Sci-Fi Offline Software Manual

Eric Sharkey(SUNY)/Hyeonseo Park/Taku Ishida(KEK)

February 22, 2000

Abstract

K2K SciFi tracker offline software: analysis flow, KUIP-based commands and Dataformats, are explained in this draft. This will be for version tagged as **fdw-1999-12-28** in k2k-repository.

Contents

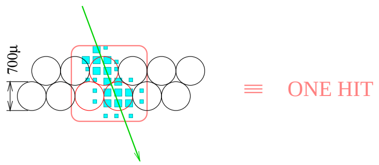

Figure 1: Definition of SciFi hit.

1 Analysis Flow

1.1 Hit Finding

- 'Module' name : libscifi(k2ksrc/scifi/libscifi)
- 'Library' name : libscifi
- Output Common block: sfdata.ch(fh/h)

Current definition of scifi hit is given in Fig. 1: A cluster of hit-fibers (Fiber Cluster) is difined as one hit.

Hit finding goes through following analysis stages:

$$CCDPixel \rightarrow PixelCluster \rightarrow HitFiber \rightarrow FiberCluster$$

which are connected by analysis routines:

- IIT Pixel Clustering
- Hit Fiber Finding
- Fiber Clustering

Also each analysis stage has self-looped filtering routines. Analysis flow is given in Fig. 2.

All SciFi analyses, including tracking, is perfomed by a call of a subroutine in libscifi/sfanal.c:

```
call sfanal
```

In KUIP-interfaced utility (FGDISP/FG^{++}) this can be involed by command **SFA**.

The analysis can be controlled by setting variables in a common libscifi/sfanal.ch:

```
 common/fgsfaf/fgdosfanl
&            ,fgiifiltermode,fgcsortmode,fgiicfiltermode
&            ,fgfhitmode,fgfhfiltermode
&            ,fgfcmode,fgfcfiltermode
&            ,fgtrackermode
```

Explanation of each mode is available in **SFSET** command description. Those values can be initialized to default varue by:

```
call fgsfanlinit
```

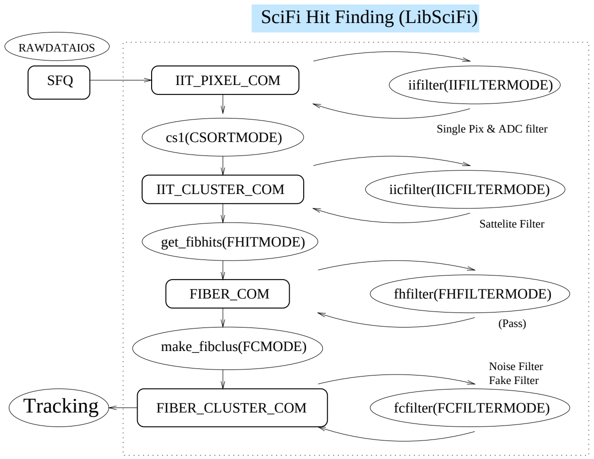

Figure 2: Fit Finding (LibSciFi) analysis flow chart

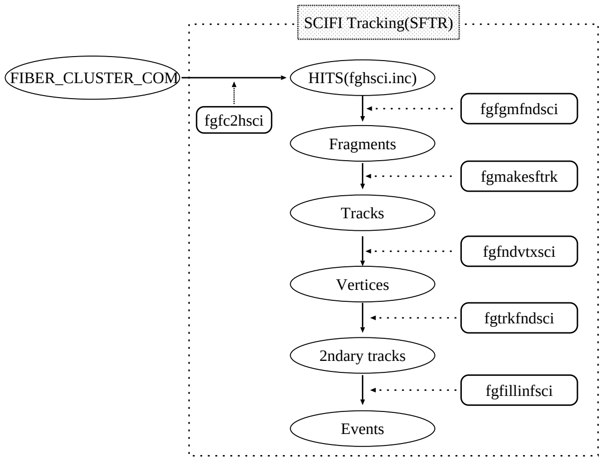

Figure 3: scifi tracking analysis flow chart

1.2 Tracking

- 'Module' name : scitr(k2ksrc/scifi/scitr) in the repository
- 'Library' name : libscitr
- Output Common blocks
  - fgscifrag.ch(fh/h) : fragment
  - fgscitrk.ch(fh/h) : track
  - fgscivrtx.ch(fh/h) : vertex
  - fgscievn.ch(fh/h) : collections of tracks and vertices which are all connected to each other

The analysis flow is shown in Figure 3. In KUIP-Interfaced utility tracker can be invoked separately by command **SFTR**. The tracking info can be cleaned up by command **CLRSFTR**.

1.3 Print Out

Contents of common can be printed by call of libscifi/sfprintf.c:

```
call sfprintf
```

In KUIP-interfaced utility this can be invoked by command **SFPRI**. This case you can specify file name to be printed out (default= on screen).

Variables to be printed can be chosen by setting variables in a common libscifi/sfpr.ch by command **SFPRM**:

```
     common/fgsfpr/fgdosfpr
    &          ,fgsfpranl,fgsfprova
    &          ,fgsfpriic,fgsfpriicpxl,fgsfpriicfcl
    &          ,fgsfprfh,fgsfprfhiicl,fgsfprfhpxl
    &          ,fgsfprfc,fgsfprfcfhl
    &          ,fgsftrprfgm,fgsftrprtrk
    &          ,fgsftrprvtx,fgsftrprevn
```

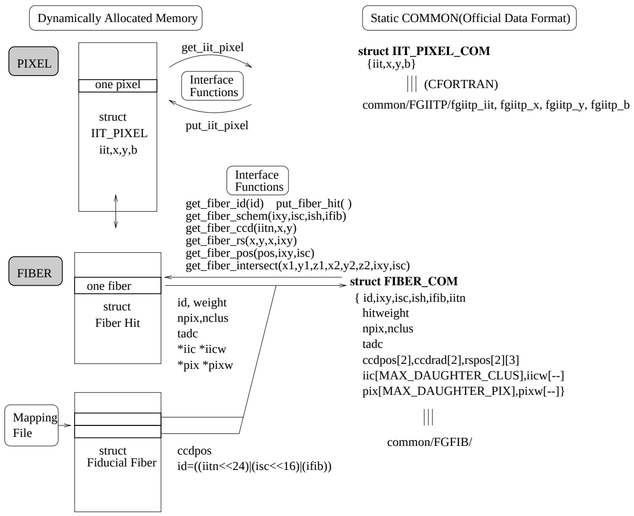

Figure 4: Schematics of LibSciFi interface function.

2 Common Structure and Interface Functions

2.1 Common Blocks

The main method to access the Sci-Fi detector data is through a series of interface functions used to fill Fortran common blocks. Each common block holds information about one stage of analysis.

Unlike traditional Fortran common blocks, not all data is available in these common blocks at the same time.

2.1.1 The SF Parameter Common Block

```
typedef struct SFPARAM_COM {
  int   num_iit,num_dim,num_lay,num_lpi,num_spl,num_ff;
  int   iit_threshold[3][MAX_IIT];
  float iit_gain[MAX_IIT];
} SFPARAM_COM;

       common/fgsfparam/fgsfparam_iit,fgsfparam_dim,fgsfparam_lay,
     &                 fgsfparam_lpi,fgsfparam_spl,fgsfparam_ff,
     &                 fgsfparam_iitthr,fgsfparam_iitg
```

The SF parameter common block contains the basic parameters which describe the geometry and configuration of the scifi detector.

num_iit is the number of image intensifier tubes, *num_dim* the number of different fiber projections (dimensions), *num_lay* the number of layers (or sections), *num_lpi* the number of layers attached to one IIT, *num_spl* the number of sheets in one projection of one layer. For the K2K SF detector these numbers are 24, 2, 20, 10, and 6. Software should not hard-code these numbers directly, but rather, reference this common block for future portability.

num_ff is the number of fiducial fibers in the EL mapping table, and this value may vary from run to run, if different tables are used.

iit_threshold contains three threshold parameters which describe how the image produced by that IIT should be treated while searching for hits. The first value contains an ADC threshold required for acceptance. Any pixel with an ADC value less than this threshold will be removed from the image. The second value contains the minimum number of pixels in a cluster, and the minimum hit weight for a fiber to be considered truly hit.

The final value, *iit_gain*, is a floating point number which can be used to scale the relative brightness of pixels recorded on one IIT to an equivalent value for another IIT.

2.1.2 The Fiber Common Block

```
typedef struct FIBER_COM {
  int   id,ixy,isc,ish,ifib,iitn;
  int   hitweight;
  int   npix;
  int   nclus;
  int   tadc;
  float ccdpos[2],ccdrad[2],rspos[2][3];
  int   iic[MAX_DAUGHTER_CLUS];
  float iicw[MAX_DAUGHTER_CLUS];
  int   pix[MAX_DAUGHTER_PIX];
  float pixw[MAX_DAUGHTER_PIX];
} FIBER_COM;

       common/fgfib/fgfib_id,fgfib_ixy,fgfib_isc,fgfib_ish,
     &            fgfib_ifib,fgfib_iitn,
     &            fgfib_hw,
     &            fgfib_hnp,fgfib_hnc,
     &            fgfib_hta,
     &            fgfib_ccdpos,fgfib_ccdrad,fgfib_rspos,
     &            fgfib_iic,fgfib_iicw,
     &            fgfib_pix,fgfib_pixw
```

The Fiber common block is the fundamental unit of information about a single fiber.

Every fiber has a unique "id" number. *id* is a 32-bit (4 byte) integer. The most significant byte is set to the IIT number to which the fiber is attached, the next byte is the sheet number, followed by the fiber number in the two least significant bytes. In C, this is denoted as $((iitn << 24)|(isc << 16)|(ifib))$.

The next four variables, *ixy*, *isc*, *ish*, and *ifib*, are the coordinates of the fiber within the detector structure. *ixy* is 1 if the fiber is an "X" fiber, and 2 if it is a "Y" fiber. *isc* is the "section" (or layer) number. A section consists of all fibers between adjacent water target layers. Sections are numbered from 1 to 20. *ish* is the sheet number. Each section is composed of 12 sheets, six in X and six in Y. Sheets are numbered from 1 to 6. *ifib* is the fiber number. Every fiber in a sheet is assigned a unique number from 1 to 1142. *iitn* is the number of the IIT to which this fiber is attached. This variable is redundant, since may be inferred from *ixy*, *ish*, and *isc*, but it is provided for convenience.

hitweight is the strength of the hit, in arbitrary units. *hitweight* is 0 if the fiber is not hit, and non-zero otherwise.

If the fiber is hit, *npix* will be set equal to the number of CCD pixels associated with the fiber, and *nclus* will contain the number of associated pixel clusters. **Not all hit finding algorithms associate fibers with pixels or clusters, so these fields may be zero, even if the fiber is hit.**

tadc is the "total adc" for the fiber hit. *tadc* may or may not be equal to hitweight.

ccdpos, *ccdrad*, and *rspos* are the coordinates of the fiber. The first element of *ccdpos* is the CCD X coordinate; the second is the CCD Y coordinate. *ccdrad* contains the X and Y radii of the ccd image of the fiber in units of pixels. *rspos* contains the coordinates of the fiber in real space. *rspos* is composed of two three-vectors. Each three-vector corresponds to an endpoint of the fiducial region of the fiber, specified in units of mm, in the detector hall coordinate system.

The remaining elements are the so called "daughter lists". When a fiber is marked as hit, this hit may be associated with pixels or clusters of pixels by the hit finding algorithm. If these associations are made, the ID numbers for the pixels/clusters are placed in the *pix*/*iic* lists. Each of these daughter elements also has associated with it a "weight". Each weight is a floating point number in the range of 0 to 1, with a 1 representing complete membership, and a fractional number representing partial membership. For example, a pixel which is on the boundary of a fiber region may be given a fractional weighting. A weight of 0 is nonsensical. The number of elements in each of these lists is recorded in *npix* and *nclus*.

2.1.3 The IIT Pixel Common Block

```
typedef struct IIT_PIXEL_COM {
  short iit,x,y,b;
} IIT_PIXEL_COM;

       common/fgiitp/fgiitp_iit,fgiitp_x,fgiitp_y,fgiitp_b
```

The IIT Pixel common block is probably the simplest common block which will be used by libscifi. It contains data for a single IIT pixel. Each element is a 16 bit number representing the iit number, x and y coordinates, and the ADC brightness for the pixel. Since there are many pixels needed for each event, short integers are used to decrease the data size.

2.1.4 The IIT Cluster Common Block

```
typedef struct IIT_CLUSTER_COM {
  int   clusterid, iit;
  int   npix, tadc;
  float x, y;
  int   pix[MAX_DAUGHTER_PIX];
  float pixw[MAX_DAUGHTER_PIX];
} IIT_CLUSTER_COM;

       common/fgiitc/fgiitc_cid,fgiitc_iit,
     &                  fgiitc_npix,fgiitc_tadc,
     &                  fgiitc_x,fgiitc_y,
     &                  fgiitc_pix,fgiitc_pixw
```

This common block stores the information about pixel clusters. The variable *clusterid* is a unique number assigned to each cluster. It has no explicit meaning or purpose other than to identify a unique cluster. Because clusters may be created, deleted, merged, or split at various stages of the analysis, it it not valid to assume that clusterids would be continuous.

The position of the cluster is given by the iit number, *iit*, and the coordinates, *x*, and *y*, which need not be integer pixel numbers. The number of pixels in the cluster is given by *npix*, and the total ADC of these pixels is given by *tadc*.

A daughter list of pixels which are members of the cluster is given by the variable *pix*. Each element of *pix* contains the pixel ID of a single pixel. Macros are available for working with pixel ID's.

2.1.5 The Fiber Cluster Common Block

```
typedef struct FIBER_CLUSTER_COM {
  int   numfibers, ixy;
  int   clusterid;
  int   weight;
  int   npix,tadc;
  int   npe;
  float pos, z;
  int   fhit[MAX_DAUGHTER_FIBERS];
  float fhitw[MAX_DAUGHTER_FIBERS];
} FIBER_CLUSTER_COM;

       common/fgfibc/fgfibc_nfib, fgfibc_ixy,
     &              fgfibc_cid, fgfibc_wgt,
     &              fgfibc_npix,fgfibc_tadc,
     &              fgfibc_npe,
     &              fgfibc_pos, fgfibc_z,
     &              fgfibc_fhit, fgfibc_fhitw
```

The Fiber Cluster common block describes a cluster of hit fibers. Each fiber cluster represents a single particle hit for a fiber plane. *numfibers* is the number of fibers in the cluster. *clusterid* is a unique identifier for the cluster and is assigned to be the same as the fiber id of one of its member fibers. (It is implicitly assumed that no two fiber clusters will share a single fiber.) *weight* is the strength of the hit, in arbitrary units. This may be total ADC or number of pixels or some other indicator. *npix* and *tadc* are optionally filled variables which contain the total number of pixels and ADC counts associated with the constituent fibers. *npe* is an estimate of the number of photoelectrons in the hit. [1] The position of the cluster is stored in *pos* and *z*. *pos* is the distance perpendicular to the beam axis (transverse coordinate); *z* is the position along the beam axis (longitudinal coordinate). Whether or not *pos* is an X or a Y coordinate can be determined from the variable *ixy*. The daughter list *fhit* contains a list of all fibers which are part of the fiber cluster, weighted by *fhitw*.

2.1.6 The Fiber Overall Common Block

```
typedef struct FIBER_OVERALL_COM {
  int   num_pixel_hits;
  int   max_pixel_hits;
  int   num_iit_clusters;
  int   num_fiber_hits;
  int   num_fiber_clusters;
  float energy;
  int   typeflag;
  int   num_pixel_hits_iit[MAX_IIT];
```

[1] This value is currently not used, but is available for future expansion.

```
  int   num_iit_clusters_iit[MAX_IIT];
  int   num_fiber_hits_iit[MAX_IIT];
  int   num_fiber_clusters_iit[MAX_IIT];
} FIBER_OVERALL_COM;

       common/fgfiboa/fgfiboa_nph,fgfiboa_mph,fgfiboa_nic,
     &                fgfiboa_nfh,fgfiboa_nfc,fgfiboa_e,fgfiboa_typ,
     &                fgfiboa_nphi,fgfiboa_nici,
     &                fgfiboa_nfhi,fgfiboa_nfci
```

The Fiber Overall common block contains some generic quantities about the event as a whole. It contains the number of elements of pixels, IIT clusters, fiber hits, and fiber clusters in *num_pixel_hits*, *num_iit_clusters*, *num_fiber_hits*, and *num_fiber_clusters*, respectively. Each of these counters has a corresponding array of counters suffixed with *_iit*. These arrays contain the number of elements of each type which are associated with each IIT. The energy and type flag variables are not currently used, but are available for future use.

max_pixel_hits is used internally by libscifi to improve speed and efficiency, and is the maximum number of pixels which can be stored in the memory currently allocated for pixels, which may be larger than the actual number of pixels in the event. This value should never be used for analysis.

2.1.7 Fortran Names

The common blocks described above use the C language naming conventions. For accessing these blocks from Fortran, it is necessary to use a completely different set of names.

The Fortran name for the common blocks described above are as follows:

```
C              Fortran

SFPARAM        fgsfparam
FIBER          fgfib
IIT_PIXEL      fgiitp
IIT_CLUSTER    fgiitc
FIBER_CLUSTER  fgfibc
IIC_FUNCS      fgiicfuncs
FCFUNCS        fgfcfuncs
FIBER_OVERALL  fgfiboa
FIBER_TRACK    fgfibt
```

The variables in each of these common blocks takes the form `<common block name>_<C name>`. For example, the variable *ixy* in the C common block FIBER becomes *fgfib_ixy* in Fortran.

Unfortunately, this tends to make some Fortran variable names very long, so, in some cases a shortened form of the C name is used. For example, the FIBER_OVERALL variable *num_pixel_hits* in C is referred to as *fgfiboa_nph* in Fortran.

A complete list of the correspondence between C and Fortran naming conventions is available in the file `sfdata.ch`. This file contains the C names at the top of the file, with the corresponding fortran names listed below.

2.1.8 SciFi Hit Common Block - fghsci.inc

```
c
        integer fgnhsci         ! Nhit of SCIFI detector
        integer fgnhscix        ! Nhit in xx-layer
        integer fgnhsciy        ! Nhit in yy-layer
        integer fghlsci         ! Nhits of each layer, 2dim:1,2 (x,y)
c
        real*4  fghsci          ! Space coordinates (cm unit)
        integer fgxysci         ! x,y (1,2)
        integer fgsctsci        ! section number (minus offset)
        integer fghsciiit       ! iit number
        integer fghscinpix      ! total number of pixels
        integer fghscitadc      ! total ADC sum
        integer fghpesci        ! Np.e.
c relation to sfdata.ch
        integer fghscifcid      ! Fiber Cluster ID (sfdata.ch)
        integer fghscifcw       ! FC weight
c...moved from fiber_hit_class.ch
        integer fiber_rs_tag    ! track correlated ?
        integer fsig_fib,fnos_fib,noise_fib
        integer fsig_list,fnos_list
c
        common/fghsci/fgnhsci
     &          ,fgnhscix,fgnhsciy
     &          ,fghlsci(FGNLYRSCI,2)
     &          ,fghsci(FGMAXHSCI,3)
     &          ,fgxysci(FGMAXHSCI)
     &          ,fgsctsci(FGMAXHSCI)
     &          ,fghsciiit(FGMAXHSCI)
     &          ,fghscinpix(FGMAXHSCI)
     &          ,fghscitadc(FGMAXHSCI)
     &          ,fghpesci(FGMAXHSCI)
     &          ,fghscifcid(FGMAXHSCI)
     &          ,fghscifcw(FGMAXHSCI)
c
     &          ,fiber_rs_tag(FGMAXHSCI)
     &          ,fsig_fib,fnos_fib,noise_fib
     &          ,fsig_list(FGMAXHSCI)
     &          ,fnos_list(FGMAXHSCI)
```

Although this common is desirable to be replaced with Libscifi Fiber Cluster Common Block, it still exists in many SciFi codes, mainly for **scitr / gdrsft**. As to compromize, variable to make connection to Fiber Cluster is given as fghscifcid.

2.1.9 The Fragment Common Block - fgscifrag.ch

```
/*-- FORTRAN section --*/
      integer fgnfrgmsci        ! number of fragments in scifi
      integer fgfrgmprojsci     ! projection (x=1,y=2) of fragment
      real    fgfrgmxyz0sci     ! start position of fragment(1 : x or y, 2 : z)
      real    fgfrgmxyz1sci     ! end position of fragment(1 : x or y, 2 : z)
      integer fgnkinkfrgmsci    ! number of kink in each fragment
```

```
      integer fgncrsfrgmsci     ! number of crossed fragments for each fragment
      integer fgnfhitsci        ! number of hits for each fragment
      integer fgffcidsci        ! hit(FiberCluster) ID for each fragment
      real    fgfhitwgtsci      ! hit weight for each fragment
c...temporary
      integer fgfhitidsci       ! hit(hsci.inc) ID for each fragment
c
      common / fgfragsci / fgnfrgmsci,fgfrgmprojsci(FGMXFGMSCI),
     +     fgfrgmxyz0sci(2,FGMXFGMSCI),fgfrgmxyz1sci(2,FGMXFGMSCI),
     +     fgnkinkfrgmsci(FGMXFGMSCI),fgncrsfrgmsci(FGMXFGMSCI),
     +     fgnfhitsci(FGMXFGMSCI),
     +     fgffcidsci(FGMXFHITSCI,FGMXFGMSCI),
     +     fgfhitwgtsci(FGMXFHITSCI,FGMXFGMSCI),
c...temporary !
     +     fgfhitidsci(FGMXFHITSCI,FGMXFGMSCI)

/*-- C section --*/
typedef struct
 { int   nfrgm; /* number of fragments in scifi */
   int   proj[FGMXFGMSCI]; /* projection (x=1,y=2) of fragment */
   float xyz0[FGMXFGMSCI][2]; /* start position of frgm.(1:x or y, 2:z) */
   float xyz1[FGMXFGMSCI][2]; /* end position of frgm.(1:x or y, 2:z) */
   int   nkink[FGMXFGMSCI]; /* number of kink in each fragment */
   int   ncrsfrgm[FGMXFGMSCI]; /* number of crossed frgm.s for each fragment */
   int   nfc[FGMXFGMSCI]; /* number of FCs for each fragment */
   int   fcid[FGMXFGMSCI][FGMXFHITSCI]; /* FC id for each fragment */
   float fcwgt[FGMXFGMSCI][FGMXFHITSCI]; /* hit weight for each fragment */
   /* temporary */
   int   fhitid[FGMXFGMSCI][FGMXFHITSCI]; /* tracker internal
                                            hit id for each fragment */
 } FGFRAGSCI_COM;
#define FGFRAGSCI COMMON_BLOCK(FGFRAGSCI,fgfragsci)
COMMON_BLOCK_DEF(FGFRAGSCI_COM,FGFRAGSCI);
```

2.1.10 The Track Common Block - fgscitrk.ch

```
/*-- FORTRAN section --*/
      integer fgntrksci         ! number of tracks in scifi
      integer fgtfgmidsci       ! fragment id for each scifi track
      integer fgflagtrksci      ! flags for each scifi track
      real    fggdtrksci        ! goodness of track fit
      real    fgzstrtrksci      ! z_0 for each track
      real    fgzstptrksci      ! z_1 for each track
      real    fgxy0trksci       ! xy_0 for each track
      real    fgslptrksci       ! slope for each track
      integer fgntvtxsci        ! number of vertices for each track
      integer fgtvtxidsci       ! vertex id for each track
c
      common / fgtracksci / fgntrksci,fgtfgmidsci(2,FGMXTRSCI),
     +     fgflagtrksci(FGMXTRSCI),fggdtrksci(FGMXTRSCI),
     +     fgzstrtrksci(FGMXTRSCI),fgzstptrksci(FGMXTRSCI),
     +     fgxy0trksci(2,FGMXTRSCI),fgslptrksci(2,FGMXTRSCI),
     +     fgntvtxsci(FGMXTRSCI),fgtvtxidsci(FGMXVXSCI,FGMXTRSCI)
```



```
/*-- C section --*/
typedef struct
 { int   ntrk; /* number of tracks in scifi */
   int   tfgmid[FGMXTRSCI][2]; /* fragment id for each scifi track */
   int   flagtrk[FGMXTRSCI]; /* flags for each scifi track */
   float gdtrk[FGMXTRSCI]; /* goodness of track fit */
   float zstrtrk[FGMXTRSCI]; /* z_0 for each track */
   float zstptrk[FGMXTRSCI]; /* z_1 for each track */
   float xy0trk[FGMXTRSCI][2]; /* xy_0 for each track */
   float slptrk[FGMXTRSCI][2]; /* slope for each track */
   int   ntvtx[FGMXTRSCI]; /* number of vertices for each track */
   int   tvtxid[FGMXTRSCI][FGMXVXSCI]; /* vertex id for each track */
 } FGTRACKSCI_COM;
#define FGTRACKSCI COMMON_BLOCK(FGTRACKSCI,fgtracksci)
COMMON_BLOCK_DEF(FGTRACKSCI_COM,FGTRACKSCI);
```

Flag of SciFi track (**fgflagtrksci**) is as follows:

	value	description
a.	0	good track
b.	fgfrgmflag(for Y)*100 + fgfrgmflag(for X)	(See Fig. 5)
c.	-iproj	single fragment track in iproj-projection
d.	-(iproj+10)	single fragment track (found in 2ndary track finding) in iproj-projection
e.	-10	tracks found in 2ndary track finding

Here,

a is good track(good correspondence between X and Y flagment) and found in the first track finding.

b is good track but one fragment(X or Y) may pass the bundle part or dead IIT region.

c is one fragment track(in fact, not allowed in present tracker)

d is one fragment track which is found in the 2ndary track finding using vertex information(mostly short tracks-2 or 3hit track)

e is good track but found in 2ndary track finding using vertex information(mostly short tracks-2 or 3 hit track)

and **fgfrgmflag**:

10 : if the fragment reaches the fiducial boundary(to the bundle side) at up-stream

20 : if the fragment reaches the fiducial boundary(to the bundle side) at down-stream

1 : the extrapolation of fragment is in the dead layer at up-stream

2 : the extrapolation of fragment is in the dead layer at down-stream

```
c*****Bundle hit******************************************************************
c*          TOP(X)          SIDE(Y)                TOP(X)          SIDE(Y)       *
c*                                                                               *
c*       ::::::::::                             ::::::::::                       *
c*       ::::::::::                             ::::::::::                       *
c*     |~~~~~~~~~~~~|   |~~~~~~~~~~~~|        |~~~~~~~~~~~~|   |~~~~~~~~~~~~|    *
c* Nu  |            |   |frgmflag=20 |        |            |   |frgmflag=10 |    *
c* --> |  -------   |   |            |   -->  |  -----     |   |      /     |    *
c*     |            |   |  \         |        |            |   |     /      |    *
c*     |____________|   |___\________|        |____________|   |____/_______|    *
c*                        ::::::::::                             ::::::::::      *
c*                        ::::::::::                             ::::::::::      *
c*                  +2000                                  +1000                 *
c*                                                                               *
c*       ::::::::::                             ::::::::::                       *
c*       ::::::::::                             ::::::::::                       *
c*     |~~~~/~~~~~~~|   |~~~~~~~~~~~~|        |~~~~\~~~~~~~|   |~~~~~~~~~~~~|    *
c*     |   /        |   |            |        |     \      |   |            |    *
c* --> |            |   |  -----     |   -->  |      \     |   | --------   |    *
c*     |frgmflag=20 |   |            |        | frgmflag=10|   |            |    *
c*     |____________|   |____________|        |____________|   |____________|    *
c*                        ::::::::::                             ::::::::::      *
c*                        ::::::::::                             ::::::::::      *
c*                  +20                                    +10                   *
c*                                                                               *
c*****Dead Layer******************************************************************
c*                                                                               *
c*       ::::::::::                             ::::::::::                       *
c*       ::::::::::      frgmflag=1             ::::::::::      frgmflag=2       *
c*     |~~~|~~~~~~~~|   |~~~|~~~~~~~~|        |~~~|~~~~~~~~|   |~~~|~~~~~~~~|    *
c*     |___|        |   |___|        |        |___|        |   |___|  \     |    *
c* --> |   ------   |   |   \        |   -->  |      ----  |   |       \    |    *
c*     |        |~~~|   |    \   |~~~|        |         |~~|   |        |~~~|    *
c*     |________|___|   |________|___|        |_________|__|   |________|___|    *
c*                        ::::::::::                             ::::::::::      *
c*                        ::::::::::                             ::::::::::      *
c*                 +100                                    +200                  *
c*                                                                               *
c*       ::::::::::                             ::::::::::                       *
c*       ::::::::::                             ::::::::::                       *
c*     |~~~|~~~~~~~~|   |~~~|~~~~~~~~|        |~~~|~~~~~~~~|   |~~~|~~~~~~~~|    *
c*     |___|\       |   |___|        |        |___|  \     |   |___|        |    *
c* --> |     \      |   |  -----     |   -->  |       \    |   |     -----  |    *
c*     |        |~~~|   |        |~~~|        |        \|~~|   |        |~~~|    *
c*     |________|___|   |________|___|        |_________|__|   |________|___|    *
c*      frgmflag=1        ::::::::::            frgmflag=2       ::::::::::      *
c*                        ::::::::::                             ::::::::::      *
c*                 +1                                       +2                   *
c*                                                                               *
c*********************************************************************************
```


Figure 5: Contribution to **fgflgtrksci** in case of **b**

2.1.11 The Vertex Common Block - fgscivrtx.ch

```
/*-- FORTRAN section --*/
      integer fgnvtxsci         ! number vertices in SCIFI
      real    fgvtxsci          ! vertex coordi. in SCIFI
      real    fgevtxsci         ! error of vertex coordi. in SCIFI
      integer fgnvtrksci        ! number of tracks for each vertex
      integer fgvtrkidsci       ! track id for each vertex
c
      common / fgvrtxsci / fgnvtxsci,
     +     fgvtxsci(3,FGMXVXSCI),fgevtxsci(3,FGMXVXSCI),
     +     fgnvtrksci(FGMXVXSCI),fgvtrkidsci(FGMXTRSCI,FGMXVXSCI)

/*-- C section --*/
typedef struct
 { int   nvtx;             /* number vertices in SCIFI */
   float vtx[FGMXVXSCI][3];   /* vertex coordi. in SCIFI */
   float evtx[FGMXVXSCI][3];  /* error of vertex coordi. in SCIFI */
   int   nvtrk[FGMXVXSCI];    /* number of tracks for each vertex */
   int   vtrkid[FGMXVXSCI][FGMXTRSCI]; /* track id for each vertex */
 } FGVRTXSCI_COM;
#define FGVRTXSCI COMMON_BLOCK(FGVRTXSCI,fgvrtxsci)
COMMON_BLOCK_DEF(FGVRTXSCI_COM,FGVRTXSCI);
```

2.1.12 The Event Common Block - fgscievn.ch

```
/*-- FORTRAN section --*/
      integer fgnevnsci         ! number of events in scifi
      integer fgnevtxsci        ! number of vertex in each scifi event
      integer fgnetrksci        ! number of tracks in each scifi event
      integer fgevtxidsci       ! vertex id of each scifi event
      integer fgetrkidsci       ! track id of each scifi event
c
      common / fgevntsci / fgnevnsci,fgnevtxsci(FGMXEVNSCI),
     +     fgnetrksci(FGMXEVNSCI),
     +     fgevtxidsci(FGMXVXSCI,FGMXEVNSCI),
     +     fgetrkidsci(FGMXTRSCI,FGMXEVNSCI)

/*-- C section --*/
typedef struct
 { int nevn;                 /* number of events in scifi */
   int nevtx[FGMXEVNSCI];      /* number of vertex in each scifi event */
   int netrk[FGMXEVNSCI];      /* number of tracks in each scifi event */
   int evtxid[FGMXEVNSCI][FGMXVXSCI];  /* vertex id of each scifi event */
   int etrkid[FGMXEVNSCI][FGMXTRSCI];  /* track id of each scifi event */
  } FGEVNTSCI_COM;
#define FGEVNTSCI COMMON_BLOCK(FGEVNTSCI,fgevntsci)
COMMON_BLOCK_DEF(FGEVNTSCI_COM,FGEVNTSCI);
```

2.2 SF Interface Functions

Many functions have been written to allow the user to easily copy data from the simple Fortran common blocks to and from the underlying data structures.

2.2.1 Pixel level functions

The following functions can be used to manipulate IIT pixel data by copying data into and out of the IIT_PIXEL common block:

```
int get_iit_pixel(int iit,int x,int y);
int get_iit_pixel_id(int pid);
int get_nth_iit_pixel(int n);
void put_iit_pixel(void);
void add_iit_pixel(void);
void append_iit_pixel(void);
void clear_iit_pixels(void);
int PIXID(IIT_PIXEL_COM p);
int PIXID3(iit,x,y);
int PIDX(int p);
int PIDY(int p);
int PIDI(int p);
```

`get_iit_pixel()` can be used to copy any given pixel from the pixel data structures into the IIT_PIXEL common block. The ADC brightness is returned to the caller. This function treats all pixels as if they exist. If it is called with an invalid pixel number (i.e. one with coordinates outside the boundary of the IIT window or for a non-existent IIT) then it treats this as an unhit pixel and returns 0 accordingly.

`get_iit_pixel_id()` is not really a function. It's a macro for get_iit_pixel() which allows you to retrieve a pixel based on its "id" number. Every pixel has a unique id number defined as $((iit << 24)|(y << 12)|x)$. Additional macros, `PIXID`, `PIXID3`, `PIDX`, `PIDY`, and `PIDI` can be used to translate from a pixel to an id number, or from an id number to an X coordinate, a Y coordinate, or an IIT number.

`get_nth_iit_pixel()` works much the same as `get_iit_pixel()` except that the argument is a single integer and only pixels which have been hit are returned. The argument is a simple counter. This function is designed to be put in simple for/while/do loops to loop over all hit pixels. It is important to note that this function numbers the first hit pixel as "1" not "0". This should be intuitive for Fortran users but counterintuitive for C hackers. If the argument passed to `get_nth_iit_pixel()` is larger than the total number of pixels, then -1 is returned and the contents of IIT_PIXEL remain unchanged.

`put_iit_pixel()` copies from the IIT_PIXEL common block into the main data structures. No arguments are required. The value of IIT_PIXEL.b will overwrite any previously stored value in the main data structures.

`add_iit_pixel()`, in contrast to `put_iit_pixel()`, does not simply overwrite the existing brightness value. The new value is set to the old value plus the value in IIT_PIXEL.b.

`clear_iit_pixels()` erases all pixel information from memory.

2.2.2 Pixel Cluster level functions

Pixel cluster information can be controlled using the IIT_CLUSTER common block and the following functions:


```
int get_iit_cluster_id(int id);
int get_iit_cluster_xy(int iit,int x,int y);
int get_nth_iit_cluster(int n);
void put_iit_cluster(void);
void add_pix_iit_cluster(int clusterid, float weight);
void merge_iit_clusters(int cid1, int cid2);
void clear_iit_clusters(void);
```

`get_iit_cluster_id()`, `get_nth_iit_cluster()`, `put_iit_cluster()`, and `clear_iit_clusters()` perform the same tasks for IIT pixel clusters that `get_iit_pixel_id()`, `get_nth_iit_pixel()`, `put_iit_pixel()`, and `clear_iit_pixels()` do for pixels.

`get_iit_cluster_xy()` returns fills the common block with the closest cluster to the point passed in as an argument. If there is no cluster near this point, -1 is returned. "Near" is currently defined as 100 pixels, but this value may be changed in the future.

`add_pix_iit_cluster()` does not use the IIT_CLUSTER common block at all. Instead, it searches for the cluster with cluster id number clusterid. If this cluster is not found, it is created. Then, the pixel currently residing in the IIT_PIXEL common block is added to this cluster. The position of the cluster center is adjusted automatically and this pixel is added to the pixel daughter list.

`merge_iit_clusters()` takes two cluster id's and merges the two clusters into one cluster. The resulting cluster will have a cluster id number which is equal to the lower of *cid1* or *cid2*. Pixel daughter lists are automatically merged, but no care is taken to ensure that the pixels from each of these clusters is unique. If the same pixel is listed in each cluster, it will appear twice in the resulting merged cluster.

2.2.3 Fiber level functions

The next class of functions deal with information pertaining to a single fiber. The information available includes the position of the fiber in the detector, the position on the CCD image, and whether or not that fiber has been marked as hit. Most of these routines use the FIBER common block.

```
int num_fidfib_sheet(int iit, int isc);
int num_fib_sheet(int iit, int isc);
int get_fiber_id(int id);
int get_fiber_schem(int ixy,int isc,int ish,int ifib);
int get_fiber_ccd(int iitn,float x,float y);
int get_fiber_rs(float x,float y,float z,int ixy);
int get_fiber_pos(float pos,int ixy,int isc);
int get_fiber_intersect(float x1,float y1,float z1,float x2,float y2,float z2,int ixy,int isc);
int get_next_fiber(void);
int get_nth_fiber_hit(int n);
void put_fiber_hit(void);
void clear_fiber_hits(void);
```

The functions `num_fidfib_sheet()` and `num_fib_sheet()` do not use any common blocks. They simply return the number of fiducial fibers or total number of fibers in the specified sheet.

The remaining routines function more or less as the routines previously described, differing only in their arguments.

`get_fiber_id()` takes a fiber id number and fills the common block with the information which describes that fiber. A fiber id number is defined as $((iit << 24)|(isc << 16)|(ifib))$, where *iit* is the IIT number, *isc* is the section number, and *ifib* is the fiber number. `get_fiber_id()` returns the hit weight of the fiber.

get_fiber_schem() is almost the same as get_fiber_id, except that it takes the fiber's schematic coordinates as arguments rather than its id number.

get_fiber_ccd() takes a CCD position and attempts to find the closest fiber. A fiber is always returned in the common block, even if the point specified is far from the fiber bundle. It is up to the user to determine if this information is useful or not.

get_fiber_rs() takes a point in the detector and a dimension (X or Y) and finds the corresponding fiber which most closely passes that point. Again, as with get_fiber_ccd(), a fiber is always returned, regardless of how far from the detector the input point is.

get_fiber_pos() is like get_fiber_rs(), except rather than specifying a single point (three coordinates in three dimensional space), the user may specify only two coordinates, one coordinate parallel to the beam axis, another perpendicular. The value of the flag *ixy* determines whether the transverse coordinate is interpreted as an X coordinate or a Y coordinate. The missing coordinate is assumed to be near the center of the detector.

get_fiber_intersect() takes two points, a dimension, and a section number as input. This routine fills the common block with the fiber in the specified section which passes closest to the line formed by the two points given as arguments. This routine can be useful for tracking.

get_next_fiber() can be used to easily loop over all fibers. It works by reading the current contents of the FIBER common block and returning the fiber with the lowest id number greater than the current fiber's id number. To use this function to loop over all fibers, the FIBER common block must be manually initialized with "fiber 0", a fictitious zeroth fiber, so that when this routine is called for the first time, the first fiber is returned.

get_nth_fiber_hit() works like get_nth_pixel_hit(), only with fibers rather than pixels.

put_fiber_hit() records the current contents of the common block in the main data structures. (Only the hit information is recorded. This function can not be used to change fiber geometry information.)

clear_fiber_hits() removes all fiber hit information from memory. Fiber geometry information is preserved and cannot be removed.

2.2.4 Fiber Cluster level functions

The fiber cluster functions use the FIBER_CLUSTER common block to manipulate information about clusters of hit fibers.

```
int get_fiber_cluster(int clusterid);
int get_nth_fiber_cluster(int n);
int get_fiber_cluster_pos(float pos, int ixy, int isc);
int get_nth_fiber_cluster_sheet(int n, int ixy, int isc, int ish);
void put_fiber_cluster(void);
void clear_fiber_clusters(void);
```

get_fiber_cluster() takes a cluster id number as an argument and fills the FIBER_CLUSTER common block with the corresponding cluster. If no cluster with the specified clusterid is found, -1, is returned, otherwise, the cluster weight is returned.

get_nth_fiber_cluster() takes an integer argument and fills the common block with the "nth" fiber cluster. If n is larger than the number of clusters, -1 is returned, otherwise, the cluster weight is returned.

get_fiber_cluster_pos() tries to find the fiber cluster closest to the position *pos* in the section *isc* in the dimension *ixy*. It is similar to get_fiber_pos().

get_nth_fiber_cluster_sheet() functions the same as get_nth_fiber_cluster() except that a single sheet is specified. With an argument of 1 it finds the first fiber cluster in that sheet. If the argument exceeds the number of clusters in that sheet, -1 is returned, otherwise the cluster weight is returned.

put_fiber_cluster() copies data from the common block into the main data structures. If a cluster with the same cluster id already exists, it is overwritten, otherwise, a new cluster is created.

clear_fiber_clusters() removes all fiber cluster information from memory.

2.2.5 Fiber Track level functions (unused)

The FIBER TRACK common block is not currently used by the scifi data analysis but is presented here for completeness. The following functions are available for manipulating track information:

```
int get_fiber_track(int trackid);
int get_nth_fiber_track(int n);
int get_nth_fiber_track_df(int n);
void add_cluster_fiber_track(int trackid);
void put_fiber_track(void);
void clear_fiber_tracks(void);
```

get_fiber_track() takes a single integer track id number and attempts to find the track with a matching id number and copy that track into the common block. If the track is successfully found, the return value is the typeflag for the track, otherwise, it is -1.

get_nth_fiber_track() finds the *n*th track and copies it into the common block. If there are fewer than *n* tracks in memory, -1 is returned, otherwise the track typeflag is returned.

add_cluster_fiber_track() is similar to add_pix_iit_cluster. It does not read the FIBER_TRACK common block, but rather the FIBER_CLUSTER block. The information about the cluster in that common block is added to the track with track id trackid. If there is no track with this track id, then a new track is created.

put_fiber_track() copies information from the FIBER_TRACK common block into the main track data structures.

clear_fiber_tracks() removes all track information from memory.

3 KUIP Commands

3.1 K2K/SciFi/GDRSFT

SFT display commands

3.1.1 OPDSFT [iopt]

`IOPT` I "Open(1)/Close(0)" `D=1`

Open another workstation (ID=2) for SFT display. Try help /graphics/workstation for controll.

3.1.2 DRSFTCNTL hmktyp hmksc hnump

`HMKTYP` I "Hit Marker Type" `D=20`
`HMKSC` R "Hit Marker Scale Size" `D=1.`
`HNUMP` I "Hit Number Printout on(1)/off(0)" `D=0`

SciFi Hit Display Control.

3.1.3 IIA

Display All IIT-pixel view in one HIGZ window.

3.1.4 IIT niit [xl xh yl yh]

`NIIT` I "IIT number" `D=1`
`XL` R "CCD region X low" `D=0.`
`XH` R "CCD region X high" `D=850.`
`YL` R "CCD region Y low" `D=0.`
`YH` R "CCD region Y high" `D=500.`

Display IIT-pixel view.

3.1.5 IIZ

Zoom Up Current CCD image by mouse clicks.

3.1.6 SFGETI

Get Infos(IITnum etc) for the point (on top/side) clicked by mouse.

3.1.7 IIBOX

Draw IIT region box on real space view.

3.1.8 FRONT

Draw FGD Front View (should go to base/fgdraw)

3.2 K2K/SciFi/LibSciFi

LibSciFi related commands

3.2.1 SFSET dosf iifl cstm iicfl fhit fhfl nfhcl fcfl trck

`DOSF` I "SciFi Analysis On/Off" `D=1`
`IIFL` I "Pixel Filter Mode" `D=3`
`CSTM` I "Pixel Clustering Mode" `D=1`
`IICFL` I "IIT Cluster Filter Mode" `D=0`
`FHIT` I "Fiber Hit Finding Mode" `D=5`
`FHFL` I "Hit Fiber Filtering Mode" `D=0`
`NFHCL` I "Hit Fiber Clustering Hit Separation" `D=2`
`FCFL` I "Fiber Cluster Filter" `D=3`
`TRCK` I "Tracking Mode" `D=1`

SciFi analysis flow mode control


```
DOSF : 0  No libscifi analysis (D)
       1  Perform libscifi analysis
IIFL : 0  No pixel filtering.
       1  Remove pixels from image that have less than
          n connected neighbors. n is specified for
          each IIT in database SFIIT.
       2  Remove pixels from image that have less than
          each ADC threshold of IIT, spiecified in
          database SFIIT.
       3  1 & 2 combined filtering is performed (D)



CSTM : 0  No pixel clustering
       1  csort   :  neighboring pixels=cluster (D)
       2  csort2  :  ADC peak search.
      (3  csort3  :  cluster serach with smoothing)
IICFL: 0  No IIT cluster filtering (D)
       1  IIT cluster filter 1
        Parent Npix > 100, distance from parent edge < 15pix
        Daughter Npix / Parent Npix < 0.1
       2  IIT cluster filter 2
        Parent > 40pix, Daughter > 10, parent-daughter < 20pix
        Parent within biggest 5th.



FHIT : 0  No fiberhit finding
       1  count ADCsum within fiber circle.
       2  assign pixelcluster to the nearest fiber(hw=ADCsum)
       3  count pixels within fiber circle.
       4  assign pixelcluster to nearest fiber     (hw=Npix)
       5  same as 3, but pixel loop within each IIT clusters.
          IIT cluster and fiber is in FIBER.iic array.
          FIBER.iicw=Npix from the cluster. (D)
FHFL : 0  no filter (D, for FCFL-2)
       1  hit fibers with hitweight less than threshold
          are omitted. Threshold values are set in database
          SFIIT.



NFHCL:<=0 No hit fiber clustering.
      > 0 Perform hit fiber clustering with maximum
          separation of fibers > NFHCL (D=2).
          example: oo***o*ooo  =  2 fiber clusters in
                   ooo**o**oo     default setting.
FCFL : 0  No filter
       1  Select heaviest Fiber Cluster in IIT cluster.
         (for FHIT=5).
       2  Remove fiber cluster with only upper/lower layer HFs
         (apply for all IIT)
       3  Both of 1 & 2(apply for high gain IIT only) (D)
       4  Both of 1 & 2(for all IIT)
```

```
TRCK : 0  No tracking.
       1  tracker-1: SFTR (HP SFT tracker) (D)
      (2) tracker-2: Beamtest tracking (Obsolete)
```

3.2.2 SFA

Perform SciFi analysis flow (Need to be controlled by command SFSET)

3.2.3 SFPRMODE dosfpr anl ova iic iicpxl iicfcl fh fhiicl fhpxl fc fcfhl frgm trck vrtx evnt

DOSFPR	I	"Printout ?"	D=1
ANL	I	"Analysis flag"	D=1
OVA	I	"Overall"	D=1
IIC	I	"IIT Cluster"	D=0
IICPXL	I	"IIC Pixlist"	D=0
IICFCL	I	"IIC FClist"	D=0
FH	I	"Fiber Hit"	D=0
FHIICL	I	"FH IICList"	D=0
FHPXL	I	"FH PixelList"	D=0
FC	I	"Fiber Cluster"	D=0
FCFHL	I	"FC FHList"	D=0
FRGM	I	"Fragment"	D=0
TRCK	I	"Track"	D=0
VRTX	I	"Vertex"	D=0
EVNT	I	"Event"	D=1

SciFi common information printout controll

3.2.4 SFPRINT [fnam]

FNAM	C	"Filename to be printed"	D=' '

LibSciFi analysis flags and overall printout

3.3 K2K/SciFi/ANASFT

SciFi tracker commands

3.3.1 SFTR

Perform SciFi Track and Vertex finding (part of SFA)

3.3.2 CLRSFTR

Clear the scifi track and vertex inf. from the memory

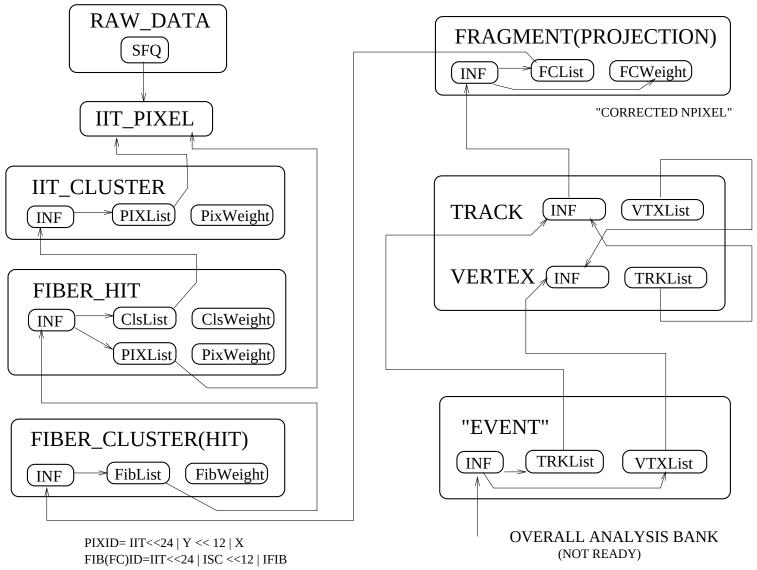

Figure 6: Bank Flow Chart

4 ZBS Bank Format

4.1 General

Flow chart of SciFi banks is given in Fig. 6

- Version number, number of segments, and length of each segment are kept in 0th segment (EMB)
- Analysis oriented banks have four digit number in its bank name to distingish verion.
- Relation between banks is kept in the first segment.

4.2 Raw Data Pixel Level Bank

4.2.1 SFQ

```
/*-------------------------------------------------------------*/
/*      Charge(Q) information from Scintillation Fiber tracker */
/* Bank Name    : SFQ */
/* Common Name  : ---(IIT_PIXEL_COM) */
/* Include file : ---libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfrawtqftn.c */
/*      comment      :    */
```

```
/*-------------------------------------------------------------*/

BANK NAME:'SFQ'

    EMB

        LEM +  1            integer :        length of the 0th segment
               2            integer :        Version
               3            integer :        Mode
               4            integer :        number of segments = 12
               5            integer :        Length of the  1st segment
               6            integer :        Length of the  2nd segment
            :

    *) mode -> number of segments == (12 or 24)

    1st segment ( for IIT 1-2 )

        IND +  1            integer :        header of FIFO/FADC
               2            integer :        X/Q1/Y/Q2
               3            integer :        X/Q1/Y/Q2
            :
        IND +  N            integer :        terminator of FIFO/FADC

    2nd segment ( for IIT 3-4 )

        IND +  1            integer :        header of FIFO/FADC
               2            integer :        X/Q1/Y/Q2
               3            integer :        X/Q1/Y/Q2
            :
        IND +  N            integer :        terminator of FIFO/FADC
```

4.3 IIT CLUSTER Level Banks

4.3.1 IITCLSTINF

```
/*-------------------------------------------------------------*/
/*      IIT cluster general information */
/* Bank Name    : IITCLSTINF */
/* Common Name  : /fgiitc/(F)  IIT_CLUSTER (C) */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sfiitclstred.wrt */
/*      comment      : */
/*-------------------------------------------------------------*/

BANK NAME: 'IITCLSTINFxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 1 (1dim, fixed-length)
    * currently still use Type-ZERO.
                          ----------
    EMB
```



```
LEM + 1 integer: length of EMB
    + 2 integer: bank version
    + 3 integer: release version  *)
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (Ncluster+1)
    + 7 integer: length of segments

    * note: release (analysis) version
 =(algorithm<<24)|(tag bit<<23)|((year%100)<<12)|(a=1,b=2,..)
* tag bit: to distinguish developement code.
98a tag bit = 1 is real 98a
98a tag bit = 0 is modified version of 98a

    segment 1 (for connection between banks)

IND + 1 integer: pixel cluster bank ID (xxxx)
    + 2 integer: pixlist ID (xxxx of IITCLSTPIXLST )
    + 3 integer: pixwgt  ID (exist=IITCLSTPIXWGT, none=0)

    segment N+1 (N <= ID of the cluster)

IND + 1 integer: clusterid
    + 2 integer: iit number
    + 3 integer: number of pixels ( of this cluster )
    + 4 integer: total adc ( of this cluster )
    + 5 float  : x (weighted mean, center?, float)
    + 6 float  : y (weighted mean, center?, float)

     *) Cluster ID: sequential digit number, non-consecutive.
        If a cluster is removed, higher numbered clusters are not changed.
                                                              ------------
```

4.3.2 IITCLSTPIXLST

```
/*-------------------------------------------------------------*/
/*      Pixel List of IIT cluster */
/* Bank Name    : IITCLSTPIXLST */
/* Common Name  : /fgiitc/(F)  IIT_CLUSTER (C) */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sgiitclsrred.wrt */
/*      comment      : */
/*-------------------------------------------------------------*/

BANK NAME: 'IITCLSTPIXLSTxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 0 (1dim, variable-length)

    EMB
```

```
LEM + 1 integer: length of EMB
    + 2 integer: bank version
    + 3 integer: release version
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (Ncluster+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :

    segment 1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: IITCLSTINF  ID

    segment N+1 (N == corresponding cluster number)

IND + 1 : ID of IIT
    + 2 : number of pixels of the cluster
    + 3 : pixel id  ( (iitn<<24) | (x<<12) | (y) )
    + 4 :      :
    + 5 :      :
```

4.3.3 IITCLSTPIXWGT

```
/*--------------------------------------------------------------*/
/*      Weight of pixels of IIT cluster */
/* Bank Name    : IITCLSTPIXWGT */
/* Common Name  : /fgiitc/(F)  IIT_CLUSTER (C) */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sfiitclstred.wrt */
/*      comment      : */
/*--------------------------------------------------------------*/

BANK NAME: 'IITCLSTPIXWGTxxxx'

    * note: XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version
    + 3 integer: release version
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (Ncluster+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :

    segment 1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: IITCLSTINF  ID

   segment N+1 (for Nth cluster)

IND + 1 integer: ID of IIT
    + 2 integer: number of pixels of the cluster
    + 3 integer: weight of 1st pixel
    + 4 integer:      :
    + 5 integer:      :
```


4.4 Fiber Hit Level Banks

4.4.1 FIBERCLSTINF

```
/*--------------------------------------------------------------*/
/*      Fiber hit general information */
/* Bank Name    : FIBERHITINF */
/* Common Name  : /fgfiber/(F)  FIBER           */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberhitred.wrt */
/*      comment      : */
/*--------------------------------------------------------------*/

BANK NAME: 'FIBERHITINFxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 1 (1dim, fixed-length)
    * currently still use Type-ZERO.
                          ----------
    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version # *)
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberH+1)
    + 7 integer: length of 1st segment
    + 8 : (will be just 7 after bank 1)

    segment #1 (for connection between banks)

IND + 1 integer: fiber hit bank ID (xxxx)
    + 2 integer: pixlst  ID (xxxx of PIXLST , none=0)
    + 3 integer: pixwgt  ID (        PIXWGT , none=0)
    + 4 integer: clstlst ID (xxxx of CLSTLST, none=0)
    + 5 integer: clstwgt ID (        CLSTWGT, none=0)

    segment #N+1 (N == number of the hit)

IND + 1 integer: fiber id number *)
            + 2         integer: weight of hit
```

```
    + 3 integer: number of pixels
            + 4         integer: number of clusters
    + 5 integer: total adc ( of this fiber )

    *) Fiber ID:   (iit<<24) | (isc<<12) | (ifib)
```

4.4.2 FIBERHITPIXLST

```
/*-----------------------------------------------------------------*/
/*      Pixel List of Fiber Hit */
/* Bank Name    : FIBERHITPIXLST */
/* Common Name  : /fgfiber/(F)  FIBER (C)       */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberhitred.wrt */
/*      comment      : */
/*-----------------------------------------------------------------*/

BANK NAME: 'FIBERHITPIXLSTxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 0 (1dim, variable-length)

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version #
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberH+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :


    segment #1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: FIBERHITINF ID

    segment #N+1 (for N-th fiber hit)

IND + 1 : ID of fiber
    + 2 : number of pixels of the hit
    + 3 : 1st pixel id  (iitn<<24) | (x<<12) | (y)
    + 4 :            :
    + 5 :            :
```

4.4.3 FIBERHITPIXWGT

```
/*-----------------------------------------------------------------*/
```



```
/*      Pixel weight of Fiber Hit */
/* Bank Name    : FIBERHITPIXWGT */
/* Common Name  : /fgfiber/(F)  FIBER (C)       */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberhitred.wrt */
/*      comment      : */
/*-----------------------------------------------------------------*/

BANK NAME: 'FIBERHITPIXWGTxxxx'

    * note: XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version #
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberH+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :

    segment #1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: FIBERHITINF ID

    segment #N+1 (for Nth fiber hit)

IND + 1 : ID of fiber
    + 2 : number of pixels of the hit
    + 3 : weight
    + 4 :      :
    + 5 :      :
```

4.4.4 FIBERHITCLSTLST

```
/*-----------------------------------------------------------------*/
/*      IIC List of Fiber Hit */
/* Bank Name    : FIBERHITCLSTLST */
/* Common Name  : /fgfiber/(F)  FIBER (C)       */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberhitred.wrt */
/*      comment      : */
/*-----------------------------------------------------------------*/

BANK NAME: 'FIBERHITCLSTLSTxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 0 (1dim, variable-length)
```

```
    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version #
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberH+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :

    segment #1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: FIBERHITINF ID
    + 3         integer: IITCLSTINF  ID

    segment #N+1 (for N-th fiber hit)

IND + 1 : ID of fiber
    + 2 : number of clusters of the hit
    + 3 : cluster ID
    + 4 :       :
    + 5 :       :
```

4.4.5 FIBERHITCLSTWGT

```
/*-----------------------------------------------------------------*/
/*      IIC weight of Fiber Hit */
/* Bank Name    : FIBERHITCLSTWGT */
/* Common Name  : /fgfiber/(F)  FIBER (C)       */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberhitred.wrt */
/*      comment      : */
/*-----------------------------------------------------------------*/

BANK NAME: 'FIBERHITCLSTWGTxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 0 (1dim, variable-length)

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version #
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberH+1)
```



```
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :

    segment #1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: FIBERHITINF ID

    segment #N+1 (for N-th fiber hit)

IND + 1 : ID of fiber
    + 2 : number of clusters of the hit
    + 3 : weight of the 1st cluster
    + 4 :       :
    + 5 :       :
```

4.5 Fiber Cluster Level Banks

4.5.1 FIBERCLSTINF

```
/*-----------------------------------------------------------------*/
/*      Hit Fiber cluster general information */
/* Bank Name    : FIBERCLSTINF */
/* Common Name  : /fgfibc/(F)  FIBER_CLUSTER   */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberclstred.wrt */
/*      comment      : */
/*-----------------------------------------------------------------*/

BANK NAME: 'FIBERCLSTINFxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 1 (1dim, fixed-length)
    * currently still use Type-ZERO.
                          ----------
    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version # *)
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberC+1)
    + 7 integer: length of 1st segment (=9)
    + 8 : (will be just 7 after bank 1)

    segment #1 (for connection between banks)

IND + 1 integer: fiber cluster bank ID (xxxx)
    + 2 integer: hitlist bank ID (xxxx of HITLST)
    + 3 integer: hitwgt bank ID  (xxxx of HITWGT)
```

```
    segment #N+1 (for Nth Fiber Cluster)

IND + 1 integer: numfibers
            + 2         integer: ixy
    + 3 integer: fiber cluster id number *)
            + 4         integer: weight of hit
            + 5         integer: number of pixels
            + 6         integer: total ADC value
    + 7 integer: number of pe's
    + 8 float  : transverse position (x or y)
    + 9 float  : z position

    *) The smallest constituent Fiber-ID number
        (iit<<24) | (isc<<12) | (ifib)
```

4.5.2 FIBERCLSTHITLST

```
/*----------------------------------------------------------------*/
/*      Hit Fiber List of  Fiber cluster */
/* Bank Name    : FIBERCLSTHITLST */
/* Common Name  : /fgfibc/(F)  FIBER_CLUSTER (C)   */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberclstred.wrt */
/*      comment      : */
/*----------------------------------------------------------------*/

BANK NAME: 'FIBERCLSTHITLSTxxxx'

    Bank Type: 0 (1dim, variable-length)

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version #
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberC+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :

    segment #1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: FIBERCLSTINF ID
            + 3 integer: FIBERHITINF  ID

    segment #N+1 (N == segment number of corresponding fiber cluster)

IND + 1 : ID of fiber cluster
```



```
    + 2 : number of fibers in the cluster
    + 3 : 1st fiber ID
    + 4 :       :
    + 5 :       :
```

4.5.3 FIBERCLSTHITWGT

```
/*----------------------------------------------------------------*/
/*      FH weight of Fiber cluster */
/* Bank Name    : FIBERCLSTHITWGT */
/* Common Name  : /fgfibc/(F)  FIBER_CLUSTER (C)    */
/* Include file : libscifi/sfdata.ch */
/*      I/O routines : libscifi/sfcookedio.c/sffiberclstred.wrt */
/*      comment      : */
/*----------------------------------------------------------------*/

BANK NAME: 'FIBERCLSTHITWGTxxxx'

    * note: XXXX : 4 digit for the identification

    Bank Type: 0 (1dim, variable-length)

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: bank version #
    + 3 integer: release version #
    + 4 (reserved)
    + 5 (reserved)
    + 6 integer: number of segments (NfiberC+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :

    segment #1   (for connection between banks)

IND + 1 integer: ID of the bank
    + 2 integer: FIBERCLSTINF  ID

    segment #N+1 (for N-th fiber cluster)

IND + 1 : ID of fiber cluster
    + 2 : number of fibers in the cluster
    + 3 : 1st fiber hitweight
    + 4 :       :
    + 5 :       :
```

4.6 Fragment (Track Projection) Level Banks

4.6.1 SCIFGMINF

```
/*------------------------------------------------------------------------*/
```

```
/*      SCIFI FRAGMENT general information */
/* FRAGMENT means the set of hits which lie on the line continuously. */
/*  */
/* Bank Name    : SCIFGMINF */
/* Common Name  : --            */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : FRAGMENT should not have the kink(if there were, */
/*   it will be two FRAGMENTS).    */
/*   But, two FRAGMENT should not be/ or can be crossed each other,*/
/*   depending on the analysis. */
/*   However, in order to avoid the confliction with Tumakov san's */
/*   code, it will be permitted 'for a moment'. */
/*   For that purpose, I need the number of kinks and the number   */
/*   of crossed fragments. */
/*-------------------------------------------------------------------*/

BANK NAME: 'SCIFGMINFxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nfrgm+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    +10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi fragment bank ID (xxxx)
    + 2 integer: scifi hit list ID (xxxx of SCIFGMHITLST)

    segment #N+1 (N == number of fragments)

IND + 1 integer: Fragment ID
    + 2 integer: x(1) or y(2) projection
            + 3         real   : xy0
            + 4         real   : z0
            + 5         real   : xy1
            + 6         real   : z1
    + 7 integer: number of kink (if no kink, 0)
    + 8 integer: number of crossed fragments
                                 (if no crossed fragment, 0)
```


4.6.2 SCIFGMHITLST

```
/*-------------------------------------------------------------*/
/*      HIT List of SCIFI FRAGMENT */
/* Bank Name    : SCIFGMHITLST */
/* Common Name  : --            */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank is the HIT list for fragment. */
/*-------------------------------------------------------------*/

BANK NAME: 'SCIFGMHITLSTxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nfrgm+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    +10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi hit list ID (xxxx)
    + 2 integer: scifi fragment bank ID (xxxx of SCIFGMINF)
    + 3 integer: scifi hit weight ID (xxxx of SCIFGMHITWGT)
    + 4 integer: Fiber Cluster bank ID (xxxx of Hit Bank)

    segment #N+1 (N == number of fragments)

IND + 1 integer: Fragment ID
    + 2 integer: number of Hits
    + 3 integer: 1st FC(HIT) id
            + 4         integer: 2nd FC(HIT) id
:
:
            + n+2        integer: n-th FC(HIT) id
```

4.6.3 SCIFGMHITWGT

```
/*-------------------------------------------------------------*/
/*      HIT Weight of SCIFI FRAGMENT */
/* Bank Name    : SCIFGMHITWGT */
/* Common Name  : --            */
/* Include file : -- */
```

```
/*      I/O routines : -- */
/*      comment      : This bank is the HIT weight for Fragment.*/
/*---------------------------------------------------------------*/

BANK NAME: 'SCIFGMHITWGTxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nfrgm+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    +10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi hit weight ID (xxxx)
    + 2 integer: scifi hit list ID (xxxx of SCIFGMHITLST)

    segment #N+1 (N == number of fragments)

IND + 1 integer: Fragment ID
    + 2 integer: number of hits
    + 3 real   : 1st HIT weight
            + 4         real   : 2nd HIT weight
:
:
            + n+2       real   : n-th HIT weight
```

4.7 Vertex-Track Level Banks

4.7.1 SCITRKINF

```
/*---------------------------------------------------------------------*/
/*      SCIFI TRACK general information */
/*  TRACK means the fragment(for fragment, See scifi.SCIFGMINF.txt). */
/*  TRACK will mostly consist of the combination of X-fragment and      */
/*   Y-fragment. But, sometimes, it can be only X or Y-fragment. */
/* */
/* Bank Name    : SCITRKINF */
/* Common Name  : --            */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank contains the list of fragments and     */
/*                  the fitting parameters ... */
```



```
/*---------------------------------------------------------------------*/

BANK NAME: 'SCITRKINFxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Ntracks+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    +10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi track bank ID (xxxx)
    + 2 integer: scifi fragment list ID (xxxx of SCIFGMINF)
    + 3 integer: scifi vertex list ID (xxxx of SCITRKVTXLST)

    segment #N+1 (N == number of tracks)

IND + 1 integer: Track ID
    + 2 integer: x fragment id or
                      if no related fragment, it will be '0'.
    + 3 integer: y fragment id or
                      if no related fragment, it will be '0'.
    + 4 integer: flag for each track
    + 5 real   : Goodness of fit
            + 6         real   : z0
            + 7         real   : z-stop
            + 8         real   : x0
            + 9         real   : y0
            +10         real   : dx/dz(z0)
            +11         real   : dy/dz(z0)

/*FLAG  0 : both side are at vertex + 10 if in the middle of track */
/* 1 : up-stream only at vertex */
/* 2 : down-stream only at vertex */
/* 3 : no related vertex */
```

4.7.2 SCITRKVTXLST

```
/*---------------------------------------------------------------*/
/*      Vertex List of SCIFI TRACK */
/* Bank Name    : SCITRKVTXLST */
```

```
/* Common Name  : --             */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank contains the list of vertex for*/
/*                each TRACK. */
/*------------------------------------------------------------*/

BANK NAME: 'SCITRKVTXLSTxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Ntracks+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    +10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi vertex list ID (xxxx)
    + 2 integer: scifi track bank ID (xxxx of SCITRKINF)
    + 3 integer: scifi vertex bank ID (xxxx of SCIVTXINF)

    segment #N+1 (N == number of the track)

IND + 1 integer: Track ID
    + 2 integer: number of vertices
    + 3 integer: 1st vertex id
    + 4 integer: 2nd vertex id
:
:
    + n+2 integer: n-th vertex id
```

4.7.3 SCIVTXINF

```
/*------------------------------------------------------------*/
/*      SCIFI VERTEX general information */
/* VERTEX means the intersection of two or more tracks and/or  */
/*     the end point of each track depending on the analysis. */
/* */
/* Bank Name    : SCIVTXINF */
/* Common Name  : --             */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank contains the information of   */
```



```
/*                VERTEX. */
/*------------------------------------------------------------*/

BANK NAME: 'SCIVTXINFxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nvertices+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    +10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi vertex bank ID (xxxx)
    + 2 integer: scifi track list ID (xxxx of SCIVTXTRKLST)

    segment #N+1 (N == number of the vertices)

IND + 1 integer: Vertex ID
    + 2 real   : x position of vertex
    + 3 real   : y position of vertex
    + 4 real   : z position of vertex
            + 5         real   : x error of vertex
    + 6 real   : y error of vertex
    + 7 real   : z error of vertex
```

4.7.4 SCIVTXTRKLST

```
/*------------------------------------------------------------*/
/*      Track List of SCIFI VERTEX */
/* Bank Name    : SCIVTXTRKLST */
/* Common Name  : --             */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank contains the list of track for */
/*                each VERTEX. */
/*------------------------------------------------------------*/

BANK NAME: 'SCIVTXTRKLSTxxxx'

note:
XXXX : 4 digit for the identification
```

```
    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nvertices+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    +10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi track list ID (xxxx)
    + 2 integer: scifi vertex bank ID (xxxx of SCIVTXINF)
    + 3 integer: scifi track bank ID (xxxx of SCITRKINF)

    segment #N+1 (N == number of the vertex)

IND + 1 integer: Vertex ID
    + 2 integer: number of tracks
    + 3 integer: 1st track id
    + 4 integer: 2nd track id
:
:
    + n+2 integer: n-th track id
```

4.8 Event Level Banks

4.8.1 SCIEVNINF

```
/*----------------------------------------------------------------------*/
/*      SCIFI EVENT general information */
/*  EVENT means the GROUP of vertices and tracks which are connected.  */
/* */
/* Bank Name    : SCIEVNINF */
/* Common Name  : --            */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank contains the number of tracks  */
/*                and vertices. */
/*----------------------------------------------------------------------*/

BANK NAME: 'SCIEVNINFxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
```



```
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nevents+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    + 10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi Event bank ID (xxxx)
    + 2 integer: scifi Vertex list ID (xxxx of SCIEVNVTXLST)
    + 3 integer: scifi Track list ID (xxxx of SCIEVNTRKLST)

    segment #N+1 (N == number of the events)

IND + 1 integer: Event ID
    + 2 integer: number of vertics
    + 3 integer: number of tracks
```

4.8.2 SCIEVNTRKLST

```
/*----------------------------------------------------------------------*/
/*      Track list of SCIFI EVENT  */
/* */
/* Bank Name    : SCIEVNTRKLST */
/* Common Name  : --            */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank contains the list of track for each */
/*                EVENT. */
/*----------------------------------------------------------------------*/

BANK NAME: 'SCIEVNTRKLSTxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nevents+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    + 10 :
```

```
    segment #1 (for connection between banks)

IND + 1 integer: scifi Track list ID (xxxx)
    + 2 integer: scifi Eevnt bank ID (xxxx of SCIEVNINF)
    + 3 integer: scifi Track bank ID (xxxx of SCITRKINF)

    segment #N+1 (N == number of the events)

IND + 1 integer: Event ID
    + 2 integer: number of Tracks
    + 3 integer: 1st track ID
    + 4 integer: 2nd track ID
:
:
    + n+2 integer: n-th track ID
```

4.8.3 SCIEVNVTXLST

```
/*----------------------------------------------------------------------*/
/*      Vertex list of SCIFI EVENT  */
/* */
/* Bank Name    : SCIEVNVTXLST */
/* Common Name  : --            */
/* Include file : -- */
/*      I/O routines : -- */
/*      comment      : This bank contains the list of vertex for each */
/*                EVENT. */
/*----------------------------------------------------------------------*/

BANK NAME: 'SCIEVNVTXLSTxxxx'

note:
XXXX : 4 digit for the identification

    EMB

LEM + 1 integer: length of EMB
    + 2 integer: version of bank
    + 3 integer: version of analysis
    + 4 integer: reserved
    + 5 integer: reserved
    + 6 integer: number of segments (Nevents+1)
    + 7 integer: length of 1st segment
    + 8 integer: length of 2nd segment
    + 9 :
    + 10 :

    segment #1 (for connection between banks)

IND + 1 integer: scifi vertex list ID (xxxx)
    + 2 integer: scifi Event bank ID (xxxx of SCIEVNINF)
    + 3 integer: scifi vertex bank ID (xxxx of SCIVTXINF)
```



```
    segment #N+1 (N == number of the events)

IND + 1 integer: Event ID
    + 2  integer: number of vertices
    + 3 integer: 1st vertex ID
                             if no vertex, it will be '0'.
    + 4 integer: 2nd vertex ID
:
:
    + n+2 integer: n-th vertex ID
```