ALEPH 94-187
SOFTWR 94-016
G. Graefe
8 December 1994

SANDY User's Guide

SANDY

SPEEDY ANALYSIS ON THE NANODST BY YVES

Version 115

Authors:

P. Colas, G. Graefe, Y. Maumary

With contributions from
D. Brown, G. Cowan, H. Duarte, I. ten Have, M. Ikeda, M. Maggi, A. Putzer, S. Schael and F. Stephan.

PREFACE

In the past the total number of events collected by ALEPH has nearly doubled every year, the number of algorithms supposed to be useful for many analyses has increased as well as their complexity and therefore their execution time. Even if the computers are faster the time effort for a heavy flavour analysis is bigger then 2 years ago. For this reason more and more people are interested to use the NanoDst in future. So it became nesseccary to change a lot compared to the last version (111) to get a small data set which contains almost the same information used for analyses as larger ones like the MiniDst or the Dst. Some new features which are supposed to be useful for many analyses have also been introduced.

The main difference between SANDY111 and SANDY115 is the V^0- section where the V^0s found by YRMIST were replaced by those from the YV0V bank. A consequence of this is the change of a lot of mnemonics in the **V0T** section. However, due to the fact that it was always policy of the SANDY authors to keep the NanoDst as small as possible not all V^0s from YV0V are kept. For more details see section 5.4.

In the last NanoDst production leptons were taken from the LEPTAG output and -separated by electrons and muons- put in different banks. This has changed as they are now kept in the new NDBM bank which is a reduced LEPTAG output bank BMLT. The LEPTAG code was changed to run on the NanoDst as well as on the Mini or Dst. However it is not nesseccary in all cases to call LEPTAG, this is explained in 5.8.

QIPBTAG will work now on the NanoDst taking information from the NBIP bank which is new too. Unfortunately this has some restrictions, read section 5.12 for more.

Completely new is the **GCO**- section (5.5) which contains Gamma COnversions. They were found either by QPAIRF or by a new routine called QACONV and stored in the NDGC bank. Inside SANDY they are loaded into the work bank and can be accessed via pointers like tracks.

The main vertex is stored for the first time on the NanoDst and can be acessed via a statement function (3.2). Another statement function have been introduced for true V^0- vertices in Monte-Carlo events (5.6).

Internally there is another difference between the old and the new format, all NanoDst banks (Nxxx) are integerized now and therefore their numbers were changed from 0 to 1. The integerization gives the opportunity to leave some day the input/output through BOS routines which will give a remarkable speed up in reading the NanoDst (factor 3-5 !). Of course it is connected with a little loss of precision compared to the Dst but on this could not be avoided as in future the NanoDst has to be produced from the MiniDst. As all MiniDst banks are integerized as well all quantities stored on the NanoDst are as precise as they are on the MiniDst.

For the platforms without HISTORIAN three 'new' files are avaliable, NCDE. INC, NMACRO. INC and NDECL.INC which can be loaded via an INCLUDE statement. An example can be found in NANO:NUUSER.FOR.

Acknowledgements

I have to thank a lot of people, D. Brown and A. Putzer for valuable suggestions and discussions, I. ten Have and D. Brown for making LEPTAG and QIPBTAG running inside SANDY and D. Brown, G. Cowan, A. Putzer and F. Stephan for providing some of the new code.

Last but not least I wish to thank P. Colas and Y.Maumary who have originally written this manual to which I only made some changes where nesseccary.

Contents

Chapter 1

INTRODUCTION

Preliminary Remark

The structure of the SANDY package was to a large extent inspired by ALPHA (ALEPH PHysics Analysis package). Many ideas were therefore 'stolen' from it and hence we acknowledge the work of its authors, H. Albrecht and E. Blucher. As there exists a very nice and detailed manual for ALPHA (ALEPH-note 94-092, SOFTWR 94-007), please refer to it for more details about those features and routines which are mentioned here as being similar or having the same functionality.

The NanoDST (Nano - Data Summary Tape) and SANDY (the package which reads it) have been created to solve several problems:

1. Running with ALPHA on the full DST sample has become very time-consuming and CPU-intensive. One would like to be able to go through the complete data sample within a few hours or to debug the programs on a significant part of the data within a few minutes.
2. Disk space is low, especially at homelabs. It is thus important to store the data in a form as compact as possible.
3. When a working group divides its analysis in various tasks, it is important that everybody in the group uses the same event selection, the same particle identification, etc., and it is also preferable that each of these processings are done by routines optimized by specialists.

These three main features of the NanoDST; time-saving, diskspace-saving, and standardization, have benefitted the D* group and other individuals.

Time is saved by not having to rerun the time-consuming algorithms several times; the thrust axis, pre-clustering into jets, the lepton identification, photon and V^0 finding, are done once and for all. The tracks[1] are sorted by hemisphere, by charge, and ordered in momentum, in order to save time in the combinatorics.

Space is saved by keeping only the final quantities, the ones which are needed to do the analysis or to check it. For each quantity (momentum vector of a track for instance) control quantities are also stored (number of hits, χ^2 per degree of freedom from the track fit). For the Monte Carlo information, only *stable*[2] tracks' momenta, particle type code and mother-daughter relationships

[1]The word track is used throughout this note with an extended meaning: it implies both charged and neutral particles.

[2]The exact meaning of *stable* will be explained in section 5.6.

are stored: the intermediate four-momenta are reconstructed from this basic information at reading time.

The general philosophy of the NanoDST consists of running a long ALPHA program once in a few months (this task being done by ALPROD) and to save only the useful quantities on a file, which can be read many times a day. For the user, the analysis becomes almost interactive. Space is saved without mercy: information is packed on a minimal number of bits wherever the maximal accuracy of 32 bits is not necessary. For instance, the dE/dx information consists of the track length, the number of samples and the truncated mean ionization with its error; these four quantities are packed into two 32 bits words. The unpacking is done by SANDY, and the routine NDEDX/NDEDXM (SANDY versions of the ALPHA routines QDEDX/QDEDXM) can be called as in ALPHA.

For analyses requiring all the details of track hits or calorimeter stories or any other additional information, SANDY provides a simple tool (routine MAKSEV) to write SEVT cards which can be used directly in an ALPHA job to select DST events.

Comments, suggestions and bug reports should be made via E-mail to BOUCROT@CERNVM or GRAEFE@ALWS.CERN.CH

1.1 Acknowledgements

I would like to thank all those who helped with their comments and suggestions. In particular, thanks are due to J. Boucrot for introducing the NanoDST datasets into SCANBOOK, to R. Edgecock for their production with ALPROD, to R. Hagelberg for creating and installing the NANO disk on CERNVM, to S. Haywood for lots of suggestions and constructive criticism, and finally to H. Meinhard for installing the NANO directory on ALWS and the copying the datasets on the ALWS disks. Finally, I thank all the users, for SANDY would not exist without them.

Chapter 2

USER'S GUIDE

Corrections, modifications or updates not yet included in this manual can be found in the file SANDYnnn.NEWS (nnn is the version number).

2.1 Getting started

For the user, SANDY is very similar to ALPHA. To run a SANDY job, one has to supply:

1. A file with the user routines in HISTORIAN input or FORTRAN format.
2. A card file containing the names of input/output data files, as well as any other parameter cards.

Three routines should be provided by the user: initialisation (NUINIT), event analysis (NUEVNT), and termination (NUTERM). Examples for these user routines are given in the file `NUUSER.INPUT` or `NUUSER.FOR` which can be found in the NANO directory or disk. There is another user routine, NUNEWR (same purpose as QUNEWR in ALPHA), which may be modified by the user. All user routines exist as dummy versions in the library.

Data and Monte Carlo NanoDST files are produced by ALPROD. For inquiries about the available datasets or creation of FILI cards, please use SCANBOOK.

2.2 Name conventions

Most FORTRAN symbols[1] defined in the SANDY package start with Q, K, C, X or N:

Q	Subroutines, real functions, variables, or arrays
K	Integer functions, variables, or arrays
X	Logical functions, variables, or arrays
C	Character functions, variables, or arrays

[1]There are few exceptions: SUBROUTINE MAKSEV, and a few variables.

N Subroutines; functions, variables, or arrays of different types (mostly internal, or originating from the NanoDST production code running in ALPHA)

The names of user routines or variables should not start with any of these letters to avoid possible confusions and problems.

2.3 Access to SANDY variables

To access the SANDY variables and pointers, one must include a set of statements at the beginning of the routine, in the same way as for an ALPHA job. There are also two sets of statements:

NCDE	COMMONs, DIMENSIONs, PARAMETERs, type declarations (like QCDE in ALPHA)
NMACRO	statement functions (like QMACRO in ALPHA)

They exist as decks in the HISTORIAN library and should therefore be included using:

```
*CA NCDE
*CA NMACRO
```

Of course one could use a statement like

```
      INCLUDE ' NANO:NCDE.INPUT '        (VAX-VMS syntax)
      INCLUDE ' NANO:NMACRO.INPUT '      (VAX-VMS syntax)
```

instead, but it is recommended to use HISTORIAN except on those platforms where this is not implemented: this guarantees that you get the correct decks without having to worry about the version of the NCDE input file (for versions older than the current one, the files are renamed `NCDEnnn.INPUT`, resp. `NMACRnnn.INPUT`).

2.3.1 Implicit none

The deck NDECL contains the declaration (integer, real) of all SANDY variables and statement functions in NCDE and NMACRO, as well as of all SANDY functions. People wishing to use IMPLICIT NONE in their code should include NDECL preceding NCDE.

2.4 User routines

2.4.1 User initialisation routine NUINIT

This routine should be used to book histograms as well as for any other user initialisations. Standard initialisation work is performed automatically before the subroutine NUINIT is called which includes:

- Initialisation of BOS (500 000 words working space and 2000 bank names)
- Initialisation of HBOOK (500 000 words working space)
- Reading data cards
- Opening the ALEPH database

The booking of histograms and n-tuples should be made via the normal HBOOK subroutine calls. There are *no* equivalents of the ALPHA QBOOK, etc., routines in SANDY.

For row-wise n-tuples, 'SANDY' is the top directory name of the RZ file given in the HIST card (please refer to chapter 3 of the HBOOK Reference Manual Version 4.17 for more details).

2.4.2 Event analysis routine NUEVNT

This routine is called once for each event. The pointers to be used to loop over the objects (charged tracks, Monte Carlo tracks, photons, jets, etc.) are described in chapters 4 and 5. Notice that the charged tracks are sorted in four groups according to their *charge* and the event *hemisphere* to which they belong (defined by the thrust axis). They are further sorted within each group by *increasing* momentum. This allows substantial speed-up when combining tracks to compute invariant masses. An example illustrating how to make the most efficient use of this feature is given in section 5.1. The photons are also grouped according to the event *hemisphere* (same as for charged tracks), and in *decreasing* energy within each group.

2.4.3 User termination routine NUTERM

This routine can be used for anything which needs to be done at the end of the job, e.g. histogram manipulations. Histogram printing is done automatically in the SANDY routine NMTERM, as well as histogram output if you give a HIST card.

WARNING: this routine should never be called directly. To force program termination, you should call the main termination routine NMTERM with a statement like:

```
CALL NMTERM('any message')
```

NMTERM, in turn, will call NUTERM. NMTERM is called automatically after all input files have been processed.

2.4.4 New run NUNEWR(IROLD,IRNEW)

This routine is called automatically once a new run is encoutered on the event input file.

2.5 Running a SANDY job

VAX

The SANDY package (i.e. libraries, source code and other relevant files like `NUUSER.INPUT` (example of user routines) can be found under the logical `NANO`.

To run a SANDY job, just type

```
SANDYRUN [xxx|?]
```

(the short form `SANDY` is also accepted) where xxx is an optional file of type `NOPT` and containing the job options (similar to the ALPHA option file `myalpha.OPT`). If there is no such option file in the current directory, it will be created for you. A '?' as argument will provide some help. Furthermore, you can type '?' at any question to get help. Note that the batch job name (if you run in batch mode), process name and log file name are set equal to the option file name (SANDY by default), and that the so-called *program name* is what will be used for the FORTRAN file (file type `FOR`), the object file (file type `OBJ`) and the executable (file type `EXE`). These two names are independent, such that you can run several jobs simultaneously with the same executable [2] if they differ only by their respective card files. With a NSPY card, the process name of a SANDY job will be modified every thousandth processed event to indicate how many events it has processed so far. The user can monitor the running of his SANDY jobs very easily, e.g. with the DCL command

```
SHOW USER/BATCH/FULL my_name
```

which will provide the following information

```
Username  Node    Process Name       PID       Terminal
my_name   AL1F01  job_name >  95K  22C006A3   (batch)
```

indicating that more than 95 000 events have been processed so far.

Any running SANDY job can be terminated at any time without losing the results from the so far processed events; the job terminates correctly after a forced call to the main termination routine NMTERM. This forced call is activated by creating a file with the same name as the job name and file type `STOP` in the user's login directory (`SYS$LOGIN`). The content of the file is irrelevant. Once the file has been created, the job will end after the next event where `KNEVT` = $(0 \textit{ modulo } 1000)$.

[2]Executables and especially FORTRAN files grow fast into very large files such that space saving becomes interesting...

IBM

A REXX procedure SANDY EXEC similar to the SANDYRUN on VAX exists on the NANO minidisk on CERNVM. There are also examples of INPUT, CARDS and EXEC files which you should copy onto your disk and edit to your needs.

2.6 Control cards

FILI	Same syntax and functionality as in ALPHA.
SEVT	Same syntax and functionality as in ALPHA.
SRUN	Same syntax and functionality as in ALPHA.
IRUN	Same syntax and functionality as in ALPHA.
NEVT	Same syntax and functionality as in ALPHA.
NSEQ	Same syntax and functionality as in ALPHA.
FILO	Same syntax and functionality as in ALPHA.
COPY	Same syntax and functionality as in ALPHA.
READ	Same syntax and functionality as in ALPHA.
TIME	Same syntax and functionality as in ALPHA.
HIST	Same syntax and functionality as in ALPHA.
NOPH	Same syntax and functionality as in ALPHA.
MASS	(no arguments) The mass of a charged track is automatically set equal to that of the lightest particle for which the hypothesis bit is set (e.g. if `XEL` and `XKA` are `TRUE` for track `I`, the mass of track `I` is set equal to the electron mass).
JETB n	The jets are based by default on energy flow (P. Janot) objects (mass cut 4 GeV/c^2). By including a JETB card with n = 1, one can select jets with charged tracks only (mass cut 4 GeV/c^2), but the track-jet relations will no longer be available.
JETC 'new_mass'	The evolution of the chosen jets to the new mass cut is done automatically when the event is read in.
NSEL 'filename'	Allows you to specify the name of the SEVT cards file produced by the routine MAKSEV.
NSPY	(no arguments) The process name of the SANDY job is modified every thousandth processed event to indicate how many events it has processed so far (for VAX-VMS batch only, see section 2.5).

Do not forget the ENDQ card at the end of the cards file.

Chapter 3

MNEMONIC SYMBOLS

Mnemonic symbols are FORTRAN variables, arrays, parameters, functions or statement functions. Those marked with a * are identical to the corresponding ones in ALPHA.

3.1 Mathematical and physical constants

```
*QQPI   = pi
*QQE    = e
*QQ2PI  = 2 * pi
*QQPIH  = pi / 2
*QQRADP = 180 / pi
*QQC    = speed of light [cm/s]
*QQH    = Planck constant / ( 2 * pi ) [GeV/s]
*QQHC   = QQH * QQC
*QQIRP  = speed of light [cm/KGauss]
```

3.2 Run and event information

```
*KRUN   : RUN number
*KEVT   : EVenT number
*KNEVT  : Number of EVenTs processed

*XMCEV  : T if Monte-Carlo event
 KFLAV  : event flavor (u=1,d=2,s=3,c=4,b=5)

*XLUMOK: T if event used to compute luminosity
 NXHVOK: T if ITC and TPC voltage OK
 NXDEOK: T if TPC dE/dx voltage and calibration OK
 NXVDOK: T if good VDET data
 NXMUOK: F if QMUIDO/MUREDO failed
 NXLEPT: F if LEPTAG failed
 NXBJSC: F if Boosted Jet Sphericity Product on CHarged tracks failed
 NXBJSF: F if Boosted Jet Sphericity Product on ENflw objects failed
 NXBEES: F if BEETAG failed
```

```
NXGIPT: F if event not useful for QIPBTAG
NXIPBT: F if QIPBTAG failed
NXGAMP: F if GAMPEX failed
NXQFVX: T if main vertex from QFNDIP, from JULIA otherwise
NXQFFA: T if QFNDIP failed

*KREVDS: detector status word from REVH bank

 NJETB : jet bank loaded into JET section 0=EJET (default)
                                          1=NDJT
```

3.2.1 Main Vertex

Per default the main vertex is determined by QFNDIP. Only if this program fails, the JULIA main vertex is kept; use NXQFVX to get the origin of the vertex.

```
QMV(1): x-coordinate of main vertex
QMV(2): y-coordinate of main vertex
QMV(3): z-coordinate of main vertex
```

3.2.2 From GETLEP

```
QELEP : LEP center of mass Energy [GeV/c**2]
IFILL : Fill number
BEAMX : X,Y,Z values of beam crossing for this run , in [cm]
DBEAMX: Errors on BEAMX , in [cm]
NHADBX: Number of HADronic events used to compute BeamX , dbeamx
```

3.2.3 Event tags

Before using the tags, one should check that the tag computation was successful with the appropriate flag (see section 3.2). The authors of the algorithms are indicated. C. Bowdery's algorithm is used with a $y_{cut} = 0.02$ and a boost of 0.965. B. Brandl's algorithm is used with a boost of 0.96.

```
BJSPCH: Boosted Jet Sphericity Product, on CHarged tracks (C.Bowdery)
BJSPEF: Boosted Jet Sphericity Product, on ENflw objects (C.Bowdery)
BBSPR1: Boosted hemisphere 1 sphericity product (B.Brandl)
BBSPR2: Boosted hemisphere 2 sphericity product (B.Brandl)
BEETT1: BEETAG transverse mass hemi 1 (E.Manelli)
BEETT2: BEETAG transverse mass hemi 2 (E.Manelli)
BEETM1: BEETAG moment of inertia hemi 1 (E.Manelli)
BEETM2: BEETAG moment of inertia hemi 2 (E.Manelli)
```

3.2.4 Input/Output logical units

```
KUPRNT : log file = 6
KUPTER : terminal (VAX interactive mode only)
```

3.2.5 Character variables

```
SAVERS: SANDY version
NCLPOT: List of POT/DST/MINI banks on this NanoDST
NCLRUN: List of run banks on this NanoDST
NCLNDS: List of NanoDST banks on this NanoDST
```

Chapter 4

SANDY OBJECTS

When running a SANDY job, for each event you have access to global quantities which are stored in variables, and to 'tracks' (i.e. any object which can be described by a 4–vector). These objects are all stored in the rows of the same array, but in different sections. This allows to use the same macro (statement function) to access a given physical quantity, regardless of the kind of the object. Similarly, the same routines can be used to perform kinematics or track operations on any object. Each object is assigned a unique number, which will be referred to as the SANDY number of the object. FORTRAN DO loops can then be used to loop over most types of objects. For each type of object, three variables are defined: KFxxx, KLxxx and KNxxx, where xxx represents the object type. The first two are pointers to the beginning and the end of the corresponding section, and KNxxx is the number of objects of type xxx. DO loops must be made from KFxxx to KLxxx. The following types of objects are currently filled:

- **CHT** : charged tracks
- **JET** : jets
- **GAM** : photons (from GAMPEX)
- **V0T** : V^0 candidates (from YV0V)
- **GCO** : gamma conversions (from QPAIRF and QACONV)
- **MCT** : Monte Carlo truth

General macros which can be used with any object are described in section 4.1. Kinematics and track operations are described in section 4.2. Macros and routines to access specific information for the different types of objects are described in the various sections of chapter 5.

4.1 Macros

Macros marked with a * are identical to the corresponding ones in ALPHA.

I is the SANDY track number.

`*QSQT(Q)`	`signed square root`
`*QP(I)`	`momentum of vector I`
`*QX(I)`	`x momentum component`

`*QY(I)`	y momentum component
`*QZ(I)`	z momentum component
`*QE(I)`	energy
`*QM(I)`	mass
`*QCH(I)`	charge
`*KCH(I)`	INT(QCH(I))
`*QCT(I)`	cos(polar angle)
`*QPH(I)`	azimuth (radians)
`*QPT(I)`	transverse momentum (w.r.t. the beam line)
`*QMSQ2(I,J)`	invariant mass squared of particles I and J
`*QMSQ3(I,J,K)`	invariant mass squared of particles I, J and K
`*QMSQ4(I,J,K,L)`	invariant mass squared of particles I, J, K and L
`*QMSQ5(I,J,K,L,M)`	invariant mass squared of particles I, J, K, L and M
`*QM2(I,J)`	invariant mass of particles I and J
`*QM3(I,J,K)`	invariant mass of particles I, J and K
`*QM4(I,J,K,L)`	invariant mass of particles I, J, K and L
`*QM5(I,J,K,L,M)`	invariant mass of particles I, J, K, L and M
`*QDMSQ(I,J)`	mass squared of the 4-momentum difference P(I) - P(J). In a decay I -> J + x, QDMSQ(I,J) gives the mass squared of x.
`*QBETA(I)`	beta
`*QGAMMA(I)`	gamma
`*QDOT3(I,J)`	scalar product of momentum vectors I and J (3-vectors)
`*QDOT4(I,J)`	scalar product of 4-vectors I and J = QE(I)*QE(J)-QDOT3(I,J)
`*QCOSA(I,J)`	cos(angle between tracks I and J) (lab frame)
`*QPPAR(I,J)`	momentum component of particle I parallel to particle J
`*QPPER(I,J)`	momentum component of particle I perpendicular to particle J
`PTOT(I)`	computes total momentum of "track" I from QX, QY, and QZ
`ENER(I)`	computes energy of "track" I from QP and QM
`XE(P,A,B)`	energy scaled by the beam energy B for a particle of momentum P and mass A
`PFRX(X,A,B)`	momentum of a particle of mass A and scaled energy X when the beam energy is B

4.2 Kinematics and track operations

Routines marked with a ℵ are identical to the corresponding one in ALPHA. Routines marked with a † return values stored on the NanoDST as truncated integers and hence may not have the same precision as in ALPHA. For the number of bits used to code these numbers, please refer to the DDL (appendix A).

4.2.1 Create a new track

`itk=KVNEW(dummy)` ℵ

4.2.2 Create a copy of a track

$\boxed{\texttt{itknew=KVCOPY(itk)}}$ ℵ

4.2.3 Copy a track

$\boxed{\texttt{CALL QCOPY(itkin,itkout)}}$

Copy all attributes of track `itkin` into track `itkout` (i.e. from `itkin` to `itkout`). `itkin` and `itkout` are both input arguments!

4.2.4 Set the mass of a track

$\boxed{\texttt{CALL QVSETM(itk,amass)}}$ ℵ

Only the energy is recomputed.

4.2.5 Add 2 tracks

$\boxed{\texttt{CALL QVADD2(isum,itk1,itk2)}}$

The momentum components and the energy are added, and the total momentum and charge is computed. The jet assignement is taken from track `itk1`. The mass is the invariant mass of the two tracks. `isum` is an input argument!

4.2.6 Add 3 tracks

$\boxed{\texttt{CALL QVADD3(isum,itk1,itk2,itk3)}}$ c.f. `QVADD2`

4.2.7 Add 4 tracks

$\boxed{\texttt{CALL QVADD4(isum,itk1,itk2,itk3,itk4)}}$ c.f. `QVADD2`

4.2.8 Add 5 tracks

$\boxed{\texttt{CALL QVADD5(isum,itk1,itk2,itk3,itk4,itk5)}}$ c.f. `QVADD2`

4.2.9 Copy the momentum 3-vector of a track

```
CALL QVGET3(p,itk)
```

Copy the momentum 3-vector of track `itk` into array `p`.

4.2.10 Copy the momentum 4-vector of a track

```
CALL QVGET4(p,itk)
```

Copy the momentum 4-vector (p_x, p_y, p_z, E) of track `itk` into array `p`.

4.2.11 Set the momentum 3-vector of a track

```
CALL QVSET3(itk,p)
```

Copy the array `p` into the momentum 3-vector of track `itk` and recompute the total momentum and energy with the old mass.

4.2.12 Set the momentum 4-vector of a track

```
CALL QVSET4(itk,p)
```

Copy the array `p` (p_x, p_y, p_z, E) into the momentum 4-vector of track `itk` and recompute the total momentum and mass.

4.2.13 Compute the cosine of the 2-body decay angle

```
angle=QDECA2(i,j)
```

For the 2-body decay M $\rightarrow$ m(`i`) + m(`j`), this function calculates the cosine of the angle between the M line of flight and the m(`i`) momentum vector in the M rest frame. Notice that care is taken of the accuracy (usage of double precision and special grouping of terms).

4.2.14 Compute the cosine of the n-body decay angle

```
angle=QDECAN(i,j)
```

For the decay M(`i`) $\rightarrow$ m(`j`)+...+m(n), this function calculates the cosine of the angle between the M(`i`) momentum vector and the m(`j`) momentum vector in the M(`i`) rest frame.

Chapter 5

OBJECT ATTRIBUTES

Mnemonic symbols marked with a * are identical to the corresponding ones in ALPHA. All variables are stored on the NanoDST as truncated integers and hence may not have the same precision as in ALPHA if a DST is used as input. Compared to a MINI the precision is almost the same. For the number of bits used to code these numbers, please refer to the DDL (appendix A).

5.1 Charged tracks

Only charged tracks satisfying the 'good track' selection criteria are kept on the NanoDST. However, if a charged track is found to be the decay product of a V^0 candidate, it is always kept. In this case, if the track is not a 'good track', then it will be in a special 'bad track' section. When looping over charged tracks from KFCHT to KLCHT, these 'bad tracks' will not be included. They will be included when looping from KFACT to KLACT.

The 'good tracks' are sorted in four groups according to their *charge* and the event *hemisphere* (defined by the thrust axis) in which they lie. They are further sorted within each group by *increasing* momentum. Following is the list of variables and pointers defined for the **CHT** section:

```
KFCHT : First CHarged Track
KLCHT : Last CHarged Track
KFPH1 : First Positive Track in Hemisphere 1
KLPH1 : Last Positive Track in Hemisphere 1
KFNH1 : First Negative Track in Hemisphere 1
KLNH1 : Last Negative Track in Hemisphere 1
KFPH2 : First Positive Track in Hemisphere 2
KLPH2 : Last Positive Track in Hemisphere 2
KFNH2 : First Negative Track in Hemisphere 2
KLNH2 : Last Negative Track in Hemisphere 2
KFBCT : First 'Bad' Charged Track
KLBCT : Last 'Bad' Charged Track
KFACT : First of All Charged Tracks (including 'bad' tracks)
KLACT : Last of All Charged Tracks (including 'bad' tracks)

KNCHT : Number of CHarged Tracks
KNPH1 : Number of Positive Tracks in Hemisphere 1
KNNH1 : Number of Negative Tracks in Hemisphere 1
```

```
KNPH2 : Number of Positive Tracks in Hemisphere 2
KNNH2 : Number of Negative Tracks in Hemisphere 2
KNBCT : Number of 'Bad' Charged Tracks
KNACT : Number of Charged Tracks (All, including 'bad' tracks)
```

5.1.1 Example

We shall illustrate here how to make the most efficient use of the track ordering feature with an example from the D* selection (ALEPH D* group).

A D^{*+} decays into a D^0 plus π^+ (called soft pi), and the D^0 further decays into a K^- and a π^+ (called hard pi). From the kinematics of the $D^{*\pm}$ decay, one can compute an upper and lower bound for the soft pi momentum (P_MAX resp. (P_MIN). A FORTRAN DO loop to select three charged tracks before combining them to see if they are compatible with being the decay products of a $D^{*\pm}$ would then look like:

```
      ...
      DO 30 i_soft_pi = KFPH1,KLPH1
        IF( ...cut3... ) GOTO 30
        IF( QP(i_soft_pi) .LT. P_MIN ) GOTO 30
        IF( QP(i_soft_pi) .GT. P_MAX ) GOTO 40 ! all the following i_soft_pi
                                               ! will have greater momentum
                                               ! and can be safely skipped
        DO 20 i_hard_pi = KFPH1,KLPH1
          IF( ...cut2... ) GOTO 20
          IF(i_hard_pi.EQ.i_soft_pi) GOTO 20
          DO 10 i_kaon = KFNH1,KLNH1
            IF( ...cut1... ) GOTO 10
            ...
            ...here you have 3 charged tracks to compute an invariant mass...
            ...
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
   40 CONTINUE
      ...
```

The standard 'good track' preselection was already performed at the NanoDST production step; cut1, cut2 and cut3 are any further cuts that might be necessary.

This program section is repeated for the four different conditions given by the D* charge and the hemisphere (Monte Carlo studies have shown that there is a negligible efficiency loss due to the splitting into hemispheres). One thereby avoids the tests on the charge of the tracks as well as trying combinations of tracks from opposite hemispheres. Together with the skipping when the next track is above the momentum threshold, these 'tricks' allow a considerable speed-up in the execution time.

5.1.2 Mnemonics

I is the SANDY track number of any charged track.

```
 XFRF(I)    : T if Chi^2 per d.o.f. is available for track I
 XC2OV(I)   : T if Chi^2 per d.o.f. OVERFLOW
 QC2DOF(I)  : Chi^2 per d.o.f.
 XSIG(I)    : T if relative error on momentum is available for track I
 XSIGOV(I)  : T if relative error on momentum OVERFLOW (>5\%)
 QSIGP(I)   : error on momentum
 QRSIGP(I)  : relative error on momentum (QSIGP/QP) (set to 50\% if OVERFLOW)
 KJET(I)    : associated jet (ONLY for standard jets from bank EJET); can be 0!
*QDB(I)     : distance of closest approach to beam axis
*QZB(I)     : z coordinate of track point where QDB is measured
*KTN(I)     : JULIA/GALEPH track number
*KFRTNI(I)  : number of coordinates in ITC
*KFRTNT(I)  : number of coordinates in TPC
 KFRTNV(I)  : number of coordinates in VDET. "3" means "3 or more"
 XVOD(I)    : T if track comes from a V0 candidate
 KVOD(I)    : SANDY number of corresponding V0 candidate
```

5.2 Jet finding

Clustering into jets is computing-time intensive (especially the first steps) and therefore performed during the NanoDST production step. The chosen algorithm is the minimal mass algorithm (QJMMCL — see ALPHA manual). Since jets can be evolved to a higher mass cut at any time, the clustering is done with a small mass cut (4.0 GeV/c^2) to accomodate most analyses.

Two sets of jets are available: the NDEJ bank contains jets obtained by clustering energy flow objects, and the NDJT bank charged tracks only. However, the track – jet relationship is only available for the energy flow jets, and furthermore some tracks may be associated to none of them. It is therefore necessary to always check that `KJET(I)` is not 0 before using it as a pointer to a jet. The NDEJ jets are loaded by default into the **JET** section. To load NDJT instead, **JETB 1** should be added to the cards.

By including **JETC 'new_ mass'** in the cards, the evolution of the chosen jets to the new mass cut is performed automatically when the event is read in. The jet evolution can also be performed at any time by a call to the NEWJET routine (section 5.2.2). Following is the list of variables and pointers defined for the **JET** section:

```
KFJET : First JET
KLJET : Last JET
KNJET : Number of JETs
```

5.2.1 Mnemonics

J is the SANDY "track number" of any jet.

```
KJMUL(J)  : charged track multiplicity in the jet J
```

5.2.2 Jet evolution

```
CALL NEWJET(njets,rmcut)
```

Evolve the jets from section **JET** to a higher mass cut. The old jets are overwritten.

```
Inputs:
     - rmcut /R   : cluster mass cut value ( Ycut = (rmcut/EVIS)**2 )
Outputs:
     - njets /I   : number of jets found or error code if <0

Calls NGJMMC, the improved, generalised Jade algorithm jet finder
with the arguments set to 'E' scheme and normal JADE algorithm
for SANDY (interface to FJMMCL).
```

5.2.3 Scaled invariant mass squared algorithm

```
CALL NJMMCL(njets,kfi,kli,kfo,klo,rmcut,evis)
```

Jet finder using the scaled invariant mass squared algorithm. Any set of objects can be fed in.

```
Inputs:
     - kfi   /I   : SANDY track number of first input object
     - kli   /I   : SANDY track number of last input object
     - rmcut /R   : cluster mass cut value ( Ycut = (rmcut/EVIS)**2 )
     - evis  /R   : visible energy for normalisation
                    (if EVIS=0., it is computed from the
                    input particle energies)
Outputs:
     - njets /I   : number of jets found or error code if <0
     - kfo   /I   : SANDY track number of first output object
     - klo   /I   : SANDY track number of last output object

Calls NGJMMC, the improved, generalised Jade algorithm jet finder
with the arguments set to 'E' scheme and normal JADE algorithm
for SANDY (interface to FJMMCL).
```

5.3 Photons

Photons from the PGPC bank (found by GAMPEX) are stored on the NanoDST in the NDPH bank, and loaded into the **GAM** section. Notice that when there were more than 4 photons in a PECO cluster, only the first 4 were kept and stored. Following is the list of variables and pointers defined for the **GAM** section:

```
KFGAM : First GAMma
KLGAM : Last GAMma
KFGH1 : First GAMma in Hemisphere 1
KLGH1 : Last GAMma in Hemisphere 1
KFGH2 : First GAMma in Hemisphere 2
KLGH2 : Last GAMma in Hemisphere 2

KNGAM : Number of GAMmas (in bank NDPH)
KNGH1 : Number of GAMmas in Hemisphere 1
KNGH2 : Number of GAMmas in Hemisphere 2
```

5.3.1 Mnemonics

I is the SANDY track number of any photon.

```
KGREG(I)   : region code  1 = barrel
                          2 = endcap
                          3 = overlap
                          0 = crack or dead storey(s)
KGMUL(I)   : number of photons in the PECO cluster 1,2 or 3. 0 if >=4
KGPECO(I)  : PECO number
             This is useful for scanning, or to know which photons belong
             to the same PECO cluster.
XGE1(I)    : T if energy fraction in stack 1 > 0
             WARNING: only for versions >= 111.3
XGE2(I)    : T if energy fraction in stack 2 > fraction in stack 1 or
               if energy fraction in stack 2 > fraction in stack 3
             WARNING: only for versions >= 111.3
```

5.3.2 π^0 finder QPI0DO

```
CALL QPIODO
```

The π^0 finder QPI0DO (J.-P. Lees) is fully implemented in SANDY. It builds π^0 candidates from GAMPEX photons taken in the GAM section of SANDY, and refits their energy-momentum applying a π^0 mass constraint $2 \cdot w_1 \cdot w_2(1 - \cos\theta_{12}) = m_{\pi^0}$. The error on the angles of the two photons is neglected and one finds, with a Lagrange multiplier approximate solution, the refitted energies w_1, w_2 minimizing $\chi^2 = \left(\frac{E_1-w_1}{S_1}\right)^2 + \left(\frac{E_2-w_2}{S_2}\right)^2$ with w_1, w_2 the refitted energies, E_1, E_2 the measured energies and S_1, S_2 the error on these energies.

The user needs only to call the routine QPI0DO (no arguments), and the results will be filled into the **GAMPI0** COMMON. However, the routines XXP0F1 (G. Batignani) or KINEFIT (M. Maggi) can be used in a standalone mode to refit the π^0 momentum using the π^0 mass constraint (see the subroutine header for the description of the argument list).

```
  Description of common GAMPI0:
  -----------------------------
*CD GAMPI0
      PARAMETER(MXPI0=200)
      COMMON/GAMPI0/IQPI0, NTPI0, PI0MOM(4,MXPI0),ITYPI0(MXPI0),
     +              IPI0GAM(2,MXPI0),CHIPI0(MXPI0)

  IQPI0             : return code  0-->OK, 1-->0 pi0, 2-->N pi0>MXPI0
  PI0MOM(4,MXPI0)   : PI0 refitted 4 momentum
  IPI0GAM(2,MXPI0): gam 1 & 2 number in the GAT section
  CHIPI0(MXPI0)     : chi2 value after refit (-999. if no convergence)
  ITYPI0(MXPI0)     : pi0 type, see below

  Description of pi0 types:
  -------------------------
   TY=1: 2 photons in same PECO, with N=2 photons in the PECO
   TY=2: 2 photons in same PECO, with N>2 photons in the PECO
   TY=3: 2 photons in different PECO, with N=1 photons in each PECO
   TY=4: 2 photons in different PECO, with N>1 photons in one PECO
```

Book QPI0DO histograms

$\boxed{\text{CALL QPI0BK}}$

Books some control histograms which will then be filled by QPI0DO.

Print π^0 candidates

$\boxed{\text{CALL PI0DEB}}$

Prints the π^0 candidates found by QPI0DO.

5.4 V^0

V^0 candidates found by the standard V^0 - finder (M.A. Ciocci, L. Rolandi) are stored on the NanoDST in the NDV0 bank and loaded into the **V0T** section. Note that some cuts to the V^0

candidates are applied before putting them on the NANO DST.

Following is the list of variables and pointers defined for the **V0T** section:

```
KFVOT : First VO candidate      (was KFDVO before)
KLVOT : Last VO candidate       (was KLDVO before)
KNVOT : Number of VO candidates (was KNDVO before)
```

5.4.1 Mnemonics

I is the SANDY track number of any V0 candidate.

```
KVOHYP(I) : Fit hypothesis (bit 0 set:K0,1:Lambda,2:AntiLambda,3:Gamma)
KVONTN(I) : FRFT track number of positive VO daughter track
KVOPTN(I) : FRFT track number of negative VO daughter track
KVOPOT(I) : SANDY track number of positive VO daughter track
KVONEG(I) : SANDY track number of negative VO daughter track
KVOIC(I)  : IcodevO from bank YVOV (see there for desription)
QVOCH(I)  : Chi square of fit
QVOVM(I)  : Fitted VO momentum
QVOTH(I)  : Polar angle of VO direction
QVOPH(I)  : Azimuthal angle of VO direction
QVOVTX(I) : x-coordinate of VO decay vertex
QVOVTY(I) : y-coordinate of VO decay vertex
QVOVTZ(I) : z-coordinate of VO decay vertex
QVODL(I)  : Decay length
QVOEDL(I) : Error on decay length
XVOSVA(I) : T if ambiguous secondary vertex
XVOKIA(I) : T if ambiguous with another hypothesis
XVOTRA(I) : T if VO shares a track with another VO
XKO(I)    : T if KO hypothesis bit TRUE
XLA(I)    : T if Lambda hypothesis bit TRUE
XAL(I)    : T if Antilambda hypothesis bit TRUE
XGA(I)    : T if Gamma hypothesis bit TRUE
```

Attributes of positive daughter from V^0:

```
QVOPP(I)  : Fitted momentum at decay vertex
QVOPTH(I) : Fitted polar angle at decay vertex
QVOPPH(I) : Fitted azimuthal angle at decay vertex
QVOPPX(I) : Fitted x component of momentum at decay vertex
QVOPPY(I) : Fitted y component of momentum at decay vertex
QVOPPZ(I) : Fitted z component of momentum at decay vertex
QVOPCH(I) : Chi square increase from QVOCHK
```

Attributes of negative daughter from V^0:

```
QVONP(I)  : Fitted momentum at decay vertex
```

QVONTH(I)	: Fitted polar angle at decay vertex
QVONPH(I)	: Fitted azimuthal angle at decay vertex
QVONPX(I)	: Fitted x component of momentum at decay vertex
QVONPY(I)	: Fitted y component of momentum at decay vertex
QVONPZ(I)	: Fitted z component of momentum at decay vertex
QVONCH(I)	: Chi square increase from QVOCHK for negative track

I is the SANDY track number of any charged track.

XVOD(I)	: T if track comes from a V0 candidate
KVOD(I)	: SANDY number of corresponding V0 candidate (or YRFT row)

5.5 Converted Photons

Gamma Conversions are stored in the NDGC bank and loaded into the **GCO** section. Two different routines have been used to find gamma conversions for the NanoDst. One of them is the standard pair finder QPAIRF, the other one is the new routine QACONV by S. Schael (ALEPH note 94-104). The advantage of the latter one is that it reconstructs converted photons where only one track is found in addition to pairs. To allow comparisons between both routines where a photon decays into two 'visible' tracks none of the conversions are removed from the NanoDst. Therefore a pair of tracks forming a gamma can be twice within the NDGC bank if it was found by both routines. So the user has to take care that these photons are used only once in an analysis.

Following is the list of variables and pointers defined for the **GCO** section:

KFGCO	: First converted Photon
KLGCO	: Last converted Photon
KNGCO	: Number of converted Photons

5.5.1 Mnemonics

I is the SANDY track number of any converted Photon candidate.

```
KGCTYP(I) : Type and origin of converted gamma:
                0: from QPAIRF
            from routine QACONV:
                1: from QFNDVO
                2: from QPAIRF
                3: single electron
```

Remember: Track numbers may be 0 if KGCTYP(I).eq.3

KGCPTN(I)	: JULIA track number of positive daughter
KGCNTN(I)	: JULIA track number of negative daughter
KGCPOT(I)	: SANDY track number of positive daughter

```
KGCNET(I) : SANDY track number of negative daughter
QGCVX(I)  : X - position of conversion point
QGCVY(I)  : Y - position of conversion point
QGCVZ(I)  : Z - position of conversion point
```

The following quantities only exist if KGCTYP(I).eq.0:

```
QGCDXY(I) : Distance in xy - plane between tracks at conversion point
QGCDZ(I)  : Distance in z - direction between tracks at conversion point
```

5.6 Monte Carlo

Monte Carlo truth particles are divided into two categories according to the value of the ALPHA stability code (KSTABC):

- *unstable* particle (NDNT bank): if KSTABC is -1 or -2
- *stable* particle (NDMS bank): for any other value of KSTABC

The momentum of an *unstable* particle is not stored, as it can be reconstructed from the momenta of its decay products. However, because of technical details specific to the different generators, this procedure does not work for the partons and the initial quarks, which are therefore also stored as *stable* particles (in the NDMS bank). Pointers are defined for the 'final' partons (at the end of the parton shower).

For V^0 particles, the 'true' decay vertex is stored on the NanoDST in the NDLV bank.

Following is the list of variables and pointers defined for the **MCT** section:

```
KFMCT : First Monte Carlo Track
KLMCT : Last Monte Carlo Track
KFMUT : First Monte Carlo Unstable Track
KLMUT : Last Monte Carlo Unstable Track
KFMST : First Monte Carlo Stable Track
KLMST : Last Monte Carlo Stable Track

KQ    : initial Quark (direct daugther of the Z0)
KQBAR : initial anti-Quark (direct daugther of the Z0)

KFPAR : First 'final' parton (not correct for HERWIG)
KLPAR : Last 'final' parton (not correct for HERWIG)

KNMCT : Number of Monte Carlo Tracks
KNMUT : Number of Monte Carlo Unstable Tracks (in bank NDNT)
KNMST : Number of Monte Carlo Stable Tracks (in bank NDMS)
```

5.6.1 Mnemonics

I is the SANDY track number of any Monte Carlo truth track.

```
 XMCNE(I)   : T if this MC "track" is a neutral particle
 XMCCH(I)   : T if this MC "track" is a charged particle
 XMCGA(I)   : T if this MC "track" is a gamma
 KPTCH(I)   : bank containing the matched track
              (0=neutral,1=photon,2=charged,-1=unstable)
*KTPCOD(I)  : ALEPH particle code
*KNDAU(I)   : number of daughters of track I
*KDAU(I,J)  : SANDY number of the Jth daughter of track I
*KNMOTH(I)  : number of mothers of track I
 KMOTH(I)   : SANDY number of the mother of track I
*KSTABC(I)  : stability code (see ALPHA manual)
              WARNING: only for versions >= 111.3
 QVOVX(I)   : X coordinate of V0 decay vertex (for true MC V0 only)
 QVOVY(I)   : Y coordinate of V0 decay vertex (for true MC V0 only)
 QVOVZ(I)   : Z coordinate of V0 decay vertex (for true MC V0 only)
```

5.6.2 Matched track

`match=KMTCH(itk)`

Returns the SANDY reconstructed track number which matches the Monte Carlo truth track `itk` best. The matching is done at the NanoDST production step. For charged tracks, at least 5 shared hits are required, and then the closest track (euclidian distance in 3–momentum space) is kept. For the photons, only distance (euclidian distance in 3–momentum space) is used, and the closest reconstructed photon is kept. One can therefore have many Monte Carlo truth photons matched to the same reconstructed photon, but it does not mean that all these links are meaningful; it is left to the user to decide upon a maximal distance cut.

5.6.3 *z* of heavy quark fragmentation

`z=QZFR(itk)` †

For a Monte Carlo truth track `itk` originating from a heavy quark decay, returns the z value used for the Peterson fragmentation function. The returned value is 0 for any other Monte Carlo truth track.

5.6.4 Descendants of a MC track

`CALL DECSEA(mothr,nd,ns,itks)`

Find all Monte Carlo truth tracks itks that come from mothr, including all intermediate states to the 'final' particles.

```
Inputs:
     - mothr/I     : SANDY track number of mother
Outputs:
     - nd     /I     : number of [[grand]^n -]daughters
     - ns     /I     : number of 'stable' particles
     - itks   /I     : 2-dim array with
                       (1,i) the SANDY track number of the
                             [[grand]^n -]daughter
                       (2,i) the generation (mother is generation 0)
                             Negative generation means 'stable'
                             particle
                       The array starts at generation 1.

For all tracks in the list of MC track, make an iterative search
up the parenthood tree until we find the [[grand]^n -]mother
mothr or 0. If the end result is 0, then that particle does not
originate from mothr.
Does NOT count particles from decays of 'usually stable' particles
such as pions, kaons and protons (i.e. if the mother of a particle
is a charged pion, charged kaon, or proton, it will not be
considered as a daughter).
A 'stable' particle means: the particle has 0 daughters or it is
an electron, muon, charged pion, charged kaon or proton.
Anti-particles are always implied.
```

5.6.5 Decay tree of a MC track

$\boxed{\texttt{CALL PRIDEC(itk)}}$

Print the decay tree of MC particle itk.

```
Inputs:
     - itk    /I     : SANDY track number
Outputs:
     - none

First call DECSEA to get all the descendants, then print the
decay tree in the log file. A # means 'stable' particle (see
description of routine DECSEA).
```

5.6.6 Descendants of given type

```
CALL DAUSEA(mothr,kode,nd,itks)
```

Find all Monte Carlo truth tracks `itks` of type `kode` that come from `mothr`. The routine will find all descendants of the given type, regardless how many intermediate states or resonances there are in between.

```
Inputs:
     - mothr /I    : SANDY track number of mother
     - kode  /I    : particle code
Outputs:
     - nd    /I    : number of [[grand]^n -]daughters of type kode
     - itks  /I    : array with the SANDY track number of the
                     [[grand]^n -]daughters

For all tracks of type kode in the list of MC track, make an
iterative search up the parenthood tree until we find the
[[grand]^n -]mother mothr or 0. If the end result is 0, then that
particle does not originate from mothr.
Does NOT count particles from decays of 'usually stable' particles
such as pions, kaons and protons (i.e. if the mother of a particle
is a charged pion, charged kaon, or proton, it will not be
considered).
```

5.7 dE/dx analysis

dE/dx information for charged tracks is stored on the NanoDST in the NDDE bank. This information can be accessed directly with macros. However, it is recommended to use the NDEDX and NDEDXM routines for the analysis.

WARNING: if the relative error on the momentum of a charged track is greater than 5%, it is stored as an overflow at the NanoDST production step and will be set to 50% when reading the NanoDST with SANDY. This can influence the sigma calculation, which may be overestimated for some high momentum tracks.

5.7.1 Mnemonics

I is the SANDY track number of any charged track.

```
*XTEX(I)    : T if dE/dx information available for track I
 KITL(I)    : useful track length in [mm]
 KINS(I)    : number of useful wire samples on track
 XRIMOV(I)  : measured ionisation OVERFLOW
 QRIMES(I)  : measured ionisation (measured and calibrated, except TC3X)
```

```
XRSIOV(I) : Relative error on the dE/dx OVERFLOW
QRSIG(I)  : Relative error on the dE/dx
            The error to be used in analysis should be calculated from:
            SIGMA**2= (RSIG*Iexp)**2 + SIG_P**2
            where Iexp is the expected ionization for a given hypothesis,
            and SIG_P is the contribution from momentum error.
```

5.7.2 dE/dx analysis of a charged track

$$\boxed{\texttt{CALL NDEDX(itk,n,rmass,q,ri,ns,tl,riexp,sigma,ier)}}$$

NB: this routine is a modified version of QDEDX, adapted for SANDY.

```
 Input arguments:
itk          SANDY track number.
n            Number of hypotheses the user wishes to try.
rmass(n)     Array of masses, one for each hypothesis.
q(n)         Array of charges, one for each hypothesis.
 Output arguments:
ri           The measured truncated mean ionisation, normalised and
             calibrated.
ns           Number of useful wire samples on the track.
tl           Useful track length [cm].
riexp(n)     Expected ionisation for each mass hypothesis, normalised
sigma(n)     One standard deviation resolution error for each hypothesis.
             This is the expected dE/dx resolution, given ns,tl,riexp, and
             the momentum resolution.
             NB: one can calculate a Chi^2 with 1 d.o.f. as:
                 Chi^2 = ((ri-riexp)/sigma)
ier          Error return code: 0 = successful completion
                                1 = not a good track
                                2 = can't find dE/dx bank
                                3 = track has no dE/dx information
                                4 = can't find calibration banks
                                    TC1X, TC2X, and/or TC3X
                                5 = cannot find RUNH or EVEH bank
                                    from which to get the run number
```

5.7.3 Modified NDEDX for Monte Carlo

$$\boxed{\texttt{CALL NDEDXM(itk,n,rmass,q,ri,ns,tl,riexp,sigma,ier)}}$$

NB: this routine is a modified version of QDEDXM, adapted for SANDY.

Analyse dE/dx for Monte Carlo events by faking the ionization with a gaussian random number. If called for real data, the result will be the same as if NDEDX were called.

```
 Input arguments:
itk           SANDY track number.
n             Number of hypotheses the user wishes to try.
rmass(n)      Array of masses, one for each hypothesis.
q(n)          Array of charges, one for each hypothesis.
 Output arguments:
ri            The measured truncated mean ionisation, normalised and
              calibrated.
ns            Number of useful wire samples on the track.
tl            Useful length of the track [cm].
riexp(n)      Expected ionisation for each mass hypothesis, normalised
sigma(n)      One standard deviation resolution error for each hypothesis.
              This is the expected dE/dx resolution, given ns,tl,riexp, and
              the momentum resolution.
              NB: one can calculate a Chi^2 with 1 d.o.f. as:
                  Chi^2 = ((ri-riexp)/sigma)
ier           Error return code: 0 = successful completion
                                 1 = not a good track
                                 2 = can't find dE/dx bank
                                 3 = track has no dE/dx information
                                 4 = can't find calibration banks
                                     TC1X, TC2X, and/or TC3X
                                 5 = cannot find RUNH or EVEH bank
                                     from which to get the run number
                                 6 = No MC truth
```

5.8 Lepton identification

Electrons and Muons are *heavy flavor* leptons selected with the LEPTAG package (M. Parsons and I. ten Have) and put in the NDBM bank. Calling LEPTAG from inside SANDY will create the full BMLT bank. This is only neccessary if the weights are needed, all other quantities are in the NDBM bank and can be accessed via the following mnemonics.

5.8.1 Mnemonics

I is the SANDY track number of a charged track where either `XEL(I)` or `XMU(I)` are TRUE.

```
KLETYP(I) : Lepton type:
              2 => e+
             12 => e+ in crack region
             22 => e+ in overlap region
              3 => e-
             13 => e- in crack region
             23 => e- in overlap region
              5 => mu+
               6 => mu-
QLEPTI(I) : Transverse momentum lepton inclusive
QLEPTE(I) : Transverse momentum lepton exclusive
XLEMU3(I) : T if Muon IDF 13
XLEMU4(I) : T if Muon IDF 14
XLEGEL(I) : T if Genuine ELectron/positron
```

If the event is a Monte Carlo event (XMCEV true), then the following pointers are although set:

```
KLEPRQ(I) : PRimary Quark flavour from FINLEP. This should be the same as
            KFLAV.
KLEDCA(I) : Decay CAtegory from FINLEP
                1 => b -> mu + charmed hadrons
                2 => b -> mu + non charmed hadrons
                3 => b -> tau -> mu
                4 => b -> c -> mu
                5 => b -> cbar -> mu
                6 => c -> mu
                7 => c -> tau -> mu
                8 => b -> c -> tau -> mu
                9 => K -> mu or pi -> mu
               10 => gamma -> mu
               11 => J/psi -> mu
               12 => psi' -> mu
               13 => other decays to muon
               14 => tau decay
               15 => muon from other sources
               16 => misidentified hadron
               17 => muon -> electron
               18 => others
               19 => error in finding some mother
               20 => assoc. Kingal track not found
            21-35 => as 1-15 but e instead
           negtve => parent quark is from a gluon
```

5.8.2 Tagging heavy flavour leptons

```
CALL LEPTAG(LDEBUG,LTOUT,IERR)
```

The Lepton tagging Package from M. Parsons and I. ten Have. Compared to the version on UPHY the input arguments are different as some are useless because inside SANDY LEPTAG just unpacks the NDBM bank and calls CALPOIDS to get the source weights for different kinds of lepton decay chains before storing all the information in the BMLT bank. In the case these weights are not needed it is not neccessary to run LEPTAG.

```
Inputs:
-------
```

```
          LOGICAL LDEBUG   = Controls whether or not debugging information
                             should be written out (.TRUE.) or not (.FALSE.).
          INTEGER LTOUT    = Unit number on which debug and leptag error
                             information is to be written.
Outputs:
--------
          INTEGER IERR     = Completion return code defined as follows:

     IERR > 0      ! The number of leptons found and stored in the BMLT bank.
     IERR = 0      ! No suitable leptons found in event.
```

and BMLT bank.

5.9 Particle identification

Electrons and muons are *heavy flavor* leptons. Pions, kaons, and protons are identified with a dE/dx cut only.

5.9.1 Mnemonics

I is the SANDY track number of any charged track.

```
 NXPAHY(IHYP): T if particle hypothesis bit set for hypothesis IHYP
               e.g. if NXPAHY(3)=.FALSE. then XPI(I) has NO meaning!
    IHYP = 1 : electron hypothesis
           2 : muon hypothesis
           3 : pion hypothesis
           4 : kaon hypothesis
           5 : proton hypothesis
 XPART(I,J): T if particle hypothesis bit J is TRUE for track I
             J = 1 : electron
             J = 2 : muon
             J = 3 : pion
             J = 4 : kaon
             J = 5 : proton
 XEL(I)    : T if electron hypothesis bit TRUE for track I
 XMU(I)    : T if muon hypothesis bit TRUE for track I
 XPI(I)    : T if pion hypothesis bit TRUE for track I
 XKA(I)    : T if kaon hypothesis bit TRUE for track I
 XPR(I)    : T if proton hypothesis bit TRUE for track I
```

5.10 Thrust

The thrust value and axis of the event, calculated with the energy flow objects, are directly available:

```
QTHRU = THRUST value (scalar)
QTTHE = theta of THRUST axis
QTPHI = phi of THRUST axis
KTHRU = pointer to THRUST vector (has module equal to THRUST value)
        NB_1: ONLY QX, QY, QZ and QP are defined for the thrust vector.
        NB_2: The angles theta and phi (and hence the axis) are integerised
              and stored with a precision of 10 [mrad].
```

5.11 Particle properties

```
*KPART('part-name')          integer particle code for 'part-name'
*CQPART(intg-code)           particle name (12 characters; trailing characters
                             filled with blank spaces)
*KPANTI('part-name',IANTI)   if IANTI=0 : integer code for 'part-name'
                             if IANTI unequal to 0 : integer code for the
                                                     antiparticle of 'part-name'
*KCANTI(intg-code,IANTI)     ...
*QPMASS('part-name')         nominal mass
*QCMASS(intg-code)           ...
*QPCHAR('part-name')         charge
*QCCHAR(intg-code)           ...
*QPLIFE('part-name')         life time
*QCLIFE(intg-code)           ...
*QPWIDT('part-name')         width
*QCWIDT(intg-code)           ...
```

5.12 b-tagging

Apart from the event shape b- tagging algorithm outputs the QIPBTAG routine can now be used inside SANDY. The routine is called in the usual way but internally it just fills information from the NBIP bank into its output variables. Therfore the commonblock BTAGRAW is empty at the end and other parameters given by JETF, TRA2 or VCUT cards have no effect. The only card that can be used is the FITP card. For more details see ALEPH note 92-135.

```
CALL QIPBTAG(IRET,NTRACK,NJET,TRKJET,FRF2TRK,PROBTRK,PROBJET,
             PROBHEMI,PROBEVT)
```

```
INPUT:
   none
```

```
OUTPUT:
   iret :            >0  didn't find NBIP bank
                    = 0  else = O.K.


   ntrack          (I) = # of tracks used to calculate PROBJET/HEMI/EVT
   njet :          (I) = # of jet found
   TRKJET(njet)    (I) = empty on NanoDst
   NDTKTRK(ntrack)(I) = NDTK row number of tracks used for analysis
   probtrk(ntrack)(R) = array(ntrack) with probabilities of each track
                          with indices as in NDTKTRK
   probjet(njet)   (R) = array(njet) with probabilities of each jet
                          with indices as in TRKJET
   probhemi(2)     (R) = array(2)    with probabilities of hemisphere
                         hemisphere(1) is defined by the leading jet
   probevt         (R) = probability of the event

    -- The parameters for this routine are given
       by the BOS cards 'FITP'
```

Chapter 6

SANDY UTILITY ROUTINES

6.1 Print a message

$\boxed{\texttt{CALL QWMESS('any message')}}$ $\aleph$

6.2 Print a message plus run, event number

$\boxed{\texttt{CALL QWMESE('any message')}}$ $\aleph$

6.3 Event output

$\boxed{\texttt{CALL NWRITE}}$

You have to specify the name of the output file in a FILO card.

6.4 Print an event

$\boxed{\texttt{CALL NPRINT}}$

Makes a printout of the current event in the log file. Prints also the Monte Carlo truth if applicable.

6.5 Produce a SEVT card for this event

$\boxed{\texttt{CALL MAKSEV}}$

At the end of the job, all SEVT cards are automatically written out to a file if this routine was called at least once. You can specify the filename with the NSEL card, or it will be called

SEVT.CARDS by default. The maximum number of selected event is 10000. WARNING: do *not* call this routine more than once per event!

6.6 Print the NanoDST production parameters

```
CALL NINFO
```

Make a printout of all NanoDST production parameters in the log file, as well as of the RHAH bank of the current run. The production parameters are available also as variables.

```
ECHMIN: MINimum CHarged Energy to accept the event

MINTRK: MINimum number of charged TRacKs
PCUTMI: lower momentum CUT for charged tracks
PCUTMA: upper momentum CUT for charged tracks
THCCUT: Cosine THeta CUT for charged tracks
NITCUT: minimum number of ITC coordinates for charged tracks
NTPCUT: minimum number of TPC coordinates for charged tracks
BDOCUT: upper DO CUT for charged tracks
BZOCUT: upper ZO CUT for charged tracks

PCUMIJ: lower momentum CUT for charged tracks before clustering
PCUMAJ: upper momentum CUT for charged tracks before clustering
THCCUJ: Cosine THeta CUT for charged tracks before clustering
NITCUJ: minimum number of ITC coordinates for charged tracks before clustering
NTPCUJ: minimum number of TPC coordinates for charged tracks before clustering
BDOCUJ: upper DO CUT for charged tracks before clustering
BZOCUJ: upper ZO CUT for charged tracks before clustering
BMYCUT: mass cut for jet algorithm QJMMCL (on the chosen RECO)
ENEVIS: VISible ENErgy for jet algorithm QJMMCL
RECOOP: REConstructed Objects OPtion (available opt.: 'RE','CO','CH')
VOCHSI: minimum CHi square increase of SIngle track from VO fitting back to
        the main vertex.
VOCHBO: minimum CHi square increase of BOth tracks from VO fitting back to
        the main vertex.
PHOCUT: lower energy CUT for PHOtons

REFMIN: MINimum energy for Energy Flow objects
RMYCUT: mass cut for jet algorithm QJMMCL (on Energy Flow objects)

ALIVER: Alephlib version
NCPROD: Version of NDSTPROD
NCALPH: ALPHa version and correction file used for production
NCDATE: DATE of the production
MCOPTI: options used for Monte-Carlo production
        bit  0 set if real data
bit  1 = ISEL : gluons selection ( 1 = keep gluons )
bit  2 = ISIN : flag for using the single particle mode
                     = 1 read the following flags
                     = 0 ignore the following flags
                When you select a single particle, only that
                final particle and the quark will be kept
                in the NDMC bank, for those events of the
                chosen flavor. For the flavors where the
                flag is 0, all tracks are kept.
bit  3 = IPI  : pion
bit  4 = IKA  : kaon
bit  5 = IMU  : muon
bit  6 = IEL  : electron
bit  7 = INI  : neutrino
bit  8 = IGA  : gamma
bit  9 = IPR  : proton
bit 10 = INE  : neutron
bit 11 = IUQ  : u quark
bit 12 = IDQ  : d quark
bit 13 = ISQ  : s quark
bit 14 = ICQ  : c quark
bit 15 = IBQ  : b quark
bit 16 = ITQ  : t quark
PMINMC: minimum momentum for MC tracks in the fragmentation
        (-1. for real data)
```

Appendix A

BANK DEFINITIONS (DDL)

Following is the 'LBF' output listing for the banks used in the SANDY NanoDST package.

```
                                                    Subschema: NANODST
==============================================================

*------*
| NDAR |  Nano Dst particle bank (from pARt)
*------*
..............................................................
          1      I   Number of Columns (=7)
          2      I   Number of particles
..............................................................
          1  PA  I   bit  0- 7 : Geant number     [*,*]
                     bit  8-15 : Geant tracking code
                                  1 = Photons
                                  2 = Electrons
                                  3 = Neutr. Hadrons +
                                  Neutrinos
                                  4 = Charged Hadrons
                                  5 = Muons
                                  6 = Geantinos
                                100 = Not tracked particle
                     bit 16-20 charge
                     bit 21-30 ANtiparticle
                               Corresponding antipart number
        2-4  NA  I   particle names
          5  MA  I   particle mass                     [*,*]
                     bit  0- 8: exponent
                          9-32: mantissa
          6  LT  I   particle lifetime                 [*,*]
                     bit  0- 8: exponent
                          9-32: mantissa
          7  MW  I   particle width                    [*,*]
                     bit  0- 8: exponent
                          9-32: mantissa
==============================================================
*-------*
| NDBM |  Nano Dst leptag output bank (from BMlt)
*-------*
..................................................................
            1      I    Number of Columns (=4)
            2      I    Number of leptons
..................................................................
            1  TA  I    bit  0- 7 : NANO track number    [*,*]
                                    max (255)
                        bit  8-15 : Particle type:
                          2 => e+
                         12 => e+ in crack region
                         22 => e+ in overlap region
                          3 => e-
                         13 => e- in crack region
                         23 => e- in overlap region
                          5 => mu+
                          6 => mu-
                        bit 16-23 : Pointer to jet in
                                    jet section
                        bit 24-31 : IDF/Truth flag
                           Bit 24 : Muon IDF 13
                           Bit 25 : Muon IDF 14
                           Bit 26 : Genuine electron/
                                            positron
            2  PI  I    Transverse momentum  (keV)        [0,*]
                           lepton inclusive
            3  PE  I    Transverse momentum  (keV)        [0,*]
                           lepton exclusive
            4  DA  I    bit  0- 3 :                       [*,*]
                           Primary quark flavour from FINLEP
                          0 => not a MC q-qbar event
                          1 => d quark
                          2 => u quark
                          3 => s quark
                          4 => c quark
                          5 => b quark
                        bit  4-11 :
                           Decay category from FINLEP
                          1 => b -> mu + charmed hadrons
                          2 => b -> mu + non charmed hadrons
                          3 => b -> tau -> mu
                          4 => b -> c -> mu
                          5 => b -> cbar -> mu
                          6 => c -> mu
                          7 => c -> tau -> mu
                          8 => b -> c -> tau -> mu
```

```
                     9 => K -> mu or pi -> mu
                    10 => gamma -> mu
                    11 => J/psi -> mu
                    12 => psi' -> mu
                    13 => other decays to muon
                    14 => tau decay
                    15 => muon from other sources
                    16 => misidentified hadron
                    17 => muon -> electron
                    18 => others
                    19 => error in finding some mother
                    20 => assoc. Kingal track not found
                 21-35 => as 1-15 but e instead
                negtve => parent quark is from a gluon
                  bit 12-21 : Code of the lepton
                              parent
                  bit 22-31 : Energy flow object
                              number
==============================================================

+------+
| NDDE |  Nano Dst DE/dx
+------+
..............................................................
    1          I    Number of Columns (=2)
    2          I    Number of selected tracks
                    with dE/dx info
..............................................................
    1   LI  I    LenandIon        [*,*]
                    Track length and ionisation
                    bit 0 -11 Track length in [mm]
                    bit 12-31 Measured truncated mean
                    ......... ionisation * 100000
    2   SE  I    SampandErr       [*,*]
                    Samples and relative error
                    bit 0 - 8 Number of samples
                    bit 9 -31 Relative error on the
                    ......... dE/dx * 100000
==============================================================

+------+
| NDEJ |  Nano Dst reconstructed Eflow Jets
+------+
..............................................................
    1          I    Number of Columns (=4)
    2          I    Number of reconstructed jets
..............................................................
    1   PX  I    PX (MeV)         [*,*]
```

```
                    Momentum X component
  2   PY  I     PY (MeV)         [*,*]
                    Momentum Y component
  3   PZ  I     PZ (MeV)         [*,*]
                    Momentum Z component
  4   EJ  I     EnergyofJet(MeV) [0,*]
                    Energy of the Jet
                    ========================
                    CHANGES vs version 111.4 :
                    integerized and was EJET
                    before
==========================================================


+------+
| NDGC |  NDST Gamma-Conversion-Bank
+------+
.....................................................................
  1          I     Number of Columns (=10)
  2          I     Number of selected Gamma Conversion candidates
.....................................................................
  1   DA   I    DAughter tracks             [*,*]
                   bit  0-7    nano track number of positron candidate
                   bit  8-15   nano track number of electron candidate
                   bit 16-17   origin of conversion:
                               0 : from QPAIRF
                               1 : from QACONV
                               2 : from QACONV
                               3 : from QACONV
                   bit 18-31   free
  2   DX   I    DXy                         [*,*]
                   distance between the two tracks in the xy-plane
                   at the closest approach to the conversion point
                   (microns)
  3   DZ   I    DZ2                         [*,*]
                   z separation of the tracks at the closest approach
                   to the conversion point (microns)
  4   XM   I    XMa                         [*,*]
                   invariant mass of the tracks at the conversion
                   point assuming they are both electrons (keV)
  5   CX   I    Conversion coordinate (X)   [*,*]
                   x-component of the conversion point (microns)
  6   CY   I    Conversion coordinate (Y)   [*,*]
                   y-component of the conversion point (microns)
  7   CZ   I    Conversion coordinate (Z)   [*,*]
                   z-component of the conversion point (microns)
  8   PX   I    Photon momentum (X)         [*,*]
                   x-component of the momentum of the
                   gamma candidate (MeV)
  9   PY   I    Photon momentum (Y)        [*,*]
                   y-component of the momentum of the
                   gamma candidate (MeV)
 10   PZ   I    Photon momentum (Z)        [*,*]
                   z-component of the momentum of the
                   gamma candidate (MeV)
==============================================================


+------+
| NDHE |   Nano Dst event HEader
+------+
..............................................................
    1          I     Number of Columns (=18)
    2          I     Number of rows
..............................................................
    1   KR  I    KRun              [0,*]
                   Run number
    2   KE  I    KEvt              [0,*]
                   Event number
    3   TR  I    ThRust            [*,*]
                   * 1 000 000
                   Thrust
    4   TP  I    TrkPointer        [*,*]
                   Pointers to track groups
                   bit 0 - 7 First photon in hemisphere 2
                   bit 8 -15 First negative track in
                   ......... hemisphere 1
                   bit 16-23 First positive track in
                   ......... hemisphere 2
                   bit 24-31 First negative track in
                   ......... hemisphere 2
    5   PF  I    PhysicsFlag       [*,*]
                   Flag
                   bit 0 ... XLUMOK
                   bit 1 ... XMCEV
                   bit 2 - 5 Event flavor
                   ......... (0 for real data)
                   bit 6 -14 Theta of thrust axis in
                   ......... radians * 100 [0,3.14]
                   bit 15-24 Phi of thrustaxis in
                   ......... radians * 100 [0,6.28]
                   bit 25 .. 1 = ITC and TPC voltage OK
                   bit 26 .. 1 = TPC dE/dx voltage and
                   ............. calibration OK
                   bit 27 .. 1 = VDET data OK
                   bit 28 .. 0 = Error in QMUIDO/MUREDO
                   bit 29 .. 0 = Error in LEPTAG
                   bit 30 .. 0 = Error in JETSPH
```

```
                 ............. (charged tracks only)
                 bit 31 .. 0 = Error in JETSPH
                 ............. (ENFW objects)
6   SC  I   SphprodCh        [*,*]
               * 1 000 000
               Boosted jet sphericity product
               using charged tracks only
7   SE  I   SphprodEnflw     [*,*]
               * 1 000 000
               Boosted jet sphericity product
               using ENFW objects
8   DS  I   DetStatus        [*,*]
               Detector status word KREVDS
9   P2  I   trkPointer2      [*,*]
               Pointers
               bit 0 - 7 First bad V0 track
               bit 8 ... 0 = Error in BEETAG
               bit 9 ... 0 = Error in QIPBTAG
               bit 10 .. 0 = Event not useful for
               ............. QIPBTAG analysis
               bit 11 .. 0 = Error in GAMPEX
               bit 12 - 15 = Return code from
                             QIPBTAG+8
               bit 16 - 24 = Number of Jets from
                             QIPBTAG
               bit 25 .. 0 = Mainvertex from JULIA
                      .. 1 = Mainvertex from QFNDIP
               bit 26 .. 0 = QFNDIP OK.
                      .. 1 = QFNDIP failed
               bit 27-31 Free
10  VX  I   mainVtxXcoord    [*,*]
               (microns)
               X coordinate of main vertex
11  VY  I   mainVtxYcoord    [*,*]
               (microns)
               Y coordinate of main vertex
12  VZ  I   mainVtxZcoord    [*,*]
               (microns)
               Z coordinate of main vertex
13  S1  I   Sphertag1        [*,*]
               * 1 000 000
               Boosted sphericity hemi1
14  S2  I   Sphertag2        [*,*]
               * 1 000 000
               Boosted sphericity hemi2
15  T1  I   Transvmass1      [*,*]
               * 1 000 000
               BEETAG transverse mass hemi 1
```

```
  16  T2  I    Transvmass2       [*,*]
                  * 1 000 000
                  BEETAG transverse mass hemi 2
  17  M1  I    Mominertia1       [*,*]
                  * 1 000 000
                  BEETAG moment of inertia hemi 1
  18  M2  I    Mominertia2       [*,*]
                  * 1 000 000
                  BEETAG moment of inertia hemi 2
                  ==========================
                  CHANGES vs version 111.4 :
                  integerized
                  removed QIPBTAG probabilities
                   from word 10 - 12
                  put main vertex coordinates in
                   word 10 - 12
                  P2 : added bits 12-26
==============================================================


+------+
| NBIP |  Nano dst q(B)IPbtag summary bank
+------+
..............................................................
   1         I    Number of Columns (=2)
   2         I    Number of tracks
..............................................................
   1   TF  I    Track Flag            [*,*]
                   bit  0- 7 NANO track number
                   bit  8-21 QIPBTAG track flag
                   bit 22-25 jet assignment
                   bit 26-27 hemisphere assignment
                   bit 28-31 free
   2   TS  I    Track Significance  [*,*]
                   track significance
                   * 1 000 000
==============================================================


+------+
| NDJT |  Nano Dst reconstructed JeTs
+------+
..............................................................
   1         I    Number of Columns (=4)
   2         I    Number of reconstructed jets
..............................................................
   1   PX  I    PX (MeV)            [*,*]
                   Momentum X component
   2   PY  I    PY (MeV)            [*,*]
                   Momentum Y component
  3   PZ  I    PZ (MeV)         [*,*]
                 Momentum Z component
  4   EJ  I    EnergyofJet(MeV) [0,*]
                 Energy of the Jet
                 ==========================
                 CHANGES vs version 111.4 :
                 integerized
==============================================================

+------+
| NDLV |  Nano Dst Monte-Carlo V0 true
+------+  decay vertex
..............................................................
   1          I    Number of Columns (=4)
   2          I    Number of Monte-Carlo V0
..............................................................
   1   PO  I    POinter            [*,*]
                  Pointer to MC track
                  [in NDMS if > rows(NDNT)]
   2   DX  I    DecayvertexX      [*,*]
                  X coord of decay vertex
                  in microns
   3   DY  I    DecayvertexY      [*,*]
                  Y coord of decay vertex
                  in microns
   4   DZ  I    DecayvertexZ      [*,*]
                  Z coord of decay vertex
                  in microns
                  ==========================
                  CHANGES vs version 111.4 :
                  integerized
==============================================================

+------+
| NDMS |  Nano Dst Monte-Carlo Stable
+------+  tracks
..............................................................
   1          I    Number of Columns (=4)
   2          I    Number of stable Monte-Carlo
                   tracks
..............................................................
   1   PX  I    PX  (MeV)         [*,*]
                  Momentum X component
   2   PY  I    PY  (MeV)         [*,*]
                  Momentum Y component
   3   PZ  I    PZ  (MeV)         [*,*]
                  Momentum Z component
   4   HI  I    HIstory           [*,*]
```

```
                     Track history
                     bit 0 - 9 Particle code
                     bit 10-17 Pointer to matched track
                     bit 18-21 KSTABC+4
                     bit 22-30 Pointer to mother
                     ......... [in NDMS if > rows(NDNT)]
                     bit 31 .. Type of matched track:
                     ......... 0=Photon,1=Charged
                     ==========================
                     CHANGES vs version 111.2 :
                     bit 18-21 Number of daughters
                     ==========================
                     CHANGES vs version 111.1 :
                     bit 22-29 Pointer to mother
                     bit 30-31 Type of matched track:
                     ......... 0=Neutral,1=Photon,2=Charged
==============================================================


+------+
| NDNT |  Nano Dst Monte-Carlo uNstable
+------+  Tracks
..............................................................
    1          I    Number of Columns (=1)
    2          I    Number of unstable
                    Monte-Carlo tracks
..............................................................
    1   HI  I    HIstory          [*,*]
                    Track history
                    bit 0 - 9 Particle code
                    bit 10-17 Fragmentation variable z
                    ......... INT((z+0.004)*250)-0.0001)
                    bit 18-21 KSTABC+4
                    bit 22-30 Pointer to mother
                    ......... [in NDMS if > rows(NDNT)]
                    ==========================
                    CHANGES vs version 111.2 :
                    bit 18-21 Number of daughters
                    ==========================
                    CHANGES vs version 111.1 :
                    bit 22-29 Pointer to mother
==============================================================


+------+
| NDPH |  Nano Dst PHotons
+------+
..............................................................
    1          I    Number of Columns (=4)
    2          I    Number of selected photons
```

```
...........................................................................
   1   PX  I    PX  (MeV)         [*,*]
                   Momentum X component
   2   PY  I    PY  (MeV)         [*,*]
                   Momentum Y component
   3   PZ  I    PZ  (MeV)         [*,*]
                   Momentum Z component
   4   PA  I    ParticleAttr      [*,*]
                   Photon attributes
                   bit 0 - 7 PECO number
                   bit 8 - 9 Region code:0=crack,
                   ......... 1=barrel,2=endcap,3=overlap
                   bit 10-11 Number of photons in this
                   ......... PECO (3 = 3 or more)
                   bit 12 .. 1 = energy in stack 1 > 0
                   bit 13 .. 1 = no minima in stack 2
                   bit 14-31 Free
                   Note: if >4 photons found in a PECO,
                   only first 4 kept.
                   ==========================
                   CHANGES vs version 111.2 :
                   bit 12-31 Free
                   ==========================
                   CHANGES vs version 111.4 :
                   integerized
==============================================================


+-------+
| NDST |   Nano DST definition bank
+-------+
...........................................................................
    1           I    Number of Columns (=150)
    2           I    Number of rows
...........................................................................
    1   MT  I    MinTrk                [1,255]
                    Minimum number of tracks
    2   LP  I    LowerPcut (MeV)       [0,20]
                    Lower momentum cut for NDTK
    3   UP  I    UpperPcut (MeV)       [0,200000]
                    Upper momentum cut for NDTK
    4   CT  I    CosTheta              [*,*]
                    Cos theta cut for NDTK
                    * 1000000
    5   IH  I    ItcHits               [0,255]
                    ITC hits cut for NDTK
    6   TH  I    TpcHits               [0,255]
                    TPC hits cut for NDTK
    7   D0  I    D0cut (microns)       [0,*]
```

```
                  D0 (beam axis) cut
   8  Z0  I    Z0cut (microns)      [0,*]
                  Z0 (beam axis) cut
   9  JL  I    JetLowerpcut (MeV)  [0,20000]
                  Lower p cut before clustering
  10  JU  I    JetUpperpcut (MeV)  [0,200000]
                  Upper p cut before clustering
  11  JC  I    JetCostheta          [*,*]
                  Cos theta cut before cluster.
                  * 1000000
  12  JI  I    JetItchits           [0,255]
                  ITC hits cut before cluster.
  13  JT  I    JetTpchits           [0,255]
                  TPC hits cut before cluster.
  14  JD  I    JetD0cut (microns)  [0,*]
                  D0 cut before clustering
  15  JZ  I    JetZ0cut (microns)  [0,*]
                  Z0 cut before clustering
  16  JY  I    JYcut  (MeV)         [0,10000]
                  Mass cut for jets
  17  VE  I    VisibleEner  (MeV)  [-10000,200000]
                  Visible energy for jets
  18  RO  I    RecoOption           [,]
                  Reconstructed object option
  19  LE  I    LowerEcut (MeV)      [0,20]
                  Lower energy cut for NDPH
20-94  PH  I    PartHyp              [*,*]
                  Parameters for Particle Hyp
                  * 1000
  95  NP  I    NumPotbanks          [0,10]
                  Number of POT/DST/MINI banks
96-105 PB  I    PotBanks             [,]
                  List of POT/DST/MINI banks
 106  NR  I    NumRunbanks          [0,10]
                  Number of run banks
107-116 RB  I   RunBanks             [,]
                  List of run banks
 117  NN  I    NumNdstbanks         [0,20]
                  Number of NDST banks
118-137 NB  I   NdstBanks            [,]
                  List of NDST banks
 138  PR  I    PRoduction           [,]
                  Version of NDSTPROD
139-140 AV  I   AlphaVers            [,]
                  Alpha version and corr file
141-142 DA  I   DAte                 [,]
                  Date of the production
 143  MC  I    MCoptions            [*,*]
```

```
                Options for MC data
 144 MM  I    MinMomentum  (MeV)  [0,100.]
                Min. mom. in fragmentation
 145 AL  I    ALephlibvers         [*,*]
                Alephlib version * 1000
 146 CE  I    ChargedEnergy (MeV) [0,10]
                Minimum charged energy
 147 ME  I    MinimumEnergy (MeV) [0,*]
                Min. energy for ENFW objects
 148 EM  I    EnfwjetMcut (MeV)   [0,*]
                ENFW jets mass cut
 149 CS  I    Chi2increaseSnglTrk [0,*]
                Cut on chi square increase for
                a single V0 track coming from
                main vertex
 150 CB  I    Chi2increaseBothTrk [0,*]
                Cut on chi square increase for
                both V0 tracks coming from
                main vertex
                ==========================
                CHANGES vs version 111.4 :
                integerized
                added words 149,150
==============================================================

+------+
| NDTK |  Nano Dst charged TracKs
+------+
..............................................................
   1          I    Number of Columns (=6)
   2          I    Number of selected charged
                   Tracks
..............................................................
   1   PX  I    PX  (MeV)        [*,*]
                   Momentum X component
   2   PY  I    PY  (MeV)        [*,*]
                   Momentum Y component
   3   PZ  I    PZ  (MeV)        [*,*]
                   Momentum Z component
   4   DZ  I    DzeroZzero       [*,*]
                   D0 and Z0
                   bit 0 -15 D0 in [10^-3 cm]
                   bit 16-31 Z0 in [10^-3 cm]
   5   TA  I    TrkAttr          [*,*]
                   Track attributes
                   bit 0 - 7 JULIA track number (max 255)
                   bit 8 - 9 Number of VDET hits
                   ......... (3 = 3 or more)
```

```
                     bit 10 .. Charge (1=+1,0=-1)
                     bit 11-14 Jet number = row of NDEJ
                     ......... (max 15)
                     bit 15-18 Lepton candidate number =
                     ......... row of NDBM (max 15)
                     bit 19-26 Pointer to row in NDDE
                     ......... (max 255)
                     bit 27 .. Electron hypothesis
                     bit 28 .. Muon hypothesis
                     bit 29 .. Pion hypothesis
                     bit 30 .. Kaon hypothesis
                     bit 31 .. Proton hypothesis
  6   TQ  I    TrkQuality       [*,*]
                     Track Quality
                     bit 0 -13 Chi square per degree of
                     ......... freedom * 100
                     bit 14-22 Relative error on
                     ......... momentum * 10000
                     bit 23-26 Number of ITC hits
                     bit 27-31 Number of TPC hits
                     =========================
                     CHANGES vs version 111.4 :
                     integerized
============================================================


+------+
| NYVO |  NDST VO-Bank
+------+
............................................................
  1          I    Number of words/VO (=13)
  2          I    Number of VO
............................................................
  1   HY  I    HYpothesis       [*,*]
                   bit  0-3  Hypothesis (mass compat.)
                   bit  4    kinematic ambiguity
                   bit  5    free
                   bit  6    common track with another
                             VO candidate
                   bit  7    free
                   bit  8-15 IcodevO + 32 (s.YVOV bank)
                   bit 16-23 positive NANO track number
                   bit 24-31 negative NANO track number
  2   VX  I    VXcoor           [*,*]
                   Fitted VO x coordinate (micron)
  3   VY  I    VYcoor           [*,*]
                   Fitted VO y coordinate (micron)
  4   VZ  I    VZcoor           [*,*]
                   Fitted VO z coordinate (micron)
  5   VM  I    VMom              [*,*]
                 Fitted V0 momentum (MeV)
  6   TH  I    THeta             [*,*]
                 Theta of V0 (mrad/10)
  7   PH  I    PHi               [*,*]
                 Phi   of V0 (mrad/10)
 8-10 PP  I    PPos              [*,*]
                 Momentum of positive
                 particle from V0 (see VM,TH,PH)
  11  C2  I    C2chisquare       [0,*]
                 Chisquare of V0 vertex fit*100
  12  DL  I    DL                [0,*]
                 Error on decay length (micron)
  13  TT  I    TT                [0,*]
                 bit  0-15 (chi for track1)*10 (QV0CHK)
                 bit 16-31 (chi for track2)*10 (QV0CHK)
                  ==========================
                  CHANGES vs version 111.4 :
                  bank completly redefined
==============================================================

+------+
| NVEC |  SANDY work bank (not in
+------+  NanoDST files)
..............................................................
    1          I    Number of Columns (=29)
    2          I    Number of selected Tracks
..............................................................
    1   QP  F    QP                [0.0,*]
                   Momentum
    2   QX  F    QX                [*,*]
                   Momentum X component
    3   QY  F    QY                [*,*]
                   Momentum Y component
    4   QZ  F    QZ                [*,*]
                   Momentum Z component
    5   QE  F    QEnergy           [0.0,*]
                   Energy
    6   QM  F    QMass             [0.0,*]
                   Mass
    7   CH  F    CHarge            [*,*]
                   Charge
    8   DB  F    DB                [*,*]
                   D0
    9   ZB  F    ZB                [*,*]
                   Z0
   10   TN  I    TrackNumber       [*,*]
                   Track number
```

```
11  PJ  I    PointerJet        [0,15]
               Pointer to jet
12  PL  I    PointerLepton     [0,15]
               Pointer to lepton cand. NDEC/UC
13  PO  I    POinter           [0,15]
               Pointer to other bank
14  PI  I    PointerIon        [0,255]
               Pointer to dE/dx (NDDE bank)
15  HY  I    HYpothesis        [*,*]
               Particle HYpothesis
               bit 0 ... Hypothesis 1 = e
               bit 1 ... Hypothesis 2 = mu
               bit 2 ... Hypothesis 3 = pi
               bit 3 ... Hypothesis 4 = K
               bit 4 ... Hypothesis 5 = p
               bit 5 -31 Free
16  C2  F    Chi2perdof        [0.0,*]
               Chi square per degree of
               freedom
17  RS  F    RelSigma          [0.0,*]
               Relative error on momentum
18  IH  I    ItcHits           [0,15]
               ITC Hits
19  TH  I    TpcHits           [0,31]
               TPC Hits
20  VH  I    VdetHits          [0,3]
               VDET Hits
21  MC  I    McCode            [0,1023]
               Monte Carlo particle code
22  MA  I    MAtchedtrack      [0,255]
               Matched reconstructed track
23  MD  I    McDaughters       [0,15]
               Number of daughters
24  MM  I    McMother          [0,511]
               Pointer to mother
25  MT  I    McType            [0,2]
               Type of MC particle
               0=Neutral,1=Photon,2=Charged
26  ZF  F    ZFragment         [0.0,1.00]
               Fragmentation variable z
27  MK  I    McKstabc          [-4,11]
               Stability code KSTABC
28  MF  I    McFirstdaug       [0,*]
               Pointer to first daughter
29  MN  I    McNextsister      [0,*]
               Pointer to next sister
..............................................................
```

Appendix B

INSTALLING SANDY

B.1 VAX

1. Define, preferably in the system table, the logical `NANO` to point to `ALEPH$GENERAL:[NANO]`.
2. Add the command `@NANO:SANDYDEF` in the system or the ALEPH login procedure.
3. If the automatic update of ALEPH software (ALEPH_UPDATE, author: D. Candlin) is installed on your site, you should update the set-up (i.e. add a `NANO.FETCH` and/or a `NANOPROD.FETCH`).

B.2 IBM

The source reference for IBM systems is the NANO minidisk on CERNVM (GIME NANO).

Appendix C

What to change when going from SANDY to ALPHA

In this section it is explained how to change a fortran file written to be used inside SANDY to one to be used inside ALPHA.

```
    SANDY:                      ALPHA:

Commondecks:

    NCDE                        QCDE
    NMACRO                      QMACRO

Subroutines (same arguments):

    NDEDX                       QDEDX
    NDEDXM                      QDEDXM
    NMTAIL                      QMTAIL
    NWRITE                      QWRITE
    NPRINT                      QWEVNT
    NINFO                       QNINFO

Subroutines (different arguments and different names):
(For details see the manuals)

    QCOPY(IFROM,ITO)            QVCOPY(ITO,IFROM)
    NJMMCL(njets,kfi,kli,       QJMMCL(njets,'name',iclass,
        kfo,klo,rmcut,evis)            ycut,evis)

    PRIDEC(ITK)                 QWTREE(ITK,'option')
```

C.1 Variables, Functions and Subroutines not usable inside ALPHA

The variables, functions and subroutines listed below are generally not available inside ALPHA. Some other -namely all those which are related to the NanoDst production- are only available if the input is from a NanoDst. They are not listed here.

- General Variables

  – variables

    The following variables are output variables from the Alephlib routine GETLEP. For that CALL GETLEP(KRUN, IFOUN, IFILL, NHADBX, QELEP, BEAMXYZ, DBEAMXYZ) will get the same information as in SANDY.

SANDY:	ALPHA:
IFILL	IFILL
QELEP	QELEP
BEAMX	BEAMXYZ(1)
BEAMY	BEAMXYZ(2)
BEAMZ	BEAMXYZ(3)
DBEAMX	DBEAMXYZ(1)
DBEAMY	DBEAMXYZ(2)
DBEAMZ	DBEAMXYZ(3)
NHADBX	NHADBX
SAVERS	(CQVERS)

  – functions

    QMV(I) gives back the X,Y and Z coordinate of the main vertex. To get the same inside ALPHA the user has to perform a loop over all vertices and to ask for KVTYPE(IVX) eq 1 (primary vertex, see section ALPHA "VERTICES" in the ALPHA manual). For this vertex the corresponding variables are:

SANDY:	ALPHA:
QMV(1)	QVX(IVX)
QMV(2)	QVY(IVX)
QMV(3)	QVZ(IVX)

  – subroutines

    There are no equivalences for the following SANDY routines inside ALPHA:
    MAKSEV

- Charged Tracks

  in general tracks are not ordered inside ALPHA so everything connected to that is not available. For bad tracks see the appendix to the ALPHA manual 'Using the NanoDst with ALPHA'.

  – variables

    There are no equivalences for the following SANDY variables inside ALPHA:
    KFPH1, KLPH1, KFNH1, KLNH1, KFPH2, KLPH2, KFNH2, KLNH2, KFBCT, KLBCT, KFACT, KLACT, KNPH1, KNNH1, KNPH2, KNNH2, KNBCT, KNACT

  – functions

SANDY:	ALPHA:
PTOT(I)	QP(I)
ENER(I)	QE(I)
XC2OV(I)	.FALSE.
QC2DOF(I)	QFRFC2(I)/FLOAT(KFRFDF(I))
XSIG(I)	
XSIGOV(I)	.FALSE.
QSIGP(I)	QSIGP(I)
QRSIGP(I)	QSIGP(I)/QP(I)
KJET(I)	
XVOD(I)	
KVOD(I)	
XE(P,A,B)	SQRT(P*P+A*A)/B
PFRX(X,A,B)	SQRT(X*X*B*B-A*A)

- Jets
  - variables

SANDY:	ALPHA:
KFJET	KFJET
KLJET	KLJET
KNJET	KNJET

  - functions
    There are no equivalences for the following SANDY functions inside ALPHA: KJMUL(I)
  - subroutines

SANDY:	ALPHA:
NEWJET	QJMMCL

- Photons
  Some of the variables listed below have a corresponding one inside ALPHA.
  - variables

SANDY:	ALPHA:
KFGAM	KFGAT
KLGAM	KLGAT
KNGAM	KNGAT
KFGH1	
KLGH1	
KFGH2	
KLGH2	
KNGH1	
KNGH2	

– functions

SANDY:	ALPHA:
KGREG(I)	
KGMUL(I)	
KGPECO(I)	KPGPPE(I)
XGE1(I)	QPGPR1(I) gt 0.0
XGE2(I)	QPGPR2(I) gt QPGPR1(I) or
	QPGPR2(I) gt (QPGPF4(I) - QPGPR1(I))

- V^0
  Most of the variables listed below have a corresponding ones inside ALPHA except the particle hypothesis flags. They have been set if a V^0 candidate is less then 30 MeV away from the correct mass of a given hypothesis. So the user can easily set them by him(her)self.

– variables

SANDY:	ALPHA:
KFVOT	KFVOT
KLVOT	KLVOT
KNVOT	KNVOT

– functions

SANDY:	ALPHA:
KVOHYP(IVO)	
KVONTN(IVO)	KYVOK1(IVO)
KVOPTN(IVO)	KYVOK2(IVO)
KVOPOT(IVO)	KFCHT-1+KYVOK1(IVO)
KVONEG(IVO)	KFCHT-1+KYVOK2(IVO)
KVOIC(IVO)	KYVOIC(IVO)
QVOCH(IVO)	QYVOC2(IVO)
QVOTH(IVO)	ACOS(QCT(IVO))
QVOPH(IVO)	QPH(IVO)
QVOVTX(IVO)	QVX(KENDV(IVO))
QVOVTY(IVO)	QVY(KENDV(IVO))
QVOVTZ(IVO)	QVZ(KENDV(IVO))
QVODL(IVO)	
QVOEDL(IVO)	
XVOKIA(IVO)	
XVOSVA(IVO)	
XVOTRA(IVO)	
XKO(IVO)	
XLA(IVO)	
XAL(IVO)	
XGA(IVO)	
QVOPP(IVO)	QP(KFCHT-1+KYVOK1(IVO))
QVOPTH(IVO)	ACOS(QCT(KFCHT-1+KYVOK1(IVO)))

QVOPPH(IVO)	QPH(KFCHT-1+KYVOK1(IVO))
QVOPPX(IVO)	QX(KFCHT-1+KYVOK1(IVO))
QVOPPY(IVO)	QY(KFCHT-1+KYVOK1(IVO))
QVOPPZ(IVO)	QZ(KFCHT-1+KYVOK1(IVO))
QVONP(IVO)	QP(KFCHT-1+KYVOK2(IVO))
QVONTH(IVO)	ACOS(QCT(KFCHT-1+KYVOK2(IVO)))
QVONPH(IVO)	QPH(KFCHT-1+KYVOK2(IVO))
QVONPX(IVO)	QX(KFCHT-1+KYVOK2(IVO))
QVONPY(IVO)	QY(KFCHT-1+KYVOK2(IVO))
QVONPZ(IVO)	QZ(KFCHT-1+KYVOK2(IVO))

- Converted Photons

  No pointers or functions are inside ALPHA for converted photons. The user has to call QPAIRF with a pair of tracks (I1,I2 where I1 here is assumed to be the positive) and afterwards to the Alephlib routine PAIRCP(XA, YA, ZAV) (for the vertex coordinates).

  - variables

    There are no equivalences for the following SANDY variables inside ALPHA:
    KFGCO, KLGCO, KNGCO

  - functions

SANDY:	ALPHA:
KGCPTN(I)	KTNO(I1)
KGCNTN(I)	KTNO(I2)
KGCPOT(I)	I1
KGCNET(I)	I2
QGCDXY(I)	DXY
QGCDZ(I)	DZ2
QGCVX(I)	XA
QGCVY(I)	YA
QGCVZ(I)	ZAV

- Monte Carlo

  - variables

    There are no equivalences for the following SANDY variables inside ALPHA:
    KFMUT, KLMUT, KNMUT, KFMST, KLMST, KNMST, KQ, KQBAR, KFPAR, KLPAR

  - functions

    There are no equivalences for the following SANDY functions inside ALPHA:
    XMCNE(I), XMCCH(I), XMCGA(I), KPTCH(I), QV0VX(I), QV0VY(I), QV0VZ(I), QZFR(I)

  - subroutines

    There are no equivalences for the following SANDY subroutines inside ALPHA:
    DECSEA, DAUSEA