ALPHA User’s Guide

Authors:
H. Albrecht/DESY, CERN
E. Blucher/CERN

With contributions from:
B. Bloch – Deveaux/Saclay
J. Boucrot/LAL – Orsay
D. Cinabro/Wisconsin
J. Hilgart/Wisconsin
E. Lancon/Saclay
M. N. Minard/LAPP
M. Pepe/CERN
P. Perez/Saclay
D. Schlatter/CERN
M. Talby/LAPP
T. Wildish/RHC

ALPHA – ALEPH PHYSICS ANALYSIS PACKAGE

VERSION 106

September 19, 1989
Contents

Chapter 1: Introduction .............................................. 1

Chapter 2: Getting Started ........................................ 2

Chapter 3: User routines ............................................ 3
  3.1 General Comments ............................................. 3
  3.2 User Initialization ........................................... 4
  3.3 Event analysis routine ....................................... 5
  3.4 User termination routine .................................... 5
  3.5 Other User Subroutines ...................................... 5

Chapter 4: Data Cards ............................................... 8
  4.1 Input/Output ................................................ 8
  4.2 Histograms ................................................ 12
  4.3 READ: Input from different card files .................. 13
  4.4 DEBU: Debug output ....................................... 13
  4.5 UNPK: POT / DST unpacking ............................... 13
  4.6 Process cards ............................................. 13
  4.7 FIEL: Magnetic field ...................................... 14
  4.8 TIME: Job time control .................................. 14
  4.9 Particle table ............................................. 14
  4.10 Weight factors for calorimeters ......................... 15
  4.11 EFLW: Energy Flow ...................................... 15
  4.12 SYNT: Syntax Checker .................................... 15

Chapter 5: Creating Histograms and Ntuples .................... 16
  5.1 Booking and Filling Histograms/Ntuples .................. 16
  5.2 Histogram output — the ALPHA cards file .............. 18

Chapter 6: Mnemonic symbols ..................................... 20
  6.1 Mathematical and physical constants ..................... 20
  6.2 Event / run header ....................................... 20
  6.3 ALPHA Internal Constants, Variables .................... 21

Chapter 7: ALPHA "Tracks" and "Vertices" ....................... 23
  7.1 Access by Fortran DO loops ............................... 23
  7.2 Loops over Ecal and Hcal objects ....................... 25
  7.3 Relationships between objects in different subdetectors 25
  7.4 Direct access to particles ............................... 26
7.5 Mother – daughter relationships ........................................... 29
7.6 Access to the "same" object .................................................. 30
7.7 Match reconstructed tracks and MC truth .............................. 31
7.8 Track – vertex relationships ............................................... 31

Chapter 8:  ALPHA Track and Vertex Attributes .............................. 32

8.1 "Track" attributes. ............................................................. 32
8.2 Track related detector data .............................................. 34
8.3 Vertex attributes ............................................................. 37

Chapter 9:  Kinematics and Track Operations ............................... 38

9.1 Scalar quantities ............................................................. 38
9.2 Vector quantities ............................................................ 38
9.3 Lorentz transformations ................................................ 46

Chapter 10:  Event Topology Routines ......................................... 48

10.1 Options for "QJ" routines .................................................. 48
10.2 Lock tracks / subsamples of tracks .................................... 49
10.3 Add momenta of all particles of a given class ....................... 51
10.4 Momentum tensor eigenvalues and eigenvectors .................. 53
10.5 Linearized momentum tensor eigenvalues and eigenvectors ...... 53
10.6 Sphericity .................................................................. 53
10.7 Thrust ........................................................................ 54
10.8 Fox–Wolfram Moments .................................................. 54
10.9 Divide event into two hemispheres .................................... 55
10.10 Missing energy, mass, momentum ...................................... 56
10.11 Jet Finding .................................................................. 56

Chapter 11:  Energy Flow .......................................................... 59

11.1 EFLW: Select Energy Flow Option ...................................... 59
11.2 Using Event Topology Routines with Energy Flow ............... 60
11.3 Examples .................................................................. 60

Chapter 12:  Other ALPHA Physics Routines ................................. 62

12.1 dE/dx Analysis .............................................................. 62
12.2 Photon conversions ....................................................... 62

Chapter 13:  ALPHA Utility Routines: Printing, Writing Events, Etc. .......................... 64

13.1 Program termination ...................................................... 64
13.2 Write the current event on the output file. ........................... 64
13.3 Set classification word written to event directory ............... 64
13.4 Print routines .............................................................. 65
13.5 Print time consumption ........................................ 65

Chapter 14: Modifying ALPHA banks .......................... 69
14.1 User track / vertex sections ............................... 69
14.2 Modifying track / vertex attributes ....................... 70

Chapter 15: Particle Table ....................................... 71
15.1 Description .................................................. 71
15.2 Particle name, particle code .............................. 71
15.3 How to spell particle names ................................ 71
15.4 Data Cards for Particle Table .............................. 72
15.5 Access to particle properties ............................... 73

Appendix A: Program Structure ................................. 74

Appendix B: Bank description .................................. 75

Appendix C: Where to find ALPHA at CERN ................. 79
C.1 ALPHA on CERNVM ........................................ 79
C.2 ALPHA on VXCERN, ALWS ................................. 80
C.3 ALPHA on the CRAY ....................................... 80

Appendix D: Standard particle table ............................ 82

Appendix E: Index ............................................... 84
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 MDST). 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 (tablular 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 on page 74) is extremely simple. Three Fortran routines have to be supplied by the user: job initialization, event processing, and job termination (see Ch. 3 on page 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 (ch. 7 on page 23).

This document describes all features of the ALPHA program. For first-time users, the important parts to read are Ch. 2 on page 2 (getting started), Ch. 3 on page 3 (user routines), Ch. 4 on page 8 (event input), Ch. 7 on page 23 (loops over tracks), and Ch. 8 on page 32 (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 on page 3).
2. A card file which contains names of input/output data files, as well as other parameters (see ch. 4 on page 8).

The libraries needed to link the program are described in Appendix C on page 79. To run ALPHA, the following files must be assigned:

\textit{unit}

7 Card file
6 Print output 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.

A command file (ALPHARUN) is available on VAX and IBM which makes these file assignments, and also performs the following tasks:

1. compiles and links the Fortran (Historian) code;
2. runs the program interactively or submits a BATCH job.

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

\textit{Examples}

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

\textbf{EVALUATE LMHLEN} LMHLEN is a parameter and NOT a variable; use EVA instead of EXA.

\textbf{QP(ITK)} Debugger commands are defined for almost (!) all mnemonic symbols which have one or more arguments (see ch. 6 on page 20). 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.
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 on page 79). 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 source file; the ALPHARUN command file described in Chapter 2 on page 2 supports both options. For all user routines, default versions exist on the ALPHA library which are loaded automatically if no Fortran 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 state-
ments or by Historian statements, as shown below.

VAX / VXCERN

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

IBM / CERNVM
INCLUDE 'QCDE INC *
INCLUDE 'QMACRO INC *

CRAY

INCLUDE 'QCDE.INC'
INCLUDE '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, DATAAs, etc.
3. QMACRO, your own statement function definitions (if any)
4. your executable Fortran statements

If you do not follow these rules, the Fortran compiler will react with a series of exceptionally unintelligible error messages.

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 on page 20), this offset is taken into account automatically, and the corresponding HAC parameters are available in QCDE (note that names of HAC parameters and of mnemonic symbols are closely related).

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

3.2 User Initialization

SUBROUTINE QUINIT

This routine should be used to book histograms and reserve space for user tracks (see 14.1 on page 69), as well as any other user initialization.

All standard initialization work is performed automatically in the ALPHA subroutine QMINIT before QUINIT is called (note the difference between "U" ( = "User") and "M"). The standard ALPHA initialization includes

- Initialization of BOS (300,000 words working space)
- Initialization of HBOOK (50,000 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 on page 6 and 3.5.5 on page 7.

3.3 Event analysis routine

```plaintext
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 on page 70. 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

```plaintext
SUBROUTINE QUTERM
```

This subroutine can be used for anything which needs to be done at the end of a job, for example, 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 on page 64):

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

QMTERM, in turn, calls QUTERM.

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 the no new versions are provided.

3.5.1 New Run
SUBROUTINE QUNEWR (IROLD,IRNEW)

This routine is called from QMNEWR 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 (QUNEWR is called only once).

QUNEWR 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 / END.

3.5.2 Unknown 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/END.

3.5.3 Initialize the histogram package

SUBROUTINE QUIHIS

NOT intended for histogram booking (use QUINIT).
Called automatically from QMINIT.

Default: Initialize HBOOK4: CALL HLIMIT (50000).

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 300,000 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.

Data cards may also be used to enter your own data into the program. If your cards are given in standard BOS format, their content will be available as standard BOS banks.

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 on page 9), data cards can be given in any order.
4. The ENDQ card must be the last in the card file.

4.1 Input/Output

4.1.1 ALEPH file types

There are several ALEPH file types:

- **NATIVE**: machine-dependent input/output
- **EPIO**: machine-independent input/output
- **EDIR**: event directories
- **DAF**: direct access files (e.g., database)
- **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 parses the data set name to determine the format. For file names which do not follow this convention, see section 4.1.2.1 on page 9.

4.1.2 FILI: Input files

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

data-set-name: See examples in 4.1.2.1 on page 9.
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.

4.1.2.1 How to specify data set names on cards: Examples

DISK FILES:

VAX / VXCEPN

IBM / CERNVM

1. FILI 'PHY:TAUPO50.NATIVE | NATIVE'
2. FILI 'PHY:TAUPO50.NATIVE'
3. FILI 'PHY:TAUPO50.DATA | NATIVE'
4. FILI 'PHY:TAUPO50 EPIO *
5. FILI 'PHY:TAUPO50.EPIO'
6. FILI 'PHY:TAUPO50.EPIO | GIME PHY'

CRAY

7. FILI 'BBP22409 EPIO | EPIO | GIME PUBXU 408'

Explanation:

1. Complete specification. " | NATIVE" defines the file format. The vertical bar separates the file name from the parameters.
2. " | NATIVE" can be omitted here because the format is already specified in the data set name.
3. "DATA" is non-standard and not recommended. In such a case, the format must be given as a parameter: " | NATIVE".
4. On IBM: * can be omitted because file mode * is default for input data sets (file mode A is the default for output data sets).
5. data set names are accepted in IBM and in VAX syntax. On VAX, the IBM file mode is ignored. On IBM, the VAX directory specification is ignored. A period is translated into a blank space or vice versa.
6. On IBM: Execute a GIME.
7. File will be "acquired" from the IBM disk and stored as $STAGE/BBP22409.EPIO.

STAGED TAPES AND CARTRIDGES:

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

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; this option should never be used on the CRAY.

A future ALEPH note will give a full description of FILI (and FILO) card formats.
4.1.2.2 Run / event selection

Use the same cards as in the JULIA program:

SEVT 15 24 68 –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.

SRUN 2 –4 68 –10  Select RUNs 2,3,4,6,8,9,10

IRUN 15 7 11 –9999999  Ignore RUNs 1,5,7,11,12,13,14,...,9999999

NEVT 5 –7  Select the 5th, 6th, and 7th record (in the order they are stored on the input file regardless of their run / event numbers).

NEVT 3  Select the 1st, 2nd, and 3rd record.
More than two numbers are not allowed on this card.

4.1.3 FILO: Output files

Event output is controlled by the FILO card and by the subroutine QWRITE (see 13.2 on page 64). The data set name and options are given on the FILO card. QWRITE sets a flag that causes the current event to be written out.

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

data set name  same as on FILI cards; see examples in 4.1.2.1 on page 9. Output tape staging is not yet available on the VAX.
File format  NATIVE, EPIO, or EDIR

parameters:

ALLR  write all run records to the output file (default).
NORU  write no run records to the output file.
SELR  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.
SREC  write all “special” records to the output file. Without this card, all records which are neither event nor run records will not be written.

NOOV  simple mind-protection against involuntarily overwriting

Examples:  FILO 'ABC NATIVE | SELRU | NATIVE | NOOV'
The 2nd “NATIVE” is redundant; see 4.1.2.1 on page 9.

CRAY:  FILO 'MYDATA EPIO | EPIO | DISPOSE'
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 on page 8) is the same as the input event type. Event directories can be created from any input event type (see below). More complete docu-
mentionation of FILI and FILI cards will be given in a future ALEPH note.

4.1.3.1 I/O with Filedefs or Assign statements

For special applications, input and output files can be defined using filedefs or assign statements. As stated in Section 6.3.4 on page 21, unit 20 is used for input and unit 50 for event output. The FILI and FILO cards must be used to define the file format. For example, to read and write EPIO files using assign statements on the VAX, the card file must include the following lines:

FILI ' | EPIO'
FILO ' | EPIO'.

The corresponding assign statements are

ASS FILEIN.EPIO FOR020
ASS FILEOUT.EPIO FOR050.

4.1.3.2 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. A full discussion of the use of event directories will be given in a future ALEPH note; a brief description of the use of event directories with ALPHA is given here.

4.1.4.1 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 A'

  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 A | 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 on page 64):

  CALL QWCLAS(IBIT)     IBIT = 1, 30

(If three bits are to be set, QCLASW has to be called three times.) QWCLAS must be called before QWRITE. If QWCLAS is not called before QWRITE, the classification word will be set to zero.
4.1.4.2 Reading data with event directories

The event directory must be specified in the FILI card:

\[ \text{FILI 'TEST EDIR A'} \]

All of the run / event selection cards (Sec. 4.1.2.2 on page 10) 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.

4.1.4.3 CLAS: Select events with certain classification word

\[ \text{Format } \quad CLAS \ \text{ibit1, ibit2, } \ldots, \ \text{ibitm} \]

\[ \text{read events with bit ibit1 and/or ibit2 etc. on (set to 1)} \]

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).

\[ \text{Format: } \quad \text{COPY (no parameters)} \]

All ALPHA features except data card handling and event input / output are switched off. User routines are not called at all. 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 be stripped down to a simple copy job which digests the standard ALPHA data cards.

4.2 Histograms

The cards used in connection with the histogram package are described in detail in Chapter 5 on page 16. For completeness, the cards are listed here also.

4.2.1 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).

\[ \text{Format: } \quad \text{HIST 'data - set - name | parameters'} \]

- data set name see 4.1.1 on page 8.
- Default file format HIS

- parameters:
  - UPDA (optional — described in 5.2.1 on page 18)
  - NOOV

4.2.2 HTIT: General histogram title

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

\[ \text{Format: } \quad \text{HTIT 'This is the general title'} \]
4.2.3 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: \texttt{NOPH}

4.3 READ: Input from different card files

The READ card allows the reading of input cards from different card files.

Format: \texttt{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.

"READ" is not "INCLUDE" — every card file is completely read from its first card up to the ENDQ card. Afterwards, the next file specified by a READ card is read integrally, etc..

4.4 DEBU: Debug output

There are two debug levels:

- \texttt{DEBU 0} minimum debug output (no BOS summary and no particle table printed).
- \texttt{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 mnemonic symbol KDEBUG.

4.5 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: \texttt{UNPK ab cd ef ...'}

The two−character options are described in the code of the ALEPHLIB routine AUNPCK. The default options correspond to the card: \texttt{UNPK "TE EC HC MU FI"}; TPC and ITC coordinates are not unpacked by default.

4.6 Process cards

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

- \texttt{NOMC} no Monte−Carlo “truth”
- \texttt{NOCH} no CHarged tracks (also excludes V0’s)
- \texttt{NOCO} no CalOrimeters
- \texttt{NOV0} no V0’s
NONE     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.

4.7 FIEL: Magnetic field

Magnetic field can be set to a given value:

FIEL 15.  Set magnetic field to 15 K gauss.

4.8 TIME: Job time control

TIME 5  causes program termination (CALL QMTERM) if less than 5 more seconds are avail-

able.

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 mnemonic symbol QTIME (see 6.3.5
on page 22). On all CERN computers, time is counted in units equivalent to IBM
370/168 seconds.

4.9 Particle table

The cards used in connection with the ALPHA particle table are described in detail in Chapter 15 on
page 71. For completeness, the cards are listed here also.

4.9.1 PMOD: Modify particle attributes

Format:  PMOD 'part − name antipart − name ' mass charge life − time widih

Parameters:
'part − name antipart − name'

see 15.3 on page 71.

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 widih:

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.

4.9.2 PNEW: New particles

Modify attributes of an existing particle.

Format:  PNEW 'part − name antipart − name ' mass charge life − time widih

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

4.9.3 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 IMCantencode
Parameters:
'part – name antipart – name'
see 15.4.3 on page 72.
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.

4.10 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: Energy Flow

The EFLW card is used to select the option used for energy flow analysis done inside ALPHA. There are currently three options; see 11 on page 59.

Format 

    \( EFLW \ n \)

\( n = 1, 2, \) or 3

4.12 SYNT: Syntax Checker

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 file (except the log file) is created or modified even if the log file indicates otherwise.
- NO event is 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 on page 6 and 3.5.4 on page 6). 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 on page 18.

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 (...)

HBOOK1 always deletes an existing histogram and creates a new one. To make it possible to update existing histograms (see 4.2.1 on page 12), 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)

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

5.1.3 Book an Ntuple

CALL QBOOKN (ID, CHTITL, NVAR, TAGS)

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

Details: 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 on page 18).

See 5.1.1 on page 16 (QBOOK1 vs. HBOOK1): Existing Ntuples will not be overwritten (see 5.1.1 on page 16).

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 QHFRI).

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 on page 5 and 3.5.4 on page 6).

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:  

\[ \text{HIST 'data-set-name | parameters'} \]

data set name see 4.1.1 on page 8.
Default file format HIS
parameters (optional):

<table>
<thead>
<tr>
<th>UPDA</th>
<th>Update existing histograms. Can be used deliberately if a previous job terminated due to time limit etc. but ...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CAUTION: with this option, the old histogram file will be overwritten (even on VAX).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOOV</th>
<th>Overwrite protection (see 4.1.3 on page 10). Cannot be used with UPDA.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On VAX: Unnecessary.</td>
</tr>
<tr>
<td></td>
<td>On IBM: Strongly recommended. The first HBOOK action is to clear an existing file unless UPDA is specified.</td>
</tr>
</tbody>
</table>

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.

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 was used.

Format:  

\[ \text{NOPH} \]
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: \textit{HTIT 'This is the general title'}
Chapter 6

Mnemonic symbols

Mnemonic symbols are Fortran variables, arrays, parameters, functions, or statement functions. The only characteristic you have to care about is whether or not they are followed by an argument. Mnemonic symbols which give access to information for specific reconstructed or Monte Carlo objects are described in Chapter 8 on page 32.

The units used in ALPHA are cm, sec, GeV, GeV/c, GeV/c**2, kG.

6.1 Mathematical and physical constants

QQE \quad e = 2.718282
QQPI \quad \pi = 3.141593
QQ2PI \quad 2 \cdot \pi
QQPIH \quad \pi / 2
QQRADP \quad 180 / \pi
QQC \quad \text{speed of light} = 2.997925E10 \text{ cm/sec}
QQIRP \quad \text{speed of light in units cm / KGauss}
\quad \text{(inverse track bending radius -- > track momentum)}
QQH \quad \text{Planck constant} / 2\pi = 6.582173E-25 \text{ GeV sec}
QQHC \quad \text{Q} \cdot \text{Q}

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

6.2 Event / run header

6.2.1 Event header: from bank EVEH

KEXP \quad \text{Exp Number}
KRUN \quad \text{Run Number}
KEVT \quad \text{Event Number}
QLEELP \quad \text{LEP Energy} = 2 \cdot \text{beam energy [GeV/c**2]} \text{ (not KeV !)}
QMFLD \quad \text{Magnetic Field} \text{ (kG)}
KEVERT \quad \text{Run Type}
KEVEDA \quad \text{Date}
KEVETI \quad \text{Time}
KEVEMI(I) \quad \text{trigger Mask I, I = 1 to 4}
KEVETY \quad \text{event TYpe}
KEVEES \quad \text{Error Status}

6.2.2 General event information: from bank DHEA

KDHEFP \quad \text{Flag for Physics identification}
KDHENX \quad \text{Nr of reconstructed vertices}
KDHENP \quad \text{Nr of positive reconstructed tracks}
KDHENM \quad \text{Nr of negative reconstructed tracks}
KDHENV \quad \text{Nr of reconstructed V0's}
KDHENJ \quad \text{Nr of reconstructed jets}
QDHEEC \quad \text{total Energy of Charged tracks}
QDHEEL \quad \text{total Energy of Cal objects}
QDHEFP \quad \text{abs(P) of energy Flow}
QDHETH  THeta of energy flow
QDHEPH  PHI of energy flow
QDHEEF  Energy Flow
QDHEET  abs value of Et
QDHET1  Theta of momentum tensor axis 1
QDHEP1  Phi of momentum tensor axis 1
QDHET2  Theta of momentum tensor axis 2
QDHEP2  Phi of momentum tensor axis 2
QDHEE1  1st Eigenvalue of momentum tensor
QDHEE2  2nd Eigenvalue of momentum tensor
QDHEE3  3rd Eigenvalue of momentum tensor
KDHERD  Flag for Reconstruction Status (future)

6.2.3 Event generator status: from bank KEVH

KKEVID  process ID
QKEVWT  WeighT

6.3 ALPHA Internal Constants, Variables

6.3.1 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

6.3.2 Program status

KSTATU  -1 : program initialisation phase
        0 : event processing
        1 : program termination phase
KDEBUG  debug level (see 4.4 on page 13)

6.3.3 Event status

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

6.3.4 input / output units

KUINPU  event input = 20
KUOUTP  event output = 50
KUEDIN  event directory input = 30
KUEDOU  event directory output = 60
KUCONS  data base = 4
KUPRTN  log file = 6
KUPTER  terminal = 76 or 0 (see 2 on page 2)
KUCARD  card input = IW(S) = 7
KUHIST  histogram output unit = 15
KURTOX  EXCH format histogram output on CRAY = 16
6.3.5 Timing

QTIME  seconds given on the TIME card (see 4.8 on page 14
QTIMEL  remaining job time (= 9999. for interactive jobs)

On all CERN computers, the time units are IBM 370/168 seconds. For getting the
execution time or time differences, use the CERNLIB (Z008) routines TIMAX(T)
and TIMAD(T) which are not called in ALPHA routines.

6.3.6 Character variables

CQVERS  ALPHA version number (3 digits)
CQVERS  ALPHA correction file (5 digits)
CQDATE  date at start of job (8 char)
CQTIME  time at start of job (8 char)
CQFINP  data set name of current event input file
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 (e.g., 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 on page 32 and 8.3 on page 37, respectively.

In the following sections, several methods for determining ALPHA track and vertex numbers are described. All of these methods can be arbitrarily 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.

\[
\text{DO 10 ITK = KFCHT, KLCHT}\\
\text{CALL HF1(47, QP(ITK), 1.)}\\
\text{10 CONTINUE}
\]

KFCHT, KLCHT number of First (Last) Charged track. KFCHT, KLCHT are mnemonic symbols; their proper values are provided automatically.
ITK loop index = ALPHA track number
QP(ITK) momentum of track ITK (see 8.1.1 on page 32)

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

\[
\text{DO 10 ITK = KFCHT, KLCHT}
\]

will not be executed.

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
Charged tracks reconstructed in the TPC, ITC, Vdet.

**Calorimeter Objects:** KFCOT, KLCOT, KNCOT
Charged tracks are ignored. Calorimeter objects can be any of the following:

- Ecal objects with no associated Hcal object.
- Heal objects with no associated Ecal object.
- Composite cal. objects consisting of at least one Ecal and Heal object associated to each other. See 7.2 on page 25 for getting access to the contributing Ecal and Heal objects separately. See the end of Section 7.3 on page 25 for a more detailed description of composite cal. objects in ALPHA.

Calorimeter objects can be further divided into:

**Isolated cal. objects:** KFIST, KLST, 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 V0's:** KFV0T, KLV0T, KNV0T
See Section 7.5.1 on page 29 for comments on the daughters of V0s.

**Tracks from DeCay Vertices:** KFDCT, KLDCT, KNDCT
Charged tracks outgoing from reconstructed DeCay vertices. The momentum and error matrix for these tracks is calculated relative to the secondary vertex position. Currently, this section includes the daughter tracks from reconstructed V0's.

**Energy flow objects:** KFEFT, KLEFT, KNEFT
The content of this section depends on the option selected with the EFLW card (see Ch. 11 on page 59) this section is not filled unless the EFLW card is included in the card file.

**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

Note that "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 on page 26. (DO loops cannot be used because the objects are not stored in subsequent memory locations.)

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

(The functions KPDIR and KFOLLO are described in (see 7.4 on page 26 below.) The corresponding loop for Hcal objects is:

```
IJOB = 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,

```
IJOB = ... any calorimeter object ...
DO 10 N = 1, KNCHGD (IJOB)
IChGD = KCHGD (IJOB, 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 Heal 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 Heal objects, so the mother—daughter or daughter—mother relation described in 7.5 on page 29 can be used for all calorimeter objects.

Note that in the current version of ALPHA, the composite calorimeter objects in ALPHA are not identical to those in the PCRL bank. ALPHA composite cal. 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 shared 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 on page 23. Two quantities must be specified:

- The particle name (example: 'E+' or 'GAMMA'); see 15.1 on page 71.
- 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 9.3 on page 46). 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 KVSAVE in 9.2.9 on page 43).

The particle name of MC particles is specified in the MC particle table (see 15.1 on page 71). Reconstructed objects have the names 'CHARGED', 'ECAL', 'HCAL', and 'CALOBJ' for charged tracks, Ecal objects, Heal objects, and unspecified (e.g. composite) calorimeter objects, respectively. The functions KVSAVE and KVSAVE can be used to create new tracks with a name. A list of standard particle names is given in D on page 82. New particle names can be introduced by using them in ALPHA subroutine calls or by specifying them on data cards (see 15.4.2 on page 72).

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 9.3 on page 46).
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. This lookup is fast: it takes about 1/3 of the time of the HBOOK routine HF1. In nested loops, however, 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 on page 26 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+', KMONE, IANTI) with KCDIRA (IP, KMONE, 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.

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

C---First select pion candidates

```
DO 5 ITK=KFCHT,KLCHT
   IF(condition to select pions) THEN
      ISAVE=KVSAVE(ITK,'PI+')
   ENDIF
5 CONTINUE
```

C---Loop over selected pions.

```
   IPIONE = KPDIR ('PI+', KRECO)
10   IF (IPIONE .NE. 0) THEN
      IPITWO = KFOLLO (IPIONE)
   ENDIF
```

```
   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 9.2.8 on page 42 for the use of KVSAVE.

7.4.5.2 $\Delta^{++} \rightarrow p \pi^+$

Proton and pion candidates must be selected and saved with KVSAVE before this code is reached (see 9.2.8 on page 42).

```
   IPROT = KPDIR ('P', KRECO)
10   IF (IPROT .NE. 0) THEN
      IPIPLU = KPDIR ('PI+', KRECO)
20   IF (IPIPLU .EQ. 0) THEN
      IF (.NOT.XSAME(IPROT,IPIPLU)) THEN  \--- important
```

```
   ... 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 on page 34) tests whether the two contributing particles are based on different reconstructed objects or simply on different mass hypotheses of the same recon-
structured 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 on page 39.

\[ \text{IMOTH} = \ldots \text{ (track number of a mother particle)} \]
\[ \text{DO 10 I = 1, KNDAU (IMOTH)} \]
\[ \text{IDAUGH = KDAU (IMOTH,I)} \]
\[ \text{CALL HFILL (47, QP(IDAUGH))} \]
\[ 10 \text{ CONTINUE} \]

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 on page 23). 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 on page 30) returns the CHT track number corresponding to a track in the DCT section.

Example:

\[ \text{DO 10 IVO=KVF0T,KLV0} \]
\[ c--- \text{First daughter of V0 (in DCT section)} \]
\[ \text{I1DCT=KDAU(IVO,1)} \]
\[ C --- \text{Corresponding track in CHT section.} \]
\[ \text{I1CHT=KCHT(I1DCT)} \]
\[ 10 \text{ CONTINUE} \]

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 9.2.8 on page 42). The QVADDx routines (9.2.2 on page 39) and the jet / event topology routines 10 on page 48) do NOT set up this relation.

\[ \text{IDAUGH} = \ldots \text{ (track number of a daughter particle)} \]
\[ \text{DO 10 I = 1, KNMOTH (IDAUGH)} \]
\[ \text{IMOTH = KMOTH (IDAUGH,I)} \]
\[ \text{CALL HFILL (47, QP(IMOTH))} \]
\[ 10 \text{ CONTINUE} \]

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:

```plaintext
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 ITKIRST (see 9.3 on page 46):
IF (KCLASS (ITKSAM) .EQ. ITKIRST) 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 9.3.1 on page 46) 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 $KCHT \leq ITK \leq KCHT$, $KCHT (ITK)$ 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:

```plaintext
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 I'th 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, KVNDAU (IVX)
       ITK = KVDAU (IVX, I)
       CALL HFILL (47, QP(ITK))
   10    CONTINUE

KVNDAU (IVX)  number of outgoing tracks
KVDAU (IVX,I)  track number of I'th outgoing track
Chapter 8

ALPHA Track and Vertex Attributes

The units used throughout ALPHA are cm, sec, GeV, GeV/c, GeV/c**2, 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

QP (I) \quad P = \text{momentum of vector I.}
QX (I) \quad X \text{ momentum component}
QY (I) \quad Y \text{ momentum component}
QZ (I) \quad Z \text{ momentum component}
QE (I) \quad \text{Energy}
QM (I) \quad \text{Mass (use QMASV0 for V0 mass; see below)}
QCH (I) \quad \text{CHarge}
KCH (I) \quad \text{NINT (QCH(I)) (be careful with quarks)}

For charged tracks, the pion mass is assumed; the mass can be changed with QVSETM (see 9.2.10.2 on page 44). For angles and more kinematics quantities, see 9.1 on page 38.

8.1.2 V0 Mass

QMASV0 (I,'name')

\text{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.

8.1.3 Track error covariance matrix

XSIG (I) \quad .TRUE. if covariance matrix available
QSIG (I,N,M) \quad \text{element (N,M) of the covariance matrix}
\quad N,M = 1,2,3,4 \text{ in the order QX,QY,QZ,QE}
QSIGEE (I) \quad \text{Error**2 on energy}
QSIGE (I) \quad \text{Error on energy}
QSIGPP (I) \quad \text{Error**2 on momentum}
QSIGP (I) \quad \text{Error on momentum}
QSIGMM (I) \quad \text{Error**2 on mass}
QSIGM (I) \quad \text{Error on mass}

\text{The mass error is not defined for particles with mass = 0.}

QSIG (I,1,1) is set to a negative value if the matrix is not available.
8.1.4 Distance to the beam line / origin vertex:

Available for charged reconstructed tracks.

QDB (I)         distance of closest approach to beam axis
QDBS2 (I)       error*2 on QDB
QZB (I)          z coordinate of track point where QDB is measured
QZBS2 (I)        error*2 on QZB
QBC2 (I)         chi*2 due to QDB and QZB.
Remark: Currently, the beam position is not available, and QDB and QZB are measured with respect to the z-axis ( = QFRFD0/QFRFZ0; see 8.2.1 on page 34)

For more geometrical track attributes, see sections 8.2.1 on page 34 and 9.1 on page 38.

8.1.5 Stability code

KSTABC (I)       Stability code

Invented for MC particles and set accordingly for reconstructed tracks and reconstructed composite particles.

A loop over all MC particles with KSTABC > 0 never counts energy twice. In decay chains, the generation of decay particles which has the best chance to be visible in the detector is selected. The energy sum of these particles gives the total energy only if no particle interacts 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.

The specific 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.

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 on page 27):
XCEQU (I, IPC)  
XCEQOR (I, IPC)  
XCEQAN (I, IPC, IANT1)

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 on page 30) or, in other words, belong to the same family (see 10.2.3 on page 50). I and J must both be reconstructed tracks or MC particles; they may, however, belong to different Lorentz frames. XSAME uses the same track bit masks as the lock algorithm. XSAME(IJET,ITK) can be used for testing whether a track ITK belongs to a given jet IJET or not (see 10.3 on page 51). An example how to use XSAME in reconstructing decay chains is given in 7.4.5 on page 28, example 2.

8.1.8 Flags, pointers, etc.

Pointers to other tracks and to vertices: see ch. 7 on page 23.

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 on page 26.

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 9.2.11 on page 45.

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 9.2.11 on page 45.

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 even if the corresponding bank does not exist. The names of these mnemonic symbols follow the same convention as the HAC parameters.

8.2.1 Global geometrical track fit: Bank FRFT

XFRF (I) .TRUE. if track fit data are available for track I
QFRFIR (I) Inverse radius of curvature in x − y projection
  Signed +ve if track bends counterclockwise, −ve if track bends clockwise
QFRFTL (I) Tangent of dip angle
QFRFP0 (I) Phi at closest approach to the z axis
QFRFD0 (I) Distance of closest approach to z axis
QFRFZ0 (I) z coordinate of track point where QFRFD0 is measured
Note: QDB and QZB (see 8.1.4 on page 33) correspond to the closest approach to the beam axis.

**QFRFAL (I)**
Multiple scattering angle between TPC and ITC

**QFRFEM (I,N,M)**
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

**QFRFC2 (I)**
Chi-square of helix fit

**KFRFDF (I)**
Number of degrees of freedom

**KFRFNO (I)**
See ALEPH BOS bank documentation

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

**KFRTNV (I)**
Number of coordinates in Vdet

**KFRTNI (I)**
Number of coordinates in ITC

**KFRTNE (I)**
Number of coordinates in ITC in following spirals

**KFRNTT (I)**
Number of coordinates in TPC

**KFRTRR (I)**
Number of coordinates in TPC in following spirals

### 8.2.3 Charged – particle identification: Bank FRID

**KFRIBP (I)**
Bit pattern for tracking devices

**KFRIDZ (I)**
Dead zone pattern for tracking devices

**KFRIBC (I)**
Bit pattern for calorimeters

**KFRIDC (I)**
Dead zone pattern for calorimeters

**QFRIPE (I)**
Electron probability

**QFRIPM (I)**
Muon probability

**QFRIP (I)**
Pion probability

**QFRIK (I)**
Kaon probability

**QFRIP (I)**
Proton probability

**QFRINK (I)**
No Kink probability

### 8.2.4 dE/dx data: Bank TEXS

**XTEX (I)**
.TRUE. if dE/dx is available for track I

**KNTEX (I)**
Number of TPC sectors on track I

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

**KTEXSI (I,N)**
Sector slot number

**QTEXTM (I,N)**
Truncated Mean of dE/dx measurements

**QTEXTL (I,N)**
Useful Track Length for dE/dx

**KTEXNS (I,N)**
Number of Samples used for dE/dx

**QTEXAD (I,N)**
Average Drift length of samples

### 8.2.5 Electron identification: Bank EIDT

**XEID (I)**
.TRUE. if electron identification is available for track I

**KEIDIF (I)**
Quality flag

**QEIDRI (I,N)**
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.

**QEIDEC (I)**
Corrected energy with electron hypothesis

**KEIDIP (I)**
Particle hypothesis (= 1 if electron)

**QEIDEI (I,N)**
Energy in centered storeys stack N
8.2.6 Muon - HCAL association: Bank HMAD

XHMA (I) \( .\text{TRUE. if HCAL data are available for track I} \)
KHMANF (I) Number of Fired planes
KHMANE (I) Number of Expected fired planes
KHMANL (I) Number of Fired planes within Last ten planes
KHMAMMH (I) Mult Hits: number of clusters in last ten planes
KHMAG (I) IGeomflag: flag of possible dead zone
QHMAED (I) Energy Deposit in corresponding Hcal storey
QHMACS (I) ChiSquare
KHMAND (I) Number of Degrees of freedom
KHMAM (I) Expected bit map
KHMAM (I) True bit map
KHMAM (I) Preliminary identification flag

8.2.7 Muon chamber data: Bank MCAD

XMCA (I) \( .\text{TRUE. if muon chamber data are available for track I} \)
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
QMCAM (I) Min. angle between extrap and measured (in muon ch) track
QMCAM (I) Cutoff on minimum angle

8.2.8 Ecal objects: Bank PECO

XPEC (I) \( .\text{TRUE. if ECAL data (PECO) are available for track I} \)
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) Calobject number

8.2.9 Ecal objects: Bank PEPT

XPEP (I) \( .\text{TRUE. if ECAL data (PEPT) are available for track I} \)
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

8.2.10 Hcal objects: Bank PHCO

XPHC (I) \( .\text{TRUE. if HCAL data (PHCO) are available for track I} \)
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)  Calobject number

8.2.11 Reconstructed V0's: Bank YV0V

XYV0 (I)  .TRUE. if V0 data are available for track I
KYV0K1 (I)  FRFT track number of positive track from V0
KYV0K2 (I)  FRFT track number of negative track from V0
QYV0VX (I)  V0 x coordinate
QYV0VY (I)  V0 y coordinate
QYV0VZ (I)  V0 z coordinate
QYV0X1 (I)  First constraint on V0 mass (r in ALEPH 88–46)
QYV0X2 (I)  Second constraint on V0 mass (b in ALEPH 88–46)
QYV0C2 (I)  Chisquare of V0 vertex fit
KYV0IC (I)  Fit hypothesis (see YV0V bank description)

8.3 Vertex attributes

The following attributes are stored in bank QVRT, and are available for all species of vertices. The argument (IVX) always refers to vertex IVX.

QVX (IVX)  x position
QVY (IVX)  y position
QVZ (IVX)  z position
KVN (IVX)  JULIA/GALEPH vertex number
QVEM (IVX,N,M)  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 a negative value if the error matrix is not available.

See Section 7.8 on page 31 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, a set of routines for doing Lorentz transformations are given.

9.1 Scalar quantities

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

QCT (I) \[ \text{Cos(Theta)} = \cos(\text{polar angle}) \]
QPH (I) \[ \text{PHI} = \text{azimuth (radians)} \]
QPT (I) \[ \text{Transverse momentum (with respect to the beam line)} \]
QBETA (I) \[ \text{beta} \]
QGAMMA (I) \[ \text{gamma} \]

Remark: Masses are set to a negative value if \( (E^2 - p^2) \) is negative.

QMSQ2 (I,J) \[ \text{(invariant mass)}^2 \text{ of particles I and J} \]
QM2 (I,J) \[ \text{invariant mass of particles I and J} \]
QMSQ3 (I,J,K) \[ \text{(invariant mass)}^2 \text{ of particles I, J, and K} \]
QM3 (I,J,K) \[ \text{invariant mass of particles I, J, and K} \]
QMSQ4 (I,J,K,L) \[ \text{(invariant mass)}^2 \text{ of particles I, J, K, and L} \]
QM4 (I,J,K,L) \[ \text{invariant mass of particles I, J, K, and L} \]
QDMSQ (I,J) \[ \text{mass}^2 \text{ of the } 4-\text{momentum difference } |p(I) - p(J)|. \]

In a decay \( I \rightarrow J + x \), QDMSQ(I,J) gives the mass\(^2\) of x.

QPPAR (I,J) \[ \text{momentum component of particle I parallel to particle J} \]
QPPER (I,J) \[ \text{momentum component of particle I perpend. to particle J} \]
QDPT3 (I,J) \[ \text{scalar product of momentum vectors I and J (3-vectors)} \]
QDPT4 (I,J) \[ \text{scalar product of 4-vectors I and J} \]

= \[ \text{QE(I)} \cdot \text{QE(J)} = \text{QDPT3(I,J)} \]
QCOSA (I,J) \[ \text{cos (angle between tracks I and J) (lab frame)} \]
QDECA2(I,J) \[ \text{cos (decay angle):} \]
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) \[ \text{extension of QDECA2 for the n-body decay } I \rightarrow J + \text{any. Note the different meaning of the first argument in QDECA2 and QDECAN.} \]

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 9.2.7 on page 42). An alternative possible
working space, which is not considered here, is the user's track section (see remarks in 9.2.7 on
page 42). New tracks can be created whenever necessary, but they should not be created, if possible,
inside loops. A warning is issued if an "input" track is used as working space.

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.

Examples:
• Flags for the lock algorithm (see 10.2.3 on page 50). Thus, with CALL QLOCK (ISUM), you
  lock ITK1 and ITK2 as well.
• Mother—daughter relation (see 7.5.1 on page 29) from ISUM to ITK1 and ITK2 but NOT the
  reverse daughter—mother relation; see KVSAVE in 9.2.8 on page 42)

In subroutine calls, the result is stored in the track denoted by the first subroutine argument: 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 on page 26). If you
really want to mix up tracks from different classes, they first have to be "saved" in the same class (see
KVSAVC in 9.2.9 on page 43).

9.2.2 Add 4—momenta of particles

9.2.2.1 Add two particles

    CALL QVADD2 (ISUM, ITK1, ITK2)

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

9.2.2.2 Add three particles

    CALL QVADD3 (ISUM, ITK1, ITK2, ITK3)

Add the 4—momenta of ITK1, ITK2, ITK3 and store the result in ISUM.
9.2.2.3 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.

9.2.2.4 Add N particles

CALL QVADDN (ISUM, ITK)

For adding more than four particles, either use QJADDP (see 10.3 on page 51) 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 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 on page 32) are copied. Otherwise, all flags, relations, etc. are copied or set as far as it makes sense. See remarks about lock algorithm in sections 10.2.1 on page 49 and 10.2.3 on page 50.

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 (9.2.8 on page 42).

9.2.4 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.

9.2.5 Drop tracks

CALL QVDROP ('part−name', ICLASS)

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

will drop tracks with any particle name in class ICLASS. The main application of this subroutine is to drop all tracks in a specific Lorentz frame. See the example in section 9.3.3 on page 47.

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. (A BOS garbage collection is absolutely forbidden in the ALPHA package!)

9.2.6 Copy track attributes into a Fortran array

For the reverse action (copy Fortran array into track), see section 9.2.10 on page 44.

9.2.6.1 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).

9.2.6.2 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).
9.2.6.3 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).

9.2.7 Create a new track

INEW = KVNEW (DUMMY)

Create a new track (see 9.2.1 on page 38) 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 particle name. None of the access methods described in ch. 7 on page 23 give access to it the only access is the user controlled variable INEW. Consequently, it can never be dropped (see 9.2.5 on page 41). The new track does NOT belong to a specific class (KRECO / KMONTE / Lorentz frame).

9.2.8 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 on page 26). Note that the mass is NOT changed in KVSAVE.

The class (KRECO / KMONTE / Lorentz frame; see 7.4.1 on page 26) 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 9.2.9 on page 43) makes it possible to implement 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 9.2.9 on page 43).

If 'part-name' is equal to '', KVSAVE only performs a copy, and the track copy has no particle name. In contrast to QVCOPY (see 9.2.3 on page 40), 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: ψ→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
   ITK2 = KFOLLOW (ITK2)
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')
   GO TO 20
ENDIF
    ITK1 = KFOLLOW (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 \( \psi \rightarrow \pi^+ \pi^- \) in the same way as \( \psi \rightarrow e^+ e^- \).

### 9.2.9 Save a track with class ICLASS

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

Save (see 9.2.8 on page 42) track ITK in track class ICLASS independent of the track history. Track classes are described in 7.4.1 on page 26. If class ICLASS does not yet exist, a new class is created. Note that the maximum number of new classes is six (see 9.3 on page 46).

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 = KVSAVE (INEW, 'BEAME+', KRECO)
```

KVSAVE has to be used here instead of KVSAVE because the track history of INEW does not specify the track class.
9.2.10 Modify track parameters

For the reverse action (copy track \( \rightarrow \rightarrow \) Fortran array), see 9.2.6 on page 41. The QVSxxx routines described below modify the specified track attributes but NEVER 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 kinematics.

9.2.10.1 Scale track momentum

```fortran
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 legally for "input" tracks.

9.2.10.2 Set mass of a track

```fortran
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 legally for "input" tracks.

9.2.10.3 Set 3-momentum of a track

```fortran
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.

9.2.10.4 Set 4-momentum of a track

```fortran
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 9.2.9 on page 43.
9.2.10.5 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.

9.2.11 Set User Track Flags

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

ITK
ALPHA "track" number

IFLAG
Flag number: 1 – 18

VAL, IVAL
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 on page 34.

9.2.12 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.

9.2.13 Zero track attributes

CALL QVZERO (ITK)

Set all attributes (momentum etc.) of ITK to 0. Note that KVNEW (see 9.2.7 on page 42) implies QVZERO.
9.3 Lorentz transformations

See also QDECAx (decay angle in the rest frame of a decaying particle) in 9.1 on page 38.

9.3.1 Boost a track and its daughters

\[
\text{IBOOST} = \text{KTLOR} (\text{ITK}, \text{IREST})
\]

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 on page 26. Another way to access boosted tracks is to use KSAME (see 7.6 on page 30), 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 9.3.3 on page 47.

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 9.2.5 on page 41) in order to reduce the number of frames in use, and to release the space occupied by the boosted tracks.

9.3.2 Boost a track

\[
\text{IBOOST} = \text{KTLOR1} (\text{ITK}, \text{IREST})
\]

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.
9.3.3 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 on page 48. In particular, selection options can be set by the routines QIOPTR or QJOPTM (see 10.1 on page 48), 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 7.4.1 on page 26)
  ITK = KPDIR ('CHARGED', IREST)
C use a loop with KFOLLOW. 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 about the same for all of these subroutines, and are described in detail in Section 10.3 on page 51. The particles 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 9.3.3 on page 47). In addition, tracks can be excluded from analysis by the routines described in this chapter (QJxxxx) with the LOCK routines described below.

10.1 Options for "QJ" 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 on page 23)
- `CO`: Calorimeter objects
- `CH`: Charged tracks
- `EC`: Energy flow using cal. stories; see 11 on page 59.
- `EF`: Energy flow using calorimeter objects and tracks or calorimeter objects alone depending on EFLW option; see section 11.1 on page 59.
- `AL`: All objects (charged tracks, cal. objects, ECAL ECAL objects, HCAL objects, V0's, V0 daughter tracks). If not applied skillfully together with LOCK, some 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) which are not contained in the loops. If no additional particles are to be considered, the argument `''` must be given. For example,

```
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 (see 10.3.3 on page 52).

```
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 code > 0 (default; see 8.1.5 on page 33
       'VI' stands for 'best chance to be visible')
'EP': Only particles with stability code 1, 2, and -3. 'EP' stands for 'energy-
       momentum conservation'.
'AL': All objects. If not applied skillfully together with LOCK, some 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 giv-
en 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 and two admittedly complicated ones with a
broader scope of applications. If desired, several locks can be used simultaneously. A track is consid-
ered "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 QLITK (ITK)

ITK ALPHA "track" number
Remarks: In contrast to the other locks described below, QLITK flags the object ITK and
nothing else. It cannot be used to lock particles belonging to a jet axis.
       Whenever an ALPHA routine makes a copy of a track locked by QLITK, the
       copy is NOT locked automatically.

10.2.2 Unlock a single track
CALL QLUITK (ITK)

ITK
ALPHA "track" number
Remark:
QLUITK opens only the lock set by QLITK. 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 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 on page 51. 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, use QLEAVE 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. Note that the lock routines are fast; they use bit masks rather than performing loops over particles.

10.2.4 Unlock tracks

CALL QLZER (IREMC)

IREMC
KRECO: reconstructed tracks
KMONTE: 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 on page 26).
IREMC is a necessary input for all lock routines. In QLOCK(ITK), its value is deduced from the track class to which ITK belongs to.

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

CALL QLREV (IREMC)

IREMC (see 10.2.4 on page 50).

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 QIADDP (SCALAR, 'vector-name', ICLASS)

For adding momenta of a few particles, see 9.2.2 on page 39. (NOTE: All of the QIxxxx routines have similar arguments. The arguments are explained fully in this explanation of QIADDP.)

10.3.1 Input argument

ICLASS Class = KRECO or KMONTE or a Lorentz frame identifier (see 7.4.1 on page 26). 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 9.2.10 on page 44).
If ICLASS refers to a Lorentz frame:

Particles not boosted into the frame are ignored without notification. The routine QTCLAS (see 9.3.3 on page 47) 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 on page 28).

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 ‘result−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 HP2 (4711, QP(ISUM), QM(ISUM), 1.)
```

Other routines may output several vectors; a loop using KFOLLOW (see 7.4.2 on page 26) must be constructed to access all of them.

Locking an output vector locks all particles contributing to it (see 10.2.3 on page 50). 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 on page 34).

10.3.3 Pseudo−particles

The output vectors of "QJ" routines (and only these) 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.

Moreover, 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 on page 51.
Scalar results EIGVAL eigenvalues in descending order with DIMENSION EIGVAL(3).
Spericity = 1.5 * (1. - EIGVAL(1))
Aplanarity = 1.5 * EIGVAL(3)
Planarity = EIGVAL(3) / EIGVAL(2)

Output vectors 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

\[ M_{jk} = \frac{1}{P} \sum_i \frac{P_j P_k}{P_i} \]

\[ j,k = 1,2,3 \]

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 on page 51.
Output arguments:

**Scalar result**  SPHERI value.
**Output vector**  Sphericity axis with the name ‘spheric-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 on page 51.

Output arguments:

**Scalar result**  THRUST value.
**Output vector (exactly one)**  Thrust axis with the name ‘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 on page 51.

Output argument:

**FOXWOL**  Fox – Wolfram moments H0 – H3; DIMENSION FOXWOL(5).
10.9 Divide event into two hemispheres

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

Input argument

ICLASS  
*described in 10.3 on page 51.*

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 COSCUT = 0.

Output arguments:

Two output vectors:

'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 on page 50).

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

Note that in the above example, two of maximum six Lorentz frames are in use. They can be dropped by the statement CALL QVDROP (' ', IVEC(IHEMI)) inside the loop (see 9.2.5 on page 41).
10.10 Missing energy, mass, momentum

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

Input argument

ICLASS described in 10.3 on page 51.
ITOTAL = 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.

Output arguments:

Scalar result PMISS: Missing momentum.
4 - vector with the name 'miss - 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 on page 51.
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.
Output arguments

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

EXAMPLE:

    DIMENSION LISTEJ(300)
    CHARACTER*13 CNAM
    C---Select option.
    CALL QJOPTR('RE',' ')
    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)
      DO 210 J = 1, NJT
      C--- get ALPHA number for next jet found:
      IF(J.GT.1) JJ = KFOLLO(JJ)
      C--- get the list of tracks merged into this jet:
      LL = 0
      DO 211 L = KFRET, KLRRET
      C--- check if this track is locked by 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)
    210 CONTINUE
    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 

\[(M/EVIS)^2 < \sqrt{2(1 - \cos2\delta)}\]

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

\[M^2 = 2(E_i E_j)(1 - \cos \theta_{ij})\]
Input arguments

ICLASS described in 10.3 on page 51.
ALPHA 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 \( \sqrt{2(1 - \cos 2\delta)} \).
DELTA Half opening angle cut in degrees.
ETA Cut on jet energies (fraction of \( 2\text{Evis} \)); only jets with energies > \( 2\eta \text{Evis} \) are kept.
EVIS The visible energy of the event; if EVIS equals 0, the visible energy is computed as the sum of the input particle energies.

Result

NJETS is the number of "jets".
'name' Vector 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

ICLASS described in 10.3 on page 51.
MINCLU Minimum number of clusters to be reconstructed. (if \( < 0 \), work space momenta are used as a start) (usually = 1)
DMAX1 Max. distance to form starting clusters (usually = 0.25 GeV)
DMAX2 Max. distance to join 2 clusters (usually = 2.5 GeV)
MULSYM

\[ = 1 \text{ for symmetric distance criterion (usual)} \]
\[ = 2 \text{ for multicity distance criterion} \]

Output Arguments

NJETS is the number of "jets"

\[ = -1 \text{ if not enough particles} \]
\[ = -2 \text{ if not enough working space (KTBOMX)} \]
TGEN Generalized thrust
DMIN Minimum distance between 2 jets

\[ = 0 \text{ when only 1 jet} \]
\[ = -1, -2 \text{ as for NJET} \]
'name' Vector containing 4-momenta of the jets.
Chapter 11

Energy Flow

In this chapter, the ALPHA implemetation of the Energy Flow algorithm of Minard and Pepe is described. (For a full description of the algorithm, see ALEPH 89 – 110.)

Global event properties (eg., total energy) calculated from the energy flow analysis done in JULIA are stored in the DHEA bank (for JULIA versions >= 225) and are available in ALPHA as fortran variables (see 6.2.2 on page 20).

Within ALPHA, it is also possible to use the energy flow analysis at the calorimeter story or calorimeter object level. As described in section 2.6 of ALEPH 89 – 110, different combinations of tracks and calorimeters may be used in the energy flow algorithm. Two of these methods can be used in ALPHA:

1. Calorimeters alone (no masking) – charged tracks are not considered and calorimeter energies are given weights according to theta and stack.

2. Tracks plus unmasked calorimeter energy – charged tracks and calorimeter energy outside of the "mask" are considered. As in method 1, weights are applied to the calorimeter energies; in the default version, weights are also applied to track momenta.

11.1 EFLW: Select Energy Flow Option

To use the energy flow analysis, the EFLW card must be given in the ALPHA card file:

**EFLW option**  option = 1, 2, or 3 (options are described below)

When the EFLW card is present, a new track section (EFT) is created. The content of the EFT section depends on the EFLW option selected. This section is similar to the CHT or COT sections already present in ALPHA – all of the usual variables exist (eg., KFEFT, KLEFT, KNEFT, etc.).

For now, it is also necessary to include the following line in your card file:

```
READ 'EFLOW'  (on IBM)
```

or

```
READ 'PHY:EFLOW' (on VAX)
```

This statement will cause the file EFLOW.CARDS to be read with your card file; EFLOW.CARDS contains the weights used for the energy flow analysis. (The weighting factors depend on the GALEPH and JULIA version.)

**EFLW Options:**

**EFLW 1**  The EFT section contains a copy of the calorimeter objects from the COT section, but the energy weights are applied according to the energy flow algorithm for calorimeters alone (method 1 above).
EFLW 2 (default)  
This option is the default if the EFLW card is given without an option. The EFT section contains the results of the full energy flow analysis: selected tracks plus cal. energy outside of the masks (method 2 above).

EFLW 3  
This option makes it possible to compare the two energy flow methods in a single job. The EFT section is filled as described for option 2. The standard calorimeter object section (COT), however, is filled using the energy flow weights for calorimeters alone (method 1 above).

11.2 Using Event Topology Routines with Energy Flow

Subroutine QJOPTR (10.1 on page 48) includes three options for using energy flow information with the QJxxxx routines described in Chapter 10 on page 48. The options are:

'\texttt{reco-option} ' 'EC': Energy flow using cal. stories.  
'TM': Energy flow using cal. stories and tracks.  
'EF': Energy flow using calorimeter objects and tracks or calorimeter objects alone depending on EFLW option used.

In most cases, the 'EF' option should be used. If the 'EC' or 'EM' option is selected, no other ALPHA features are supported (e.g., it is not possible to lock stories or to maintain mother–daughter relationships between jets and stories). The analysis done in JULIA, however, is done at the story level, so the 'EM' and 'EC' options make it possible to repeat calculations done in JULIA with different event topology algorithms.

11.3 Examples

11.3.1 Calculate total energy from energy flow algorithm with calorimeters alone

a. In card file:

\begin{verbatim}
  EFLW 1  
  READ 'EFLOW'
\end{verbatim}

b. In user routine:

\begin{verbatim}
  E=0.  
  DO 10 I=KFEFT,KLEFT  
  10 E=E+QE(I)
\end{verbatim}

11.3.2 Compare total energy from two energy flow methods

a. In card file:

\begin{verbatim}
  EFLW 3  
  READ 'EFLOW'
\end{verbatim}

b. In user routine:
c--Energy for EF with Cal. only.
   ECAL=0.
   DO 10 I=KFCOT,KLCOT
   10 ECAL=ECAL+QE(I)

11.3.3 Calculate F - W moments from energy flow with calorimeters alone

a. In card file:

   EFLW 1
   READ 'EFLOW'

b. In user routine:

   DIMENSION FOXWOL(5)
   CALL QJOPTR('EF', '')
   CALL QJFOXW(FOXWOL, KRECO)
Chapter 12

Other ALPHA Physics Routines

12.1 dE/dx Analysis

CALL QDEDX(ITK,NHYP,RMASS,Q,R1,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:

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

**NHYP**
Number of hypotheses the user wishes to try.
If NHYP = 1, then RMASS, Q, R1, and SIGMA may be scalar variables.

**RMASS(nhys)**
Array of masses, one for each hypothesis.

**Q(nhys)**
Array of charges, one for each hypothesis.

Output arguments:

**RI**
The measured truncated mean ionization, normalized such that RI = 1 corresponds to minimum ionizing.

**NS**
Number of useful wire samples on the track. This is not filled in the case of MDST input.

**TL**
Useful length of the track (cm). This is not filled in the case of MDST input.

**RIEXP(nhys)**
Expected ionization for each mass hypothesis, normalized such that RIEXP = 1 corresponds to minimum ionizing.

**SIGMA(nhys)**
One standard deviation resolution error for each hypothesis. This is the expected dE/dx resolution, given NS, TL, RIEXP, and the momentum resolution. 
Note that one can calculate a chi-squared with 1 d.o.f. as: 
\[ \text{chi}^2 = ((\text{RI} - \text{RIEXP})/\text{SIGMA})^2 \]

**IER**
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

12.2 Photon conversions

CALL QPAIRF (I1,I2,DXY,DZ0,DZ2,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 on page 23 and 8.1.2 on page 32).

Input arguments:

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

Output arguments:

DXY  distance(cm) in the xy plane between the two tracks at the closest approach to the materialization point.
DZ0  Distance(cm) in z between the two tracks at the origin.
DZ2  The z separation of the tracks at the closest approach to the materialization point.
DTH  the theta difference of the two tracks.
RMA  the rho value at the materialization point.
ZMA  the z value at the materialization point.
XMA  The invariant mass of the tracks at the materialization point assuming they are both electrons.
NC1(2) 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.
DIN1(2) 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.
P(3)  Summed momentum of the two tracks at the materialization point in the order x,y,z.
IER  = 0 if calculation is successful; 1 otherwise.
Chapter 13

ALPHA Utility Routines: Printing, Writing Events, 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 on page 10). This routine can be called from user routines it is called automatically from QMEVNT if the COPY option (4.1.5 on page 12) 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. QWCLAS must be called before QWRITE. If QWCLAS is not called before QWRITE, the classification word will be set to zero.
13.4 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 Print time consumption

CALL QWTIME

Called automatically from QMTERM.

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 13.4 on page 65)
'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 13.4 on page 65). '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 Heal 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 Heal 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 13.4 on page 65)
Output see printer output when called with option 'H'.

13.5.8 Print all tracks or vertices in section

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
KSMCT MC particles
KSREV Reconstructed vertices
KSMCV MC vertices.

'option' one of '0', or '1' (see 13.4 on page 65).
'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 'I' (see 13.4 on page 65)

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 on page 38). 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.

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) : Translate "track" into "vertex" and "QVEC" into "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.

```fortran
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)
en
...
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 on page 75).
"F" + 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 on page 27): `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: \texttt{PMOD \textit{part} \textminus \textit{name} \textit{antipart} \textminus \textit{name} \ mass \ charge \ life \time \ width}

Parameters:
\textit{\texttt{\textit{part} \textminus \textit{name} \ antipart \textminus \textit{name}}} \\
see 4.9.1 on page 14. \\
The attributes of a particle and its antiparticle are modified at the same time. \\
If a particle is its own antiparticle, the same name has to be given twice.

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

Example:

\begin{verbatim}
PMOD 'GAMMA GAMMA ' 0.001
\end{verbatim}

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

\textbf{Mistake:}

\begin{verbatim}
PMOD 'PI+ PIO ' .14
\end{verbatim}

because \textit{pi+} and \textit{pio} 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

\begin{itemize}
  \item new table entries are created;
  \item a warning is issued;
  \item the program execution continues.
\end{itemize}

15.4.2 PNEW: New particles

Format: \texttt{PNEW \textit{part} \textminus \textit{name} \textit{antipart} \textminus \textit{name} \ mass \ charge \ life \time \ width}

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

\begin{itemize}
  \item PNEW causes a warning if the particles are already known;
  \item PMOD causes a warning if the particles are unknown;
\end{itemize}

program execution continues in either case.

15.4.3 PTRA: Modify particle names in the MC particle table

The PTRA 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:

\begin{itemize}
  \item Start with the integer code given for each generated particle.
\end{itemize}
• Get the corresponding particle name from the MC particle table.
• This name is relevant inside the ALPHA program.

**Format:**  
PTRA 'part – name antipart – name' iMCcode iMCanticode

**Parameters:**  
'part – name antipart – name'  
see 15.4.1 on page 72.  
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 QCPTTRA is equivalent to the PTRA card and can be called from QUNEWWR 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 on page 27.

**KPART ('part – name')**  
Integer particle code for 'part – name'

**CQPART (intg – code)**  
Particle name (12 characters; trailing characters filled with blank spaces)

**KPANTI ('part – name', IANTI)**  
If IANTI = 0: integer code for 'part – name'  
If IANTI unequal to 0: integer code for the antiparticle of  
'part – name'

**KCANTI (intg – code, IANTI)**  
...  
**QPMASS ('part – name')**  
nominal mass

**QCMASS (intg – code)**  
...  
**QPCHAR ('part – name')**  
charge  
**QCCHAR (intg – code)**  
...  
**QPLIFE ('part – name')**  
life time  
**QCLIFE (intg – code)**  
...  
**QPWIDTH ('part – name')**  
width  
**QCWIDTH (intg – code)**  
...

To check the particle names of ALPHA "tracks", see sections 8.1.6 on page 33 and 8.1.8 on page 34.
Appendix A

Program Structure

QMAIN
  |
  +---- QMINIT
    |
    | +-- QUBIOS  init BOS
    | +-- QUIHIS  init histogram package
    | +-- QMALPH  initialize ALPHA
    |
    +-- QINIT    user initialisation *

+-+---- QMREAD
  |
  | +-- AOPEN  open event input files
  | +-- AREAD  read next record
  |
  | +---- QNEWR called for every new run
  |
  | +-- QTERM  terminate job if eof or time limit or ...

++-- QMEVNT
  |
  | +-- QFILL  event preparation
  |
  | +--- QEVNT  event analysis *

called from anywhere:

QTERM
  |
  | +-- QTERM  user termination *
  |
  | +-- QUTHIS  output histograms

STOP  Fortran STOP

Arrows * indicate the important user routines.
Appendix B

Bank description

All banks described here must never be written to any 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  EEnergy
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 CClass
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
30  LK  I  QLITK flag
31  DB  F  Distance of closest approach to beam axis (if track pointing
to the main vertex) or to the corresponding secondary vertex.
32  ZB  F  Z coordinate of point where DB is measured.
```
Error Matrix

36-45 EM F triangular covariance matrix

46 CF F chi2 from last kinematical fit
47 EW F weight from energy flow analysis
48 ... 65 User space

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

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 TEXS
4 AT I offset for corresponding rows in bank TEXS (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 MGAD

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

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

1 number of words / vertex
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 BT B vertex BiTs (not yet defined)

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

Error Matrix (filled only on special request):

11-16 EM F triangular error matrix

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.

*-----*
| QBPT  | 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 PTRA 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**
  - ALPHAvsn FORTRAN K
    - Fortran code of all ALPHA subroutines and functions
  - ALCORvsn 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 on page 3)

- **HISTORIAN FILES**
  - ALPHAvsn OLDLIB K
    - Historian library
  - ALPHAvsn CORR K
    - Correction file
  - QUUSER INPUT K
    - Historian source for the routines QUINIT,QUEVNT,QUTERM (model routines which have to be filled)

- **INPUT TO THE LINK STEP**
  - ALCORvsn TEXT K
    - Correction file
  - ALPHAvsn TXTLIB K
    - ALPHA library
    - Other required libraries: ALEPHLIB, BOS77, CERNLIB

- **ALPHA CARD FILE (SAMPLE)**
  - ALPHACARDS K

- **ALPHARUN COMMAND FILE (SEE CH.**
  - ALPHARUN EXEC K
    - options are stored in: LASTING GLOBALV A

- **ALPHA NEWS**
  - ALPHAvsn NEWS K
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 QUINIT,QUEVNT,QUTERM (model routines which have to be filled)

  **[PHY.INC]QCDE.INC**
  **[PHY.INC]QMACRO.INC**
  Include files QCDE, QMACRO (see 3.1.2 on page 3)

- **HISTORIAN FILES**
  
  **PHY:ALPHA{vsn}.HLB**
  Historian library
  
  **PHY:ALPHA{vsn}.CORR**
  Correction file
  
  **PHY:QUUSER.INPUT**
  Historian source for the routines QUINIT,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, CERNLIB

- **ALPHA CARD FILE (SAMPLE)**

- **ALPHARUN COMMAND FILE (SEE CH. 2 on page 2)**
  
  **PHY:ALPHA.CARDS**
  
  **PHY:ALPHARUN.COM**
  options stored in ALPHA.OPT (by default)

- **ALPHA NEWS**
  
  **PHY:ALPHA{vsn}.NEWS**

C.3 ALPHA on the CRAY

The files needed to run ALPHA on the CRAY are on the CRAYXU 40i disk on CERNVM. They are (vsn = three digit ALPHA version number):
• **FORTRAN FILES**
  ALPHA\text{vsn} CRAYFOR
  ALCOR\text{vsn} CRAYFOR
  QUUSER CRAYFOR

• **ALPHA CARD FILE (SAMPLE)**
  ALPHA CARDS

• **SAMPLE JOB FILE**
  ALPRUN JOB
Appendix D

Standard particle table

<table>
<thead>
<tr>
<th>1 gamma</th>
<th>2 e+</th>
<th>3 e−</th>
<th>4 nu</th>
<th>5 mu+</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 mu−</td>
<td>7 p0</td>
<td>8 p+</td>
<td>9 p−</td>
<td>10 K0</td>
</tr>
<tr>
<td>11 K+</td>
<td>12 K−</td>
<td>13 n</td>
<td>14 p−</td>
<td>15 p#</td>
</tr>
<tr>
<td>16 K0s</td>
<td>17 eta</td>
<td>18 Lam0</td>
<td>19 Sig+</td>
<td>20 Sig0</td>
</tr>
<tr>
<td>21 Sig−</td>
<td>22 Xi0</td>
<td>23 Xi−</td>
<td>24 Ome−</td>
<td>25 n#</td>
</tr>
<tr>
<td>26 Lam#0</td>
<td>27 Sig#−</td>
<td>28 Sig#0</td>
<td>29 Sig#+</td>
<td>30 Xi0#</td>
</tr>
<tr>
<td>31 Xi#+</td>
<td>32 Ome#+</td>
<td>33 tau+</td>
<td>34 tau−</td>
<td>35 D+</td>
</tr>
<tr>
<td>36 D−</td>
<td>37 DO</td>
<td>38 D0−</td>
<td>39 D0+</td>
<td>40 D−</td>
</tr>
<tr>
<td>41 Lamc+</td>
<td>42 W+</td>
<td>43 W−</td>
<td>44 Z0</td>
<td>45 Deuteron</td>
</tr>
<tr>
<td>46 Tritium</td>
<td>47 Alpha</td>
<td>48 Geantino</td>
<td>49 GeantMino</td>
<td>50 GeantBino</td>
</tr>
<tr>
<td>51 GeantAbino</td>
<td>52 GeantPhino</td>
<td>53 Lamc−</td>
<td>54 Higgs</td>
<td>55 etab</td>
</tr>
<tr>
<td>56 K0</td>
<td>57 K0#</td>
<td>58 nue</td>
<td>59 nue#</td>
<td>60 numu</td>
</tr>
<tr>
<td>61 numu#</td>
<td>62 nutau</td>
<td>63 nutau#</td>
<td>64 a1+</td>
<td>65 a1−</td>
</tr>
<tr>
<td>66 a10</td>
<td>67 f0_975</td>
<td>68 a0_980</td>
<td>69 etapr</td>
<td>70 chic0</td>
</tr>
<tr>
<td>71 chic1</td>
<td>72 chic2</td>
<td>73 Jpsi</td>
<td>74 psipr</td>
<td>75 etac</td>
</tr>
<tr>
<td>76 rho+</td>
<td>77 rho−</td>
<td>78 K±</td>
<td>79 K±−</td>
<td>80 K±0</td>
</tr>
<tr>
<td>81 K±#0</td>
<td>82 D±0</td>
<td>83 D±#0</td>
<td>84 D±#</td>
<td>85 D±−</td>
</tr>
<tr>
<td>86 D±#±</td>
<td>87 D±#−</td>
<td>88 rho0</td>
<td>89 omega0</td>
<td>90 Phi</td>
</tr>
<tr>
<td>91 chib+</td>
<td>92 chib1</td>
<td>93 chib2</td>
<td>94 Upsilo0</td>
<td>95 Ups2s</td>
</tr>
<tr>
<td>96 etat</td>
<td>97 chit0</td>
<td>98 chit1</td>
<td>99 chit2</td>
<td>100 gluon</td>
</tr>
<tr>
<td>101 uquark</td>
<td>102 dquark</td>
<td>103 sqark</td>
<td>104 squark</td>
<td>105 bquark</td>
</tr>
<tr>
<td>106 tquark</td>
<td>107 tquark#</td>
<td>108 tquark#</td>
<td>109 squark#</td>
<td>110 cquark#</td>
</tr>
<tr>
<td>111 bquark#</td>
<td>112 tquark#</td>
<td>113 Theta</td>
<td>114 Thetapr</td>
<td>115 B−</td>
</tr>
<tr>
<td>116 B+</td>
<td>117 B0</td>
<td>118 B±0</td>
<td>119 Bs0</td>
<td>120 Bs0#</td>
</tr>
<tr>
<td>121 Bc−</td>
<td>122 Bc+</td>
<td>123 Bc±</td>
<td>124 Bc±−</td>
<td>125 Bc±0</td>
</tr>
<tr>
<td>126 Bc±0</td>
<td>127 Bs±0</td>
<td>128 Bs±0#</td>
<td>129 Bs±#</td>
<td>130 Bc±0</td>
</tr>
<tr>
<td>131 T0</td>
<td>132 T±0</td>
<td>133 T±</td>
<td>134 T±−</td>
<td>135 T±+</td>
</tr>
<tr>
<td>136 Ts−</td>
<td>137 Tc0</td>
<td>138 Tc±0</td>
<td>139 Tb+</td>
<td>140 Tb−</td>
</tr>
<tr>
<td>141 Tc±0</td>
<td>142 Tc±0#</td>
<td>143 Tc±#</td>
<td>144 Tc±#−</td>
<td>145 Tc±#+</td>
</tr>
<tr>
<td>146 Tc±#−</td>
<td>147 Tc±#0</td>
<td>148 Tc±#0#</td>
<td>149 Tc±#0#−</td>
<td>150 Tc±#0#+</td>
</tr>
<tr>
<td>151 Delta++</td>
<td>152 Delta#--</td>
<td>153 Delta+</td>
<td>154 Delta#--</td>
<td>155 Delta0</td>
</tr>
<tr>
<td>156 Delta#0</td>
<td>157 Delta−</td>
<td>158 Delta#−</td>
<td>159 Sig#±</td>
<td>160 Sig#--</td>
</tr>
<tr>
<td>161 Sig±0</td>
<td>162 Sig±0#</td>
<td>163 Sig±#</td>
<td>164 Sig±#−</td>
<td>165 Xi0</td>
</tr>
<tr>
<td>166 Xi±#0</td>
<td>167 Xi±#0#</td>
<td>168 Xi±#0#−</td>
<td>169 Sigc++</td>
<td>170 Sigc#--</td>
</tr>
<tr>
<td>171 Sigc+</td>
<td>172 Sigc−</td>
<td>173 Sigc0</td>
<td>174 Sigc#0</td>
<td>175 Xi±</td>
</tr>
<tr>
<td>176 Xic−</td>
<td>177 Xic0</td>
<td>178 Xic#0</td>
<td>179 Ome0</td>
<td>180 Ome±0</td>
</tr>
<tr>
<td>181 Xic#0</td>
<td>182 Xi0c−</td>
<td>183 Xi0c0</td>
<td>184 Xi0c#0</td>
<td>185 Sigc±++</td>
</tr>
<tr>
<td>186 Sigc±#−</td>
<td>187 Sigc±#0</td>
<td>188 Sigc±#0#</td>
<td>189 Sigc±#0#−</td>
<td>190 Sigc±#0#+</td>
</tr>
<tr>
<td>191 Xic±0</td>
<td>192 Xic±0#</td>
<td>193 Xic±0#−</td>
<td>194 Xic±0##</td>
<td>195 Ome±0</td>
</tr>
<tr>
<td>196 Ome±0#</td>
<td>197 Xicc++</td>
<td>198 Xicc#--</td>
<td>199 Xicc+</td>
<td>200 Xicc#--</td>
</tr>
<tr>
<td>201 Ome++</td>
<td>202 OmeCC#++</td>
<td>203 OmeCC#++</td>
<td>204 OmeCC#++</td>
<td>205 OmeCC#++</td>
</tr>
<tr>
<td>206 OmeCC#++</td>
<td>207 OmeCC#++#</td>
<td>208 OmeCC#++#</td>
<td>209 OmeCC++</td>
<td>210 OmeCC++</td>
</tr>
<tr>
<td>211 Sigb−</td>
<td>212 Sigb−#</td>
<td>213 Sigb0</td>
<td>214 Sigb#0</td>
<td>215 Sigb−</td>
</tr>
<tr>
<td>216 Sigb−#</td>
<td>217 Xi0b</td>
<td>218 Xi0b#</td>
<td>219 Xi#b</td>
<td>220 Xib#−</td>
</tr>
<tr>
<td>221 Omeb−</td>
<td>222 Omeb#−</td>
<td>223 Xibc+</td>
<td>224 Xibc#−</td>
<td>225 Xibc0</td>
</tr>
<tr>
<td>226 Xibc0#</td>
<td>227 Omebc0</td>
<td>228 Omebc0#</td>
<td>229 Omebc0#</td>
<td>230 Omebc0#</td>
</tr>
<tr>
<td>231 Xibb0</td>
<td>232 Xibb0#</td>
<td>233 Xibb−</td>
<td>234 Xibb#−</td>
<td>235 Omebb−</td>
</tr>
<tr>
<td>236 Omebb#−</td>
<td>237 Omebc0</td>
<td>238 Omebc0#</td>
<td>239 Sigt++</td>
<td>240 Sigt#--</td>
</tr>
<tr>
<td>241 Sigt++</td>
<td>242 Sigt#--</td>
<td>243 Sig0</td>
<td>244 Sig0#</td>
<td>245 Xit+</td>
</tr>
</tbody>
</table>
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 Lamb+ 272 Lamb#- 273 Xi0t+ 274 Xi0t-- 275 Xi0t0
276 Xi0t0#0 277 Xi0t0c0++ 278 Xi0t0c0-- 279 Xi0t0c+ 280 Xi0t0c#-
281 Ome0t0c+ 282 Ome0t0c-- 283 Sigm+a+ 284 Sigm+a- 285 Sigm+a0
286 Sigm+a#0 287 Sigm+a#- 288 Sigm+a#+ 289 Xi0b0 290 Xi0b+a+
291 Xib-- 292 Xib+a+ 293 Omeb+a 294 Omeb+a+ 295 Xibc+a+
296 Xibc+a- 297 Xibc+a0 298 Xibc+a#0 299 Omebc+a 300 Omebc+a#0
301 Omebcc+a+ 302 Omebcc+a#-- 303 Sigt+a++ 304 Sigt+a-- 305 Sigt+a+
306 Sigt+a- 307 Sigt+a0 308 Sigt+a#0 309 Xit+a+ 310 Xit+a--
311 Xit+a0 312 Xit+a#0 313 Omet+a0 314 Omet+a#0 315 Xitc+a++
316 Xitc+a-- 317 Xit+a+ 318 Xit+a#- 319 Omebc+a+ 320 Omebc+a#-
321 Ometcc+a++ 322 Ometcc+a-- 323 Xitb+a+ 324 Xitb+a-- 325 Xitb+a0
326 Xitb+a#0 327 Ometb+a0 328 Ometb+a#0 329 Ometbc+a+ 330 Ometbc+a#-
331 Ometbb+a#0 332 Ometbb+a#0 333 Xitb+a 334 Xitb+a- 335 Xitb0
336 Xitb0#0 337 Ometb+a0 338 Ometb+a#0 339 Ometbc+a 340 Ometbc+a#
341 Ometbb0 342 Ometbb+a0 343 Xi0tb+a 344 Xi0tb+a- 345 Xi0tb0
346 Xi0tb+a#0 347 Ome0tb0 348 Ome0tb+a0 349 Ome0tb+a# 350 Ome0tb+a#
351 Xibb#0 352 Xibb+a#0 353 Xibb+a- 354 Xibb+a#+ 355 Omebb+a-
356 Omebb+a#+ 357 Omebbc+a0 358 Omebbc+a#0 359 Omebb+a- 360 Omebb+a#
361 beame+ 362 beame- 363 Charged 364 Ecal 365 Hcal
366 CALobj 367 last_st_par
Appendix E

Index

An index of all Fortran variables, parameters, etc. can be found in the code of subroutine QDATA.

[c] Constant = Fortran parameter  
[f] Fortran function  
[r] Fortran subroutine  
[sf] Fortran statement function  
[v] Fortran variable/array stored in a COMMON block

actual mass  
add  
ALEPH file types  
ALLR  
ALPHA  
ALPHARUN  
angle  
antiparticle

of an individual particle: 8.1.1 on page 32  
vectors: 9.2.2 on page 39 and 10.3 on page 51  
4.1.1 on page 8  
parameter on FILO data card: 4.1.3 on page 10  
initialization in QMALTH called from QMINIT: 3.2 on page 4  
command file: 2 on page 2  
azimuth, polar angle: 9.1 on page 38  
decay angle: 9.1 on page 38  
access to antiparticles: 7.4.4 on page 27  
definition on data cards: 4.9.1 on page 14

batch job  
bending radius  
beta  
BMACRO  
boost  
BOS

see ALPHARUN: 2 on page 2  
of a reconstructed charged track: 8.2.1 on page 34  
see QBETA: 9.1 on page 38  
standard BOS statement functions: 3.1.2 on page 3  
Lorentz: 9.3 on page 46  
initialization in QUIBOS: 3.5.5 on page 7

c  
calorimeter objects  
CARDs  
charge  
charged tracks  
CLAS  
class  
COPY  
copy  
CQDATE  
CQFINP  
CQFOUT  
CQPART  
CQTIPN  
CQTIME  
CQVERS  
create  
cross product  
DAF

[c] speed of light: 6.1 on page 20  
7.1 on page 23 and 7.3 on page 25  
file type: 4.1.1 on page 8  
of an individual particle: 8.1.1 on page 32  
on particle table see QCCHAR / QPCHAR: 15.5 on page 73  
7.1 on page 23 and 7.3 on page 25  
Data card for use with event directories: 4.1.4.3 on page 12  
= vector class 7.4.1 on page 26 and 8.1.8 on page 34  
data card: 4.1.5 on page 12  
track vectors into other track vectors: 9.2.3 on page 40 and 9.2.8 on page 42  
track vectors into Fortran arrays: 9.2.6 on page 41  
Fortran arrays into track vectors: 9.2.10 on page 44  
[v] date at start of job: 6.3.6 on page 22  
[v] name of current input file: 6.3.6 on page 22  
[v] name of output file: 6.3.6 on page 22  
[f] particle name for a given integer code: 15.5 on page 73  
[f] track's particle name: 8.1.8 on page 34  
[v] time at start of job: 6.3.6 on page 22  
[v] ALPHA version: 6.3.6 on page 22  
new track vector: 9.2.7 on page 42  
QVCROS: 9.2.4 on page 40  
file type: direct access files: 4.1.1 on page 8
**data base** opened in QMINIT: 3.2 on page 4
**data cards** description 4 on page 8
**data set name** conventions 4.1.1 on page 8; examples 4.1.2.1 on page 9
**daughter particles** 7.5.1 on page 29 and 7.8 on page 31
**DEBU** data card: 4.4 on page 13
**debug** special VAX debugger features: 2 on page 2
**debug level** see KDEBUG: 6.3.2 on page 21
**decay angle** 9.1 on page 38
**dE/dx** 12.1 on page 62, 8.2.4 on page 35
**DHEA** bank: 6.2.2 on page 20
**dot product** 9.1 on page 38
**drop tracks** 9.2.5 on page 41
**DST unpacking** 4.5 on page 13

**e** constant: 6.1 on page 20
**Ecal objects** 7.1 on page 23 and 7.3 on page 25
**EDIR** event directory: 4.1.4 on page 11
**EFLW** energy flow data card: 11.1 on page 59
**ENDQ** BOS data card: 4 on page 8
**energy** see QE: 8.1.1 on page 32 / missing energy: 10.10 on page 56
**Energy flow** Ch. 11 on page 59
**EPIO** file type: machine independent input / output: 4.1.1 on page 8
**EVEH** bank: 6.2.1 on page 20
**event directories** 4.1.4 on page 11
**event input** see FILI data card: 4.1.2 on page 8
**event output** see FILO data card 4.1.3 on page 10 and routine QWRITE: 13.2 on page 64
**event processing** see QUEVNT: 3.3 on page 5

**FIEL** Data card to set magnetic field: 4.7 on page 14
**file types** = ALEPH file types: 4.1.1 on page 8
**FILI** data card: event input data set(s) 4.1.2 on page 8
**FILO** data card: event output data set 4.1.3 on page 10
**flags** user: 8.1.8 on page 34, 9.2.11 on page 45
**Fox – Wolfram** moments: 10.8 on page 54
**frame** access to Lorentz frames: 7.4.1 on page 26

**gamma** see QGAMMA: 9.1 on page 38

**h** constant 6.1 on page 20
**HAC parameters** bank offset: 3.1.3 on page 4
**hbar** constant: 6.1 on page 20
**HBOOK** initialization in QUIHIS: 3.5.3 on page 6
termination in QUTHIS: 3.5.4 on page 6

**Heal objects** 7.1 on page 23 and 7.3 on page 25
**hemispheres** see QJHEMI: 10.9 on page 55
**HIS** histogram file type 5.2.1 on page 18
**HIST** histogram file data card: 5.2.1 on page 18
**histograms** Ch. 5 on page 16
**histogram output** see 5.2.1 on page 18 and 5.2.2 on page 18
**Historian** Ch. 2 on page 2
**HTIT** data card: general histogram title 4.2.2 on page 12

**INCLUDE** Fortran statement: 3.1.2 on page 3
**initialization** see QMINIT and QUINIT: 3.2 on page 4
**invariant mass** 9.1 on page 38
KLRET [v] last reconstructed track: 7.1 on page 23
KLUNDS [sf] Lund status code: 8.1.8 on page 34
KLV0T [v] last particle pointing to V0: 7.1 on page 23
KMCxx [sf] bank MCAD = muon chambers: 8.2.7 on page 36
KMOPTH [sf] access to mother particle: 7.5.2 on page 29
KMTCH [sf] match MC – reco. tracks: 7.7 on page 30
KNAST [v] number of cal objects assoc. to a charged track: 7.1 on page 23
KNCALG [sf] number of associated charged tracks: 7.3 on page 25
KNCHT [v] number of charged tracks: 7.1 on page 23
KNCOT [v] number of cal objects: 7.1 on page 23
KNDAU [sf] number of daughters: 7.5.1 on page 29
KNDCT [v] number of decay tracks: 7.1 on page 23
KNEC [sf] number of associated Ecal objects: 7.3 on page 25
KNEFIL [v] number of events on current input file: 6.3.1 on page 21
KNEWJT [v] number of events on output file: 6.3.1 on page 21
KNEVT [v] number of events read in up to now: 6.3.1 on page 21
KNCHEL [sf] number of associated Heal objects: 7.3 on page 25
KNCHEL [v] number of isolated cal objects: 7.1 on page 23
KNJET [v] number of reconstructed jets: 7.1 on page 23
KNMC [v] number of MC particles: 7.1 on page 23
KMONO [sf] number of mother particles: 7.5.2 on page 29
KNMTC [sf] number of matching particles: 7.7 on page 30
KNOVT [v] number of overlap objects: 7.1 on page 23
KNREIN [v] number of records read from current input file: 6.3.1 on page 21
KREI [v] number of reconstructed tracks: 7.1 on page 23
KNTX [sf] number of TPC sectors for dE/dx: 8.2.4 on page 35
KNV0T [v] number of particle pointing to V0s: 7.1 on page 23
KORIV [sf] vertex at origin of track: 7.8 on page 31
KPART [f] integer code from particle name: 15.5 on page 73 and 7.4.3 on page 27
KPDIR [f] direct access to particles: 7.4.2 on page 26
KPDIRA [f] direct access to (anti)particles: 7.4.4 on page 27
KPEC [sf] bank PECO: 8.2.8 on page 36
KPEPXX [sf] bank PEPT: 8.2.9 on page 36
KPHCXX [sf] bank PHCO: 8.2.10 on page 36
KRUN [v] run number: 6.2.1 on page 20
KSAME [sf] access to same objects: 7.6 on page 30
KSMITH [sf] number of shared hits in match: 7.7 on page 30
KSTABC [sf] stability code: 8.1.5 on page 33
KSTATU [v] job status (init / event proc. / term): 6.3.2 on page 21
KTEX [sf] dE/dx bank TEXS: 8.2.4 on page 35
KTLX [f] Lorentz transformation: 9.3.1 on page 46
KTLXRI [f] Lorentz transformation: 9.3.2 on page 46
KTN [sf] Julia/Galeph track number: 8.1.8 on page 34
KTPCOD [sf] track’s particle code: 8.1.8 on page 34
KUCARD [v] log. unit for the card file: 6.3.4 on page 21
KUCRC2 [v] 2nd log. unit for card files: 6.3.4 on page 21
KUCONS [v] log.unit for the data base: 6.3.4 on page 21
KUINPU [v] log. unit for event input: 6.3.4 on page 21
KUOUTP [v] log. unit for event output: 6.3.4 on page 21
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KUPRNT</td>
<td>log. unit for the line printer output: 6.3.4 on page 21 and 6.3.4 on page 21</td>
</tr>
<tr>
<td>KUPTER</td>
<td>log. unit for the terminal: 6.3.4 on page 21 and 6.3.4 on page 21</td>
</tr>
<tr>
<td>KVBFILG</td>
<td>[sf] vertex bit flags</td>
</tr>
<tr>
<td>KVDAU</td>
<td>[sf] access to tracks from a vertex: 7.8 on page 31</td>
</tr>
<tr>
<td>KVINC7P</td>
<td>[sf] incoming particle to a vertex: 7.8 on page 31</td>
</tr>
<tr>
<td>KVN</td>
<td>[sf] Julia/Galeph vertex number: 8.3 on page 37</td>
</tr>
<tr>
<td>KVNDAU</td>
<td>[sf] number of outgoing tracks: 7.8 on page 31</td>
</tr>
<tr>
<td>KVNEW</td>
<td>[f] create new track vector: 9.2.7 on page 42</td>
</tr>
<tr>
<td>KVSAVE</td>
<td>[f] save track: 9.2.8 on page 42</td>
</tr>
<tr>
<td>KVSVC</td>
<td>[f] save track in specific class: 9.2.9 on page 43</td>
</tr>
<tr>
<td>KYV0XX</td>
<td>[sf] bank YV0V: 8.2.11 on page 37</td>
</tr>
</tbody>
</table>

**lifetime**
- on particle table: see QCLIFE / QPLIFE: 15.5 on page 73

**line printer**
- see KUPRNT

**lock**
- 10.2 on page 49

**log. units**
- 6.3.4 on page 21

**loops**
- over tracks (= vectors) and vertices: 7 on page 23

**Lorentz**
- transformations: 9.3 on page 46
- decay angle: 9.1 on page 38

**LUCLUS**
- jet finding algorithm: 10.11.3 on page 58

**main program**
- see QMAIN (2; 3)

**mass**
- actual mass of an individual particle: 8.1.1 on page 32
- invariant mass of a system of particles: 9.1 on page 38
- missing mass: 10.10 on page 56
- nominal mass on the particle table: 15.5 on page 73

**match**
- reconstructed tracks and MC particles: 7.7 on page 30

**MC flag**
- for an event: 6.3.3 on page 21

**MC particle**
- loops over ...: 7.1 on page 23 and 7.4 on page 26

**MC particle code**
- 15.1 on page 71

**MC particle table**
- 15.1 on page 71

**MINI flag**
- current event input = MINI: 6.3.3 on page 21

**missing**
- mass, energy, momentum: 10.10 on page 56

**momentum**
- of a particle see QP 8.1.1 on page 32 / missing momentum 10.10 on page 56

**Monte Carlo**
- see MC

**mother particle**
- 7.5.2 on page 29

**NATIVE**
- file type: machine - dependent input/output 4.1.1 on page 8

**NEVT**
- parameter on FILI / FILO data cards (q.v.)

**new track**
- KVNEW: 9.2.7 on page 42

**nominal mass**
- on particle table: 15.5 on page 73

**NOOV**
- parameter on FILO / HIST data cards (q.v.)

**NORU**
- parameter on FILO data card: 4.1.3 on page 10

**NOVX**
- ALPHA process cards: 4.6 on page 13

**Ntuples**
- 5 on page 16

**output**
- (events) see FILO card 4.1.3 on page 10 and routine QWRITE: 13.2 on page 64
- (histograms) see HIST data card: 5.2.1 on page 18

**parameters**
- HAC parameters: 3.1.3 on page 4
particle—antiparticle relation

7.4.4 on page 27

particle
attributes: 15.5 on page 73
direct access to specific particles: 7.4 on page 26

particle code
7.4.3 on page 27 and 15.2 on page 71
examples in 7.4.5 on page 28

particle systems
invariant mass: 9.1 on page 38

particle table
data cards: 15.4 on page 72
MC table: 15.1 on page 71 and 15.4 on page 72
standard table 15.4 on page 72

PAW
interactive editing of histograms and Ntuples: 5.2.1 on page 18

photon conversions
12.2 on page 62

pi
constant: 6.3.6 on page 22
Planck
constant: 6.3.6 on page 22

PMOD
data card: modify particle table 15.4.1 on page 72

PNEW
data card: new entry into particle table 15.4.2 on page 72

PTRA
data card: modify MC particle code translation: 15.4.3 on page 72

POT
unpacking: 4.5 on page 13

process
ALPHA process cards: 4.6 on page 13

pseudo—particle
10.3.3 on page 52

QBETA
[sf] beta of a particle: 9.1 on page 38

QBOOKN
[r] book Ntuples: 5.1.3 on page 16

QBOOKR
[r] book Ntuples with run— and event number: 5.1.4 on page 17

QBOOK1
[r] book 1—dimensional histograms: 5.1.1 on page 16

QBOOK2
[r] book 2—dimensional histograms: 5.1.2 on page 16

QCDE
macro: all parameters, commons etc.: 3.1.2 on page 3

QCDESHE
short subset of QCDE

QCFxxx
macros containing statement functions

QCH
[sf] track’s charge: 8.1.1 on page 32

QCOSA
[sf] cos (angle between two tracks): 9.1 on page 38

QCT
[sf] cos (theta): 9.1 on page 38

QDATA
[r] (quasi) block data

QDB
[sf] distance to beam axis: 8.1.4 on page 33

QDBS2
[sf] error••2 on QDBS2: 8.1.4 on page 33

QDECAN
[f] decay angle: 9.1 on page 38

QDECA2
[f] decay angle: 9.1 on page 38

QDHEXx
[v] header bank DHEA: 6.2.2 on page 20

QDMSQ
[sf] mass difference ••2: 9.1 on page 38

QDOT3
[sf] dot product (3—vector): 9.1 on page 38

QDOT4
[sf] dot product (4—vector): 9.1 on page 38

QE
[sf] energy: 8.1.1 on page 32

QEIDxx
[sf] bank EIDT = electron identification: 8.2.5 on page 35.

QELEP
[v] LEP energy: 6.2.1 on page 20

QFRFXX
[sf] bank FRFT = track fit: 8.2.1 on page 34

QFRXX
[sf] bank FRID: 8.2.3 on page 35

QFRTXX
[sf] bank FRTL = appendix to FRFT: 8.2.2 on page 35

QGAMMA
[sf] particle’s gamma: 9.1 on page 38

QHFN
[r] fill Ntuple: 5.1.6 on page 17
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Page</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>QHFNR</td>
<td>[r] fill Ntuple with run - and event number: 5.1.6 on page 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QHFR</td>
<td>[r] fill Ntuple with run - and event number: 5.1.5 on page 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QHMAxx</td>
<td>[sf] bank HMAD = Hcal - muon association: 8.2.6 on page 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJADDP</td>
<td>add 4 - vectors: 10.3 on page 51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJEIG</td>
<td>eigenvalues of mom. tensor: 10.4 on page 53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QFOXW</td>
<td>Fox – Wolfram moments: 10.8 on page 54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJHEMI</td>
<td>divide the event into two hemispheres: 10.9 on page 55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJMISQ</td>
<td>missing energy, mass, and momentum: 10.10 on page 56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJMDCL</td>
<td>jet finding – scaled minimum distance algorithm: 10.11.2 on page 57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJMMLCL</td>
<td>jet finding – scaled invariant mass sq. algorithm: 10.11.1 on page 56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJLUCL</td>
<td>jet finding – LUCLUS: 10.11.3 on page 58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJSPH</td>
<td>sphericity: 10.6 on page 53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJTHRU</td>
<td>thrust value / axis: 10.7 on page 54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QKEVxx</td>
<td>[v] bank KEVH: 6.2.3 on page 21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLITK</td>
<td>[r] lock individual track: 10.2.1 on page 49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLOCK</td>
<td>[r] lock track family: 10.2.3 on page 50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLOCK2</td>
<td>[r] lock track family: 10.2.6 on page 51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLEVR</td>
<td>[r] reverse lock: 10.2.5 on page 51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLEVR2</td>
<td>[r] reverse lock: 10.2.6 on page 51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLUITK</td>
<td>[r] unlock individual track: 10.2.2 on page 49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLZER</td>
<td>[r] zero lock: 10.2.4 on page 50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLZER2</td>
<td>[r] zero lock: 10.2.6 on page 51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QM</td>
<td>[sf] particle's mass: 8.1.1 on page 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMACRO</td>
<td>macro: statement functions: 3.1.2 on page 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMAIN</td>
<td>ALPHA main program (2; 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMASV0</td>
<td>V0 mass: 8.1.2 on page 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMCxxx</td>
<td>bank MCAD = muon chambers: 8.2.7 on page 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMFID</td>
<td>[v] ALEPH magnetic field: 6.2.1 on page 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMINIT</td>
<td>[r] system initialization: 3.2 on page 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMSQ2, QMSQ3, QMSQ4</td>
<td>[sf] invariant mass**2: 9.1 on page 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMTER</td>
<td>[r] system termination: 3.4 on page 5 and 13.1 on page 64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QM2, QM3, QM4</td>
<td>[sf] invariant mass: 9.1 on page 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QNTEX</td>
<td>[sf] number of sectors for dE/dx: 8.2.4 on page 35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QP</td>
<td>[sf] momentum: 8.1.1 on page 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPCHAR</td>
<td>[f] particle table charge: 15.5 on page 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPECxx</td>
<td>[sf] bank PECO: 8.2.8 on page 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPEPxx</td>
<td>[sf] bank PEPT: 8.2.9 on page 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPHCxx</td>
<td>[sf] bank PHCO: 8.2.10 on page 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPH</td>
<td>[sf] track's azimuth: 9.1 on page 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPLIFE</td>
<td>[f] particle table life time: 15.5 on page 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPMSS</td>
<td>[f] particle table mass: 15.5 on page 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPPAR</td>
<td>[sf] momentum parallel to a vector: 9.1 on page 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPPER</td>
<td>[sf] momentum perpendicular to a vector: 9.1 on page 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPT</td>
<td>[sf] transverse momentum: 9.1 on page 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPWIDT</td>
<td>[f] particle table width: 15.5 on page 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QQC</td>
<td>[c] speed of light: 6.3.6 on page 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QQE</td>
<td>[c] e: 6.3.6 on page 22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
QQH  [c] hbar: 6.3.6 on page 22
QQHC [c] hbar * c 6.3.6 on page 22
QQIRP [c] factor between inv. bending radius and momentum: 6.3.6 on page 22
QQPI [c] \( \pi: 6.3.6 \) on page 22
QQPIH [c] \( \pi / 2: 6.3.6 \) on page 22
QQRADP [c] 360 / \( \pi: 6.3.6 \) on page 22
QQ2PI [c] 2 \( \pi: 6.3.6 \) on page 22

QSIG  [sf] track’s error matrix: 8.1.3 on page 32
QSIGE, QSIGEE, QSIGM, QSIGMM, QSIGP, QSIGPP
  [sf] track’s error matrix: 8.1.3 on page 32
QSUSTR  [r] allocate user’s track space 14.1.1 on page 69
QSUSVX  [r] allocate user’s vertex space 14.1.2 on page 69

QTCLAS  [r] Lorentz transformation: 9.3.3 on page 47
QTEXxx  [sf] bank TEXS = dE/dx: 8.2.4 on page 35
QTIME  [v] as given on the TIME data card: 6.3.5 on page 22
QTIMEL  [v] remaining job time: 6.3.5 on page 22

QUEVNT  [r] event processing user routine: 3.3 on page 5
QUIBOS  [r] initialise BOS: 3.5.5 on page 7
QUIHIS  [r] initialise histograms: 3.5.3 on page 6
QUNIT  [r] user initialization routine: 3.2 on page 4
QUNEW  [r] user routine: called for every new run: 3.5.1 on page 5
QUSEC  [r] special records: 3.5.2 on page 6
QUTERM  [r] user termination routine: 3.4 on page 5
QUTHIS  [r] terminate histograms: 3.5.4 on page 6

QVADD2, QVADD3, QVADD4, QVADDN  
  [r] add track vectors: 9.2.2 on page 39
QVCOPY  [r] copy track vectors: 9.2.3 on page 40
QVCROS  [r] cross product: 9.2.4 on page 40
QVDROP  [r] drop tracks: 9.2.5 on page 41
QVEM  [sf] vertex error matrix: 8.3 on page 37
QVGTS  [r] copy error matrix into Fortran array: 9.2.6 on page 41
QVGTS3, QVGTS4  [r] copy track vector into Fortran array: 9.2.6 on page 41
QVSCAL  [r] scale track momentum: 9.2.10 on page 44
QVSETM  [r] set mass of a track: 9.2.10 on page 44
QVSETS  [r] copy Fortran array into error matrix: 9.2.10 on page 44
QVSET3, QVSET4  [r] copy Fortran array into track vector: 9.2.10 on page 44
QVSB  [r] subtract track vectors: 9.2.12 on page 45
QVX, QVY, QV  [sf] vertex position: 8.3 on page 37
QVZERO  [r] zero track vector: 9.2.13 on page 45

QWEVNT  [r] print whole event: 13.5.3 on page 65
QWHEREAD [r] print event header: 13.5.4 on page 66
QWITK  [r] print individual track(s): 13.5.6 on page 66
QWIVX  [r] print individual vertices: 13.5.7 on page 67
QWMESS  [r] message routine: 13.5.1 on page 65
QWMESE  [r] message routine: 13.5.2 on page 65
QWRITE  [r] event output routine: 13.2 on page 64
QWSEC  [r] print section of tracks/vertices: 13.5.8 on page 67
QWTIME  [r] print time consumption: 13.5 on page 65
QWTREE  [r] print decay chain tree: 13.5.9 on page 67
QX  [sf] x-momentum: 8.1.1 on page 32
QY [sf] y—momentum: 8.1.1 on page 32
QZ [sf] z—momentum: 8.1.1 on page 32
QZB [sf] z—distance to interaction point: 8.1.4 on page 33
QZBS2 [sf] error••2 on QZB: 8.1.4 on page 33
QYV0xx [sf] bank YV0V: 8.2.11 on page 37

READ data card: read cards from several card files: 4.3 on page 13
run selection 4.1.2.2 on page 10

same objects in diff. Lorentz frames, with diff. hypotheses: 7.6 on page 30
same two particles based on the same object? see XSAME: 8.1.7 on page 34
save tracks KVSAVE; KVSAVC: 9.2.9 on page 43 and 9.2.8 on page 42
scale momentum QVSCAL: 9.2.10 on page 44
sections in vector / vertex banks: 7.2 on page 25 and 13.5.8 on page 67
user vector / vertex sections: 9.2.7 on page 42
selection see run/event selection: 4.1.2.2 on page 10
SELR parameter on FILO data card 4.1.3 on page 10
set mass QVSETM: 9.2.10 on page 44
SEVT data card: select events 4.1.2.2 on page 10
slow control read s.c. records: 3.5.2 on page 6
speed of light constant: 6.3.6 on page 22
sphericity 10.6 on page 53, 10.4 on page 53
SRUN data card: select runs 4.1.2.2 on page 10
start ALPHA interactively or in batch: 2 on page 2
STOP Fortran statement: forbidden: 3.4 on page 5
submit a job Ch. 2 on page 2
subtract track vectors: 9.2.12 on page 45
SYNT data card: indicates a syntax check run 4.12 on page 15

tapes 4.1.2 on page 8
terminal output see KUPER
thrust 10.7 on page 54
TIME data card: time to terminate the job properly 4.8 on page 14
time remaining job time: see QTIMEL 6.3.5 on page 22
title general title for HBOOK histograms: 5.2.3 on page 19
topology 10 on page 48
track class 7.4.1 on page 26
track number Ch. 7 on page 23

unpack POT/DST unpacking: 4.5 on page 13
UNPK data card: control POT/DST unpacking
UPDA parameter on HIST data card 5.2.1 on page 18
unit log. input / output units: 6.3.4 on page 21
units ALEPH phys. unit system: 6 on page 20
user routines Ch. 3 on page 3
user track / vertex sections 9.2.7 on page 42

vector synonym for "track" or "particle"
vector class 7.4.1 on page 26 and 8.1.8 on page 34
vector number see track or vertex number
vector operations 9.2 on page 38
vertex number Ch. 7 on page 23
V0 mass 8.1.2 on page 32
width see particle table: 15.5 on page 73
write events see FILO data card 4.1.3 on page 10 and QWRITE: 13.2 on page 64
write on line printer, terminal
                      Ch. 2 on page 2

XCEQAN, XCEQOR, XCEQU
           [sf] check particle name: 8.1.6 on page 33
XEID [sf] does EIDT exist? 8.2.5 on page 35
XFRF [sf] do FRFT and FRTL exist? 8.2.1 on page 34
XHMA [sf] does HMAD exist? 8.2.6 on page 36
XLOCK [sf] locked? 10.2 on page 49
XMC [sf] MC particle? 8.1.8 on page 34
XMCA [sf] does MCAD exist? 8.2.7 on page 36
XMCEV [v] MC event? 6.3.3 on page 21
XMINI [v] event input MINI? 6.3.3 on page 21
XPEQAN,XPEQOR,XPEQU
           [f] test particle name: 8.1.6 on page 33
XSAME [sf] based on the same object? 8.1.7 on page 34
XTEX [sf] does TEXS exist? 8.2.4 on page 35

zero track vectors: 9.2.13 on page 45