, the MISC is optimized to support software written in the Forth programming language.

For more information, consult the P24 MISC Processor Manual (Ref. 6).

3.3.2 Forth Interpreter and Programming Language

When booted, each MISC runs an embedded Forth interpreter (or kernel), which occupies ~8Kbytes of RAM.

Forth is a structured, imperative, stack-based, computer programming language and programming environment. It is a language with a simple syntax and many keywords. This is in contrast to Algol-style languages (such as Pascal and C/C++), which have a complex syntax and few keywords.

Forth programs are made of many small procedures. Forth is compiled, yet has no compiler in the traditional sense. Essentially, it's a population of subroutines and an interpreter. The subroutines are called words. (In this article, words will appear in UPPER-CASE.) The dictionary is a data structure that associates the compiled words with their string names. The interpreter can invoke words that perform compilation actions, thereby extending the dictionary in the middle of a program. Figure 1 shows a flow chart of a Forth interpreter. The interpreter evaluates white space-delimited strings taken from an input stream, such as a console or file, usually in one pass.

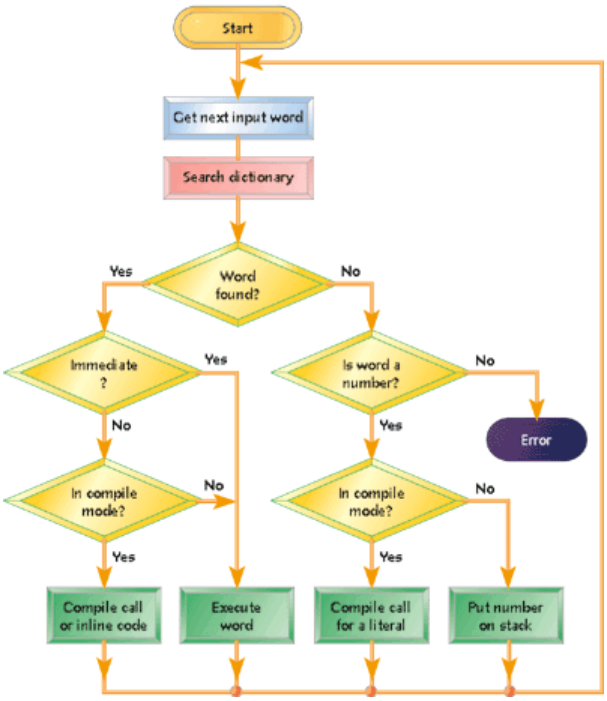

Figure 3 –A Forth interpreter's flow chart

You write a Forth program by defining new words, and run it by executing the top-level word. Forth manipulates data on a parameter stack that is separate from the return stack. Although static variables can be defined, words generally pop their parameters from the parameter stack, and push their results onto it. For example, the built-in word + pops the top two values, adds them, and pushes the sum back onto the stack. Bitwise AND operates similarly. The word < pops two values, compares them, and pushes the result (0 or -1). So a Forth programmer would code (2+3)*(4+5) as 2 3 + 4 5 + *, in reverse polish notation (RPN).

The Forth standard specifies a boolean result as all 0's or all 1's, which is 0 or -1 in twos complement arithmetic. This allows you to mix arithmetic and boolean operations, for example << 7 AND. The compiler allows any kind of type mixing, as Forth is typeless.

With most data kept on the parameter stack, there's less need than in other languages to track variable names or addresses, and temporary storage is automatic. Several built-in words manipulate the stack by rotating, removing, copying, or displaying items from various stack positions: SWAP swaps the top two items, DROP removes the top stack item, and OVER copies the second stack item to the top of the stack, thereby increasing the stack size by one.

Words that manipulate character strings generally require a pointer as the second item on the stack and the string length on the top. Branching and looping words also use the stack. IF pops and tests the value on top of the stack. If the value is non-zero, the next word executes. Otherwise, control passes to the word following the ELSE, if present. BEGIN starts an indefinite loop and the corresponding END pops and tests the top stack value, looping back to BEGIN if it's zero. DO/LOOP pairs repeat until an index (passed on the stack) increments to or

beyond a limit (also passed on the stack). An important difference is that the index and limit are copied to the return stack, to avoid cluttering the parameter stack.

For more information about the Forth language, see, for instance, http://en.wikipedia.org/wiki/Forth_(programming_language).

The specific Forth interpreter embedded in NuSTAR MISCs is a Caltech customization of eForth: see http:/www.forth.org/eforth.html

3.3.3 Multi-tasking environment

The Forth operating system supports simple round robin multi-tasking, in which the responsibility for hand-off of cpu control resides with the user-defined task. There is no operating system interference in task switching, which occur in only a few clock cycles. The multi-tasking system consists simply of a few subroutines (Forth "words") that allow the creation of tasks, their insertion into the round robin, and the control of their state ("asleep" or "awake"). Tasks which are asleep are efficiently bypassed in the round robin. Typically a task will put itself to sleep while it waits for input from another task or an interrupt service routine which will later wake the task once the input is ready.

Since the forth system command interpreter/compiler (OPERATOR) is one of the tasks in the round robin and it is desired to keep the response time to commands short, the tasks are typically designed such that the maximum time around the loop is less than about 10 msec.

3.3.4 Interrupts

11 prioritized and vectored interrupts are supported. See section 3.3.5 below for more information.

3.3.5 Memory Map – Parts of this section to be updated for EPI-Hi

The system's hardware defined memory map includes SRAM, EEPROM, and boot code that is internal to the MISC FPGA. The entire address space of 24 bit words ranges from HEX 0 through 3FFFF. The lower half is devoted to SRAM and the upper half to EEPROM. Multiple pages of EEPROM are supported by extended address bits held in gbus addressable i/o.

Some addresses have special functions and are decoded separately. These include address 0, which is the address where execution begins after a reset. The instruction at this address is hard-coded into the FPGA and is a jump to address $20001. The address range $20001 through $2000F is also specially decoded depending on the status of the "serial-boot" input pin on the FPGA. If the serial-boot line is high this indicates that boot should occur over the serial interface and the address range $20001 through $2000F points to the serial boot code which is hardwired into the FPGA. Otherwise it points to EEPROM, where is stored a short program to boot from the EEPROM.

Addresses 1 through 11 are the addresses to which interrupts 0 through 10 vector, respectively. The Forth operating system, which includes the standard Forth dictionary of subroutines, begins at address 12 and extends up through address 2580. Application specific code begins at 2581 and grows upward by the addition of code, tables, etc. to the dictionary. Applications are

free to define data structures outside the dictionary, and these typically grow from high memory down toward the dictionary.

The Forth system "READ" and "OK" words, that respectively receive a text file and compile it into the dictionary, normally use the address range $1B000 through $1FFFF as their text buffer.

3.3.6 Software Boot, Upload, and Patching Capabilities

Details TBD – to be provided by Rick Cook prior to PDR

3.3.7 Fault Protection Features

3.3.7.1 Watchdog Timer

The DPU will support a watchdog timer system that will allow a reset of the DPU MISC if the main program fails to service the watchdog interrupt.

If the DPU MISC resets due to a watchdog timer exception, the following occurs:

1. TBD – details to be provided by Rick Cook prior to PDR

3.3.7.2 DPU Monitoring of Detector Module Health

CEB is capable of monitoring the health of FPM and OBEB MISCs by counting the data packets and command responses received. It is capable of autonomously performing a memory dump/reboot of any of these systems if an anomaly is detected. This function can be enabled/disabled by command.

Note: currently not planning a watchdog time system for the detector modules.

3.3.7.3 SRAM Monitoring and Redundant MRAM - **may need update for EPI-Hi**

The DPU Stores dual copies of boot code for all MISCs in MRAM.

An EDAC is implemented in FPGA as a result of JPL SEU review. Single and multi-bit error rates will be monitored in space and included in telemetry. Only a few single bit errors are expected (corrected by EDAC) and no multi-bit errors over the 2 year mission.

Checksums over static SRAM (& MRAM) contents will be calculated periodically in-flight and included in telemetry.

3.3.7.4 MISC Stack Underflow/Overflow Protection

There is none – this is up to the software designer. Note that the stacks are circular, and are 256 levels deep now.

3.4 DPU FSW Design

3.4.1 DPU – Spacecraft Interface

The electrical interface between the DPU and the spacecraft is described in detail in the GIS and SI ICDs (Refs. 3 and 4). Briefly, there are two serial interfaces;

- COMMAND: an RS-422 serial interface running at 115.2kbps for transferring commands and software uploads from the spacecraft to the instrument.
- DATA: an RS-422 serial interface running at 115.2kbps for transferring telemetry data from the instrument to the spacecraft.

3.4.2 DPU Command and Data Interfaces with Instrument Subsystems

There are two standard RS-422 serial interfaces between the DPU and each of the detector modules. The first is a bi-directional interface for the transfer of commands and boot code from the DPU to the module, and the transfer of command responses from the module to the DPU. This interface runs at 57.6kpbs and is multiplexed between the modules. The second serial interface is unidirectional, for the transfer of data from the subsystem to the DPU. The data interface is not multiplexed between the subsystems – each subsystem has its own data interface with the DPU.

3.4.3 DPU FSW Tasks

The FSW running in the DPU MISC performs the following tasks:

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	

Figure 4 shows the DPU FSW processing loop, and identifies the interrupt service routines, round-robin tasks, and data structures that together comprise the DPU FSW.

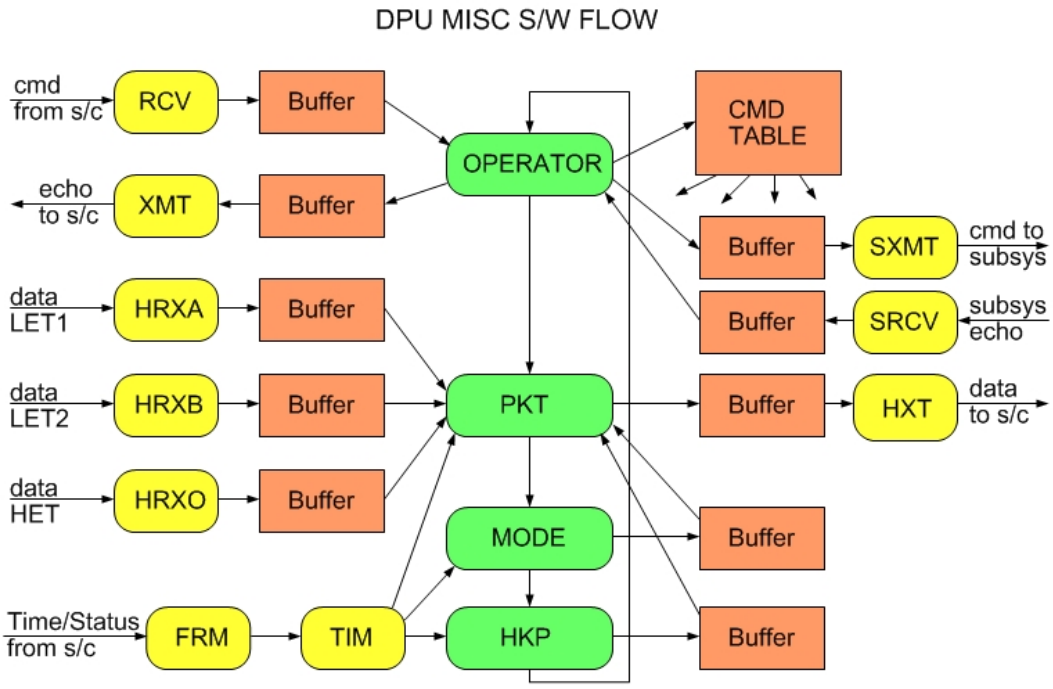

Figure 4 –DPU MISC FSW Processing Loop

The following tasks are hooked into the CEBs round-robin multi-tasker:

- OPERATOR - forth system command interpreter/compiler
- PKT - packetizer/packet scheduler
- MET - metrology system data collector
- MODE – operational mode controller

Most of the interrupts come from UARTs. The TIM timer interrupt is generated by a hardware timer within each MISC.

A vectored scheduler (not shown) runs within the TIM interrupt service routine and provides millisecond scheduling resolution.

3.5 Detector Module FSW Design

3.5.1 Detector Module-DPU Interface

This interface is described in Section 3.4.2

3.5.2 Interface to Detector Elements

This is same as for STEREO. The description is TBD.

3.5.3 Detector Module FSW Tasks

The FSW running in each of the two three Detector Module MISCs (LET1, LET2, and HET) performs the following 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% *

Figure 5 shows the FP FSW processing loop, and identifies the interrupt service routines, round-robin tasks, and data structures that together comprise the FP FSW.

FIGURE IS TBD

Figure 5 –Detector Module MISC Processing Loop

The following tasks are hooked into the s round-robin multi-tasker:

- OPERATOR - forth system command interpreter/compiler
- EVP - event processor
- HKP - housekeeping collector
- LKS - detector leakage current collector, balancer
- RTS - rate/livetime collector
- CAL - CZT detector stim pulse cycling controller
- TBD

As in the DPU, a vectored scheduler (not shown) runs within the TIM interrupt service routine and provides millisecond scheduling resolution.

3.5.4 Event Processing

See EPI-Hi Event Processing and Particle Identification Scheme (Ref. 9).

4 Development Environment

This is described in the Flight Software Development Plan. A brief summary will be given here.

TBD.

5 Inherited Software Components

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 and 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 and NuSTAR
- The software for processing instrument commands, routing commands to satellite MISCs, and packetizing command responses is the same as for STEREO

6 Design Requirements Traceability and Software Verification

The EPI-Hi Flight Software Verification Matrix is a working document that is expected to change as the software verification process evolves. It will therefore be provided as a separate document.

7 Acronyms and Terms

ADC - Analog to Digital Converter

ALU - Arithmetic Logic Unit

APID - Application Process Identifier

CCSDS - Consultative Committee for Space Data Systems

DPU – Data Processing Unit

EDAC - Error Detection and Correction

MRAM – Magnetoresistive Random Access Memory

FPGA - Field Programmable gate Array

FSW - Flight Software

HK - Housekeeping

I/O - Input Output

ICD - Interface Control Document

JPL - Jet Propulsion Lab

LC - Leakage Current

LVPS - Low Voltage Power Supply

MISC - Minimal Instruction Set Computer

NuSTAR - Nuclear Spectroscopic Telescope Array

RISC - Reduced Instruction Set Computer

S/C - Spacecraft

SRAM - Static Random Access Memory

VLSI - Very Large Scale Integration