

ALEPH 93-1  
SOFTWR 93-1  
E. Blucher  
16 December 1992

# ALPHA User's Guide

ALPHA

ALEPH PHYSICS ANALYSIS PACKAGE

Version 114/115

Authors:

H.Albrecht and E.Blucher

With contributions from:

A.Belk, B.Bloch-Devaux, J.Boucrot, C.Bowdery, J.Carr, D.Cinabro, R. Forty, C.Gay,  
R.Hagelberg, S.Haywood, J.Hilgart, R. Jacobsen, P.Janot, R.Johnson, E.Lançon,  
M.N.Minard, H.G.Moser, J.Nash, M.Pepe-Altarelli, P.Perez, F.Ranjard, M.Scarr,  
D.Schlatter, H.Seywerd, M.Talby, G.Taylor, S.Wasserbaech, J.Wear, and T.Wildish

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Getting Started</b>	<b>2</b>
<b>3</b>	<b>User routines</b>	<b>3</b>
3.1	General Comments . . . . .	3
3.1.1	Name conventions . . . . .	3
3.1.2	Including ALPHA features in Fortran code . . . . .	3
3.1.3	“HAC” parameters . . . . .	5
3.1.4	Implicit None . . . . .	5
3.2	User Initialization . . . . .	5
3.3	Event analysis routine . . . . .	5
3.4	User termination routine . . . . .	6
3.5	Other User Subroutines . . . . .	6
3.5.1	New Run . . . . .	6
3.5.2	Unkown Record Type . . . . .	7
3.5.3	Initialize the histogram package . . . . .	7
3.5.4	Terminate the histogram package . . . . .	7
3.5.5	Initialize BOS . . . . .	7
<b>4</b>	<b>Data Cards</b>	<b>8</b>
4.1	Input/Output . . . . .	8
4.1.1	ALEPH file types . . . . .	8

4.1.2	FILI: Input files . . . . .	9
4.1.3	FILO: Output files . . . . .	11
4.1.4	Event Directories . . . . .	12
4.1.5	COPY: Copying events . . . . .	13
4.2	ALPHA Process cards . . . . .	14
4.3	UNPK: POT / DST unpacking . . . . .	14
4.4	READ: Input from different card files . . . . .	15
4.5	DEBU: Debug output . . . . .	15
4.6	TIME: Job time control . . . . .	15
4.7	Histograms . . . . .	16
4.7.1	HIST: Write histogram file . . . . .	16
4.7.2	HTIT: General histogram title . . . . .	16
4.7.3	NOPH: Histogram Printing . . . . .	16
4.8	FIEL: Magnetic field . . . . .	16
4.9	FRF0: Use track fit without vertex detector . . . . .	17
4.10	Weight factors for calorimeters . . . . .	17
4.11	EFLW and EFLJ: Energy Flow . . . . .	17
4.12	Particle table . . . . .	17
4.12.1	PMOD: Modify particle attributes . . . . .	18
4.12.2	PNEW: New particles . . . . .	18
4.12.3	PTRA: Modify particle names in the MC particle table . . . . .	18
4.13	SYNT: Syntax Check . . . . .	19
<b>5</b>	<b>Creating Histograms and Ntuples</b>	<b>20</b>
5.1	Booking and Filling Histograms/Ntuples . . . . .	20
5.1.1	Book a 1–dimensional histogram . . . . .	20
5.1.2	Book a 2–dimensional histogram . . . . .	21

5.1.3	Book an Ntuple . . . . .	21
5.1.4	Book an Ntuple with run, event number . . . . .	22
5.1.5	Fill Ntuple plus run, event number . . . . .	22
5.1.6	Fill Ntuple with many variables . . . . .	22
5.1.7	Fill Ntuple with many variables plus run, event number . . . . .	22
5.1.8	Sample ALPHA program to book and fill histogram, Ntuple . . . . .	23
5.2	Histogram output – the ALPHA cards file . . . . .	23
5.2.1	HIST: Write histogram file . . . . .	23
5.2.2	NOPH: Histogram Printing . . . . .	24
5.2.3	HTIT: General histogram title . . . . .	24
<b>6</b>	<b>Mnemonic symbols</b>	<b>25</b>
6.1	Mathematical and physical constants . . . . .	25
6.2	Run information . . . . .	25
6.3	Event information . . . . .	26
6.3.1	Event header: from bank EVEH . . . . .	26
6.3.2	Event directory information . . . . .	26
6.3.3	Event generator status: from bank KEVH . . . . .	27
6.3.4	Detector HV status: from banks REVH, LOLE . . . . .	27
6.3.5	Trigger Information: from XTEB or XTRB, XTCN . . . . .	27
6.3.6	General event information: from bank DHEA . . . . .	28
6.3.7	Beam position from BOM system: from bank BOMB . . . . .	28
6.4	ECAL Wire Energies . . . . .	29
6.5	ALPHA Internal Constants, Variables . . . . .	29
6.5.1	Event counts . . . . .	29
6.5.2	Program status . . . . .	29
6.5.3	Event status . . . . .	29

6.5.4	Input / output units . . . . .	29
6.5.5	Timing . . . . .	30
6.5.6	Character variables . . . . .	30
<b>7</b>	<b>ALPHA “Tracks” and “Vertices”</b>	<b>31</b>
7.1	Access by Fortran DO loops . . . . .	31
7.1.1	ALPHA “TRACKS” . . . . .	32
7.1.2	ALPHA VERTICES: . . . . .	33
7.2	Loops over ECAL and HCAL objects . . . . .	33
7.3	Relationships between objects in different subdetectors . . . . .	34
7.4	Direct access to particles . . . . .	35
7.4.1	Particle name and class . . . . .	35
7.4.2	Example: Loop over all MC generated positrons . . . . .	35
7.4.3	Particle name versus integer particle code – time consumption . . . . .	36
7.4.4	Loops over a particle and its antiparticle . . . . .	36
7.4.5	Analysis of particle systems: Examples . . . . .	37
7.5	Mother – daughter relationships . . . . .	38
7.5.1	Mother to daughters . . . . .	38
7.5.2	Daughter to mother(s) . . . . .	39
7.6	Access to the “same” object . . . . .	39
7.6.1	Loops over copies of the “same” object using KSAME . . . . .	39
7.6.2	Find original copy of a charged track . . . . .	40
7.7	Match reconstructed tracks and MC truth . . . . .	40
7.8	Track – vertex relationships . . . . .	41
<b>8</b>	<b>ALPHA Track and Vertex Attributes</b>	<b>42</b>
8.1	“Track” attributes. . . . .	42

8.1.1	Basic attributes . . . . .	42
8.1.2	V0 Mass . . . . .	42
8.1.3	Track error covariance matrix . . . . .	43
8.1.4	Distance to the beam position: . . . . .	43
8.1.5	Stability code . . . . .	44
8.1.6	Test a particle's name . . . . .	44
8.1.7	Test if particles are based on the same object . . . . .	45
8.1.8	Flags, pointers, etc. . . . .	45
8.2	"Track" related detector data . . . . .	45
8.2.1	Global geometrical track fit: Bank FRFT . . . . .	46
8.2.2	Number of coordinates used for the global fit: Bank FRTL . . . . .	46
8.2.3	Charged-particle identification: Bank FRID . . . . .	46
8.2.4	dE/dx data: Bank TEXS . . . . .	47
8.2.5	Electron identification: Bank EIDT . . . . .	47
8.2.6	Muon - HCAL association: Bank HMAD . . . . .	48
8.2.7	Muon chamber data: Bank MCAD . . . . .	48
8.2.8	QMUIDO Muon Identification: Bank MUID . . . . .	48
8.2.9	ECAL objects: Bank PECO . . . . .	49
8.2.10	ECAL objects: Bank PEPT . . . . .	49
8.2.11	HCAL objects: Bank PHCO . . . . .	49
8.2.12	Reconstructed V0s: Bank YV0V . . . . .	50
8.2.13	Photons from GAMPEC: Bank EGPC . . . . .	50
8.2.14	Energy Flow: Bank EFOL . . . . .	51
8.2.15	Neutral objects from PCPA: Bank PCQA . . . . .	51
8.3	Vertex attributes . . . . .	51

<b>9</b>	<b>Kinematics and Track Operations</b>	<b>52</b>
9.1	Scalar quantities . . . . .	52
9.2	Vector quantities . . . . .	53
9.2.1	General Remarks . . . . .	53
9.2.2	Add 4–momenta of particles . . . . .	54
9.2.3	Recalculate 4–Vector of V0 . . . . .	55
9.2.4	Copy a track . . . . .	55
9.2.5	Cross product . . . . .	56
9.2.6	Drop tracks . . . . .	56
9.2.7	Copy track attributes into a Fortran array . . . . .	56
9.2.8	Create a new track . . . . .	57
9.2.9	Save a track . . . . .	57
9.2.10	Save a track inside particle/antiparticle loop . . . . .	58
9.2.11	Save a track and set its mass . . . . .	58
9.2.12	Save a track with class ICLASS . . . . .	58
9.2.13	Modify track parameters . . . . .	59
9.2.14	Set User Track Flags . . . . .	60
9.2.15	Subtract track momenta . . . . .	60
9.2.16	Zero track attributes . . . . .	61
9.3	Kinematic fitting . . . . .	61
9.4	Vertex fitting with YTOP . . . . .	61
9.5	Lorentz transformations . . . . .	63
9.5.1	Boost a track and its daughters . . . . .	63
9.5.2	Boost a track . . . . .	63
9.5.3	Boost all tracks of a given class . . . . .	64

<b>10 Event Topology Routines</b>	<b>65</b>
10.1 Options for “QJxxxx” routines . . . . .	65
10.1.1 Set option for reconstructed objects . . . . .	65
10.1.2 Set option for MC particles . . . . .	66
10.2 Lock tracks / subsamples of tracks . . . . .	66
10.2.1 Lock a single “track” . . . . .	67
10.2.2 Unlock a single “track” . . . . .	67
10.2.3 Lock a track “family” . . . . .	67
10.2.4 Unlock tracks (locked with QLOCK) . . . . .	68
10.2.5 Reverse the lock state (corresponding to QLOCK) . . . . .	68
10.2.6 Second Lock . . . . .	68
10.3 Add momenta of all particles of a given class . . . . .	68
10.3.1 Input argument . . . . .	69
10.3.2 Results . . . . .	69
10.4 Momentum tensor eigenvalues and eigenvectors . . . . .	70
10.5 Linearized momentum tensor eigenvalues and eigenvectors . . . . .	70
10.6 Sphericity . . . . .	70
10.7 Thrust . . . . .	71
10.8 Fox–Wolfram Moments . . . . .	71
10.9 Divide event into two hemispheres . . . . .	72
10.10 Missing energy, mass, momentum . . . . .	73
10.11 Jet Finding . . . . .	73
10.11.1 Scaled Invariant Mass Squared Algorithm . . . . .	73
10.11.2 Scaled Minimum Distance Algorithm . . . . .	75
10.11.3 JETSET algorithm LUCLUS from LUND . . . . .	75
10.11.4 PTCLUS: Jet-finding algorithm . . . . .	76



<b>11 Energy Flow</b>	<b>77</b>
11.1 ENFLW Energy Flow . . . . .	77
11.2 Mask Energy Flow . . . . .	78
11.3 PCPA-based Energy Flow . . . . .	79
<b>12 Other ALPHA Physics Routines</b>	<b>81</b>
12.1 dE/dx Analysis . . . . .	81
12.1.1 Calculate dE/dx for Track ITK . . . . .	81
12.1.2 Modified QDEDX for Monte Carlo . . . . .	82
12.1.3 Check TPC High Voltage for dE/dx . . . . .	82
12.1.4 Check Existence of dE/dx Calibration for Run . . . . .	83
12.2 Photon conversions . . . . .	83
12.3 Muon Identification: QMUIDO . . . . .	84
12.4 Utility Routines for VDET Analysis . . . . .	86
12.4.1 Number of VDET hits per layer for track ITK . . . . .	86
12.4.2 VDET HV status . . . . .	86
12.4.3 VDET Readout Status . . . . .	87
<b>13 ALPHA Utility Routines: Printing, Writing Events, Timing, etc.</b>	<b>88</b>
13.1 Program termination . . . . .	88
13.2 Write the current event on the output file . . . . .	88
13.3 Set classification word written to event directory . . . . .	88
13.4 Timing . . . . .	89
13.4.1 Print job time consumption . . . . .	89
13.4.2 Measure time consumption of part of program . . . . .	89
13.5 Print routines . . . . .	90
13.5.1 Print a message . . . . .	90

13.5.2	Print a message plus run, event number . . . . .	90
13.5.3	Print full event summary (many pages) . . . . .	90
13.5.4	Print event header (one line) . . . . .	91
13.5.5	Print full event header (many lines) . . . . .	91
13.5.6	Print information for “track” . . . . .	91
13.5.7	Print information for vertex . . . . .	92
13.5.8	Print summary for categories of tracks or vertices . . . . .	92
13.5.9	Print decay tree of track ITK. . . . .	92
<b>14</b>	<b>Modifying ALPHA banks</b>	<b>93</b>
14.1	User track / vertex sections . . . . .	93
14.1.1	Reserve user space for tracks . . . . .	93
14.1.2	Reserve user space for vertices . . . . .	94
14.2	Modifying track / vertex attributes . . . . .	94
<b>15</b>	<b>Particle Table</b>	<b>95</b>
15.1	Description . . . . .	95
15.2	Particle name, particle code . . . . .	95
15.3	How to spell particle names . . . . .	96
15.4	Data cards for particle table . . . . .	96
15.4.1	PMOD: Modify particle attributes . . . . .	96
15.4.2	PNEW: New particles . . . . .	97
15.4.3	PTRA: Modify particle names in the MC particle table . . . . .	97
15.5	Access to particle properties . . . . .	98
<b>A</b>	<b>Program Structure</b>	<b>99</b>
<b>B</b>	<b>Bank description</b>	<b>100</b>

<b>C</b>	<b>Where to find ALPHA at CERN</b>	<b>105</b>
C.1	ALPHA on CERNVM . . . . .	105
C.2	ALPHA on VXCERN, ALWS . . . . .	106
C.3	ALPHA on the CRAY . . . . .	107
C.4	ALPHA on DECstations, SHIFT . . . . .	107
<b>D</b>	<b>Using the Mini-DST with ALPHA</b>	<b>109</b>
D.1	Doing analysis with the Mini . . . . .	109
D.2	Differences between POT/DST and Mini-DST . . . . .	110
D.3	Writing a Mini-DST . . . . .	110
<b>E</b>	<b>Standard particle table</b>	<b>112</b>
	<b>Index</b>	<b>114</b>

# Chapter 1

## Introduction

The ALEPH Physics Analysis package ALPHA is intended to simplify Fortran programs for physics analysis. Although all ALEPH data types can be processed with ALPHA, the program is designed primarily for analysis of JULIA output (POT, DST, or MINI). All event input/output is done by ALPHA – the user has to provide only the name(s) of the input/output data set(s). ALPHA also provides easy access to physical variables (e.g., momentum, energy), so the user can write physics analysis programs without detailed knowledge of the ALEPH data structure (tabular BOS banks). An extensive set of utility routines (e.g., kinematics, event shape, etc.) is available as part of the ALPHA package.

The program structure (Appendix A) is extremely simple. Three Fortran routines are normally supplied by the user: job initialization, event processing, and job termination (see Ch. 3). Reconstructed objects (tracks, vertices, cal. objects) can be accessed with simple DO loops. For Monte Carlo generated events, the MC “truth” information is accessible in the same way as reconstructed tracks and vertices (see Ch. 7).

This document describes all features of the ALPHA program. For first-time users, the important parts to read are Ch. 2 (getting started), Ch. 3 (user routines), Ch. 4 (event input), Ch. 7 (loops over tracks), and Ch. 8 (track attributes).

## Chapter 2

### Getting Started

Two files must be provided to run an ALPHA job:

1. A file which contains the Fortran or Historian code for the user subroutines (see Ch. 3).
2. A card file which contains names of input / output data files, as well as other parameters (see Ch. 4).

The libraries needed to link the program are described in Appendix C. To run ALPHA, the following files must be assigned:<sup>1</sup>

#### **unit**

**6** Print output file

**7** Card file

**76** (optional) In an interactive session, this unit may be assigned to the terminal. Short messages will be sent to the terminal and long listings sent to the output file.

Command files are available for the VAX (ALPHARUN), IBM (ALPHARUN), CRAY (CRALPHA on CERNVM), and UNIX (alpharun and SFALPHA on CERNVM) which make these file assignments, and also perform the following tasks:

1. compile and link the Fortran (Historian) code;
2. run the program interactively or submit a BATCH job.

On the VAX, ALPHARUN also facilitates the use of a set of VAX debugger command files which simplify ALPHA program debugging (see Appendix C).

---

<sup>1</sup>Other units used by ALPHA are listed in Section 6.5.4; units 90, 91, 92, and 93 are always free for private output files.

# Chapter 3

## User routines

In this chapter, ALPHA routines which are intended to be modified by the user are described. Normally, only three routines are provided by the user: initialization (QUINIT), event analysis (QUEVNT), and program termination (QUTERM). Models for these three subroutines are available (see Appendix C). Other subroutines which may be modified by the user are also described here. User routines can be provided either as a plain Fortran file or as a Historian input file; the ALPHARUN command file described in Chapter 2 supports both options. For all user routines, default versions exist on the ALPHA library which are loaded automatically if no user code is given.

### 3.1 General Comments

#### 3.1.1 Name conventions

All Fortran symbols defined in the ALPHA package start with Q, K, C, or X:

- 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 (always in combination with Q as 2nd character).

To avoid conflicts with the hundreds of variables defined in the ALPHA package, it would be safest if your own Fortran names for subroutines, variables, etc. did *NOT* start with Q, K, X, or CQ.

#### 3.1.2 Including ALPHA features in Fortran code

In addition to subroutines, the ALPHA package consists of a set of statements which have to be included at the beginning of user subroutines or functions. There are two sets of these statements:

- QCDE** COMMONs, DIMENSIONs, EQUIVALENCEs, PARAMETERs, DATAs, type declarations (all ALPHA symbols starting with C or X are individually declared as CHARACTER or LOGICAL, respectively).

**QMACRO** statement function definitions (from the user's point of view, statement functions look exactly like "normal" Fortran functions, but their execution is faster).

The BOS array RW(...) and IW(...), as well as the BMACRO statement functions (RTABL, etc.), are included in QCDE and QMACRO.

These sets of statements can be included in user subroutines by machine-dependent Fortran statements or by Historian statements, as shown below.

#### VAX / VXCERN, ALWS

```
INCLUDE 'PHYINC:QCDE.INC'  
INCLUDE 'PHYINC:QMACRO.INC'
```

#### IBM / CERNVM

```
INCLUDE 'QCDE INC *'  
INCLUDE 'QMACRO INC *'
```

#### CRAY

```
INCLUDE 'QCDE.INC'  
INCLUDE 'QMACRO.INC'
```

#### UNIX / DECstation, SHIFT

```
INCLUDE '/aleph/phy/qcde.inc'  
INCLUDE '/aleph/phy/qmacro.inc'
```

#### using HISTORIAN

```
*CA QCDE  
*CA QMACRO
```

**Important!** The following sequence of statements must be observed:

1. SUBROUTINE or FUNCTION statement
2. QCDE, your own COMMONs, DIMENSIONs, etc.
3. DATA statements
4. QMACRO, your own statement function definitions (if any)
5. your executable Fortran statements

### 3.1.3 “HAC” parameters

The HAC (Handy ACcess) parameters denote the offset of attributes within each BOS bank. For banks accessible by mnemonic symbols in ALPHA (see Ch. 6), this offset is taken into account automatically, and the corresponding HAC parameters are available in QCDE (note that names of HAC parameters and mnemonic symbols are closely related).

A separate include file / comdeck QHAC is provided for the HAC parameters of all banks appearing on POT/DST/MINI event files which are NOT available in QCDE. QHAC can be included in the same way as QCDE and may be used in conjunction with it.

### 3.1.4 Implicit None

The include file (common deck) QDECL contains the declaration (integer, real) of all ALPHA variables and statement functions in QCDE and QMACRO. People wishing to use IMPLICIT NONE should include QDECL in the same way as QCDE.

## 3.2 User Initialization

### **SUBROUTINE QUNIT**

This routine should be used to book histograms and to perform other user initialization. All standard initialization work is performed automatically in the ALPHA subroutine QMINIT before QUNIT is called. The standard ALPHA initialization includes

- Initialization of BOS (500,000 words working space)
- Initialization of HBOOK (100,000 words working space)
- Reading data cards
- Opening the ALEPH data base
- Initialization of ALPHA.

These space allocations are large enough for most applications; they can be increased by modifying the routines described in sections 3.5.4 and 3.5.5.

## 3.3 Event analysis routine

### **SUBROUTINE QUEVNT(QT,KT,QV,KV)**

*QUEVNT* is called once for each event. The current event is read in, unpacked, and ready to be analyzed when QUEVNT is called.



**Subroutine arguments** *QT,KT,QV,KV* are used for special applications; see 14.2. The subroutine arguments must be given even if they are not used.

**IMPORTANT:** Do NOT perform a BOS garbage collection in QUEVNT or in any routine called by QUEVNT.

### 3.4 User termination routine

#### **SUBROUTINE QUTERM**

This subroutine can be used for anything which needs to be done at the end of a job (e.g., histogram manipulations). Histogram output is done automatically in the ALPHA routine QMTERM.

QUTERM must never be called directly. For program termination, use the statement (see 13.1):

```
CALL QMTERM ('any message')
```

QMTERM, in turn, calls QUTERM. QMTERM is called automatically after all input files have been processed.

### 3.5 Other User Subroutines

The routines in this section normally do not have to be modified. As mentioned above, default versions of all user routines are loaded if no new versions are provided.

#### 3.5.1 New Run

#### **SUBROUTINE QUNEW (IROLD,IRNEW)**

This routine is called from QMNEW once a new run is encountered on the event input file, i.e.,

- either a run record is read on the input file
- or the run number in an event record has changed
- or both conditions are fulfilled.

QUNEW may be used to initialize run-dependent data or to print run statistics.

#### **Input arguments**

**IROLD** old run number: = 0 if called for the first time.

**IRNEW** new run number: = 0 if called from QMTERM during the program termination.

**Default** no action: RETURN.

### 3.5.2 Unkown Record Type

#### SUBROUTINE QUSREC

This routine is called whenever a record is read that is neither a run nor event record (e.g., slow control record); the routine can be used to analyze these special records.

**Default:** no action: RETURN.

### 3.5.3 Initialize the histogram package

#### SUBROUTINE QUIHIS

NOT intended for histogram booking (use QUNIT).

- Called automatically from QMINIT.

**Default:** Initialize HBOOK4: CALL HLIMIT (100000).

Note: Users of *very* large NTUPLES will sometimes have to make modifications to the call to HROPEN included in QMIHIS. See CERN Computer Newsletter CNL201 p. 21 for details.

### 3.5.4 Terminate the histogram package

#### SUBROUTINE QUTHIS

- Called automatically from QMTERM.

**Default:** Terminate HBOOK: CALL HISTDO If the HIST data card is given, write output on histogram file.

### 3.5.5 Initialize BOS

#### SUBROUTINE QUIBOS

The length of the BOS working space COMMON /BCS/ is explicitly declared in this subroutine.

- Called automatically from QMINIT.

**Default:** initialize BOS with 500,000 words working space.

## Chapter 4

### Data Cards

In this chapter, the ALPHA data cards are described. The cards file is used to control input and output for ALPHA, and is used to control many ALPHA features. For completeness, all ALPHA cards are listed in this chapter; some cards are described in more detail in other chapters.

The following rules should be followed for all entries in the card file.

1. Start the text of your cards in column 1.
2. Use only upper case characters unless the lower case characters are significant.
3. Except for FILI cards (4.1.2), data cards can be given in any order.
4. The ENDQ card must be the last entry in the card file.

Data cards may also be used to enter your own data into the program. If your cards are given in standard BOS format, their contents will be available as standard BOS banks. For example, if the card **CUTS 4 3.7** appears in the ALPHA card file, the following Fortran may be used to get access to the values:

```
ICUTS=IW(NAMIND('CUTS'))
IF(ICUTS.NE.0)THEN
  ICUT1=IW(ICUTS+1)
  RCUT1=RW(ICUTS+2)
ENDIF
```

### 4.1 Input/Output

#### 4.1.1 ALEPH file types

There are several ALEPH file types:

<b>NATIVE</b>	machine-dependent input/output
<b>EPIO</b>	machine-independent input/output
<b>EDIR</b>	event directories

**DAF**            direct access files (e.g., data base)  
**CARDS**        card image files (e.g., ALPHA data cards)  
**HIS**            histogram files (machine-dependent HBOOK format)  
**EXCH**         histogram files (machine-independent HBOOK format)

The ALEPH file type cannot be recognized automatically. The file type should be given as 2nd part of the data set name (extension on VAX; file type on IBM).

**Examples:**

On VAX:

MYFILE.EPIO

On IBM:

MYFILE EPIO \*

ALPHA uses the data set name to determine the format. For file names which do not follow this convention, see the following section.

#### 4.1.2 FILI: Input files

**Format** *FILI* 'data-set-name | parameters'

Any number of FILI cards may be given – the data sets are read in the order the cards are given. Different file formats (e.g., NATIVE, EPIO) and data from POT, DST, and MINI can be processed in the same job. The program SCANBOOK can be used to create FILI cards with the proper format.

#### How to specify data set names on cards: Examples

**Disk files:**

	VAX / ALWS	IBM / CERNVM
1	FILI 'PHY:HADRON.NATIVE'	FILI 'HADRON.NATIVE'
2	FILI 'SCR:HADRON.EPIO   EPIO'	FILI 'HADRON.EPIO   EPIO'
3	FILI 'SCR:HADRON.DATA   EPIO'	FILI 'HADRON.DATA   EPIO'
4	FILI 'AL\$EDIR:M0012700.EDIR'	FILI 'M0012700.EDIR   GIME EDIR'
5	FILI 'I12345   EDIR'	FILI 'I12345   EDIR   GIME EDIR'
	CRAY	
6	FILI 'D0006898 EDIR   GIME PUBXU 209'	

Explanation:

1. Complete specification. “.NATIVE” defines the file format.
2. “| EPIO” can be omitted here because the format is already specified in the data set name. The vertical bar separates the file name from the parameters.
3. “DATA” is non-standard and not recommended. In such a case, the format must be given as a parameter: “ | EPIO”.
4. On IBM: Execute a GIME of the event directory disk (GIME EDIR is equivalent to GIME PUBXU 209).
5. This short format may be used only for standard data files: ABxxxx or Ixxxxx.
6. File will be “acquired” from the IBM disk and stored as /pool/aleph/BBP22409.EPIO.

### Staged tapes and cartridges:

The same format can be used for IBM, CRAY, and UNIX:

```
FILI 'ALDATA | EPIO | CART AC0349.1.SL options'
```

Here, ALDATA is the data set name, AC0349 is the cartridge VID, 1 is the FSEQ, and SL denotes a standard labeled tape. The options are different for each computer. On the IBM, the option SIZE 200 is usually used to allocate space for the tape, because the default size allocation is only 22 Mbytes.

### Run / event selection

The following cards may be used to select particular runs or events for analysis.

- SEVT 15 2 4 6 8 -11** Select EVenTs 2,4,6,8,9,10,11 of run 15 The 1st number is a run number, the following ones are event numbers. Negative numbers define a range of events. It is possible to include several SEVT cards in a card file, but only one SEVT card can be given for each run. The SEVT card, as well as the SRUN card described below, will work if the input files are ordered to have increasing run/event numbers. If the input files are not in sequential order, the selection cards will work correctly only if the NSEQ card (see below) is included in the card file.
- SRUN 2 -4 6 8 -10** Select RUNs 2,3,4,6,8,9,10. See note under SEVT on sequential order of runs.
- IRUN 1 5 7 11 -9999999** Ignore RUNs 1,5,7,11,12,13,14,...,9999999
- NEVT 5 -7** Select the 5th, 6th, and 7th records (in the order they are stored on the input file regardless of their run / event numbers).
- NEVT 3** Select the 1st, 2nd, and 3rd records. More than two numbers are not allowed on this card.

**NSEQ** This card must be included to use the selection functions described above with files that do not have run/event numbers in increasing order.

### 4.1.3 FILO: Output files

Event output is controlled by the FILO card and by the subroutine QWRITE (see 13.2). The data set name and options are given on the FILO card. Calling QWRITE writes the current event to the output file. The COPY card (see 4.1.5) may also be used to write events to a file. If a FILO card is given, all run records will be written out by default (see ALLR, NORU, and SELR below).

**Format:** *FILO 'data-set-name | parameters'*

<b>data set name</b>	same as on FILI cards; see examples in 4.1.2.
	<b>File format</b> NATIVE, EPIO, or EDIR
<b>parameters:</b>	(optional)
<b>ALLR</b>	write all run records to the output file (default).
<b>NORU</b>	write no run records to the output file.
<b>SELR</b>	write run records as soon as the first event record corresponding to it is written. It can be used if few events are selected from a large data sample; without this option, the output file may consist mainly of run records. With SELR, only run records which are followed by event records are written.
<b>SREC</b>	write all "special" records to the output file. Without this card, all records which are neither event nor run records will not be written.
<b>NOOV</b>	simple-minded protection against involuntarily overwriting data sets. If this parameter is given AND the output data set already exists, the program will stop. Note that problems with overwriting do not arise on the VAX.
<b>Examples:</b>	<i>FILO 'ABC NATIVE   SELR   NATIVE   NOOV'</i> The 2nd "NATIVE" is redundant; see 4.1.2.
<b>CRAY:</b>	<i>FILO 'MYDATA EPIO   EPIO   DISPOSE'</i> The output file will be sent back to the user's reader on IBM.

More than one FILO card is not accepted. If you want to write on several output units simultaneously, use the standard BOS routines.

The output event type (POT, DST, MINI – see 4.1.1) is the same as the input event type unless the MINI card is given (see below). Event directories can be created from any input event type (see 4.1.4).

### MINI: Select Mini-DST for output file

If the MINI card is given, the output file specified with the FILO card will be written in Mini-DST format; see Appendix D and the Mini-DST User's Guide for details.

### COMP: Data compression

Integer numbers are written in compressed format by default. The data card

**COMP 'NONE'** suppresses the compression.

### NWRT: Number of events to write out

**NWRT 15** Set maximum number of events to be written on the output file to 15.

## 4.1.4 Event Directories

Event directories make it possible to read ALEPH data files in direct access mode.

### Creating Event Directories

There are two ways to create an event directory with ALPHA.

- One can specify EDIR as a file type in the FILO card:

```
FILO 'TEST.EDIR'
```

The event directory can be created by using the COPY data card, or by calling QWRITE from the user program.

- It is also possible to create the event directory at the same time as another output file. The required FILO card is

```
FILO 'TEST.EPIO | WITH TEST.EDIR '
```

With either of the above options, it is also possible to set the 30 bit classification word stored for each event in the event directory. For each bit which is to be set, the user must call the routine QWCLAS (see 13.3):

```
CALL QWCLAS(IBIT)    IBIT = 1, 30
```

If three bits are to be set, QWCLAS has to be called three times. Note that a call to QWCLAS simply turns on a single bit while leaving other bits unchanged. The initial classification word is the one read from the input file; therefore, the classification word must be zeroed by calling QWCLAS with IBIT=0 before storing your own values. If QWCLAS is not called, the classification word will be set equal to that on the input file.

## Reading data with event directories

The event directory must be specified in the FILI card:

```
FILI 'TEST.EDIR'
```

All of the run / event selection cards (Sec. 4.1.2) can be used with event directories. If the CLAS card (described below) is given in the card file, only events with certain classification words will be read from the input file.

### CLAS: Select events with certain classification word

**Format**        *CLAS* *ibit1, ibit2, ... , ibitn* read events with bit *ibit1* and/or *ibit2* etc. = 1

It is also possible to make more complicated selections based on the event classification word by supplying a new version of the routine BSELEC. This routine should be extracted from the ALPHA library and modified. The default version of BSELEC, shown below, checks to see if a MASK has been supplied with the CLAS card. If so, it checks to see if the event classification word IWORD and MASK have any bits in common. Events are read in only if BSELEC is .TRUE. The line KCLASW=IWORD should not be changed; this line allows access to the event directory classification word inside of ALPHA (*e.g.*, inside QUEVNT).

```
LOGICAL FUNCTION BSELEC (IWORD,MASK)
INCLUDE 'QCDE INC *'
BSELEC = .TRUE.
IF (MASK.NE.0 .AND. IAND(MASK,IWORD).EQ.0) BSELEC = .FALSE.
KCLASW=IWORD
END
```

### 4.1.5 COPY: Copying events

The COPY card directs ALPHA to copy events using the data cards described above (*i.e.*, FILI, FILO, SEVT, SRUN, IRUN, NEVT, NWRT).

**Format** *COPY* (no parameters)

All ALPHA features except data card handling and event input / output are switched off. User routines are never called. Most data cards not referring to event input / output are ignored. Therefore, if the COPY card is used, any ALPHA program (Fortran code or load module) can serve as a simple copy job which digests the standard ALPHA data cards.



## 4.2 ALPHA Process cards

To reduce processing time, certain categories of objects can be excluded from ALPHA analysis (i.e. the ALPHA variables will not be filled).

<b>NOMC</b>	no Monte Carlo “truth”
<b>NOCH</b>	no CHarged tracks (also excludes V0s)
<b>NOEM</b>	no Error Matrix for charged tracks
<b>NOV0</b>	no V0s
<b>NOCO</b>	no CalOrimeters
<b>NOPC</b>	no NEutral OBjects (from PCPA)
<b>NOGA</b>	no GAMpec objects (from EGPC)
<b>NONE</b>	no ALPHA banks will be filled. This option is useful if you don’t want to use any of ALPHA’s “track” and vertex sections, but you want to use ALPHA to do all of the I/O and bank unpacking.

## 4.3 UNPK: POT / DST unpacking

Unpacking of POT / DST banks is performed automatically. To save time, coordinates and some other banks are normally NOT unpacked. The default unpack options can be modified with the UNPK card.

**Format** *UNPK ‘ab cd ef ... ’*

The two–character options have the following meanings:

<b>AL</b>	all banks are unpacked but no coordinate sorting is done
<b>VD</b>	VDET coordinates
<b>IT</b>	ITC coordinates
<b>TP</b>	TPC coordinates
<b>TE</b>	dE/dx
<b>EC</b>	ECAL (electron id. )
<b>HC</b>	HCAL
<b>MU</b>	Muons
<b>FI</b>	track fits
<b>SO</b>	to sort coordinates in phi to redo pattern recognition

**CR** cal. object relationship banks  
' ' NO unpacking

The default options correspond to the card: *UNPK 'TE EC HC MU FI'*; TPC and ITC coordinates are not unpacked by default.

## 4.4 READ: Input from different card files

The READ card allows input cards to be read from different card files.

**Format** *READ 'card-file-name'*

The default file format is CARDS.

Card files may contain any number of READ cards. Files specified on a READ card may contain other READ cards. Recursive READ cards (file Z contains a READ 'Y' card, and file Y a READ 'Z' card) are not allowed.

Note that each card file specified with a READ card must end with an ENDQ card.

## 4.5 DEBU: Debug output

There are two debug levels:

**DEBU 0** minimum debug output (no BOS summary and no particle table printed).  
**DEBU 1** (default) Print BOS statistics and particle table summary at the end of the job. Print a message for each step in the ALPHA initialization and termination.

The debug level is available as the variable KDEBUG.

## 4.6 TIME: Job time control

**TIME 5** causes program termination (CALL QMTERM) if less than 5 seconds are available.

**Remarks** If no TIME card is given, 15 seconds is assumed by default. The number on the TIME card must be given WITHOUT a decimal point. In ALPHA, it is converted to a floating point number and is available as the variable QTIME (see 6.5.5). On all CERN computers, time is counted in IBM 370/168 seconds.

## 4.7 Histograms

The cards used in connection with the histogram package are described in detail in Chapter 5. For completeness, the cards are listed here also.

### 4.7.1 HIST: Write histogram file

The HIST card must be supplied to write histograms and Ntuples to a histogram file which can be edited / modified / analyzed in a subsequent interactive session (PAW).

**Format** *HIST* 'data-set-name | parameters'

**data set name**            see 4.1.1.

**Default file format**    HIS

**parameters:**            (optional – described in 5.2.1)

**UPDA**

**NOOV**

### 4.7.2 HTIT: General histogram title

The HTIT card corresponds to the HBOOK routine HTITLE; it assigns a general title to all histograms.

**Format:**            *HTIT* 'This is the general title'

### 4.7.3 NOPH: Histogram Printing

Including the NOPH card suppresses the printing of HBOOK histograms to the terminal or log file; histograms will still be written to a direct access file if the HIST card was used.

**Format:**            *NOPH*

## 4.8 FIEL: Magnetic field

Magnetic field can be set to a given value:

**FIEL 15.**            Set magnetic field to 15 KGauss.

## 4.9 FRF0: Use track fit without vertex detector

If the FRF0 card is included, the FRFT bank with NR=0 (which has track parameters found without hits from the vertex detector) will be used to fill the charged track variables rather than FRFT NR=2. Only FRFT NR=2 is available on the MiniDST.

## 4.10 Weight factors for calorimeters

Weight factors for the 3 ECAL stacks can be given by the data card

**CEEW 1. 1. 1.** Set weight factors to 1. for each stack (default).

A weight factor for the HCAL stack can be given by the data card

**CHEW 1.** set weight factor to 1. for HCAL (default).

## 4.11 EFLW and EFLJ: Energy Flow

The EFLW card enables the filling of energy flow objects in ALPHA (see Ch. 11). By default, the EFLW card selects the ENFLW (Janot) energy flow package. Using the same card with option 2:

**EFLW 2**

will select the mask energy flow algorithm. The mask energy flow is not available on the MiniDST.

Replacing the EFLW card with the EFLJ card causes ALPHA to store jets based on energy flow objects in addition to the energy flow objects themselves (see Ch. 11).

## 4.12 Particle table

The cards used in connection with the ALPHA particle table are described in detail in Chapter 15. For completeness, the cards are listed here also.

#### 4.12.1 PMOD: Modify particle attributes

**Format** *PMOD* 'part-name antipart-name' mass charge life-time width

**Parameters:**

'part-name antipart-name' see 15.3. The attributes of a particle and its antiparticle are modified at the same time. If a particle is its own anti-particle, the same name has to be given twice.

**mass charge life-time width:** Real numbers (with decimal point). The charge of the antiparticle is set to -charge. If less than four numbers are given, the remaining particle attributes are not changed.

#### 4.12.2 PNEW: New particles

Modify attributes of an existing particle.

**Format** *PNEW* 'part-name antipart-name' mass charge life-time width

Same parameters and format as PMOD; used to create new particles.

#### 4.12.3 PTRA: Modify particle names in the MC particle table

The PTRA card can be used to assign an arbitrary particle name to a specific MC integer code.

**Format** *PTRA* 'part-name antipart-name' iMCcode iMCanticode

**Parameters:**

'part-name antipart-name' see 15.4.3. denote the names for the particle and its antiparticle which have to be used inside the ALPHA program.

**iMCcode:** integer particle code used in the MC generator (WITHOUT decimal point and NOT included in apostrophes.)

**iMCanticode:** integer particle code used by the MC generator for the corresponding antiparticle.

### 4.13 SYNT: Syntax Check

The general structure of the BOS card reading routines does not allow for a thorough syntax check of data cards. To prevent long jobs from dying as a result of syntax errors, ALPHA provides a facility to check the data cards. If the data card

**SYNT** is given, then

- all data cards are read in;
- the existence (or, if required, the non-existence) of all input/output files is checked;
- NO files (except the log file) are created or modified even if the log file indicates otherwise;
- NO events are processed.

## Chapter 5

### Creating Histograms and Ntuples

The standard histogram package in ALPHA is HBOOK4. If you don't want to use HBOOK, the only system routines which are called automatically and which refer to HBOOK are the histogram initialization / termination routines QUIHIS and QUTHIS (3.5.3 and 3.5.4). Some utility routines which simplify calls to HBOOK routines or provide additional protection against deleting existing histograms are described below. Histogram output is directed by entries in the card file, and is described in section 5.2.

#### 5.1 Booking and Filling Histograms/Ntuples

All of these routines call standard HBOOK4 routines.

##### 5.1.1 Book a 1-dimensional histogram

```
CALL QBOOK1 (ID, CHTITL, NX, XMI, XMA, VMX)
```

The arguments are the same as for CALL HBOOK1 (...):

##### Input arguments:

<b>ID</b>	histogram ID number – nonzero integer
<b>CHTITL</b>	histogram title – character variable up to 80 characters
<b>NX</b>	number of bins
<b>XMI</b>	lower edge of lowest bin
<b>XMA</b>	upper edge of highest bin
<b>VMX</b>	normally set equal to 0.– see HBOOK manual for details.

HBOOK1 always deletes an existing histogram and creates a new one. To make it possible to update existing histograms (see 4.7.1), QBOOK1 creates a new histogram only if it does not yet exist. An existing histogram remains unchanged. Therefore, whenever you want to update histogram files, use QBOOK1 instead of HBOOK1. For new histograms, QBOOK1 and HBOOK1 are identical.

### 5.1.2 Book a 2–dimensional histogram

**CALL QBOOK2 (ID, CHTITL, NX, XMI, XMA, NY, YMI, YMA, VMX)**

QBOOK2 includes the same features as QBOOK1. The arguments are the same as for CALL HBOOK2 (...):

#### Input arguments:

<b>ID</b>	histogram ID number – nonzero integer
<b>CHTITL</b>	histogram title – character variable up to 80 characters
<b>NX</b>	number of bins in X
<b>XMI</b>	lower edge of lowest X bin
<b>XMA</b>	upper edge of highest X bin
<b>NY</b>	number of bins in Y
<b>YMI</b>	lower edge of lowest Y bin
<b>YMA</b>	upper edge of highest Y bin
<b>VMX</b>	normally set equal to 0.– see HBOOK manual for details.

### 5.1.3 Book an Ntuple

**CALL QBOOKN (ID, CHTITL, NVAR, TAGS)**

The arguments are NOT the same as for CALL HBOOKN (...):

#### Input arguments:

<b>ID</b>	Ntuple ID number – nonzero integer
<b>CHTITL</b>	Ntuple title – character variable up to 80 characters
<b>NVAR</b>	number of variables
<b>TAGS</b>	name of character array of dimension NVAR containing names for variables to be stored.

**CALL QBOOKN (ID,CHTITL,NVAR,TAGS)**

corresponds to :

**CALL HBOOKN (ID,CHTITL,NVAR,'ALPHA',1024,TAGS).**

'ALPHA' is the ZEBRA directory name referring to the file given on the HIST card (5.2.1). See 5.1.1 (QBOOK1 vs. HBOOK1) : Existing Ntuples will not be overwritten (see 5.1.1).



#### 5.1.4 Book an Ntuple with run, event number

```
CALL QBOOKR (ID, CHTITL, NVAR, TAGS)
```

The arguments are the same as for CALL QBOOKN (...). QBOOKR books a Ntuple with NVAR+2 variables. The two additional variables contain the run and event number. TAGS consists of NVAR array elements. Two tags KRUN and KEVT are appended automatically.

#### 5.1.5 Fill Ntuple plus run, event number

```
CALL QHFR (ID, A)
```

Fills the Ntuple ID with the array A and with run and event number. The arguments are the same as for HFN (ID, A). KRUN and KEVT are filled as variables NVAR+1 and NVAR+2 (see QBOOKR).

#### 5.1.6 Fill Ntuple with many variables

```
CALL QHFN (ID, A1, A2, A3, ..., An)
```

Fills the Ntuple ID with the variables A1 ... An ( $n < 50$ ). CALL QHFN (ID, A1, A2) corresponds to

```
DIMENSION A(50)
A(1) = A1
A(2) = A2
CALL HFN (ID, A)
```

#### 5.1.7 Fill Ntuple with many variables plus run, event number

```
CALL QHFNR (ID, A1, A2, A3, ..., An)
```

Fills the Ntuple ID with the variables A1 ... An ( $n < 50$ ; see QHFN) and with run / event number as variables n+1 and n+2 (see QHFR).

### 5.1.8 Sample ALPHA program to book and fill histogram, Ntuple

The following example books and fills a histogram and Ntuple. See Chapters 7 and 8 for explanations of the ALPHA variables used.

```
      SUBROUTINE QUNIT
      CHARACTER*4 TAGS(2)
      DATA TAGS/'ECHG','NTRK'/
C--- Book histogram to store momentum distribution for all charged
C--- tracks.
      CALL QBOOK1(1,'Momentum',100,0.,50.,0.)
C--- Book Ntuple to store charged energy and number of charged tracks
C--- per event.
      CALL QBOOKN(1000,'Event parameters',2,TAGS)
      END
      SUBROUTINE QUEVNT (QT,KT,QV,KV)
C-----
      INCLUDE 'PHYINC:QCDE.INC'           !VAX
      DIMENSION QT(KCQVEC,1), KT(KCQVEC,1), QV(KCQVRT,1), KV(KCQVRT,1)
      INCLUDE 'PHYINC:QMACRO.INC'        !VAX
C-----
      IF(KNCHT.EQ.0)RETURN
      ECHRG=0.
C
C--- sum energy; histogram track momentum
C
      DO 20 IT=KFCHT,KLCHT
          ECHRG=ECHRG+QE(IT)
          CALL HF1(1,QP(IT),1.)
20 CONTINUE
      CALL QHFN(1000,ECHRG,FLOAT(KNCHT))
      END
```

## 5.2 Histogram output – the ALPHA cards file

### 5.2.1 HIST: Write histogram file

Unless the NOPH card is included in the card file (see below), 1– and 2–dimensional histograms are written to the log file in the program termination phase (i.e., after return from QUTERM; see 3.4 and 3.5.4).

The HIST data card is necessary for writing histograms and Ntuples to a histogram file which can be used in a subsequent interactive session (PAW).

**Format** *HIST* 'data-set-name | parameters'

**data set name**            see 4.1.1.

**Default file format**    HIS

**parameters (optional) :**

**UPDA**                    Update existing histograms. Can be used deliberately if a previous job terminated due to time limit etc. but ...

**CAUTION** with this option, the old histogram file will be overwritten (even on VAX).

**NOOV**                    Overwrite protection (see 4.1.3). Cannot be used with UPDA.

**On VAX :**            Unnecessary.

**On IBM :**            Strongly recommended. The first HBOOK action is to clear an existing file unless UPDA is specified.

The histogram file is a direct access file in machine-dependent format as required for PAW input. The command files RTOX and RFRX transform those files into machine-independent sequential files (file type EXCH) and vice versa. (The histogram files returned from the CRAY are in EXCHange format; they must be converted to direct access files with RFRX before they can be used with PAW.)

Only one histogram file can be specified using the HIST card. If you need several output files, use the standard HBOOK4 input / output routines and book Ntuples with different ZEBRA directory names. The directory name used by ALPHA is '//ALPHA'.

### 5.2.2 NOPH: Histogram Printing

Including the NOPH card in the card file will suppress the printing of HBOOK histograms to the terminal or log file; histograms will still be written to a direct access file if the HIST card is used.

**Format :**            *NOPH*

### 5.2.3 HTIT: General histogram title

The HTIT card assigns a general title to all histograms; it corresponds to the HBOOK routine HTITLE.

**Format :**            *HTIT 'This is the general title'*

## Chapter 6

### Mnemonic symbols

Mnemonic symbols are Fortran variables, arrays, parameters, functions, or statement functions. Mnemonic symbols which give access to information for specific reconstructed or Monte Carlo objects are described in Chapter 8. When possible, the names of the mnemonic symbols follow the same convention as the HAC parameters.

The units used in ALPHA are cm, sec, GeV, GeV/c, GeV/c<sup>2</sup>, kG.

#### 6.1 Mathematical and physical constants

<b>QQE</b>	$e = 2.718282$
<b>QQPI</b>	$\pi = 3.141593$
<b>QQ2PI</b>	$2\pi$
<b>QQPIH</b>	$\pi/2$
<b>QQRADP</b>	$180/\pi$
<b>QQC</b>	speed of light = 2.997925E10 cm/sec
<b>QQIRP</b>	speed of light in units cm / KGauss (inverse track bending radius $\rightarrow$ track momentum)
<b>QQH</b>	Planck constant / $2\pi = 6.582173E-25$ GeV sec
<b>QQHC</b>	$QQH * QQC$

Note: The standard ALEPH constants (ALCONS) are also available.

#### 6.2 Run information

<b>KRINNE</b>	number of events in run (with HV on)
<b>KRINLF</b>	LEP fill number
<b>KRINDQ</b>	data quality (see bank description for RLUM variable RQ)
<b>QRINLU</b>	Luminosity (from database if available)

<b>KRINNZ</b>	number of $Z \rightarrow$ hadrons (from database if available)
<b>KRINNB</b>	number of Bhabhas (from database if available)
<b>QELEP</b>	LEP energy (from database if available)
<b>QMFLD</b>	magnetic field (best estimate). Taken from data card FIEL (if given), run header bank RALE, or run header bank AFID for MC events. If $ABS(QMFLD) > 20$ QMFLD is set to 15.
<b>QVXNOM, QVYNOM, QVZNOM</b>	run-by run position of interaction point used to calculate the distance between tracks and interaction point; taken from the database bank LFIL. (See KBPSTA below.)
<b>QVXNSG, QVYNSG, QVZNSG</b>	(Statistical error) <sup>2</sup> on QVXNOM etc.; not the beam width.
<b>KBPSTA</b>	Beam Position Status: method used to determine QVXNOM, QVYNOM, etc. <ul style="list-style-type: none"> <li>• KBPSTA = 0: No mean beam position for the run</li> <li>• KBPSTA = 1: Mean beam position computed with VDET, per RUN (this is the status of most of the runs in 1991)</li> <li>• KBPSTA = 2: Mean beam position computed with VDET, per FILL (this happens when the run has too few events)</li> <li>• KBPSTA = 3: Mean beam position computed WITHOUT VDET, per FILL (this is the status of all runs in 1989/1990 and of the runs in 1991 which do not have a working VDET)</li> </ul>
<b>QDBOFS</b>	average systematic offset of D0; see 8.1.4.

## 6.3 Event information

### 6.3.1 Event header: from bank EVEH

<b>KEXP</b>	Exp Number
<b>KRUN</b>	Run Number
<b>KEVT</b>	Event Number
<b>KEVERT</b>	Run Type
<b>KEVEDA</b>	DAte
<b>KEVETI</b>	TIme
<b>KEVEMI(I)</b>	trigger Mask I, I = 1 to 4
<b>KEVETY</b>	event TYpe
<b>KEVEES</b>	Error Status

### 6.3.2 Event directory information

<b>KCLASW</b>	Event directory classification word
---------------	-------------------------------------

### 6.3.3 Event generator status: from bank KEVH

**KKEVID** process ID  
**QKEVWT** WeighT

### 6.3.4 Detector HV status: from banks REVH, LOLE

**XHVTRG** =.TRUE. if XLUMOK checks are satisfied. (To save time, XHVTRG should be used rather than the function XLUMOK.)  
**KREVDS** Detector status word from REVH bank  
**XVLCAL** = .TRUE. if the LCAL is OK (i.e., the LOLE bank is present and there is no error condition)  
**XVSATR** = .TRUE. if SATR HV is OK  
**XVITC** = .TRUE. if ITC HV is OK  
**XVTPC** = .TRUE. if TPC HV is OK (bit 15)  
**XVTPCD** = .TRUE. if all TPC HV is OK (dE/dx bit – bit 4)  
**XVECAL** = .TRUE. if ECAL HV is OK (i.e., all ECAL HV bits are on)  
**XVHCAL** = .TRUE. if HCAL HV is OK (i.e., all HCAL HV bits are on)

Note: For VDET HV status, see Sec. 12.4.2.

### 6.3.5 Trigger Information: from XTEB or XTRB, XTCN

**KXTET1** Level 1 trigger bit pattern  
**KXTET2** Level 2 trigger bit pattern  
**KXTEL2** Level 2 bit pattern after applying the enabled trigger mask  
**KXTCGC** Number of GBXs since the last event readout  
**KXTCLL** Number of level 1 yes conditions since the last event readout  
**KXTCBN**  $e^-$  bunch number  
**KXTCCL** level 1 control word  
**KXTCHV** HV status word (equivalent to KREVDS above)  
**KXTCEN** mask of enabled triggers

### 6.3.6 General event information: from bank DHEA

<b>KDHEFP</b>	Flag for Physics identification
<b>KDHENX</b>	Number of reconstructed vertices
<b>KDHENP</b>	Number of positive reconstructed tracks
<b>KDHENM</b>	Number of negative reconstructed tracks
<b>KDHENV</b>	Number of reconstructed V0's
<b>KDHENJ</b>	Number of reconstructed jets
<b>QDHEEC</b>	total Energy of Charged tracks
<b>QDHEEL</b>	total Energy of CaL objects
<b>QDHEPF</b>	abs(P) of energy Flow
<b>QDHETH</b>	THeta of energy flow
<b>QDHEPH</b>	PHi of energy flow
<b>QDHEEF</b>	Energy Flow
<b>QDHEET</b>	abs value of Et
<b>QDHET1</b>	Theta of momentum tensor axis 1
<b>QDHEP1</b>	Phi of momentum tensor axis 1
<b>QDHET2</b>	Theta of momentum tensor axis 2
<b>QDHEP2</b>	Phi of momentum tensor axis 2
<b>QDHEE1</b>	1st Eigenvalue of momentum tensor
<b>QDHEE2</b>	2nd Eigenvalue of momentum tensor
<b>QDHEE3</b>	3rd Eigenvalue of momentum tensor

Note: The energy flow results in this bank are based on the mask energy flow algorithm run in JULIA.

### 6.3.7 Beam position from BOM system: from bank BOMB

<b>QVXBOM</b>	$x$ beam position from BOM
<b>QVYBOM</b>	$y$ beam position from BOM
<b>KERBOM</b>	Error code for BOM

- $< 0$  fatal error, QVXBOM and QVYBOM are filled with QVXNOM and QVYNOM
- $= 0$  BOM data good
- $= 1$  BOM data in  $x$  disagrees with VDET average
- $= 2$  BOM data in  $y$  disagrees with VDET average
- $= 3$  BOM data in both  $x$  and  $y$  disagrees with VDET average

## 6.4 ECAL Wire Energies

**QEECWI(IMOD)** ECAL wire energy for module IMOD in GeV. Modules 1 – 12 refer to endcap A, 13 – 24 to the barrel, and 25 – 36 to endcap B.

## 6.5 ALPHA Internal Constants, Variables

### 6.5.1 Event counts

**KNEVT** Total number of events read in  
**KNEFIL** Number of events read from the current input file  
**KNREIN** Number of records read from the current input file (including run records)  
**KNEOUT** Number of events written to the output file

### 6.5.2 Program status

**KSTATU** -1: program initialization; 0: event processing; 1: program termination  
**KDEBUG** debug level (see 4.5)

### 6.5.3 Event status

**XMINI** = **.TRUE.** if event read from Mini-DST = **.FALSE.** if event read from POT or DST  
**XMCEV** = **.TRUE.** if MC truth available for the event

### 6.5.4 Input / output units

**KUINPU** event input = 20, 21  
**KUOUTP** event output = 50  
**KUEDIN** event directory input = 30  
**KUEDOU** event directory output = 60  
**KUCONS** data base = 4  
**KUPRNT** log file = 6  
**KUPTER** terminal = 76 or 0 (see Ch. 2)  
**KUCARD** card input = 7  
**KUCAR2** second card input = 8



**KUHIST**      histogram output unit = 15  
**KURTOX**      EXCH format histogram output on CRAY = 16

Note: Units 90, 91, 92, and 93 are always free for private output files.

### 6.5.5 Timing

**QTIMEL**      time remaining before time limit  
**QTIME**      seconds given on the TIME card (see 4.6)

On all CERN computers, the time units are IBM 370/168 seconds.

### 6.5.6 Character variables

**CQVERS**      ALPHA version number (6 digits)  
**CQDATE**      date at start of job (8 char)  
**CQTIME**      time at start of job (8 char)  
**CQFOUT**      data set name of event output file = ' ' if no output file given

## Chapter 7

### ALPHA “Tracks” and “Vertices”

Before QUEVNT is called for each event, ALPHA fills its own data structure with information from the event. Each “tracklike” object (eg., tracks, calorimeter objects, energy flow objects, etc.) is assigned a unique number. (A “tracklike” object is any object which can be described with a 4 vector.) This ALPHA “track” number is equal to the JULIA “track” number + a constant. Unique ALPHA numbers are also assigned to vertices (reconstructed vertices and Monte Carlo vertices). The constant is introduced in order to obtain a unique numbering scheme for all species of “tracks” or vertices (in JULIA and GALEPH, different species start with the number 1). In the description below, ITK always refers to an ALPHA “track” number and IVX to an ALPHA vertex number.

The properties of the tracks and vertices are found using functions which refer to the ALPHA “track” and vertex numbers. For example, the energy of ALPHA “track” ITK is QE(ITK). The properties available for each tracklike object and each vertex are described in sections 8.1 and 8.3, respectively.

In the following sections, several methods for determining ALPHA track and vertex numbers are described. All of these methods can be nested. Functions which give simple access to relationships between different types of objects are also described.

#### 7.1 Access by Fortran DO loops

In ALPHA, fortran DO loops can be used to loop over most types of objects. For each type of object, three variables are defined: KFxxx, KLxxx, KNxxx. xxx represents the type of object. The last letter of the variables is either T (tracklike) or V (vertex). DO loops must be made from KFxxx to KLxxx; KNxxx is the number of objects of type xxx.

For example, the following three lines will make a histogram of the momentum spectrum of charged particles.

```
DO 10 ITK = KFCHT, KLCHT
    CALL HF1 (47,QP(ITK),1.)
10 CONTINUE
```

**KFCHT, KLCHT** number of first (last) charged track.

**ITK** loop index = ALPHA track number

**QP(ITK)** momentum of track ITK (see 8.1.1)

The number of charged tracks is given by the variable KNCHT. KNCHT stands for KLCHT - KFCHT + 1. Therefore, if KNCHT = 0, KLCHT = KFCHT - 1.

The objects which can be accessed with these DO loops are listed in the following two sections.

### 7.1.1 ALPHA “TRACKS”

**Charged Tracks: KFCHT, KLCHT, KNCHT** If the FRF0 card is present in the ALPHA cards file, the NR=0 version of the FRFT bank (track parameters determined without vertex detector coordinates) will be used. Otherwise, the NR=2 version of FRFT (TPC + ITC + VDET tracks) will be used. Only FRFT NR=2 tracks are available on the MiniDST.

**Calorimeter Objects: KFCOT, KLCOT, KNCOT** Calorimeter objects can be any of the following:

- ECAL objects with no associated HCAL object.<sup>1</sup>
- HCAL objects with no associated ECAL object.
- Composite cal. objects consisting of at least one ECAL and HCAL object associated to each other. See 7.2 for getting access to the contributing ECAL and HCAL objects separately. See the end of Section 7.3 for a more detailed description of composite cal. objects in ALPHA.

Calorimeter objects can be further divided into:

**ISolated cal. objects: KFIST, KLIST, KNIST** Cal objects with NO associated charged track.

**ASsociated cal. objects: KFAST, KLAST, KNAST** Cal objects with one or more associated charged track.

**“REconstructed” objects: KFRET, KLRET, KNRET** “REconstructed” objects are:

- Charged tracks;
- Calorimeter objects (see above) which are NOT associated to charged tracks (ISolated cal. objects).

**Reconstructed V0s: KFV0T, KLV0T, KNV0T** See Section 7.5.1 for comments on the daughters of V0s.

**Tracks from DeCay Vertices: KFDCT, KLDCT, KNDCT** Charged tracks outgoing from reconstructed DeCay vertices. The momenta for these tracks are calculated relative to the secondary vertex position. Currently, this section includes the daughter tracks from reconstructed V0s.

**Energy Flow objects: KFEFT, KLEFT, KNEFT** This section includes selected charged tracks and ECAL and HCAL clusters remaining after subtracting track energies. These objects may also be accessed with their particle name ‘EFLW’ using the functions KPDIR and KFOLLO (described in 7.4). This section is not filled unless the EFLW card is included in the card file (see Ch. 11);

---

<sup>1</sup>For ECAL wire energies, see 6.4.

**Neutral Calorimeter Objects: KFNET, KLNET, KNNET** Neutral objects derived from the PCPA bank. These objects may also be accessed with their particle name 'NEOB' using the functions KPDIR and KFOLLO (described in 7.4).

**Photons from GAMPEC: KFGAT, KLGAT, KNGAT** These objects may also be accessed with their particle name 'GAMP' using the functions KPDIR and KFOLLO (described in 7.4).

**Jets from EJET: KFJET, KLJET, KNJET** Jets based on EFLW objects using QJMMCL with YCUT = 0.003. These objects may also be accessed with their particle name 'EJET' using the functions KPDIR and KFOLLO (described in 7.4). They may be used as input for jet finding with a higher YCUT (see 10.11.1 and 11.1).

**Monte Carlo particles ("truth") KFMCT, KLMCT, KNMCT**

### 7.1.2 ALPHA VERTICES:

**REconstructed Vertices: KFREV, KLREV, KNREV** Currently, this category includes only the main vertex (the first vertex, KFREV) and V0s.

**Monte Carlo vertices ("truth"): KFMCV, KLMCV, KNMCV**

ALPHA "tracks" and vertices are stored in the banks QVEC and QVRT, respectively.

## 7.2 Loops over ECAL and HCAL objects

If ECAL and HCAL objects are topologically associated to each other, the loops described above give access to composite calorimeter objects rather than to each contributing ECAL and HCAL object separately. It is also possible to get access to all ECAL and HCAL objects, regardless of whether or not they are associated to other reconstructed objects. (The loops described below are equivalent to looping through the PECO and PHCO banks.)

The following statements perform a loop over all ECAL objects; see 7.4. (DO loops cannot be used because the objects are not stored in consecutive locations.)

```
      IOBJ = KPDIR ('ECAL', KRECO)
10 IF (IOBJ .EQ. 0) GO TO 999
C...      Analysis of the ECAL object IOBJ ...
      IOBJ = KFOLLO (IOBJ)
      GO TO 10
```

(The functions KPDIR and KFOLLO are described in 7.4.) The corresponding loop for HCAL objects is:

```
      IOBJ = KPDIR ('HCAL', KRECO)
10 etc ...
```

### 7.3 Relationships between objects in different subdetectors

The JULIA program provides relationships between objects reconstructed in the various detector components if they are topologically associated to each other. These relations are available in ALPHA and can be used for charged tracks, ECAL objects, HCAL objects, and composite calorimeter objects. (Below, IOBJ is any ALPHA “track” number referring to a charged track, cal. object, ECAL object, or HCAL object.)

**KNCHGD (IOBJ)** Number of charged tracks associated to IOBJ.

**KCHGD (IOBJ, N)** The Nth charged track associated to IOBJ.

For example,

```
IOBJ = ... any calorimeter object ...
DO 10 N = 1, KNCHGD (IOBJ)
  ICHGD = KCHGD (IOBJ, N)
C ... analysis of a charged track ICHGD associated to IOBJ ...
10 CONTINUE
```

Note: If IOBJ in the example above is a charged track itself, then KNCHGD (IOBJ) is 1 and KCHGD (IOBJ,1) gives IOBJ. Similarly:

**KNECAL (IOBJ)** Number of ECAL objects associated to IOBJ.

**KECAL (IOBJ, N)** The Nth ECAL object associated to IOBJ.

**KNHCAL (IOBJ)** Number of HCAL objects associated to IOBJ.

**KHCAL (IOBJ, N)** The Nth HCAL object associated to IOBJ.

The relation from a composite calorimeter object ICOMP to each of its contributing ECAL and HCAL object is provided by the relations described above: KECAL (ICOMP,N) and KHCAL (ICOMP,N). In addition, the composite object is treated as “mother” of the contributing ECAL and HCAL objects, so the mother–daughter or daughter–mother relation described in 7.5 can be used for all calorimeter objects.

Note that the composite calorimeter objects in ALPHA are not identical to those in the PCRL bank. ALPHA composite calorimeter objects include at most one HCAL object, while the PCRL objects may include many HCAL objects. ALPHA starts with each HCAL object and adds the ECAL objects that are associated to it. If an ECAL object is associated to more than one HCAL object, its energy is divided equally among the HCAL objects.

## 7.4 Direct access to particles

### 7.4.1 Particle name and class

In addition to the loops described above, it is possible to access particles by their name. In many cases, this method is faster and the code is easier to read than the standard loops described in 7.1. Two quantities must be specified:

- The particle name (example: 'E+' or 'GAMMA'); see 15.1.
- The object (= track = particle) class which distinguishes between reconstructed tracks, the Monte-Carlo truth, and any Lorentz frame derived from one of them:
  - Class KRECO: Reconstructed objects read from the event input file and everything derived from them except Lorentz boosted objects.
  - Class KMONTE: Monte-Carlo truth.
  - Each Lorentz frame is considered as its own class (see 9.5). These classes are denoted by the number of the object which defines the Lorentz rest frame.

KRECO and KMONTE are available everywhere as integer Fortran parameters. Their actual values are  $-1$  and  $-2$ , respectively. Positive integers denote Lorentz frames. Integers less than  $-2$  can be used to create your own particle classes (see KVSAVC in 9.2.12).

The particle name of MC particles is specified in the MC particle table (see 15.1). Reconstructed objects have the names 'CHARGED', 'ECAL', 'HCAL', 'CALOBJ', 'EFLW', 'NEOB', and 'GAMP' for charged tracks, ECAL objects, HCAL objects, unspecified (*e.g.*, composite) calorimeter objects, energy flow objects, neutral calorimeter objects, and GAMPEC photons, respectively. The functions KVSAVE, KVSAVC, and KIDSAV (see 9.2.11) can be used to create new tracks with a name. A list of standard particle names is given in App. E. New particle names can be introduced by using them in ALPHA subroutine calls or by specifying them on data cards (see 15.4.2).

### 7.4.2 Example: Loop over all MC generated positrons

```
      ITK = KPDIR ('E+', KMONTE)
10    IF (ITK .EQ. 0) GO TO 90
C     ... e+ analysis ...
      ITK = KFOLLO (ITK)
      GO TO 10
90    CONTINUE ...
```

**KPDIR** ('particle-name', ICLASS) '*particle-name*': Character string (1 to 12 characters).  
*ICLASS*: Track class (see 7.4.1): KRECO or KMONTE or a track number  
ITKRST if ITKRST has been used before to define the rest frame for a Lorentz boost (see 9.5).

**KFOLLO** (ITK) The following particle with the same particle name in the same class.

**Remarks:** The term “FOLLOWing” refers to some arbitrary ordering. Lower case characters in particle names are translated to upper case. It is safest, however, to use only upper case characters with ALPHA.

### 7.4.3 Particle name versus integer particle code – time consumption

Using character particle names in function calls makes the code easier to read, but it implies a lookup in a table. Although the lookup is fast, in nested loops it may be desirable to save this time. Consequently, some (not all) functions are provided in two versions: one which expects the particle name as an argument and another which expects the corresponding integer particle code and thus saves the lookup time. The second version is denoted by a “C” (= “Code”) as the 2nd character of the function name.

Using integer particle codes, the example given in section 7.4.2 becomes:

```
C ... somewhere in the job or subroutine initialization:
  IP = KPART ('E+')
C ...
  ITK = KCDIR (IP, KMONTE)
10  IF (ITK .EQ. 0) GO TO 90
C   ... analysis of the e+ ...
  ITK = KFOLLO (ITK)
  GO TO 10

90  CONTINUE ...
```

**IP = KPART('name')** must be called before IP is used. The particle name is the basic reference to a particle. The integer code may change from one job to another.

**KCDIR (IP, ICLASS)** First particle with the given particle code in class ICLASS.

### 7.4.4 Loops over a particle and its antiparticle

The particle table contains the relation between particles and antiparticles, so loops over particles (or systems of particles) and their corresponding (systems of) antiparticles can be performed easily.

**Example:** Loop over MC – generated e+ and e–:

```
DO 90 IANTI = 0,1
  ITK = KPDIRA ('E+', KMONTE, IANTI)
10  IF (ITK .EQ. 0) GO TO 90
C   ... analysis of the e+ or e- ...
  ITK = KFOLLO (ITK)
  GO TO 10

90  CONTINUE ...
```

**KPDIRA** ('particle-name', ICLASS, IANTI) If IANTI=0, KPDIRA returns the first particle with the given name in the class ICLASS. If IANTI is not equal to 0, the first corresponding antiparticle is given.

To use the integer particle code (see 7.4.3), replace

KPDIRA ('E+', KMONTE, IANTI) with KCDIRA (IP, KMONTE, IANTI).

#### 7.4.5 Analysis of particle systems: Examples

Systems of particles can be analyzed by nesting loops with KPDIR and KPDIRA. The two examples given below illustrate cases in which care must be taken to avoid multiple counting of the same particle combinations.

##### Combinations of the same particles: $\pi^+ \pi^+$

```
C---First select pion candidates
  DO 5 ITK=KFCHT,KLCHT
    IF(condition to select pions) THEN
      ISAVE=KIDSAV(ITK,'PI+')
    ENDIF
  5 CONTINUE
C---Loop over selected pions.
  IPIONE = KPDIR ('PI+', KRECO)
  10 IF (IPIONE .NE. 0) THEN
    IPITWO = KFOLLO (IPIONE)           <--- important
  20  IF (IPITWO .NE. 0) THEN
    ... analysis of the pi+ pi+ system ...
    IPITWO = KFOLLO (IPITWO)
    GO TO 20
  ENDIF
  IPIONE = KFOLLO (IPIONE)
  GO TO 10
ENDIF
```

The 2nd  $\pi^+$  (IPITWO) has to be initialized with KFOLLO and NOT with KPDIR. See section 9.2.11 for the use of KIDSAV.



## $\Delta^{++} \rightarrow p \pi^+$

Proton and pion candidates must be selected and saved with KVSAVE or KIDSAV before this code is reached (see 9.2.9).

```
      IPROT = KPDIR ('P', KRECO)
10  IF (IPROT .NE. 0) THEN
      IPIPLU = KPDIR ('PI+', KRECO)
20  IF (IPIPLU .NE. 0) THEN
      IF (.NOT.XSAME(IPROT,IPIPLU)) THEN      <--- important
C      ... analysis of the p pi+ system ...
      ENDIF
      IPIPLU = KFOLLO (IPIPLU)
      GO TO 20
      ENDIF
      IPROT = KFOLLO (IPROT)
      GO TO 10
      ENDIF
```

The logical function XSAME (see 8.1.7) tests whether the two contributing particles are based on different reconstructed objects or simply on different mass hypotheses of the same reconstructed object.

## 7.5 Mother – daughter relationships

### 7.5.1 Mother to daughters

The connection from a mother to its daughters is available for MC particles and for composite particles established by the QVxxxx routines described in 9.2.2.

```
      IMOTH = ... (track number of a mother particle)
      DO 10 I = 1, KNDAU (IMOTH)
          IDAUGH = KDAU (IMOTH,I)
          CALL HFILL (47, QP(IDAUGH))
10  CONTINUE
```

**KNDAU (ITK)** number of daughters for track ITK = 0 if no daughter exists

**KDAU (ITK,I)** track number of Ith daughter

**Note for V0s:** The daughters of a V0 (section V0T) are stored in the DCT section (see 7.1). These tracks are copies of tracks in the CHT section, but their momenta are recalculated relative to the secondary vertex position. The function KCHT (see 7.6.2) returns the CHT track number corresponding to a track in the DCT section.

**Example:**

```
      DO 10 IVO=KFBVOT,KLVOT
c--- First daughter of V0 (in DCT section)
      I1DCT=KDAU(IVO,1)
C --- Corresponding track in CHT section.
      I1CHT=KCHT(I1DCT)
      10 CONTINUE
```

### 7.5.2 Daughter to mother(s)

The connection from a daughter to its mother(s) is available for MC particles and for daughters of “saved” composite particles (see KVSAVE in 9.2.9). The QVADDx routines (9.2.2) and the jet / event topology routines 10) do NOT set up this relation.

```
      IDAUGH = ... (track number of a daughter particle)
      DO 10 I = 1, KNMOTH (IDAUGH)
          IMOTH = KMOTH (IDAUGH,I)
          CALL HFILL (47, QP(IMOTH))
      10 CONTINUE
```

**KNMOTH (ITK)** Number of mothers of track ITK. Note that MC particles as read in from the event input file have no or one mother.

**KMOTH (ITK,I)** Track number of the Ith mother.

## 7.6 Access to the “same” object

The “same” object means:

- any copy of an object;
- for reconstructed tracks, the “same” object with different mass or vertex hypothesis;
- The “same” object boosted into any Lorentz frame.

### 7.6.1 Loops over copies of the “same” object using KSAME

Example:

```
      ITKSAM = KSAME (ITK)
      10 IF (ITKSAM .EQ. ITK) GO TO 90
C      ... analysis of the same object, e.g.: search for the object
C      in a specific Lorentz frame ITKRST (see >):
```

```

        IF (KCLASS (ITKSAM) .EQ. ITKRST) THEN
C      ...
        ENDIF
        ITKSAM = KSAME (ITKSAM)
        GO TO 10
    90 CONTINUE ...

```

**Remarks:** This loop is terminated if it arrives at the original track. KSAME never returns 0. The same particle can be boosted several times into the same Lorentz frame provided that the boosts are performed with different mass or other hypotheses (see 9.5.1); if you start with the original track ITK, the most recently boosted hypothesis is reached first.

### 7.6.2 Find original copy of a charged track

For copies of charged tracks, the function KCHT returns the original track number in the CHT section.

**KCHT (ITK)** If  $KFCHT \leq ITK \leq KLCHT$ , **KCHT (ITK)** is equal to ITK. Otherwise (i.e., ITK is a copy of a track in the CHT section), **KCHT (ITK)** equals the corresponding track number in the CHT section. This function can be used only for charged tracks; for other objects, use KSAME.

## 7.7 Match reconstructed tracks and MC truth

The relation between reconstructed and MC particles is not necessarily one-to-one. Therefore, a loop has to be constructed:

```

        ITK1 = ... (any given MC or reconstructed track number)
        DO 10 I = 1, KNMTCH(ITK1)
            IF (KSMTCH (ITK1,I) .LE. (min. required shared hits)) GO TO 10
            ITK2 = KMTCH (ITK1,I)
C      ...
    10 CONTINUE

```

If ITK1 is a reconstructed track then ITK2 is a matching MC track. If ITK1 is a MC track then ITK2 is a matching reconstructed track.

**KNMTCH (ITK)** Number of matching candidates for track ITK.

**KMTCH (ITK,I)** Track number of Ith matching particle.

**KSMTCH (ITK,I)** Number of shared hits between MC and reconstructed track.

**Remarks:**

- The match is performed on the basis of shared hits in the TPC and IPC.
- The correspondence between MC and calorimeter objects is stored in the POT banks PEMH and PHMH. This information will be made available in a future version of ALPHA.

## 7.8 Track – vertex relationships

**IVX = KORIV (ITK)** origin vertex of a track

**IVX = KENDV (ITK)** end vertex of a track

**ITK = KVINCP (IVX)** particle incoming to vertex IVX

To find the tracks outgoing from a vertex, the following loop must be performed:

```
      IVX = ... (vertex number; defined before)
      DO 10 I = 1, KVND AU (IVX)
        ITK = KVDAU (IVX,I)
        CALL HFILL (47, QP(ITK))
10    CONTINUE
```

**KVND AU (IVX)** number of outgoing tracks

**KVDAU (IVX,I)** track number of Ith outgoing track

## Chapter 8

### ALPHA Track and Vertex Attributes

Not all of the attributes listed in this chapter are available when using the Mini-DST. See Appendix D, as well as the Mini-DST User's Guide, for a list of variables which are filled from the MINI.

The units used throughout ALPHA are cm, sec, GeV, GeV/c, GeV/c<sup>2</sup>, kG.

#### 8.1 "Track" attributes.

These quantities are defined for all ALPHA "tracks" (e.g., charged tracks, cal. objects, MC truth, etc.) "I" always refers to the ALPHA "track" number.

##### 8.1.1 Basic attributes

<b>QP (I)</b>	P = momentum of vector I.
<b>QX (I)</b>	<i>x</i> momentum component
<b>QY (I)</b>	<i>y</i> momentum component
<b>QZ (I)</b>	<i>z</i> momentum component
<b>QE (I)</b>	Energy
<b>QM (I)</b>	Mass (use QMASV0 for V0 mass; see below)
<b>QCH (I)</b>	CHarge
<b>KCH (I)</b>	NINT (QCH(I)) (be careful with quarks)

For charged tracks, the pion mass is assumed; the mass can be changed with QVSETM (see 9.2.13). For angles and more kinematics quantities, see 9.1.

##### 8.1.2 V0 Mass

**QMASV0 (I,'name')** Mass of V0 with hypothesis 'name'

The function QMASV0(I,'name') provides the mass for a given V0 hypothesis, where 'name' is the name from the ALPHA particle table or the abbreviation listed here:

- 'K0S' or 'K0'
- 'Lam0' or 'LA'
- 'Lam#0' or 'AL'
- 'GAMMA' or 'GA'.

This function can be used only for  $KFV0T \leq I \leq KLV0T$ . See also QIDV0, Sec. 9.2.3.

### 8.1.3 Track error covariance matrix

<b>XSIG (I)</b>	<b>.TRUE.</b> if covariance matrix available
<b>QSIG (I,N,M)</b>	element (N,M) of the covariance matrix N,M = 1,2,3,4 in the order QX,QY,QZ,QE
<b>QSIGEE (I)</b>	Error <sup>2</sup> on energy
<b>QSIGE (I)</b>	Error on energy
<b>QSIGPP (I)</b>	Error <sup>2</sup> on momentum
<b>QSIGP (I)</b>	Error on momentum
<b>QSIGMM (I)</b>	Error <sup>2</sup> on mass
<b>QSIGM (I)</b>	Error on mass The mass error is not defined for particles with mass = 0.

QSIG (I,1,1) is set to -1 if the matrix is not available.

### 8.1.4 Distance to the beam position:

Available for charged reconstructed tracks.

<b>QDB (I)</b>	distance of closest approach to beam axis
<b>QDBS2 (I)</b>	error <sup>2</sup> on QDB
<b>QZB (I)</b>	z coordinate of track point where QDB is measured
<b>QZBS2 (I)</b>	error <sup>2</sup> on QZB
<b>QBC2 (I)</b>	$\chi^2$ due to QDB and QZB.

The coordinates of the beam position used for these values are QVXNOM, QVYNOM, and QVZNOM (see 6.2). The average value of QDB may have a small offset from zero as a result of systematic tracking errors. The offset QDBOFS (see 6.2), which is typically less than 50 microns, may be subtracted from QDB(I) in order to yield  $\langle QDB \rangle = 0$ .

For more geometrical track attributes, see sections 8.2.1 and 9.1.

### 8.1.5 Stability code

**KSTABC (I)**    Stability code

The stability code is designed to avoid double counting when making loops over Monte Carlo particles. The possible values of KSTABC are:

- 1**     Particle does not decay.
- 2**     Neutral particle that decays in the calorimeter volume. Charged particle that decays in the TPC or calorimeter volume. Here, TPC and calorimeter volumes are full cylinders (including the beam pipe region).
- 3**     One of the ancestors of this stable particle has interacted with matter. Energy and momentum are NOT conserved.
- 0**     Decay products of “stable” particles including all garbage in the calorimeter.
- 1**    Particle decays immediately (resonance etc.).
- 2**    Particle decays with finite decay length but before reaching the detector volume (see above).
- 3**    Particle interacts with matter before reaching the detector volume. The decay products do not conserve energy and momentum.

A loop over all MC particles with  $KSTABC > 0$  selects the generation of decay particles which will probably be visible in the detector – energy is never counted twice. The energy sum of these particles gives the total generated energy only if no particle interacted with matter inside the detector volume. A loop over MC particles with  $KSTABC = 1, 2,$  and  $-3$  is similar, but it always gives the generated total energy.

### 8.1.6 Test a particle’s name

**XPEQU (I, 'part-name')**     = **.TRUE.** if track I is a particle with the name ‘part-name’.

**XPEQOR (I, 'part-name')**    = **.TRUE.** if track I is a particle with the name ‘part-name’ or if it is the corresponding antiparticle.

**XPEQAN (I, 'part-name', IANTI)** = **.TRUE.** if track I is a particle with the name ‘part-name’ and if  $IANTI = 0$ . = **.TRUE.** if track I is the antiparticle corresponding to ‘part-name’ and if  $IANTI$  is not equal to 0.

The same functions exist for integer particle codes  $IPC = KPART$  (‘part-name’) instead of the particle names (see 7.4.3):

```
XCEQU (I, IPC)
XCEQOR (I, IPC)
XCEQAN (I, IPC, IANTI)
```

### 8.1.7 Test if particles are based on the same object

**XSAME (I,J)** = **.TRUE.** if tracks I and J or one of their daughters, granddaughters, etc. are based on the same object (see 7.6) or, in other words, belong to the same family (see 10.2.3). I and J must both be reconstructed tracks or MC particles; they may, however, belong to different Lorentz frames. XSAME uses the same bit masks as the lock algorithm. XSAME(IJET,ITK) can be used for testing whether a track ITK belongs to a given jet (see 10.3). An example how to use XSAME in reconstructing decay chains is given in 7.4.5, example 2.

### 8.1.8 Flags, pointers, etc.

Pointers to other tracks and to vertices: see ch. 7.

**KTN (I)** JULIA / GALEPH track number

**KCLASS(I)** Track class:

- -1 (= KRECO) for reconstructed tracks
- -2 (= KMONTE) for MC truth
- = 0: track attributes = 0
- > 0: Lorentz frame. See 7.4.

**KTPCOD (I)** track's Particle Code

**CQTPN (I)** track's particle name (12 char.). = ' ' if particle code = 0

**KLUNDS (I)** LUND status code (MC particles only)

**XMC (I)** **.TRUE.** if MC particle

**KRDFL (I,IFLAG)** Integer value of user flag IFLAG (IFLAG=1–18). Flag is set to IVAL with CALL QSTFLI(I,IFLAG,IVAL); see 9.2.14.

**QRDFL (I,IFLAG)** Floating–point value of user flag IFLAG (IFLAG=1–18). Flag is set to VAL with CALL QSTFLR(I,IFLAG,VAL); see 9.2.14.

## 8.2 “Track” related detector data

These mnemonic symbols give access to information in BOS banks corresponding to an ALPHA “track”. These symbols return the integer or floating point value 0 if detector data are not available for a track. The names of these mnemonic symbols follow the same convention as the HAC parameters.



### 8.2.1 Global geometrical track fit: Bank FRFT

If the FRF0 card is present in the ALPHA cards file, the NR=0 version of the FRFT bank (track parameters determined without vertex detector coordinates) will be used. Otherwise, the NR=2 version of FRFT (TPC + ITC + VDET tracks) will be used.

<b>XFRF (I)</b>	<b>.TRUE.</b> if track fit data are available for track I
<b>QFRFIR (I)</b>	Inverse radius of curvature in x-y projection Signed positive if track bends counterclockwise, negative if track bends clockwise
<b>QFRFTL (I)</b>	Tangent of dip angle
<b>QFRFP0 (I)</b>	Phi at closest approach to the z axis
<b>QFRFD0 (I)</b>	Distance of closest approach to z axis
<b>QFRFZ0 (I)</b>	z coordinate of track point where QFRFD0 is measured Note: QDB and QZB (see 8.1.4) correspond to the closest approach to the beam axis.
<b>QFRFAL (I)</b>	Multiple scattering angle between TPC and ITC
<b>QFRFEM (I,N,M)</b>	Element N,M of the error covariance matrix N,M = 1,2,3,4,5,6 in the order IR,TL,PH,D0,Z0,AL. Note that the error matrix is valid at the innermost point used in the track fit, and therefore does not include multiple scattering in material before the tracking chambers.
<b>QFRFC2 (I)</b>	$\chi^2$ of helix fit
<b>KFRFDF (I)</b>	Number of degrees of freedom
<b>KFRFNO (I)</b>	Option flag for track fit

### 8.2.2 Number of coordinates used for the global fit: Bank FRTL

<b>KFRTNV (I)</b>	Number of coordinates in Vdet
<b>KFR TNI (I)</b>	Number of coordinates in ITC
<b>KFR TNE (I)</b>	Number of coordinates in ITC in following spirals
<b>KFR TNT (I)</b>	Number of coordinates in TPC
<b>KFR TNR (I)</b>	Number of coordinates in TPC in following spirals

### 8.2.3 Charged-particle identification: Bank FRID

<b>KFRIBP (I)</b>	Bit pattern for tracking devices
<b>KFRIDZ (I)</b>	Dead zone pattern for tracking devices
<b>KFRIBC (I)</b>	Bit pattern for calorimeters
<b>KFRIDC (I)</b>	Dead zone pattern for calorimeters

<b>QFRIPE (I)</b>	Electron probability
<b>QFRIPM (I)</b>	Muon probability
<b>QFRIPI (I)</b>	Pion probability
<b>QFRIPK (I)</b>	Kaon probability
<b>QFRIPP (I)</b>	Proton probability
<b>QFRINK (I)</b>	No Kink probability
<b>KFRIQF (I)</b>	Track Quality Flag from UFITQL
<b>XFRIQF (I)</b>	<b>.TRUE.</b> if KFRIQF(I) = 1 or 3

### 8.2.4 dE/dx data: Bank TEXS

Note: These functions return uncalibrated numbers. In general, dE/dx information should be accessed with subroutines QDEDX and QDEDXM (see 12.1).

<b>XTEX (I)</b>	<b>.TRUE.</b> if dE/dx is available for track I
<b>KNTEX (I)</b>	Number of TPC sectors on track I

In the following, N is the loop index of: *DO 10 N = 1, KNTEX(I)*

<b>KTEXSI (I,N)</b>	Sector slot number
<b>QTEXTM (I,N)</b>	Truncated Mean of dE/dx measurements
<b>QTEXTL (I,N)</b>	Useful Track Length for dE/dx
<b>KTEXNS (I,N)</b>	Number of Samples used for dE/dx
<b>QTEXAD (I,N)</b>	Average Drift length of samples

### 8.2.5 Electron identification: Bank EIDT

<b>XEID (I)</b>	<b>.TRUE.</b> if electron identification is available for track I
<b>KEIDIF (I)</b>	Quality flag
<b>QEIDRI (I,N)</b>	R(N) estimator, N = 1 ... 7. N = 1: Energy balance; N = 2: compactness; N = 3,4: long. profile; N = 5: dE/dx; N = 6: Dtheta barycenter; N = 7: Dphi barycenter.
<b>QEIDEC (I)</b>	Corrected energy with electron hypothesis
<b>KEIDIP (I)</b>	Particle hypothesis (= 1 if electron)
<b>QEIDEI (I,N)</b>	Energy in centered storeys stack N

### 8.2.6 Muon – HCAL association: Bank HMAD

XHMA (I)	.TRUE. if HCAL data are available for track I
KHMANF (I)	Number of Fired planes
KHMANE (I)	Number of Expected fired planes
KHMANL (I)	Number of Fired planes within Last ten planes
KHMAMH (I)	Mult Hits: number of clusters in last ten planes
KHMAIG (I)	IGeomflag: flag of possible dead zone
QHMAED (I)	Energy Deposit in corresponding HCAL storey
QHMACS (I)	$\chi^2$
KHMAND (I)	Number of Degrees of freedom
KHMAIE (I)	Expected bit map
KHMAIT (I)	True bit map
KHMAIF (I)	Preliminary identification flag

### 8.2.7 Muon chamber data: Bank MCAD

XMCA (I)	.TRUE. if muon chamber data are available for track I N = 1,2: Int/Ext chambers
KMCANH (I,N)	Number of associated hits
QMCADH (I,N)	Minimum distance hit–track
QMCADC (I,N)	Cutoff on hit–track distance
QMCAAM (I)	Min. angle between extrapolated and measured (in muon ch.) track
QMCAAC (I)	cutoff on minimum angle

### 8.2.8 QMUIDO Muon Identification: Bank MUID

XMUI (I)	.TRUE. if QMUIDO information is available for track I
KMUIIF (I)	Identification Flag
QMUISR (I)	Sum of HCAL residuals
QMUIDM (I)	Distance between track and closest muon chamber hit.
KMUIST (I)	FRFT track number of shadowing track

### 8.2.9 ECAL objects: Bank PECO

XPEC (I)	.TRUE. if ECAL data (PECO) are available for track I
QPECER (I)	Raw energy.
QPECE1 (I)	Fraction of energy in stack 1
QPECE2 (I)	Fraction of energy in stack 2
QPECTH (I)	Theta
QPECPH (I)	Phi
QPECEC (I)	Energy corrected for geometrical effects
KPECKD (I)	Region code – see ALEPH 88–134
KPECCC (I)	Correction code – see bank description
KPECRB (I)	Relation bits – see bank description
KPECPC (I)	PCOB number of associated cal. object

### 8.2.10 ECAL objects: Bank PEPT

XPEP (I)	.TRUE. if ECAL data (PEPT) are available for track I
QPEPT1 (I)	Theta in stacks 1 and 2
QPEPP1 (I)	Phi in stacks 1 and 2
QPEPT3 (I)	Theta in stack 3
QPEPP3 (I)	Phi in stack 3

### 8.2.11 HCAL objects: Bank PHCO

XPHC (I)	.TRUE. if HCAL data (PHCO) are available for track I
QPHCER (I)	Raw energy
QPHCTH (I)	Theta
QPHCPH (I)	Phi
QPHCEC (I)	Energy corrected for geometrical effects
KPHCKD (I)	Region code – see ALEPH 88–134
KPHCCC (I)	Correction code – see bank description
KPHCRB (I)	Relation bits – see bank description
KPHCPC (I)	PCOB number of associated cal. object

### 8.2.12 Reconstructed V0s: Bank YV0V

<b>XYV0 (I)</b>	<b>.TRUE.</b> if V0 data are available for track I
<b>KYV0K1 (I)</b>	FRFT track number of positive track from V0
<b>KYV0K2 (I)</b>	FRFT track number of negative track from V0
<b>QYV0VX (I)</b>	V0 x coordinate
<b>QYV0VY (I)</b>	V0 y coordinate
<b>QYV0VZ (I)</b>	V0 z coordinate
<b>QYV0X1 (I)</b>	First constraint on V0 mass (r in ALEPH 88-46)
<b>QYV0X2 (I)</b>	Second constraint on V0 mass (b in ALEPH 88-46)
<b>QYV0C2 (I)</b>	$\chi^2$ of V0 vertex fit
<b>KYV0IC (I)</b>	Fit hypothesis (see YV0V bank description)
<b>QYV0DM (I)</b>	Minimum distance between helices
<b>QYV0S1 (I)</b>	Psi angle for + track from V0
<b>QYV0S2 (I)</b>	Psi angle for - track from V0

### 8.2.13 Photons from GAMPEC: Bank EGPC

<b>XEGP (I)</b>	<b>.TRUE.</b> if GAMPEC data are available for track I
<b>QEGPR1 (I)</b>	Energy fraction in stack 1
<b>QEGPR2 (I)</b>	Energy fraction in stack 2
<b>QEGPF4 (I)</b>	Energy fraction in 4 central towers
<b>QEGPDM (I)</b>	Distance to the closest track (cm)
<b>KEGPST (I)</b>	$NST1+100*NST2+10000*NST3$ , $NSTi$ =number of storeys in stack i
<b>KEGPQU(I)</b>	QQuality flag <ul style="list-style-type: none"><li>• <math>CRCK + 10*DST1 + 100*DST2 + 1000*DST3</math></li><li>• <math>DSTi = 1</math> if dead storey(s) in stack i</li><li>• <math>CRCK = 1</math> if photon in crack region</li></ul>
<b>KEGPPE (I)</b>	Row number of corresponding PECO cluster

### 8.2.14 Energy Flow: Bank EFOL

<b>XEFO (I)</b>	<b>.TRUE.</b> if energy flow (EFOL) data are available for track I
<b>KEFOTY (I)</b>	Type of energy flow object (see Sec. 11.2)
<b>KEFOLE (I)</b>	PECO number of associated ECAL object
<b>KEFOLT (I)</b>	FRFT number of associated charged track
<b>KEFOLH (I)</b>	PHCO number of associated HCAL object
<b>KEFOLC (I)</b>	PCOB number of associated calorimeter object
<b>KEFOLJ (I)</b>	EJET number of associated jet

### 8.2.15 Neutral objects from PCPA: Bank PCQA

<b>XPCQ (I)</b>	<b>.TRUE.</b> if PCQA data are available for track I
<b>KPCQNA (I)</b>	Nature of neutral object (see Sec. 11.3)

## 8.3 Vertex attributes

The following attributes are all vertices. The argument (IVX) always refers to vertex IVX.

<b>QVX (IVX)</b>	x position
<b>QVY (IVX)</b>	y position
<b>QVZ (IVX)</b>	z position
<b>KVN (IVX)</b>	JULIA/GALEPH vertex number
<b>KVTYPE(IVX)</b>	vertex type (as in PYER) = 1 for primary vertex; = 2 for secondary vertex
<b>QVEM (IVX,N,M)</b>	element (N,M) of the covariance matrix N,M = 1,2,3 in the order QVX,QVY,QVZ

QVEM (IVX,1,1) is set to  $-1$  if the error matrix is not available.

See Section 7.8 for pointers between ALPHA tracks and vertices.

## Chapter 9

# Kinematics and Track Operations

In this chapter, the kinematics utility routines available in ALPHA are described. Also, many routines for creating new tracks and modifying existing tracks are described. First, calculations with scalar results are summarized. Next, routines with vector results are described (e.g., cross product). Finally, routines for doing kinematic fits, vertex fits, Lorentz transformations are discussed.

### 9.1 Scalar quantities

The arguments I,J,K,L are ALPHA “track” numbers.

<b>QCT (I)</b>	cos (polar angle)
<b>QPH (I)</b>	PHi = azimuth (radians)
<b>QPT (I)</b>	Transverse momentum (with respect to the beam line)
<b>QBETA (I)</b>	beta (see 8.1.1 for mass assumption)
<b>QGAMMA (I)</b>	gamma

**Note:** Returned masses are negative if  $(E^2 - p^2)$  is negative.

<b>QMSQ2 (I,J)</b>	(invariant mass) <sup>2</sup> of particles I and J
<b>QM2 (I,J)</b>	invariant mass of particles I and J
<b>QMSQ3 (I,J,K)</b>	(invariant mass) <sup>2</sup> of particles I, J, and K
<b>QM3 (I,J,K)</b>	invariant mass of particles I, J, and K
<b>QMSQ4 (I,J,K,L)</b>	(invariant mass) <sup>2</sup> of particles I, J, K, and L
<b>QM4 (I,J,K,L)</b>	invariant mass of particles I, J, K, and L
<b>QDMSQ (I,J)</b>	mass <sup>2</sup> of the 4-momentum difference $p(I) - p(J)$ . In a decay $I \rightarrow J + x$ , QDMSQ(I,J) gives the mass <sup>2</sup> of x.
<b>QPPAR (I,J)</b>	momentum component of particle I parallel to particle J
<b>QPPER (I,J)</b>	momentum component of particle I perpendicular to particle J

- 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)
- QDECA2(I,J)** cos (decay angle): In a two-body decay  $x \rightarrow I + J$ , the decay angle is the angle between particle x and particle I, measured in the rest frame of particle x (i.e., the angle between the boost direction and particle I).
- QDECAN(I,J)** extension of QDECA2 for the n-body decay  $I \rightarrow J + \text{any}$ . Note the different meaning of the first argument in QDECA2 and QDECAN.
- QMDIFF(I,'part')** mass difference between I and particle table mass of 'part'.
- QMCHI2(I,'part')**  $\chi^2$  resulting from mass difference between I and particle table mass of 'part'. This function is equivalent to
- $$(QM(ITK) - QPMASS('part - name'))^2 / QSIGMM(ITK).$$
- QMCHIF(I)**  $\chi^2$  of mass-constrained fit (KVFITM or KVFITA - see 9.3). QMCHIF(I)=-1 if track I was not the result of a fit.
- QVDIF2(IV1,IV2)** distance between vertices IV1 and IV2 in  $r - \phi$  (see 9.4).
- QVDIF3(IV1,IV2)** distance between vertices IV1 and IV2 in 3 dimensions (see 9.4).
- QVCHIF(I)**  $\chi^2$  of vertex fit (KVFITN or KVFITV - see 9.4).

## 9.2 Vector quantities

### 9.2.1 General Remarks

Except where noted below (e.g., mass), the attributes of "tracks" read from the input tape cannot be changed by the user. To modify attributes of an "input" track, a copy of the track must be made.

The following example illustrates some features of the routines described in this section.

```

ISUM = KVNEW (DUMMY)
DO 10 ITK1 = ... , ...
DO 10 ITK2 = ... , ...
CALL QVADD2 (ISUM, ITK1, ITK2)
C    ... analysis of the sum of ITK1 and ITK2, for example:
CALL HF2 (4711, QP(ISUM), QM(ISUM),1.)
10 CONTINUE

```

The function KVNEW (DUMMY) creates a new track (ISUM) in the system area which is needed as working space for most of the subroutines described here (see 9.2.8). New tracks can be created whenever necessary, but to avoid exceeding the size of the BOS array, *they should not*



be created inside loops. A warning is issued if an “input” track is used as working space (*i.e.* if an “input” track is given as the output track of a routine).

Subroutine QVADD2 (ISUM, ITK1, ITK2) adds the 4–momenta of tracks ITK1 and ITK2 and stores the resulting composite particle as track ISUM (see 9.2.2). All track–track and track–vertex relations, flags, etc. are set in QVADD2. For example, all flags for the lock algorithm are set (see 10.2.3). Thus, with CALL QLOCK (ISUM), you lock ITK1 and ITK2 as well as ISUM. The mother–daughter relation (see 7.5.1) from ISUM to ITK1 and ITK2 is stored, but NOT the reverse daughter–mother relation; see KVSAVE in 9.2.9).

In subroutine calls, the result is stored in the track denoted by the first subroutine argument: for example, CALL QVCOPY (ITO, IFROM) copies track IFROM to track ITO.

Do not mix up tracks from different classes. ITK1 and ITK2 in QVADD2 must belong the same class (KRECO or KMONTE or a Lorentz frame derived from one of them; see 7.4.1). If you really want to mix up tracks from different classes, they must first be “saved” in the same class (see KVSVC in 9.2.12).

## 9.2.2 Add 4–momenta of particles

### Add two particles

```
CALL QVADD2 (ISUM, ITK1, ITK2)
```

Add the 4–momenta of ITK1 and ITK2 and store the result in ISUM.

### Add three particles

```
CALL QVADD3 (ISUM, ITK1, ITK2, ITK3)
```

Add the 4–momenta of ITK1, ITK2, ITK3 and store the result in ISUM.

### Add four particles

```
CALL QVADD4 (ISUM, ITK1, ITK2, ITK3, ITK4)
```

Add the 4–momenta of the particles ITKn ( $n = 1$  to 2,3, or 4) and store the result in ISUM.

### Add N particles

```
CALL QVADDN (ISUM, ITK)
```

For adding more than four particles, either use QJADDP (see 10.3) or construct a loop with QVADDN:

```

      ISUM = KVNEW (DUMMY)
      DO 10 ITK = ... , ...
10    CALL QVADDN (ISUM, ITK)

```

The sum of all track momenta is stored in ISUM.

Before using track ISUM in such loops, its momentum must be set to zero. This is done in KVNEW. When reusing ISUM for another loop, however, it must be zeroed by CALL QVZERO (ISUM).

### 9.2.3 Recalculate 4-Vector of V0

```
CALL QIDV0 (ITK, 'PI+', 'PI-')
```

Recalculates the 4-vector of a "V0" object ITK (i.e., a reconstructed neutral track pointing to a V0) by using the 3-vectors of the decay particles and masses denoted by the two particle names given as function arguments. The attributes of ITK are overwritten by the new 4-vector. The attributes of the decay particles remain unchanged. For saving a V0 mass hypothesis, the function KVSAVE (9.2.9) or KVFITM (9.3) must be called. For example,

```

      DO 10 ITK=KVVOT,KLVOT
          CALL QIDV0 (ITK, 'P', 'PI-')
          IF (QMCHI2 (ITK, 'LAMO') .LE. 9.)
&      ISAVE = KVFITM (ITK, 'LAMO', IER)
          CALL QIDV0 (ITK, 'P#', 'PI+')
          IF (QMCHI2 (ITK, 'LAMO') .LE. 9.)
&      ISAVE = KVFITM (ITK, 'LAM#0', IER)
10    CONTINUE

```

### 9.2.4 Copy a track

```
CALL QVCOPY (ITO, IFROM)
```

Copy the track attributes from IFROM to ITO. If one of the tracks is in the user's track section, only the basic attributes (see 8.1.1) are copied. Otherwise, all flags, relations, etc. are copied. See remarks about lock algorithm in sections 10.2.1 and 10.2.3.

QVCOPY should be used only if a specific track ITO has to be overwritten. Another copy routine which is protected against overwriting tracks is KVSAVE (9.2.9).

### 9.2.5 Cross product

```
CALL QVCROS (ICROSS, ITK1, ITK2)
```

Store the cross product of the vectors ITK1 and ITK2 in ICROSS. Space for ICROSS can be reserved by ICROSS = KVNEW (DUMMY).

Mother – daughter relation: ITK1 and ITK2 are daughters of ICROSS.

### 9.2.6 Drop tracks

```
CALL QVDROP ('part-name', ICLASS)
```

Drop all tracks with name 'part-name' in the class ICLASS. For example,

```
CALL QVDROP ( ' ', ICLASS)
```

will drop tracks with any track in class ICLASS. The main application of this subroutine is to drop all tracks in a specific Lorentz frame. See the example in section 9.5.3.

If ICLASS = KRECO or ICLASS = KMONTE: Only tracks created in the analysis program are dropped; tracks coming from the event input file cannot be dropped. No garbage collection takes place.

### 9.2.7 Copy track attributes into a Fortran array

(To copy a Fortran array into a track, see section 9.2.13.)

#### Copy 3–momentum of a track

```
CALL QVGET3 (ARR, ITK)
```

Copy the 3–momentum (px,py,pz) of track ITK into the Fortran array ARR with DIMENSION ARR(3).

#### Copy 4–momentum of a track

```
CALL QVGET4 (ARR, ITK)
```

Copy the 4–momentum (px,py,pz,E) of track ITK into the Fortran array ARR with DIMENSION ARR(4).

## Copy covariance matrix of a track

```
CALL QVGETS (ERRMAT, ITK)
```

Copy the 4\*4 covariance matrix (order: px,py,pz,E) of track ITK into the symmetric Fortran matrix ERRMAT with DIMENSION ERRMAT(4,4).

### 9.2.8 Create a new track

```
INEW = KVNEW (DUMMY)
```

Create a new track (see 9.2.1) with momentum = energy = 0. The corresponding space is allocated dynamically and NOT kept when a new event is read in. INEW is a track without a particle name. None of the access methods described in Ch. 7 give access to it; the only access to the track is with the track number INEW. Consequently, it can never be dropped (see 9.2.6). The new track does NOT belong to a specific class (KRECO / KMONTE / Lorentz frame).

### 9.2.9 Save a track

```
ISAVE = KVSAVE (ITK, 'part-name')
```

To save track ITK means to copy it into a new track ISAVE and to assign a particle name to the track copy. This particle name can be used later for direct access to this particle (see 7.4). Note that the mass is NOT changed in KVSAVE (see KIDSAV, 9.2.11).

The class (KRECO / KMONTE / Lorentz frame; see 7.4.1) of a saved track is given by its history (in the example below, the class of JPSI is set equal to that of ITK1 and ITK). A dedicated routine KVSAVC (see 9.2.12) makes it possible to copy a track into a different or new class. KVSAVC must be used instead of KVSAVE if the track class cannot be deduced from the track history (see example in 9.2.12).

If 'part-name' is equal to ' ', KVSAVE only performs a copy, and the track copy has no particle name. In contrast to QVCOPY (see 9.2.4), KVSAVE never overwrites a track.

In a decay chain, the daughter-mother relation is established by KVSAVE. The inverse relation (mother-daughter) is established in routines like QVADDx.

**Example:**  $\psi \rightarrow e^+e^-$ :

```
ISUM = KVNEW (DUMMY)
ITK1 = KPDIR ('E+', KRECO)
10 IF (ITK1 .NE. 0) THEN
    ITK2 = KPDIR ('E-', KRECO)
20 IF (ITK2 .NE. 0) THEN
C ... all e+ e- combinations:
```

```

        CALL QVADD2 (ISUM, ITK1, ITK2)
C ... cut on invariant mass and save J/psi candidates:
        IF (ABS (QM(ISUM) - QPMASS ('JPSI')) .LT. (your cut))
&         ITKPSI = KVSAVE (ISUM, 'JPSI')
        ITK2 = KFOLLO (ITK2)
        GO TO 20
    ENDIF
    ITK1 = KFOLLO (ITK1)
    GO TO 10
ENDIF

```

The daughter–mother relation is established only for the accepted (i.e., saved)  $\psi$ s. In subsequent loops, the  $\psi$ (s) is (are) directly accessible by their name and can be used, for example, to analyze  $\psi' \rightarrow \psi\pi^+\pi^-$  in the same way as  $\psi \rightarrow e^+e^-$ .

### 9.2.10 Save a track inside particle/antiparticle loop

```

ISAVE = KVSAVA (ITK, 'part-name', IANTI)

```

This routine has the same function as KVSAVE, but is intended to be used inside of loops over particles and antiparticles. If IANTI is 0, the track is saved as 'part-name'; if IANTI is nonzero, the track is saved as the corresponding antiparticle.

### 9.2.11 Save a track and set its mass

```

ISAVE = KIDSAV (ITK, 'part-name')

```

This function does the same thing as KVSAVE, but also sets the mass of track ISAVE to the mass of 'part-name'. As in KVSAVE, the original track ITK is not changed. For charged tracks, KIDSAV will save tracks as the appropriate particle or antiparticle depending on their charge. For example KIDSAV(ITK, 'K+') will save positive tracks as  $K^+$  and negative tracks as  $K^-$ .

### 9.2.12 Save a track with class ICLASS

```

ISAVE = KVSAVC (ITK, 'part-name', ICLASS)

```

Save (see 9.2.9) track ITK in track class ICLASS independent of the track history. Track classes are described in 7.4.1. If class ICLASS does not yet exist, a new class is created. Note that the maximum number of new classes is six (see 9.5).

It is possible but not recommended to put a reconstructed track into the class KMONTE (MC truth) or vice versa. The lock algorithm will not work for these tracks.

**Example:** Create and save a beam particle in track class KRECO.

```
DIMENSION VEC(4)
VEC(1) = 0.                                px
VEC(2) = 0.                                py
VEC(3) = QELEP * 0.5                       beam energy
VEC(4) = VEC(3)                           energy = momentum
INEW = KVNEW (DUMMY)
CALL QVSET4 (INEW, VEC)
IBEAM = KVSAVC (INEW, 'BEAME+', KRECO)
```

KVSAVC has to be used here instead of KVSAVE because the track history of INEW does not specify the track class. See 9.2.8 and 9.2.13 for explanations of KVNEW and QVSET4.

### 9.2.13 Modify track parameters

(To copy a track into a Fortran array, see 9.2.7.)

The QVSxxx routines described below modify the specified track attributes but do not change any flag or pointer. Thus, all track–track relations (KMOTH, KDAU, KSAME, etc.) which have been established remain valid even if the routines completely overwrite the kinematic quantities.

#### Scale track momentum

```
CALL QVSCAL (ITK, FACTOR)
```

Multiply the momentum of track ITK by the factor FACTOR. The energy of ITK is set according to the new momentum and the old mass value. QVSCAL can be called for “input” tracks.

#### Set mass of a track

```
CALL QVSETM (ITK, AMASS)
```

Set the mass of track ITK to AMASS. The new energy of ITK is set according to the new mass and the old (unchanged) momentum. QVSETM can be called for “input” tracks.

#### Set 3–momentum of a track

```
CALL QVSET3 (ITK, ARR)
```

Copy the Fortran array ARR containing px, py, pz with DIMENSION ARR(3) into the momentum vector of track ITK. The new track energy is calculated from the new momentum and the old mass.

### Set 4–momentum of a track

```
CALL QVSET4 (ITK, ARR)
```

Copy the Fortran array ARR containing px, py, pz, E with DIMENSION ARR(4) into the momentum vector of track ITK. All basic track attributes are recalculated. See example in section 9.2.12.

### Set covariance matrix of a track

```
CALL QVSETS (ITK, ERRMAT)
```

Copy the 4\*4 Fortran matrix ERRMAT containing the track's covariance matrix in the order px,py,pz,E with DIMENSION ERRMAT(4,4) into the covariance matrix of track ITK.

### 9.2.14 Set User Track Flags

```
CALL QSTFLR (ITK,IFLAG,VAL) and CALL QSTFLI(ITK,IFLAG,IVAL)
```

<b>ITK</b>	ALPHA “track” number
<b>IFLAG</b>	Flag number: 1 – 18
<b>VAL, IVAL</b>	Value to be stored in flag IFLAG

Each ALPHA “track” has 18 user flags which may be set to any integer or real value. QSTFLR and QSTFLI are used to set a flag to a real number or to an integer, respectively. Once these flags are set, they can be read with the functions KRDFL(ITK,IFLAG) (integer) and QRDFL(ITK,IFLAG) (real); see section 8.1.8.

### 9.2.15 Subtract track momenta

```
CALL QVSUB (IDIFF, ISUM, ISUB)
```

Subtract the vector ISUB from ISUM and store the result in IDIFF. Space for IDIFF can be reserved by IDIFF = KVNEW (DUMMY).

- If  $QE(ISUM) < QE(ISUB)$ , the result is meaningless.
- If  $QP(IDIFF) > QE(IDIFF)$ , the result gets a negative mass.
- A warning is issued in either case.

### 9.2.16 Zero track attributes

`CALL QVZERO (ITK)`

Set all attributes (momentum, etc.) of `ITK` to 0. Note that `KVNEW` (see 9.2.8) implies `QVZERO`.

## 9.3 Kinematic fitting

`IFIT = KVFITM (ITK, 'part-name', IER)`

Performs a mass-constrained fit for the decaying particle `ITK`. This fit readjusts the 4-vector of `ITK` by using the constraint  $E^2 - p^2 = \text{mass}(\text{'part-name'})^2$  (method: Lagrange multiplier). In particular, the fit improves the 3-momentum resolution. `KVFITM` determines the 4-momentum of the decaying particle only; the 4-vectors of the decay products are not recalculated and remain unchanged. Therefore, the momenta of the daughter particles will not add up to the fitted momentum of the mother exactly.

`IFIT = KVFITA (ITK, 'part-name', IER, IANTI)`

This function is similar to `KVFITM`. It is intended to be used inside loops over particles and antiparticles. The particle given by `'part-name'` is used if `IANTI` is 0; the corresponding antiparticle is used if `IANTI` is nonzero.

## 9.4 Vertex fitting with YTOP

The following functions provide an interface to the `YTOPOL` package in `ALEPHLIB`.

`IFIT = KVFITN (ND, ID, 'part-name')`

Fit `ND` tracks stored in `ID` to a common vertex. `IFIT` is the number of new track coming into the vertex; this track is stored with the name `'part-name'` and can be accessed with `KPDIR`, etc. The vertex number is the end vertex of track `IFIT`:

$$IVX = KENDV(IFIT).$$

`IFIT = -1` if the fit fails.

`IFIT = KVFITV (IV,ND, ID, 'part-name')`

Same as `KVFITN` except that vertex `IV` is used as an additional constraint in the fit.

Both functions refit the track parameters of the input tracks and calculate the 4-vector and error matrix of the new track (`IFIT`) at the fit vertex. The fit vertex position and error matrix are stored in the end vertex of `IFIT`: `KENDV(IFIT)`. There can be any number of input tracks, but if



$NTR > 10$ , KVFITx will first vertex tracks 1-10 and then add the following tracks to this vertex. Input tracks can be either charged tracks, V0s, or tracks resulting from a previous fit.

The  $\chi^2/NDF$  for the vertex fit may be accessed with the statement function QVCHIF(IVX), where IVX is the end vertex of IFIT. The number of degrees of freedom for the routines are:

$NDF = 2 * ND - 3$  for KVFITN

$NDF = 2 * ND$  for KVFITV

The following statement functions give the distance between two vertices IV1 and IV2.

$DIST = QVDIF2(IV1,IV2)$  distance in  $r - \phi$

$DIST = QVDIF3(IV1,IV2)$  distance in 3 dimensions

**Example:**

Assume that you have a  $D^0 \rightarrow K\pi$  candidate (IDO) and a lepton (ILEP) from a  $B$ -meson decay. The following code finds the vertex of the  $B$  decay.

```

CALL QVSETM(IPION,QPMASS('PI+'))      !pion mass
CALL QVSETM(IKAON,QPMASS('K+'))      !kaon mass
ITL(1) = IPION
ITL(2) = IKAON
IDO    = KVFITN(2,ITL,'DOKp')
IF(IDO.GT.0) THEN
  RMDO = QM(IDO)                      ! vertex refitted D0 mass
  IVDO = KENDV(IDO)                   ! D0 vertex
  CHI2 = QVCHIF(IVDO)                 ! chi**2 of the D0 vertex
C
  ITL(1) = ILEP
  ITL(2) = IDO
  IB     = KVFITN(2,ITL,'Blep')       ! fit B vertex
  IF(IB.GT.0) THEN
    IVB = KENDV(IB)                   ! B vertex
    CHI2 = QVCHIF(IVDO)               ! chi**2 of B vertex
    DIST = QVDIF3(IVB,IVDO)          ! distance between B and D0
                                     ! vertex
  ENDIF
ENDIF
ENDIF

```

## 9.5 Lorentz transformations

See also QDECAx (decay angle in the rest frame of a decaying particle) in 9.1.

### 9.5.1 Boost a track and its daughters

`IBOOST = KTLOR (ITK, IREST)`

Boost the track ITK into the rest frame of IREST and store the result in IBOOST.

The sample of all tracks boosted into the rest frame of any track IREST constitutes its own track class which is denoted by the track number IREST, and which can be accessed directly as described in 7.4. Another way to access boosted tracks is to use KSAME (see 7.6), which makes it possible to jump from a given track to the same track in other Lorentz frames.

A track can be boosted into its own rest frame. The result is a vector with the initial direction and a momentum very close to 0.

KTLOR does not boost a track into a given frame twice. It returns, instead, the number of the already boosted track. This rule is only valid as long as you leave the mass and the particle name unchanged.

If a composite track is to be boosted, all daughters, granddaughters, etc. (but NOT mothers, etc!) of the track are boosted at the same time. The mother–daughter and daughter–mother relationships among the boosted tracks are established. If these relationships are not needed, use KTLOR1 or QTCLAS described below.

The track to be boosted (ITK) and the track which defines the rest frame (IREST) may belong to different track classes. No check is done that the boost makes sense. Note, however, an important restriction: If more than one track is boosted into a frame, all of them must come from the same class. This restriction prevents putting reconstructed tracks and MC truth into the same track class; see example in 9.5.3.

A maximum of six Lorentz frames can be used simultaneously. Frames which are not used any more can be dropped by *CALL QVDROP (' ', IREST)* (see 9.2.6) to reduce the number of frames in use, and to release the space occupied by the boosted tracks.

### 9.5.2 Boost a track

`IBOOST = KTLOR1 (ITK, IREST)`

Same function as KTLOR except that daughters are NOT boosted. A track boosted by KTLOR1 has no daughters or mothers, even if these relatives exist in the original frame.

### 9.5.3 Boost all tracks of a given class

```
CALL QTCLAS (ICLASS, IREST)
```

Boost the tracks in class ICLASS (= KRECO or KMONTE or a Lorentz frame previously defined) into the rest frame of track IREST. The track selection follows exactly the same rules as described for the event topology routines in Chapter 10. In particular, selection options can be set by the routines QJOPTR or QJOPTM (see 10.1), and locked tracks are not boosted. As in KTLOR1, daughters are NOT boosted and mother–daughter relations are NOT available.

#### Example:

```
IREST = ... this momentum vector defines the rest system.
C boost the reconstructed tracks:
  CALL QTCLAS (KRECO, IREST)
C if you want to boost MC particles into the same frame, first make a
C copy of IREST - do not mix up KRECO and KMONTE in the same class:
  ICOPY = KVSAVE (IREST, 'COPY')
  CALL QTCLAS (KMONTE, ICOPY)
C ...
C later reference to the boosted particles (see >)
  ITK = KPDIR ('CHARGED', IREST)
C use a loop with KFOLLO. The same for MC particles:
  ITK = KPDIR ('E+', ICOPY)
C ...
C drop all boosted tracks in frame IREST:
  CALL QVDROP (' ', IREST)
```

# Chapter 10

## Event Topology Routines

All of the subroutines described in this chapter perform loops over tracks or particles. The arguments and loop algorithms are similar for all of these subroutines, and are described in detail in Section 10.3. The “tracks” to be considered are selected with the routines QJOPTR (for reconstructed tracks) and QJOPTM (for Monte Carlo tracks); these routines also specify tracks to be used by the Lorentz transformation routine QTCLAS (see 9.5.3). In addition, the LOCK routines described in Section 10.2. can be used to exclude tracks from analysis by the QJxxxx routines described in this chapter.

### 10.1 Options for “QJxxxx” routines

#### 10.1.1 Set option for reconstructed objects

```
CALL QJOPTR ('reco-option', 'additional')
```

##### Input arguments:

**'reco-option'** One of the following options:

- **'RE'**: “REconstructed” tracks (**default**; see 7.1)
- **'CO'**: Calorimeter Objects
- **'CH'**: CHarged tracks
- **'EF'**: ENFLW or mask energy flow objects depending on ELFW option; see Ch. 11.
- **'EJ'**: YCUT=0.003 jets based on objects in EF section; see 11.2.
- **'PC'**: PCPA-based energy flow using PCPA neutral objects and selected charged tracks; see 11.3.
- **'AL'**: All objects (charged tracks, cal. objects, ECAL objects, HCAL objects, V0s, V0 daughter tracks, etc.). If not applied skillfully together with LOCK, many objects will be counted twice.
- **'NO'**: NO object. Only objects specified by 'additional' (see below) will be taken into account.

**'additional'** Particle name of one or several additional particle(s) to be analyzed. If no additional particles are to be considered, the argument ' ' must be given (*e.g.*, CALL QJOPTR('CO',' ')).

The following example would cause the QJ routines to consider charged tracks and all particles called MISS-VECTOR; MISS-VECTOR might be a pseudo-particle created by one of the routines described later in this chapter.

```
CALL QJOPTR ('CH', 'MISS-VECTOR').
```

Specifying additional reconstructed particles (QJOPTR) has no impact on MC particles (QJOPTM) and vice versa.

### 10.1.2 Set option for MC particles

```
CALL QJOPTM ('MC-option', 'additional')
```

'MC-option' One of the following options:

- 'VI': Only particles with a stability codes  $> 0$ . VI stands for 'best chance to be visible'. (**default**: see 8.1.5)
- 'EP': Only particles with stability codes 1, 2, or  $-3$ . EP stands for energy-momentum conservation.
- 'AL': All objects. If not applied carefully together with LOCK, many objects will be counted twice.
- 'NO': No object. Only objects specified by 'additional' will be taken into account.

'additional' Same as for QJOPTR.

## 10.2 Lock tracks / subsamples of tracks

The "LOCK" routines described here make it possible to exclude tracks from analysis by the routines (QJxxxx) described in this chapter. This feature can be used both to flag background tracks and to restrict the analysis to a subsample of all tracks (*e.g.*, to consider only tracks which contribute to a given jet). In any user routine, you may test the lock status of a given track ITK with **XLOCK(ITK)** which is **.TRUE.** if the track has been locked.

Every track has three independent locks: one simple one (QLTRK) and two more complicated ones (QLOCK and QLOCK2) with a broader scope of applications. If desired, several locks can be used simultaneously. A track is considered "unlocked" if and only if all three locks are open.

Opening and closing locks is done only in user routines; no track is locked unless it is explicitly locked by the user.

### 10.2.1 Lock a single “track”

`CALL QLTRK (ITK)`

**ITK**                    ALPHA “track” number

**Remarks:**            In contrast to the other locks described below, QLTRK locks the object ITK and its direct copies only (including the same object with a different vertex assignment) -- no other associated objects are affected.

### 10.2.2 Unlock a single “track”

`CALL QLUTRK (ITK)`

**ITK**                    ALPHA “track” number

**Remark:**              QLUTRK opens only the lock set by QLTRK. If another lock is still closed, the track remains locked.

### 10.2.3 Lock a track “family”

`CALL QLOCK (ITK)`

**ITK**                    ALPHA “track” number

The family of track ITK consists of:

- The track ITK itself.
- All copies of track ITK which have been made or will be made, including Lorentz boosts of ITK.
- For charged tracks, all associated cal. objects; for cal. objects, all associated charged tracks.
- For reconstructed tracks, all tracks based on the same reconstructed object but assigned to different vertices, used with different mass hypotheses, etc..
- Daughters, granddaughters, great-granddaughters, ... ; i.e., all kinship in descending line.
- Mothers, grandmothers, great-grandmothers, ... ; i.e., all kinship in directly (!) ascending line. If you use QLOCK for declaring a reconstructed particle to be background, all its ancestors (composite particles based on it) are implicitly declared to be background.
- Jets and other “pseudo particles” described in 10.3. If you lock a jet, you lock all contributing particles. If you lock a particle, you lock all jet vectors to which the particle belongs. To lock all particles not belonging to a jet, user QLREV described below.

Reconstructed tracks and MC truth are treated separately; locking a reconstructed track has no effect on any MC track and vice versa. Lock does not work if you mix up reconstructed tracks and MC.

## 10.2.4 Unlock tracks (locked with QLOCK)

```
CALL QLZER (IREMC)
```

**IREMC** = KRECO for reconstructed tracks and KMONTE for MC truth

Note that the lock algorithm works for all Lorentz frames simultaneously, and that the specification of a particular frame is NOT allowed (in contrast to 7.4.1). Reconstructed objects and Monte Carlo objects are treated separately. QLZER opens the lock QLOCK for all tracks. Tracks may remain locked if other locks are still closed. It is not possible to remove the lock QLOCK for a single track. Using two locks simultaneously (see 10.2.6) should provide all the facilities that are needed.

## 10.2.5 Reverse the lock state (corresponding to QLOCK)

```
CALL QLREV (IREMC)
```

**IREMC** (see 10.2.4).

- All unlocked tracks will be locked.
- All locked tracks will be unlocked provided that there is no other closed lock and, for composite particles, that there is no locked daughter, granddaughter, ... after the QLREV operation.

Calling QLREV a second time reestablishes the initial lock state. The mnemonic symbol XLREV(IREMC) is set to .TRUE. if the lock state is reversed. At the begin of the event processing and after calling QLZER(IREMC), XLREV(IREMC) is .FALSE..

## 10.2.6 Second Lock

```
CALL QLOCK2(ITK)
```

QLOCK2 works in the same way as QLOCK. If one of these locks is used to flag background tracks, the other one can be used to select subsamples of the non-background tracks. Also available: CALL QLZER2 (IREMC), CALL QLREV2 (IREMC), and the logical function XLREV2(IREMC).

## 10.3 Add momenta of all particles of a given class

```
CALL QJADDP (SCALAR, 'vector-name', ICLASS)
```

For adding momenta of a few particles, see 9.2.2. (NOTE: All of the QJxxxx routines have similar arguments. The arguments are explained fully in this explanation of QJADDP.)

### 10.3.1 Input argument

**ICLASS** Class = KRECO or KMONTE or a Lorentz frame identifier (see 7.4.1). If ICLASS is KRECO, note that initially all charged particles have the pion mass and all neutral objects have mass = 0. This can be modified by CALL QVSETM (see 9.2.13). If ICLASS refers to a Lorentz frame, particles not boosted into the frame are ignored without notification. The routine QTCLAS (see 9.5.3) performs a Lorentz transformation of all tracks belonging to a class. If a particle has been boosted several times into the same frame, the most recently boosted hypothesis will be used (see remarks in 7.4.5).

### 10.3.2 Results

A scalar result is stored in the first subroutine argument. In QJADDP, the scalar result is the 3-momentum sum of all particles. An output vector is specified by its name, which is the second subroutine argument 'vector-name'. If you are interested in the scalar result only and not in the output vector, specify a blank space ' '. QJADDP has exactly one output vector: the sum of all 4-momenta. The following example shows how to use this vector.

```
CALL QJADDP (PSUM, 'ADD-ALL', KRECO, ...)  
ISUM = KPDIR ('ADD-ALL', KRECO)  
CALL HF2 (4711, QP(ISUM), QM(ISUM), 1.)
```

Other routines may output several vectors; a loop using KFOLLO (see 7.4.2) must be constructed to access all of them.

Locking an output vector locks all particles contributing to it (see 10.2.3). You can test whether a track ITK contributes to an output vector ISUM by using the logical symbol XSAME (ITK, ISUM) (Sec. 8.1.7).

The output vectors of "QJ" routines are called "pseudo-particles". In some routines described below, these pseudo-particles represent an axis rather than a 3- or 4-vector; the momentum value may or may not be meaningful. For consistency, an energy assuming mass = 0 is calculated in these cases.

In addition, pseudo-particles are treated differently than "real" particles:

- A warning is issued if the same name is used for a pseudo-particle and a "real" particle.
- Existing pseudo-particles are dropped automatically if the same name and the same class is used in another call to a "QJ" routine. Thus, in

```
CALL QJADDP (PSUM, 'ADD-ALL', KRECO, ...)  
CALL QJADDP (PSUM, 'ADD-ALL', KRECO, ...)  
CALL QJADDP (PSUM, 'ADD-ALL', KMONTE, ...)
```

the output vector of the first call is not available after the second call. Thus, output vectors from different calls are never mixed up. Since the third call refers to a different class, the vector from the second call is not dropped. Note that you are free to invent new names in every new call to a "QJ" or any other routine.



## 10.4 Momentum tensor eigenvalues and eigenvectors

`CALL QJEIG (EIGVAL, 'eigenvector', ICLASS)`

See also QJSPHE in 10.6 for sphericity value and axis.

**Input argument:**

**ICLASS** described in 10.3.

**Results:**

**EIGVAL** eigenvalues in descending order with DIMENSION EIGVAL(3).

- Sphericity = 1.5 \* (1. - EIGVAL(1))
- Aplanarity = 1.5 \* EIGVAL(3)
- Planarity = EIGVAL(3) / EIGVAL(2)

**'eigenvector'** Three eigenvectors:

- IMAJOR = KPDIR ('eigenvector', ICLASS)
- ISEMI = KFOLLO (IMAJOR)
- IMINOR = KFOLLO (ISEMI)

## 10.5 Linearized momentum tensor eigenvalues and eigenvectors

`CALL QJTENS (EIGVAL, 'eigenvector', ICLASS)`

Same as QJEIG except that a different normalization is used. The momentum tensor for this calculation is defined as

$$M_{jk} = \frac{1}{P} \sum_i \frac{p_{ji} p_{ki}}{p_i} \quad (10.1)$$

$$j, k = 1, 2, 3 \quad (10.2)$$

**Input arguments and results are as described for QJEIG.**

## 10.6 Sphericity

`CALL QJSPHE (SPHERI, 'spheri-axis', ICLASS)`

Calculates sphericity value and sphericity axis. See also QJEIG in 10.4 for eigenvalues and eigenvectors of the momentum tensor.

**Input argument:**

**ICLASS**            described in 10.3.

**Results:**

**SPHERI**            Sphericity value  
'spheri-axis'      Sphericity axis.

**Error conditions:**

**Zero or one track** SPHERI value 0.; output vector = 0.,0.,0.,0.

**Two tracks**        SPHERI = 0.; output vector = track vector with largest p.

## 10.7 Thrust

```
CALL QJTHRU (THRUST, 'thrust-axis', ICLASS)
```

**Input argument:**

**ICLASS**            described in 10.3.

**Results:**

**THRUST**            Thrust value.  
'thrust-axis'      Thrust axis.

**Error conditions:**

**No track**            THRUST value 0.; output vector = 0.,0.,0.,0.

**One track**          thrust value = 1; output vector = track vector.

## 10.8 Fox-Wolfram Moments

```
CALL QJFOXW(FOXWOL, ICLASS)
```

**Input argument:**

**ICLASS**            described in 10.3.

**Result:**

**FOXWOL**            Fox-Wolfram moments H0 – H4; DIMENSION FOXWOL(5).

## 10.9 Divide event into two hemispheres

```
CALL QJHEMI ('same-s', 'opp-s', ICLASS, IVEC, COSCUT)
```

### Input arguments:

**ICLASS**           described in 10.3.  
**IVEC**             Track number of vector which defines the “hemi”spheres.  
**COSCUT**           The cosine of the opening angle of a cone around IVEC. Tracks inside this cone belong to the same side, and all other ones belong to the opposite side. The word “hemisphere” is correct if  $\text{COSCUT} = 0$ .

### Results:

**'same-s'**           The 4-momentum sum of tracks on the same side as IVEC.  
**'opp-s'**            The 4-momentum sum of tracks on the side opposite to IVEC.

The two output vectors can be used to assign tracks to one of the the two hemispheres with the lock algorithm (10.2.3).

In the following example, the event is divided into two hemispheres according to the thrust axis. Then, each hemisphere is boosted separately into the rest frame of all contributing tracks.

```
DIMENSION IVECT(2)
C---Thrust axis
  CALL QJTHRU (THRU, 'THRUST', KRECO)
  ITHRU = KPDIR ('THRUST', KRECO)
C---Two hemispheres:
  CALL QJHEMI ('SAME', 'OPPO', KRECO, ITHRU, 0.)
  IVECT(1) = KPDIR ('SAME', KRECO)
  IVECT(2) = KPDIR ('OPPO', KRECO)
C---Lock all tracks in the 'oppo' hemisphere:
  CALL QLOCK (IVECT(2))
C---Loop over both hemispheres:
  DO 10 IHEMI = 1, 2
C---Transform all selected tracks into the rest frame of IVECT(IHEMI):
  CALL QTCLAS (KRECO, IVECT(IHEMI))
C---Now, do the analysis. For example:
C---Plot the thrust in the boosted frame.
  CALL QJTHRU (THRUB, ' ', IVECT(IHEMI))
  CALL QHF1 (4711, THRUB, 1.)
C---QLREV: locked tracks -> unlocked tracks and vice versa.
C---This selects tracks in the hemisphere 'OPPO' for next loop.
  CALL QLREV (KRECO)
10 CONTINUE
```

Note that in the above example, two of the maximum six Lorentz frames are in use. They can be dropped by the statement `CALL QVDROP (' ', IVECT(IHEMI))` inside the loop (see 9.2.6).

## 10.10 Missing energy, mass, momentum

```
CALL QJMISS (PMISS, 'miss-vector', ICLASS, ITOTAL)
```

### Input arguments:

<b>ICLASS</b>	described in 10.3.
<b>ITOTAL</b>	= 0: Missing energy, etc. is calculated with respect to the total energy vector (0.,0.,0.,QELEP). > 0: Calculation is done with respect to vector ITOTAL.

### Results:

<b>PMISS</b>	Missing momentum.
<b>'miss-vector'</b>	vector containing missing momentum, mass, and energy.

### Error conditions:

- Total energy > LEP energy QELEP.
- Missing momentum > missing energy.
- In both cases, the output vector contains energy = PMISS and mass = 0.

## 10.11 Jet Finding

### 10.11.1 Scaled Invariant Mass Squared Algorithm

```
CALL QJMMCL (NJETS, 'name', ICLASS, YCUT, EVIS)
```

A loop runs over all pairs of tracks and finds the pair which has the smallest invariant mass  $M$ . If  $(M/EVIS)^2 < YCUT$ , these 2 tracks are merged (i.e., 4-momenta added).

The loop is then rerun over the new list of tracks which has lost 2 particles and gained the merged pair. When no remaining pair has a low enough mass, the track list contains a set of merged tracks called jets.

The mass  $M$  of 2 tracks is defined as  $M^2 = 2E_1E_2(1 - \cos\theta_{12})$ .

### Input arguments:

**ICLASS** described in 10.3.  
**YCUT** Cut on the scaled invariant mass of 2 tracks. Pairs of tracks are merged if their scaled invariant mass is smaller than YCUT.  
**EVIS** The visible energy of the event. If EVIS equals 0 , the visible energy is computed as the sum of the input particle energies.

### Results:

**NJETS** is the number of "jets" .  
**'name'** Vectors containing 4-momenta of the jets.

### EXAMPLE:

```

      DIMENSION LISTEJ(300)
      CHARACTER*13 CNAM
C---Select option: charged tracks
      CALL QJOPTR('CH',' ')
C---calculate visible energy from input tracks:
      EVISRE = 0.
      YCUT = 0.02
      CALL QJMMCL(NJT,'MMCLUS_RE_vis',KRECO,YCUT,EVISRE)
      CNAM = 'MMCLUS_RE_vis'
      WRITE(KUPRNT,*) '# of jets reconstructed ', CNAM, ':', NJT
      IF(NJT.GT.0) THEN
C--- get ALPHA number for first jet found:
      JJ = KPDIR(CNAM,KRECO)
      20  IF(JJ .NE. 0) THEN
C--- get the list of tracks merged into this jet:
      LL = 0
      DO 211 L = KFCHT, KLCHT
C--- check if this track belongs to this jet:
      IF(.NOT.XSAME(JJ,L)) GOTO 211
      LL = LL + 1
      LISTEJ(LL) = L
      211  CONTINUE
      WRITE(KUPRNT,*) 'Jet # ', J
      WRITE(KUPRNT,*) QX(JJ),QY(JJ),QZ(JJ),QE(JJ)
      WRITE(KUPRNT,*) 'List of tracks merged into this jet:'
      WRITE(KUPRNT,*) (LISTEJ(L),L=1,LL)
C--- get ALPHA number for next jet found:
      JJ = KFOLLO(JJ)
      GOTO 20
      ENDIF
      ENDIF
```

### 10.11.2 Scaled Minimum Distance Algorithm

CALL QJMDCL (NJETS, 'name', ICLASS, ALPHA, DELTA, ETA, EVIS)

A loop runs over all pairs of tracks and finds the pair which has the smallest invariant mass  $M$ . If  $(M/EVIS^\alpha)^2 < \sqrt{2(1 - \cos 2\delta)}$ , these 2 tracks are merged (i.e., 4-momenta added). The loop is then rerun over the new list of tracks which has lost 2 particles and gained the merged pair. When no remaining pair has a low enough mass, the track list contains a set of merged tracks. If these tracks have energies bigger than  $2\eta Evis$ , they are called jets. The mass  $M$  of 2 tracks is defined as  $M^2 = 2(E_1 E_2)^\alpha (1 - \cos \theta_{12})$ .

#### Input arguments:

<b>ICLASS</b>	described in 10.3.
<b>ALPHA</b>	Weight of track energies and $Evis$ , in the calculation of the scaled mass. Pairs of tracks are merged if their scaled mass is smaller than $\sqrt{2(1 - \cos 2\delta)}$ .
<b>DELTA</b>	Half opening angle cut in degrees.
<b>ETA</b>	Cut on jet energies (fraction of $2Evis$ ); only jets with energies $> 2\eta Evis$ are kept.
<b>EVIS</b>	The visible energy of the event; if $EVIS$ equals 0, the visible energy is computed as the sum of the input particle energies.

#### Results:

<b>NJETS</b>	is the number of "jets".
<b>'name'</b>	Vectors containing 4-momenta of the jets.

### 10.11.3 JETSET algorithm LUCLUS from LUND

CALL QJLUCL (NJETS, 'name', ICLASS, MINCLU, DMAX1, DMAX2, MULSYM, TGEN, DMIN)

#### Input arguments:

<b>ICLASS</b>	described in 10.3.
<b>MINCLU</b>	Minimum number of clusters to be reconstructed. (if $< 0$ , work space momenta are used as a start) (usually=1)
<b>DMAX1</b>	Max. distance to form starting clusters (usually=0.25GeV)
<b>DMAX2</b>	Max. distance to join 2 clusters (usually=2.5 GeV)
<b>MULSYM</b>	<ul style="list-style-type: none"><li>• = 1 for symmetric distance criterion (usual)</li><li>• = 2 for multicity distance criterion</li></ul>

**Results:**

**NJETS** is the number of “jets”  
• = -1 if not enough particles  
• = -2 if not enough working space (KTBOMX)

**TGEN** Generalized thrust

**DMIN** Minimum distance between 2 jets  
• = 0 when only 1 jet  
• = -1 , -2 as for NJET

**'name'** Vectors containing 4-momenta of the jets.

#### 10.11.4 PTCLUS: Jet-finding algorithm

CALL QJPTCL (NJETS,'name',ICLASS,NJTLIM,YJTLIM,EVIS)

The PTCLUS jet-finding algorithm is described in ALEPH 89 – 150.

**Input arguments:**

**ICLASS** described in 10.3.

**NJLITM** maximum number of jets to search for; if NJLITM=0, the algorithm finds the number of jets using YJTLIM.

**YJLITM** maximum allowed distance between two clusters (in  $M^2 / \text{EVIS}^2$ ); 0.02 is a typical value.

**EVIS** visible energy. If EVIS=0, the visible energy is calculated.

**Results:**

**NJETS** is the number of “jets”. (-1 if algorithm fails)

**TGEN** Generalized thrust

**'name'** Vectors containing 4-momenta of the jets.

# Chapter 11

## Energy Flow

Three energy flow packages are currently in use in ALEPH: the mask algorithm of Minard and Pepe-Altarelli, the PCPA-based energy flow of Bonissent, and the ENFLW package of Janot. In this chapter, the ALPHA interfaces for these algorithms are described. Although all three packages are available in ALPHA, it is likely that only the ENFLW package will be supported in the future. Therefore, users are advised to use ENFLW energy flow.

### 11.1 ENFLW Energy Flow

To use the ENFLW energy flow analysis, the EFLW card must be given in the ALPHA card file.<sup>1</sup> If the EFLW card is present, the EFT section of ALPHA will be filled with selected charged tracks and neutral ECAL and HCAL clusters. These objects can be accessed with DO loops (KFEFT, KLEFT, KNEFT – see 7.1.1) or with the particle name ‘EFLW’ using the functions KPDIR and KFOLLO (described in 7.4). The charged tracks that appear in the EFT section are copies of standard ALPHA charged tracks from the CHT section. Therefore, if a charged track in the CHT section is locked (using QLTRK or QLOCK), the corresponding track in the EFT section will be locked also (and vice versa). All statement functions providing information about charged tracks can be used directly with charged tracks in the EFT section – it is not necessary to find the corresponding track in the CHT section first, as was required with the UPHY version of the ENFLW package.

The following statement functions may be used to access additional information on EFLW objects:

**XEFO (I)**      **.TRUE.** if energy flow (EFOL) data are available for “track” I

**KEFOTY (I)**    Type of energy flow object:

- 0 = Track
- 1 = Electron
- 2 = Muon
- 3 = Track from V0
- 4 = Electromagnetic

---

<sup>1</sup>As stated in Appendix C, additional libraries must be linked to use ENFLW with the DST. ALPHARUN users will be asked whether they want to use ENFLW or QMUIDO with DSTs when they run the exec – the proper libraries will then be linked automatically.



- 5 = ECAL hadron/residual
- 6 = HCAL element
- 7 = LCAL element

**KEFOLE (I)**    PECO number of associated ECAL object  
**KEFOLT (I)**    FRFT number of associated charged track  
**KEFOLH (I)**    PHCO number of associated HCAL object  
**KEFOLC (I)**    PCOB number of associated calorimeter object  
**KEFOLJ (I)**    EJET number of associated jet

To use the event topology routines described in Chapter 10 with these energy-flow objects, use option 'EF' with subroutine QJOPTR (see 10.1):

```
CALL QJOPTR('EF', ' ')
```

**Example:**

The following code calculates the total energy of an event and finds the thrust using energy flow objects.

```

E=0.
DO 10 I = KFEFT, KLEFT
  E=E + QE(I)
10 CONTINUE
C--- Find thrust
CALL QJOPTR('EF', ' ')
CALL QJTHRU(THRU, 'THRU', KRECO)

```

Jets based on energy flow objects using QJMMCL with YCUT = 0.003 (see Sec. 10.11.1) are stored in the EJET bank. If the EFLJ card is used instead of the EFLW card, the EFT section will be filled as described above, and these jets will be stored in the JET section. The jets may be accessed with DO loops (KFJET, KLJET, KNJET) or with the particle name 'EJET' using the functions KPDIR and KFOLLO. The energy flow objects making up these jets can be found with XSAME as described in Sec. 8.1.7. To save time, these jets may be used as input for jet-finding with a higher YCUT (see 10.11.1) by calling QJOPTR with the option EJ:

```
CALL QJOPTR('EJ', ' ').
```

XSAME may be used to find the original energy flow objects (in the EFT section) making up the final jets.

## 11.2 Mask Energy Flow

The mask energy flow is not available on the MiniDST, and may eventually be dropped from the DST. Therefore, users are advised to use the ENFLW energy flow described above.

The use of this algorithm is identical to that of the ENFLW algorithm except that the EFLW card must be used with option 2:

**EFLW 2**

(Similarly, the card **EFLJ 2** must be used instead of **EFLJ**.)

If EFLW 2 is present, the EFT section will be filled with the results of the mask energy flow. In this case, all of the statement functions described above, as well as the event topology routines, will apply to the results of the mask energy flow.

Energy-flow properties (eg., total energy) calculated from the mask-based energy flow analysis done in JULIA are stored in the DHEA bank and are available in ALPHA as fortran variables (see 6.3.6).

### 11.3 PCPA-based Energy Flow

The PCPA-based energy flow uses neutral objects derived from the PCPA bank in addition to selected charged tracks. The logical function XFRIQF(ITK) may be used to test whether a track has been included for the PCPA energy-flow analysis. The PCPA neutral objects are stored in the NET section by default. (Filling of the NEOB section may be disabled by including the card NOPC in the ALPHA card file.) These objects can be accessed with DO loops (KFNET, KLNET, KNNET – see 7.1.1) or with the particle name ‘NEOB’ using the functions KPDIR and KFOLLO (described in 7.4).

To use the event topology routines described in Chapter 10 with PCPA-based energy flow (*i.e.*, selected charged tracks plus PCPA neutral objects), use option ‘PC’ with subroutine QJOPTR (see 10.1):

```
CALL QJOPTR('PC', ' ').
```

To use only NEOB objects:

```
CALL QJOPTR('NO', 'NEOB').
```

The following statement functions may be used to access additional information on NEOB objects:

**XPCQ (I)**      **.TRUE.** if PCQA data are available for “track” I

**KPCQNA (I)**    Nature of neutral object (see Sec. 11.3)

- 1 Isolated gamma
- 2 Gamma from multi-gamma neutral cluster
- 3 Gamma from identified  $\pi^0$
- 4 Gamma from electron bremsstrahlung
- 5 Gamma from electromagnetic charged cluster

- 10 Unresolved gamma-gamma
- 12 Residual electromagnetic energy from neutral cluster
- 13 Residual electromagnetic energy from charged cluster
- 17 Neutral hadron
- 18 Residual hadronic energy from neutral cal object
- 19 Residual hadronic energy from charged cal object with no HCAL component
- 20 Residual hadronic energy from charged cal object with HCAL component
- 21 contribution from an ECAL cluster for which EBNEUT was in error
- 22 contribution from an LCAL object

**Example:**

The following code calculates the total energy of an event and finds the sphericity using PCPA-based energy flow.

```

C---   First add up neutral energy
      E = 0.
      DO 10 I = KFNET, KLNET
      E=E+QE(I)
10 CONTINUE
C---   Add energies of selected tracks
      DO 20 I = KFCHT, KLCHT
      IF(XFRIQF(I)) E = E + QE(I)
20 CONTINUE
C---   Find sphericity
      CALL QJOPTR('PC', ' ')
      CALL QJSPHE(SPHE, 'SPHE', KRECO)

```

## Chapter 12

### Other ALPHA Physics Routines

#### 12.1 dE/dx Analysis

##### 12.1.1 Calculate dE/dx for Track ITK

```
CALL QDEDX(ITK,NHYP,RMASS,Q,RI,NS,TL,RIEXP,SIGMA,IER)
```

This routine is an ALPHA interface to the ALEPHLIB routine TIDHYP. Note that the user must check the return code IER before trying to use any of the output arguments – not all charged tracks have dE/dx information!

##### Input arguments:

<b>ITK</b>	ALPHA track number of a charged reconstructed track.
<b>NHYP</b>	Number of hypotheses the user wishes to try. If NHYP=1, then RMASS, Q, RIEXP, and SIGMA may be scalar variables.
<b>RMASS(nhyp)</b>	Array of masses, one for each hypothesis.
<b>Q(nhyp)</b>	Array of charges, one for each hypothesis.

##### Output arguments:

<b>RI</b>	The measured truncated mean ionization, normalized such that RI=1 corresponds to minimum ionizing.
<b>NS</b>	Number of useful wire samples on the track.
<b>TL</b>	Useful length of the track (cm).
<b>RIEXP(nhyp)</b>	Expected ionization for each mass hypothesis, normalized such that RIEXP=1 corresponds to minimum ionizing.
<b>SIGMA(nhyp)</b>	One standard deviation resolution error for each hypothesis. This is the expected dE/dx resolution, given NS, TL, RIEXP, and the momentum resolution. <b>Note</b> that one can calculate a $\chi^2$ with 1 d.o.f. as: $\chi^2 = ((RI - RIEXP) / SIGMA)^2$
<b>IER</b>	Error return code=0: successful return.

- =1: cannot find the track, or ITK is not a charged KRECO track.
- =2: cannot find the measured dE/dx information (bank TEXS).
- =3: input KRECO charged track has no dE/dx information.
- =4: cannot find the necessary database calibration banks, TC1X, TC2X, and/or TC3X.
- =5: cannot find RUNH or EVEH bank
- =6: there is no valid dE/dx calibration for this run

### 12.1.2 Modified QDEDX for Monte Carlo

```
CALL QDEDXM(ITK,NHYP,RMASS,Q,RI,NS,TL,RIEXP,SIGMA,IER)
```

This routine serves the same purpose as QDEDX, but treats Monte Carlo differently. QDEDX takes the dE/dx from the detailed simulation program TPCSIM. QDEDXM, however, only takes the number of samples and the track length from TPCSIM, from which a prediction for the resolution is obtained. The measured momentum and the Monte Carlo true mass then are used to predict the mean dE/dx, which is smeared by a gaussian random number to give the simulated dE/dx.<sup>1</sup> The advantage of this approach is that it is easy to adjust the parameterization to give agreement with data, whereas to do so with TPCSIM is nontrivial and would require regeneration of the Monte Carlo data set. The disadvantage is that the non-gaussian tails (which are small and arise primarily on the high side, due to unresolved track overlap) are not simulated. An option does exist to try to get the best of both worlds: by calling QMTAIL one can set a parameter to tell QDEDXM to retain the tail simulated by TPCSIM beyond a specified number of standard deviations. The distribution below that number of standard deviations then is obtained from the gaussian random number generator. Clearly this solution is not perfect, since the distributions generally will not match at the chosen cut value.

The arguments to this routine are identical to those of QDEDX. When QDEDXM is used on Monte Carlo events, error code 6 means that no Monte Carlo truth information is available. Note that if QDEDXM is used with real data, it is identical to QDEDX.

```
CALL QMTAIL(CUT)
```

QMTAIL is an entry point in QDEDXM which can be used to set the cut value, in standard deviations, beyond which the dE/dx non-gaussian tail produced by TPCSIM is retained. By default, CUT is set to 999.

### 12.1.3 Check TPC High Voltage for dE/dx

```
LOGICAL FUNCTION TCHKHV(KRUN,KEVT,IFLG)
```

The function TCHKHV, from the ALEPHLIB, is used to check the TPC high voltage before using the dE/dx information from the TPC. It checks the data base bank TDBS to find whether

<sup>1</sup>If this routine is called more than once for the same Monte Carlo event, the results will be different because a different random number will be used.

any sectors were being intentionally operated at reduced voltage during the run in question. If so, then only the normal TPC HV bit is checked. Otherwise, the “dE/dx” HV bit is checked.

**Input arguments:**

**KRUN**                    ALEPH run number.  
**KEVT**                    ALEPH event number.

**Output arguments:**

**IFLG**                    What kind of test was made?  
                              • =0: test was made on dE/dx HV bit.  
                              • =1: test was made on TPC tracking HV bit.  
                              • =2: no test was made (banks not found). TCHKHV=.FALSE.  
**TCHKHV**                = .TRUE. if HV is on; .FALSE. otherwise.

#### 12.1.4 Check Existence of dE/dx Calibration for Run

LOGICAL FUNCTION TCHKEX(KRUN)

TCHKEX returns .TRUE. if a valid dE/dx calibration exists run KRUN. If a valid calibration does not exist (because it has not yet been done or because the run was bad), then TCHKEX returns .FALSE. This routine resides in the ALEPHLIB.

## 12.2 Photon conversions

CALL QPAIRF (I1,I2,DXY,DZ0,DZ2,DTH,RMA,ZMA,XMA,NC1,DIN1,NC2,DIN2,P,IER)

This routine is an ALPHA interface to the ALEPHLIB routine PAIRFD. Electrons from photon conversion initially will have parallel trajectories. This algorithm finds the point on each helix where the tracks are parallel in the X–Y plane and pass closest together; this point is called the materialization point. Note that photon conversions are also found in JULIA, and are available as V0s (see Sections 7.1 and 8.1.2).

**Input arguments:**

**I1**                        ALPHA track number of a charged track.  
**I2**                        ALPHA track number of a another charged track.

### Output arguments:

<b>DXY</b>	distance(cm) in the xy plane between the two tracks at the closest approach to the materialization point.
<b>DZ0</b>	Distance(cm) in z between the two tracks at the origin.
<b>DZ2</b>	The z separation of the tracks at the closest approach to the materialization point.
<b>DTH</b>	the theta difference of the two tracks.
<b>RMA</b>	the rho value at the materialization point.
<b>ZMA</b>	the z value at the materialization point.
<b>XMA</b>	The invariant mass of the tracks at the materialization point assuming they are both electrons.
<b>NC1,2</b>	Number of coordinates with radius less than RMA for track 1,2. 0 if no coordinate information is available or if there are no such coordinates.
<b>DIN1,2</b>	Radial distance between the coordinate closest to the origin and RMA for track 1,2; variable is 0. if no coordinate information is available or if there are no such coordinates.
<b>P(3)</b>	Summed momentum of the two tracks at the materialization point in the order x,y,z.
<b>IER</b>	= 0 if calculation is successful; 1 otherwise.

## 12.3 Muon Identification: QMUIDO

```
CALL QMUIDO(ITK,IRUN,IBE,IBT,IM1,IM2,NEXP,NFIR,N10,N03,  
XMULT,RAPP,ANG,ISHAD,SUDNT,IDF,IMCF,IER)
```

This routine is an ALPHA interface to the ALEPHLIB routine AMUID.<sup>2</sup> This routine simply collects useful information from the banks HMAD, MCAD, and MUID. For users who only look at the identification flag IDF, this routine is redundant since the flag is now stored in the bank MUID and can be accessed with the statement function KMUIxx (see section xx). Users are encouraged to make use of the access via statement functions. This routine can be used to look in more detail at the muon identification or to provide backwards compatibility with the QMUIDO version 7.0 (UPHY) calling sequence.

### Input argument:

**ITK** ALPHA track number of a charged reconstructed track.

---

<sup>2</sup>As described in Appendix C, additional libraries must be linked to use QMUIDO (and the MUID bank) with pre-1992 DSTs.

**Output arguments:**

<b>IRUN</b>	No longer filled (needed for backwards compatability)
<b>IBE</b>	Bitmap of the planes EXPECTED to have fired in the HCAL
<b>IBT</b>	Bitmap of the planes which have fired in the HCAL
<b>IM1</b>	Number of associated muon chamber hits in the inner layer
<b>IM2</b>	Number of associated muon chamber hits in the outer layer
<b>NEXP</b>	Number of planes expected to have fired in the HCAL
<b>NFIR</b>	Number of planes fired in the HCAL
<b>N10</b>	Number of planes fired in the last ten expected HCAL planes
<b>N03</b>	Number of planes fired in the last three expected HCAL planes
<b>XMULT</b>	Excess hit multiplicity in the last ten planes on the HCAL
<b>RAPP</b>	Distance between track extrapolation and closest muon chamber hit in standard deviations (the distribution is only approximately normal)
<b>ANG</b>	Angle between track extrapolation and closest muon chamber hits in standard deviations (the distribution is only approximately normal). Only available for tracks with at least one muon chamber hit in each layer
<b>ISHAD</b>	Shadowing flag = 0 if track is not shadowed; otherwise it is the JULIA track number of the shadowing track.
<b>SUDNT</b>	Sum of HCAL hit to track residuals in the last 10 planes.
<b>IDF</b>	Official muon identification flag. <ul style="list-style-type: none"><li>• = 1 if muon flagged only by HCAL</li><li>• = 2 if muon flagged only by MUON</li><li>• = 3 if muon flagged by both HCAL and MUON 3 is the .AND. of 1 and 2</li><li>• = 10 is one hit in each layer of MUON chambers but failing tight matching criteria</li><li>• = 11 is good HCAL pattern</li><li>• = 12 is one and only one MUON hit</li><li>• = 13 is good HCAL + one and only one muon</li><li>• = 14 is good HCAL + one hit in each layer</li><li>• = 15 is one hit in each layer of MUON chambers passing tight matching criteria</li><li>• = 0 not a muon</li><li>• = -1 to -15 as above but lost shadowing contest</li></ul>
<b>IMCF</b>	Monte Carlo true source of this track <ul style="list-style-type: none"><li>• = 0 ambiguous or data</li></ul>



- = 1 primary b
- = 2 secondary c
- = 3 primary c
- = 4 b to tau
- = 5 other muon
- = 6 non decaying hadron or electron
- = 7 decay hadron

**IER** No longer filled (needed for backwards compatability)

## 12.4 Utility Routines for VDET Analysis

### 12.4.1 Number of VDET hits per layer for track ITK

`CALL QVDHIT(ITK,IVHIT)`

**Input argument:**

**ITK** ALPHA track number of a reconstructed charged track.

**Output argument:**

**IVHIT**

- IVHIT(1) Number of VDET hits in  $r - \phi$  on inner layer
- IVHIT(2) Number of VDET hits in  $r - \phi$  on outer layer
- IVHIT(3) Number of VDET hits in  $z$  on inner layer
- IVHIT(4) Number of VDET hits in  $z$  on outer layer

### 12.4.2 VDET HV status

`LOGICAL FUNCTION XVDEOK(dummy)`

The function XVDEOK<sup>3</sup> returns `.TRUE.` if the VDET HV is on for the current event and `.FALSE.` otherwise. XVDEOK uses the HV bits and also calls KVGGOOD (see below).

If you want to use tracks with high quality VDET data, first, check that the HV is on and second, check that the tracks in which you are interested have VDET hits (see QVDHIT above).

---

<sup>3</sup>Until ALPHA 115 is released, XVDEOK must be declared LOGICAL by the user.

### 12.4.3 VDET Readout Status

INTEGER FUNCTION KVGGOOD(dummy)

During several periods in 1991 and 1992, there were problems with the VDET readout which caused VDET information to be read out when the HV was off. The VDET hits read out during these periods are just noise and can distort tracks fitted with the VDET. The function KVGGOOD identifies whether the current event is in one of the bad periods, and if so, whether or not the HV was on.

**KVGGOOD** readout and HV status:

- = 0: no readout problems, HV is either ON or OFF.
- = +1: readout problems, HV is ON.
- = -1: readout problems, HV is OFF.

## Chapter 13

# ALPHA Utility Routines: Printing, Writing Events, Timing, etc.

### 13.1 Program termination

`CALL QMTERM ('any message')`

Can be called from anywhere.

**Calls** QUTERM, QUTHIS, QWMESS.

**Input argument:**

**'any message'** character string, The message will be printed and should contain the reason for the program termination.

### 13.2 Write the current event on the output file

`CALL QWRITE`

The file name is specified on the FILO card (see 4.1.3). This routine can be called from user routines; it is called automatically from QMEVNT if the COPY option (4.1.5) is selected. If QWRITE is called more than once for the same event, the event will be written only once.

### 13.3 Set classification word written to event directory

`CALL QWCLAS (IBIT)`

**Input argument:**

**IBIT** Turn on bit IBIT in classification word. IBIT = 1 – 30.

QWCLAS has to be called once for each bit which is to be set; i.e., if three bits are to be set, QCLASW has to be called three times. Note that a call to QWCLAS simply turns on a single bit while leaving other bits unchanged. The initial classification word is the one read from the input file; therefore, the classification word must be zeroed before storing your own values:

```
CALL QWCLAS(0).
```

If QWCLAS is not called, the classification word will be set equal to that on the input file.

## 13.4 Timing

### 13.4.1 Print job time consumption

```
CALL QWTIME
```

Called automatically from QMTERM.

### 13.4.2 Measure time consumption of part of program

```
CALL QTIMED(INUM)
```

This subroutine measures the time between two subsequent calls to QTIMED. Time statistics can be kept for up to 9 different subroutine calls (INUM = 1 - 9). The time consumption summary is printed automatically during job termination. The summary includes the number of calls and the total time / call. The first CALL QTIMED sets the start time. The time consumption for QTIMED itself (0.25 msec on CERNVM) is not subtracted in the time summary. (The CERNLIB routines TIMED/TIMAD should not be used with QTIMED.)

#### Example:

```
CALL QTIMED(1)
CODE A
CALL QTIMED(2)
CODE B
CALL QTIMED(3)
```

#### Results:

QTIMED	Ncalls	total_time	time/call	%
1	499	35	0.07	92.2
2	500	1	0.002	2.6
3	500	2	0.004	5.2

In this example, time 2 gives the time consumption of 'CODE A', time 3 gives the time consumption of 'CODE B', and time 1 gives the time consumption between CALL QTIMED(3) and CALL QTIMED(1).

## 13.5 Print routines

The routines described in this section are used to print information about events or to print messages. Some of the routines have the subroutine argument **'option'**.

**'option'** is composed of one or several characters. Each character has a special meaning:

- 'H'** print a header line. Without this option, you will get a sequence of numbers without any description. With this option, an extra line containing the mnemonic symbols for the numbers given underneath is printed.
- '0'** print an empty line and the header line.
- '1'** start at a new page and print the header line.
- ' '** blank space = no option

More options can be given for specific subroutines.

### 13.5.1 Print a message

`CALL QWMESS ('any message')`

#### Input argument:

**'any message'** (character string or character variable) If the 1st character of 'any message' is '0' or '1', it is taken as carriage control character ('0': empty line; '1': new page). If it is not '0' nor '1', it is taken as part of the message.

### 13.5.2 Print a message plus run, event number

`CALL QWMESE ('any message')`

### 13.5.3 Print full event summary (many pages)

`CALL QWEVNT`

Calls QWHEAD, QWSEC, QWTREE

### 13.5.4 Print event header (one line)

CALL QWHEAD ('option', 'any text')

#### Input arguments:

'option' one of 'H', '0', or '1' (see 13.5)

'any text' message; may be blank space: ' '

**Output** see printer output of QWHEAD called with option 'H'. Here, as in many other print routines, it's a matter of taste which data are important enough to be printed, and comments are welcome. For better readability, the output should always fit onto one printer line.

### 13.5.5 Print full event header (many lines)

CALL QWHFUL ('option', 'any text')

Subroutine arguments are the same as for QWHEAD.

### 13.5.6 Print information for "track"

CALL QWITK (ITK, 'option')

#### Input arguments:

ITK ALPHA track number.

'option' one of 'H', '0', or '1' (see 13.5). 'L': Do not print locked tracks.

**Output** see printer output when called with option 'H'.

Meaning of column "det. data":

**F** general track fit data are available

**T** dE/dx data are available

**H** HCal data are available

**M** muon chamber data are available

**E** Ecal data are available

**H** Hcal data are available

... **rightmost characters:**

**C** object is associated to one or several charged tracks

**E** object is associated to one or several Ecal objects

**H** object is associated to one or several Hcal objects

### 13.5.7 Print information for vertex

CALL QWIVX (IVX, 'option')

#### Input arguments:

**IVX**            ALPHA vertex number.  
**'option'**        one of 'H', '0', or '1' (see 13.5)  
**Output**        see printer output when called with option 'H'.

### 13.5.8 Print summary for categories of tracks or vertices

CALL QWSEC (ISEC, 'option')

Calls QWITK, QWIVX

#### Input arguments:

**ISEC**            section number = section in QVEC and QVRT:  
                  **KSOVT**    Overlap objects  
                  **KSCHT**    Charged tracks  
                  **KSIST**    Isolated = neutral cal objects  
                  **KSAST**    Cal objects associated to charged tracks  
                  **KSV0T**    Neutral tracks pointing to reconstructed vertices  
                  **KSDCT**    Tracks outgoing from reconstructed vertices  
                  **KSEFT**    Energy flow objects  
                  **KSNET**    Neutral objects from PCPA  
                  **KSGAT**    Photons from GAMPEC  
                  **KSJET**    Jets from energy flow objects  
                  **KSMCT**    MC particles  
                  **KSREV**    Reconstructed vertices  
                  **KSMCV**    MC vertices.  
**'option'**        one of '0', or '1' (see 13.5). 'L': Do not print locked tracks.

### 13.5.9 Print decay tree of track ITK.

CALL QWTREE (ITK, 'option')

#### Input arguments:

**ITK**            Track / particle number. to the output)  
**'option'**        one of 'H', '0', or '1' (see 13.5)  
**Output:**        Similar to CALL QWITK.

## Chapter 14

### Modifying ALPHA banks

ALPHA subroutines provide protection against inadvertently overwriting data read from the input file. In this section, we describe how to modify the internal ALPHA banks (QVEC and QVRT) intentionally. For “standard” operations (creating new tracks, vector operations, Lorentz transformations, etc.), ALPHA utility routines are available (see ch. 9). The tools described here can be used when standard utilities do not exist.

#### 14.1 User track / vertex sections

The subroutines QSUSTR or QSUSVX may be used to reserve certain track / vertex numbers for your own exclusive usage; they will never be modified by any ALPHA utility routine unless explicitly required. These routines may be called from the user initialization routine QUNIT.

##### 14.1.1 Reserve user space for tracks

```
CALL QSUSTR (NUSTR)
```

Note that ALPHA does not clear (zero) this user space after each event.

##### Input argument

**NUSTR**                    number of user tracks in bank QVEC

- The track numbers reserved are 1 ... NUSTR.
- The first track number used in any ALPHA routine will be NUSTR+1.

User track space is allocated only if this routine is called.



### 14.1.2 Reserve user space for vertices

`CALL QSUSVX (NUSVX)`

Same as QSUSTR (14.1.1) : Replace “track” by “vertex” and “QVEC” by “QVRT”.

Utility routines can be called with user tracks as arguments. For these tracks, only the basic attributes (columns 1 to 7) are modified : QX,QY,QZ,QE,QP,QM,QCH. All other columns are left unchanged (and NOT set to 0!).

## 14.2 Modifying track / vertex attributes

All internal ALPHA banks are standard tabular BOS banks and can be modified like other banks. For the banks QVEC and QVRT, an additional possibility is foreseen: these banks are passed as arguments to subroutine QUEVNT and can be used as ordinary 2–dimensional arrays.

```
SUBROUTINE QUEVNT (QT,KT,QV,KV)
  DIMENSION QT(KCQVEC,1), KT(KCQVEC,1), QV(KCQVRT,1), KV(KCQVRT,1)
  ...
  QT(JQVEQP,ITK) = 1.
  CALL ABC (QT,KT,QV,KV)
END
...
SUBROUTINE ABC (QT,KT,QV,KV)
```

**Remarks :** QT and KT (tracks) refer to the same array (integer / real\*4) and actually to the address of the bank QVEC plus a 2–word offset for the bank header (LMHLEN). QV and KV are defined similarly for bank QVRT (vertices).

**Dimension :** Use the mnemonic symbols KCQVEC and KCQVRT (Fortran parameters defined in QCDE) for the number of columns. The number of rows can be set to any positive number.

**QT(JQVEQP,ITK) :** Row number = ALPHA track number. column number = attribute. For all attributes, parameters are available in QCDE. The parameter names follow the usual convention (see App. B). “J” + 3 char. of the bank name + 2 char. attribute description.

# Chapter 15

## Particle Table

### 15.1 Description

The particle table contains the following particle attributes: nominal mass, charge, life time, width, and particle – antipart. relation.

In every ALPHA job, an internal particle table is built which combines data from the following sources:

- Data cards described below.
- The “standard” ALEPH particle table stored on the data base. This table contains all standard model particles (three generations) which can be produced at LEP energies, plus some exotic particles.
- The “MC” particle table stored in the run record of MC event files. This table contains the standard table, and if necessary, extra particles specific to the MC generator.

If particle attribute values from different sources do not agree, they are taken from data cards with highest and from the MC table with lowest priority. The standard printout produced at job termination indicates where the values come from.

New particles can be defined with the PNEW card (see below), or by using their names in ALPHA subroutine calls. If particles are created in subroutine calls, a warning message is printed.

### 15.2 Particle name, particle code

Particles can be specified either by their name (example: ‘GAMMA ’) or by their integer particle code.

**General rule:** Only the particle name is relevant. The integer code may change from one job to another; if you wish to use the integer code, it must be initialized in each job by calling the function (see 7.4.3): *integer = KPART ('part-name')*.

## 15.3 How to spell particle names

On data cards, every particle name (1 ... 12 characters) has to be terminated by exactly one blank space.

### Example

```
PMOD 'PI+ PI- ' 0.14      ! is correct !
PMOD 'PI+ PI-' 0.14      ! SERIOUS MISTAKE !
```

In the Fortran program, this extra blank space can be omitted or typed.

Lower case characters are translated into upper case characters. It would be wise, nevertheless, to use UPPER case characters only.

## 15.4 Data cards for particle table

### 15.4.1 PMOD: Modify particle attributes

**Format** *PMOD 'part-name antipart-name ' mass charge life-time width*

#### Parameters:

**'part-name antipart-name'** see 4.12.1. The attributes of a particle and its antiparticle are modified at the same time. If a particle is its own anti-particle, the same name has to be given twice.

**mass charge life-time width:** Real numbers (with decimal point). The charge of the antiparticle is set to  $-charge$ . If less than four numbers are given, the remaining particle attributes are not changed.

#### Example:

```
PMOD 'GAMMA GAMMA ' 0.001
```

sets the photon mass to 1 MeV; the other particle attributes (charge, lifetime, width) are not changed.

#### Mistake:

```
PMOD 'PI+ PI0 ' .14
```

because pi+ and pi0 are NOT antiparticles of each other. Once a particle-antiparticle relation is established (for example on the standard table), it can never be changed.

If the particle names given on this card are not yet established in the table then

- new table entries are created;
- a warning is issued;
- the program execution continues.

### 15.4.2 PNEW: New particles

**Format** *PNEW 'part-name antipart-name' mass charge life-time width*

PNEW has the same function as the PMOD card (15.4.1) and has the same parameters, except

- PNEW causes a warning if the particles are already known;
- PMOD causes a warning if the particles are unknown;

program execution continues in either case.

### 15.4.3 PTR A: Modify particle names in the MC particle table

The PTR A card assigns an arbitrary particle name to a specific MC integer code. It has to be used, for example, if different MC data sets with contradictory particle tables are read in one job.

The standard procedure to denote the nature of MC generated particles:

- Start with the integer code given for each generated particle.
- Get the corresponding particle name from the MC particle table.
- This name is relevant inside the ALPHA program.

**Format** *PTR A 'part-name antipart-name' iMCcode iMCanticode*

**Parameters:**

**'part-name antipart-name'** see 15.4.1. the names for the particle and its antiparticle which have to be used inside the ALPHA program.

**iMCcode:** integer particle code used in the MC generator (WITHOUT decimal point and NOT included inside the apostrophes.)

**iMCanticode:** integer particle code used by the MC generator for the corresponding antiparticle.

The routine QCPTR A is equivalent to the PTR A card and can be called from QUN EWR whenever a new MC particle table is read in.

## 15.5 Access to particle properties

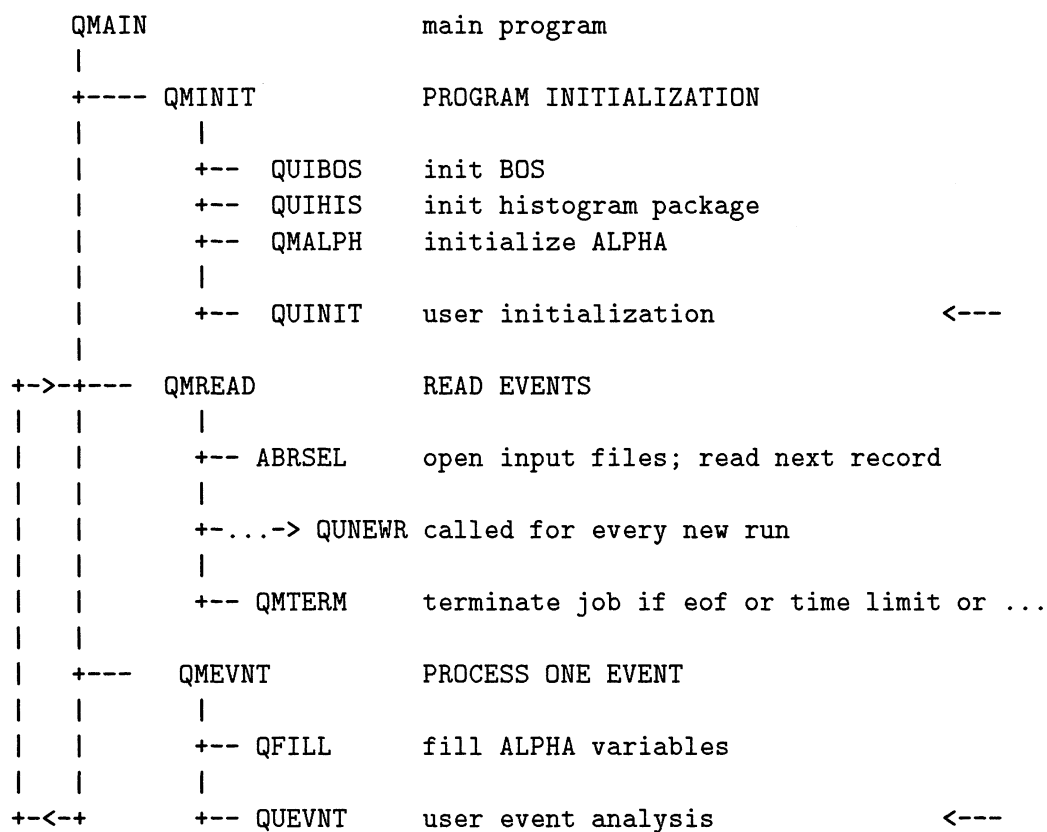
Inside an ALPHA job, particle properties can be obtained by specifying the particle either by name (symbols starting with the characters “QP” or “KP”) or by the integer code (“QC” or “KC”). The particle code has to be set by calling the function **ICODE = KPART ('part-name')** at least once per job. For more details, see 7.4.3.

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

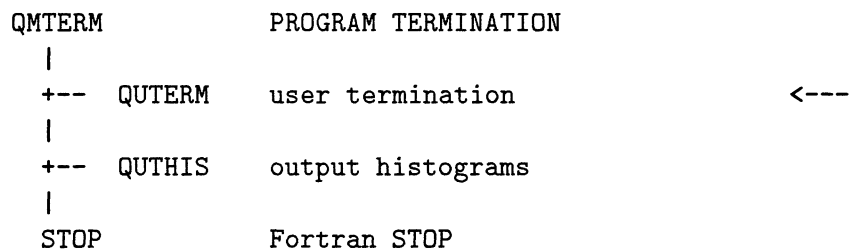
To check the particle names of ALPHA “tracks”, see sections 8.1.6 and 8.1.8.

# Appendix A

## Program Structure



called from anywhere :



Arrows (<---) indicate the important user routines.

# Appendix B

## Bank description

All banks described here must not be written to an output file.

```
*-----*
| QVEC |           TRACKS
*-----*
```

- 1 number of words / track
- 2 maximum allowed number of tracks

### Basic attributes

- 1 QX F PX
- 2 QY F PY
- 3 QZ F PZ
- 4 QE F Energy
- 5 QM F Mass
- 6 QP F momentum
- 7 CH F CHarge

### Flags

- 8 TN I JULIA / GALEPH Track Number
- 9 SC I stability code
- 10 KS I LUND status code
- 11 CL I track CLass
- 12 PA I ALEPH particle code
- 13 QD I offset for corresponding row in QDET (NOT the row number !)
  
- 14 NP I pointer to Next Particle / same particle code / same class
- 15 SP I Same Particle, different hypothesis or Lorentz frame
- 16 OV I Origin Vertex no
- 17 EV I End Vertex no
  
- 18 ND I Number of Decay particles
- 19 DL I offset of 1st daughter in particle List QLIN
- 20 NO I Number of mother particles

21	OL	I	offset of 1st mother in particle List QLIN
22	NM	I	Number of Matches
23	ML	I	Match list = row offset in banks QMTL and QMTS
24	BM	I	Bit masks
34	LK	I	QLITK flag
35	DB	F	Distance of closest approach to beam axis (if track pointing to the main vertex) or to the corresponding secondary vertex.
36	ZB	F	Z coordinate of point where DB is measured.
37	SD	F	Error**2 on DB
38	SZ	F	Error**2 on DZ
39	CB	F	Chi**2 corresponding do DB and DZ.

Error Matrix

40-49	EM	F	triangular covariance matrix
50	CF	F	chi**2 from last kinematical fit
51	EW	F	weight from energy flow analysis
52-69	US		User space

\*-----\*  
| QDET |  
\*-----\*

DETECTOR INFORMATION

1			number of words / track
2			maximum allowed number of tracks
1	AF	I	offset for corresponding row in bank FRFT (NOT row number !)
2	AL	I	offset for corresponding row in bank FRTL
3	NT	I	number of segments in bank TEXTS
4	AT	I	offset for corresponding rows in bank TEXTS (MAX : 5)
8	LT	I	last allowed AT address
9	AE	I	offset for corresponding row in bank EIDT
10	AH	I	offset for corresponding row in bank HMAD
11	AM	I	offset for corresponding row in bank MCAD
12	CF	I	calorimeter flag (<0 : ass, > 0 : isol, abs=1 : ECAL; =2 : HCAL)
13	EC	I	offset for corresponding row in bank PECO
14	HC	I	offset for corresponding row in bank PHCO
15	ET	I	offset for corresponding row in bank PEPT
16	FI	I	offset for corresponding row in bank FRID



17 NF I number of associated charged tracks  
 18 FL I offset of 1st ass. ch. track in list QLIN  
 19 NE I number of associated ECAL objects  
 20 EL I offset of 1st ass. ECAL object in list QLIN  
 21 NH I number of associated HCAL objects  
 22 HL I offset of 1st ass. HCAL object in list QLIN  
 23 LH I overlap - associated  
 24 EF I offset for corresponding row in bank EFOL  
 25 PC I offset for corresponding row in bank PCQA  
 26 EG I offset for corresponding row in bank EGPC  
 27 MU I offset for corresponding row in bank MUID

\*-----\*  
 | QVRT |  
 \*-----\*

VERTICES

1 number of words / vertex  
 2 maximum allowed number of vertices  
  
 1 VX F XPosition  
 2 VY F YPosition  
 3 VZ F ZPosition  
 4 VN I JULIA /GALEPH Vertex number  
 5 TY B vertex TYpe  
  
 6 IP I track number of Incoming Particle  
 7 ND I Number of Decay particles  
 8 DL I offset for decay particle list  
 9 AY I offset for corresponding row in YVOV  
 10 AF I offset for corresponding row in FVER  
  
 11-16 EM F triangular error matrix  
  
 17 CF F chi\*\*2 for vertex fit -- filled by KVFITN, KVFITV  
 18-25 ET F track-vertex error matrix  
 26-30 User space

\*-----\*  
 | QFPA |  
 \*-----\*

FIRST PARTICLE (FOR DIRECT ACCESS)

1 number of track classes = 8  
 2 maximum number of rows  
  
 1 xx I ALPHA track number

\*-----\*  
| QLIN |                    ONE - TO MANY PARTICLE RELATIONS  
\*-----\*                    (e.g. : daughter -> mother)

- 1    number of words / row = 1
- 2    maximum allowed number of relations

1    xx    I    ALPHA track number

\*-----\*  
| QMTL |                    MATCH LIST  
\*-----\*

- 1    number of columns = 1
- 2    maximum allowed number of track matches

1    xx    I    ALPHA Track number

\*-----\*  
| QMTS |                    NUMBER OF SHARED HITS IN MATCH LIST  
\*-----\*

- 1    number of columns = 1
- 2    maximum allowed number of track matches

1    xx    I    Number of shared hits

\*-----\*  
| QPAR |                    INTERNAL PARTICLE TABLE  
\*-----\*

- 1    number of words / particle = 10
- 2    maximum allowed number of particles

The attributes are exactly the same as in the PART bank.

\*-----\*  
| QPBT |                    PARTICLE FLAGS FOR BOOKKEEPING  
\*-----\*                    (Parallel to QPAR bank)

- 1    number of columns = 1
- 2    maximum allowed number of particles

1 xx I bit flag  
bit 1 : used in function KPC  
bit 2 : particle attributes set by a data card  
bit 3 : particle attributes set by MC table  
bit 4 : pseudo-particle  
bit 5 : particle defined on a PTR A card

\*-----\*

| QPLI | POINTERS FROM QPAR TO QFPA

\*-----\*

1 number of columns = 1  
2 maximum allowed number of particles

1 xx I pointer

\*-----\*

| QTRA | MC PARTICLE TRANSLATION TABLE

\*-----\*

1 number of columns = 1  
2 maximum allowed number of particles in MC table

1 xx I internal particle code

## Appendix C

### Where to find ALPHA at CERN

#### C.1 ALPHA on CERNVM

The files needed to run ALPHA on the IBM are on the PHY disk. Type: **GIME PHY** or add a line: **'EXEC GIME PHY'** in your **PROFILE EXEC** to get the PHY minidisk at login time. They are (vsn=three digit ALPHA version number):

- Fortran files
  - ALPHA**vsn **FORTRAN K** Fortran code of all ALPHA subroutines and functions
  - ALCOR**vsn **FORTRAN K** Fortran code of corrections to current ALPHA version
  - QUUSER FORTRAN K** Fortran code of the routines **QUINIT,QUEVNT,QUTERM** (model routines which have to be filled)
  - QCDE INC K**
  - QMACRO INC K** Include files **QCDE, QMACRO** (see 3.1.2)
- HISTORIAN files
  - ALPHA**vsn **OLDLIB K** Historian library
  - ALPHA**vsn **CORR K** Correction file
  - QUUSER INPUT K** Historian input for the routines **QUINIT,QUEVNT,QUTERM** (model routines which have to be filled)
- Input to the link step
  - ALCOR**vsn **TEXT K** Correction file
  - ALPHA**vsn **TXTLIB K** ALPHA library; Other required libraries: **ALEPHLIB, BOS77, MINI, CERNLIB**
  - To use ENFLW or QMUIDO on DST** **ALENFLW TEXT** must be linked, and the **ENFLW**vsn and **JULIA** libraries must be loaded
- ALPHA card file (sample)
  - ALPHA CARDS K**
- ALPHARUN command file (see Ch. 2)
  - ALPHARUN EXEC K** options stored in **LASTING GLOBALV A**
- ALPHA documentation
  - ALGUIDE TEX K** LATEX source for this document
  - ALPHA**vsn **NEWS K** Description of changes in new ALPHA versions

## C.2 ALPHA on VXCERN, ALWS

The files needed to run ALPHA on the VAX are in the PHY: directory (ALEPH\$GENERAL:[PHY]). They are (vsn = three digit ALPHA version number):

- Fortran files
  - PHY:ALPHA**vsn.**FOR** Fortran code of all ALPHA subroutines and functions
  - PHY:ALCOR**vsn.**FOR** Fortran code of corrections to current ALPHA version
  - PHY:QUUSER.FOR** Fortran code of the routines QUNIT,QUEVNT,QUTERM (model routines which have to be filled)
  - PHYINC:QCDE.INC**
  - PHYINC:QMACRO.INC** Include files QCDE, QMACRO (see 3.1.2)
- HISTORIAN files
  - PHY:ALPHA**vsn.**HLB** Historian library
  - PHY:ALPHA**vsn.**CORR** Correction file
  - PHY:QUUSER.INPUT** Historian input for the routines QUNIT,QUEVNT,QUTERM (model routines which have to be filled)
- Input to the link step
  - PHY:ALCOR**vsn.**OBJ** Correction file
  - PHY:ALPHA**vsn.**OLB** ALPHA library
  - PHY:ALPHA**vsn.**D.OLB** with /DEBUG option; Other required link libraries: ALEPHLIB, BOS77, MINI, CERNLIB
  - To use ENFLW or QMUIDO on DST** PHY:ALENFLW.OBJ must be linked, and the ENFLWvsn and JULIA libraries must be searched
- ALPHA card file (sample)
  - PHY:ALPHA.CARDS**
- ALPHARUN command file (see Ch. 2)
  - PHY:ALPHARUN.COM** options stored in ALFPARAM.OPTB (by default)
- ALPHA news
  - ALEPH\$GENERAL:[PHY.DOC]ALGUIDE.TEX** LATEX source for this document
  - PHY:ALPHA**vsn.**NEWS** Description of changes in new ALPHA versions

ALPHARUN facilitates the use of a set of VAX debugger command files which simplify ALPHA program debugging.

### Examples

**EXAMINE IW(512:515)** Standard VAX debug command; to be used for all Fortran variables and arrays.

**EVALUATE LMHLEN** LMHLEN is a parameter and NOT a variable; use EVA instead of EXA.

**QP(ITK)** Debugger commands are defined for almost all mnemonic symbols which have one or more arguments (see Ch. 6). In this context, “QP” is a debugger command which has to be followed by the same argument(s) (given as numbers or variable names) as the mnemonic symbol QP in Fortran.

### C.3 ALPHA on the CRAY

The files needed to run ALPHA on the CRAY are stored on the CRAYXU 401 disk on CERNVM; many files are also kept in the \$KEEP area on the CRAY. They are (vs<sub>n</sub>=three digit ALPHA version number):

- Fortran files
  - ALPHA<sub>vs<sub>n</sub></sub> CRAYFOR** Fortran code of all ALPHA subroutines and functions
  - ALCOR<sub>vs<sub>n</sub></sub> CRAYFOR** Fortran code of corrections to current ALPHA version
  - QUUSER CRAYFOR** Fortran code of the routines QUNIT,QUEVNT,QUTERM (model routines which have to be filled)
- Input to the link step
  - ALCOR<sub>vs<sub>n</sub></sub> CRAYFOR** Corrections are fetched from CERNVM and recompiled for each job
  - ALPHA<sub>vs<sub>n</sub></sub> CRLIB or \$KEEP/alpha114.lib** ALPHA library Other required link libraries: ALEPHLIB, BOS77, MINI, CERNLIB
  - To use ENFLW or QMUIDO on DST** \$KEEP/alenflw.obj must be linked, and the ENFLW<sub>vs<sub>n</sub></sub> and JULIA libraries must be searched
- ALPHA card file (sample)
  - ALPHA CARDS**
- CRALPHA command file to submit jobs to CRAY from CERNVM (see Ch. 2)
  - CRALPHA EXEC G (on IBM)** options stored in LASTING GLOBALV A

### C.4 ALPHA on DECstations, SHIFT

The following files are needed to run ALPHA on UNIX machines other than the CRAY. All files refer to the current ALPHA version.

- Fortran files
  - /aleph/phy/srcalpha/\*.f** Fortran code of all ALPHA subroutines and functions

**/aleph/phy/alcor.f** Fortran code of corrections to current ALPHA version

**/aleph/phy/qcde.inc**

**/aleph/phy/qmacro.inc** Include files QCDE, QMACRO (see 3.1.2)

- Input to the link step

**/aleph/phy/alcor.o** Correction file

**/aleph/lib/libalpha.a** ALPHA library Other required link libraries: ALEPHLIB, BOS77, MINI, CERNLIB

**To use ENFLW or QMUIDO on DST** /aleph/phy/alenflw.o must be linked, and the ENFLWvsn and JULIA libraries must be searched

- ALPHA card file (sample)

**/aleph/phy/alpha.cards**

- ALPHA command file (see Ch. 2)

**alpharun** command file on DECstations and SHIFT

**SFALPHA EXEC K (on IBM)** submits jobs from CERNVM to SHIFT

# Appendix D

## Using the Mini-DST with ALPHA

In this appendix, a brief introduction to the use of the Mini-DST with ALPHA is given. For more details, consult the current ALEPH Mini-DST User's Guide (version 8.0 or later).

### D.1 Doing analysis with the Mini

The Mini-DST file(s) to be read (data or event directory) should be declared with a standard FILI card. ALPHA will convert the Mini-DST banks to POT banks, and the ALPHA variables will be filled. The logical XMINI will be set to `.TRUE.`.

Since the Mini-DST contains many of the basic quantities from the POT/DST, most of the ALPHA quantities are available from the Mini-DST, and most of the ALPHA code will work. Below is a list of quantities which are NOT available on the Mini-DST – there may be others.

- XTCN: all
- FRFT: KFRFNO.
- FRTL: KFRTNE, KFRTNR.
- FRID: KFRIBC, KFRIBC, KFRIPE, KFRIPM, KFRIPI, KFRIPK, KFRIPP, KFRINK.
- EIDT: QEIDEC, QEIDRI(I,N=1,4,5).
- PECO: QPECEC, KPECKD, KPEPPC.
- PEPT: all.
- PHCO: QPHCER, KPHCKD, KPHCCC, KPHCRB, KPHCPC.
- YV0V: KYV0IC.
- EGPC: QEGPR1, QEGPR2, QEGPF4, QEGPDM, KEGPST.
- EFOL: KEFOLC.
- Vertex: QVEM.

The energy flow stored on the new Mini-DST corresponds to the ENFLW algorithm; until the end of 1992, PCPA will be available also.<sup>1</sup>

---

<sup>1</sup>See Section 11.1 for information on using ENFLW. When using the Mini-DST, it is not necessary to link the JULIA and ENFLW libraries.



All of ALPHA's event topology routines work with the Mini-DST. Also, the routines described in Ch. 12 may be used. In particular, QMUIDO, QDEDXM, QPAIRF, QVDHIT, and XVDEOK will work.

## D.2 Differences between POT/DST and Mini-DST

In addition to the absence of some information, other differences may be observed when using the Mini. The Mini-DST has limited precision, so some quantities will have slightly different values from those found when reading the POT or DST. Non-trivial differences are:

- FRFT/0 is generally not available on the Mini-DST.
- For ITC tracks, the  $z_0$  error is limited to 25cm, which increases QBC2 substantially.
- If the total number of vertices is greater than 30, only the main vertices and the best  $V^0$  vertices (ordered by chi-squared) up to a total of 30 will be available when reading the Mini-DST.
- Further, the  $V^0$  quantities available are the result of swimming the tracks, and if greater precision is required, a refit must be performed. Infrequently, problems arise when the charge of fitted track (FRFT/2) is different from that of the original track used (FRFT/0), and hence the two  $V^0$  tracks appear to have the same charge.
- The chi-squared and NDF for the helix fit for charged tracks may not be the same as those found for the DST, but the chi-squared per NDF should be correct within the foreseen precision.
- The values of R2 and R3 are truncated in the range [-10.23,+10.23], but where there are indications of problems, the values are set to +1000.
- For the GAMPEK photons, only the presence of dead stories is recorded, not information for each storey. Hence DST1,2,3 of the variable KEGPQU are all 0 or all 1.

## D.3 Writing a Mini-DST

To write your own Mini-DST file from a DST (or POT), you must do the following:

1. Read POT or DST (preferable), declared with a standard FILI card.
2. Declare an output file which will contain your Mini-DST with a FILO card.
3. Invoke integer compression with COMP 'INTE'.
4. ALPHA must be informed that you wish to create a Mini-DST with the MINI card.
5. You should call QWRITE whenever you wish to output an event - according to your selection criteria. Alternatively, you can use a COPY<sup>2</sup> card, and select events with CLAS and SEVT.

---

<sup>2</sup>In this case, it is not possible to store the QMUIDO and ENFLW information on your Mini-DST.

6. If you wish to have access to ENFLW and QMUIDO information, you must ensure that the appropriate code and cards are supplied. You need an EFLJ card, and should link to JULIA, ENFLW, and ALENFLW (see footnote on p. 77 and Appendix C).

When you have written a small sample of events, it is worth **checking** that these are readable before creating many events.

To make your own Mini-DST from the standard ALEPH Mini-DST, you should refer to the ALEPH Mini-DST User's Guide.

# Appendix E

## Standard particle table

1 gamma	2 e+	3 e-	4 nu	5 mu+
6 mu-	7 pi0	8 pi+	9 pi-	10 K0l
11 K+	12 K-	13 n	14 p	15 p#
16 K0s	17 eta	18 Lam0	19 Sig+	20 Sig0
21 Sig-	22 Xi0	23 Xi-	24 Ome-	25 n#
26 Lam#0	27 Sig#-	28 Sig#0	29 Sig##+	30 Xi#0
31 Xi##+	32 Ome##+	33 tau+	34 tau-	35 D+
36 D-	37 D0	38 D#0	39 Ds+	40 Ds-
41 Lamc+	42 W+	43 W-	44 Z0	45 Deuteron
46 Tritium	47 Alpha	48 Geantino	49 GeantMino	50 GeantBino
51 GeantAbino	52 GeantPhino	53 Lamc#-	54 Higgs	55 etab
56 K0	57 K0#	58 nue	59 nue#	60 numu
61 numu#	62 nutau	63 nutau#	64 a1+	65 a1-
66 a10	67 f0_975	68 a0_980	69 etapr	70 chic0
71 chic1	72 chic2	73 Jpsi	74 psipr	75 etac
76 rho+	77 rho-	78 K**	79 K*-	80 K*0
81 K**0	82 D*0	83 D**0	84 D**	85 D*-
86 Ds**	87 Ds*-	88 rho0	89 omega0	90 Phi
91 chib0	92 chib1	93 chib2	94 Upsilon	95 Ups2s
96 etat	97 chit0	98 chit1	99 chit2	100 gluon
101 uquark	102 dquark	103 squark	104 cquark	105 bquark
106 tquark	107 uquark#	108 dquark#	109 squark#	110 cquark#
111 bquark#	112 tquark#	113 Theta	114 Thetapr	115 B-
116 B+	117 B0	118 B#0	119 Bs0	120 Bs#0
121 Bc-	122 Bc+	123 B*-	124 B**	125 B*0
126 B**0	127 Bs*0	128 Bs**0	129 Bc*-	130 Bc**
131 T0	132 T#0	133 T+	134 T-	135 Ts+
136 Ts-	137 Tc0	138 Tc#0	139 Tb+	140 Tb-
141 T*0	142 T**0	143 T**	144 T*-	145 Ts**
146 Ts*-	147 Tc*0	148 Tc**0	149 Tb**	150 Tb*-
151 Delta**	152 Delta#--	153 Delta+	154 Delta#-	155 Delta0
156 Delta#0	157 Delta-	158 Delta##+	159 Sig**	160 Sig##-
161 Sig*0	162 Sig**0	163 Sig*-	164 Sig##+	165 Xi*0
166 Xi**0	167 Xi*-	168 Xi**+	169 Sigc**	170 Sigc#--
171 Sigc+	172 Sigc#-	173 Sigc0	174 Sigc#0	175 Xic+
176 Xic#-	177 Xic0	178 Xic#0	179 Omec0	180 Omec#0
181 Xi0c+	182 Xi0c#-	183 Xi0c0	184 Xi0c#0	185 Sigc***

186 Sigc*#--	187 Sigc*+	188 Sigc*#-	189 Sigc*0	190 Sigc*#0
191 Xic*+	192 Xic*#-	193 Xic*0	194 Xic*#0	195 Omec*0
196 Omec*#0	197 Xicc*+	198 Xicc*#--	199 Xicc+	200 Xicc*#-
201 Omecc+	202 Omecc*#-	203 Xicc*++	204 Xicc*#--	205 Xicc*+
206 Xicc*#-	207 Omecc*+	208 Omecc*#-	209 Omeccc*+	210 Omeccc*#--
211 Sigb+	212 Sigb*#-	213 Sigb0	214 Sigb*#0	215 Sigb-
216 Sigb*#+	217 Xib0	218 Xib*#0	219 Xib-	220 Xib*#+
221 Omeb-	222 Omeb*#+	223 Xibc+	224 Xibc*#-	225 Xibc0
226 Xibc*#0	227 Omebc0	228 Omebc*#0	229 Omebcc+	230 Omebcc*#-
231 Xibb0	232 Xibb*#0	233 Xibb-	234 Xibb*#+	235 Omebb-
236 Omebb*#+	237 Omebbc0	238 Omebbc*#0	239 Sigt*++	240 Sigt*#--
241 Sigt+	242 Sigt*#-	243 Sigt0	244 Sigt*#0	245 Xit+
246 Xit*#-	247 Xit0	248 Xit*#0	249 Omet0	250 Omet*#0
251 Xitc*++	252 Xitc*#--	253 Xitc+	254 Xitc*#-	255 Ometc+
256 Ometc*#-	257 Ometcc*++	258 Ometcc*#--	259 Lamb0	260 Lamb*#0
261 Xi0b0	262 Xi0b*#0	263 Xi0b-	264 Xi0b*#+	265 Xi0bc+
266 Xi0bc*#-	267 Xi0bc0	268 Xi0bc*#0	269 Ome0bc0	270 Ome0bc*#0
271 Lamt+	272 Lamt*#-	273 Xi0t+	274 Xi0t*#-	275 Xi0t0
276 Xi0t*#0	277 Xi0tc0*++	278 Xi0tc0*#--	279 Xi0tc+	280 Xi0tc*#-
281 Ome0tc+	282 Ome0tc*#-	283 Sigb*++	284 Sigb*##-	285 Sigb*0
286 Sigb*#0	287 Sigb*-	288 Sigb*#+	289 Xib*0	290 Xib*#0
291 Xib*-	292 Xib*#+	293 Omeb*-	294 Omeb*#+	295 Xibc*+
296 Xibc*#-	297 Xibc*0	298 Xibc*#0	299 Omebc*0	300 Omebc*#0
301 Omebcc*+	302 Omebcc*#-	303 Sigt*++	304 Sigt*#--	305 Sigt*+
306 Sigt*#-	307 Sigt*0	308 Sigt*#0	309 Xit*+	310 Xit*#-
311 Xit*0	312 Xit*#0	313 Omet*0	314 Omet*#0	315 Xitc*++
316 Xitc*#--	317 Xit*+	318 Xit*#-	319 Ometc*+	320 Ometc*#-
321 Ometcc*++	322 Ometcc*#--	323 Xitb*+	324 Xitb*#-	325 Xitb*0
326 Xitb*#0	327 Ometb*0	328 Ometb*#0	329 Ometbc*+	330 Ometbc*#-
331 Ometbb*0	332 Ometbb*#0	333 Xitb+	334 Xitb*#-	335 Xitb0
336 Xitb*#0	337 Ometb0	338 Ometb*#0	339 Ometbc+	340 Ometbc*#-
341 Ometbb0	342 Ometbb*#0	343 Xi0tb+	344 Xi0tb*#-	345 Xi0tb0
346 Xi0tb*#0	347 Ome0tb0	348 Ome0tb*#0	349 Ome0tbc+	350 Ome0tbc*#-
351 Xibb*0	352 Xibb*#0	353 Xibb*-	354 Xibb*#+	355 Omebb*-
356 Omebb*#+	357 Omebbc*0	358 Omebbc*#0	359 Omebbb-	360 Omebbb*#+
361 beame+	362 beame-	363 Charged	364 Ecal	365 Hcal
366 CALobj	367 last_st_par			

# Index

- [c]Constant = Fortran parameter
- [f]Fortran function
- [s]Fortran subroutine
- [sf]Fortran statement function
- [v]Fortran variable/array stored in a COMMON block
  
- add** vectors: 9.2.2 on p. 54 and 10.3 on p. 68
- ALEPH file types** 4.1.1 on p. 8
- ALLR** parameter on FILO data card: 4.1.3 on p. 11
- ALPHA** initialization in QMALPH called from QMINIT: 3.2 on p. 5
- ALPHARUN** command file: Ch. 2 on p. 2 and App. C on p. 105
- angle**
  - azimuth, polar angle: 9.1 on p. 52
  - decay angle: 9.1 on p. 52
- antiparticle**
  - access to antiparticles: 7.4.4 on p. 36
  - definition on data cards: 4.12.1 on p. 18
  
- batch job** see ALPHARUN: 2 on p. 2
- beam position** see QVXNOM, etc.: 6.2 on p. 25, 8.1.4 on p. 43, and 6.3.7 on p. 28
- bending radius** of a reconstructed charged track: 8.2.1 on p. 46
- beta** see QBETA: 9.1 on p. 52
- BMACRO** standard BOS statement functions: 3.1.2 on p. 3
- book** histograms: 5.1 on p. 20
- boost** Lorentz: 9.5 on p. 63
- BOM** beam position from BOMs: 6.3.7 on p. 28
- BOS** initialization in QUIBOS: 3.5.5 on p. 7
  
- c** [c] speed of light: 6.1 on p. 25
- calorimeter objects** 7.1 on p. 31 and 7.3 on p. 34
- CARDS** file type: 4.1.1 on p. 8
- charge**
  - of an individual particle: 8.1.1 on p. 42
  - on particle table: 15.5 on p. 98
- charged tracks** 7.1 on p. 31 and 7.3 on p. 34
- CLAS** Data card for use with event directories: 4.1.4 on p. 12
- class**
  - reading class. word for EDIRs: 6.3.2 on p. 26
  - setting class. word for EDIRs: 13.3 on p. 88
  - “track” class: 7.4.1 on p. 35 and 8.1.8 on p. 45
  
- COPY** data card: 4.1.5 on p. 13

## **copy**

track vectors into other track vectors: 9.2.4 on p. 55 and 9.2.9 on p. 57

track vectors into Fortran arrays: 9.2.7 on p. 56

Fortran arrays into track vectors: 9.2.13 on p. 59

**CQDATE** [v] date at start of job: 6.5.6 on p. 30

**CQFOUT** [v] name of output file: 6.5.6 on p. 30

**CQPART** [f] particle name for a given integer code: 15.5 on p. 98

**CQTPN** [f] track's particle name: 8.1.8 on p. 45

**CQTIME** [v] time at start of job: 6.5.6 on p. 30

**CQVERS** [v] ALPHA version: 6.5.6 on p. 30

**CRAY** App. C on p. 105

**create new track**: 9.2.8 on p. 57

**cross product QVCROS**: 9.2.5 on p. 56

**DAF** file type – direct access files: 4.1.1 on p. 8

## **data**

base – opened in QMINIT: 3.2 on p. 5

cards – description 4 on p. 8

set name –conventions 4.1.1 on p. 8; examples 4.1.2 on p. 9

**daughter particles** 7.5.1 on p. 38 and 7.8 on p. 41

**DEBU** data card: 4.5 on p. 15

## **debug**

special VAX debugger features: 2 on p. 2

level – see KDEBUG: 6.5.2 on p. 29

**decay angle** 9.1 on p. 52

**dE/dx** 12.1 on p. 81, 8.2.4 on p. 47

**DHEA** bank: 6.3.6 on p. 28

**dot product** 9.1 on p. 52

**drop tracks** 9.2.6 on p. 56

**DST unpacking** 4.3 on p. 14

**D0** see QDB

**e** constant: 6.1 on p. 25

## **ECAL**

objects 7.1 on p. 31 and 7.3 on p. 34

wire energy – see QEECWI: 6.4 on p. 29

**EDIR** event directory: 4.1.4 on p. 12

**EFLW** energy flow data card: 11.2 on p. 78

**EFOL** see energy flow

**EGPC** see GAMPEC

**ENDQ** BOS data card: 4 on p. 8

## **energy**

beam energy, see QELEP: 6.2 on p. 25

for ALPHA tracks, see QE: 8.1.1 on p. 42

missing energy: 10.10 on p. 73

**energy flow** Ch. 11 on p. 77

**ENFLW** energy flow: 11.1 on p. 77

**EPIO** file type: machine-independent input / output: 4.1.1 on p. 8

**EVEH** bank: 6.3.1 on p. 26

## event

- directories: 4.1.4 on p. 12
- input – see FILI data card: 4.1.2 on p. 9
- output– see FILO data card 4.1.3 on p. 11 and routine QWRITE: 13.2 on p. 88
- processing – see QUEVNT: 3.3 on p. 5

**FIEL** Data card to set magnetic field: 4.8 on p. 16

**file types** = ALEPH file types: 4.1.1 on p. 8

**FILI** data card – input data set(s): 4.1.2 on p. 9

**FILO** data card – output data set: 4.1.3 on p. 11

**flags** user: 8.1.8 on p. 45, 9.2.14 on p. 60

**Fox-Wolfram** moments: 10.8 on p. 71

**frame** access to Lorentz frames: 7.4.1 on p. 35

**FRF0** data card – ignore vertex det. in track fit: 4.9 on p. 17

**gamma** see QGAMMA: 9.1 on p. 52

**gamma conversions** see QPAIRF: 12.2 on p. 83

**GAMPEC** photons: 7.1.1 on p. 32 and 8.2.13 on p. 50

**h** constant 6.1 on p. 25

**HAC parameters** bank offset: 3.1.3 on p. 5

**hbar** constant: 6.1 on p. 25

**HBOOK** 5.1 on p. 20

- initialization – QUIHIS: 3.5.3 on p. 7

- termination – QUTHIS: 3.5.4 on p. 7

**HCAL objects** 7.1 on p. 31 and 7.3 on p. 34

**hemispheres** see QJHEMI: 10.9 on p. 72

**High Voltage** 6.3.4 on p. 27

**HIS** histogram file type 5.2.1 on p. 23

**HIST** histogram file data card: 5.2.1 on p. 23

**histograms** Ch. 5 on p. 20

**histogram output** see 5.2.1 on p. 23 and 5.2.2 on p. 24

**Historian** Ch. 2 on p. 2

**HTIT** data card: general histogram title 4.7.2 on p. 16

**IBM App. C** on p. 105

**Implicit None** 3.1.4 on p. 5

**INCLUDE** Fortran statement: 3.1.2 on p. 3

**initialization** see QMINIT and QUINIT: 3.2 on p. 5

**invariant mass** 9.1 on p. 52

**IRUN** data card: ignore runs 4.1.2 on p. 10

**jets** 10.11 on p. 73

**KBFLAG** [sf] track flag bits

**KBMASK** [sf] track mask bits

**KCANTI** [sf] particle → antiparticle: 15.5 on p. 98

**KCDIR** [sf] direct access to particles: 7.4.2 on p. 35

**KCDIRA** [sf] direct access to (anti)particles: 7.4.4 on p. 36

**KCHGD** [sf] list of associated charged tracks: 7.3 on p. 34

**KCLASS** [sf] class KRECO,KMONTE,Lorentz fr.: 8.1.8 on p. 45  
**KCLASW** [v] event directory class. word: 6.3.2 on p. 26  
**KCH** [sf] track's charge: 8.1.1 on p. 42  
**KCHT** [f] original copy of a charged track: 7.6.2 on p. 40  
  
**KDAU** [sf] access to daughter particles: 7.5.1 on p. 38  
**KDEBUG** [v] debug level: 6.5.2 on p. 29  
**KDHExx** [v] event header bank DHEA: 6.3.6 on p. 28  
**KECAL** [sf] list of associated ECAL objects: 7.3 on p. 34  
**KEIDxx** [sf] bank EIDT: 8.2.5 on p. 47  
**KENDV** [sf] track's end vertex: 7.8 on p. 41  
**KEVExx** [v] event header bank EVEH: 6.3.1 on p. 26  
**KEVH** bank: 6.3.3 on p. 27  
**KEVT** [v] current event number: 6.3.1 on p. 26  
**KEXP** [v] experiment number: 6.3.1 on p. 26  
**KFAST** [v] first cal object associated to a charged track: 7.1 on p. 31  
**KFCHT** [v] first charged track: 7.1 on p. 31  
**KFCOT** [v] first cal object: 7.1 on p. 31  
**KFDCT** [v] first decay track: 7.1 on p. 31  
**KFIST** [v] first isolated cal object: 7.1 on p. 31  
**KFJET** [v] first reconstructed jet: 7.1 on p. 31  
**KFMCT** [v] first MC particle: 7.1 on p. 31  
**KFOLLO** [sf] following track: 7.4.2 on p. 35  
**KFOVT** [v] first overlap object: 7.1 on p. 31  
**KRDFL** [sf] read user flag: 8.1.8 on p. 45  
**KFRET** [v] first reconstructed track: 7.1 on p. 31  
**KFRFxx** [sf] bank FRFT = track fit: 8.2.1 on p. 46  
**KFRIdx** [sf] bank FRID: 8.2.3 on p. 46  
**KFRTxx** [sf] bank FRTL: 8.2.2 on p. 46  
**KFV0T** [v] first particle pointing to V0: 7.1 on p. 31  
**KHCAL** [sf] list of associated HCAL objects: 7.3 on p. 34  
**KHMAxx** [sf] bank HMAD = HCAL–muon association: 8.2.6 on p. 48  
**Kinematic fitting** 9.3 on p. 61  
**KKEVxx** [v] bank KEVH 6.3.3 on p. 27  
**KLAST** [v] last cal object associated to a charged track: 7.1 on p. 31  
**KLCHT** [v] last charged track: 7.1 on p. 31  
**KLCOT** [v] last cal object: 7.1 on p. 31  
**KLDCT** [v] last decay track: 7.1 on p. 31  
**KLIST** [v] last isolated cal object: 7.1 on p. 31  
**KLJET** [v] last reconstructed jet: 7.1 on p. 31  
**KLMCT** [v] last MC particle: 7.1 on p. 31  
**KLOVT** [v] last overlap object: 7.1 on p. 31  
**KLRET** [v] last reconstructed track: 7.1 on p. 31  
**KLUNDS** [sf] Lund status code: 8.1.8 on p. 45  
**KLV0T** [v] last particle pointing to V0: 7.1 on p. 31  
**KMCAxx** [sf] bank MCAD = muon chambers: 8.2.7 on p. 48  
**KMOTH** [sf] access to mother particle: 7.5.2 on p. 39  
**KMTCH** [sf] match MC – reco. tracks: 7.7 on p. 40



**KNAST** [v] number of cal objects assoc. to a charged track: 7.1 on p. 31  
**KNCHGD** [sf] number of associated charged tracks: 7.3 on p. 34  
**KNCHT** [v] number of charged tracks: 7.1 on p. 31  
**KNCOT** [v] number of cal objects: 7.1 on p. 31  
**KNDAU** [sf] number of daughters: 7.5.1 on p. 38  
**KNDCT** [v] number of decay tracks: 7.1 on p. 31  
**KNECAL** [sf] number of associated ECAL objects: 7.3 on p. 34  
**KNEFIL** [v] number of events on current input file: 6.5.1 on p. 29  
**KNEOUT** [v] number of events on output file: 6.5.1 on p. 29  
**KNEVT** [v] number of events read in up to now: 6.5.1 on p. 29  
**KNHCAL** [sf] number of associated HCAL objects: 7.3 on p. 34  
**KNIST** [v] number of isolated cal objects: 7.1 on p. 31  
**KNJET** [v] number of reconstructed jets: 7.1 on p. 31  
**KNMCT** [v] number of MC particles: 7.1 on p. 31  
**KNMOTH** [sf] number of mother particles: 7.5.2 on p. 39  
**KNMTCH** [sf] number of matching particles: 7.7 on p. 40  
**KNOVT** [v] number of overlap objects: 7.1 on p. 31  
**KNREIN** [v] number of records read from current input file: 6.5.1 on p. 29  
**KNRET** [v] number of reconstructed tracks: 7.1 on p. 31  
**KNTEX** [sf] number of TPC sectors for dE/dx: 8.2.4 on p. 47  
**KNV0T** [v] number of particle pointing to V0s: 7.1 on p. 31  
**KORIV** [sf] vertex at origin of track: 7.8 on p. 41  
**KPART** [f] integer code from particle name: 15.5 on p. 98 and 7.4.3 on p. 36  
**KPDIR** [f] direct access to particles: 7.4.2 on p. 35  
**KPDIRA** [f] direct access to (anti)particles: 7.4.4 on p. 36  
**KPECxx** [sf] bank PECO: 8.2.9 on p. 49  
**KPEPxx** [sf] bank PEPT: 8.2.10 on p. 49  
**KPHCxx** [sf] bank PHCO: 8.2.11 on p. 49  
**KRUN** [v] run number: 6.3.1 on p. 26  
**KSAME** [sf] access to same objects: 7.6 on p. 39  
**KSMTCH** [sf] number of shared hits in match: 7.7 on p. 40  
**KSTABC** [sf] stability code: 8.1.5 on p. 44  
**KSTATU** [v] job status (init / event proc. / term): 6.5.2 on p. 29  
**KTEXxx** [sf] dE/dx bank TEXS: 8.2.4 on p. 47  
**KTJOR** [f] Lorentz transformation: 9.5.1 on p. 63  
**KTJOR1** [f] Lorentz transformation: 9.5.2 on p. 63  
**KTN** [sf] Julia/Galeph track number: 8.1.8 on p. 45  
**KTPCOD** [sf] track's particle code: 8.1.8 on p. 45  
**KUCARD** [v] log. unit for the card file: 6.5.4 on p. 29  
**KUCRD2** [v] 2nd log. unit for card files: 6.5.4 on p. 29  
**KUCONS** [v] log. unit for the data base: 6.5.4 on p. 29  
**KUINPU** [v] log. unit for event input: 6.5.4 on p. 29  
**KUOUTP** [v] log. unit for event output: 6.5.4 on p. 29  
**KUPRNT** [v] log. unit for the line printer output: 6.5.4 on p. 29 and 6.5.4 on p. 29  
**KUPTER** [v] log. unit for the terminal: 6.5.4 on p. 29 and 6.5.4 on p. 29  
**KVBFLG** [sf] vertex bit flags  
**KVDAU** [sf] access to tracks from a vertex: 7.8 on p. 41  
**KVFITM** [f] kinematic fitting: 9.3 on p. 61

**KVGOOD** [f] VDET readout: 12.4.3 on p. 87  
**KVINCP** [sf] incoming particle to a vertex: 7.8 on p. 41  
**KVN** [sf] Julia/Galeph vertex number: 8.3 on p. 51  
**KVNDAU** [sf] number of outgoing tracks: 7.8 on p. 41  
**KVNEW** [f] create new track vector: 9.2.8 on p. 57  
**KVSAVE** [f] save track: 9.2.9 on p. 57  
**KVSAVC** [f] save track in specific class: 9.2.12 on p. 58  
**KYV0xx** [sf] bank YV0V: 8.2.12 on p. 50

**lifetime** on particle table: see QCLIFE / QPLIFE: 15.5 on p. 98  
**line printer** see KUPRNT  
**lock** 10.2 on p. 66  
**logical units** 6.5.4 on p. 29  
**loops** over tracks (= vectors) and vertices: 7 on p. 31  
**Lorentz** transformations: 9.5 on p. 63; see also decay angle: 9.1 on p. 52  
**LUCLUS** jet finding algorithm: 10.11.3 on p. 75

**main program** see QMAIN  
**mass**  
 of an individual particle: 8.1.1 on p. 42  
 invariant mass of a system of particles: 9.1 on p. 52  
 missing mass: 10.10 on p. 73  
 nominal mass in the particle table: 15.5 on p. 98  
**match** reconstructed tracks and MC particles: 7.7 on p. 40  
**Mini-DST** App. D on p. 109  
**MINI**  
 card: 4.1.3 on p. 11 and App. D on p. 109  
 flag for Mini-DST input: 6.5.3 on p. 29  
**missing** mass, energy, momentum: 10.10 on p. 73  
**momentum** of a particle see QP 8.1.1 on p. 42 / missing momentum 10.10 on p. 73  
**Monte Carlo**  
 flag for an event: 6.5.3 on p. 29  
 loops over MC particles: 7.1 on p. 31 and 7.4 on p. 35  
 particle code: 15.1 on p. 95  
 particle table: 15.1 on p. 95  
**mother particle** 7.5.2 on p. 39  
**MUID** access to MUID (QMUIDO) information: 8.2.8 on p. 48

**NATIVE** file type: machine-dependent input/output 4.1.1 on p. 8  
**NATIVE** parameter on FILI / FILO data cards (q.v.)  
**NEVT** data card: select NEVT events: 4.1.2 on p. 10  
**new track** KVNEW: 9.2.8 on p. 57  
**nominal mass** on particle table: 15.5 on p. 98  
**NOOV** parameter on FILO / HIST data cards (q.v.)  
**NOPH** no histogram printing: 5.2.2 on p. 24  
**NORU** parameter on FILO data card: 4.1.3 on p. 11  
**NOxx** ALPHA process cards: 4.2 on p. 14  
**Ntuples** Ch. 5 on p. 20

## output

events – see FILO card: 4.1.3 on p. 11 and routine QWRITE: 13.2 on p. 88  
histograms – see HIST data card: 5.2.1 on p. 23

**parameters** HAC parameters: 3.1.3 on p. 5

## particle

analysis of particle systems: 7.4.5 on p. 37  
–antiparticle relation: 7.4.4 on p. 36  
attributes: 15.5 on p. 98  
code: 7.4.3 on p. 36 and 15.2 on p. 95  
direct access to specific particles: 7.4 on p. 35  
invariant mass of particle systems: 9.1 on p. 52  
table  
  data cards: 15.4 on p. 96  
  MC table: 15.1 on p. 95 and 15.4 on p. 96  
  standard table 15.4 on p. 96

**PAW** interactive analysis of histograms and Ntuples: 5.2.1 on p. 23

**PCPA** neutral objects from PCPA: 7.1.1 on p. 32 and 11.3 on p. 79

**photon conversions** see QPAIRF: 12.2 on p. 83

**photons** from GAMPEC: 7.1.1 on p. 32 and 8.2.13 on p. 50

**pi** constant: 6.5.6 on p. 30

**Planck** constant: 6.5.6 on p. 30

**PMOD** data card: modify particle table 15.4.1 on p. 96

**PNEW** data card: new entry into particle table 15.4.2 on p. 97

**PTRA** data card: modify MC particle code translation: 15.4.3 on p. 97

**POT** unpacking: 4.3 on p. 14

**process** ALPHA process cards: 4.2 on p. 14

**PTCLUS** jet finding algorithm: 10.11.4 on p. 76

**QBETA** [sf] beta of a particle: 9.1 on p. 52

**QBOOKN** [s] book Ntuples: 5.1.3 on p. 21

**QBOOKR** [s] book Ntuples with run and event number: 5.1.4 on p. 22

**QBOOK1** [s] book 1–dimensional histograms: 5.1.1 on p. 20

**QBOOK2** [s] book 2–dimensional histograms: 5.1.2 on p. 21

**QCDE** macro: all parameters, commons etc.: 3.1.2 on p. 3

**QCDESH** short subset of QCDE

**QCFxxx** macros containing statement functions

**QCH** [sf] track's charge: 8.1.1 on p. 42

**QCOSA** [sf] cos (angle between two tracks): 9.1 on p. 52

**QCT** [sf] cos (theta): 9.1 on p. 52

**QDATA** [s] (quasi) block data

**QDB** [sf] distance to beam axis: 8.1.4 on p. 43

**QDBS2** [sf] error<sup>2</sup> on QDBS2: 8.1.4 on p. 43

**QDECAN** [f] decay angle: 9.1 on p. 52

**QDECA2** [f] decay angle: 9.1 on p. 52

**QDEX** [s] dE/dx analysis: 12.1 on p. 81

**QDEXM** [s] dE/dx analysis: 12.1.2 on p. 82

**QDHExx** [v] header bank DHEA: 6.3.6 on p. 28

**QDMSQ** [sf] mass difference <sup>2</sup>: 9.1 on p. 52

**QDOT3** [sf] dot product (3–vector): 9.1 on p. 52  
**QDOT4** [sf] dot product (4–vector): 9.1 on p. 52  
**QE** [sf] energy: 8.1.1 on p. 42  
**QEECWI** [v] ECAL wire energy: 6.4 on p. 29  
**QEIDxx** [sf] bank EIDT = electron identification: 8.2.5 on p. 47.  
**QLEP** [v] LEP energy: 6.3.1 on p. 26  
**QFRFxx** [sf] bank FRFT = track fit: 8.2.1 on p. 46  
**QFRIdx** [sf] bank FRID: 8.2.3 on p. 46  
**QFRTxx** [sf] bank FRTL = appendix to FRFT: 8.2.2 on p. 46  
**QGAMMA** [sf] particle’s gamma: 9.1 on p. 52  
**QHFN** [s] fill Ntuple: 5.1.6 on p. 22  
**QHFNR** [s] fill Ntuple with run and event number: 5.1.6 on p. 22  
**QHFR** [s] fill Ntuple with run and event number: 5.1.5 on p. 22  
**QHMAxx** [sf] bank HMAD = HCAL–muon association: 8.2.6 on p. 48  
**QIDV0** [s]Recalculate V0 4–vector: 9.2.3 on p. 55  
**QJADDP** [s]add 4–vectors: 10.3 on p. 68  
**QJEIG** [s]eigenvalues of mom. tensor: 10.4 on p. 70  
**QJFOXW** [s]Fox–Wolfram moments: 10.8 on p. 71  
**QJHEMI** [s]divide the event into two hemispheres: 10.9 on p. 72  
**QJMISS** [s]missing energy, mass, and momentum: 10.10 on p. 73  
**QJMDCL** [s]jet finding – scaled minimum distance algorithm: 10.11.2 on p. 75  
**QJMMCL** [s]jet finding – scaled invariant mass sq. algorithm: 10.11.1 on p. 73  
**QJLUCL** [s]jet finding – LUCLUS: 10.11.3 on p. 75  
**QJOPTM** [s]select MC particles for QJxxxx routines: 10.1.2 on p. 66  
**QJOPTR** [s]select reconstructed objects for QJxxxx routines: 10.1.1 on p. 65  
**QJPTCL** [s]jet finding – PTCLUS: 10.11.4 on p. 76  
**QJSPHE** [s]sphericity: 10.6 on p. 70  
**QJTENS** [s]linearized momentum tensor: 10.5 on p. 70  
**QJTHRU** [s]thrust value / axis: 10.7 on p. 71  
**QKEVxx** [v] bank KEVH: 6.3.3 on p. 27  
**QLTRK** [s] lock individual track: 10.2.1 on p. 67  
**QLOCK** [s] lock track family: 10.2.3 on p. 67  
**QLOCK2** [s] lock track family: 10.2.6 on p. 68  
**QLREV** [s] reverse lock: 10.2.5 on p. 68  
**QLREV2** [s] reverse lock: 10.2.6 on p. 68  
**QLUTRK** [s] unlock individual track: 10.2.2 on p. 67  
**QLZER** [s] zero lock: 10.2.4 on p. 68  
**QLZER2** [s] zero lock: 10.2.6 on p. 68  
**QM** [sf] particle’s mass: 8.1.1 on p. 42  
**QMACRO** macro: statement functions: 3.1.2 on p. 3  
**QMAIN ALPHA** main program: App. A on p. 99  
**QMASV0** [f]V0 mass: 8.1.2 on p. 42; see also QIDV0.  
**QMCAXx** [sf] bank MCAD = muon chambers: 8.2.7 on p. 48  
**QMCHI2** [f]  $\chi^2$  from mass difference: 9.1 on p. 52  
**QMDIFF** [f] mass difference: 9.1 on p. 52  
**QMFLD** [v] ALEPH magnetic field: 6.3.1 on p. 26  
**QMINIT** [s] system initialization: 3.2 on p. 5  
**QMUIDO** [s] muon identification: 12.3 on p. 84 and 8.2.8 on p. 48

**QMSQ2,QMSQ3,QMSQ4** [sf] invariant mass<sup>2</sup>: 9.1 on p. 52  
**QMTERM** [s] system termination: 3.4 on p. 6 and 13.1 on p. 88  
**QM2,QM3,QM4** [sf] invariant mass: 9.1 on p. 52  
**QNTEX** [sf] number of sectors for dE/dx: 8.2.4 on p. 47  
**QP** [sf] momentum: 8.1.1 on p. 42  
**QPAIRF** [s] photon conversions: 12.2 on p. 83  
**QPCHAR** [f] particle table charge: 15.5 on p. 98  
**QPECxx** [sf] bank PECO: 8.2.9 on p. 49  
**QPEPxx** [sf] bank PEPT: 8.2.10 on p. 49  
**QPHCxx** [sf] bank PHCO: 8.2.11 on p. 49  
**QPH** [sf] track's azimuth: 9.1 on p. 52  
**QPLIFE** [f] particle table life time: 15.5 on p. 98  
**QPMASS** [f] particle table mass: 15.5 on p. 98  
**QPPAR** [sf] momentum parallel to a vector: 9.1 on p. 52  
**QPPER** [sf] momentum perpendicular to a vector: 9.1 on p. 52  
**QPT** [sf] transverse momentum: 9.1 on p. 52  
**QPWIDT** [f] particle table width: 15.5 on p. 98  
**QQC** [c] speed of light: 6.5.6 on p. 30  
**QQE** [c] e: 6.5.6 on p. 30  
**QQH** [c] hbar: 6.5.6 on p. 30  
**QQHC** [c] hbar \* c 6.5.6 on p. 30  
**QQIRP** [c] factor between inv. bending radius and momentum: 6.5.6 on p. 30  
**QQPI** [c]  $\pi$ : 6.5.6 on p. 30  
**QQPIH** [c]  $\pi / 2$ : 6.5.6 on p. 30  
**QQRADP** [c]  $360 / \pi$ : 6.5.6 on p. 30  
**QQ2PI** [c]  $2 \pi$ : 6.5.6 on p. 30  
**QRDFL** [sf] read user flag: 8.1.8 on p. 45  
**QSIGxx** [sf] track's error matrix: 8.1.3 on p. 43  
**QSTFLI** [s] set user flag (integer): 9.2.14 on p. 60  
**QSTFLR** [s] set user flag (real): 9.2.14 on p. 60  
**QSUSTR** [s] allocate user's track space 14.1.1 on p. 93  
**QSUSVX** [s] allocate user's vertex space 14.1.2 on p. 94  
**QTCLAS** [s] Lorentz transformation: 9.5.3 on p. 64  
**QTEXxx** [sf] bank TEXS = dE/dx: 8.2.4 on p. 47  
**QTIME** [v] as given on the TIME data card: 6.5.5 on p. 30  
**QTIMEL** [v] remaining job time: 6.5.5 on p. 30  
**QUEVNT** [s] event processing user routine: 3.3 on p. 5  
**QUIBOS** [s] initialize BOS: 3.5.5 on p. 7  
**QUIHIS** [s] initialize histograms: 3.5.3 on p. 7  
**QUINIT** [s] user initialization routine: 3.2 on p. 5  
**QUNEW** [s] user routine: called for every new run: 3.5.1 on p. 6  
**QUSREC** [s] special records: 3.5.2 on p. 7  
**QUTERM** [s] user termination routine: 3.4 on p. 6  
**QUTHIS** [s] terminate histograms: 3.5.4 on p. 7  
**QVADD2, QVADD3, QVADD4, QVADDN** [s] add track vectors: 9.2.2 on p. 54  
**QVCOPY** [s] copy track vectors: 9.2.4 on p. 55  
**QVCROS** [s] cross product: 9.2.5 on p. 56  
**QVDHIT** [s] VDET hits: 12.4.1 on p. 86

**QVDROP** [s] drop tracks: 9.2.6 on p. 56  
**QVEM** [sf] vertex error matrix: 8.3 on p. 51  
**QVGETS** [s] copy error matrix into Fortran array: 9.2.7 on p. 56  
**QVGET3,QVGET4** [s] copy track vector into Fortran array: 9.2.7 on p. 56  
**QVSCAL** [s] scale track momentum: 9.2.13 on p. 59  
**QVSETM** [s] set mass of a track: 9.2.13 on p. 59  
**QVSETS** [s] copy Fortran array into error matrix: 9.2.13 on p. 59  
**QVSET3,QVSET4** [s] copy Fortran array into track vector: 9.2.13 on p. 59  
**QVSUB** [s] subtract track vectors: 9.2.15 on p. 60  
**QVX,QVY,QVZ** [sf] vertex position: 8.3 on p. 51  
**QVZERO** [s] zero track vector: 9.2.16 on p. 61  
**QWCLAS** [s] set classification word for EDIRs: 13.3 on p. 88  
**QWEVNT** [s] print whole event: 13.5.3 on p. 90  
**QWHEAD** [s] print event header: 13.5.4 on p. 91  
**QWHFUL** [s] print full event header: 13.5.5 on p. 91  
**QWITK** [s] print individual track(s): 13.5.6 on p. 91  
**QWIVX** [s] print individual vertices: 13.5.7 on p. 92  
**QWMESS** [s] message routine: 13.5.1 on p. 90  
**QWMESE** [s] message routine: 13.5.2 on p. 90  
**QWRITE** [s] event output routine: 13.2 on p. 88  
**QWSEC** [s] print section of tracks/vertices: 13.5.8 on p. 92  
**QWTIME** [s] print time consumption: 13.4.1 on p. 89  
**QWTREE** [s] print decay chain tree: 13.5.9 on p. 92  
**QX** [sf] x-momentum: 8.1.1 on p. 42  
**QY** [sf] y-momentum: 8.1.1 on p. 42  
**QZ** [sf] z-momentum: 8.1.1 on p. 42  
**QZB** [sf] z-distance to interaction point: 8.1.4 on p. 43  
**QZBS2** [sf] error<sup>2</sup> on QZB: 8.1.4 on p. 43  
**QYV0xx** [sf] bank YV0V: 8.2.12 on p. 50

**READ** data card; read cards from several card files: 4.4 on p. 15

**run**

change: 3.5.1 on p. 6  
 information: 6.2 on p. 25  
 selection: 4.1.2 on p. 10

**same**

objects in diff. Lorentz frames, with diff. hypotheses: 7.6 on p. 39  
 two particles based on the same object – see XSAME: 8.1.7 on p. 45

**save tracks** KVSAVE; KSAVC: 9.2.12 on p. 58 and 9.2.9 on p. 57

**scale momentum** QVSCAL: 9.2.13 on p. 59

**SCANBOOK** creating FILI cards: 4.1.2 on p. 9

**selection** see run/event selection: 4.1.2 on p. 10

**SELR** parameter on FILO data card 4.1.3 on p. 11

**set mass** QVSETM: 9.2.13 on p. 59

**SEVT** data card: select events 4.1.2 on p. 10

**slow control** read s.c. records: 3.5.2 on p. 7

**speed of light** constant: 6.5.6 on p. 30

**sphericity** 10.6 on p. 70, 10.4 on p. 70

**SRUN** data card: select runs 4.1.2 on p. 10  
**start ALPHA** interactively or in batch: Ch. 2 on p. 2  
**STOP** Fortran statement: forbidden: 3.4 on p. 6  
**submit a job** Ch. 2 on p. 2  
**subtract** track vectors: 9.2.15 on p. 60  
**SYNT** data card: indicates a syntax check run 4.13 on p. 19

**tapes** 4.1.2 on p. 9

**terminal output** see KUPTER

**thrust** 10.7 on p. 71

**TIME** data card: time to terminate the job properly 4.6 on p. 15

**time** remaining job time: see QTIMEL 6.5.5 on p. 30

**timing** time consumption: 13.4 on p. 89

**title** general title for HBOOK histograms: 5.2.3 on p. 24

**topology** routines: Ch. 10 on p. 65

**track**

class: 7.4.1 on p. 35

track number: Ch. 7 on p. 31

**trigger information** 6.3.5 on p. 27

**UNIX** App. C on p. 105

**unpack** POT/DST unpacking: 4.3 on p. 14

**UNPK** data card: control POT/DST unpacking

**UPDA** parameter on HIST data card 5.2.1 on p. 23

**unit** log. input / output units: 6.5.4 on p. 29

**units** ALEPH phys. unit system: 6 on p. 25

**user routines** Ch. 3 on p. 3

**user track / vertex sections** 9.2.8 on p. 57

**VDET**

utility routines: 12.4 on p. 86

tracks not using VDET: 4.9 on p. 17

**vector** synonym for “track” or “particle”

class: 7.4.1 on p. 35 and 8.1.8 on p. 45

number – see track or vertex number

operations: 9.2 on p. 53

**vertex number** Ch. 7 on p. 31

**V0 mass** 8.1.2 on p. 42 and 9.2.3 on p. 55

**VAX** App. C on p. 105

**width** see particle table: 15.5 on p. 98

**write**

events – see FILO data card: 4.1.3 on p. 11 and QWRITE: 13.2 on p. 88

on line printer, terminal: Ch. 2 on p. 2

**XCEQAN, XCEQOR, XCEQU** [sf] check particle name: 8.1.6 on p. 44

**XEID** [sf] does EIDT exist ? 8.2.5 on p. 47

**XFRF** [sf] do FRFT and FRTL exist ? 8.2.1 on p. 46

**XHMA** [sf] does HMAD exist ? 8.2.6 on p. 48

**XHVTRG** [v] detector HV and trigger status: 6.3.4 on p. 27  
**XLOCK** [sf] track locked? 10.2 on p. 66  
**XLUMOK** see XHVTRG: 6.3.4 on p. 27  
**XMC** [sf] MC particle? 8.1.8 on p. 45  
**XMCA** [sf] does MCAD exist? 8.2.7 on p. 48  
**XMCEV** [v] MC event? 6.5.3 on p. 29  
**XMINI** [v] event input MINI? 6.5.3 on p. 29  
**XPEQAN,XPEQOR,XPEQU** [f] test particle name: 8.1.6 on p. 44  
**XSAME** [sf] tracks based on the same object? 8.1.7 on p. 45  
**XTEX** [sf] does TEXS exist? 8.2.4 on p. 47  
**XVITC,XVTPC, etc.** [v] detector HV status: 6.3.4 on p. 27  
**XVDEOK** [f] VDET HV: 12.4.2 on p. 86

**YCUT** see QJMMCL

**zero** track vectors: 9.2.16 on p. 61  
**Z0** see QZB