CLAS-NOTE-96-014

Sc 2.2

TIME OF FLIGHT SCINTILLATOR
RECONSTRUCTION SOFTWARE

K.B.BEARD AND J.FICENEC

JULY 23, 1996

Sc 2.2 is the current release of the CLAS time of flight scintillator (Sc) event reconstruction software. It incorporates all essential functions and required data structures and supports the BOS event stuctures. This code links to RECSIS (the standard CLAS shell); it has a modular structure to allow easy modification of algorithms, but its current algorithms should be sufficient for the forseable future. This version does not include second pass analysis.

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 time of flight scintillators (Sc) first reconstruction package was written back in 1991; however, there have been so many changes in the reconstruction philosphy since than that no code was reused. This Sc package draws heavily on experience gained from the Ec [1] package and ought to link easily into the standard host program, RECSIS [2]. Sc consists of only a few modules with very clearly defined functions. The data input to Sc is via CLAS BOS 0. format [3], the output is via standard, RECSIS users' group (RUG), BOS banks described in the appendix.

Online documentation is maintained in World Wide Web (WWW) pages[4].

2 Important Concepts

2.1 Philosophy

The goal of the Sc package is to accomplish correct reconstruction of CLAS time of flight scintillators 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 `common`s, and the `common`s constructed to organize the analysis. This structure may be easily modified by adding or deleting parameters.

Sc 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

[1] Ec 3.2.1, Forward Electromagnetic Calorimeter Reconstruction Software, K.B.Beard, CEBAF CLAS-NOTE-94-024 (1995)

[2] RECSIS, `http::/www.cebaf.gov/clas/recsis/recsis.html`

[3] CLAS Event Format with BOS, Version 1.00, CLAS-NOTE 94-012 (1994)

[4] `http://www.cebaf.gov/ beard/Sc.html`

upgrades. Sc 2.2 represents a minimalist approach; it contains no graphics nor analysis code; it restricts its output to information about the hits.

2.2 Coding Comments

Sc has been written to conform to the CLAS software standards [5] with very few exceptions. Standardized nomenclature has been used throughout Sc; the first two letters identify the package, the next three the functionality. The new standards [6] have been incorporated from the start into the new code.

2.3 Terminology and Units

For the purpose of Sc, many details of the detectors' construction are ignored. Only the active region is considered, and the detector is considered homogenous. Not all units are the CLAS standard; angles are in radians, distances in centimeters, and time in nanoseconds, but unless otherwise specified energy is in normal minimum ionizing particle (NMIP) units.

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.

Each *paddle* consists of one or two scintillators ganged together for readout; the fact that it consists of discrete pieces of plastic with seperate photomultipliers is ignored. Each paddle has its own IJK coordinate system. It is defined such that the origion is in the center of the paddle, centered between the corners, with K a normal to the surface radially away from the target, and I in the CLAS sector midplane, and $J = K \times I$ (parallel to the major paddle axis). All the geometry is calculated from the **SCG** BOS bank during initialization.

A paddle is really the union of the signals from one (either phiward or anti-phiward) end of the paddle. It is characterized by an ADC and TDC value, given each event in the **SC** BOS bank. The calibrations are given in the **SCC** BOS bank are assumed to not change during a run. To conform

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

[6] `http://www.cebaf.gov/clas/online`

with common usage, **left** always corresponds with **anti-phiward** (or *counter-clockwise*), and **right** to **phiward** (or *clockwise*).

The reconstructed information forms a set of *hits*. Each hit contains time, energy, and position information.

2.4 Error Messages

The Sc package is still being developed; extensive testing is required. This release carries some 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[7] utilities used throughout Ec were not used in Sc for greater compatability with the new standards, however the same effect was achived via cruder means. Error messages are reported to `RECMES` and to the parent routine.

```
ScSrv_initialize>ScSrv_store_from_BOS: bad BOS bank "SCG "#1 does
NOT exist!
```

3 Input

The only supported data inputs are BOS banks; this version only uses the **SC**, **SCC**, and **SCG** BOS banks for input. Descriptions of the BOS banks appear in the appendix, CLAS-NOTE 94-012, and in the WWW pagespages [8].

4 Output

The results of Sc may be retrieved only via BOS banks. This version only outputs the **ScIf**, **SCI**, **ScRw**, and **SCRB** banks. The use of **Sc** instead of

[7] SwGus 1.1.2, SoftWare group General Utility Software, K.B.Beard, CEBAF CLAS-NOTE-94-023 (1994)

[8] `http://www.cebaf.gov/ beard/Sc.html`

SC denotes that these banks are *not* generally expected to be written out.

5 Reconstruction Proceedure

This reconstruction algorithm is very simple. All sectors are processed by `Sc_reconstruct_all`. For each sector, first, the **SC** bank is unpacked by `ScEvu_sector`; calibrations and time walk corrections and all other operations limited to one end are done in `ScCal_sector`. The formulas used in this version are:

$$T_{CAL} = C_{TD0} + C_{TD1}T_{TDC} + C_{TD2}T_{TDC}^2$$
$$T_{WC} = T_{CAL} + C_{TWO}A_{ADC} - (C_{TW1}/A_{ADC}^{C_{TW2}} - C_{TW1}/A_{FullScale}^{C_{TW2}})$$

Next, the hit positions are determined by combining information from the ends in `ScFit_sector`. No information from other packages is used up to this point. Inline documentation describes the form of the calibrations, time walk corrections, and hit position and time determination. In this version:

$$DISP_T = (v_L v_R(T_L - T_R) - (v_L - v_R)L/2)/(v_L + v_R)$$
$$DISP_A = (\lambda_L \lambda_R ln(A_R/A_L) - (\lambda_L - \lambda_R)L/2)/(\lambda_L + \lambda_R)$$

After looping over the sectors, `ScDrv_sector`, can reform the hits using information from external packages, if available. Sc 2.2 does not have that feature implimented in this release. This routine also organizes the hits for all sectors into a single time ordered list.

The information is placed into the appropriate BOS banks defined by `ScFitItm_BOS.CMN` (**SCI**), `ScFit_BOS.CMN` (**ScRw**), and `ScDrv_BOS.CMN` (**SCRB**).

The lists are maintained in various `COMMON`s for subsequent use in a second pass; to reset the internal lists before the next event, `Sc_reset_all` is called.

6 Linking

This package links into the standard RECSIS package. Always refer to the web page [9] for the latest version.

[9] `http://www.cebaf.gov/clas/recsis/recsis.html`

7 Simulation

This package includes no simulation routines; it does provide support for unpacking the **SCH** banks provided in the GSIM [10] simulation program in the `ScSim_store_from_BOS` routine.

8 Histograms

This package has a few debugging histograms defined; they are not part of the normal initialization or reconstruction, and are created using `ScGus_hist_init` and `ScGus_hist_fill`.

9 Future Directions and Comments

If known PMTs are dead, additional hit position information may be extracted from the data. In addition, information from ajoining bundles of strips may be used to improve hit information, and a hit spanning two bundles could be consolidated. It is intended by the author that this additional information be used in the `ScDrv_sector` routine and the dead tube information be used in the `ScFit_sector` routine.

[10] `http://www.cebaf.gov/clas/gsim/userguide.html`

A Sc BOS banks

BOS banks are the preferred way to transfer information between reconstruction packages. A brief description of each follows; all input banks' numbers are the relevant CLAS sectors, while all output banks, except **ScIf**, have only bank#1. All banks other than **SC** use 4 byte words. It is important to note that BOS bank names are case sensitive and that trailing blanks are significant and expected. For example, the name of the **SC** bank is "`SC  `", *not* "`SC`").

A.1 The SC raw data bank

The **SC** BOS bank contains the raw data produced by the ADC and TDC FASTBUS modules as 16 bit integer words. It is read by `ScEvu_store_from_BOS`. Note that *left* denotes counter-clockwise (CCW) ad viewed from the target, anti-phiward in the CLAS coordinate system. There is one bank for each sector which contains data.

```
name  format  elements        description
SC    B16     id              paddle#
              TDCl            raw TDC left
              ADCl            raw ADC left
              TDCr            raw TDC right
              ADCr            raw ADC right
```

A.2 The SCG Geometry Bank

The **SCG** BOS bank contain information on the size and position of the paddles. It is the *only* geometric input and is read by `ScSrv_store_from_BOS`. There is one bank per sector.

name	format	elements	description
SCG	2I,9F	id	paddle id#, 1-57
		panel	panel number - to which plane the paddle belongs
		CCW end X	X center of CCW end of paddle in CLAS system
		CCW end Y	Y center of CCW end
		CCW end Z	Z center of CCW end
		CW end X	X center of CW end
		CW end Y	Y center of CW end
		CW end Z	Z center of CW end
		WIDTH	width (cm) (~|| CLAS theta)
		THICK	thickness (cm) (~|| CLAS r)
		DELTA	difference in length of the higher-theta slab minus the lower-theta slab divided by 4 (cm)

A.3 The SCC Calibration Bank

The **SCC** BOS bank contains the information required to calibrate the strips. It is accessed only during initialization by `ScCal_store_from_BOS`, not event-by-event. There is one bank per sector.

name	format	elements	description
SCC	4I,44F	id	paddle id#, 1-57
		date	UNIX univiersal time of calibraton (32 bits)
		version	sequential version# of calibration
		status	4 byte status word (see appendix)
		TD0L	(Left) gives 0 time at center for tube (ns)
		TD0Lu	(Left) uncertainty (ns)
		TD0R	(Right) gives 0 time at center for tube (ns)
		TD0Ru	(Right) uncertainty (ns)
		TD1L	(Left) TDC**1 coefficient (ns/ch)
		TD1Lu	(Left) uncertainty (ns/ch)
		TD1R	(Right) TDC**1 coefficient (ns/ch)
		TD1Ru	(Right) uncertainty (ns/ch)
		TD2L	(Left) TDC**2 coefficient
		TD2Lu	(Left) uncertainty (ns/ch)
		TD2R	(Right)TDC**2 coefficient
		TD2Ru	(Right) uncertainty (ns/ch)
		TW0L	(Left) time walk constant (ns)
		TW0Lu	(Left) uncertainty (ns)
		TW0R	(Right) time walk constant (ns)
		TW0Ru	(Right) uncertainty (ns)
		TW1L	(Left) time walk 1st factor
		TW1Lu	(Left) uncertainty
		TW1R	(Right) time walk 1st factor
		TW1Ru	(Right) uncertainty
		TW2L	(Left) time walk 2nd factor
		TW2Lu	(Left) uncertainty
		TW2R	(Right) time walk 2nd factor
		TW2Ru	(Right) uncertainty
		ADPL	(Left) ADC pedestal (ch)
		ADPLu	(Left) uncertainty
		ADPR	(Right) ADC pedestal (ch)

ADPRu	(Right) uncertainty
ADSL	(Left) ADC scale (N.I.M.P./ch)
ADSLu	(Left) uncertainty (N.I.M.P./ch)
ADSR	(Right) ADC scale (N.I.M.P./ch)
ADSRu	(Right) uncertainty (N.I.M.P./ch)
VEFL	(Left) effective velocity of light (cm/ns)
VEFLu	(Left) uncertainty (cm/ns)
VEFR	(Right) effective velocity of light (cm/ns)
VEFRu	(Right) uncertainty (cm/ns)
ATNL	(Left) attenuation length (cm)
ATNLu	(Left) uncertainty (cm)
ATNR	(Right) attenuation length (cm)
ATNRu	(Right) uncertainty (cm)
TSIG0L	(Left) 1st parameter of measured resolution (ns)
TSIG0R	(Right) 1st parameter of measured resolution (ns)
TSIG1L	(Left) 2nd parameter of measured resolution
TSIG1R	(Right) 2nd parameter of measured resolution

A.4 The SCH Simulated Hit Bank

The **ScH** BOS bank contains information on the simulated hits from GSIM.
It is accessed by `ScSim_store_from_BOS`, and there is one bank per sector.

```
name  format  elements       description
SCH   7F,2I   x              x of entrance into time-of-flight mother volume
              y              y (cm)
              z              z
              cx             track x dir cosine
              cy             track y dir cosine
              cz             track z dir cosine
              pmom           track momentum (GeV/c)
              track          track number
              id             track Particle Data Group (PDG) id
```

A.5 The ScIf Information Bank

The **ScIf** BOS bank contains both calibration and geometric information on each paddle for use with the **SCI** and **SCRB** reconstructed hit banks, and is written by `ScInf_fetch_to_BOS`. There is one bank for each sector.

```
name  format  elements  description
ScIf  2I,22F  SECTOR/ID paddle id# in least significant 2 bytes,
                        sector# in most significant 2 bytes
              STATUS    general status information (see appendix)
              Xo        CLAS X of center of paddle
              Yo        CLAS Y
              Zo        CLAS Z
              Ix        CLAS X component of I unit vector
              Iy        CLAS Y component
              Iz        CLAS Z component of I unit vector
              Jx        CLAS X component of J unit vector (major axis)
              Jy        CLAS Y component of J unit vector (major axis)
              Jz        CLAS Z component of J unit vector (major axis)
              Kx        CLAS X component of K unit vector (minor axis)
              Ky        CLAS Y component of K unit vector (minor axis)
              Kz        CLAS Z component of K unit vector (minor axis)
              delW      dist. center to surface of paddle || I (width/2) (cm)
              delL      dist. center to surface of paddle || J (length/2) (cm)
              delT      dist. center to surface of paddle || K (thickness/2) (cm)
              nmip      energy deposit for a normal min. ion. particle (GeV)
              VELavg    average effective speed of light in paddle (cm/nS)
              ATNavg    average attenuation length in paddle (cm)
              velL      (Left) effective speed of light in paddle (cm/nS)
              velR      (Right) effective speed of light in paddle (cm/nS)
              atnL      (Left) attenuation length in paddle (cm)
              atnR      (Right) attenuation length in paddle (cm)
```

A.6 The SCI Fitted Intermediate Results Bank

The **SCI** BOS bank contains intermediate reconstructed results for all sectors. It is written by `ScFitItm_fetch_to_BOS`.

name	format	elements	description
SCI	2I,6F	SECTOR/ID	paddle id# in least significant 2 bytes, sector# in most significant 2 bytes
		STATUS	general status information (see appendix)
		TL	(Left) calibration-corrected time, such that all TOF counters are timed relative to each other (nS)
		AL	(Left) calibration corrected and normalized to normal incidence minimum ionizing energy deposit
		TR	(Right) calibration-corrected time, such that all TOF counters are timed relative to each other (nS)
		AR	(Right) calibration corrected and normalized to normal incidence minimum ionizing energy deposit
		POST	position from center along paddle as determined from the time (cm)
		POSA	position from center along paddle as determined from the energy (cm)

A.7 The ScRw Fitted Working Results Bank

The **ScRw** BOS bank contains reconstructed results for all sectors. It contains similar information, but at the first pass level, as the **SCRB** bank and is written by `ScFit_fetch_to_BOS`.

name	format	elements	description
ScRw	2I,18F	SECTOR/ID	paddle id# in least significant 2 bytes, sector# in most significant 2 bytes
		STATUS	general status information (see appendix)
		HIT/PSST	HIT gives status information on the hit; PSST is the processing status (see appendix)
		Thit	time w.r.t. trigger start (nS)
		dThit	uncertainty in Thit (nS)
		Ehit	energy deposit (GeV) (corrected for attenuation)
		dEhit	uncertainty in Ehit (GeV)
		Qhit	estimated quality of hit (1.00=ideal)
		Xhit	CLAS X position derived from the time (cm)
		Yhit	CLAS Y
		Zhit	CLAS Z
		dIx	CLAS X component of dI (paddle width/2) (cm)
		dIy	CLAS Y
		dIz	CLAS Z
		dJx	CLAS X component of dJ (hit width/2) (cm)
		dJy	CLAS Y
		dJz	CLAS Z
		dKx	CLAS X component of dK (paddle thickness/2) (cm)
		dKy	CLAS Y
		dKz	CLAS Z

A.8 The SCRB Drived Working Results Bank

The **SCRB** BOS bank contains best and latest reconstructed results for all sectors. It represents the highest level of analysis available, but on the first pass contains the same information as the **ScRw** bank. There is one time-ordered bank for all sectors, and it is written by `ScDrv_fetch_to_BOS`.

name	format	elements	description
SCRB	2I,9F	SECTOR/ID	paddle id# in least significant 2 bytes, sector# in most significant 2 bytes
		STATUS	general status information (see appendix)
		HIT/PSST	HIT gives status information on the hit; PSST is the processing status (see appendix)
		TOF	time of flight (nS) (if no tracking is available, it is the time relative to the trigger rather than the flight time)
		dTOF	uncertainty in TOF (nS)
		EDP	energy deposited (GeV)
		dEDP	uncertainty in EDP (GeV)
		POST	position derived from the time measurements (cm)
		dPOST	uncertainty in POST (cm)
		POSA	position derived from the amplitude measurements (cm)
		dPOSA	uncertainty in POSA (cm)

B Sc COMMON blocks

The time-of-flight counter (Sc) 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 Sc package. Each step of analysis is characterized by separate `common` blocks:

```
DESCRIPTION                       COMMON name

geometry:                         ScSrv_geometry
calibrations:                     ScCal_calibrations
unpacked raw event:               ScEvu_data
reconstructed (standalone):       ScFit_hits
reconstructed (best):             ScDrv_hits
```

C Sc Parameters

The Sc code is constructed on a very carefully chosen parameter space. This space provides the organization used throughout all routines within Sc; it has *no* bearing on the BOS banks and is *only* for internal use.

C.1 Sc General Parameters

The Sc package uses parameters to define the relevant characteristics of objects; not all objects use all parameters. While these parameters are only for internal Sc use, their descriptions are included here and in `Sc_general.PAR`. For convenience, `Sc_local_ONLY.PAR` proves short parameter names (labels) for use *within* Sc. Note that parameter `Sc_Gev == 1`; this was done for notational clarity (for example, 5.*GeV is 5 GeV in the standard CLAS units).

```
parameter       local name       description
----------------------------------------------
Sc_TDC          TDC              raw TDC value
Sc_ADC          ADC              raw ADC value
Sc_time         TIME             CLAS time (ns)
Sc_unTIME       unTIME           uncertainty in time (ns)
Sc_NMIP         NMIP             energy in Normal Minimum Ionizing Particle
                                 equivalents
Sc_unNMIP       unNMIP           uncertainty in energy (NMIP)
Sc_GeV          GeV              energy in GeV
Sc_unGeV        unGeV            uncertainty in energy (GeV)
Sc_DISPt        DISPt            displacement from center using time (cm)
Sc_unDISPt      unDISPt          uncertainty in time displacement (cm)
Sc_DISPa        DISPa            displacement from center using energy (cm)
Sc_unDISPa      unDISPa          uncertainty in energy displacement (cm)
Sc_quality      QUALITY          quality (ideal=1.0)
Sc_width        WIDTH            transverse axis (cm)
Sc_length       LENGTH           longest axis (cm)
Sc_thickness    THICK            shortest axis (cm)
Sc_step         step             step in ganged slabs (cm)
```

```
Sc_X             X              CLAS X axis
Sc_Y             Y              CLAS Y
Sc_Z             Z              CLAS Z
Sc_I             I              local I axis (~|| CLAS theta)
Sc_J             J              local J axis (~|| CLAS phi at midplane)
Sc_K             K              local K axis (~|| CLAS r)

Sc_antiPHIward   LEFT CCW       counter-clockwise as viewed from target
Sc_middle        MID            middle
Sc_PHIward       RIGHT CW       clockwise as viewed from target

Sc_low           LOW            low value
Sc_mean          MEAN           mean value
Sc_high          HIGH           high value
```

C.2 Sc Local Parameters

While prefixing all parameters with `Sc_` conforms to the standards, it is more convinient to define more convenient parameters for local use. In addition to those shown previously, additional ones are provided by `Sc_local_ONLY.PAR`.

```
parameter                 description
----------------------------------------------
TT                        pi- a fundamental constant
degree                    radians/degree
infinity                  practical infinity

MeV                       MeV energy unit
KeV                       KeV energy unit
nSec                      nS standard CLAS time unit
pSec                      picosecond time unit
centimeter                cm standard CLAS distance unit
meter                     meter length unit

byte                      size of one 8 bit byte (256)
halfword                  size of a half word (2 bytes) (256*256)

speed_of_light            speed of light (cm/ns)

index_of_refraction       index of plastic scintillator

speed_in_plastic          speed of light in plastic (cm/ns)

density_of_plastic        density of plastic (g/cm^3)

minimum_ionizing_dEdM     energy loss for a m.i.p. (GeV/(g/cm^3))

dEdX_minimum_ionizing     energy loss for a m.i.p. in plastic (GeV/cm)
```

C.3 Sc Calibration Parameters

The calibration constants for Sc are numerous enough that a set of parameters exists to define them in `ScCal_local.PAR`.

```
parameter                 description
----------------------------------------------
TD0                       gives 0 time at center for tube (nS)
TD1                       TDC**1 coefficient (nS/ch)
TD2                       TDC**2 coefficient
TW0                       time walk constant (nS/ADCch)
TW1                       time walk 1st factor
TW2                       time walk 2nd factor
ADP                       ADC pedestal (ch)
ADS                       ADC scale relative to normal min. ion. (NMIP/ch)
VEF                       effective velocity (cm/nS)
ATN                       attenuation length#1 (cm)
TSIG0                     1st of measured resolution (ns)
TSIG1                     2nd of measured resolution
```

D Subroutines and Functions

The progamming conventions used by Sc are consistent; *sector* is the CLAS sector and has a value of 1 to 6.

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.

The only routines expected to be called by by RECSIS are `ToF_init`, `ToF_bevt`, and `ToF_evnt`, which in turn respectively call `Sc_initialize_all`, `Sc_reset_all`, and `Sc_reconstruct_all`.

D.1 Initialization and Cleanup

These routines are used to calculate and fill the internal `commons` used in the reconstructions with geometric and calibration constants. Success or failure is returned by *OK* and an explaination for failure is placed in *err*.

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

This routine performs all geometric and calibration initializations for all sectors. It calls ScSrv_default_fetch_to_BOS, ScSrv_initialize, ScCal_default_fetch_to_BOS ScCal_initialize, and ScInf_fetch_to_BOS for each sector. It then calls Sc_reset_all just to ensure the arrays are empty.

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

This routine determines the end points and corners of all paddles as well as the local coordinate system for each paddle. The quantities are stored in the ScSrv_geometry `common`. It should be called once per *sector*.

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

This routine stores the calibration constants for each paddle within *sector*, and calculates various quantities stored in the ScCal_calibrations `common`. It must be called once per *sector*.

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

This routine creates and fills the **ScIf** BOS bank for each *sector*.

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

This routine zeros the length of all dynamic internal arrays in Sc, and deletes the **SCI**, **ScRw**, and **SCRB** banks, if they exist. It should be called before beginning a new event.

D.2 Event Reconstruction

The event reconstruction has only a simple algorithm. Various corrections and improvements can easily be added later.

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

This routine processes the data for all sectors, and should be called once per event. First, it loops over the sectors and the TDCs and ADCs from the paddles by calling ScEvu_store_from_BOS, then calls ScCal_sector to perform calibrations and corrections which only depend on one end of the paddle. Then ScFit_sector is used to determine hit information without input from any other package. ScDrv_sector then may use external information to recalculate the hits. The results are finially stored into BOS by ScFitItm_fetch_to_BOS, ScFit_fetch_to_BOS, and ScDrv_fetch_to_BOS, sorted by the increasing time of the hit.

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

Given the data already stored in the internal `commons`, this routine rescales the raw data and performs the time walk corrections. It reads the ScEvu_data `common` and partially fills the ScFit_hits `common`.

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

Given the data already stored in the ScFit_hits `common`, this routine creates and determines characteristics of the hits using local information only. It stores the results back into the ScFit_hits `common`.

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

Given the data already stored in the ScFit_hits `common`, this routine may use external information to refit the hits. It stores the results into the ScDrv_hits `common`.

D.3 BOS interface routines

All inter-package communication is via the BOS utilities. These routines connect Sc to BOS banks. The BOS input banks are as described in CLAS-NOTE 94-012 (and subsequently modified by RUG) and the output banks defined here. Banks that begin with either **sc** or **Sc** rather than an **SC** are generally *not* intended to be written out.

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

If, and only if, the **SCG** BOS bank does not exist, this routine creates one with default values for a *sector*.

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

This routine moves the information from the geometry BOS bank (**SCG**) containing the location and size of the paddles into ScSrv_geometry `common` during initialization.

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

If, and only if, the **SCC** BOS bank does not exist, this routine creates one with default values for a *sector*.

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

This routine moves the information from the calibration BOS bank (**SCC**) containing the both energy and time calibrations into ScCal_calibrations `common` during initialization.

Subroutine ScEvu_store_from_BOS (fnd, sector, OK, err)
Integer sector
Logical fnd, OK
Character(*)* err

This routine moves the information from the raw BOS bank (**SC**) containing ADC and TDC values into the ScEvu_data `common`. *fnd* indicates whether data was found in this *sector*.

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

Creates and fills the **SCI** BOS bank with information from the ScFit_hits `common`.

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

Creates and fills the **ScRw** BOS bank with information from the ScFit_hits `common`.

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

Creates and fills the **SCRB** BOS bank with information from the ScDrv_hits `common`.

D.4 Utilities

These are common mathematical and utility functions used by Sc.

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

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

Subroutine ScGus_sort_increasing (Nv,v,ordered)
Integer Nv, ordered(*)
Real v(*)

Given a list of real values *v(*)* of length *Nv*, it returns a list *ordered* containing the order of the values sorted in increasing order.

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

Given a local coordinate (*i,j,k*) of paddle#*id* in a *sector*, returns the standard CLAS coordinate (*x,y,z*).

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

Given a the CLAS standard coordinate (*x,y,z*) returns the CLAS spherical coordinate (*r,theta,phi*).

Subroutine ScGus_sph_XYZ (r,theta,phi, x,y,z)
Integer id, sector
Real r,theta,phi, x,y,z

Given a the CLAS spherical coordinate (*r,theta,phi*), returns the standard CLAS coordinate (*x,y,z*).

Integer Function ScGus_make_HitPsst (Hw,End,asID,nthH,asEnd, Dc,Ec,Cc, St,qID)
Integer Hw,End,asID,nthH,asEnd, Dc,Ec,Cc, St,qID

This routine packs the bits for the HIT/PSST word used in the **ScRw** and **SCRB** banks. The bits are packed with the inputs:
Hw- the number of this hit within paddle#ID, either 1 or 2,
End- Left only (1), Right only (2), or Left and Right (3) channels were used to obtain the information),
asID- an associated paddle#ID used for this hit,
nthH- 1st or 2nd hit number for this other paddle# ID,
asEnd- Left only (1), Right only (2), or Left and Right(3) (crossings) associated paddle ID determination,
Dc- associated track in DC used, 00=>DC not used),
Ec- associated hit in EC used, 00=>EC not used),
Cc- associated hit in CC used, 00=>CC not used),
St- associated hit in ST used, 00=>ST not used), and
qID- overall quality assignment and processing passes, or 8 bit ID.

Integer Function ScGus_make_Status (hitID,id,sector)
Integer hitID,id,sector

Packs the *hitID*, paddle#*id*, and *sector* into the status word used for the **ScIf**, **SCI**, **ScRw**, and **SCRB** banks.

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

Defines histograms used for debugging Sc; it does *not* make the `HLIMIT` call. It creates the histograms using `HBOOK1` and `HBOOK2` from within the //PAWC/SC subdirectory.

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

Fills the histograms defined by ScGus_hist_init.

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

This routine is strictly for debugging use. It uses RECMES calls to print out the contents of all **SCH**, **SCI**, **ScRw**, and **SCRB** banks. For each paddle hit in **SCI**, ScGus_paddle_info is called.

Subroutine ScGus_paddle_info (id,sector,OK,err)
Integer id, sector
Logical OK
Character(*)* err

This routine is strictly for debugging use. It uses RECMES calls to print out the contents of all important quanties within the ScSrv_geometry and ScCal_calibration commons for a given paddle#*id* in a *sector*.

Subroutine ScSim_store_from_BOS (N,x,y,z,dx,dy,dz,P,tk,pID,OK,err)
Integer N, tk(*),pID(*)
Real x(*),y(*),z(*),dx(*),dy(*),dz(*),P(*)
Logical OK
Character(*)* err

This routine reads the **SCH** simulation BOS bank, sorts the values according to momentum, and returns the array of *N* hits.

E Special Words

To conserve space, special integer longwords are packed with multiple bits of information. The two words used in this version are **HIT/PSST** and **STATUS**, and are packed using the `ScGus_make_HitPsst` and `ScGus_make_status` routines respectively.

E.1 STATUS Word

The **STATUS** word is used in the **ScIf**, **SCI**, **ScRw**, and **SCRB** banks.
From the *least* to *most* significant bits:

```
6 bits-for sequential hit number (unused in SCC)
2 bits-for channels fully functional (00 neither, 10 left only, 01 right
       only, 11 both)
2 bits-HV Left (00 both ganged tubes off, 01 low theta on, 10 high theta on,
       11 both on)
2 bits-HV Right (00 both ganged tubes off, 01 low theta on, 10 high theta on,
       11 both on)
2 bits-New Gain Match (00 neither, 10 left only, 01 right only, 11 both)
2 bits-New TDC slope and quadratic parameters  (00 neither, 10 left only,
       01 right only, 11 both)
2 bits-New ADC slope and pedestal parameters (00 neither, 10 left only, 01
       right only, 11 both)
2 bits-New Time Walk corrections (00 neither, 10 left only, 01 right only,
       11 both)
2 bits-TDC Quality (00 both poor, 10 left good, 01 right good, 11 both good)
2 bits-ADC Quality (00 both poor, 10 left good, 01 right good, 11 both good)
8 bits-(least significant 8 bits-of) calibration version# (from SCC)
```

New implies that these parameters were input on the date specified in **SCC**; otherwise they were taken from a previous calibration. For paddle IDs#40 and above, two tubes are ganged together into a single ADC and TDC channel. Each tube is connected to a single slab.

E.2 HIT/PSST Word

The **HIT/PSST** word is used in the **ScRw** and **SCRB** banks. From the *least* to *most* significant bits:

```
2 bits-the number of this hit within paddle#ID (1 or 2)
2 bits-to indicate if Left only, Right only, or Left and Right
       channels were used to obtain the information
6 bits-if there is an associated paddle#ID used for this hit
2 bits-indicate the (1st or 2nd) hit number for this other paddle#ID
2 bits-indicate if Left only, Right only, or Left and Right channels
       were used in the associated paddle ID for the hit (crossings)
       associated paddle ID determination
2 bits-spare in this half-word
2 bits-associated track in DC (cell tracking vs. full tracking 00=>DC
       not used)
2 bits-associated hit in EC (electromagnetic vs. hadronic 00=>EC not used)
2 bits-associated hit in CC (include quality of assignment 00=>CC not used)
2 bits-associated hit in ST (include quality/sector bridge 00=>ST not used)
8 bits-overall quality assignment and processing passes, or use for ID
```

F Costs

This Sc package does not use the SwGusTrace utilities used by the Ec package to determine the relative cost of each routine. This the entire package is a minimum implimentation and uses little memory or CPU resources.

G Acknowledgements

The vast majority of this work was not supported by any institution outside of the use of CEBAF facilities. This author (KBB) wishes to thank Dr.s Burkert and Funsten for their steadfast support.