DRAFT version 2.1
S. Margetis, M.A. Lisa, and D. Weerasundara
Comments by several people
December 1997

STAR DST objects

1 Introduction

This is the beginning of an attempt to specify the information saved on official STAR[1] DSTs. A DST is defined to be a collection of event information, for a collection of events, with no (physics) selection on events included. The event information represents our best knowledge about the individual interaction and it should contain all the necessary information needed for further physics analysis which will lead to the extraction of physics signals. This **may** include information that allows partial reconstruction of the event, either through explicit data saved (e.g. saving hits as well as tracks), or through implicit pointers via, e.g. object oriented databases. This draft intends to be the starting point of a collaboration-wide effort in defining these needs.
Specifically, the aims of this document are:

- To **identify** all the relevant *objects*, i.e. STAF tables or structures (e.g. 'tracks' and 'vertices'), and to **specify** the contents of these objects.
- To stimulate discussion on the costs and benefits of storing objects (e.g. hits) that could be used to partially **redo** the event reconstruction, without returning to the raw data. To this end, numerical examples are given for different scenarios.
- To collect relevant experience of collaborators from similar experiments– NA49 and E895 in particular.

Design of the STAR DST should reflect an awareness of the purpose/nature of the DST.

- STAR DSTs will contain many millions of events over the years. This, in turn, requires objects which contain only the necessary information needed for the subsequent analyses. Compressing information wherever possible is desirable.
- Since this effort is primarily targeting DST production with very large number of events, it is not addressing the needs of analysis sub-groups which might want to save small DSTs with large number of specific, non-standard objects (i.e. the dE/dx calibration, alignment or distortion software effort).

Before we proceed further we have to define and understand the various systems in which quantities are calculated. There are two categories:

- **ICS** – the detector Internal Coordinate System. This is the reference frame of the individual detector subsystem (e.g. implying a geometrical center of the TPC or SVT etc.). Local alignment software will provide information about where in this frame individual parts of the detector are located. The ICS defines a system for measuring spatial information, as well as momentum information.

- **SCS** – the overall STAR reference frame. Here is where detector-specific information from different detectors is combined. Global alignment information will be needed before the transition from local to STAR coordinates is possible. Again, the SCS defines a system for measuring spatial information, as well as momentum information. In momentum, this system should be the C.M. frame for *that* collision, which may include a sidewards boost due to finite beam crossing angle, and boost along the beam direction for asymmetric beams. Connections must be made between detector-specific ICS and the global SCS.

DST objects as well as public detector-specific objects (tables) are dealing with the latter category only. At this point we propose the use of non detector-specific objects in DSTs.

Figure 1 shows a simplified model (flow chart) of the STAR off-line analysis (for an extensive treatment of STAR off-line analysis see ref [2]). In this, 'raw' data coming from either a simulation or the experiment are fed into the event reconstruction chain and the results are summarized (per event) into a collection of DST objects. Intermediate ('mini' or 'fat') DSTs are necessary, as we mentioned already, for specific tasks which will eventually interact and update the STAR 'calibration' Database, but they are beyond the scope of this write-up.

A proposed collection of such DST objects is shown in the bottom part of Fig. 1. We would like to stress here that it is not necessary for all objects to be saved for all events and/or kinds of triggers. The output should be customized to the specific needs of the intended analyses on the produced DSTs. We should also note that one event is a collection of any combination of these objects with the exception of the trigger-object 'Scalers' which is read asynchronously by the Data Acquisition System (see below).

In order to specify the contents of the proposed objects other experiments (NA49 and E895/EOS) served as examples.

2 The MOCK DATA CHALLENGE

In the following special markings and comments are going to modify the proposed objects for the specific needs of the 'Mock Data Challenge' coming up next summer. Discussion with the physics analysis group leaders will update and better define the final tables. A minus sign (-) in front of a variable means that the variable can be omitted for the MDC. A plus sign (+) means the opposite and the letter m means it might need modification.

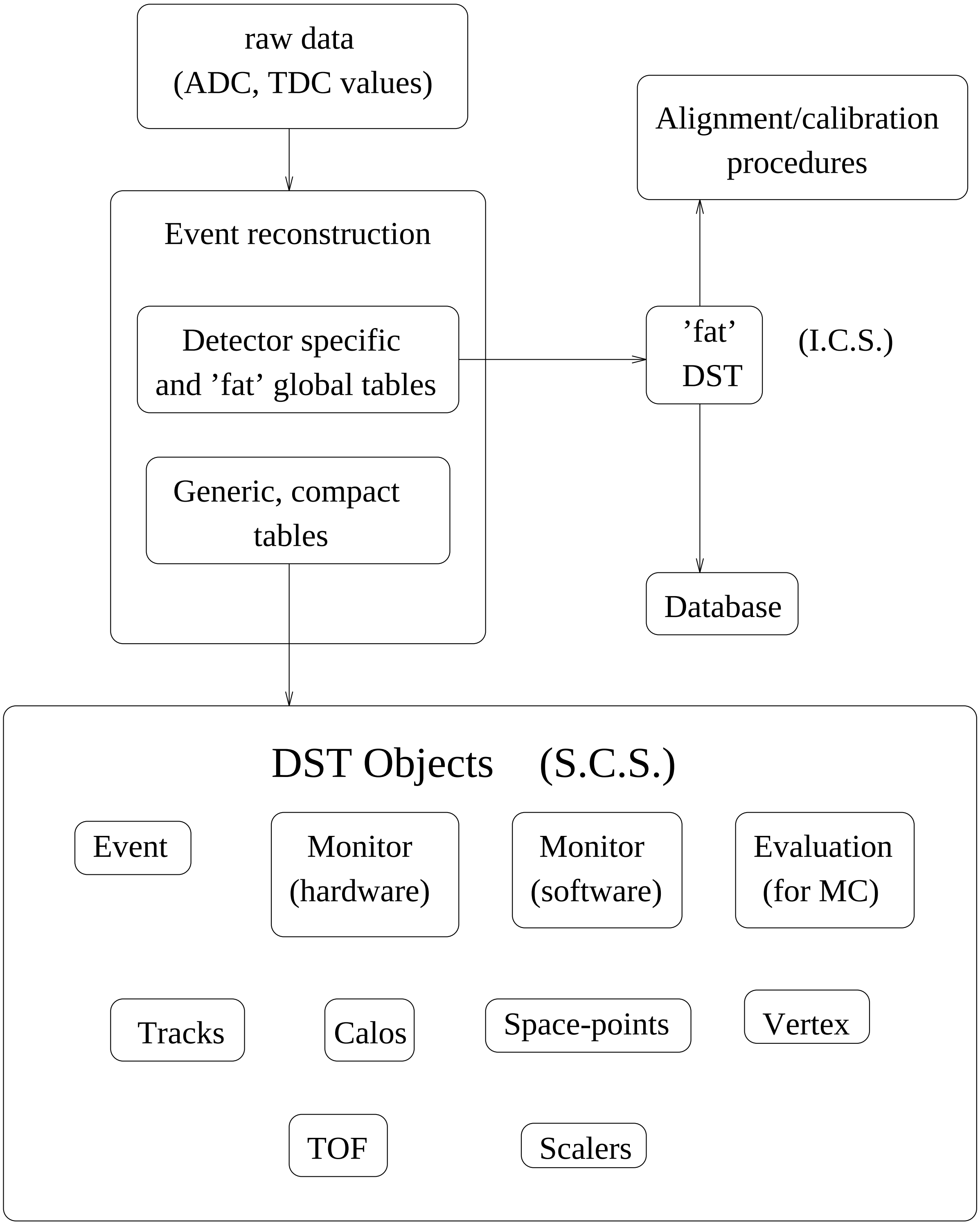

Figure 1: The DST objects (in the Star Coordinate System, described in text), shown in relation to the 'fat' DST objects (in Internal Coordinate Systems), event reconstruction, database, and calibration procedures.

3 The Event object

Below is a proposed structure for the Event object which is nothing more than an event header.

```
/*  dst_event.idl
*   Table: dst_event
*/
   struct dst_event  {
       long/byte       event_type;    /* e.g. Au+Au, p+p, laser ..       */
       long/byte       n_run;         /* Run number                      */
m      long            n_event[2];    /* Event number-64bits per run     */
       long/byte       trig_mask;     /* Trigger mask                    */
       long            date;          /* Date                            */
-      long            time;          /* Time                            */
   } ;
/*  Last mod. for dst_event:   $Date:  $ */
```

MDC: n_event will suffice with array depth one.

4 The Monitor object

In this structure some useful information about the state of the detectors and the environment is saved for future reference. The are two monitoring structures, one for hardware monitoring and one for monitoring with software. Since both objects have only one instance (row) per event they do not add much to the overall size, even if they contain somewhat redundant (but useful) information. The actual dimension of the arrays can be set later, closer to the start-up.

```
/*  dst_monitor_had.idl
*   Table: dst_monitor_had
*/
   struct dst_monitor_had  {
-      float     pressure;     /* pressure (mb)                            */
+      long      gas;          /* gas mixture used in TPCs                 */
       float     gain[4];      /* e.g.  gas_gain                           */
       float     v_drift[4];   /* drift velocity                           */
-      float     temp[4];      /* e.g. gas temperature, probes             */
       float     mag_field[4]; /* field ON/OFF, probes                     */
   } ;
/*  Last mod. for dst_monitor_had:   $Date:  $ */

MDC: actual array size might be smaller (1--2)
```

```
/*  dst_monitor_soft.idl
*   Table: dst_monitor_soft
*/
   struct dst_monitor_soft  {
       long      n_points[4];  /* number of reconstructed points          */
       long      n_tracks[4];  /* number or reconstructed tracks          */
       long      n_vertex_track;/* number of tracks associated to event vertex  */
-      float     sum_charge[4];/* sum of all charge                       */
-      float     avg_res_x[4]; /* average residuals in x                  */
-      float     avg_res_y[4]; /* average residuals in y                  */
   } ;
/*  Last mod. for dst_monitor_soft:   $Date:  $ */

MDC: TPC (tpt) tracks, points [baseline info]
```

5 The Vertex object

In this structure we propose to combine event vertex information as well as candidate secondary vertex information(V0s, kinks etc.) which is of common type. Such common type of information is for example the vertex coordinates. Individual analyses can then have auxiliary vertex tables containing all the variables specific to the analysis. Each auxiliary table will have a unique integer identification number. The generic vertex table (below) can then point to (and into) the auxiliary table using the auxiliary table id and a foreign key. Naturally, the auxiliary table may point as well to the generic vertex table, using the primary key in the generic vertex table.

```
/*  dst_vertex.idl
*   Table: dst_vertex
*/
   struct dst_vertex  {
       long      id;            /* Primary key                          */
       long/byte iflag;         /* bitmask of e.g. quality information    */
       long      id_det;        /* detector id where vertex was found        */
       long      id_vtx;        /* vertex type                          */
       float     x;
       float     y;             /* vertex coordinates in STAR reference system  */
       float     z;
-      long      sigma;         /* position uncertainty (x,y,z packed)    */
       float     pchi2;         /* P(chi^2,ndf) of vertex fit        */
       long     id_aux_tbl;     /* id of auxiliary vertex table     */
       long  id_aux_ent; /* foreign key into auxiliary vertex table */
```

```
   } ;
/*  Last mod. for dst_vertex:   $Date:  $ */
```

MDC: ev0out, exiout and tkfout tables can be the auxiliary tables with minor modifications.

6 The Point object

Although it is not foreseen to save any space-point information into STAR DSTs, it might be imperative for some analyses to save a subset of the space-point information, i.e. space-points on tracks of V0 candidates, or space-points on or near very-low-relative-momentum pion pairs. Since 'space-points' are most likely the more numerous object (followed by 'tracks'), storage space is an important issue.

As an example, the E895 collaboration has decided to save (compressed) space points in their DSTs. The structure proposed here is based upon the E895 scheme. See Section 11.2 of this document for details.

The proposed structure below is in the STAR reference system which makes it easier to handle in inter-detector software. Through extensive packing of information it is also by more than a factor of two smaller than a typical individual detector point structure; this packing scheme may still be optimized.

One space-saving aspect of this packed table is the absence of a primary key. This removes an entire word. If it is desired to have the track point to a space point, then one may use the array index. However, it is probably not necessary at all for the track to point to the space-point, since the space-point points to the track to which it belongs.

```
/*  dst_point.idl
*   Table: dst_point
*/
   struct dst_point  {
       long      hw_position;   /* detector, sec, row, pad, time sizes  */
       long      position[2]; /* compressed x,y,z of space-point      */
       long      charge;        /* total charge and max_adc             */
       long      id_track;      /* Foreign key into tracks table        */
   } ;
/*  Last mod. for dst_point:   $Date:  $ */
```

The data is compressed into these fields with a particular scheme, which must be well-defined (see Sections 11.2.2 and 11.2.4 for similar considerations in the E895 experiment). One possible compression scheme is given here.

hw_position (32 bits)

- Bits 0-3 are used to identify the subdetector (TPC, SVT, TOF...).
- Bits 4-10 identify the sector/ladder/slat.

- Bits 11-20 identify the padrow/layer-wafer.
- Bits 21-24 identify the number of pads/anodes associated with the space-point.
- Bits 25-31 identify the number of time buckets associated with the space-point.

position (64 bits)

- Bits 0-20 give x position (SCS).
- Bits 21-40 give y position (SCS).
- Bits 41-60 give z position (SCS).
- Bits 61-63 are flags (e.g. "is the space-point overlapping with a neighbor?")

This scheme for position is just one possibility. Indeed, it may be more natural to store the SCS position in cylindrical coordinates. However, it is clear that 64 bits is plenty for storage of space-point position. For example, 20 bits for z gives 0.4 micron storage accuracy for the $\sim$44 cm-long SVT, and 4 micron storage accuracy for the $\sim$4 m-long TPC. If there are 10^6 space-points in each event, storage of the space-point position in two 32-bit words instead of 3 leads to a savings of 4 MBytes/event.

charge (32 bits)

- Bits 0-19 for integrated/summed charge associated with space-point.
- Bits 20-31 give the maximum ADC value contained in the cluster.

Saving the charge associated with each space-point allow post-DST innovation in particle identification schemes (for a simple example, one may try using truncated mean that depends on dip angle).

7 The Track object

This structure is the result of combining detector-specific tracking information into single, 'global' tracks. It uses the helix track parametrization and the track parameters are given at the track's closest point to the event vertex. Note that whether or not the primary vertex is used in the track fit the resulting curve is never (well, very unlikely) going to go through it, therefore z0, r0 and phi0 have non zero values. Multiple entries of the same track are possible since one track might be compatible with several secondary vertex hypotheses, i.e. the same set of points will be fitted to a different vertex hypothesis together with other points each time.

'Track' and 'vertex' is a typical example of a many-to-many relationship, but as we mentioned above, track refitting breaks this to a many-to-single relation. This is why there is a single vertex key in the structure.

We anticipate to have two instances of this table saved on DSTs. One for global tracks fitted w/out the main vertex included in the fit, and one with a subset of selected tracks fitted with the main vertex included in the fit.

```
/*  dst_track.idl
*   Table: dst_track
*   Comment: Parameters are given in the STAR
*                      reference system. The track parameters
*            are given at the d.c.a. (fitted) from event vertex.
*/
   struct dst_track  {
       long      id;            /* Primary key                            */
       long/byte      iflag;         /* bitmask quality information        */
       long/byte      id_det;        /* detector id                        */
       long/byte      n_point;       /* number of points                   */
       long/byte      n_max_point;   /* maximum number of points possible */
       long/byte      n_fit_point;   /* number of points used in fit      */
       float     phi0; [or x0]        /* phi angle at start                */
       float     r0;   [or y0]        /* r reference radius at start       */
       float     z0;            /* z-coord. at start                      */
       float     psi;           /* momentum angle at start                */
       float     tanl;          /* tan(dip) =pz/pt at start               */
       float     invpt;         /* 1/pt at start                          */
-       float     covar[15];     /* covariance matrix                      */
       float     x_first[3];    /* coord. of first measured point    */
       float     x_last[3];     /* coord. of last measured point     */
       float     charge[2];     /* 'truncated mean' charge and variance   */
       long      id_particle;   /* id of corresp. entry in particle table  */
       long      id_vertex;     /* vertex id to which this track belongs  */
   } ;
/*  Last mod. for dst_track:   $Date:  $ */
```

MDC: a trimmed down glotrk table can be this one.

8 The Scalers object

We do not know how to do this thing for the moment in STAR!!!

MDC: we do not need it. Normalization of spectra can be done using event generator information.

9 The TOF object

```
/*  dst_tof.idl
*   Table: dst_tof
*/
   struct dst_tof  {
       long      iflag;         /* quality flag               */
```

```
      long      id_det;        /* detector element id       */
      float     x;             /*                           */
      float     y;             /*                           */
      float     z;             /*                           */
      float     pathl;         /* tracklength from target to hit */
      float     charge;        /* normalized charge of hit  */
      float     tof;           /* tof  from target to hit   */
      float     sig_tof;       /* sigma of tof              */
      float     beta;          /* pathl/tof (/c)            */
      float     sig_beta;      /* sigma of beta             */
      float     mass2;         /* squared mass              */
      float     sig_mass2;     /* sigma of mass**2          */
      float     id_track;      /* track id associated to this tof entry*/
   } ;
/*  Last mod. for dst_tof:   $Date:  $ */
```

What is left unspecified, for the moment, is the calorimeter as well as the evaluation structures.

10 Proposed STAR DST objects data volume

Here is given a rough estimate of the size of the DST per event, based on the structures discussed above.

Data Structure	Words/Entry	Entries	Bytes
Event	7	1	28
Vertex	10	~70	2800
Tracks	37	~6000	~888000
Monitor-hardware	17	1	68
Monitor-software	21	1	84
Scalers	16394	1	65576
TPC hits (compressed)	5	~200000	~4000000
TOF	14	7776	435456
Total (without hits)			1.39 M
Total (with ALL TPC hits)			5.39 M

11 DST Strategies of similar experiments

Many of the issues involved in STAR DST design have been considered by similar experiments. Here, we briefly discuss the DST design employed by the NA49 [4] experiment at CERN, and the E895 [5] experiment at the AGS; both are TPC-centered experiments with a similar event reconstruction philosophy as STAR. Both experiments are mature and currently in the DST design and implementation phase.

Although the STAR event reconstruction scheme is documented, there may be less familiarity with the E895 and NA49 schemes. A sketch of them is given here, for context.

11.1 NA49 DST Strategy

NA49 employs four large TPCs to measure Pb+Pb collisions at 160 AGeV at the CERN SPS. Along with the TPCs, which dominate the data volume and reconstruction philosophy, are TOF walls and calorimetry. The TPC system consists of four TPCs. Two small TPCs (VTPC) sit in each of the two vertex magnets and two large TPCs (MTPC) are located left and right of the beam in the field free region down stream of the magnet system.

Data reduction in NA49 is designed as a two-stage process, in which raw data is processed into DSTs, and the DSTs are further reduced into mini-DSTs. The production of the mini-DSTs represents no selection on event type; thus they are not micro-DSTs.

The average raw data size of a typical NA49 event is 8 MB. An average event size on the DST is typically 2.2 MB, and on the mini-DST, it is ∼0.31 MB. Thus, we achieve a factor of ∼25 data reduction in going from raw data to mini-DST.

11.1.1 Event Reconstruction and DST Production

Space points and particle tracks are reconstructed within each TPC and the tracks from each TPC are matched to find global tracks spanning more than one TPC. PID is performed with the dE/dx calculations from MTPC tracks and TOF in formation. DST contains information from full event reconstruction which includes space point reconstruction, track finding and fitting, event vertex fitting, topological search (V0 and kinks), track dE/dx calculation and PID. Since the NA49 DST production represents an intermediate stage in going from raw data to physics analysis, the DSTs contain lots of redundant tracking information from all TPCs to facilitate the performance evaluation of the reconstruction software. Although the DSTs contain redundent tracking information, they do not contain any form of space point information since the topological searches are done at the event reconstruction stage. Each DSTs event contains 49 C-structures which amounts to ∼2.19 MB of information. Following is a full list of DST data structures in the ascending order in data volume for a typical event.

```
-----------------------------------------------------------
Structure Name       Words/Entry    Entries   Bytes
-----------------------------------------------------------
covar_vtx_fit             6             1        24
bpd_vertex                7             1        28
tofgl                     7             1        28
tofgr                     7             1        28
tofl                      7             1        28
```

tofr	7	1	28
Scripts	1	10	40
monitor_tofgl	12	1	48
monitor_tofgr	12	1	48
monitor_tofl	12	1	48
monitor_tofr	12	1	48
all	15	1	60
mtl	15	1	60
mtr	15	1	60
vt1	15	1	60
vt2	15	1	60
vertex_fit	18	1	72
monitor_mtl	23	1	92
monitor_mtr	23	1	92
monitor_vt1	23	1	92
monitor_vt2	23	1	92
sihodo	25	1	100
veto	35	1	140
event	42	1	168
avertex_mt_kink	61	1	264
beam	100	1	400
ring	481	1	1924
covar_kink_fit	451	1	1980
bos_gtof	800	1	3200
avertex_v0	18	69	4968
vertex	18	81	5832
tof_hits	9	173	6228
v0info	24	69	6624
bos_tofl	2000	1	8000
bos_tofr	2000	1	8000
track_tof	17	143	9724
covar_v0_fit	45	69	12420
g_track_ref	4	1598	25568
rtrack_quali	13	540	28080
covar_itrack	10	1174	46960
g_ptr_list	8	1598	51136
mini_pnt_hdr	13	1598	83096
covar_track	6	3883	93192
itrack	24	1174	112704
covar_rtrack	15	2900	174000
atrack	12	3883	186384
mini_pnt_lst	1	84874	339496
track	27	4039	436212
rtrack	33	4074	537768

```
Total: 49 Structures                               2185704
-----------------------------------------------------------
```

NA49 experiment collects about 1 million events per year and therefore will produce about 2 TB of DSTs per year.

11.1.2 mini-DST Production and Physics Analysis

Because of the substantially large size of NA49 DST data volume, a further 'lossless' data compression step is desirable to produce mini-DSTs[6], which should by definition represent the minimum data volume containing the complete event kinematics. These mini-DSTs will be used as the basis for most of the NA49 physics analyses and it further lowers the bandwidth and storage requirements for the physics analyses to be carried out at the local institutions. No event selection is done at the mini-DST production stage.

Each mini-DST event contains C/C++ data structures for;

- Event
- Beam
- Beam Position
- Primary and secondary vertices
- Particles (TPC and TOF)
- Monitor (TPC and TOF)

The total mini-DST data volume per event is estimated to be about 301 kB. Following is the estimated mini-DST data volume break down by structures.

```
----------------------------------------------------------------------------
   Structure                 Entries                Words/Entry       Bytes
----------------------------------------------------------------------------
     Event             :           1                   37                148

     Vertex            :         ~60                   13              ~3120

     Particles
        particle       :       ~1600                   16            ~102400
        rparticle      :       ~1600                   13             ~83200
        tof_particle   :        ~150                   11              ~6600

     Covariance Matrix:        ~1600                   15             ~96000

     V0 Fit Hypothesis:          ~70                   10              ~2800
```

Beam				
Beam	:	1	100	400
Sihodo data	:	4	199	3184
Sihodo beam	:	1	25	100
BPD data	:	8	23	736
BPD beam	:	1	29	116
Calorimeters				
Ring	:	1	481	1924
Veto	:	1	35	140
TPC Monitors	:	4	23	368
TOF Monitors	:	4	12	192
Total	:	75357 words		301428

11.2 E895 DST Strategy

E895 uses the EOS TPC to measure excitation functions at the Brookhaven AGS. Collisions at 2, 4, 6, and 8 AGeV have been measured. The TPC may be considered the only relevant detector to the current discussion. E895 is currently producing DSTs. This document will be updated as needed.

The philosophy in E895 is to proceed directly from the raw data to the DST. This strategy has been inherited from the EOS experiment, but modified/improved to reflect the experience of EOS.

The size of a raw event ranges from 0.5 - 1.1 MB as the energy is raised from 2 to 8 AGeV. Details are below. The DST data reduction factor is of order 10-20%.

11.2.1 Event Reconstruction and Structures Used in Physics Analysis

Event reconstruction to the DST level in E895 follows a similar path as in other TPC experiments (except for a slightly unusual iterative procedure of hitfinding and track-finding): [Hitfinding] + [Trackfinding] + [Hitfitting] + [Trackfitting] + [Track dEdx calculation] + [Primary Vertex Position] + [Event Summarization]. PID assignment based on the energy loss of the track, and secondary vertex searches are left for after DST production.

In some ideal sense, a DST event should be a list of particles (including decays) that are perfectly reconstructed from the data. Indeed, PID and vertex-finding analysis modules exist which do construct such a list. However, experience from EOS and pre-DST analyses of E895 data indicates that small problems in tracking or topological studies (V0 searches) can severely hamper physics analysis. An example is the bad effects that track-splitting has on HBT analysis. Therefore, it was the decision of

E895 to store as much information as possible, in order to allow partial reconstruction or fixing of problems as new methods are developed (e.g. identifying and merging split tracks via space-point information).

The dominant structures filled by the reconstruction chain, and used in subsequent analysis, are the HITS and TRKS tables. Physics analysis modules exist (some new and some inherited from EOS) which take these tables as input. It was decided not to change these structures radically, as it would require a major rewrite of code. As fortran structures, they are given explicitly below:

HITS

```
STRUCTURE /hits_row_st/
   integer satf    ! equal 1 for saturated hits, 0 otherwise
   real alpha      ! Track crossing angle w.r.t wire normal
   real decon      ! Deconvolution Algorithm Identifier
   real dedx_ampl  ! de by amplitude
   real dedx_area  ! de by integrating signal
   real dx         ! uncertainty in x
   real dy         ! uncertainty in y
   real geantid    ! Geant track ID
   real lambda     ! dip angle
   real npads      ! N pad cluster
   real padrow     ! Padrow number
   real sigma_prf  ! calculated width of pad response
   real trackid    ! Track identification number if any
   real x          ! x coordinate along pad row
   real xdev_fit   ! x residual w.r.t fitted track
   real y          ! y coordinate, drift direction
   real ydev_fit   ! y residual w.r.t fitted track
   real z          ! z coordinate along beam direction
   integer pad1st  ! padnumber of first pad in a cluster
   integer ntracks ! num of tracks the tracker associates with hit
   real dedx       ! dedx accounting for crossing angles
END STRUCTURE      ! /hits_row_st/
```

TRKS

```
STRUCTURE /trks_row_st/
   integer geantid ! geant track number of best match
   integer ia(3)   ! Mass number (3 best hypotheses)
   integer iz(3)   ! charge (3 best hypotheses)
   real weight(3)  ! confidence level of PID hypotheses
   integer nhits   ! Number of hits associated with track
   integer parent  ! parent GEANT particle id for v0
   integer trackid ! Track number
   real chi2       ! Chi-square
   real r0(3)      ! Rigidity before vertex fit
```

```
      real x0(3)       ! space coords at pt corresponding to r0
      real xb0(3)      ! space coordinates at 1st hit
      real xe0(3)      ! space coordinates at last hit
      real dca         ! distance of closest approach to vertex
      real dedx        ! truncated mean dedx
      real dqphi       ! cos of half angle
      real dr(3)       ! Uncertainty in rigidity
      real qphi        ! reaction plane azimuth
      real r(3)        ! Rigidity at r_x,r_y,r_z
      real x(3)        ! space coords at pt corresponding to r
      real xb(3)       ! space coordinates at 1st hit
      real xe(3)       ! space coordinates at last hit
      real hpar(5)     ! Helix Fit parameters at beginning of track
      real rb(3)       ! Rigidity at 1st hit
      real sigma       ! average width of hits on track
      real frac_seen   ! Ratio: nhits to calculated padrows crossed
   END STRUCTURE       ! /trks_row_st/
```

So the HITS table has 21 4-byte variables, for 84 bytes/hit. The TRKS table has 55 4-byte variables, for 220 bytes/track. These structures are clearly too "fat" to store as DST. As an example, for the 2 A GeV events, the raw data is 0.488 MB/event. about 10000 hits and 160 tracks are reconstructed from this data, which would give 10000*84 bytes + 160*220 bytes = 0.875 MB/event, dominated by the hits.

11.2.2 Packing the HITS and TRKS Tables in DST

In order to do data *reduction*, the E895 collaboration examined which elements of the HITS and TRKS tables were essential/desirable for physics analysis. Next, the measurement resolution of quantities (e.g. the x-coordinate of a hit) was evaluated in order to determine the amount of space (bits) to give to storage of the quantity. Finally, a tight packing scheme for the HITS and TRKS into new tables, HITS_PACK and TRKS_PACK was developed. These smaller, bit-encoded tables are stored in the DST. When a user reads a DST file, the first analysis module he runs is UNPACK, which decodes the HITS_PACK and TRKS_PACK tables and fills HITS and TRKS.

Saving the hits in DSTs has several obvious advantages, some of which have been mentioned. These include advanced and developing topological searches, track refitting, trying different PID schemes (e.g. by using a different truncated mean), and mending of split tracks. There is clear benefit.

However, as seen above, without packing, HITS dominates the data volume, and so there is clear cost. For the cost/benefit ratio to be reasonable, the volume must be limited as much as possible through packing.

It was decided to save: the position (x,y,z), the ID of the track, the dEdx of the hit, the number of tracks that claim the hit, and some flags. The number of bits to spend on each variable was decided as follows:

- **x** - The position along the padrow is measured with a resolution of roughly 500

μm, and has a span of 96 cm. Taking a span of 100 cm (allowing for slight shifts of TPC in the magnet coordinate system), the span/resolution ratio is 2000. Therefore using 11 bits to save the x coordinate would give storage precision equal to measurement resolution. One would like to do a bit better than this– say a factor of 10– in order not to degrade the data, so 14 bits are used. Thus, the storage precision is 100 cm/2^{14} = 61 μm.

- **y** - Similar considerations apply for the position along the time direction. Here again the span is roughly 100 cm, though the measurement resolution is a bit worse– about 800 μm. 14 bits are again used, for a 61 μm storage precision.
- **z** - The z coordinate, along the beam direction, is given essentially by the padrow number. (Distortion corrections due to a non-uniform magnetic field are applied during the unpacking procedure.) E895 has 128 padrows, so 7 bits are used.
- **dEdx** - The dEdx of the hit is saved to allow innovations in particle ID, or recalculation of *track* dEdx truncated mean in case the hit is reassigned in later tracking. Detailed studies show that using 14 bits to store log10(dEdx) is more than sufficient to preserve PID resolution.
- **Track ID** - 10 bits is used for this, allowing for events with multiplicity 1024.
- **Flags** - 4 bits are used to store flags relating to the "quality" of a hit and the number of tracks that claim it (almost always 1).

This packing scheme takes 63 bits. The TAS infrastructure used by E895 allows saving only 4-byte variables, so these are packed into 2 I*4's, leaving one bit free. Thus, the hits for the 2 A GeV event take 8 kB, instead of 840 kB for the unpacked table.

The TRKS table is also packed, but the efficiency is less crucial, and will not be discussed here. For E895, the packed TRKS table is encoded into 23 I*4's, so it is 92 bytes/track, compared to 220 bytes/track in the unpacked version.

For the various energies, relevant numbers are given below, for saving packed hits and tracks tables, or only the tracks table. Numbers are per event.

Beam Energy	Raw Size	Ngood hits	Ntracks	DST Size (reduction) TRKS only	DST Size (reduction) HITS and TRKS
2 AGeV	488K	7000	160	14.7K (3%)	56K (14%)
4 AGeV	578K	11000	280	25.7K (4%)	89K (20%)
6 AGeV	905K	12000	350	32.2K (3%)	96K (14%)
8 AGeV	1080K	16000	450	41.4K (4%)	128K (16%)

11.2.3 Other Event-wise Tables Stored in DST

Other data tables, such as the table storing the position of the primary vertex, are also stored in the DST on an event-by-event basis. These contain the obvious information, and are small, so efficiency of packing is not necessary.

- CAMAC table (containing various raw ADC and TDC values)
- VTX table containing the position of the primary vertex
- EVENT_SUM table, structure shown below.

The EVENT_SUM table contains information summarizing the event and the processing of the event. The structure is shown below.

EVENT_SUM

```
STRUCTURE /event_sum_row_st/
   integer evtnum        ! DST event number (``empty events'' excluded)
   integer good_trkmul   ! number of ``good'' tracks
   integer hitmul        ! number of hits
   integer iev           ! raw event number (``empty events'' not excluded)
   integer trkmul        ! number of tracks in TRKS table
   real ave_nhits        ! average TRK length
   real dedx_cpu         ! cpu seconds in dEdx module
   real hit_cpu          ! cpu seconds in hitfinding module (1st pass)
   real hit2_cpu         ! cpu seconds in hitfinding module (2nd pass)
   real hre              ! fraction of found hits used in found tracks
   real trk_cpu          ! cpu seconds in tracking module (1st pass)
   real trk2_cpu         ! cpu seconds in tracking module (2nd pass)
END STRUCTURE            ! /event_sum_row_st/
```

11.2.4 Run-wise Tables Stored in DST

There are "non-data" tables stored in the DST that pertain to the run as a whole. These are the so-called "run summary table" and the switch tables containing run-time parameters for the analysis modules (e.g. tracker) called.

The switch tables for each of the analysis modules that was used to reconstruct the event. So, the following switch tables are stored:

- Switch table for Hitfinder
- Switch table for Tracker
- Switch table for dEdx Module
- Switch table for Primary Vertex Finder
- Switch table for Packing Module (*see discussion below).*

The switches for a given analysis module are quite specific to the module, but it is worthwhile to point out that the Packing Module (which does the bit-encoding of the HITS and TRKS tables) has its own switches, which are crucial when unpacking the data upon reading the DST. These switches include the upper and lower boundary of each quantity being packed (e.g. the x-coordinate is between -50 cm and +50 cm). It also includes a "version number," to allow for innovations in packing (e.g. to give some quantity better storage precision at the expense of another). This version numbering scheme has proved invaluable, as E895 currently is on version 4 of packing. Since we store the version, and the unpacking module unpacks the data according to the version switch, "old" DSTs can be read equally as well as "new" ones.

- Run summary table

Finally, a "summary table" is filled, containing information about the percentage of hits used by the tracker (a quality check), average multiplicity and RMS of multiplicity distribution. Also in this table are quantities calculated as the processing proceeds; information of this type includes the average dEdx gain for this run. Such information obviously makes the job of post-DST modules, like PID assignment, much easier. The RUN_SUM table structure is given here.

RUN_SUM

```
STRUCTURE /run_sum_row_st/
   character*80 infile   ! input file name
   integer nevents       ! number of (non-empty) events
   integer queue         ! daq taping queue
   integer run           ! run number
   real dedxgain         ! gain factor for this run
   real avehitmul        ! average number of hits/event
   real avehre           ! average fraction of hits used in tracks
   real avenhits         ! average track length
   real avetrkmul        ! average track multiplicity
END STRUCTURE            ! /run_sum_row_st/
```

On a technical point, it is clear that the summary table is only filled at the *end* of DST production for a given file, whereas it is convenient for the DST consumer to have such information at the onset of DST reading (whether for scanning for run quality, or to pick up calibration constants such as dEdx gain). Like STAR, E895 uses the xdf format for storage of tables. An advantage of this format is that xdf files can be concatenated (via the "cat" command in a unix shell). Therefore, the data portion of the DST (HITS and TRKS tables) are put to one file, while the switches and summary table are written at the end of processing to another file, and then the two are concatenated such that the switches and summary tables appear at the beginning of the resultant DST. This is a final step in the automated procedure to produce an E895 DST file.

11.2.5 Summary of E895 Experience

E895 is an experiment similar to STAR in philosophy and setup, if not in energy range. The outline of its DST strategy has been outlined above. Based on previous EOS experience, E895 has decided to do some things differently than STAR. Perhaps the main difference is the storage of hits; this has costs (increased DST size), but benefits that prevail, for E895.

E895 experience points out some things to keep in mind in DST design and production:

- Know resolution with which quantities are measured and store values not more than 10x better than that.
- Pack data efficiently.
- At least consider cost/benefit of explicit storage of low-level objects (hits).
- Switch and Summary tables at beginning of run allow for quick quality scanning and pick-up of higher-order calibration constants.
- Use a well-documented packing version numbering scheme, to allow for flexibility and innovation in data packing, without making previous DSTs obsolete.

References

[1] STAR – Conceptual Design Report, PUB–5347(1992)4C-44 and 4H-13

[2] L. Ray, et al., STARNote 0281: 'STAR Offline Simulations and Analysis Software Design'.

[3] E. Judd et al.; Talk given at the STAR collab. meeting, BNL (1996)

[4] Web documentation for the NA49 collaboration can be found at http://na49info.cern.ch/na49/

[5] Web documentation for the E895 collaboration can be found at http://cnr2.kent.edu/∼e895/

[6] Web documentation for the NA49 mini-DST proposal can be found at http://hpna49-1.cern.ch/∼rolandg/str_proposal.html