STEREO
IMPACT SEP

Flight Software Development Plan

SEPSoftwareDevelopmentPlan_C.doc
Version B – 2001-09-04

Andrew Davis, Caltech Space Radiation Laboratory

Document Revision Record

Rev.	Date	Description of Change	Approved By
A	2001-July-24	Preliminary Draft	-
B	2001-Aug-21	Incorporate Software Requirements Doc and reorganize per Project template	-
C	2001-Sept-04	Updated figures, added SEP Common Software Material	

Distribution List

Rick Cook, Caltech SRL
Alan Cummings, Caltech SRL
Andrew Davis, Caltech SRL
Branislav Kecman, Caltech SRL
Allan Labrador, Caltech SRL
Richard Mewaldt, Caltech SRL
Robert Radocinsky, Caltech SRL
Ed Stone, Caltech SRL
Mark Wiedenbeck, Caltech SRL
Donald Reames, GSFC
Tycho von Rosenvinge, GSFC
Kristin Wortman, GSFC
Dave Curtis, UCB
Glen Mason, UMD
Peter Walpole, UMD

Table of Contents

1. Overview

1.1. *Introduction*

The IMPACT SEP instrument suite consists of four instruments – LET, HET, SEPT and SIT. Four microprocessors are dedicated to controlling, interfacing, and acquiring data from these instruments. This document defines the requirements and the development plan for the flight software that will reside in these four microprocessors.

SEP flight software will be developed at the Caltech Space Radiation Laboratory (SRL) and at the Laboratory for High Energy Astrophysics at GSFC. Previously, the Caltech group developed the flight software for the SIS and CRIS instruments on ACE, and for several balloon instruments. The GSFC group developed the flight software for the EPACT instrument suite on WIND.

1.2. *Document Conventions*

In this document, TBD (To Be Determined) means that no information currently exists. TBR (To Be Resolved) means that a statement is preliminary. In either case, the acronym is typically followed by the initials of those responsible for providing the information.

1.3. *Applicable Documents*

Some of these documents and drawings can be found on the Berkeley STEREO/IMPACT website: http://sprg.ssl.berkeley.edu/impact/dwc/
Others are currently available from Caltech SRL.

1. Phase A Report/PAIP (Performance Assurance Implementation Plan)
2. LET Science Requirements Document
3. HET, SEPT and SIT Science Requirements Documents (TBD-SEP Team)
4. P24 MISC processor manual
5. IMPACT Intra-Instrument Serial Interface ICD
6. SEP Instrument Suite ICD (TBD-Kecman/WRC)
7. P24 MISC G-buss I/O Interface Document (TBD-WRC)
8. DRAFT_ICD_IMPACT_a (IMPACT/Spacecraft ICD, on the APL web page)
9. LET Science Data Frame Format Specification
10. SEP Flight Software User Manual

1.4. *Acronyms*

ACE Advanced Composition Explorer
DPU Data Processing Unit
HET High Energy Telescope
ICD Interface Control Document
IMPACT In situ Measurements of Particles and CME Transients

LET Low Energy Telescope
MISC Minimal Instruction Set Computer
SEP Solar Energetic Particles
SIT Suprathermal Ion Telescope
SEPT Solar Electron Proton Telescope
SRL Space Radiation Laboratory

2. Host System and Interfaces

2.1. *System Overview*

A short description of the SEP instrument suite may appear here (TBR-AD). Details may be found in the Phase A Report (Reference 1).

The LET, HET and SIT instruments each require a dedicated microprocessor for onboard data processing. The microprocessor used in each case will be the P24 MISC (Minimal Instruction Set Computer), described below and in Reference 4. Processed data from the MISCs associated with these three instruments will be gathered by the SEP MISC (also a P24 MISC processor), and formatted for transmission to the IMPACT DPU (per Reference 5 ICD). The SEPT instrument does not require a dedicated microprocessor, and data from SEPT will flow directly to the SEP MISC. Some processing of SEPT data will occur in the SEP MISC before the data are formatted and transmitted to the IMPACT DPU (TBR-Caltech/SEPT). Figure 1 shows a block diagram of the SEP Instrument Suite.

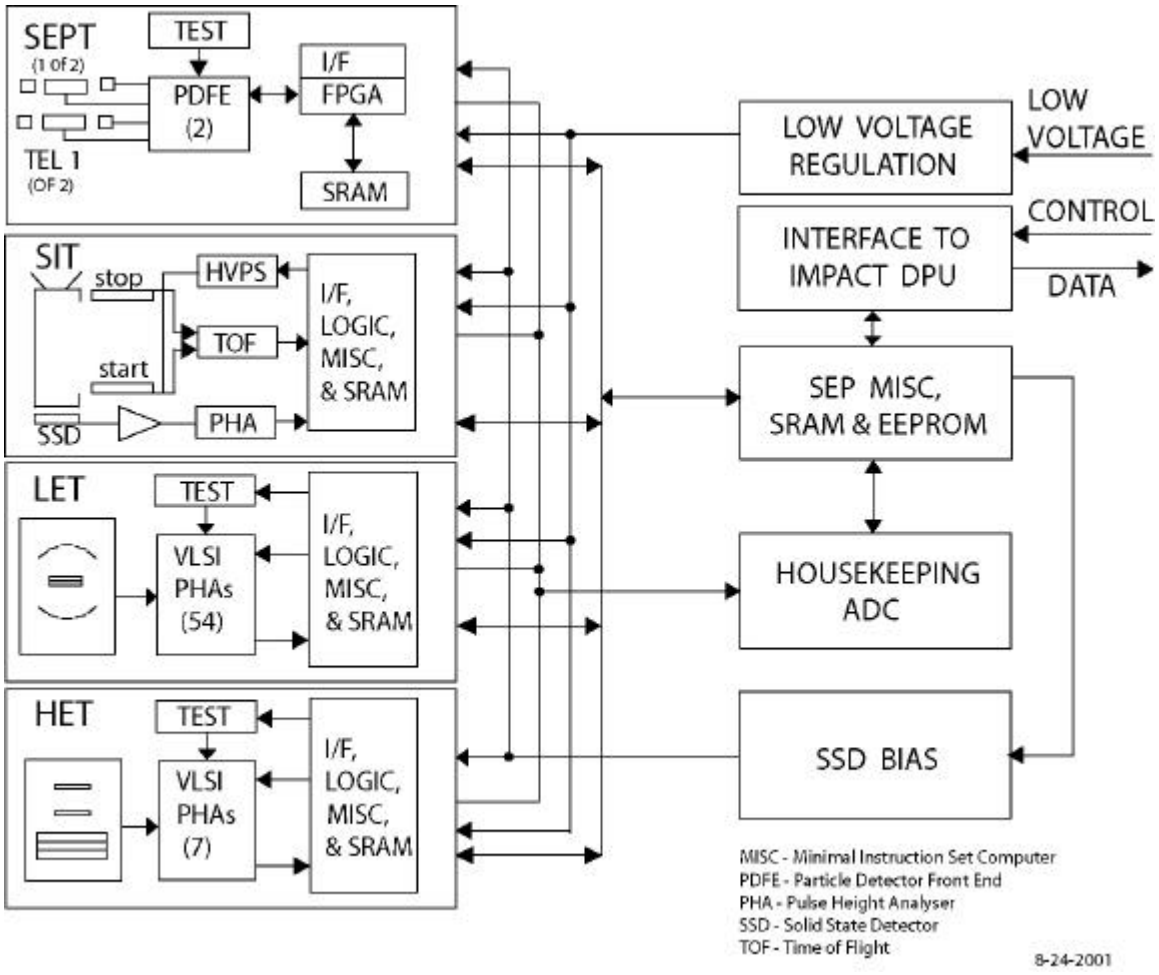

Figure 2.1: SEP Instrument Suite Block Diagram

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. For STEREO SEP applications, the MISC is implemented in the ACTEL 54SX72A FPGA. The current implementation runs at 10MHz (tested at 16MHz also).

2.2.1. **Code Memory**

Current MISC development boards are provided with 128K×24 SRAM and 128K×24 EEPROM. The MISC can boot either from EEPROM or over a serial link (configurable via jumper). During boot, system software is copied from EEPROM (or serial link) to RAM and the software runs from RAM.

2.2.2. **Memory Map**

The MISC is capable of addressing a memory page of 256Kwords (each word is 24 bits). Other pages can be reached by pushing a 24-bit address on the return stack and executing

the RET instruction, as described in Reference 4, but this capability is not expected to be required for STEREO (128K words of SRAM per MISC should be adequate).

2.2.3. I/O Bus (G-Buss) Peripherals

G-buss peripherals are described in detail in Reference 7. (TBD-WRC)
Currently, G-buss peripherals include an interrupt control and status register at address 0, supporting seven prioritized interrupts. Two of these are currently in use to support RS232 serial I/O. G-buss functions that may/will be added include (TBD-WRC):

1) A timer to produce periodic interrupts at a programmable interval.
2) Scratch-pad registers.
3) I/O ports.
4) Additional serial I/O UARTs.

2.2.4. Watchdog Timer

The processor includes a watchdog timer system that will be used to detect software crashes and either reset the processor or set a flag in telemetry (TBR-WRC).

2.2.5. Operating System

A Forth operating system with an embedded optimizing forth compiler is implemented. Multi-tasking is implemented via a round-robin system similar to the system implemented in the Harris RTX2010 microprocessor.

2.2.6. Boot PROM

Included in the FPGA implementation of the MISC are 16 words of prom that currently hold a small program to boot over the serial link. Alternatively, booting can occur directly from external EEPROM. The boot method is selected with a jumper on the MISC development board. In serial boot mode, after initial power up, the MISC will receive a certain number of bytes over the serial link. Every three bytes received are packed into a 24-bit word, with the first byte going into the most significant slot and the third byte going into the least significant slot. Words are stored beginning at address 1 in SRAM. Execution begins at address 1 following the serial transmission. The boot from EEPROM is similar, however the boot code itself is stored beginning at address $20001 near the start of the EEPROM. This boot program copies a certain number of words from EEPROM starting at $20010 to SRAM starting at address 1. After the copy it jumps to address 1. The serial versus EEPROM boot behavior is achieved by altering the memory map depending on the jumper. The MISC starts at address 0 after power on reset, so address 0 is always mapped to the first location available in the internal ACTEL prom. This prom location currently contains a jump to location $20001. Addresses $20001 through $2000F are mapped either to the internal prom, for serial boot, or to the external EEPROM, for EEPROM boot.

2.3. *External Interfaces*

2.3.1. Interface between the SEP MISC and the IMPACT DPU

The SEP MISC will interface with the IMPACT DPU via a serial interface as defined in Reference 5 (prepared by Dave Curtis of UCB). Specific commands, telemetry and data formats transferred over this interface will be defined in Reference 9, and other TBD-SEP documents.

2.3.2. Interface between the SEP MISC and the SEP instruments

There will be two low-speed serial interfaces between the four SEP instruments and the SEP MISC. 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 instruments to the SEP MISC. These interfaces will be defined in Reference 6 (TBD-Kecman/WRC).

2.4. *Hardware/Software Interfaces*

The hardware/software interfaces for the SEP and LET MISCs will be defined by Rick Cook of Caltech and documented in a SEP Flight Software Users manual and Reference 7.

The hardware/software interfaces for the HET and SIT MISCs will be defined by Rick Cook/GSFC (TBR-Caltech/GSFC) and documented in the SEP Flight Software Users manual and Reference 7.

The SEP common software will include a set of common routines that will be used to interface to the MISC unique hardware. These routines will be defined by Rick Cook in Reference 7, and in the SEP Software Users manual.

3. Software Requirements

3.1. *Top Level Requirements*

Caltech will develop three software packages for STEREO IMPACT:

1. SEP Common Software
2. SEP Central Software
3. LET Software

GSFC will develop two software packages for STEREO IMPACT:

1. HET Software
2. SIT Software

The SEP Common software consists of the Forth operating system and a set of common interface routines and utility functions applicable to all four MISC processors in the SEP instrument suite. The SEP Central software consists of the software routines unique to the SEP MISC, including the software dedicated to onboard processing of SEPT data. The LET, HET, and SIT software packages consist of the software routines unique to each instrument.

3.2. *SEP Common Software Requirements*

This section is preliminary (TBD-Caltech).

3.2.1. Forth Operating System

A version of the Forth operating system will be developed for the MISC that implements all standard Forth words necessary to implement flight software for the SEP instrument suite. This Forth system will allow for the use of assembly-language subroutines where necessary for performance considerations.

3.2.2. Interface Routines and Utility Functions

The SEP Common Software will include an API that will enable access to the I/O interfaces common to all SEP MISCs. A round-robin multi-tasking environment will also be provided.

Other requirements for the SEP Common Software are TBR-SEP.

3.3. *LET Software Requirements*

3.3.1. LET Power-On Initialization

In flight configuration, it is expected that the LET DPU will boot via the serial link from the SEP MISC. Thus, only the SEP MISC will need EEPROM during flight.

The boot process will occur in multiple stages. First the forth operating system will be transferred over the serial link and loaded (see section 2.2.6). Once forth is running on the LET MISC, transfer of the LET flight software from the SEP MISC to the LET MISC will proceed.

After the boot process, the LET MISC will configure the LET front-end electronics into a default state. The default state will be defined in tables loaded into SRAM during the boot process.

3.3.2. **Science Data Acquisition**

The LET software will be responsible for acquiring science data from the LET front-end electronics. Event control logic on the MISC FPGA will interrupt the MISC processor when event data is ready to be read into memory. Figure 3.1 illustrates the data flow logic. We may want to add more detail here.

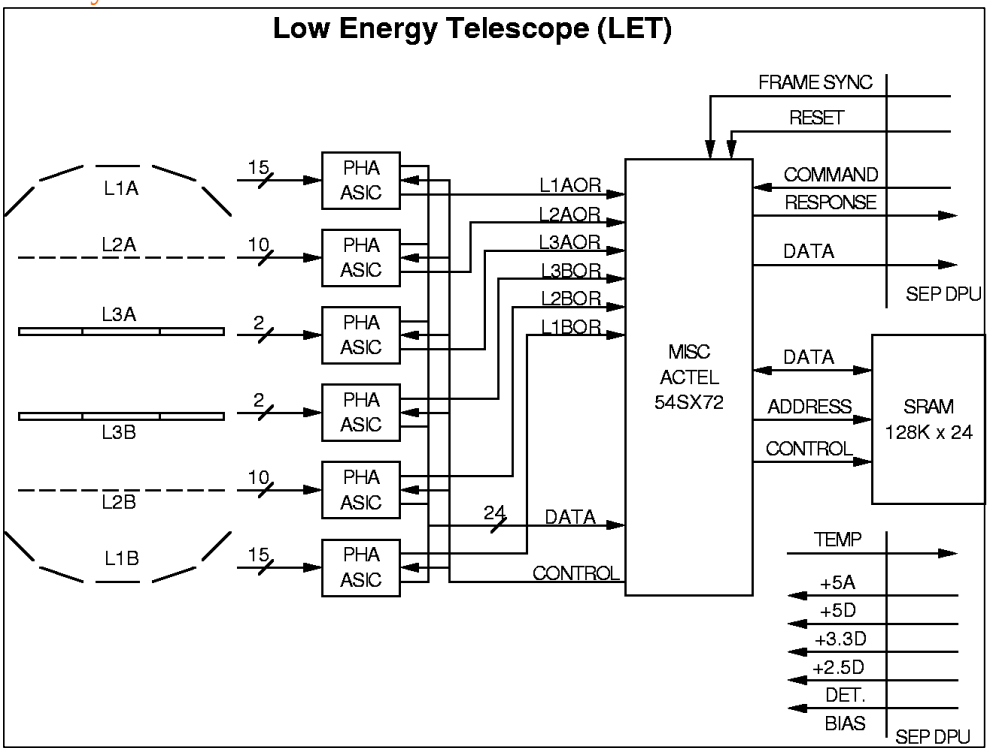

Figure 3.1: LET Logic/Data Flow

3.3.3. **Science Data Processing**

The telemetry bandwidth allocated to LET is adequate to telemeter only a fraction of the events recorded by the instrument. Therefore most events must be processed onboard, by the LET software. The objective of the onboard event processing software is to analyze the data gathered during each event and assign a species and energy to the particle that generated the event. The species and energy determination will be carried out using a 2-dimensional detector response matrix appropriate for the class of event being processed.

Once the software has determined the position of an event in the response matrix, the appropriate counter will be incremented. It is expected that the size of a response matrix will be of order 128×128, with of order 100 software bins overlaid on the matrix. Each software bin will have a counter associated with it. Note that this means that many cells of the response matrix will lie within the same software bin and will thus be associated with the same counter. The counters associated with the software bins will be read out periodically, and this constitutes the data which will be telemetered.

When the MISC receives an event interrupt, the interrupt service routine will read the pulse-height data from the LET ASICs into a FIFO. A fast-sort software routine will read

events from this FIFO and assign a class to each event. The number and makeup of these classes is TBD-LET, but the sort criteria will be based upon the event coincidence tags, the IDs of the hit detectors, and the raw pulse-height-data. The fast-sort routine will pass each event into one of a series of FIFOS, depending on the class assigned. Higher-level event processing routines will read and process events from these class FIFOs. Figure 3.2 illustrates this flow of event data through the LET software. Figure 3.3 illustrates an algorithm for one of the higher-level event processing routines.

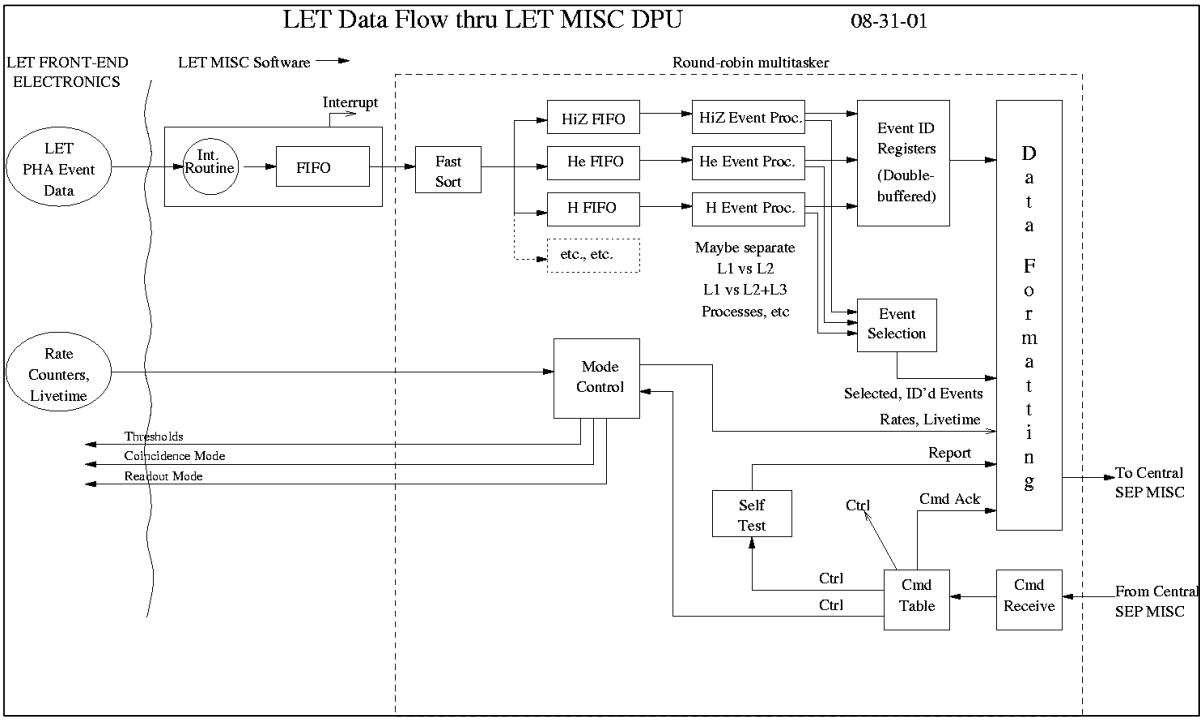

Figure 3.2: LET Data Processing

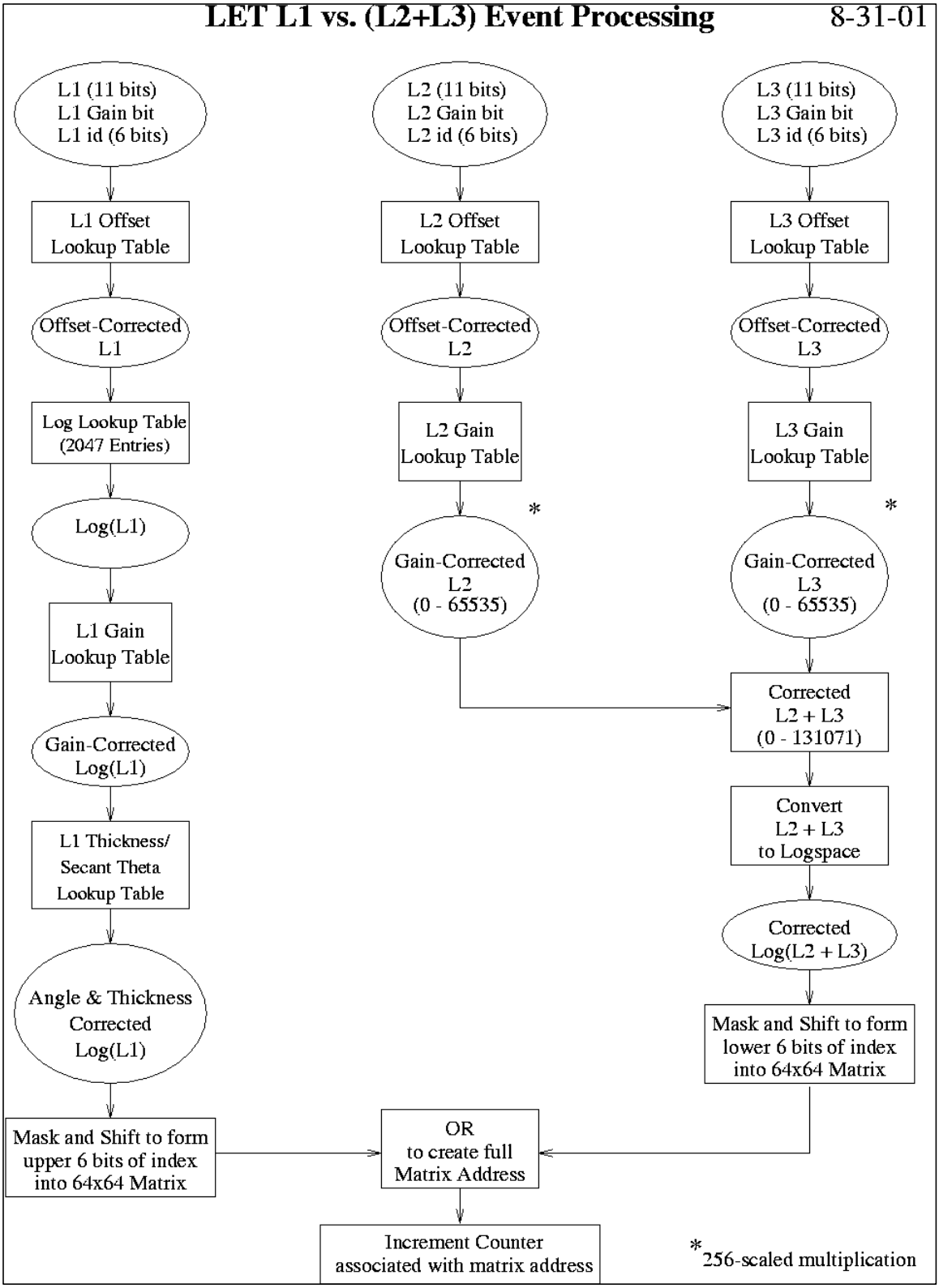

Figure 3.3: LET Event Processing Algorithm

3.3.4. **Beacon Data**

The LET software shall calculate beacon data every 60 seconds for the following species and energy ranges:

Species	Energy Range (MeV/nucleon)
Protons	1.5 - 6
He	1.5 - 6
^{3}He/^{4}He	2 - 6
CNO	1.5 - 6
CNO	10 - 30
Fe	1.5 - 6
Fe	10 - 30

LET beacon data will be transmitted to the SEP MISC, where it will be incorporated into the common SEP beacon data packet.

Formatting of beacon data is (TBD-SEP team).

3.3.5. **Housekeeping and Status Data Acquisition**

The SEP MISC will be responsible for gathering housekeeping data (voltages, currents, temperature, etc.) LET-specific software status information and the contents of command tables (TBD-LET) will be gathered by the LET software and incorporated into the LET science packets.

3.3.6. **Science Data Formatting**

LET science data formatting is described in detail in reference 6.

The LET MISC will format LET science data (rates and events) into LET science frames, and transmit the frames to the SEP MISC via the uni-directional data serial interface. Each LET science frame will contain rates and events accumulated over one minute. Each LET science frame will be sub-divided into 60 LET science minor frames. One minor frame per second will be transmitted to the SEP MISC.

Formatting of LET science frames into CCSDS packets will be performed by the SEP MISC.

3.3.7. **Command Processing**

Commands for LET will be received via the command serial interface to the SEP MISC. Command responses from LET will be sent via the command serial interface to the SEP MISC, where they will be formatted for telemetry. Error conditions will be indicated by flags in the LET science data packets.

3.3.8. **Safing**

Safing refers to any autonomous function the system requires to put the instrument in a safe mode in the event of an off-nominal condition. The conditions to be monitored include the spacecraft status (in particular shut-down warnings and thruster firing

warnings), as well as instrument housekeeping. The list of items to monitor and default actions to be taken is TBD-WRC.

3.3.9. **Reliability**

The software must be robust in handling errors or failures. It is not expected that automatic onboard correction of error conditions will be implemented, but whenever possible (consistent with instrument safety) the system should remain in an operational state following an error. Error conditions will be flagged in the science telemetry and corrective action will be initiated from the ground.

The software will maintain a command table containing all variable settings (thresholds, gains, modes, etc. (TBR-LET)). This table will be used to refresh the settings periodically. Checksums will be calculated periodically on this table, and the other lookup tables. If a checksum changes, that fact will be reported in the science telemetry.

The hardware watchdog timer shall be reset in the periodic timer interrupt handler. That module in turn shall keep track of the status of all other modules using a software watchdog scheme (TBR-WRC).

On reset the system shall start up automatically and begin operating in a default mode.

3.4. ***HET Software Requirements***

This section is TBD-GSFC.

3.5. ***SIT Software Requirements***

This section is TBD-GSFC and UMD.

3.6. *SEP MISC and SEPT Software Requirements*

This section is TBD-Caltech and SEPT team.

3.6.1. **SEP Power-On Initialization**

3.6.2. **Science Data Acquisition**

3.6.3. **Science Data Processing**

3.6.4. **Beacon Data Processing**

3.6.5. **Housekeeping and Status Data Acquisition**

3.6.6. **Data Formatting**

3.6.7. **Command Processing**

3.6.8. **Safing**

3.6.9. **Reliability**

4. Software Development

"Top-down" and "Bottom-up" approaches to software development will run in parallel for the SEP instrument suite. 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 in 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 processing algorithms to verify feasibility of MISC hardware approach

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, SEP team members will develop a set of science requirements for each of the SEP instruments. These requirements will be recorded in a science requirements document for each instrument. At the same time, the SEP team will also develop preliminary designs of the instruments and their front-end electronics.

Using the science requirements, the preliminary instrument and electronics designs, and consultations with other SEP team members, SEP engineers and developers will derive a set of software requirements for each instrument MISC and the SEP MISC. These 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 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 Section 3 of the Software Development Plan (this document).

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
- descriptions of all inputs, outputs and data formats
- data processing algorithms
- ICD between the SEP MISC and the SEP instruments
- P24 MISC processor manual
- P24 MISC G-buss I/O Interface Document
- Command lists
- other TBD design specs

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. The rest of this section is TBD-AD, and will describe coding guidelines and standards to be followed.

4.1.4. System Testing and Acceptance Phase

End-to-end instrument, electronics, and software functional tests will be done using accelerator tests, radiation sources, built-in self-test routines and test procedures. Software walkthroughs will occur at at software peer reviews, reports will be presented at PDR and CDR.

4.2. *Development Environment and Equipment Needed*

This section is TBD-AD and will cover the following:
Describe software tools required to develop and test the software (i.e., compilers, assemblers, debuggers, code editors, documentation tools, etc.)

Describe the equipment needed to develop and test the software (i.e, breadboards, ground system GSE, development stations, networks, servers)
ETU's, GSE, compilers, debug, etc.

Describe whether tools and development equipment are existing and available or will the equipment be purchased for this project. If provided by others then define who is providing the equipment and when.

4.3. *Product Assurance*

The following is based on the approach taken on ACE for software configuration control and flight software integrity assurance. It is expected that a similar approach will be taken for STEREO SEP. Since Caltech and GSFC have not yet discussed a common approach, the following currently applies only to Caltech. It is appropriate when there are only two or three programmers, and each has well-defined sections. GSFC needs to review this section and either contribute a GSFC version, or modifications to this version.

4.3.1. Software Configuration Management/Backup Plan

- Only one person will be authorized to load flight software into flight CPUs – the lead electrical engineer.
- The lead engineer will maintain separate directories for the LET DPU and SEP DPU flight software. The directory name will include the date the code was last modified.
- At the beginning of any day on which the lead engineer will be working on the flight software, the flight software directories will be duplicated, and the new

directories renamed with the current date. Any changes will be made within the new directories.

- Beginning in the implementation phase, a change log will be maintained by the lead engineer containing the date of any change, a description of the change, and the reason for the change.
- At the end of each day of software work, the lead engineer will backup the current flight software directories to a different computer and/or to removable media.
- One or two software developers will be working under the lead engineer. Each developer will be required to implement version control for his/her portion of the code.
- Periodically, a software developer will deliver his/her portion of the code to the lead engineer, who will incorporate it into the main software package.
- 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 EEPROM burn-in process, checksums will be calculated and recorded. During boot of a flight CPU, the EEPROMS 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.

4.3.2. **Walkthroughs**

This section is TBD-AD/WRC and will cover the following:
Describe what types of walkthroughs are planned and what code and or design products will be subject to the walkthrough process. Describe special techniques if any (i.e. Code Inspections). For instrument flight software code walkthroughs should be conducted on all mission critical flight software. Mission critical flight software with sufficient flight heritage might be exempted from the code walkthrough process.

4.3.3. **Test Plan**

This section is TBD-AD/WRC and will cover the following:
This section should describe the plans for testing the software. Methods, procedures, and products should be defined for the different phases of testing including unit test, build test, and acceptance test. Description should include the use of the requirements/test verification matrix, negative testing methods, use of automated procedures, test result reports, the level of independence of test team, etc.

4.3.4. **Software Problem Reporting and Tracking**

This section is TBD-AD/WRC and will cover the following:
Describe how software problems will be tracked including any tools which are planned such as MS Access Database for tracking and reporting discrepancies.

4.3.5. **Risk Assessment**

This section is TBD-AD/WRC and will cover the following kinds of issues:
single programmer for many critical software components…???

4.3.6. **Software Maintenance**

This section is TBD-AD/WRC - Describe plans for on-orbit software maintenance.

5. Management Plan

5.1. *Build Plan*

This section is TBD-AD/WRC. The Project wants the following:
Identify dependencies -- i.e. on time breadboard delivery, hardware spec completed, etc.
Identify Builds and describe build contents.

5.2. *Reviews*

This section is TBD-AD/WRC.
Suggested List

- *Software Requirements Review*
- *Software Design Review*
- *Code Walkthroughs*
- *Software Acceptance Review*

Present current software metrics: Estimated CPU, memory, EEPROM margins at PDR, CDR, and SAR.

5.3. *Documents and Source Code*

This section is TBD-AD/WRC.
Suggested List

1. *Software Development Plan -- Complete plan at PDR, update for CDR*
2. *Software Source Code & Description document -- with each build*
3. *Software Test Procedures -- may be co-developed as instrument procedures*
4. *Software Requirements Traceability Matrix – update monthly*
5. *Software Users Guide -- cmd & tlm descriptions at least*

Monthly status reports to include list of outstanding DRs, PTRs, updated RTM

5.4. *Heritage and Reuse*

Previously, the Caltech group developed the flight software for the SIS and CRIS instruments on ACE, and for several balloon instruments. The GSFC group developed the flight software for the EPACT instrument suite on WIND. Many algorithms and software routines developed for thee projects will be reused for STEREO SEP.

5.5. *Manpower*

This section is TBD-AD/Alan Cummings
Provide quarterly estimates of manpower for the life of the project.

5.6. *Staffing Plan*

Caltech – LET DPU and SEP DPU software.
Rick Cook – Lead Electronics Engineer
Andrew Davis, Allan Labrador – Software development

GSFC – HET DPU and SIT DPU Software.
Don Reames, Kristen Wortman, Bob Baker

JPL – SEP GSE software
Robert Radocinsky, Mark Wiedenbeck

Algorithm development for onboard data processing will involve extensive cross-collaboration between the SEP team members.

5.7. *Interaction between Caltech and GSFC*

This section is TBR-Caltech/GSFC.
GSFC will develop flight software for the SIT and HET MISCs, using manuals and development tools provided by Caltech.

An ICD (Reference 6) will describe the communications interface between the SEP instruments and the SEP DPU, with adequate information to allow GSFC to design and build hardware and software that utilizes the interface.

Prior to SEP integration, Caltech will provide a SEP DPU to GSFC to allow testing of the interface between HET, SIT and the SEP DPU.

5.8. *Schedule*

This section is TBD-AD/Alan Cummings

- *Documentation release schedule*
- *Software Review schedule*
- *Development environment and tool procurement schedule*
- *Dependencies Schedule -- i.e. ETU delivered for software I&T*

- *Build Schedule*

Build Test and Acceptance Test Schedule