# Onboard Event Processing Scheme for the HET and LET1, and LET2 Sensors on Epi-Hi

Andrew Davis, Version 4, Jan 2014

## Changelog

**Version 4**: Jan 2014: Added Appendix describing proposed Event Categories

**Version 3**: Sept 2013: Added Appendix describing proposed species ID Algorithm

## Introduction

This document is intended to document the event processing scheme for the LET1, LET2 and HET sensors on the Epi-Hi instrument.

### Event Processing Requirements:

* Identify ion Charge (Z), and (for selected Zs) Mass (M), given the energy (dE, in MeV) it deposits in a detector of thickness L of silicon, and its residual energy (Ep, in MeV) when it exits that detector. i.e., we know dE, Ep, and L. Energy range ~1 – 100 MeV/nuc. (species-dependent energy-range, exact ranges TBR).
* Identify electrons with an energy range ~0.5 – 6 MeV (exact range TBR).
* Identify ENAs triggering deep in the detector stack.
* Calculate the energy of each identified ion and electron with the accuracy required to fulfill the L4 requirements.
* Sort identified ions and electrons into energy, angular-distribution, and time bins per the L4 requirements.
* Be able to process at least [300, TBD] helium events per second, including both Z and M calculations. Since helium (and neon) event processing requires the most CPU resources, other mixes of incoming events will be processed faster than this. Note: the maximum coincidence event rate incoming through the front-end electronics FIFO will be ~1000 - 2000/sec (exact number TBD).

### CPU and RAM Resources

**RAM:** 512Kwords total available (each word is 24 bits wide)

**CPU:** Perhaps it is best to illustrate the MISC CPU constraints with examples. These examples apply to the current version of the MISC floating-point library. The MISC can perform the following operations at the following rates:

* Integer multiply (or divide): ~218,000/sec
* Floating-point multiply (or divide): ~59,000/sec
* Floating-point add (or subtract): ~40,000/sec
* Natural Logarithm: ~714/sec
* Ax: ~550/sec

These are rough numbers; there is some dependence on the values of the inputs. Clearly, logs and powers, if needed, must be implemented using table lookup or approximation. Floating-point add/subtract and multiply/divide operations must be minimized.

## Event Processing Overview

The overall architecture of the event processing task, and the method by which it is called from the main round-robin multitasking loop, is the same as for STEREO LET.

Within the flight software of each of the Epi-Hi sensors, the event processing software exists as a task (or module). Let’s call the task EPROC. Periodically, a governing task polls the incoming event FIFO, and, if an event is available in the FIFO, the EPROC task is called to process the event. Based on analysis of the event, EPROC increments certain counters, and sets the “priority” of the event. The EPROC task then exits.

Other tasks manage the counters, and manage the selection of events for telemetry based on the event priorities set by EPROC. These tasks are not discussed here.

Ignoring STIM events and HAZARD flags for brevity, The EPROC Task can be broken into a series of sub-tasks. A proposed set of subtasks (derived from the STEREO LET event processing) follows:

* XTALK: check cross-talk flags/data generated by the PHASICs for this event. Check for other known cross-talk features.
* SORT: sort events into categories, e.g. RNG2A, RNG3B, etc.
* MULTI: handle events with multiple hits in a detector layer.
* EUNPACK: Select dE, Ep, L, values, for use by SCALC task
	+ For deeper ranges, there may be multiple sets of dE, Ep, L
* CRIT: apply any trajectory or other consistency criteria (these will be range-dependent).
* SCALC: For each (dE, Ep, L), identify the species (electron, proton, 3He, 4He, C, …)
* CRIT2: apply consistency criteria for events with multiple sets of dE, Ep, L
* ECALC: For electrons and protons, calculate energy (ETOT). For Z>=2, calculate ETOT/M.
* Neutrals: Neutrals processing task, selecting ENA candidates

These sub-tasks are called sequentially within EPROC. At each stage in the sequence, a decision is made whether to continue to the next stage, or to halt the processing of the event. If the decision is to halt, then the appropriate counters are incremented, and the appropriate event priority is set. No events go uncounted.

We now discuss each of these sub-tasks in detail.

### XTALK

The crosstalk information contained in the raw event data generated by the PHASICs is examined. See the latest version of the PHASIC manual for details. Based on the results, the processing of the event may be halted.

Experience may dictate that further analysis of actual event data may be needed to filter out crosstalk. This analysis would be similar to that done for STEREO LET, but might be more complicated for Epi-Hi. This is TBD.

### SORT

Assign each event a category based on the ADC hit pattern and the coincidence-logic flags. An example category would be “Range 2, A-side, 0 degrees”.

Each raw event that the flight software reads from the event FIFO will have passed the coincidence-logic rules implemented jointly in the PHASICs and the FPGA firmware.

The data for each event consists of a block of N ADC values that have been read out from the PHASICs, a list of the addresses of the N ADC values, a list gain-level flags, and a set of TBD flags generated by the coincidence-logic firmware.

As of this version (4), the coincidence-logic equations to be implemented in the FPGA firmware are not yet fully-defined. So we don’t yet know all the inputs available to the SORT software subtask. However, we can define the list of event categories that we want the events to be sorted into, from a science perspective, and this should help to define the coincidence-logic equations.

See Appendix A for the proposed flight software event category list.

### MULTI

The flight software must handle multiple hits in the same detector layer. For STEREO LET, multiple non-crosstalk hits in a layer resulted in a halt of further processing of the event. We must decide what the appropriate actions are for the Epi-Hi sensors. There may be different rules for each layer in each sensor. All this is TBD.

### EUNPACK

Fully unpack the raw event data into data structures suitable for further event processing. For charged-particle categories, identify dE, Ep, L, values, for use by SCALC module. For deeper ranges, there may be multiple sets of dE, Ep, L, values.

### CRIT

Apply any trajectory or other consistency criteria (these will be range-dependent). These are TBD.

### SCALC

For charged-particle categories: For each set of dE, Ep, L values, separate electrons from ions. For ions, identify the species.

See Appendix B for the proposed flight software species ID algorithm.

### CRIT2

For charged-particle categories: For events with multiple sets of dE, Ep, L values, apply consistency criteria. These are TBD.

### ECALC

For electrons and protons, calculate ETOT. For Z>=2, calculate ETOT/M.

### ENA

For ENAs, do something TBD.

## Appendix A: Proposed List of Event Categories

The SORT subtask assigns each incoming event to a category via a TBD process. The category determines how the processing of the event proceeds in the flight software. For each telescope (LET1, LET2, HET) we can divide the list of categories into three groups:

* Charged Particles
* ENAs
* Rejects (events that fail the SORT process)

## LET1 Categories

### Charged Particles

Since LET1 is double-ended, it is possible for an event to trigger both ends of the instrument. In this case, the event could consist of a single penetrating particle, or a chance- coincidence of multiple particles. The SORT task must sort this out. If the event appears to be a single-particle penetrator, then we proceed on that basis. If it appears to be a multiple-particle event, then the choices are a) assign the event to the Reject group, or b) split the event into two. For STEREO LET, the onboard software rejected multiple-particle events.

For each of the main categories below, I list a “Coincidence Requirement”. This is not necessarily what will be encoded in the coincidence logic implemented in the PHASICs /FPGA (I don’t know what those rules will be), but it IS what the software SORT task will require.

Only the A-side categories are listed, the B-side versions may be inferred from the A-side versions.

#### Range 1 A

These are particles that stop in L1A.

Coincidence requirement: L0A.L1A.~L2A

Other probable requirements:

* Along with ~L2A, require no hits deeper thanL2A
* Only L0A-L1A detector combinations that result in a trajectory that includes the active area of L2A are acceptable. Events with bad trajectories will be assigned to a Reject category.

Issues: the geometry factor for acceptable Range 1 events is quite different from deeper-range events. Thus Range 1 events likely need a separate set of counting bins from events in deeper ranges.

#### Range 2 A

These are particles that stop in L2A.

Basic coincidence requirement: L0A.L1A.L2A.~L3A

OR ~L0A.L1A.L2A.~L3A

Other probable requirements:

* Require ~L2Ag
* Along with ~3A, require no hits deeper thanL3A
* TBD consistency between L0A and L1A signals (if L0A signal is present)

Issues:

* The geometry factors for L0A.L1A.L2A and L1A.L2A triggers are different. Thus we may need 2 sets of counting bins, one for each trigger mode.
* Under what circumstances do we allow the ~L0A trigger? Given the proposed collimators at the L0 and L1 levels, and the coarse trajectory information, it will be difficult to determine the amount of detector material traversed by particles that do not trigger L0A, especially those that would be below threshold in L0A anyway. Even if we restrict the L1.L2 trigger to the center bulls-eyes of L1 and L2, the collimators at L0 are still in the field of view.

#### Range 3 A

These are particles that stop in L3A.

Basic coincidence requirement: L0A.L1A.L2A.L3A.~L4A

OR ~L0A.L1A.L2A.L3A.~L4A

Other probable requirements:

* Require ~L2Ag.
* Require ~L3Ag???
* Along with ~L4A, require no hits deeper thanL4A
* TBD consistency between L0A and L1A signals (if L0A signal is present)
* TBD consistency between L1A and L2A signals (if L0A signal is present)

Issues:

* Same issues as for Range 2 A
* The geometry factor for Range 3 is slightly different from the geometry factor for Range 2. In the flight software, events are counted in energy/nucleon bins. Each bin mau have a contribution from several ranges. In ground analysis, we will need to calculate a unique geometry factor for each species/energy bin to account for this. I think this also is a good reason to keep separate sets of counting bins for each trigger mode.

### Rejects

These are events that fail the SORT process. Events may also be rejected by the XTALK or the SCALC process, and events rejected by those tasks will also be assigned a category from this list.

* xxx

## Appendix B: Proposed Flight Software Species Identification Algorithm

We assume that the algorithm has the following inputs: dE, Ep, and L.

Recall that L is the amount of material traversed in the dE detector. (It is NOT the thickness of the detector.) Ignoring any detector thickness variations, L is quantized. For LET1 (and LET2), there are 4 values of L for each Range. Thus, if there are 7 Ranges in LET1, there could be up to 28 unique Ls to consider. In practice, many of these 28 Ls will have the same nominal value.

For STEREO HET/LET, for a given Range, we identified species by mapping the dE and Ep signals into a 400x128 2D matrix. We would ideally have used a unique matrix for each L, but there was not enough RAM, so we had to “angle-correct” the dE and Ep signals, so that we could use a single 2D matrix for all Ls in a given Range.

In this algorithm, instead of defining a 2D matrix for each L, we instead define a set of 1D “boundary tables” in dE-Ep space. For each species, two 1D boundary tables are required to define the region in dE-Ep space occupied by that species. A 3rd boundary table is also required, but it is common to all species, and defines the (roughly) diagonal cut-line in dE-Ep space that separates forward-going from backward-going particles. In some cases, adjacent species may share a boundary table.

For several reasons, it is useful to work in a 2D “log-space”, as was done for STEREO. So, for instance, we transform an L1 signal to log-space as follows:

logL1 = INT\*( N ( log(L1) - log(L1THRESH) ) ), where N = DIM/log(DYNAMIC\_RANGE)

L1THRESH is the nominal lowest-possible L1 threshold, DYNAMIC\_RANGE = Full\_scale/L1THRESH, and DIM is the value that we want for logL1 at full-scale. INT\* indicates an integer function with proper rounding.

Note that the transformation of dE and Ep to log-space is done by table-lookup in flight software. There are no logarithm calculations done in the flight software. A separate log-lookup table is required or each (Full\_scale, THRESH) pair, but each table needs only 2048 entries (we have 11-bit ADCs).

Once a dE or Ep signal is transformed to log-space, any required gain or detector-thickness corrections are done as “add” operations, rather than “multiply” operations.

The 1D boundary tables are also defined in this dE-Ep log-space. To 1st order, these boundary tables can be calculated from L, using range-energy software, but further empirical tweaking will be required to optimally fit empirical data. The number of entries in each boundary table is set be the DIM used in the transformation of Ep signals to log-space, and the Ep energies used to compute the tables are chosen to be evenly-spaced in logEp space.

Figures 1 and 2 show what this all looks like, for LET 1, Range=2 (L=25um), and with DIM=2048 for dE, and DIM=512 for Ep. The boundaries shown are calculated from Mark’s range-energy software, and are only approximately correct at this time.



***Figure 1:*** *EPI-Hi LET Monte Carlo Range 2 He events, plotted in 512×2048 log-space, with species boundary-curves superimposed. The boundaries shown would allow identification of p, 3He, 4He, C, N, O, Ne, Na, Mg, Al, Si, Ar, S, Ca, Fe, and Ni. Boundaries for e- not shown. NOTE: the figure is rotated to fit!*



***Figure 2:*** *EPI-Hi LET Monte Carlo Range 2 He events, plotted in 512×2048 log-space, with species boundary-curves superimposed. Same as Figure 1, but zoomed (and rotated) to show He data.*

The data plotted in Figures 1 and 2 are results from Forth event-processing software running on a MISC test unit, with “hi-res” He data from Mark’s Monte Carlo used as input. Raw dE and Ep signals (channels) from the Monte Carlo output were fed to the Forth software on the MISC. For each event, the software calculated logdE, logEP, and identified the species. Events identified as 4He by the MISC software are colored blue in the plots. Events identified as 3He are colored black. Events failing the “penetrating” cut are colored red.

### MISC Performance and Memory requirements

From figures 1 and 2, it appears that ~512 entries in each boundary-table will be adequate (DIM=512 for logEp). Thus, each 1D boundary-table will require 512 words. Each species will require 2 boundary tables. Assuming we define ~22 species, the boundary-table RAM requirements for a given L will be 22×2×512 = 22.5Kwords. Let’s round up to 23Kwords to allow for defining the penetrating cut-line. This overstates the RAM requirements, since some adjacent species will have a common boundary.

We have 512Kwords available. Thus, we may define boundaries for 12 unique Ls using a little more than half the available RAM.

Note: we could conserve RAM (at expense of CPU cycles) by storing two table-entries per 24-bit word.

Given dE, Ep (raw channel values), and Range, the current MISC Forth software can process events at a rate of ~9,600/second.

The current event-processing test-code looks like this:

 FOR Each Event i

 From dE, calculate logdE

 From EP, calculate logEp

 IF (Penetrating) Then

 Species(i) = PEN

 ELSE

 Species(i) = FindBoundary

ENDIF

 ENDFOR

Calculating logdE (or logEp) includes the following operations:

 Subtract channel offset

 Check ADC bounds ( 0 < ADC < 2047 )

 Lookup log(dE) via table lookup

 Apply Gain and thickness corrections (ADDs in log-space)

 Subtract log(THRESH)

The FindBoundary routine performs a binary-search through the boundary-tables, using logEp as the index. Z=1 boundaries are checked prior to beginning the binary-search.