CLAS-NOTE-94-024

Ec 3.2.1

FORWARD ELECTROMAGNETIC CALORIMETER
RECONSTRUCTION SOFTWARE

K.B.BEARD

JANUARY 16, 1995

Ec 3.2.1 is the current release of the CLAS forward electromagnetic calorimeter (Ec) event reconstruction software. It incorporates all essential functions and required data structures and supports both the CLAS-CODA and BOS event stuctures. This code should easily link to whatever CLAS shell eventually emerges; its modular structure allows easy modification of algorithms, and its current algorithms already deal with the problems of complicated hadronic events and dead channels.

1 Introduction

The CEBAF Large Angle Spectrometer (CLAS) consists of multiple detector packages; each package requires unique software for event reconstruction. To successfully produce an integrated analysis environment, these packages must cooperate with one another and conform to a common set of standards.

The Ec [1] package was conceived to link into some sort of yet unspecified host program; the **Ec.exe** program allows nearly all functions to be exercised one event at a time, while the **Ec_engineH.exe** and **Ec_engineI.exe** programs call those routines from a simple shell. The purpose of these programs is to test the Ec package on a large number of GEANT created events. The data input to Ec is via CODA-CLAS 2.0 event buffers [2] or CLAS BOS 0. format [3], the output is via the EcAcc (**Ec Access**) routines, the EcRsl (**Ec Results**) common blocks, and BOS banks.

Early testing with data taken at Brookhaven National Laboratory with the prototype indicated the necessity of dealing with complicated hadronic events in a systematic way "up front", rather than treating them as a special case. It is also expected that not all channels in the calorimeters will be working all the time; the problem of these "dead" channels cannot be ignored if the reconstructions are to be useful.

2 Important Changes Between Ec3.2.1 and Ec3.1.3

Ec no longer is dependent on the author's personal library; all the utility routines are in the SwGus library [4]. The problem of large memory requirements for pixel routines has been addressed by using the FORTRAN parameter **Ec_MAXpixels** in **Ec_general.PAR** to control both the memory size and the response of all pixel routines to limited memory; all pixel routines will do the best they can with whatever memory is available.

[1] Ec 3.1.3, Forward Electromagnetic Calorimeter Reconstruction Software, CEBAF CLAS-NOTE-94-002 (1994)

[2] CLAS Event Format, Version 2.00, CLAS-NOTE 93-002, March 24, 1993

[3] CLAS Event Format with BOS, Version 1.00, CLAS-NOTE 94-012

[4] SwGus 1.1.2, SoftWare group General Utility Software, CLAS-NOTE 93-023

3 Important Concepts

3.1 Philosophy

The goal of the Ec package is to accomplish correct reconstruction of CLAS forward electromagnetic calorimeter events using clean, structured, maintainable, and documented code. Conceptually, information at each stage of analysis is stored in common blocks; the subroutines are operators used on one common to fill another; and only instructions and status information are passed as arguments. The parameters are carefully chosen and ordered to characterize all information about objects within commons, and the commons constructed to organize the analysis. This structure may be easily modified by adding or deleting parameters.

Ec has been made as modular as possible; it is intended to be customized by including only those pieces required at any time. Each logical operation is a subroutine; each subroutine is a separate file to simplify debugging and upgrades.

3.2 Coding Comments

Ec has been written to conform to the CLAS software standards [5] with very few exceptions. Standardized nomenclature has been used throughout Ec; the first two letters identify the package, the next three the functionality.

The Ec package compiles and runs correctly under DEC-VMS, DEC-Ultrix, and HP-UX; no routine should be specific to a platform.

Most subroutines use the SwGusTrace routines to track of the time spent in each routine; all I/O control and error reporting is through SwGus utilities.

3.3 Terminology and Units

For the purpose of Ec, many details of the detectors construction are ignored. Only the active region is considered, and the detector is considered homogenous. Each calorimeter is divided into an *inner* and an *outer layer*; the volume of space viewed by a single phototube is called a *strip*. The sides

[5] The Hitchhiker's Guide to the Galaxy: CLAS Software Manual (Rev 1.0), CLAS-NOTE-90-008, July 16, 1990

of the calorimeter are denoted *U, V,* and *W*. The shortest strip is numbered *1*.

The Ec package (Figure 1.a and 1.b) uses the following name conventions; a *pixel* is the volume formed by the intersection of a U, V, and W strip. A *pseudopixel* is the volume formed from a (triangular) group of pixels (pixel#1297 contains pixel#1-4). A *hit* is a reconstruction of energy deposited which was read by intersecting U, V, and W strips within a layer. A *shower* is formed by the overlap (in local IJ space) of an inner and outer hit. Note that such a shower may represent an electromagnetic shower, a minimum ionizing path, or any event depositing energy in both layers. Events corresponding to energy which is measured in a strip, but cannot be represented by intersecting U, V, and W strips, are called *unrecon*, short for unreconstructable.

The CLAS XYZ coordinate system is right handed with Z parallel to the beam axis, X horizontal through the center of sector 1, and the origin at the nominal target position. The CLAS spherical system is the normal compliment to the XYZ system.

The sector coordinate system S123 is right handed and sector dependent with S1 parallel to the beam and S2 through the center of the sector.

The local right handed coordinate system is IJK and has its origin such that K is normal into the inner face of the calorimeter and parallel to the radial vector from the target. I is away from the beam in the plane of the inner face of the calorimeter. The local UVW system is the nonorthogonal system measured along the edge of the calorimeter.

All units are the CLAS standard; angles in radians, energy in GeV, distances in centimeters, and time in nanoseconds.

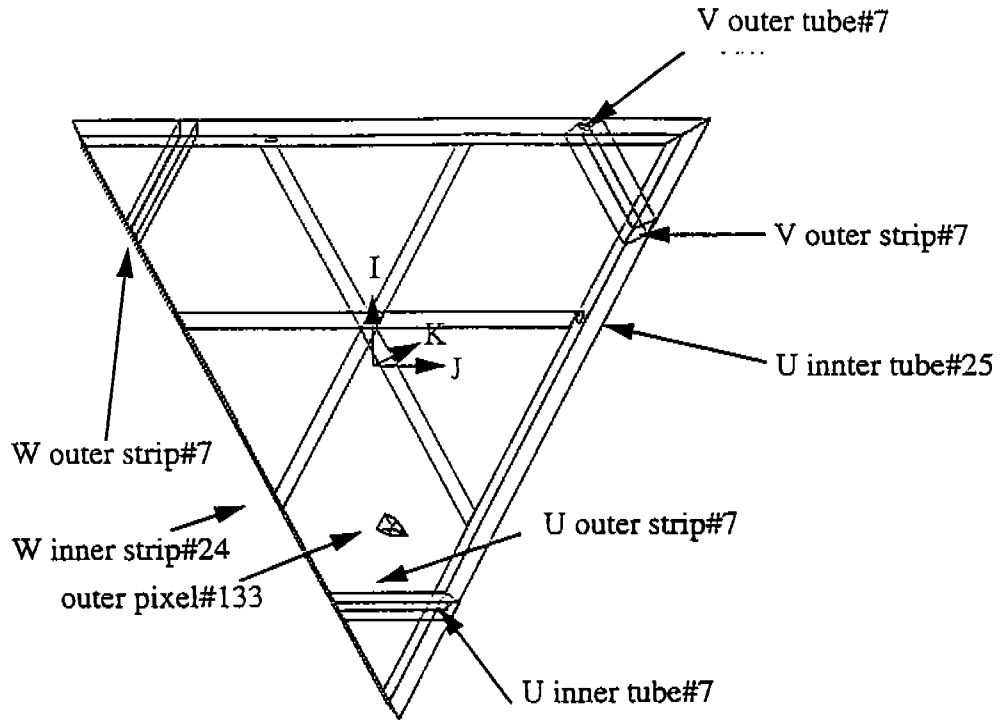


Figure 1a. Ec objects

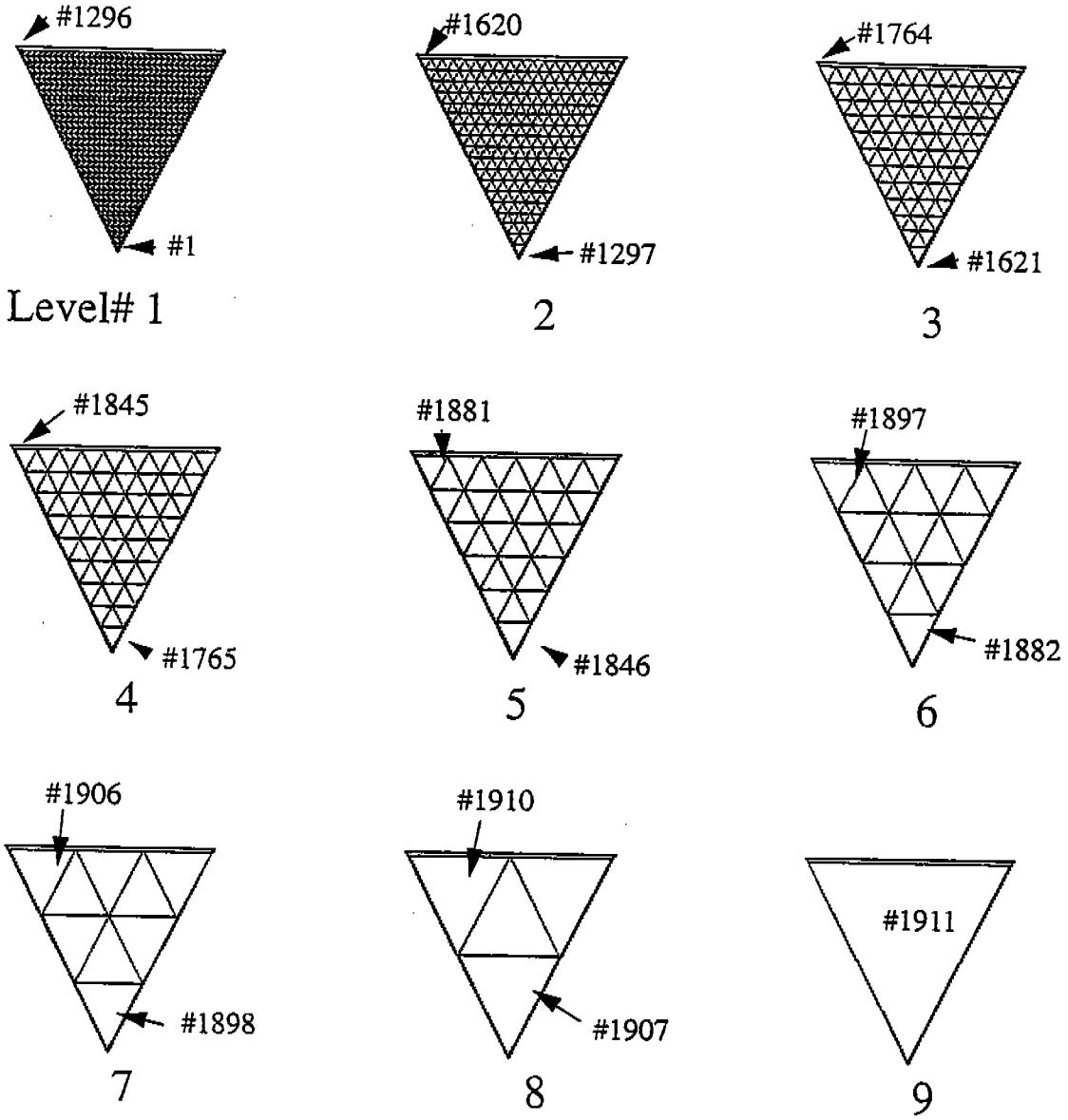


Figure 1b. Pseudopixels.

3.4 Error Messages

The Ec package is still being developed; in particular the reconstruction algorithms are being improved and made more robust with respect to missing information (dead photomultiplier channels) and much faster. This release carries a considerable burden of error checking; as the correctness of the code is verified these routines will gradually be removed in newer releases. The *OK* status returned by most subroutines and the *err* messages are primarily to detect flawed code; the path to the error as well as the error is returned. In general, correct code should never produce an *OK=.FALSE.* status unless it cannot continue. The SwGus utilities are used to prepend error messages; for example:

```
ERROR: Ec_recon_BOS>EcFit_analyze>EcFitMetro>EcFit_FuzzyMetropolis>
       EcFit_pixel_masked:unable to mask any pixels level#1
```

4 Input

The only supported data inputs are CODA-CLAS 2.0 event buffers or via BOS banks. Commands may be issued directly by calling subroutines or by using the *pseudo_Ec* interface. **Ec.exe** allows all Ec functions to be exercised, **Ec_engineH.exe** reads CLAS-CODA 2.0 format files and **Ec_engineI.exe** reads BOS 0 format files.

The interactive interface is one taken from the *pseudoQ* package [6]; it allows deeply nested command files and is generally very similar to the LAMPF Q system interface [7].

It is not necessary to use the pseudoQ interfaces at all; they are provided merely as a convenient way to communicate with the package.

5 Output

The results of Ec may be retrieved via BOS banks, the *EcAccess* routines, or the *EcRsl* common.

[6] CAMAC_chat, A CAMAC Communication and Software Development Tool, CLAS-NOTE-91-026, January 14, 1992

[7] Q Release Notes and Distribution Information, Release: March 17, 1990, Document MP-1-3413-6, Los Alamos National Laboratory.

Descriptions of the BOS banks appear in the appendix and CLAS-NOTE 94-012. All Ec generated banks use *SwGus_BOS_comment* for documentation.

The **Ec_engineI.exe** program is an example of using the EcAcc (**Ec Access**) routines and putting the results into an CERNLIB Ntuple [8]. The interactive routine *pseudo_EcAcc* will report type and quantity of information available on specific objects within a common. It can be used to build a *vector*; a list of characteristics of a specified object within a common that can subsequently be packed into any user array. This technique decouples details of the Ec package (the ordering of characteristics, for example) from the host code. **Ec_engineI.exe** extracts information from the simulated event , calculates a few quantities based on that, and packs the resulting data and concatenates all vectors to form a single CERNLIB Ntuple. The resultant Ntuple can be used by PAW [9], CERNLIB's interactive routine to project histograms and make cuts.

The **only** direct access to any Ec common should be the EcRsl (**Ec Results**) common. The include files are **EcRsl_export.PAR**, **EcRsl_status.CMN**, **EcRsl_export.CMN**, and **EcRsl_par_desc.CMN**, they define the information content, structure, meaning, and status of the output results. The status should always be checked before the results are used.

6 Graphics

The EcGra (**Ec Graphics**) subpackage was developed to support software development, and be compatible with a general purpose host. It is based on the CERNLIB HIGZ package [10] and assumes the set window scale is the physical scale (all scaling, offsets, etc. are done by HIGZ behind the scenes). In this way various packages' graphics can be easily superimposed without communication between the packages. The display options are preselected by a call to *pseudo_EcGra* or *EcGra_option*.

[8] HBOOK User Guide, Version 4, Y250, October 28, 1987, CERN Computer Centre Programming Library

[9] PAW - Physics Analysis Workstation, The Complete Reference, Version 1.07, Q121, October 1989, CERN Programming Library

[10] HIGZ - High level Interface to Graphics and Zebra, Q120, March 10, 1988, CERN Computer Centre Programming Library

The graphics are fully three dimensional; the CLAS XYZ, sector S123, and Ec IJK coordinate systems are supported as are rotations about the vertical and horizontal axes; hidden lines and color are not supported. Sectors, layers, and objects may be plotted independently. A common energy scale is used throughout but may be changed at any time. The call is of the form:

```
call EcGra_plot(layer, sector)
```

For both layers and all sectors, specify *layer=0* and *sector=0*.

7 Simulations

These internal simulations are crude parameterizations and are for internal testing **only**; detailed GEANT simulations [11] should be used to produce realistic events.

8 Reconstruction

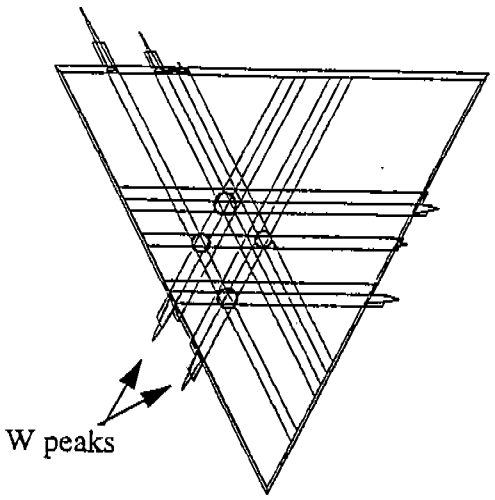
The reconstruction of an interaction within the forward calorimeter may be viewed as producing 2-dimensional maps of the calorimeter with energy density and time distributed over the inner and outer layers. The are four reconstruction algorithms supported in this version of Ec; **Quick**, **Fast**, **Edge1**, and **Metro**.

Electromagnetic events tend to be simple, while hadronic events often deposit energy unevenly over a large areas. Complicated pion events are shown in Figures 2 and 3.

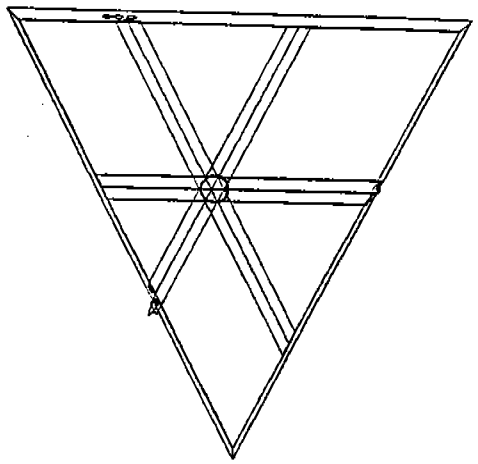
Controls and settings used by the algorithms are accessable via *pseudo_EcFit* or *EcFit_set_controls*.

Each has advantages and disadvantages; the conversion from ADC and TDC values to energy and time of the strips is done by *EcCal_sector*.

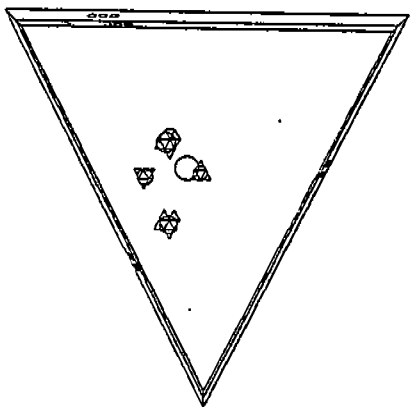
[11] GEANT Simulation of CLAS Forward Calorimeter Performance, CLAS-NOTE 93-009, August 17, 1993



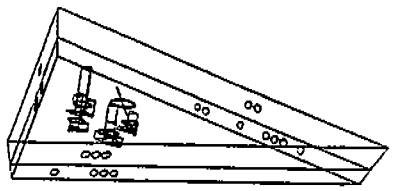
Inner Tubes and Strips



Outer Tubes and Strips

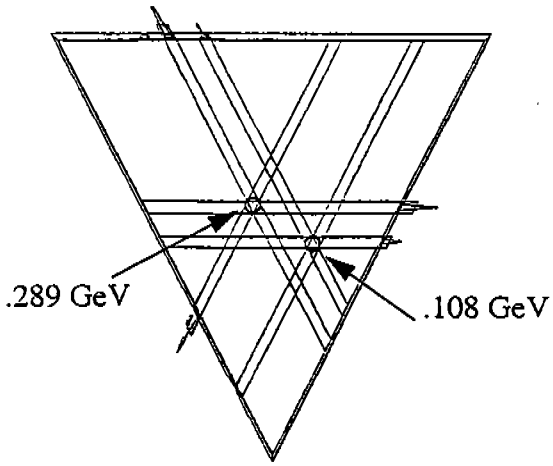


Pixels and Hits

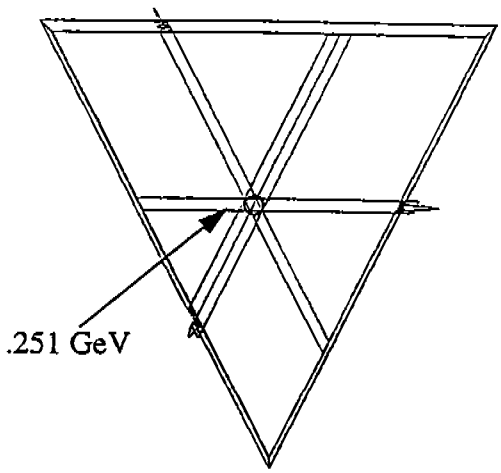


Rotated view of Ec

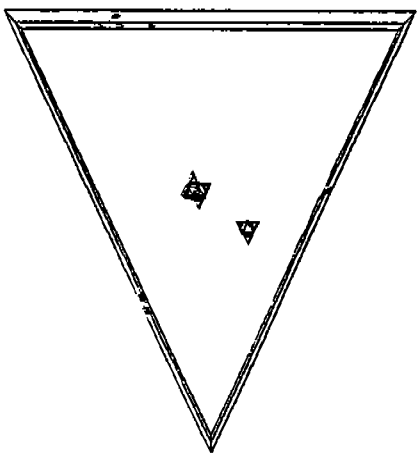
Figure 2. Complicated 1 GeV/c pion event



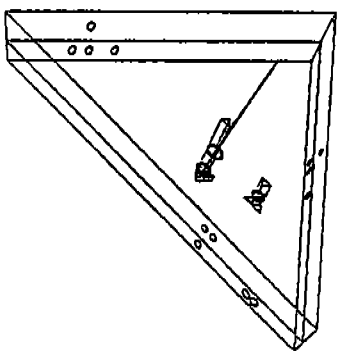
Inner Tubes and Strips



Outer Tubes and Strips



Pixels and Hits



Rotated view of Ec

Figure 3. Another 1 GeV/c pion event.

8.1 Dead Channels

Dead channels within the calorimeter present a challenge to the reconstruction; V. Burkert has estimated that as much as 1% of the channels might not be functioning. If this fraction is represented by η, the chance of a particle depositing significant energy in at least one dead channel is $1-(1-\eta)^N$, where N is the number of channels over which the particle deposits its energy. For $\eta = 1\%$, a μ ($N = 6$) has $\sim 6\%$, a γ or e ($N = 14$) has $\sim 13\%$, and a π or n ($N = 18$) has $\sim 17\%$; hence this problem cannot be treated as a rare and special case.

8.2 The Quick Algorithm

Quick is the least CPU expensive algorithm and is used by *EcFitQuick_scan*. It simply uses the firing strip IDs to form contiguous groups on each edge, and used the geometric overlap to form clusters. Groups which cannot be assigned to clusters become unreconstructable, while the clusters become hits. Since it does not attempt to determine the center of gravity of the groups, just the high and low limits, this technique does not determine position well, but only requires ~0.5 million instructions (MI). Since it doesn't not allow for attenuation corrections, it provides poor energy reconstruction. Its main viture is to quickly determine whether there is anything of interest within the sector. It does not currently deal with dead channels, so its sensitivity to them is of order η, but could easily be modified to reduce the order to η^2.

8.3 The Fast Algorithm

The Fast algorithm used by *EcFitFast* assigns pixels energies based on the strips in *EcFitFast_fill_pixels*; it then uses *EcFit_pixels_hits* to form clusters of pixels and thence hits. *EcFit_hits_showers* is used to sort the hits into showers. It requires about ~2.5 MI.

Each pixel's energy is assigned using the formula

$$E_{pix} = \beta\left(\frac{E_u E_v E_w}{(E_u+E_v+E_w)^3}\right)^{\alpha}(E_u + E_v + E_w)$$

where E_u, E_v, and E_w are the maximum of the strip and δ. α and δ are adjustable constants and β is used to normalize the overall distribution to

the edges. Typically, $\alpha \sim 0.35$ and $\delta \sim 1$ MeV and are determined using the Metropolis algorithm. The sensitivity to dead channels is of order η, but could easily be modified to reduce the sensitivity to order η^2 if the coefficients were determined pixel by pixel.

The quality is assigned by *EcFitPixel_est_reliability*, but that routine is not yet implimented.

8.4 The Edge Algorithm

The **Edge1** algorithm in *EcFitEdge_1st* takes the signals from the strips on a single edge (U, V, or W) and groups them into *peaks* using *EcFitEdge1st_strips_peaks*. Peaks from each edge are combined in *EcFit_peaks_hits* such that the geometric constraints (*EcGus_dalitz_rule*: $u/Lu+v/Lv+w/Lw = 2$) are met to form hits. Hits from the inner and outer layer are combined to form showers by *EcFit_hits_showers*. Pixels are not used in this algorithm, but are subsequently calculated by *EcFit_hits_pixels* for comparison and display. Edge1 requires about ~5 MI and seems the most promising algorithm for general use. Histograms created using this technique are shown in Figure 4 and 5. Figure 4 is for 1 GeV/c electrons, while Figure 5 is for 1 GeV/c pions.

This algorithm makes a "best guess" for the energy depositied in dead channels; its sensitivity to dead channels is of order η^2.

The **Edge2** algorithm, not supported in this release, refits the peaks using information from the hits and a more sophisticated algorithm, and then refits the hits using the recalculated peaks. This will greatly reduce the number of spurious peaks and hits from complicated events and improves the position resolution; its sensitivity to dead channels is of order η^3.

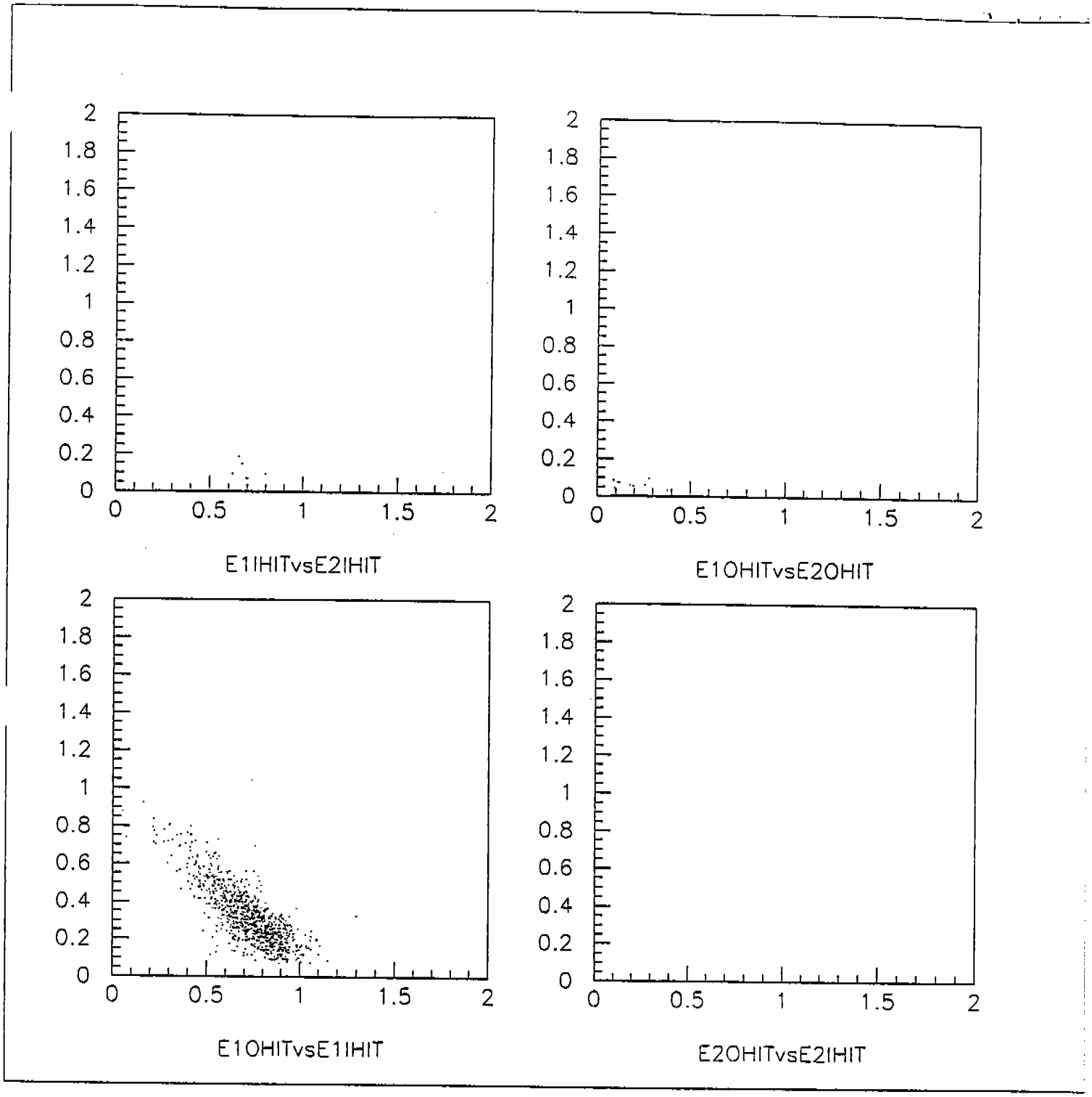


Figure 4. 2nd vs 1st hits: 1 GeV/c e-

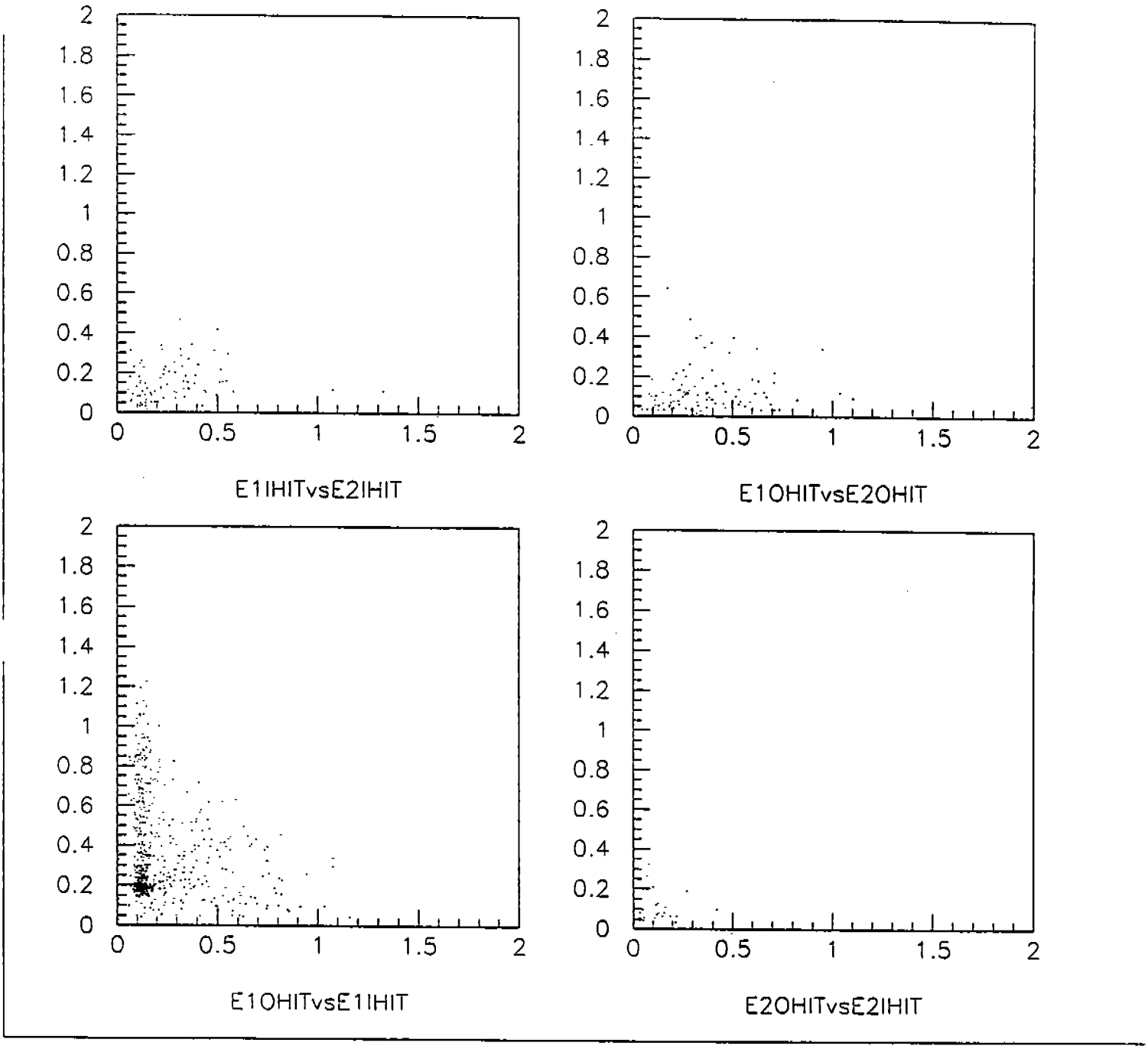


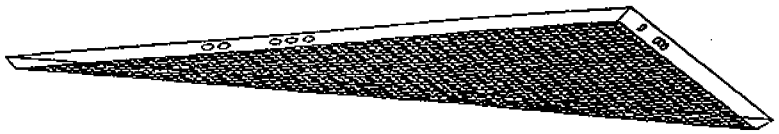
Figure 5. 2nd vs 1st hits: 1 GeV/c pi-

8.5 The Metropolis Algorithm

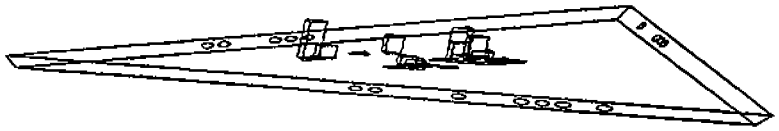
A Metropolis algorithm is a general technique to solve for many variables when the number of constraints is less than the number of variables. Given starting values for the variables, it iteratively relaxes toward a solution.

In this application, the variables are the pixel energies and the contraints the strip energies; it attempts to calculate the appropriate energy for each pixel by iteratively assigning and projecting small energy units. The level determines the size pseudopixels used; the resolution decreases and speed increases with increasing level, with 1 being the finest (pseudopixel) level, 9 representing the coarsest: a single pseudopixel containing all unitpixels. Usually, the level#1 is used, giving 1296 pixels/layer.

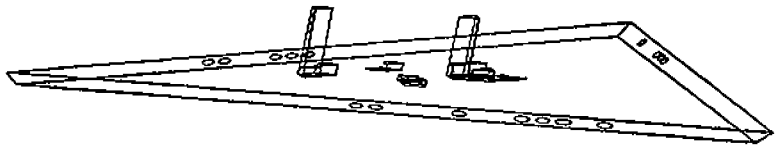
First, in *EcFit_FuzzyMetropolis*, energy is distributed over all pixels according to a starting distribution; currently, either a uniform *EcFit_pixel_uniform*, a masked *EcFit_pixel_masked*, or the results of the Fast algorithm *EcFit-Fast_fill_pixels*. Then a list of pixels sorted on energy is used to determine from using a random number which pixel is to be assigned a small amount of energy. The response of each pixel was determined during the Ec initialization by *EcSrv_response_init*; this energy is either stored or discarded depending on whether sufficient energy remains in each strip to deduct the projected pixel energy; the strips loose the projected energy if the energy is kept. The decision to keep or reject the energy is not a simple step function, but a linear ramp function from 0 to 100% over an interval of uncertainty, with a random number to determine where on the ramp it may fall ("fuzzy" logic). The kept energy forms a new distribution; on the next cycle all pixels are filled with the new distribution and the total energy normalized to that of the original strips, and the entire process repeated. The reconstruction quality is assigned at the end by *EcFitPixel_est_reliability*, but that routine is not yet implimented. This routine is very expensive and robust and can require ~6500 MI; this number can be greatly decreased by judicious selection of step sizes. This algorithm will probably only be useful for rare and complicated events and to determine the coefficients for the Fast algorithm; however it is extremely insensitive to dead channels (order η^3).



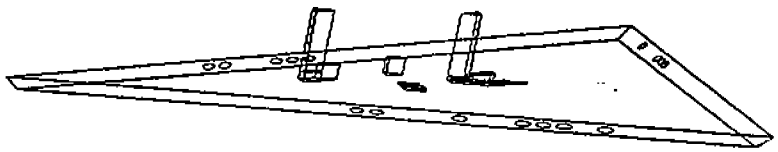
uniform starting distribution



1st iteration



2nd iteration



final iteration

Figure 6. Metropolis Algorithm

9 Linking

The Ec package can be linked as a stand alone program (Ec.exe) or as part of a larger package. The parameter **Ec_MAXpixels** in **Ec_general.PAR** determines the amount of memory available for the pixel routines. If **Ec_MAXpixels** < 1296, no pixels will be available; if **Ec_MAXpixels** $= 1911$, all (pseudo)pixels will be enabled. To compile correctly, **Ec_MAXpixels** > 0. The code will automatically adapt during initialization to the available memory. The code is constructed so various pieces may be easily omitted when linking, although the memory required for the things other than the pixels is small.

External nonsystem libraries essential to Ec are the CLAS utilities, **lib_SwGus.a**, and BOS utilities **libh1util.a**, **libbos.a**, and **libfpack.a**. Other libraries which may be needed for all Ec functions are CERNLIB **libgenlib.a**, **libgraflib.a**, **libgrafX11.a**, **libpacklib.a**, **libkernlib.a**, CODA format library **libevgen.a**, and the cLAS eVENT dISPLAY [12] **libced.a**.

The Ec library, **lib_EcLib.a**, contains all Ec functions and subroutines.

10 Future Directions and Comments

The author has attempted to faithfully follow the CLAS standards to produce code suitable for whatever analysis shell eventually will be used for CLAS. Ec's structure reflects somewhat the changing directions of the software working group; its modularity has made it easy to adapt. To date, however, this package has been insufficiently tested; the reconstruction controls should be better determined. To accomplish this, large GEANT simulations are needed. Only after one has great confidence in the code should error and status checking be reduced; it represents a significant fraction of the code and the time per event.

Electromagnetic events are reconstructed about as well as can be done given the physical limitations of the detector. Hadronic events still present a challenge; a significant fraction ($\sim$0.5%) prove complicated. Ec's philosophy is to do its best job and report the results in a structured way for use in another package. The complexity can be reduced by raising thresholds, for example, at some loss in information; this illustrates the importance of testing and optimization of the controls. Particle identification has not been

[12] Using the *ced* library, CEBAF CLAS-NOTE-94-004 (1994)

included in this package; in conversations [13] it was decided to leave that function to some (then undefined) inter-detector track linking and particle identification package. From tests on the data from the prototype, π/e rejection would be about ~90% using just the total energy, ~97% using the inner-outer energy, and ~99% also using the hit width provided the particle's momentum is known from other sources. Even better rejection can be obtained by examining the "quality" factor of the reconstructed objects.

The problem of dead channels and complicated events has been addressed in the fundamental structure of Ec, rather than being left as a collection of special cases, each requiring laborious manipulation after the intial algorithms fail. All of the Ec algorithms can be improved as testing warrents; the fundamental structure and organization is adequate to the task. Specifically, the time of events is reconstructed, but the time is not used in the reconstruction.

The Ec package represents three years of effort; the author would like to acknowledge Dr. L. Dennis of Florida State U., Dr. H. Funsten of the College of William and Mary, and Dr. V. Burkert of CEBAF for their advice and encouragement.

[13] L. Dennis, private communication

A Ec BOS banks

BOS banks are the preferred way to transfer information between reconstruction packages. A brief description of each follows; all banks' numbers are the relevant CLAS sectors. All banks other than **EC** use 4 byte words.

The EC BOS bank contains the raw data produced by the ADC and TDC FASTBUS modules as 16 bit integer words. It is used by *EcEvu_store_from_BOS* and *EcEvu_fetch_to_BOS*.

name	format	elements	description
EC	B16	packed_ID	PMT#+256*(3*(layer-INNER)+(edge-U+1))
		TDC	raw TDC
		ADC	raw ADC

The **EcCl** BOS bank contains the information required to calibrate the strips. It is accessed only during initialization, not event-by-event. It is used by *EcCal_store_from_BOS* and *EcCal_fetch_to_BOS*.

EcCl	F,(3I,5F)	ToS{nS}	overall sector-zero time
		layer	layer-INNER+1 {1-2}
		edge	edge-U+1 {1-3}
		strip	strip#=PMT# {1-36} [shortest=1]
		Eo{GeV}	ADC=0 energy
		Ech{GeV}	ADC GeV/channel
		To{nS}	TDC=0 time
		Tch{nS}	TDC nS/channel
		attenuation{cm}	strip attenuation length

The **EcPi** and **EcPo** BOS banks contain information on the reconstructed unitpixels in the EcFit common. They are accessed by *EcFit_fetch_to_BOS* and *EcFit_reset_BOS*.

EcPi,EcPo	4I,5F	id	pixel ID number
		Uid	intersecting U strip ID number
		Vid	intersecting V strip ID number
		Wid	intersecting W strip ID number
		E{GeV}	energy deposited within pixel
		T{nS}	time of deposition

	Y{cm}	
	Z{cm}	

The **EcHi** and **EcHo** BOS banks contain information on the reconstructed hits in the EcDrv common. They are accessed by *EcDrv_fetch_to_BOS* and *EcDrv_reset_BOS*.

EcHi,EcHo I,7F	id	hit ID number
	E{GeV}	total hit energy
	T{nS}	mean time of hit
	WIDTH{cm}	RMS width of hit
	QUALITY	reconstruction quality factor
	X{cm}	CLAS position of center of hit
	Y{cm}	
	Z{cm}	

The **EcS** BOS bank contains information on the reconstructed showers in the EcDrv common. It is accessed by *EcDrv_fetch_to_BOS* and *EcDrv_reset_BOS*.

EcS I,7F	id	shower ID number
	E{GeV}	total shower energy
	T{nS}	mean time of shower
	WIDTH{cm}	RMS width of inner hit of shower
	QUALITY	reconstruction quality factor
	X{cm}	CLAS position of center of inner hit
	Y{cm}	
	Z{cm}	

The **Ec?i** and **Ec?o** BOS banks contain information on the unreconstructable energy in the EcDrv common. Upper and lower limits are given for each instance. They are accessed by *EcDrv_fetch_to_BOS* and *EcDrv_reset_BOS*.

Ec?i,Ec?o I,10F	id	unreconstructable energy ID number
	lo-E{GeV}	low energy limit
	lo-T{nS}	low time limit
	lo-X{cm}	low CLAS position
	lo-Y{cm}	
	lo-Z{cm}	
	hi-E{GeV}	high energy limit
	hi-T{nS}	high time limit
	hi-X{cm}	high CLAS position
	hi-Y{cm}	
	hi-Z{cm}	

B Ec COMMON blocks

The forward calorimeter (Ec) reconstruction software encompasses several stages of analysis and processes each sector separately. All information other than controls and status are passed via common blocks. In turn these are defined using carefully selected parameters; hence these parameters define the information content of the Ec package. Each step of analysis is characterized by separate common blocks:

DESCRIPTION	COMMON name
raw event block:	EcEvb
unpacked raw event:	EcEvu
calibrated strips:	EcCal
edge fitting algorithm:	EcFit_edge
metropolis fitting algorithm:	EcFit_pixels
general fitted results:	EcFit_general
best reconstruction results:	EcDrv_general
general status information:	EcGus_status

Only *EcRsl* files should ever be included into routines outside the Ec package. Note that the *EcRsl* parameters are different from those used by the other Ec commons.

exported results:	EcRsl_export

C Ec parameters

The Ec package uses parameters to define the relevant characteristics of objects; not all objects use all parameters. While these parameters are only for internal Ec use, their descriptions are included here and in Ec_par_desc.CMN.

parameter	label	description
Ec_undefined	UNDEFINED	reserved undefined Ec parameter
Ec_ADC	ADC	raw ADC value
Ec_TDC	TDC	raw TDC value
Ec_nearest	NEAREST	nearest geometrically to PMT
Ec_midpoint	MIDPOINT	geometric midpoint from PMT
Ec_furthest	FURTHEST	furthest geometrically from PMT
Ec_lowest	LOWEST	lowest numeric value
Ec_mean	MEAN	numeric mean
Ec_highest	HIGHEST	highest numeric value
Ec_inner	INNER	Ec inner layer
Ec_outer	OUTER	Ec outer layer
Ec_total	TOTAL	Ec cover/total of inner and outer layer
Ec_r	R	CLAS r spherical coord. (cm)
Ec_theta	THETA	theta (radians)
Ec_phi	PHI	phi (radians)
Ec_x	X	CLAS X rectangular coord. (cm)
Ec_y	Y	Y
Ec_z	Z	Z
Ec_s1	S1	local sector S1 rectangular coord. (cm)
Ec_s2	S2	S2
Ec_s3	S3	S3

Ec_i	I	local Ec I rectangular coord. (cm)
Ec_j	J	J
Ec_k	K	K
Ec_u	U	local Ec U edge coord. (cm)
Ec_v	V	V
Ec_w	W	W
Ec_energy	ENERGY	energy (GeV)
Ec_time	TIME	time (nS)
Ec_width	WIDTH	width (cm)
Ec_quality	QUALITY	quality
Ec_unrecon	UNRECONSTRUCTED	unreconstructed energy (GeV)
Ec_dark	DARK	unseen energy (GeV)
Ec_distance	DISTANCE	distance (cm)
Ec_attenuation	ATTENUATION	fractional attenuation
Ec_feyness	FEYNESS	fractional number of dark channels
Ec_dimness	DIMNESS	fractional efficiency of channels
Ec_last_par	LAST_PARM	end of Ec parameter list marker

D Subroutines and Functions

The progamming conventions used by Ec are consistent; *sector* is the CLAS sector and has a value of 1 to 6; *layer* is the **INNER** (*Ec_inner*) or **OUTER** (*Ec_outer*) layer of the calorimeter. In some cases, *layer* can also be **COVER** (*Ec_cover*), the outermost surface of the outer layer. Note that valid *layer* values are **not** 0, 1, or 2. The *axis* or *edge* of the calorimeter are either **U** (*Ec_U*), **V** (*Ec_V*), or **W** (*Ec_W*). Note that axes are **not** counted 1, 2, 3.

If a routine can fail, it usually has a success or failure flag *OK*, and an explanation for failure *err*. If a called routine within the routine fails, the message *err* is prepended with the name of the calling routine and passed to its calling routine.

Routines generally expected to be called by casual users are denoted by **boldface** type and are listed below.

`Ec_initialize_all`
`Ec_recon_all`
`Ec_recon_BOS`
`EcGus_status_report`
`EcGra_option`
`EcGra_plot`
`pseudo_EcAcc`
`pseudo_Ec`

D.1 Initialization

These routines are used to calculate and fill the internal commons used in the reconstructions with geometric and calibration constants.

Subroutine Ec_force_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

This routine resets all Ec arrays in an Ec *sector*, not just those which have been used. It is **only** used when initializing Ec; *Ec_reset* should be used elsewhen.

Subroutine Ec_force_reset_all (OK, err)
Logical OK
Character(*)* err

This routine resets all Ec arrays in all sectors. It should **not** be used in place of *Ec_reset_all.*

Subroutine **Ec_initialize_all** (OK, err)
Logical OK
Character(*)* err

This routine initializes all Ec arrays in all sectors. It gets its geometry from *EcDba_init.*

Subroutine Ec_simple_initialize (setup_file, OK, err)
Character(*)* setup_file, err
Logical OK

This routine initializes all Ec arrays in all sectors with the exception of pixel arrays.

Subroutine EcCal_init (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Puts the energy and time calibrations into the correct arrays; takes the calibrations (if possible) from BOS bank EcCl.

Subroutine EcDba_init (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

This routine is expected to access a database of some kind to retrieve the fundamental quantities needed to establish the Ec geometry and calibrations in a *sector*. Currently, it calculates the position of the forward calorimeters based on the include file **EcDba_basic.PAR.**

Subroutine EcGus_initialize_version (OK, err)
Logical OK
Character(*)* err

Updates the version information in **EcGus_status.CMN.**

Subroutine EcSrv_pseudopixel_init (IDrng, bin, layer, sector, OK, err)
Integer IDrng, bin, layer, sector
Logical OK
Character(*)* err

Used to setup the pseudopixel structure, this routine, given a em layer and a valid CLAS *sector*, will create all pseudopixels beginning with ID#*IDrng* created by *bin* strips binned together.

Subroutine EcSrv_response_init (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Fills in the pixel response table for a given *layer* and *sector*.

Subroutine EcSrv_strip_init (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Calculates all constants known to Ec about the strips within a *sector*; calls *EcSrv_tube_init*.

Subroutine EcSrv_tube_init (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Calculates all constants known to Ec about photomultipliers within a *sector*.

Subroutine EcSrv_pixel_init (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Initializes all pixel arrays to the limit of memory specified in the `include` file **Ec_general.PAR**.

Subroutine EcSrv_init (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Initializes geometry of one *sector*; requires that *EcDba_init* have been already called and calls *EcSrv_strip_init* and *EcSrv_pixel_init*.

D.2 Event Reconstruction

The event reconstruction has several algorithms available; each is discussed elsewhere in detail. The raw, digitialized information begins in *EcEvu*. The working space is in *EcFit*; derived results for a fit are stored in *EcDrv*. The *EcRsl* results are a copy of the *EcDrv*, but allow a different parameter space to be used for convenience of other packages. Only the *EcRsl* common should be directly accessed by other packages. The routines *Ec_recon_BOS* and *Ec_recon_all* combine all necessary steps for reconstruction.

Subroutine pseudo_EcFit (firstline, Ok)
Character(*)* firstline
Logical Ok

This interface allows one to specify reconstruction technique and controls.

```
EcFit>                  analyze event
      /HElp                     list options
      /RESet                    reset Ec package
      /COmplete[:level]         complete pix tree
      /ANalyze:n                reconstruct event- sec#n
         :All                     all sectors

      /TECHnique:mthd           analysis method
      /DEFAULT:mthd             set default method
      /SECtor:n                 set sector number
      /LAYer:m                  Ec layer
      /AXIS:desc                Ec axis U, V, or W
      /THReshold:v              min. threshold for listing

      /STAGe[:level]            pseudopixel level
      /PIXel[:id]               show pixel info.
      /SHOwer[:id]              show shower info.
      /HITs[:id]                show hit info.
      /STRIPs:id                show strip info.
      /PEAKs[:id]               show peak info.
      /UNITs:rng                show/set energy unit
```

```
/IValue:i              (I) control value
/RValue:r              (R) control value
/SET:control           set control
/QUEry:control         show control
```

For example, to specify the edge algorithm and see the current control settings:

```
ECFIT> /tech:EDGE1     !set edge technique
ECFIT> /query          !show all controls
ECFIT> /analyze:ALL    !analyze one (unpacked) event in all sectors
```

Subroutine **Ec_recon_all** (in_idx, in_bffr, in_ptrs, OK, err)
Integer in_idx, in_bffr(*), in_ptrs(*)
Logical OK
Character(*)* err

This routine performs all reconstruction, starting with the CLAS-CODA 2.0 array *in_ptrs* with data starting at location *in_idx* and pointer table *in_ptrs*, and produces the BOS banks **EcS** (showers), **EcHi** (inner hits), **EcHo** (outer hits), **Ec?i**, (inner unreconstructed), **Ec?o** (outer unreconstructed), **EcPi** (inner pixels), **EcPo** (outer pixels), and **ECRC** (all hits). The choice of reconstruction technique is made via *pseudo_EcFit* or *EcFit_set_default*.

Subroutine **Ec_recon_BOS** (OK, err)
Logical OK
Character(*)* err

This routine performs all reconstruction, starting with the BOS bank **EC** and finishing with the BOS banks **EcS** (showers), **EcHi** (inner hits), **EcHo** (outer hits), **Ec?i**, (inner unreconstructed), **Ec?o** (outer unreconstructed), **EcPi** (inner pixels), **EcPo** (outer pixels), and **ECRC** (all hits). The choice of reconstruction technique is made via *pseudo_EcFit* or *EcFit_set_default*.

Subroutine Ec_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Clears the pointers for data in one *sector* in preparation for an event.

Subroutine Ec_reset_all (OK, err)
Logical OK
Character(*)* err

Clears the pointers for data in all sectors in preparation for an event.

Subroutine Ec_simple_recon (OK, err)
Logical OK
Character(*)* err

Uses the edge algorithm to reconstruct one event in all sectors, beginning with BOS bank EC.

Subroutine EcCal_force_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the entire calibrated data array in one *sector*; should not be used event-by-event.

Subroutine EcCal_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the pointer for calibrated data in one *sector*.

Subroutine EcCal_sector (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Converts the unpacked data to energies and times in one *sector*.

Subroutine EcDrv_force_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the entire derived results array in one *sector*; should not be used event-by-event.

Subroutine EcDrv_keep_results (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Moves the fitted quantities into derived results for one *sector*.

Subroutine EcDrv_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the pointer for the results array in one *sector*.

Subroutine EcEvu_force_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the entire unpacked data array in one *sector*; should not be used event-by-event.

Subroutine EcEvu_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the pointer for the unpacked data array in one *sector*.

Subroutine EcFit_analyze (technique, sector, OK, err)
Character(*)* technique, err
Integer sector
Logical OK

After the data has been unpacked, this routine uses the reconstruction *technique* to produce derived results for one *sector*. If *technique* is blank, the default technique is used; current valid *techniques* are "QUICK" (a very simple overlap algorithm), "FAST" (a one step pixel assignment algorithm), "EDGE1" (a Dalitz-condition algorithm), and "METRO1" through "METRO8" (fuzzy Metropolis algorithm with various binning levels.

Subroutine EcFit_force_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the entire fitted quantities array in one *sector*; should not be used event-by-event.

Subroutine EcFit_FuzzyMetropolis (level, layer, sector, OK, err)
Integer level, layer, sector
Logical OK
Character(*)* err

Given unpacked data, uses the Metropolis algorithm to iteratively assign energies to all (pseudo)pixels of a given *level* for a *layer* and *sector*. Assumes a pixel starting distribution is already in place; called by *EcFitMetro*.

Subroutine EcFit_get_default (technique, OK, err)
Character(*)* technique, err
Logical OK

Recovers the current default reconstruction *technique*.

Subroutine EcFit_hits_peaks (layer, sector, OK, note)
Integer layer, sector
Logical OK
Character(*)* err

Used for testing, development, graphics, and EDGE2; projects hits into the corresponding peaks on the edges of a given *layer* and *sector*.

Subroutine EcFit_hits_pixels (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Used for testing, development, graphics, and EDGE2; projects hits into the corresponding unitpixels of a given *layer* and *sector*.

Subroutine EcFit_hits_showers (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Attempts to associate hits in the inner and outer layers to form showers for a given *sector*.

Subroutine EcFit_peaks_hits (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Used for the edge algorithms, attempts to use the Dalitz condition ($u/Lu+v/Lv+w/Lw=2$) to associate peaks to form hits within a given *layer* and *sector*.

Subroutine EcFit_pixel_masked (level, layer, sector, OK, err)
Integer level, layer, sector
Logical OK
Character(*)* err

Used for the pixel algorithms, distributes energy uniformly only among pixels of a given *level* (1-9) which are "masked" by the strips which fired. This provides a starting point for a Metropolis algorithm iterative search in *EcFit_FuzzyMetropolis*.

Subroutine EcFit_pixel_uniform (level, layer, sector, OK, err)
Integer level, layer, sector
Logical OK
Character(*)* err

Used for the pixel algorithms, distributes energy uniformly among all pixels of a given *level*. This can provide a starting point for a Metropolis algorithm iterative search.

Subroutine EcFit_pixels_hits (level, layer, sector, OK, err)
Integer level, layer, sector
Logical OK
Character(*)* err

Used for the pixel algorithms; "lumps" pixels at a given *level* together to form hits for a given *layer* within a *sector*.

Subroutine EcFit_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Clears the pointers of the fitted quanties for one *sector*.

Subroutine EcFit_set_default (technique, OK, err)
Character(*)* technique, err
Logical OK

Sets the default reconstruction *technique*; checks to verify that it is valid.

Subroutine EcSrv_pixel_reset (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Clears the reconstructed information associated with pixels of a given *layer* and *sector*.

Subroutine EcSrv_complete_pseudopixels (level, layer, sector, OK, err)
Integer level, layer, sector
Logical OK
Character(*)* err

Given reconstructed pixels of a *level*, fills reconstructed information for all other levels of pixels within a *layer* and *sector*.

Subroutine EcFit_unitpixels_only (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Removes all pixels (except unitpixels, level=1) from the fitted information for one *layer* and *sector*.

Subroutine EcFitEdge1st_strips_peaks (axis, layer, sector, OK, err)
Integer axis, layer, sector
Logical OK
Character(*)* err

Used for the edge algorithms, associates strips to form peaks on one *axis* (**Ec_U**, **Ec_V**, or **Ec_W**) of one *layer* (**Ec_INNER** or **Ec_OUTER**) of one *sector*. Only the moments of the strip energy distribution are used to estimate the peaks.

Subroutine EcFitEdge_1st (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

The first level edge algorithm; strips are used to form U-V-W peaks, peaks are used to form hits, inner and outer hits are used to form showers in

EcFit_peaks_hits. The peaks are corrected for attenuation and time, but not iteratively fit.

Subroutine EcFitEdge_est_reliability (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Fills in the quality of reconstructed objects for the edge algorithm. Here quality is defined as the RMS error over the total energy.

Subroutine EcFitEdge_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the objects used in the edge algorithm.

Subroutine EcFitFast (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

A non-iterative pixel algorithm; each pixel is assigned energy based on a formula in *EcFitFast_fill_pixels*. The pixels are grouped into hits by *EcFit_pixels_hits*, hits are grouped into showers by *EcFit_hits_showers*.

Subroutine EcFitFast_fill_pixels (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Fills uintpixels using a formula and the pixel response table.

Subroutine EcFitMetro (level, sector, OK, err)
Integer level, sector
Logical OK
Character(*)* err

Given unpacked data, initially fills pixels using *EcFit_pixel_masked* and then uses *EcFit_FuzzyMetropolis* to establish pixel quantities; it then associates pixels into hits (using *EcFit_pixels_hits*) and hits into showers (using

EcFit_hits_showers) for a given *level* (1= finest, 9=coarsest), *layer*, and *sector*.

Subroutine EcFitPixel_est_reliability (layer, sector, OK, err)
Integer layer, sector
Logical OK
Character(*)* err

Fills in the quality of reconstructed objects for the pixel algorithms.

Subroutine EcFitPixel_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Resets the list of reconstructed pixels for one *sector*.

Subroutine EcFitQuick_scan (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Attempts to very quickly and crudely establish the number of hits within the layers of one *sector*. This algorithm uses just the strip locations to seek overlapping regions. No attenuation corrections are made.

Subroutine EcSrv_pixel_select (level, thr, N, IDs, layer, sector, OK, err)
Integer level, N, IDs(*), layer, sector
Real thr
Logical OK
Character(*)* err

Returns the number *N* and *IDs* of pixels within pseudopixel *level*, *layer* and *sector* with energy exceeding threshold *thr* are returned.

D.3 Reconstruction Controls and Calibrations

All important reconstruction controls may be set using *EcFit_set_controls* or *pseudo_EcFit*. The units are always the standard CLAS units; GeV and cm, or a dimensionless fraction or integer. Typical values are shown in this *pseudo_EcFit* input file:

```
!QUICK algorithm
    /rv:.001   /set:EcFitQuick_threshold          !GeV

!FAST algorithm
    /rv:.001   /set:EcSimFast_strip_threshold     !GeV
    /rv:.001   /set:EcSimFast_pixel_threshold     !GeV

!EDGE1 algorithm
    /rv:.0010  /set:EcFitEdge_strip_threshold     !GeV
    /rv:.003   /set:EcFitEdge_peak_threshold      !GeV
    /rv:.010   /set:EcFitEdge_hit_threshold       !GeV

!METROPOLIS algorithm
    /rv:.96    /set:EcFitMetro_FRACreqd           !stop looping after 0.-1.
    /rv:.001    /set:EcFitMetro_stepsize          !GeV
    /rv:.003    /set:EcFitMetro_pixel_threshold   !GeV
    /rv:.001    /set:EcFitMetro_strip_threshold   !GeV
    /iv:10      /set:EcFitMetro_MAXiterations     !maximum
    /iv:99999   /set:EcFitMetro_MAXloops          !max loops/iteration
```

The calibrations depend on the data input; they can be set via the EcCl BOS bank or via the *pseudo_EcCal* routine, as shown in this *pseudo_EcCal* input file:

```
!length units always cm
    /atten:376.   !cm
!change energy units from GeV [default] to MeV for legibility
    /unit:MeV
!define energy/channel and ADC=0 energy using current [MeV] units
    /Ech:0.321 !MeV/ch
    /E0:0.0    !MeV for ADC=0
```

```
!time always in nS; define time/channel and TDC=0 time
    /Tch:0.05    !nS/ch
    /T0:0.0      !nS for TDC=0
!select all sectors, all layers, all axes, set all channels [1-36]
/-sec /-lay /-axis /set:1::36
```

Dead channels are denoted with a negative attenuation length.

D.4 The Export COMMON

The EcRsl common was created to satisfy the desires of users who wish direct access to a common containing the reconstructed results without coupling directly to internal Ec structures. The EcRsl commons use these parameters to identify the relevant characteristics of objects; note that they are **not** the same as the Ec parameters. These parameters are required when using the *EcRsl* commons directly. Only the include files **EcRsl_export.PAR**, **EcRsl_export.CMN**, **EcRsl_status.CMN**, and **EcRsl_par_desc.CMN** should be used when accessing the EcRsl commons directly.

parameter	description
EcRsl_undefined	reserved undefined EcRsl parameter
EcRsl_nearest	nearest geometrically to PMT
EcRsl_middle	geometric middle
EcRsl_furthest	furthest geometrically from PMT
EcRsl_lowest	lowest numeric value
EcRsl_mean	numeric mean
EcRsl_highest	highest numeric value
EcRsl_inner	Ec inner layer
EcRsl_outer	Ec outer layer
EcRsl_cover	Ec cover/total of inner and outer layer
EcRsl_r	CLAS r spherical coord. (cm)
EcRsl_theta	theta (radians)
EcRsl_phi	phi (radians)
EcRsl_x	CLAS X rectangular coord. (cm)
EcRsl_y	Y
EcRsl_z	Z
EcRsl_energy	energy (GeV)
EcRsl_time	time (nS)
EcRsl_width	width (cm)

```
EcRsl_quality           quality
EcRsl_dark              unseen energy (GeV)

EcRsl_last_par          unused last EcRsl parameter
```

Subroutine pseudo_EcRsl (firstline, Ok)
Character(*)* firstline
Logical Ok

This interface gives access to the exported results.

```
EcRsl>                  export ReSuLts
      /HElp                     list options
      /SECtor[:n]               CLAS sector
      /FORce                    force reset
      /RESet                    reset
      /LAYer[:m]                Ec layer
      /SHOwer[:id]              show shower info.
      /HITs[:id]                show hit info.
```

Subroutine EcRsl_file_results (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Moves derived information into the export results common for a given *sector*.

Subroutine EcRsl_force_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Clears the entire results structure for one *sector*; not intended to be used event-by-event.

Subroutine EcRsl_reset (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Clears the pointers of results list for one *sector*.

D.5 Error and Status Checking

Nearly every Ec routine checks all pertinent status information each time it is called; this overburden is significant, but considered worthwhile during code development. Eventually, as the code is tested and certified, much of this error checking can be removed.

Subroutine EcGus_check_sector (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

This routine simply checks that the CLAS *sector* exists and has been initialized.

Subroutine EcGus_check_status (subject, status, sector, OK, err)
Character(*)* subject, err
Logical status, OK
Integer sector

Given a *subject* within a *sector*, this routine returns its *status*. The *status* indicates whether the contents of *subject* are valid. Any Ec prefix is ignored in *subject*; and *subject* is case insensitive. Supported *subjects* include (bold type denotes minimum abreviation) "**INIT**ialized" (complete Ec initialization of all sectors), "**SRV**ey" or "**SuRV**eyed" (successful geometric construction of all sectors), "**DBA**ccess" or "**DataBaseA**ccessed" (successful access to geometric constants and calibrations for all sectors), "**EVU**npacked" or "**EVentU**npacked" (event unpacking done for one *sector*), "**CALi**brated" (conversion of channels to energy and time done for one *sector*), "**FIT**ted" (at least some reconstruction done for one *sector*), and "**DRV**ived" or "**DeRi**Ved" (reconstructed information stored for one *sector*).

Subroutine EcGus_status_report (data, done, sector, OK, err)
Logical data, done, OK
Integer sector
Character(*)* err

For a given *sector*, report whether there is data present (*data*) and whether unpacking has been attempted (*done*).

Subroutine EcGus_version_report (description, length)
Character(*)* description
Integer length

Returns the current Ec version *description*, with nonblank length *length*.

Subroutine EcGus_print_status_report (IO)
Integer IO

Print all current status information to (already opened) FORTRAN channel#*IO*.

D.6 Mathematics

These are common mathematical functions used by Ec.

Real Function EcGus_gamma_function (x)
Real x

Evaluates the gamma function of x.

$$\Gamma(x) = \int_0^\infty e^{-z} z^{x-1} dz$$

Subroutine EcGus_Gaussian_smear (centroid, FWHM, where)
Real centroid, FWHM, where

Produces a random point *where* distributed according to a Gaussian centered on *centroid* with full-width-half-maximum *FWHM*.

D.7 Geometry

These routines all deal with the geometry and conversions between the CLAS XYZ and spherical, sector S123, and Ec local IJK and UVW coordinate systems.

Logical Function EcGus_dalitz_rule (u, v, w, MAXerr, i, j, k, RMS, layer, sector)
Real u, v, w, MAXerr, i, j, k, RMS
Integer layer, sector

Give possible projections *u, v, w* along the edges (presumably made parallel to the strips); determines if it satisfies the rule $u/Lu+v/Lv+w/Lw = 2$ (within a range *MAXerr*). If it does, the position of the point projected *i, j, k* and the root mean square "spread" *RMS* of the point is returned.

Real Function EcGus_attenuation (dst, id, axis, layer, sector, OK, err)
Real dst
Integer id, axis, layer, sector
Logical OK
Character(*)* err

Given a distance *dst* from the photomultiplier end of a strip#*id*, on an *axis* in a *layer* within a *sector*, this routine returns the effective attenuation factor.

Subroutine EcGus_ij_path (i, j, pU, pV, pW, layer, sector)
Real i, j, pU, pV, pW
Integer layer, sector

Given a point at (*i,j*) within a *layer* and *sector*, finds the path lengths (*pU,pV,pW*) along to the strips to the edge.

Subroutine EcGus_ij_uvw (i, j, u, v, w, layer, sector)
Real i, j, u, v, w
Integer layer, sector

Given a point at (*i,j*) within a *layer* and *sector*, returns the (*u,v,w*) edge coordinates formed by projections parallel to the strips.

Integer Function EcGus_ijk_pixel (i, j, k, layer, sector, OK)
Real i, j, k
Integer layer, sector
Logical OK

Given a point (*i,j,k*) within a sector, returns the *layer* ID number of the pixel containing the point, if any.

Subroutine EcGus_ijk_XYZ (i, j, k, x, y, z, sector)
Real i, j, k, x, y, z
Integer sector

Given a point (*i,j,k*) within a *sector*, returns the CLAS coordinate (*x,y,z*).

Subroutine EcGus_normalize (x, y, z)
Real x, y, z

Given a vector (*x,y,z*), normalizes its length to 1, if possible.

Subroutine EcGus_s123_strike (q1, q2, q3, d1, d2, d3, h1, h2, h3, layer, sector, OK)
Real q1, q2, q3, d1, d2, d3, h1, h2, h3
Integer layer, sector
Logical OK
Character(*)* err

Linearly extrapolate from sector system point (*q1,q2,q3*) along vector (*d1,d2,d3*) to calorimeter *layer* and *sector*; return point of contact (*h1,h2,h3*).

Subroutine EcGus_s123_XYZ (q1, q2, q3, x, y, z, sector)
Real q1, q2, q3, x, y, z
Integer sector

Given a sector system point (*q1,q2,q3*) within a *sector*, return the CLAS system location (*x,y,z*).

Subroutine EcGus_sph_XYZ (r, theta, phi, x, y, z)
Real r, theta, phi, x, y, z

Given a point in the CLAS spherical system (*r,theta,phi*), returns the CLAS rectangular coordiantes (*x,y,z*).

Subroutine EcGus_uvw_IJK (M, dm, N, dn, C, dc, i, j, k, layer, sector)
Real dm, dn, dc, i, j, k
Integer M, N, C, layer, sector

Given 2 of 3 edge coordinates (M,dm and N,dn), finds the third (C,dc) and the local coordinates (i,j,k). For example, if M=U, N=V, then C=W.

Subroutine EcGus_vec_IJK_XYZ (i, j, k, x, y, z, sector)
Real i, j, k, x, y, z
Integer sector

Given a local vector (i,j,k) within a *sector*, finds the same vector (x,y,z) in the CLAS system.

Subroutine EcGus_vec_S123_XYZ (d1, d2, d3, x, y, z, sector)
Real d1, d2, d3, x, y, z
Integer sector

Given a sector vector ($d1$,$d2$,$d3$) within a *sector*, finds the same vector (x,y,z) in the CLAS system.

Subroutine EcGus_vec_XYZ_ijk (x, y, z, i, j, k, sector)
Real x, y, z, i, j, k
Integer sector

Given a CLAS vector (x,y,z) within a *sector*, finds the same vector (i,j,k) in the local system.

Subroutine EcGus_vec_XYZ_S123 (x, y, z, d1, d2, d3, sector)
Real dx, dy, dz, d1, d2, d3
Integer sector

Given a CLAS vector (x,y,z) within a *sector*, finds the same vector ($d1$,$d2$,$d3$) in the sector system.

Subroutine EcGus_XYZ_ijk (x, y, z, i, j, k, sector)
Real x, y, z, i, j, k
Integer sector

Given a CLAS point (x,y,z), finds the local coordinate (i,j,k) for a *sector*.

Subroutine EcGus_XYZ_s123 (x, y, z, q1, q2, q3, sector)
Real x, y, z, q1, q2, q3
Integer sector

Given a CLAS point (x,y,z), finds the sector coordinate ($q1$,$q2$,$q3$) and for a given *sector*.

Subroutine EcGus_XYZ_sector (x, y, z, sector)
Real x, y, z
Integer sector

Given a CLAS point (x,y,z), finds which CLAS *sector* it falls into.

Subroutine EcGus_XYZ_sph (x, y, z, r, theta, phi)
Real x, y, z, r, theta, phi

Given a CLAS point at (x,y,z), finds the spherical coordinates (r,$theta$,phi).

Subroutine EcGus_XYZ_strike (x, y, z, dX, dY, dZ, hX, hY, hZ, layer, sector, OK)
Real x, y, z, dX, dY, dZ, hX, hY, hZ
Integer layer, sector
Logical OK

Extrapolates from (x,y,z) along (dX,dY,dZ) to the point where it intersects the specified *layer* (**Ec_INNER-Ec_COVER**) of the calorimeter within a *sector* at (hX, hY, hZ).

Subroutine EcGus_XYZ_uvw (x, y, z, u, v, w, layer, sector)
Real x, y, z, u, v, w
Integer layer, sector

Given a CLAS point (x,y,z), finds the edge coordinate (u,v,w) and the *sector*. Extrapolation is normal to the *layer*.

D.8 Lookup

Subroutine EcGus_find_parm (Ndesc, desc, par_numbers, OK, err)
Integer Ndesc, par_numbers(*)
Logical OK
Character(*)* desc(*), err

This routine looks up the current parameter numbers *par_numbers* for a list *desc* of parameter names of length *Ndesc*.

Subroutine EcGus_particle (description, M, Q, OK, err)
Character(*)* description, err
Real M, Q

Given a *description* of a particle ("e+", "muon", etc.) returns the mass *M* and charge *Q*, if possible.

Subroutine EcGus_specifics (description, layerOK, layer, edgeOK, edge, limitOK, limit)
Character(*)* description
Integer layer, edge, limit
Logical layerOK, edgeOK, limitOK

Given a *description*, attempt to extract a *layer* (**INNER-OUTER-COVER**), *edge* (**U-V-W**), and *limit* (**MIN-MEAN-MAX**). Success for each is indicated by *layerOK*, *edgeOK*, and *limitOK*.

Subroutine EcEng_tuple_test (FLAG, msg, OK, err)
Integer sector
Logical OK
Character(*)* err

When using the *Ec_engine** programs, this routine sets a *FLAG*=.TRUE. when a preset condition has been satisfied. The condition is specified using *pseudo_EcEngine*.

D.9 BOS interface

The standard inter-package communication is now planned to be via the BOS utilities. These routines connect Ec to BOS banks. BOS banks may be created "on-the-fly" using *pseudo_EcAcc*. All BOS banks used by Ec are documented using *SwGus_BOS_comment*. The BOS input banks are as described in CLAS-NOTE 94-012.

The fitted EcFit banks describe the pixels; the inner pixels are described in bank **EcPi** and the outer in bank **EcPo**. Each are of the format "4I, 5F", with the quanties *id, Uid, Vid, Wid, E, T, X, Y, Z* in the standard CLAS system and units. *Uid, Vid, Wid* are the numbers of the strips which correspond to pixel *id*.

The EcDrv banks describe derived quantities which are extrinsic to the reconstruction algorithm. The first, **EcS**, describes showers constructed from inner and outer hits. It has format "I, 7F" and quantities *id, E, T, WIDTH, QUALITY, X, Y, Z* in the standard CLAS system and units. The *ids* are numbered sequentially from the largest. Next, the inner **EcHi** and outer **EcHo** hit banks describe energy clusters reconstructed with format "I, 7F" and quantities *id, E, T, WIDTH, QUALITY, X, Y, Z*. Remaining unreconstructable energy is stored into **Ec?i** and **Ec?o** for the inner and outer layer respectively. They have format "I, 10F", with the quantities *id, lo-E, T, X, Y, Z, hi-E, T, X, Y, Z* describing the lower and upper limits of the energy.

The EcAcc interface can specify vectors and BOS banks if the routine *EcAcc_fetch_to_BOS* is subsequently called.

The BOS files and banks may be accessed via the *pseudo_EcEvu_BOS* interface

Subroutine pseudo_EcEvu_BOS (firstline, Ok)
Character(*)* firstline
Logical Ok

This interface allows one to open, close, read, and write BOS/FPACK files. The **FETCH** option calls all routines which move Ec arrays into BOS banks; it is unnecessary when *Ec_recon_BOS* is used. Documentation is done using the SwGus utilities.

```
EcEvu_BOS>              BOS 0. format I/O
      /HElp                    list options
```

```
      /INITitialize          initialize BOS
file  /INPut                 open BOS input
      /-INPut                close BOS input
file  /OUTput                open BOS output
      /-OUTput               close BOS output
      /GETevent              get next event
      /PUTevent              output an event
      /STOre                 store BOS->EcEvu
      /FETch                 fetch Ec->BOS
      /REset                 delete BOS banks
      /LIst                  list current banks
         :bank                 one bank
      /DOCument:bank         print comments on bank
```

For example, to read and reconstuct one event:

```
EC> BOS
ECEVU_BOS> bosinputfile.evt/IN  !open "bosinputfile.evt"
ECEVU_BOS> /get /store          !get 1 event and use it to fill EcEvu
ECEVU_BOS> %                    !return to EC>
EC> FIT /fit:all                !ECFIT> /fit:all     !reconstruct all sectors
EC> BOS /list                   !ECEVU_BOS> /list    !list all BOS banks
```

Subroutine EcAcc_reset_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Delete BOS banks created by the *EcAcc* routines in one *sector*.

Subroutine EcAcc_reset_BOS_all (OK, err)
Logical OK
Character(*)* err

Delete all BOS banks created by the *EcAcc* routines.

Subroutine EcAcc_fetch_BOS_all (OK, err)
Logical OK
Character(*)* err

Fill all BOS banks defined by the *EcAcc* routines.

Subroutine EcAcc_fetch_to_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Fill all BOS banks defined by the *EcAcc* routines for one *sector*.

Subroutine EcCal_store_BOS_all (anyOK, err)
Logical anyOK
Character(*)* err

Move all calibration constants found in BOS bank EcCl into the internal arrays for all sectors, if possible. If any sector can be done, *anyOK*=.TRUE. It is only called during initialization.

Subroutine EcCal_store_from_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Move all calibration constants found in BOS bank EcCl into the internal arrays for one *sector*.

Subroutine EcCal_fetch_BOS_all (allOK, err)
Logical allOK
Character(*)* err

Move all calibration constants found in the internal arrays for all sectors into BOS banks EcCl, if possible. If any sector can be done, *anyOK*=.TRUE. It is called as required by *pseudo_EcCal*.

Subroutine EcCal_fetch_to_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Move calibration constants found in the internal arrays for one *sector* into BOS bank **EcCl**.

Subroutine EcDrv_fetch_BOS_all (OK, err)
Logical OK
Character(*)* err

Move derived results into the BOS banks **EcHi** (inner hits), **EcHo** (outer hits), **Ec?i**, (inner unreconstructed), and **Ec?o** (outer unreconstructed), and **EcS** (showers) for all sectors.

Subroutine EcDrv_fetch_to_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Move derived results into the BOS banks **EcHi** (inner hits), **EcHo** (outer hits), **Ec?i**, (inner unreconstructed), **Ec?o** (outer unreconstructed), and **EcS** (showers) for one *sector*.

Subroutine EcDrv_reset_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Deletes BOS banks **EcHi**, **EcHo**, **Ec?i**, **Ec?o**, and **EcS** as required for one *sector*.

Subroutine EcDrv_reset_BOS_all (OK, err)
Integer sector
Logical OK
Character(*)* err

Deletes BOS banks **EcHi**, **EcHo**, **Ec?i**, **Ec?o**, and **EcS** as required for all sectors.

Subroutine EcEvu_fetch_BOS_all (OK, err)
Integer sector
Logical OK
Character(*)* err

Copies ADC and TDC data for all sectors into BOS bank **EC**.

Subroutine EcEvu_fetch_to_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Copies ADC and TDC data for one *sector* into BOS bank **EC**.

Subroutine EcEvu_store_BOS_all (OK, err)
Logical OK
Character(*)* err

Extracts ADC and TDC data for all sectors from BOS bank **EC**.

Subroutine EcEvu_store_from_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Extracts ADC and TDC data for one *sector* from BOS bank **EC**.

Subroutine EcFit_fetch_BOS_all (OK, err)
Logical OK
Character(*)* err

Fills BOS banks **EcPi** (inner pixels) and **EcPo** (outer pixels) for all sectors.

Subroutine EcFit_fetch_to_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Fills BOS banks **EcPi** (inner pixels) and **EcPo** (outer pixels) for one *sector*.

Subroutine EcFit_reset_BOS (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Deletes BOS banks **EcPi** and **EcPo** for one *sector*.

Subroutine EcFit_reset_BOS_all (OK, err)
Logical OK
Character(*)* err

Deletes BOS banks **EcPi** and **EcPo** for all sectors.

Subroutine EcRsl_fetch_BOS_all (OK, err)
Logical OK
Character(*)* err

Fills BOS bank **ECRC** (inner and outer hits) for all sectors.

Subroutine EcRsl_reset_BOS_all (OK, err)
Logical OK
Character(*)* err

Fills BOS bank **ECRC** (inner and outer hits) for one *sector*.

D.10 EcAcc Routines

These routines provide a very robust form of accessing Ec information; they are intended for use by external packages, allowing those packages to "learn" what is available from Ec. In this way, external packages need not "know" anything about the internal structures, and can easily cope with internal changes. EcAcc can produce *vectors* describing objects which can be mapped easily into BOS banks or HBOOK Ntuples on-the-fly.

Subroutine pseudo_EcAcc (firstline, Ok)
Character(*)* firstline
Logical Ok

The EcAccess routines are intended to provide Ec information via a robust interface. Vectors may be defined and used to create BOS banks or fill Ntuples (using *pseudo_EcEngine*).

```
EcAcc>                  reconstructed information
     /HElp                      list options
     /SECtor:s                  CLAS sector
     /INFOrmation               list available info on object
     /AVailable                 get number of objects
     /Nth:n                     on Nth object in class
     /ID:id                     on object id in class
     /CHaracteristics           list reqd characteristics*
        :desc                     add "desc" to list
     /SPECification             show current specification
          :spec                   set new "spec"*
     /OBJect                    show current object
          :obj                    set new "obj"*
     /COMmon                    show current common
          :cmn                    set new "common"*
     /BOS:name                  set BOS bank name*
     /MAke_comment              make BOS comment*
        :short                    short comment
        :?  long desc             use whole line
 name/CREate                    create an access vector
     /DELETE                    delete all vectors
```

```
    name/FILL_Nth:nth            fill nth access vector
    name/FILL_Id:ID              fill id access vector
        /LIst                    list all access vectors
        /CLear              clear defaults

  *= cancel /op with /-op
```

For example:

```
EC> access                              !call pseudo_EcAcc
ECACC>  /-char /-spec /-obj /-common !cancel any&all previous settings
ECACC> /common:? /info                  !commons currently available?
   common: ?

ERROR: EcAcc_information:HELP requested
   commons: EcFit-EcDrv
   object:
   specifications:

ECACC> /common:ECDRV /obj:? /info   !objects in common "EcDrv"?
   common: EcDrv
   object: ?

ERROR: EcAcc_information>EcDrv_information:HELP requested
 common: EcDrv
 object: SHOWERs-HITs-UNREConstructed
 specifications:

ECACC> /object:HIT /info  !spec's and char.'s allowed on object "HIT"?
 object: HIT

 common: EcDrv
 object: HIT
 specifications: INNER-OUTER
 characteristics:
           R
           THETA
```

```
PHI
X
Y
Z
S1
S2
S3
I
J
K
U
V
W
ENERGY
TIME
WIDTH
QUALITY
UNRECONSTRUCTED
DARK
DISTANCE
ATTENUATION
FEYNESS
DIMNESS
LAST_PARM

ECACC>  %<return>          !exit pseudo_EcAcc
EC>
```

For example, to create BOS banks describing the inner and outer hits which will be automatically filled later by *Ec_recon_BOS*:

```
ECACC> /-char
ECACC> /specific:INNER /object:HIT /common:ECDRV /bos:ECHI
ECACC> /char:X /char:Y /char:Z /char:ENERGY /char:TIME
ECACC> INHIT/create                    !create vector INHIT & BOS bank ECHI
ECACC> /specific:OUTER /bos:ECHO
ECACC> OUTHIT/create                   !create vector OUTHIT & BOS bank ECHO
```

```
ECACC> /list
   INHIT
              INNER HIT ECDRV
            4  X
            5  Y
            6  Z
           27  ENERGY
           28  TIME
   OUTHIT
              OUTER HIT ECDRV
            4  X
            5  Y
            6  Z
           27  ENERGY
           28  TIME
```

Subroutine EcAcc_add_tuple_tag (first, last, Nmax, tag, sector, OK, err)
Integer first, last, Nmax, sector
Character(*)* tag(*), err
Logical OK

Fills an Ntuple's *tags* starting with index *first* for *Nmax* occurances of all EcAcc vectors for a given *sector*.

Subroutine EcAcc_build_vector (name, specific, object, common, Nchar, char, OK, err)
Character(*)* name, specific, object, common, char(*), err
Integer Nchar
Logical OK

Defines an EcAcc vector *name* for a *specific object* within a *common*, described by *Nchar* characteristics *char*. For example, to create a vector describing only the CLAS position, energy, and time of a hit in the outer layer, *name*="EcHo", *common*="EcDrv", *object*="HIT", *specific*="OUTER", *Nchar*=5, *char(1)*="X", *char(2)*="Y", *char(3)*="Z", *char(4)*="ENERGY", and *char(5)*="TIME".

Subroutine EcAcc_duplicate_tuple_tags (Ntag, name, tag, OK, err)
Integer Ntag
Character(*)* name, tag(*), err
Logical OK

The CERNLIB Ntuple associates a case insensitive 8 character label *tag* with each entry in the Ntuple during the *HBOOKN* call. The CERNLIB PAW program has a problem handling Ntuples with duplicated tags; even if the entry if referenced by ID number, the first entry with the duplicated tag is always used. To avoid this problem, this routine is called by *pseudo_EcEngine* before *HBOOKN* and attempts to make every *tag* unique. *EcAcc_duplicate_tuple_tags* scans the list of *Ntag names* (assumed to be in the form "*characteristic_#vector_#*" with the first "*#*" representing the vector ID and the second the sector) and attempts to make every corresponding *tag* unique, if possible. First it begins by removing or compressing a trailing sector number, next substitutes short characteristic names in place of the Ec keywords, and finally drops trailing vector letters. For example, *name(1)*="**ENERGY_1SHWR_3**" might make *tag(1)*="**E1SHW3**".

Subroutine EcAcc_fill_ID (id, name, nth, info, sector, OK, err)
Integer id, nth, sector
Character(*)* name, err
Real info(*)
Logical OK

Fills in the array *info* and sequence position *nth* for the requested vector *name* and *id* for a given *sector*.

Subroutine EcAcc_fill_Nth (nth, name, id, info, sector, OK, err)
Integer nth, id, sector
Character(*)* name, err
Real info(*)
Logical OK

Fills in the array *info* and *id* number for the requested *nth* vector *name* inside a given *sector*.

Subroutine EcAcc_fill_tuple (first, last, Nmax, tuple, sector, OK, err)
Integer first, last, Nmax, sector
Real tuple(*)
Logical OK
Character(*)* err

Fills the array *tuple* beginning at index *first* with all the EcAcc vectors for one *sector*, with up to *Nmax* occurances of each vector; the listing ends at index *last*.

Subroutine EcAcc_find_ID (name, ID, OK, err)
Character(*)* name, err
Integer ID
Logical OK

Checks to see if vector *name* with *ID* exists within the *sector*.

Subroutine EcAcc_find_Nvectors (name, Nfnd, sector, OK, err)
Character(*)* name, err
Integer Nfnd, sector
Logical OK

Returns number of occurances of vector *name* within the *sector*.

Subroutine EcAcc_information (specification, object, common, Nchr, chr, OK, err)
Character(*)* specification, object, common, chr(*), err
Integer Nchr
Logical OK

This subroutine returns information about *objects*, *specifications*, and *commons* within Ec. Setting *common*="?" will produce a list of valid commons; *object*="?" will produce a list of valid objects within a common. If the class of *object* of *common* exists, the *specification* returned contains all allowed values for selecting a particular set objects within a sector, and contains the range of each dimension allowed for that object. A "." separates required fields; "-" the possible values of each field. For example, for *common*="EcDrv_general", *object*="HIT", the returned *specification* is "INNER-OUTER", for *object*="UNREC" the returned *specification* is "LOW-MEAN-HIGH.INNER-OUTER". Hence, "OUTER" "HIT" is a hit in the outer layer,

and "HIGH INNER" "UNREC" gives an upper limit on characteristics of an unreconstructed object in the inner layer. All information about an object consists of characteristics; the number of supported characteristics (*Nchr*) and a character array (*chr*) filled with a brief ASCII label for each characteristic are returned. For *common*="EcDrv_general", *object*="HIT", there are 23 characteristics, beginning with "R", "THETA", "PHI", ... and running to "END of parameters" and *specification*="INNER-OUTER".

Subroutine EcAcc_Nfound (specific, object, common, Nfnd, sector, OK, err)
Character(*)* specific, object, common, err
Integer Nfnd, sector
Logical OK

Returns the number *Nfnd* of *specific objects* within a *common* for a particular *sector*. If the *specific object* of *common* and *sector* are valid (ex: "INNER", "HIT", "EcDrv_general", 3), the status is checked and the number of objects present is put into *Nfnd*.

Subroutine EcAcc_relate_id (id, spec1, obj1, cmn1, Nids, ids, spec2, obj2, cmn2, sector, OK, err)
Integer id, Nids, ids(*), sector
Character(*)* spec1, obj1, cmn1, spec2(*), obj2(*), cmn2(*), err
Logical OK

For a given *id* of a specific object *spec1 obj1* of common *cmn1*, returns the associated specific objects *spec2 obj2* that are in common *cmn2* for a given *sector*. Not implimented in this release.

Subroutine EcAcc_relational_information (object, cmn1, others, cmn2, OK, err)
Character(*)* object, cmn1, cmn2, others, err
Integer sector
Logical OK

For an *object* within a common *cmn1*, returns a list of related objects *others* within common *cmn2*. Not implimented in this release.

Subroutine EcAcc_request_ID (id, specific, object, common, nth, Nchar, char, info, sector, OK, err)
Integer id, nth, Nchar, sector
Character(*)* specific, object, common, char(*), err
Real info(*)
Logical OK

Similar to *EcAcc_request_Nth*, this routine seeks the *id* number rather than the index *nth* number.

Subroutine EcAcc_request_Nth (nth, specific, object, common, id, Nchr, chr, info, sector, OK, err)
Character(*)* specification, object, common, chr(*), err
Integer nth, id, Nchr, sector
Real info(*)
Logical OK

This subroutine fills the array *info* with the values of requested characteristics (*chr*) of the *nth specific object* of *common*. Checks are made that the *nth specific object* of *common* and the characteristics thereon exist. For example, for the 1st (*nth*=1) specified object (*specification*="INNER", *object*="HIT") in the sixth sector (*sector*=6), a request for only the energy (*Nchr*=1, *chr(1)*="ENERGY") would fill *info(1)* with the energy of that particular object. The *id* is the returned internal Ec ID number for the object.

Subroutine EcDrv_information (valid_specs, object, Nlabels, labels, OK, err)
Character(*)* valid_specs, object, labels(*), err
Integer Nlables, sector
Logical OK

Similar to *EcAcc_information*, but limited to the *EcDrv* common.

Subroutine EcDrv_Nfound (specify, object, Nfnd, sector, OK, err)
Character(*)* specify, object, err
Integer Nfnd, sector
Logical OK

Similar to *EcAcc_Nfound*, but limited to the *EcDrv* common.

Subroutine EcDrv_relate_id (id, spec_obj, object, Nids, ids, spec_other, other, sector, OK, err)
Integer id, Nids, ids(*), sector
Character(*)* spec_obj, object, spec_other(*), other(*), err
Logical OK

Similar to *EcAcc_relate_id*, but limited to the *EcDrv* common. Not implimented in this release.

Subroutine EcDrv_relational_information (object, others, OK, err)
Character(*)* object, others, err
Logical OK

Similar to *EcAcc_relational_information*, but only for the *EcDrv* common. Not implimented in this release.

Subroutine EcDrv_request_ID (id, specific, object, Nth, Nchar, char, info, sector, OK, err)
Integer id, Nth, Nchar, sector
Character(*)* specific, object, char(*), err
Real info(*)
Logical OK

Similar to *EcAcc_request_ID*, but restricted to the *EcDrv* common.

Subroutine EcDrv_request_Nth (Nth, specific, object, id, Nchar, char, info, sector, OK, err)
Integer Nth, id, Nchar, sector
Character(*)* specific, object, char(*), err
Real info(*)
Logical OK

Similar to *EcAcc_request_Nth*, but restricted to the *EcDrv* common.

Subroutine EcEvu_information (valid_specs, object, Nlabels, labels, OK, err)
Character(*)* valid_specs, object, labels(*), err
Integer Nlabels, sector
Logical OK

Similar to *EcAcc_information*, but limited to the *EcEvu* common.

Subroutine EcEvu_Nfound (specify, object, Nfnd, sector, OK, err)
Character(*)* specify, object, err
Integer Nfnd, sector
Logical OK

Similar to *EcAcc_Nfound*, but limited to the *EcEvu* common.

Subroutine EcEvu_request_ID (id, specific, object, Nth, Nchar, char, info, sector, OK, err)
Integer id, Nth, Nchar, sector
Character(*)* specific, object, char(*), err
Real info(*)
Logical OK

Similar to *EcAcc_request_ID*, but restricted to the *EcEvu* common.

Subroutine EcEvu_request_Nth (Nth, specific, object, id, Nchar, char, info, sector, OK, err)
Integer Nth, id, Nchar, sector
Character(*)* specific, object, char(*), err
Real info(*)
Logical OK

Similar to *EcAcc_request_Nth*, but restricted to the *EcEvu* common.

Subroutine EcFit_information (valid_specs, object, Nlabels, labels, OK, err)
Character(*)* valid_specs, object, labels(*), err
Integer Nlabels
Logical OK

Similar to *EcAcc_information*, but limited to the *EcFit* common.

Subroutine EcFit_Nfound (specify, object, Nfnd, sector, OK, err)
Character(*)* specify, object, err
Integer Nfnd, sector
Logical OK

Similar to *EcAcc_Nfound*, but limited to the *EcFit* common.

Subroutine EcFit_request_ID (id, specific, object, Nth, Nchar, char, info, sector, OK, err)
Integer id, Nth, Nchar, sector
Character(*)* specific, object, char(*), err
Real info(*)
Logical OK

Similar to *EcAcc_request_ID*, but restricted to the *EcFit* common.

Subroutine EcFit_request_Nth (Nth, specific, object, id, Nchar, char, info, sector, OK, err)
Integer Nth, id, Nchar, sector
Character(*)* specific, object, char(*), err
Real info(*)
Logical OK

Similar to *EcAcc_request_Nth*, but restricted to the *EcFit* common.

Subroutine EcSrv_pixel_info (id, Ndsc, dsc, info, layer, sector, OK, err)
Integer id, Ndsc, layer, sector
Character(*)* dsc(*), err
Real info(*)
Logical OK

For pixel#*id*, fills array *info* with the value of *Ndsc* descriptions *dsc* for a given *layer* and *sector*.

D.11 Simple Simulations

These routines are used for simple, quick testing of new code. They are not intended as a substitute for proper GEANT simulations.

Subroutine pseudo_EcSim (firstline, Ok)
Character(*)* firstline
Logical Ok

This interface drives simple, crude simulations for onboard testing of Ec. If the particle is not at the inner surface of the calorimeter, its path there is extrapolated along the momentum vector ignoring magnetic fields.

```
EcSim>                  simulate event
      /HElp                     list options
      /RESet                    reset Ec package
      /STRike                   strike Ec package
      /REQuest                  request a shower#
      /TECHnique                simulation technique
           :mthd                  "mthd"
      /SIMulate                 simulate requested showers
      /STAtus                   show status
          :n                      of nth request
          :All                    of all requests

      /SYStem                   set coordinate system
          :choice                 S123, XYZ, IJK
      /unit                     set energy unit scale
           :scale                 eV, KeV, MeV, GeV
      /PARTicle:type            set particle (e+, P, ...)
      /LOCation:a:b:c           location cm#
      /MOMentum:a:b:c           momentum#
      /TIME:t                   time nS
      /MAss:r                   mass
      /CHArge:q                 charge q
      /COMment:comment          comment on track#
      /SECtor:n                 set shower number
```

For example, to create two photons striking sector#3:

```
ECSIM> /reset                    !just to be safe
ECSIM> /unit:MeV                 !MeV energy units from now on
ECSIM> /system:IJK /sec:3        !local Ec system, sector#3
ECSIM> /mom:0:0:1000             !IJK momentum (0, 0, 1000) MeV/c
ECSIM> /loc:-20:-20:0            !I=-20 J=-20 K=0 cm
ECSIM> /particle:PHOTON          !sets mass & charge
ECSIM> /req                      !enter track#1 into list to simulate
ECSIM> /loc:+20:+20:0            !I=+20 J=+20 K=0 cm
ECSIM> /req                      !enter track#2 into list to simulate
ECSIM> /sim                      !simulate 2 EM showers; fill BOS bank EC
```

Subroutine EcDig_shower (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

After energy has been deposited into energy bins by *EcSimFast** routines, uses the calibration constants and attenuations to fill the EcEvu unpacked data arrays in *sector*.

Subroutine EcSim_prepare_from_XYZ (OK, err)
Logical OK
Character(*)* err

After tracks (particles) to be simulated have been requested by *EcSim_request_XYZ_shower*, this routine fills in all missing information required for the simulation.

Subroutine EcSim_reqd_tracks (method, allOK, err)
Character(*)* method, err
Logical allOK

After a list of tracks to simulate has been prepared (by *EcSim_prepare_from_XYZ*), actually fills energy bins in the detector space using a particular *method*. Only *method*="FAST" is supported.

Subroutine EcSim_request_XYZ_shower (desc, x, y, z, pX, pY, pZ, M, Q, T, sector, OK, err)
Real x, y, z, pX, pY, pZ, M, Q, T
Character(*)* desc, err
Integer sector
Logical OK

Adds a track (with comment *desc*) with momentum (pX,pY,pZ) GeV/c, mass M GeV/c^2, charge Q, and time T to the list of tracks to be simulated.

Subroutine EcSim_reset_all (OK, err)
Logical OK
Character(*)* err

Clears all simulation information.

Subroutine EcSimFast_EM_shower (E, T, i, j, k, dirI, dirJ, dirK, OK, err)
Real E, T, i, j, k, dirI, dirJ, dirK
Logical OK
Character(*)* err

Fills bins in the calorimeter for an electromagnetic shower of energy E beginning at time T at local coordinate (i,j,k) along axis (*dirI*,*dirJ*,*dirK*).

Subroutine EcSimFast_min_ion (E, T, i, j, k, dirI, dirJ, dirK, OK, err)
Real E, T, i, j, k, dirI, dirJ, dirK
Integer sector
Logical OK
Character(*)* err

Fills bins in the calorimeter for a minimum ionizing particle of energy E beginning at time *To* at local coordinate (i,j,k) along axis (*dirI*,*dirJ*,*dirK*).

Subroutine EcSimFast_reqd_tracks (allOK, err)
Logical allOK
Character(*)* err

Fills energy bins in the detector space using a very simple algorithm to describe energy deposition; overlaps all requested showers and uses *EcDig_shower* to produce unpacked data.

D.12 CLAS-CODA 2.0 Related Routines

The following routines work with the CLAS-CODA 2.0 format data. They will not be supported in the future.

Subroutine pseudo_EcEvu_CODA (firstline, Ok)
Character(*)* firstline
Logical Ok

This interface allows one to open, close, read, or write CLAS-CODA 2.0 format files.

```
EcEvu_CODA>              CODA CLAS 2.0 format I/O
      /HElp                    list options
file  /INput                   open CODA input
      /-INput                  close CODA input
file  /OUTput                  open CODA output
          :option                :Append
      /-OUTput                 close CODA output
      /GETevent[:nth]             get nth event
      /PUTevent                put out an event
      /STOre                   store into COMMONs
      /FETch                   fetch from COMMONs
      /REset                   clear CODA bank
```

Subroutine EcEvu_fetch (layer, sector, strt, buf, ptrs, OK, err)
Integer sector
Logical OK
Character(*)* err

Copies the EcEvu unpacked data array into a (properly defined) CLAS-CODA buffer *buf* beginning at *strt* with pointer table *ptrs* for one *layer* and *sector*. Updates *ptrs*.

Subroutine EcEvu_fetch_all (strt, buf, ptrs, OK, err)
Integer sector
Logical OK
Character(*)* err

Copies the EcEvu unpacked data array into a (properly defined) CLAS-CODA buffer *buf* beginning at *strt* with pointer table *ptrs* for all layers sectors. Updates *ptrs*.

Subroutine EcEvu_store (strt, buf, ptrs, sector, OK, err)
Integer strt, buf(*), ptrs(*), sector
Logical OK
Character(*)* err

Fills the EcEvu unpacked data array from a CLAS-CODA buffer *buf* beginning at *strt* with pointer table *ptrs* for one *layer* and *sector*.

Subroutine EcEvu_store_all (strt, buf, ptrs, OK, err)
Integer strt, buf(*), ptrs(*), sector
Logical OK
Character(*)* err

Fills the EcEvu unpacked data array from a CLAS-CODA buffer *buf* beginning at *strt* with pointer table *ptrs* for all layers and sectors.

D.13 Graphics

The Ec package graphics are based on CERNLIB HIGZ calls; the HIGZ window is assumed to exist and use the **physical** dimensions. All Ec graphics are done by layering one image atop another; only line segments are used. Problems of exceeding the window and scaling the picture are left entirely to HIGZ; all units are the CLAS standard (cm). Hence, a one meter line is 100 units long in the HIGZ window. The Ec graphics can plot in the CLAS XYZ, sector S123, and Ec local IJK coordinate systems. All graphics are three dimensional; there are no hidden lines.

For the EcGra routines **only**, *layer* may have special values. These are 0=both INNER and OUTER layers, 1=INNER, 2=OUTER, as well as **Ec_INNER** and **Ec_OUTER**. All sectors are represended by *sector*=0. Similarly, *axis*=0 means all, 1=U, 2=V, 3=W, as well as **Ec_U**, **Ec_V**, and **Ec_W**.

An object's representation is controlled by the options; one particular object may be selected, none, or all above a certain energy threshold. The energy scale used in the representation may also be set through *EcGra_option* or *pseudo_EcGra*.

Subroutine pseudo_EcGra (firstline, Ok)
Character(*)* firstline
Logical Ok

This interface allows controls of the Ec graphics. Commands not prefaced with a "/" are sent to *EcGra_option*.

```
EcGra> instruction/option
       /HElp                  list options
       /DEMOnstration         example of rotation
            :n                  steps
       /TUMble                another example of rotation
            :n                  steps
       /PLot                  generate picture
[file] /MEtafile:n            open type n metafile
       /SECtor:n              sector number (0=all)
       /LAYer:n               layer number (0=all)
       /LIMit:x1:x2:y1:y2     picture limits*
```

```
/Number:n               number of arguments*
/IDs:m                  id(s) list
/VALue:r                real value(s)
/DEGrees:r              degrees for value(s)
/RESet                  reset window
/CLear                  clear
/WOrkstation            specify workstation type
/THReshold:v            min. threshold in MeV*
/STAGe:n                pseudopixel level (0=all)
/PIXels                 pixel above threshold*
/HITs                   hits above threshold*
/SHOWers                showers above threshold*
/PEAKs                  peaks above threshold*
/STRIPs                 strips above threshold*
/TUBEs                  tubes triggered
```

For example, to size the window and plot sector#1 with the local Ec coordinate I to the left horizontally and J vertical:

```
ECGRA> /window:-400:400:-400:400 !window goes from -400 to +400 cm horz, vert
ECGRA> SCALE/val:0.010           ! .010 GeV/cm energy scale (set EcGra_option)
ECGRA> -Ihorz Jvert              !-I horizontal J vertical (set EcGra_option)
ECGRA> /sector:1 /plot           !only sector#1, plot now
ECGRA> /demo                     !walk the display around the vertical axis
```

Subroutine EcGra_ced_go (OK, err)
Logical OK
Character(*)* err

This routine calls the cLAS eVENT dISPLAY and passes it any available pixel information via special ced routines.

Subroutine EcGra_HitPlot (layer, sector, OK, note)
Integer layer, sector
Logical OK
Character(*)* err

Plots the hits for a given *layer* and *sector*.

Subroutine EcGra_join (N)
Integer N

Draws a polygon of N points loaded into the common within **EcGra.CMN** after performing all coordinate selections and rotations. This allows all *EcGra*plot* routines to be ignorant of the viewing coordinate system, viewing angle, etc.

Subroutine **EcGra_option** (option, N, ids, vals, layer, sector)
Character(*)* option, err
Integer N, ids(*)layer, sector
Real vals(*)

Attempts to process an *option*, possibly qualified by a list of *N ids* or values *vals*, for a *layer* and *sector*. The *options* are case insensitive. The option "?" will cause a list of supported options to be printed. Briefly, **XHORZ, XVERT, YHORZ, YVERT, ZHORZ, ZVERT, 1HORZ, 1VERT, 2HORZ, 2VERT, 3HORZ, 3VERT, IHORZ, IVERT, JHORZ, JVERT, KHORZ,** and **KVERT** select the CLAS XYZ, sector S123, or local Ec IJK frame. A "-" prefix denotes negation. Overall rotation of the image relative to the window is selected by **ROTVERT, ROTHORZ,** and **ROT**. The apparent depth of strips and pixels is controlled by **3D** (physical dimension) or **HIST** (energy representation). Display of individual classes of objects is set by **TUBEu, TUBEv, TUBEw, TUBE, STRIPu, STRIPv, STRIPw, STRIP, PEAKu, PEAKv, PEAKw, PEAK, HIT, PIXEL,** and **SHOWER**. The minimum energy object to display is set by **THRESH**; the conversion from energy to distance by **SCALE**. The special option **Default** sets all options to reasonable default values, and **Echoing** causes subsequent options to be repeated to FORTRAN IO channel#6.

Subroutine EcGra_PeakPlot (axis, layer, sector, OK, err)
Integer axis, layer, sector
Logical OK
Character(*)* err

Plots the peaks for a given *axis, layer* and *sector*.

Subroutine EcGra_pixelplot (layer, sector)
Integer layer, sector

Plots the pixels (above threshold or requested) for a given *layer* and *sector*.

Subroutine **EcGra_plot** (layer, sector)
Integer layer, sector

Plots everything according to the (previously made) options for *layer* and *sector*. If *layer* and *sector* are 0, all layers and sectors are plotted.

Subroutine EcGra_ShowerPlot (sector, OK, err)
Integer sector
Logical OK
Character(*)* err

Plots the showers for a *sector*.

Subroutine EcGra_TubePlot (axis, layer, sector, OK, err)
Integer axis, layer, sector
Logical OK
Character(*)* err

Plots the phototubes which fired for a given *axis*, *layer* and *sector*.

D.14 Command and Control

Interfaces allow the user to communicate with the Ec routines. They are based on the *SwGus_Qcommand* interface.

Subroutine EcFit_set_controls (cmd, variable, Iset, Rset, IO, OK, err)
Character(*)* cmd, variable, err
Integer Iset, IO
Real Rset
Logical OK

Given the name of an EcFit *variable* and a command *cmd*, will either set or recover the value of *variable*. If *cmd*="set", *variable* is changed to the approriate value (either *Iset* or *Rset*). If *cmd*='query', either *Iset* or *Rset* is given the value of *variable*. If *cmd*= "?", all known variables and their values are printed to channel#6.

Subroutine **pseudo_Ec** (firstline, Ok)
Character(*)* firstline
Logical Ok

This routine gives access to all other interfaces except *pseudo_EcEngine*. All Ec functions may be exercised, and it is the only point connecting EcGra, EcSim, ced, and CODA routines.

```
Ec> routine [instruction]
      /HElp                  list options
      /STATus                short report
      /DEBUG                 show selected
            :routine         debug "routine"
            :*               all flags
            :-               no flags
      /VERsion               report version number
      /TRace                 trace ON&dump
            :n                 to n.TRACE
            :REset             reset
      /-TRace                trace OFF&dump
      /EXit                  stop
```

```
operations: CED            event display
          : INITitialize   initialize geometry
          : FORce          forced reset
          : RESet          reset event

interfaces: ACCess           access info.
          : BOS              BOS event I/O
          : CALibrations     calibrations
          : CODA             CODA event I/O
          : FITting          analyze event
          : SIMulation       simulate event
          : RSLts            exported results
          : GRAphics         graphics
```

Subroutine pseudo_EcCal (firstline, Ok)
Character(*)* firstline
Logical Ok

Currently, calibration constants may come from either this routine or the EcCl BOS bank.

```
EcCal>                  calibration constants
      /HElp                     list options
      /SECtor:n                 set shower number
      /LAYer:m                  Ec layer (INNER or OUTER)
      /AXIS:desc                Ec axis (U, V, or W)
      /UNITs:scale              energy units (eV, KeV, MeV, GeV)
      /E0:value                 energy pedestal
      /Ech:value                energy/channel slope
      /T0:value                 time(nS) pedestal
      /Tch:value                time(nS)/channel slope
      /ATTenuation:length       attenuation_length(cm)
      /IDs[:id]                 show strip info.
      /SET[:id]                 set strip info.
```

To set all the calibrations in all strips the same:

```
ECCAL> /-layer /-axis /-sector          !deselect all
 layer:INNER - OUTER
 axis: U                          - W
 sector:             1 -             6

ECCAL> /Tch:0.1 /atten:400. /unit:MeV /Ech:3.448 !set values
 Tch:   0.1000000    nS/channel
 attenuation length:    400.0000     cm
 Ech:   3.4480002E-03 GeV/channel

ECCAL> /SET:1::36                                !set all PMT channels
ECCAL> /lay:INNER /axis:W /sec:3 /ID:13          !inquire of one PMT
 layer:INNER - INNER
 axis: W                          - W
 sector:             3 -             3
       id axis layer sector     Eo      Ech     To    Tch    atten. length
       13     W INNER  3 0.000000 0.003448  0.000  0.100       400.000

ECCAL>
```

Subroutine pseudo_EcEngine (firstline, EXIT)
Character(*)* firstline
Logical EXIT

This can call *pseudo_Ec* and hence all other Ec interfaces. It is used in the *Ec_engine** programs to select, plot, and/or halt on interesting events defined by a Ntuple entry being in or out of bounds. Ntuples are created from EcAcc vectors.

```
EcEngine>            analysis shell
   /HElp             list options
   /RETurn           normal return
   /EXIT             exit program properly
   /ABORT            stop suddenly
   /MAXimum:n        maximum number of objects/event
   /TECHnique:mthd   analysis method
   /Nevents:n        number events to process
```

```
    /ERRor:op        error action [HALT, CONT]
    /PLOT:op         plotting [TEST, ERR, ALL]
    /SECtors:m:n     selected sectors
    /STATus          show status
file/NTUPle[:id]     create Ntuple of EcAcc vectors
    /-NTUPle         properly close Ntuple
    /LIst[:m]        list Ntuple entries
    /LOw:x           lower limit
    /HIgh:y          upper limit
    /TEST[:op]       show tests ([HALT,CONT])
         :m            within bounds #m
         :-m           without bounds #m
    /-TEST           cancel tests
    /LUND            LUND header to Ntuple
         [:fmt]        CLAS-CODA, BOS style
    /-LUND           no LUND header

    EC               enter Ec interface
       cmnd            give Ec command
```

E Temporary Shells

From its inception, Ec was intended to be just a collection of routines for forward calorimeter reconstruction; it was never intended to "run" things. In the absence of a standard CLAS analysis shell, these are a few simple shells to call the Ec routines. They were intended for testing Ec and as examples of how to use Ec routines.

The simplest shell is *Ec_simple*; it opens a BOS 0. format data file, reads and reconstructs the data, and fills BOS banks. It has no other output and no options.

The *Ec* shell just calls the *pseudo_Ec* interface; all options and Ec functions may be exercised. It is intended only for event-by-event debugging and control.

The next shell is *Ec_engineI*; it uses the *pseudo_EcEngine* interface to specify all options. It reads BOS 0. format data files, its (optional) output is an HBOOK Ntuple built from EcAcc vectors; all options are accessable via the interface. All or some events may be analyzed; all or select events may be displayed or examined, and the same event may be reanalyzed using different options.

Ec_engineH is similar but reads CLAS-CODA 2.0 format data files.

E.1 Ec

This routine calls *pseudo_Ec*; it is intended for simple debugging and single event testing. If PAW is run before Ec, the HIGZ X-windows will always be established properly.

```
   EC> /help              !show options
........
   EC> /trace             !turn on SwGusTrace routines
   EC> INIT               !initialize
   EC> /version           !show current version
   EC> SIM  @e_1GeV.sim   !ECSIM> simulate 1 event
   EC> /status            !show COMMON block status
   EC> FIT  /anal:ALL     !ECFIT> analyze (using default algorithm)
   EC> GRA  /plot         !ECGRA> plot Ec picture
   EC> BOS  /list         !ECEVU_BOS> show BOS banks
   EC>
   EC> BOS  5k_elec.evt/input      !open BOS 0. file "5k_elec.evt"
   EC> BOS  /get /put              !get&store next event
   EC> FIT /tech:METRO1 /anal:ALL  !set new technique&analyze
   EC> /trace                      !show CPU times
   EC> CED                         !call ced event display
   EC> %                           !quit
```

E.2 Ec_engineH

Almost identical with *Ec_engineI*, this shell reads CLAS-CODA 2.0 format data files. It will not be supported in future releases.

E.3 Ec_engineI

This shell reads BOS 0. format files; its output is a fixed length CERNLIB HBOOK Ntuple. After the data file is opened, the *pseudo_EcEngine* interface is called for all setup; that interface can call the *pseudo_Ec* interface which in turn can call any other interface.

```
        Ec_engineI.............K.B.Beard 9/94

  Simple analysis engine for using the Ec3.2 package
  using BOS 0. format


  >>>for testing of Ec package-  type /HELP for help

 List of valid workstation types:
       0:  Alphanumeric terminal
    1-10:  Describe in file higz_windows.dat
  n.host:  Open the display on host (1 < n < 10)
    7878:  FALCO terminal
    7879:  xterm

 Metafile workstation types:
    -111:  HIGZ/PostScript (Portrait)
    -112:  HIGZ/PostScript (Landscape)
    -113:  HIGZ/Encapsulated PostScript
  -777/8:  HIGZ/LaTex

  workstation type?
1         !only integer understood, NOT #.node.address.net

 Version 1.20/11 of HIGZ started
  BOS 0. format input file?
e.evt

 OPEN BOSINPUT UNIT=22 FILE="e.evt" READ STATUS=OLD
 ECENGINE> @setup.ecengine       !run setup command file(s)
 ECENGINE> e.rzdat/NTUPLE !define output Ntuple
```

```
ECENGINE> /low:1.5/high:99999 /TEST:46 /TEST:CONT
ECENGINE> /PLOT:TEST
ECENGINE> /Nevents:100          !stop after 100 events
ECENGINE> %                     !begin analysis
.............
           100 events just processed
           100 events analyzed total
ECENGINE> Ec Gra /plot      !show last event
ECENGINE> /EXIT             !cleanly exit and close files
```

F Costs

The Ec analysis takes both CPU time and memory; estimates of both requirements are necessary. The SwGusTrace utilities were used to measure the CPU time; the memory requirements were determined by recompiling and relinking Ec with different **Ec_MAXpixels** in **Ec_general.PAR**. With pixels turned **off**, **Ec_MAXpixels**=1, Ec requires ~4 Mbytes of core; with all pixels **on**, **Ec_MAXpixels**=1911, Ec requires ~12 Mb. The following is from a run on 1 GeV pions on an HP 735 (clas02.cebaf.gov) with around 150 million instructions/second (MIPS).

```
 ECENGINE> Ec /trace
***begin: SwGusTrace_dump: ****
where                                                    #calls CPU time(se
Ec_recon_BOS..................................................  999   59.020
Ec_reset_all..................................................  999    3.170
Ec_reset..................................................... 5994    2.590
EcEvu_store_BOS_all...........................................  999    2.180
EcEvu_store_from_BOS......................................... 5994    1.030
EcEvu_reset.................................................11982     .360
EcRsl_fetch_BOS_all...........................................  999     .110
EcFit_analyze................................................ 1197   46.580
EcCal_sector................................................. 1197     .010
EcFitEdge_1st................................................ 1197   38.760
EcFitEdge1st_strips_peaks.................................... 7182   16.500
EcFit_peaks_hits............................................. 2394    1.120
EcGus_dalitz_rule............................................ 4098     .090
EcFit_hits_pixels............................................ 2394   15.670
EcFitEdge_est_reliability.................................... 2394     .130
EcSrv_complete_pseudopixels.................................. 1219   11.970
EcFit_hits_showers........................................... 1197     .070
EcDrv_keep_results........................................... 1197    2.820
EcRsl_file_results........................................... 1197    3.590
EcFit_fetch_to_BOS........................................... 1197    2.610
EcDrv_fetch_to_BOS........................................... 1197    3.150
EcSrv_pixel_info............................................. 3588     .110
EcDrv_reset.................................................. 1196     .020
```

```
EcFit_reset..................................................... 1196      .640
EcFitEdge_reset................................................. 1196      .060
EcFitPixel_reset................................................ 1196      .030
EcCal_reset..................................................... 1196      .100
*****end: SwGusTrace_dump: ****
```