CLAS-NOTE-97-004

Ls 2.0

RECONSTRUCTION PACKAGE
LINKING SOFTWARE
K.B.BEARD

MAY 15, 1997

Ls 2.0 is the first release of the CLAS reconstruction package Linking Software. It incorporates all the essential functions and required structures and provides a flexible and robust connection between the BOS banks from various reconstruction packages. This code links to RECSIS (the standard CLAS shell); it has a modular structure to allow easy modification of algorithms.

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.

Many authors with a wide variety of preferences have written the software used to reconstruct the events in each of the detector packages. A lamentable lack of standardization has developed; very little of the standardization put forth in 1990 [1] has survived, and a consensus has not yet emerged.

Some packages were documented [2] [3] [4] [5] [6] [7] well, others to a much lesser extent [8] [9] [10].

The Ls package began as Fu, the **Fu**ll detector package, but due to vociferous objections to the name and a shift in concept from a package that didn't depend on the Dc tracking into one that did, the name was changed to Ls, the **L**inking **s**ystem.

This package draws heavily on experience gained from the Sc and Ec packages and links easily into the standard host program, RECSIS [11]. Ls consists of only a relatively few modules with very clearly defined functions.

The *only* communication between the world and Ls is via Tcl shell variables used in RECSIS and CLAS BOS 0. format BOS banks described in the appendix. Online documentation is maintained in World Wide Web (WWW)

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

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

[3] Sc 2.2, Time of Flight Scintillator Reconstruction Software, K.B.Beard and J.Ficenec, CEBAF CLAS-NOTE-96-014 (1996)

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

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

[6] `http://www.cebaf.gov/~ficenec/Sc.html`

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

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

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

[10] `http://www.cebaf.gov/~manak/offline.html`

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

pages [12].

2 Important Concepts

2.1 Philosophy

The goal of the Ls package is to accomplish correct linking of the reconstructed information from the various detector packages of CLAS using clean, structured, maintainable, **flexible** and documented code. Conceptually, only a few instructions for Ls are stored in `COMMON` blocks; all other information is stored in BOS banks. Ls has a somewhat different philosophy than Ec and Sc; in those packages, the BOS banks were used only for external communication; in Ls, all event-by-event communication is exclusively via the BOS banks. Only some configuration information is kept in local `COMMON`s.

Even the descriptions of the BOS banks accessed by Ls are BOS banks; in this way, other packages have freedom to alter their banks on the fly and have Ls accomodate itself to the changes.

Ls 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. Ls 2.0 represents a minimalist approach; it contains no graphics, only geometric linking, and a very simple electron identificaion. It is intended to provide a robust framework for other packages to use and develop.

2.2 Coding Comments

Ls has been written to conform to the CLAS software standards [13] with very few exceptions. Standardized nomenclature has been used throughout Ls; the first two letters identify the package, the next three the functionality. The newer standards [14] have been incorporated from the start into the new code. A few routines have been borrowed from the SwGus[15] package without

[12] `http://www.cebaf.gov/~beard/Ls/`

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

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

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

modification.

2.3 Terminology and Units

Ls tries to keep everything in the CLAS standard units; angles are in radians, distances in centimeters, and time in nanoseconds. This version supports five coordinate systems in common use. The preferred system in Ls is **CLAS**, followed by **SCTR**.

The CLAS XYZ coordinate system (**CLAS**) 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 (**SPHR**) is the normal compliment to the XYZ system.

The sector-based coordinate system used by SDA (**SCTR**) is also right handed, S1 is parallel to the beam axis, S2 is through the center of the sector, and S3 is S1×S2. For sector#1, S1=Z, S2=X, and S3=Y, while for sector#4, S1=Z, S2=-X, and S3=-Y.

The newer sector-based coordinate system used by TRKL (**SCTN**) is also right handed, S3 is parallel to the beam axis, S1 is through the center of the sector, and S2 is S3×S1. For sector#1, S3=Z, S1=X, and S2=Y, while for sector#4, S3=Z, S1=-X, and S2=-Y.

The spherical form of the **SCTN** system is **SPHS**.

For the purposes of Ls, the *track* is the pattern that Ls is trying to fit, every other chunk of information (including the information from the source feeding Ls) is called a *hit*. The matching routines (*LsTkc_*_match*) produce a bank linking each track to some number of hits; the inverse tables of hits to tracks is generated in *LsDrv_fetch*. Every hit and track is accounted for; no hit nor track is ever discarded by Ls.

The version number for Ls uses a simple scheme of the form A.B.C; where A is a major revision or change to the parameter space, B is a change to any internal `COMMON`s, and C are bug fixes limited to a few routines. This release is version Ls 2.0.7.

2.4 Error Messages

The Ls 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. Some SwGus[16] utilities have been used for convenience. Error messages are reported to the parent routine and eventually to `RECMES`. Ls never attempts to send long error messages to the `RECMES` routine; it will crash the program if too long an error is reported to it.

An example error message might be:

```
Ls_initialize>LsTkc_initialize: Sc job control "LSJC"#49 not found
```

3 Input

The only supported data inputs are BOS banks and Tcl variables; the instructions to Ls are in the form of **LSJC** Ls Job Control banks; they describe the name and characteristics of all the other banks Ls needs to read. Descriptions of the BOS banks appear in the appendix, and in the WWW pages [17].

4 Output

The results of Ls may be retrieved only via BOS banks. All of the banks created by Ls are intended to be accessable by the other packages. The *LsRsl_fetch* routine creates a custom bank in whatever form is described by `LsRsl_BOS.CMN` (currently **TRKS**). The main index bank, **LSIX**, contains pointers to other banks, and separate banks of track-to-hit and hit-to-track tables for each package are created. Each master tracking bank and master index bank contains only one track; the indices run from 1 to N.

5 Reconstruction Proceedure

The flow of Ls is very straightforward. Upon initialization, the **LSJC** control bank is sought for each package and the source; if it is found, the package will

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

[17] `http://www.cebaf.gov/~beard/Ls/`

be processed. Each event, the source bank is sought; for each entry in each source bank a filled master tracking bank **LSTX** and an empty master index bank **LSIX** is generated. The calls to the *LsTkc_*_match* geometric matching routines fill the master index bank and create the track-to-hit banks. A call to the classification routine *LsCls_eTest* also fills in the master tracking bank. Then the hit-to-track banks are generated by *LsDrv_fetch* and a generic output by *LsRsl_fetch*. In Ls 2.0.7, the generic output bank is **TRKS**.

The matching proceedure is very simple in this version; a source gives the expected impact points in each detector package, and Ls tries to find the best match between the expected positions and the positions of the hits by calculating displacements and then sorting the possibilities by increasing size. Ls Tcl variables set an upper limit, after which the combination is discarded.

6 Linking

This package links into the standard RECSIS package. Always refer to the web page [18] for the latest version.

7 Histograms

This package has a one row-wise Ntuple defined; it is intended only for debugging and testing and is manipulated via the *LsHis_** routines.

8 Future Directions and Comments

In the immediate future, Ls would benefit from the addition of instructions on unscrambling bit-packed quantities in BOS banks; room was left for that purpose in the LSJC definitions. This would be best done by including this feature into the *LsGus_Nrecover* and *LsGus_recover* routines and modifying the *LsTkc_*_match* to use those routines. That would greatly simply those routines and code maintainence.

Another improvement would be to modify *LsRsl_fetch* to take its instructions also from a **LSJC** bank.

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

9 Acknowledgements

The author wishes to express his gratitude to Dr. R. Minehart and the University of Virginia for their financial support of this project, and to James Madison University and Dr. V. Burkert of the Thomas Jefferson National Accelerator Facility for additional support.

A Ls BOS banks

BOS banks are the preferred way to transfer information between reconstruction packages. The BOS scheme reserves a contiguous section of memory called a *bank* in a large FORTRAN `COMMON`; all boundary checking and so on is left to the users to enforce. The memory may be treated as `INTEGER*2`, `INTEGER*4`, `REAL`, `CHARACTER`, and so on as needed. Each bank is indentified by a one (4 byte) word *name* (treated as `CHARACTER*4` by BOS) and a one word *index* (treated as `INTEGER*4`). The standard usage only supports upper case names without special characters (such as nulls); 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`").

Some banks only have index#0, others only #1, still others #sector (1-6), and others #1-n. While the standard version of BOS disallowed bank indicies less than 1, the BOS libraries used by CLAS have been modified to support index#0.

If a structure for the bank is defined, each *entry* represents a repeat of that structure. A bank may have one or more entries. Generally, the bank name, index, and entry are sufficient to uniquely describe an object in BOS.

Also, one should note that Ls often uses a header on its BOS banks; the first entry may have a different structure, but always the same size, as all subsequent entries. This is not supported by the DATMAN routines used by RECSIS to list the banks, but works fine under BOS. Also, some of the current RECSIS utilities (such as `bdump`) crash the program when used on large Ls BOS banks.

A brief description of each Ls BOS bank follows.

A.1 The LSJC Ls Job Control bank

Conceptually, the Linking System Job Control (LSJC) BOS bank is a static bank and is intended to provide a instructions to the rest of the Ls package about which banks to use. There exists one LSJC bank per bank description. The index is the same as the package id number (see `Ls_general.PAR`), while the structure is defined by `LsTkc_BOS.CMN` and in `lsjc.ddl`.

The header contains general information about the bank; its name, highest index allowed, size, coordinate system, etc. The body is repeated for each keyword of interest; for example, the position of a hit in the forward calorimeter may be described by the keywords X, Y, and Z. The keywords use the Ls parameter numbers for identification; quantities associated with two parameters (such as X.error.Ec) can use the upper halfword of the "keyID" field (see *LsGus_packI4* and *LsGus_unpackI4*). The routine *LsJcl_dump* sends a listing of the file to RECMES. An example appears in Appendix C.3.

```
header...........
     pkgNAME- [I*4<=>C*4] package name ("Dc ", ..., "Ec ", etc.)
     BOSname- [I*4<=>C*4] BOS bank name
     BOSsize- [I] size of 1 element of bank in (4 byte) words
     BOSrange- [I] maximum number of this bank (1, 6, ... N)
     BOScoord- [I*4<=>C*4] coordinate system name ("CLAS","SCTR",...)
     reserved
     reserved
     reserved
     reserved
     reserved
     reserved
body.............
     keyNAME- [I*4<=>C*4] keyword name
     keyFORMAT- [I*4<=>C*4] keyword format
     keyLOCATION- [I] first keyword location within bank
     keySCALE- [F] scaling factor to standard CLAS units
     keyID- [I] Ls parameter ID#, packed into the upper and lower halfword.
     reserved
     reserved
     reserved
     reserved
     reserved
     reserved
     reserved
```

A.2 The LSTX Master Tracking Bank

The **LSTX** BOS bank is created using information from the **LSJC**#(Source) bank. It provides the template for all subsequent linking. All null status words indicate the absence of anything useful being present- space is preserved even for absent detector packages.

Its form is defined by `LsTix_BOS.CMN` and in `lstx.ddl`. There is one bank for each "track"; the list of banks terminates with the first missing index. There is one bank per hypothetical track, and they are created by *LsIdx_create_list*. An example of a dump from *LsTix_dump* appears in Appendix C.3.

```
header...........
     packageID- [I] Ls source package ID number (see Ls_general.PAR)
     name- [I*4=C*4] bank name ("TRKL", etc.)
     index- [I] BOS bank number
     entry- [I] nth entry in bank
     momentum- [F] track momentum (GeV/c)
     sector- [I] CLAS sector#
     reserved
     reserved
     reserved
     reserved
     reserved
     reserved
     reserved
     reserved
     reserved
body.............
     package- [I] Ls package ID number
     status- [I] status of following information (0=>void)
     X- [F] CLAS X location predicted (cm)
     Y- [F] CLAS Y location predicted
     Z- [F] CLAS Z location predicted
     S1- [F] sector S1 location predicted
     S2- [F] sector S2 location predicted
     S3- [F] sector S3 location predicted
     tX- [F] CLAS tX direction predicted
```

```
tY- [F] CLAS tY direction predicted
tZ- [F] CLAS tZ direction predicted
tS1- [F] CLAS tS1 direction predicted
tS2- [F] CLAS tS2 direction predicted
tS3- [F] CLAS tS3 direction predicted
path- [F] path (cm) along track back to target
```

A.3 The LSIX Master Tracking Bank

Conceptually, the Linking System Index list (**LSIX**) BOS bank is a dynamic, temporary bank and is intended to keep a list of the relationships between the hits from various packages. There is one bank for each "track"; the list of banks terminates with the first missing index. Its structure is defined by `LsIdx_BOS.CMN` and in `lsix.ddl`. The (unfilled) bank is created by *LsIdx_create_list* and updated during the matching and electron testing via *LsIdx_update* calls. Null (zero) status words indicate the complete absence of useful information. An example dump from *LsIdx_dump* appears in Appendix C.3.

```
header......
     ID- source track ID
     confidence- confidence
     mass- mass (GeV)
     Px-momentum (CLAS X component) (GeV/c) at target
     Py-momentum (CLAS Y component)
     Pz-momentum (CLAS Z component)
     sector- CLAS sector#
     reserved- reserved
     reserved- reserved
     reserved- reserved
     reserved- reserved
     reserved- reserved
 body........
     ID- [I] Ls package ID number
     BANK- [I*4<=>C*4] bank name ("TRKL", ..., "ECRB", etc.)
     index- [I] BOS bank number
     entry- [I] nth entry in bank
     status- [I] status information
     quality- [F] quality (T-test)
     displacement- [F] displacement from projected track
     e_test- estimated probability of being an electron
     path- [F] estimated path length back to target
     reserved- reserved
     reserved- reserved
     reserved- reserved
```

A.4 The Ls to Detector Package Banks

The **LSTR**, **LSPT**, **LSVT**, **LSDC**, **LSCC**, **LSSC**, **LSEC**, and **LSLC** BOS banks are generated by the matching routines (*LsTkc_*_match*) for each package. They give the link from the Ls track to none to many hits in each package, and there is one bank for each track.

```
header....
    name_source-         [I*4<=>C*4] source bank name ("TRKL", etc.)
    idx_source-          [I] BOS bank index (1, 2, ... )
    entry_source-        [I] nth sequential entry in BOS bank#index
    status_source-       [I] reserved for future use
    quality_source-      [F] reserved for future use
    reserved-            reserved for future use
    displacement_source- [F] reserved for future use
    eTEST_source-        [F] electron quality test (T-test) (if available)
    path_source-         [F] projected path length (cm)
    reserved-            reserved for future use
    reserved-            reserved for future use
    reserved-            reserved for future use
body....
    name_pkg-         [I*4<=>C*4] detector bank name ("ECRB", etc.)
    idx_pkg-          [I] BOS bank index (1, 2, ... )
    entry_pkg-        [I] nth sequential entry in BOS bank#inde
    status_pkg-       status of match
    quality_pkg-      [F] reserved for future use
    reserved-         reserved for future use
    displacement_pkg- [F] distance from projection to hit (cm)
    eTEST_pkg- electron quality test (T-test)
    path_pkg- [F] projected path length through detector (cm)
    reserved- reserved for future use
    reserved- reserved for future use
    reserved- reserved for future use
```

A.5 The Detector Package to Ls Bank

The Linking System package identifies the relationship between each detector package hit and from none to many Ls (hypothetical) tracks in the **LSTQ**, **LSPQ**, **LSVQ**, **LSDQ**, **LSCQ**, **LSSQ**, **LSEQ**, and **LSLQ** banks. There exist one bank for every hit in each package, and all are generated from a single call to *LsDrv_fetch*. Its structure is defined by `LsDrv_BOS.CMN`.

```
header=hit
     name- [I*4<=>C*4] hit bank name ("CCRC", etc.)
     index- [I] BOS bank index (0,1,2,...)
     entry- [I] nth sequential entry in BOS source#id_bank
     status- not used
     ID- Ls' hit ID#
     reserved- reserved for future use
body=track
     name- [I*4<=>C*4] bank name ("TRKL", etc.)
     index- [I] BOS bank index (0,1,2,...)
     entry- [I] nth sequential entry in BOS source#id_bank
     status- status from master index LSIX bank
     id- Ls' track ID#
     reserved- reserved for future use
```

B Ls Parameters

The Ls code is constructed on a very carefully chosen parameter space; it should be altered only with great care. This space provides the organization used throughout all routines within Ls.

The Ls package uses parameters to define the relevant characteristics of objects; not all objects use all parameters. These parameters are used by Ls internally. Other packages need only "know" about the values of the parameters to create the **LSJC** BOS banks. The routine *Ls_SeekID* is intended to allow other packages to determine the correct numbering without consulting the latest Ls documentaion. The routine *Ls_describe* dumps a table of the current parameter values to RECMES.

```
Ls  #   name              description
 |    0 null              => null, nothing, unused (a place holder)
 |    1 Time              => time (nS)
 |    2 unTime            => uncertainty in time (nS)
 |    3 Energy            => energy (usually GeV)
 |    4 unEnergy          => uncertainty in energy (usually GeV)
 |    5 Momentum          => momentum (GeV/c)
 |    6 unMomentum        => uncertainty in momentum (GeV/c)
 |    7 Mass              => estimated mass (GeV/c2)
 |    8 unMass            => uncertainty in mass (GeV/c2)
 |    9 Path              => estimated path back to target (cm)
 |   10 unPath            => uncertainty in path back to target (cm)
 |   11 r                 => CLAS spherical radial coordinate (cm)
 |   12 theta             => CLAS spherical theta coordinate (radians)
 |   13 phi               => CLAS spherical phi coordinate (radians)
 |   14 X                 => CLAS (lab frame||floor through sec#1) X (cm)
 |   15 Y                 => CLAS (lab frame straight up) Y (cm)
 |   16 Z                 => CLAS (lab frame||beam ) Z (cm)
 |   17 S1                => BN's sector S1 (||beam) (cm)
 |   18 S2                => BN's sector S2 (through center of sec.) (cm)
 |   19 S3                => BN's sector S3 (||Dc wires) (cm)
 |   20 I                 => any local IJK coord. I (cm)
 |   21 J                 => any local IJK coord. J (cm)
 |   22 K                 => any local IJK coord. K (cm)
```

```
 |   23 tX                => tangent to CLAS X
 |   24 tY                => tangent to CLAS Y
 |   25 tZ                => tangent to CLAS Z
 |   26 tS1               => tangent to BN's sector S1
 |   27 tS2               => tangent to BN's sector S2
 |   28 tS3               => tangent to BN's sector S3
 |   29 tI                => tangent to local I
 |   30 tJ                => tangent to local J
 |   31 tK                => tangent to local K
 |   32 status            => integer status word (zero=absent)
 |   33 charge            => electrial charge state
 |   34 sector            => CLAS sector
 |   35 ID                => internal identification#
 |   36 quality           => estimated quality (1.0=perfect)
 |   37 width             => width of something
 |   38 error             => error of something
 |   39 low-antiphiward   => low, antiphiward
 |   40 mean-middle       => mean, middle
 |   41 high-phiward      => high, phiward
 |   42 Void              => void- reserved space
 |   43 Source            => Source- generic input, source of all tracking
 |   44 Tr                => Tr- trigger package
 |   45 Pt                => Pt- photon tagger package
 |   46 Vt                => Vt- vertex detector package
 |   47 Dc                => Dc- drift chamber package
 |   48 Cc                => Cc- cerenkov counter package
 |   49 Sc                => Sc- time-of-flight package
 |   50 Ec                => Ec- forward calorimeter package
 |   51 Lc                => Lc-Ec1- large angle calorimeter package
 |   52 Ls                => Ls- linking system package
 |   53 Drain             => Drain- generic output
Ls  #   name              description
```

C Subroutines and Functions

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 RECSIS are the *Ls_** routines, with the exception of *LsTcl_initialize*.

C.1 Main Routines

These routines are used to calculate and fill the internal `COMMON`s used in the linking with pointers to and positions within BOS banks. Success or failure is returned by *OK* and an explanation for failure is placed in *err*.

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

This routine calls the *LsTcl_*_initialize* routines, which in turn register Tcl variables used for control from within RECSIS.

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

This routine calls *LsJcl_create_defaults*, *LsJcl_dump*, *LsJcl_initialize*, and *LsDrv_initialize*.

Subroutine Ls_link1st (OK, err)
Logical OK
Character(*)* err

This routine processes an event. First it calls *LsIdx_create_list* and generates the master tracking banks (**LSTX**) and master index banks (**LSIX**). It then calls the *LsTkc_*_match* routines to fill the master index and generate the track-to-hit banks, then calls *LsCls_eTest* for classification information. Then it calls *LsDrv_fetch* to generage all hit-to-track banks, then calls *LsRsl_fetch* to create the generic output bank.

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

This routine removes all the dynamic Ls BOS banks. This operation is normally done automatically by RECSIS.

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

This routine will remove **all** the static and dynamic Ls BOS banks including the **LSJC** Ls Job Control banks.

C.2 Initialization

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

If the appropriate **LSJC** Ls Job Control banks do **not** exist already, default versions of the bank are created via calls to the *LsJcl_*_default* routines.

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

Initializes the names for the detector-to-track banks.

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

Calls the *LsTkc_initialize* routine and creates the names for the Ls to detector banks.

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

Extracts and tests information from each **LSJC** job control bank and stores that information in a `COMMON` within `LsTkc_BOS.CMN`.

C.3 Bank Dumps and Examples

It is essential to examine events in detail to understand the performance of the package; these routines aid debugging and development.

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

Sends a listing of the contents of each **LSJC** Ls Job Control bank to RECMES.

The output lists the bank name, its range of indices, its coordinate system (either "CLAS", "SPHR", "SCTR", "SCTN", or "SPHS"), and for each key the (possibly truncated) Ls name for the variable, its location within the bank, is either "I" or "F" format, and the scaling factor to use. For the defaults in Ls 2.0.7:

```
LsJcl_dump I: Source -> "LSJC"#43 => "TRKL"#1-6 coordinate system="SCTR"
           I: "ID  " ->#1 => ID.null.Source                      I 1.000
           I: "Mome" ->#3 => Momentum.null.Source                F 1.000
           I: "char" ->#5 => charge.null.Source                  I 1.000
           I: "qual" ->#2 => quality.null.Source                 F 1.000
           I: "S2Vt" ->#12 => S2.Vt.Source                       F 1.000
           I: "S3Vt" ->#13 => S3.Vt.Source                       F 1.000
           I: "S1Vt" ->#14 => S1.Vt.Source                       F 1.000
           I: "tS2V" ->#15 => tS2.Vt.Source                      F 1.000
           I: "tS3V" ->#16 => tS3.Vt.Source                      F 1.000
           I: "tS1V" ->#17 => tS1.Vt.Source                      F 1.000
           I: "PatV" ->#18 => Path.Vt.Source                     F 1.000
           I: "S2Dc" ------ => S2.Dc.Source                      F 1.000
           I: "S3Dc" ------ => S3.Dc.Source                      F 1.000
           I: "S1Dc" ------ => S1.Dc.Source                      F 1.000
           I: "tS2D" ------ => tS2.Dc.Source                     F 1.000
           I: "tS3D" ------ => tS3.Dc.Source                     F 1.000
           I: "tS1D" ------ => tS1.Dc.Source                     F 1.000
           I: "PatD" ------ => Path.Dc.Source                    F 1.000
           I: "S2Cc" ->#19 => S2.Cc.Source                       F 1.000
           I: "S3Cc" ->#20 => S3.Cc.Source                       F 1.000
```

```
            I: "S1Cc" ->#21 => S1.Cc.Source                      F 1.000
            I: "tS2C" ->#22 => tS2.Cc.Source                     F 1.000
            I: "tS3C" ->#23 => tS3.Cc.Source                     F 1.000
            I: "tS1C" ->#24 => tS1.Cc.Source                     F 1.000
            I: "PatC" ->#25 => Path.Cc.Source                    F 1.000
            I: "S2Sc" ->#27 => S2.Sc.Source                      F 1.000
            I: "S3Sc" ->#28 => S3.Sc.Source                      F 1.000
            I: "S1Sc" ->#29 => S1.Sc.Source                      F 1.000
            I: "tS2S" ->#30 => tS2.Sc.Source                     F 1.000
            I: "tS3S" ->#31 => tS3.Sc.Source                     F 1.000
            I: "tS1S" ->#32 => tS1.Sc.Source                     F 1.000
            I: "PatS" ->#33 => Path.Sc.Source                    F 1.000
            I: "S2Ec" ->#34 => S2.Ec.Source                      F 1.000
            I: "S3Ec" ->#35 => S3.Ec.Source                      F 1.000
            I: "S1Ec" ->#36 => S1.Ec.Source                      F 1.000
            I: "tS2E" ->#37 => tS2.Ec.Source                     F 1.000
            I: "tS3E" ->#38 => tS3.Ec.Source                     F 1.000
            I: "tS1E" ->#39 => tS1.Ec.Source                     F 1.000
            I: "PatE" ->#40 => Path.Ec.Source                    F 1.000
            I: "S2Lc" ->#41 => S2.Lc.Source                      F 1.000
            I: "S3Lc" ->#42 => S3.Lc.Source                      F 1.000
            I: "S1Lc" ->#43 => S1.Lc.Source                      F 1.000
            I: "tS2L" ->#44 => tS2.Lc.Source                     F 1.000
            I: "tS3L" ->#45 => tS3.Lc.Source                     F 1.000
            I: "tS1L" ->#46 => tS1.Lc.Source                     F 1.000
            I: "PatL" ->#47 => Path.Lc.Source                    F 1.000
LsJcl_dump  I: Dc -> "LSJC"#47 => "TRKL"#1-6 coordinate system="SCTR"
            I: "ID  " ->#1 => ID.null.Dc                         I 1.000
            I: "Mome" ->#3 => Momentum.null.Dc                   F 1.000
            I: "char" ->#5 => charge.null.Dc                     I 1.000
            I: "qual" ->#2 => quality.null.Dc                    F 1.000
            I: "S2Vt" ->#12 => S2.Vt.Dc                          F 1.000
            I: "S3Vt" ->#13 => S3.Vt.Dc                          F 1.000
            I: "S1Vt" ->#14 => S1.Vt.Dc                          F 1.000
            I: "tS2V" ->#15 => tS2.Vt.Dc                         F 1.000
            I: "tS3V" ->#16 => tS3.Vt.Dc                         F 1.000
            I: "tS1V" ->#17 => tS1.Vt.Dc                         F 1.000
```

```
            I: "PatV" ->#18 => Path.Vt.Dc                          F 1.000
            I: "S2Dc" ------ => S2.Dc.Dc                           F 1.000
            I: "S3Dc" ------ => S3.Dc.Dc                           F 1.000
            I: "S1Dc" ------ => S1.Dc.Dc                           F 1.000
            I: "tS2D" ------ => tS2.Dc.Dc                          F 1.000
            I: "tS3D" ------ => tS3.Dc.Dc                          F 1.000
            I: "tS1D" ------ => tS1.Dc.Dc                          F 1.000
            I: "PatD" ------ => Path.Dc.Dc                         F 1.000
            I: "S2Cc" ->#19 => S2.Cc.Dc                            F 1.000
            I: "S3Cc" ->#20 => S3.Cc.Dc                            F 1.000
            I: "S1Cc" ->#21 => S1.Cc.Dc                            F 1.000
            I: "tS2C" ->#22 => tS2.Cc.Dc                           F 1.000
            I: "tS3C" ->#23 => tS3.Cc.Dc                           F 1.000
            I: "tS1C" ->#24 => tS1.Cc.Dc                           F 1.000
            I: "PatC" ->#25 => Path.Cc.Dc                          F 1.000
            I: "S2Sc" ->#27 => S2.Sc.Dc                            F 1.000
            I: "S3Sc" ->#28 => S3.Sc.Dc                            F 1.000
            I: "S1Sc" ->#29 => S1.Sc.Dc                            F 1.000
            I: "tS2S" ->#30 => tS2.Sc.Dc                           F 1.000
            I: "tS3S" ->#31 => tS3.Sc.Dc                           F 1.000
            I: "tS1S" ->#32 => tS1.Sc.Dc                           F 1.000
            I: "PatS" ->#33 => Path.Sc.Dc                          F 1.000
            I: "S2Ec" ->#34 => S2.Ec.Dc                            F 1.000
            I: "S3Ec" ->#35 => S3.Ec.Dc                            F 1.000
            I: "S1Ec" ->#36 => S1.Ec.Dc                            F 1.000
            I: "tS2E" ->#37 => tS2.Ec.Dc                           F 1.000
            I: "tS3E" ->#38 => tS3.Ec.Dc                           F 1.000
            I: "tS1E" ->#39 => tS1.Ec.Dc                           F 1.000
            I: "PatE" ->#40 => Path.Ec.Dc                          F 1.000
            I: "S2Lc" ->#41 => S2.Lc.Dc                            F 1.000
            I: "S3Lc" ->#42 => S3.Lc.Dc                            F 1.000
            I: "S1Lc" ->#43 => S1.Lc.Dc                            F 1.000
            I: "tS2L" ->#44 => tS2.Lc.Dc                           F 1.000
            I: "tS3L" ->#45 => tS3.Lc.Dc                           F 1.000
            I: "tS1L" ->#46 => tS1.Lc.Dc                           F 1.000
            I: "PatL" ->#47 => Path.Lc.Dc                          F 1.000
LsJcl_dump I: Cc -> "LSJC"#48 => "CCRC"#0-0 coordinate system="SPHS"
```

```
           I: "sect" ->#1 => sector.null.Cc                      I 1.000
           I: "thet" ->#7 => theta.null.Cc                       I 0.017
           I: "phi " ->#9 => phi.null.Cc                         I 0.262
           I: "Time" ->#6 => Time.null.Cc                        I 0.050
           I: "Ener" ->#5 => Energy.null.Cc                      I 0.010
LsJcl_dump I: Sc -> "LSJC"#49 => "SCRW"#1-1 coordinate system="CLAS"
           I: "Time" ->#2 => Time.null.Sc                        F 1.000
           I: "Ener" ->#1 => Energy.null.Sc                      F 1.000
           I: "X   " ->#9 => X.null.Sc                           F 1.000
           I: "Y   " ->#10 => Y.null.Sc                          F 1.000
           I: "Z   " ->#11 => Z.null.Sc                          F 1.000
           I: "IX  " ->#12 => I.X.Sc                             F 1.000
           I: "IY  " ->#13 => I.Y.Sc                             F 1.000
           I: "IZ  " ->#14 => I.Z.Sc                             F 1.000
           I: "JX  " ->#15 => J.X.Sc                             F 1.000
           I: "JY  " ->#16 => J.Y.Sc                             F 1.000
           I: "JZ  " ->#17 => J.Z.Sc                             F 1.000
           I: "KX  " ->#18 => K.X.Sc                             F 1.000
           I: "KY  " ->#19 => K.Y.Sc                             F 1.000
           I: "KZ  " ->#20 => K.Z.Sc                             F 1.000
LsJcl_dump W: Sc -> "LSJC"#49 contains no SECTOR information!
LsJcl_dump I: Ec -> "LSJC"#50 => "ECRB"#0-0 coordinate system="CLAS"
           I: "X   " ->#17 => X.null.Ec                          F 1.000
           I: "Y   " ->#18 => Y.null.Ec                          F 1.000
           I: "Z   " ->#19 => Z.null.Ec                          F 1.000
           I: "sect" ->#49 => sector.null.Ec                     I 1.000
           I: "errX" ->#23 => error.X.Ec                         F 1.000
           I: "errY" ->#24 => error.Y.Ec                         F 1.000
           I: "errZ" ->#25 => error.Z.Ec                         F 1.000
           I: "loEn" ->#1 => low-antiphiward.Energy.Ec           F 1.000
           I: "hiEn" ->#2 => high-phiward.Energy.Ec              F 1.000
           I: "loTi" ->#5 => low-antiphiward.Time.Ec             F 1.000
           I: "hiTi" ->#6 => high-phiward.Time.Ec                F 1.000
           I: "Iwid" ->#43 => I.width.Ec                         F 1.000
           I: "Jwid" ->#44 => J.width.Ec                         F 1.000
```

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

Sends a listing of the contents of each **LSTX** Ls master tracking bank to RECMES.

A typical event:

```
LsTix_dump I: ***begin**> "LSTX"# 1
LsTix_dump I: source ID: 47
           I: source: "TRKL"# 5 entry# 1
           I: MOMENTUM: 1.2659522
           I: sector: 5
           I: Source [ 43]: status= 0
           I: Tr [ 44]: status= 0
           I: Pt [ 45]: status= 0
           I: Vt [ 46]: status= 1F
           I: Vt X= 366.0253601
           I: Vt Y= -1366.0253906
           I: Vt Z= 999.9999390
           I: Vt S1= 1000.0000000
           I: Vt S2= 1000.0000000
           I: Vt S3= 1000.0000000
           I: Vt tX= 0.0000000
           I: Vt tY= 0.0000000
           I: Vt tZ= 0.0000000
           I: Vt tS1= 0.0000000
           I: Vt tS2= 0.0000000
           I: Vt tS3= 0.0000000
           I: Vt path= 0.0000000
           I: Dc [ 47]: status= 0
           I: Cc [ 48]: status= 1F
           I: Cc X= -53.9609451
           I: Cc Y= -94.8721619
           I: Cc Z= 394.4217834
           I: Cc S1= 394.4218140
           I: Cc S2= 109.1421814
```

```
I: Cc S3= 0.7045212
I: Cc tX= 0.9915203
I: Cc tY= 0.1297354
I: Cc tZ= -0.0074975
I: Cc tS1= -0.0713607
I: Cc tS2= -0.1086054
I: Cc tS3= 0.9915203
I: Cc path= 415.7066040
I: Sc [ 49]: status= 1F
I: Sc X= -60.5635834
I: Sc Y= -104.9204865
I: Sc Z= 486.1976318
I: Sc S1= 486.1976318
I: Sc S2= 121.1455994
I: Sc S3= 0.0106278
I: Sc tX= 0.9915293
I: Sc tY= 0.1296674
I: Sc tZ= -0.0074965
I: Sc tS1= -0.0713259
I: Sc tS2= -0.1085470
I: Sc tS3= 0.9915292
I: Sc path= 508.2666626
I: Ec [ 50]: status= 1F
I: Ec X= -61.9622650
I: Ec Y= -107.0491028
I: Ec Z= 505.6417542
I: Ec S1= 505.6417542
I: Ec S2= 123.6883774
I: Ec S3= -0.1363807
I: Ec tX= 0.9915294
I: Ec tY= 0.1296662
I: Ec tZ= -0.0074965
I: Ec tS1= -0.0713252
I: Ec tS2= -0.1085459
I: Ec tS3= 0.9915293
I: Ec path= 527.8768921
I: Lc [ 51]: status= 1F
```

```
           I: Lc X= 0.0000000
           I: Lc Y= 0.0000000
           I: Lc Z= 0.0000000
           I: Lc S1= 0.0000000
           I: Lc S2= 0.0000000
           I: Lc S3= 0.0000000
           I: Lc tX= 0.0000000
           I: Lc tY= 0.0000000
           I: Lc tZ= 0.0000000
           I: Lc tS1= 0.0000000
           I: Lc tS2= 0.0000000
           I: Lc tS3= 0.0000000
           I: Lc path= 0.0000000
           I: Ls [ 52]: status= 0
LsTix_dump I: ***end**> "LSTX"# 1
LsTix_dump I: ***begin**> "LSTX"# 2
LsTix_dump I: source ID: 47
           I: source: "TRKL"# 6 entry# 1
           I: MOMENTUM: 0.6865143
           I: sector: 6
           I: Source [ 43]: status= 0
           I: Tr [ 44]: status= 0
           I: Pt [ 45]: status= 0
           I: Vt [ 46]: status= 1F
           I: Vt X= 4.9811449
           I: Vt Y= -9.8258429
           I: Vt Z= 16.8488045
           I: Vt S1= 16.8488045
           I: Vt S2= 11.0000010
           I: Vt S3= -0.5991264
           I: Vt tX= 0.7902449
           I: Vt tY= 0.5925158
           I: Vt tZ= -0.1563268
           I: Vt tS1= 0.1608752
           I: Vt tS2= -0.5912970
           I: Vt tS3= 0.7902449
           I: Vt path= 18.5649109
```

```
I: Dc [ 47]: status= 0
I: Cc [ 48]: status= 1F
I: Cc X= 71.2676315
I: Cc Y= -250.8888245
I: Cc Z= 266.6989136
I: Cc S1= 266.6989136
I: Cc S2= 252.9098816
I: Cc S3= -63.7249603
I: Cc tX= 0.6045206
I: Cc tY= 0.7708812
I: Cc tZ= -0.2007416
I: Cc tS1= 0.2115934
I: Cc tS2= -0.7679734
I: Cc tS3= 0.6045205
I: Cc path= 374.0751953
I: Sc [ 49]: status= 1F
I: Sc X= 104.2055359
I: Sc Y= -370.0446472
I: Sc Z= 360.4353943
I: Sc S1= 360.4354248
I: Sc S2= 372.5708008
I: Sc S3= -94.7778549
I: Sc tX= 0.6041092
I: Sc tY= 0.7714210
I: Sc tZ= -0.1999047
I: Sc tS1= 0.2125884
I: Sc tS2= -0.7680224
I: Sc tS3= 0.6041092
I: Sc path= 529.2189331
I: Ec [ 50]: status= 1F
I: Ec X= 110.6356125
I: Ec Y= -393.2730713
I: Ec Z= 378.7063599
I: Ec S1= 378.7063599
I: Ec S2= 395.9022522
I: Ec S3= -100.8234634
I: Ec tX= 0.6041050
```

```
           I: Ec tY= 0.7714311
           I: Ec tZ= -0.1998782
           I: Ec tS1= 0.2126163
           I: Ec tS2= -0.7680179
           I: Ec tS3= 0.6041050
           I: Ec path= 559.4635010
           I: Lc [ 51]: status= 1F
           I: Lc X= 112.9402924
           I: Lc Y= -401.5976562
           I: Lc Z= 385.2543335
           I: Lc S1= 385.2543030
           I: Lc S2= 404.2638245
           I: Lc S3= -102.9899063
           I: Lc tX= 0.6041033
           I: Lc tY= 0.7714342
           I: Lc tZ= -0.1998713
           I: Lc tS1= 0.2126240
           I: Lc tS2= -0.7680171
           I: Lc tS3= 0.6041033
           I: Lc path= 570.3025513
           I: Ls [ 52]: status= 0
LsTix_dump I: ***end**> "LSTX"# 2
```

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

Sends a listing of the contents of each **LSIX** Ls master index bank to RECMES.

A typical event:

```
LsIdx_dump I: ***begin**> "LSIX"# 1
           I: source ID: 1
           I: confidence: 0.0000000
           I: mass: 0.0000000
           I: Px: 0.0000000
           I: Py: 0.0000000
           I: Pz: 0.0000000
           I: sector: 5
           I: Source [ 43]: status= -1
           I: .ID# 43
           I: .bank="TRKL"# 5 entry# 1
           I: .quality= 0.0000000
           I: .eTEST= 0.0000000
           I: .path= 0.0000000
           I: Tr [ 44]: status= 0
           I: Pt [ 45]: status= 0
           I: Vt [ 46]: status= 0
           I: Dc [ 47]: status= 1
           I: .ID# 47
           I: .bank="TRKL"# 5 entry# 1
           I: .quality= 1.0000000
           I: .eTEST= 1.0000000
           I: .path= 0.0000000
           I: Cc [ 48]: status= 1
           I: .ID# 48
           I: .bank="CCRC"# 0 entry# 1
           I: .quality= 0.5000000
           I: .eTEST= 0.9600000
           I: .path= 415.7066040
           I: Sc [ 49]: status= -1
```

```
               I: .ID# 49
               I: .bank="SCRW"# 1 entry# 5
               I: .quality= 0.9844940
               I: .eTEST= 0.5000000
               I: .path= 508.2666626
               I: Ec [ 50]: status= -1
               I: .ID# 50
               I: .bank="ECRB"# 0 entry# 1
               I: .quality= 0.7228715
               I: .eTEST= 0.9998742
               I: .path= 527.8768921
               I: Lc [ 51]: status= 0
               I: Ls [ 52]: status= 0
     LsIdx_dump I: ***end**> "LSIX"# 1
     LsIdx_dump I: ***begin**> "LSIX"# 2
               I: source ID: 2
               I: confidence: 0.0000000
               I: mass: 0.0000000
               I: Px: 0.5425144
               I: Py: 0.4067705
               I: Pz: -0.1073206
               I: sector: 6
               I: Source [ 43]: status= -1
               I: .ID# 43
               I: .bank="TRKL"# 6 entry# 1
               I: .quality= 0.0000000
               I: .eTEST= 0.0000000
               I: .path= 0.0000000
               I: Tr [ 44]: status= 0
               I: Pt [ 45]: status= 0
               I: Vt [ 46]: status= 0
               I: Dc [ 47]: status= 1
               I: .ID# 47
               I: .bank="TRKL"# 6 entry# 1
               I: .quality= 1.0000000
               I: .eTEST= -1.0000000
               I: .path= 0.0000000
```

```
            I: Cc [ 48]: status= 0
            I: Sc [ 49]: status= -1
            I: .ID# 49
            I: .bank="SCRW"# 1 entry# 6
            I: .quality= 0.6545454
            I: .eTEST= 0.5000000
            I: .path= 529.2189331
            I: Ec [ 50]: status= 0
            I: Lc [ 51]: status= 0
            I: Ls [ 52]: status= 0
LsIdx_dump I: ***end**> "LSIX"# 2
```

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

Sends a listing of all the Ls parameters, ids, names, and descriptions to RECMES. An example of its output is shown in Appendix B.

Integer Function Ls_SeekID (name,OK, mss)
Character(*)* name
Logical OK
Character(*)* mss

Given a description *name*, it tries to return the appropriate parameter ID number and a description to *mss*. If successful, *OK*= .TRUE., otherwise *OK*= .FALSE. and *mss* contains the error message. A list of common synonyms is also kept in this routine.

C.4 Ls General Utility Routines

Integer Function LsGus_c2i (word)
Character(*)* word

Given a one (4 byte) *word* of type `CHARACTER*4`, returns the equivalent integer word (same bits, different variable type).

Subroutine LsGus_copy (Old, IndexOld, New, IndexNew, OK, err)
Character(*)* Old, New,
Integer IndexOld,IndexNew
Logical OK
Character(*)* err

Copies BOS bank *Old#IndexOld* to *New#IndexNew*, but treats the new banks as having only one (large) entry.

Subroutine LsGus_cx2in (string,N,word)
Character(*)* string
Integer N, word(*)

Converts a *string* into an array of *word*s of length N.

Integer Function LsGus_extract (word, pattern)
Integer word, pattern

Given a bitmask *pattern*, apply it to *word* and shift the bytes down until the mask would apply to the least significant bit.

Integer Function LsGus_f2i (value)
Real value

Converts a *value* into the equivalent integer word (same bits, different variable type).

Subroutine LsGus_find_sector (Syst,d1,d2,d3,sector,OK, err)
Character(*)* Syst
Real d1,d2,d3
Integer sector
Logical OK
Character(*)* err

Given a coordinate system *Syst* (either "CLAS" or "SPHR") and a coordinate *d1,d2,d3* (either X-Y-Z or R-THETA-PHI), finds the CLAS *sector*.

Subroutine LsGus_history (story,addition)
Character(*)* story,addition

For complicated comments or error messages,; cleans up *story* and adds the *addition* to the end of the *story*.

*Character*4 Function* LsGus_i2c (word)
Integer word

Given a one (4 byte) *word* of type `INTEGER*4`, returns the equivalent `CHARACTER*4` word (same bits, different variable type).

Real Function LsGus_i2f (word)
Integer word

Given a one (4 byte) *word* of type `INTEGER*4`, returns the equivalent `REAL*4` word (same bits, different variable type).

Subroutine LsGus_in2cx (string,N,word)
Integer N, word(*)
Character(*)* string

Converts *N* `INTEGER*4` *word*s into the equivalent character *string*.

Subroutine LsGus_normalize (x,y,z,L)
Real x,y,z,L

Given a vector of length x,y,z, returns the length L and adjusts x,y,z to have unit length (1.0).

Integer Function LsGus_packI4 (low,high)
Integer low,high

Packs the lowest halfword of *low* and *high* into the high and low halves of a longword.

Subroutine LsGus_purge (bank,index,OK)
Character(*)* bank

Integer index
Logical OK

Checks for the size and existence of BOS bank *bank#index* and zeros the contents. This ensures that unused status words are zero.

Subroutine LsGus_sort_decreasing (N,value,threshold,ordered)
Integer N
Real value(*), threshold
Integer ordered(*)

Given a list of *N values*, returns the number *N* above *threshold* and a list *ordered* of the relative position of the entries in a decreasing order.

Subroutine LsGus_sort_increasing (Nvalues,value,threshold,ordered)
Integer N
Real value(*), threshold
Integer ordered(*)

Given a list of *N values*, returns the number *N* below *threshold* and a list *ordered* of the relative position of the entries in an increasing order.

Subroutine LsGus_system (Sysi,seci,d1,d2,d3,Sysf,secf,OK, err)
Character(*)* Sysi, Sysf
Integer seci,secf
Real d1,d2,d3
Logical OK
Character(*)* err

Given a location *d1,d2,d3* in coordinate system of type *Sysi* (either "CLAS", "SCTR", "SPHR", "SCRN", of "SPHS") and sector#*Sysi*, converts to *d1,d2,d3* in the *Sysf* coordinate system of sector#*Sysf*. The sector number is ignored for the "CLAS" and "SPHR" systems.

Subroutine LsGus_unpackI4 (input,low,high)
Integer input,low,high

The *input* 4 byte word has its most significant halfword copied into *high* and the least significant halfword copied into *low*. It is the inverse function of *LsGus_packI4*.

C.5 Ls Ntuple Routines

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

Opens a file and creates a row-wise disk Ntuple using the CERNLIB HBOOK package for packing information from the **LSIX** master index bank.

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

Fills the Ntuple created by *LsHis_create* with information from the **LSIX** master index bank.

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

Properly closes the Ntuple and file created by *LsHis_create*.

C.6 Ls Bank Access Routines

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

Reads the source, if any, and creates a filled **LSTX** and an empty **LSIX** bank for each potential track.

Subroutine LsIdx_extract (index, pkg, Info, OK, err)
Integer index, pkg, Info(*)
Logical OK
Character(*)* err

Copies the part relevant to package#*pkg* from bank **LSIX#index** to the *Info* array.

Subroutine LsIdx_update (index, pkg, Info, OK, err)
Integer index, pkg, Info(*)
Logical OK
Character(*)* err

Copies from the *Info* array into the part relevant to package#*pkg* of bank **LSIX#index**.

Subroutine LsTix_extract (index, pkg, Info, OK, err)
Integer index, pkg, Info(*)
Logical OK
Character(*)* err

Copies the part relevant to package#*pkg* from bank **LSTX#index** to the *Info* array.

C.7 LsJcl Default Routines

Ls expects that the **LSJC** banks exist to tell it what to do; if they don't, Ls can do nothing. If the banks have not been generated by other packages, they are generated by the *LsJcl_*_default* routines. Refer to Appendix C.3 for listings of the default values.

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

Calls the other *LsJcl_*_match* routines.

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

Creates **LSJC**#Cc with default values if none exists already.

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

Creates **LSJC**#Dc with default values if none exists already.

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

Creates **LSJC**#Ec with default values if none exists already.

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

Creates **LSJC**#Lc with default values if none exists already.

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

Creates **LSJC**#Sc with default values if none exists already.

C.8 Interface Routines

The Tcl variables are used in the *LsTkc_*_match* routines and the *LsCls_*_** functions.

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

Initializes Tcl variables with RECSIS that are used by *LsTkc_Cc_match* and *LsCls_Cc_AT*. Refer to `LsJcl_Cc_controls.CMN` for the list.

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

Initializes Tcl variables with RECSIS that are used by *LsTkc_Dc_match* and *LsCls_Dc_neg*. Refer to `LsJcl_Dc_controls.CMN` for the list.

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

Initializes Tcl variables with RECSIS that are used by *LsTkc_Ec_match* and *LsCls_Ec_EM*. Refer to `LsJcl_Ec_controls.CMN` for the list.

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

Initializes Tcl variables with RECSIS that are used by *LsTkc_Sc_match* and *LsCls_Sc_MI*. Refer to `LsJcl_Sc_controls.CMN` for the list.

C.9 Track Matching Routines

These routines are provided as templates; currently they reflect some of the physics of the packages, but could be improved with the detailed knowledge of the particulars of each detector.

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

Attempts to find geometric matches between the projected track intersection points in **LSTX**#trk and the hits reported in the banks specified by **LSJC**#Cc. The results are stored in **LSIX**#trk and **LSCC**#trk.

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

Attempts to find geometric matches between the projected track intersection points in **LSTX**#trk and the hits reported in the banks specified by **LSJC**#Dc. The results are stored in **LSIX**#trk and **LSDC**#trk. If the Dc package is the source, then one unique perfect match is always reported.

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

Attempts to find geometric matches between the projected track intersection points in **LSTX**#trk and the hits reported in the banks specified by **LSJC**#Ec. The results are stored in **LSIX**#trk and **LSEC**#trk.

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

Attempts to find geometric matches between the projected track intersection points in **LSTX**#trk and the hits reported in the banks specified by **LSJC**#Sc. The results are stored in **LSIX**#trk and **LSSC**#trk. Preference is given for displacements along a paddle over displacements perpendicular to one.

C.10 Track Classification Routines

These routines are just provided as templates and do not do sophisticated testing- it is intended that these routines can call individual detector package libraries at some future date.

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

This routine calls the *LsCls_*_** functions for each track.

Real Function LsCls_Cc_AT (index, OK, err)
Logical OK
Character(*)* err

For Ls track number *index* finds the matching hit in **LSIX**#*index* and returns the probability that the hit was **a**bove the Cerenkov **t**hreshold.

Real Function LsCls_Dc_neg (index, OK, err)
Logical OK
Character(*)* err

For Ls track number *index* finds the matching hit in **LSIX**#*index* and returns the probability that the hit (track) was **neg**ative; hence a response of +1.0 indicates the track behaved as a negative particle.

Real Function LsCls_Ec_EM (index, OK, err)
Logical OK
Character(*)* err

For Ls track number *index* finds the matching hit in **LSIX**#*index* and returns the probability that the hit was an **e**lectro**m**agnetic shower.

Real Function LsCls_Sc_MI (index, OK, err)
Logical OK
Character(*)* err

For Ls track number *index* finds the matching hit in **LSIX**#*index* and returns the probability that the hit was a **m**inimum **i**onizing event.

C.11 Output Routines

Ls' only output is via the BOS banks.

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

After **LSIX** banks and all the track-to-hit banks are generated, examines every possible hit and generates a bank for each package giving the hit-to-track map. These banks are **LSTQ**, **LSPQ**, **LSVQ**, **LSDQ**, **LSCQ**, **LSSQ**, **LSEQ**, and **LSLQ**. They are intended to allow other packages to know which hits were *not* assigned by Ls and which may be used for a subsequent neutral source call.

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

After **LSIX** banks are generated, fills a bank described by `LsRsl_BOS.CMN`. In Ls 2.0.7, this is the **TRKS** bank and there is one bank for each sector.