ALEPH 99-087
SOFTWR 99-001
J. Boucrot
January 31, 2000

ALPHA User's Guide

ALPHA

ALEPH PHYSICS ANALYSIS PACKAGE

Versions $\geq$ 125

Authors:

H.Albrecht , E.Blucher and J. Boucrot

With contributions from:

A.Belk, B.Bloch−Devaux, C.Bowdery, D.Brown, J.Carr, D.Casper, D.Cinabro,G.Dissertori, R. Forty, P.Gay, G.Graefe, R.Hagelberg, S.Haywood, J.Hilgart, R. Jacobsen, P.Janot, R.Johnson, E.Lançon, J.P.Lees, M.N.Minard, H.G.Moser, J.Nash, M.Pepe−Altarelli, P.Perez, F.Ranjard, M.Scarr, D.Schlatter, O.Schneider, H.Seywerd, M.Talby, G.Taylor, P. Teixeira-Dias, I.Tomalin, S.Wasserbaech, J.Wear, and T.Wildish

Contents

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 files (POTs), and of DSTs and MINIs. All event input/output is done by ALPHA − the user has to provide only the name(s) of the input/output data set(s). ALPHA also provides easy access to physical variables (e.g., momentum, energy), so the user can write physics analysis programs without detailed knowledge of the ALEPH data structure (tabular BOS banks). An extensive set of utility routines (e.g., kinematics, event shape, secondary vertex finding, b-tagging, MINI-writing, etc.) is available as part of the ALPHA package.

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

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

ALPHA has grown up with time , several general utility packages coming from UPHY being incorporated as they become more popular and tested :

ALPHA 114	in September 1992 : QMUIDO, interfaces to YTOP
ALPHA 115	in May 1993 : ENFLW, SLUMOK, PGPC photons
ALPHA 116	in December 1993 : GET_BP, QFNDIP, QIPBTAG
ALPHA 117	in May 1994 : QPI0DO, QVSRCH, NANO reading
ALPHA 118	in October 1994 : QTRUTH
ALPHA 120	in March 1995 : QSELEP, MINI 10 reading, PGAC photons
ALPHA 121	in October 1995 : NANO writing, new beam spot
ALPHA 122	in May 1996 : new matching, CVS code manager
ALPHA 123	in October 1998 : QPCORR, LEP BOMs, new QELEP
ALPHA 124	in May 1999 : MINI writing, new QIPBTAG, QDDX

Chapter 2

Getting Started

Two files must be provided to run an ALPHA job:

1. A file which contains the Fortran or CVS code for the user subroutines (see Ch. 3).
2. A card file which defines the input / output data files, and allows the user to select several optional parameters (see Ch. 4).

The libraries needed to link the program are described in Appendix C. To run ALPHA, the following Fortran units must be assigned:[1]

unit	
6	Print output file
7	Card file
76	(optional): in an interactive session, this unit may be assigned to the terminal. Short messages will be sent to the terminal and long listings sent to the output file.

A command file, **alpharun**, described in Appendix C is available on all ALEPH computers to make these file assignments, and to perform the following tasks:

1. compile and link the Fortran (or CVS) code;
2. link with all additional needed libraries (BOS, ALEPHLIB, JULIA, MINI);
3. run the program interactively or submit a BATCH job.

On the DEC/VMS machines, **alpharun** also facilitates the use of a set of DEC debugger command files which simplify ALPHA program debugging.

[1]Other units used by ALPHA are listed in Section 6.7.4; Fortran units 90, 91, 92, and 93 are always free for private output files.

Chapter 3

User routines

In this chapter, ALPHA routines which are intended to be modified by the user are described.

Normally, only three routines have to be provided by the user: initialization (QUINIT), event analysis (QUEVNT), and program termination (QUTERM). Models for these three subroutines are available (see Appendix C). Other subroutines which may be modified by the user are also described in this chapter.

User routines can be provided either as a plain Fortran file or as a CVS source file; the **alpharun** command file described in Appendix C supports both options.

For all user routines, default versions exist in the ALPHA library which are loaded automatically if no user code is given.

3.1 General Comments

3.1.1 Name conventions

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

Q subroutines; real functions, variables, or arrays

K integer functions, variables, or arrays

X logical functions, variables, or arrays

C character functions, variables, or arrays (always in combination with Q as 2nd character).

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

3.1.2 Including ALPHA features in Fortran code

In addition to subroutines, the ALPHA package contains include files which have to be included at the beginning of user subroutines or functions. There are several such includes, three of them being the following:

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).

QDECL Contains all integer/real declarations of all ALPHA internal variables, to be put in the code by people intending to use IMPLICIT NONE.

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

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

Fortran includes for DEC machines:

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

Fortran includes for UNIX Machines:

```
INCLUDE '$ALEPH/phy/qcde.inc'
INCLUDE '$ALEPH/phy/qmacro.inc'
```

CVS statements for CVS source files, on all machines:

```
#include "qdecl.h"
#include "qcde.h"
#include "qmacro.h"
```

Important! The following sequence of statements must be observed:

1. SUBROUTINE or FUNCTION statement
2. QDECL, if you need it (e.g.if you want to use IMPLICIT NONE)
3. QCDE
4. your own variable declarations, COMMONs, DIMENSIONs, etc.
5. your own DATA statements
6. QMACRO
7. your own statement function definitions (if any)
8. your executable Fortran statements

If you don't do so, you may end up with specially unintelligible Fortran compiler diagnostics...

3.1.3 "HAC" parameters

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

A separate include file QHAC is provided for the HAC parameters of all banks appearing on POT/DST/MINI event files which are NOT available in QCDE. QHAC contains only HAC parameters of banks needed by ALPHA and which are official ALEPH banks maintained in the standard documentation system. It can be included in the same way as QCDE and may be used in conjunction with it.

3.1.4 Implicit None

The include file QDECL must be used by people wishing to use IMPLICIT NONE in their program.

3.1.5 Booking of private BOS banks in ALPHA

To be faster, ALPHA stores internally a lot of BOS bank indices in the ALPHA commons. Therefore one should absolutely avoid any BOS garbage collection during the execution of ALPHA since the program would crash immediately.

If you need to book your own BOS banks: use the BOS routines NBANK, MBANK, WBANK. One should absolutely avoid the use of the ALEPHLIB routine AUBOS since it performs a BOS garbage collection if the space is too small in the BOS array. Of course, the use of the BOS routines BGARB or WGARB in ALPHA is absolutely forbidden.

3.2 User Initialization

SUBROUTINE QUINIT

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

- Initialization of BOS (default : 1500,000 words working space)
- Initialization of HBOOK (default : 100,000 words working space)
- Reading special files containing constants and calibrations
- Reading the user's 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 3.5.3 and 3.5.5.

3.3 Event analysis routine

SUBROUTINE QUEVNT(QT,KT,QV,KV)

QUEVNT is called once for each event.

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

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

IMPORTANT: Do NOT perform a BOS garbage collection (CALL BGARB(IW)) in QUEVNT or in any routine called by QUEVNT: this would cause ALPHA to crash irrecoverably. If you must use private BOS banks, you have to book them using the BOS routines NBANK,MBANK or WBANK. DO NOT use the AUBOS routine from the ALEPHLIB, which performs an automatic BOS garbage collection when no more space is available.

3.4 User termination routine

SUBROUTINE QUTERM

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

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

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

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

3.5 Other User Subroutines

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

3.5.1 New Run

SUBROUTINE 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 may be used to initialize run−dependent data or to print run statistics.

Input arguments

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

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

Default no action: RETURN.

3.5.2 Unkown Record Type

SUBROUTINE QUSREC

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

Default: no action: RETURN.

3.5.3 Initialize the histogram package

SUBROUTINE QUIHIS

NOT intended for histogram booking (use QUINIT).

- Called automatically from QMINIT.

Default: Initialize HBOOK : CALL HLIMIT (100000).

Warnings: If the above size of the HBOOK array is too small, which can happen with a very large number of scatter plots and/or Ntuples, one can increase it by modifying the parameter LQPAW in the QUIHIS routine. Users who want to write on output file *very* large NTUPLES (more than 16 Mbytes) will have to give appropriate NREC and/or RECL parameters in their HIST data card (see 5.2.1 on p. 31).

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 1500,000 words working space.

If you need more space, you have to edit the source code of this routine (see below where to find it), to increase the value of the PARAMETER LQBOS to the desired value, and to put the modified QUIBOS routine in your CVS files or into your FORTRAN user file.

The CVS source code of QUIBOS can be found in the following files:

on the UNIX machines in $ALROOT/alphavsn/pack/quibos.F

(vsn is the ALPHA version number, e.g. 124)

on the ALWS cluster in phy:quibos.F

Chapter 4

Data Cards

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

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

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

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

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

4.1 Input/Output

4.1.1 ALEPH file types

There are several ALEPH file types:

NATIVE	machine−dependent input/output (RAW data are written in VAX NATIVE format)
EPIO	machine−independent input/output (all official ALEPH datasets other than RAW data)
EDIR	event directories (for fast access to data using the CLAS data card)

DAF	direct access files (e.g., ADBSCONS 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 the extension part of the data set name.

Examples:

On DEC/VMS computers:

```
MYFILE.EPIO
```

On UNIX:

```
myfile.epio
```

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

4.1.2 FILI: Input datasets

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

Any number of FILI cards may be given − the data sets are read in the order the cards are given. Different file formats (e.g., NATIVE, EPIO) and data from POT, DST, and MINI can be processed in the same job.

The program SCANBOOK should be used to create FILI cards with the correct syntax.

How to specify file or EDIR names on FILI cards: Examples

Disk files:

```
            DEC / VMS                           UNIX

1  FILI 'HADR.NATIVE'              native files not recommended
2  FILI 'HADR.EPIO | EPIO'         FILI '{cwd}/hadr.epio'
3  FILI 'HADR.DATA | EPIO'         FILI '{cwd}/hadr.data | EPIO'
4  FILI 'AL$EDIR:AB1234.EDIR'      FILI '/aleph/edir/ab1234.edir'
```

Explanation:

1. Complete specification. " .NATIVE" defines the file format.
2. "| EPIO" can be omitted here because the format is already specified in the data set name. The vertical bar separates the file name from the parameters.
3. "DATA" is non−standard and not recommended. In such a case, the format must be given as a parameter: " | EPIO". Please not that this EPIO parameter must be UPPER CASE, even on UNIX machines.

How to specify tapes (cartridges, DLTs, Redwoods,..)

The same format can be used for DEC/VMS and UNIX:

```
FILI 'ALDATA | EPIO | CART AS2010.25.AL -s nnn'
```

Here, ALDATA is the data set name, AS2010 is the cartridge VID, 25 is the file sequence number of the dataset on the cartridge, and AL denotes an ANSI labelled tape.

At CERN and in some Homelab centers, the corresponding tape will not be read directly by ALPHA: the system will first stage the specified file (or several files in one go) on disk; then ALPHA will read this disk file.

At CERN only the additional parameter -s nnn must be added at the end of a FILI CARD to ask for the staging of a file with size nnn greater than 590 Mbytes. This is needed for large files which otherwise will be truncated by the stager, resulting in a BOS reading error.

Don't forget that the best solution to have access to cartridges or DLTs, either real data or Monte Carlo data with or without EDIRs, is to use the FILI CARDS generated by the SCANBOOK interactive facility. SCANBOOK generates the FILI cards with the correct syntax for the datasets you want to analyse. Moreover if needed they will be generated with the exact -s nnn parameter corresponding to the real size of each dataset.

stagelist: query on staged datasets on CERN UNIX computers

An interactive facilty called **stagelist** is available on CERN UNIX computers to know which datasets are presently staged. This may be very useful to make tests, in order to avoid the staging time which may be very long.

At most 3 arguments may be provided to **stagelist** to select datasets, e.g.:

stagelist MINI 1994 tau will give the list of staged Monte Carlo MINIs with 1994 geometry containing tau lepton simulations

stagelist POT 1996 DA will give the list of staged POTs of 1996 real data.

4.1.3 Run / event selection

The following cards may be used to select particular runs or events for analysis. These data cards are standard from BOS, none of them being specific to ALPHA.

SEVT 15 2 4 6 8 −11 Select EVenTs 2,4,6,8,9,10,11 of run 15 The 1st number is a run number, the following ones are event numbers. Negative numbers define a range of events. It is possible to include several SEVT cards in a card file, but only one SEVT card can be given for each run. The SEVT card, as well as the SRUN card described below, will work if the input files are ordered to have increasing run/event numbers. If the input files are not in sequential order, the selection cards will work correctly only if the NSEQ card (see below) is included in the card file.

SRUN 2 −4 6 8 −10 Select RUNs 2,3,4,6,8,9,10. See note under SEVT on sequential order of runs.

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

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

NEVT 3 Select the 1st, 2nd, and 3rd events. More than two numbers are not allowed on this card. Be careful that this card selects a total number of events READ by BOS on the input stream if there is no CLAS card (see below) , and the number of events SELECTED if there is a CLAS nn card .

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

NWRT 1000 This card allows to write only the first 1000 selected events on output tape . More that 1 number is not allowed on this card .

Remark: There is no standard card to skip a single event (no “IEVT” data card). Skipping an event must be done either through a SEVT card or through a modification of the QMRDSB subroutine (reading routine of ALPHA).

4.1.4 FILO : Output files

Event output is controlled by the FILO card and by the subroutine QWRITE (see 13.2). The data set name and options are given on the FILO card. Calling QWRITE writes the current event to the output file. The COPY card (see 4.1.7) may also be used to write events to a file. If a FILO card is given, all run records will be written out by default (see ALLR, NORU, and SELR below).

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

data set name same as on FILI cards; see examples in 4.1.2.

File format NATIVE, EPIO, or EDIR

parameters: (optional)

label if the output is on cartridge or DLT, will be the magnetic label for this tape. Usually set as ALDATA for POTs,DSTs,MINIs and to KINDATA for KINGAl outputs.

ALLR write all run records to the output file (Default when writing output POTs/DSTs). Must be used if you intend to read the output file/EDIR with the QFND option for real data of 1993 and after (QFNDIP needs special banks from the run header).

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 (default when writing a MINI). 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. DO NOT use this option if you write an output EDIR !

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−minded protection against involuntarily overwriting data sets. If this parameter is given AND the output data set already exists, the program will stop. Note that problems with overwriting do not arise on the VAX.

DISP The DISPose option ensures that your EPIO output file will be sent back to the UNIX platform from which you have sent your ALPHA batch job.

Examples: *FILO 'abc.native | SELR | NATIVE | NOOV'* The 2nd "NATIVE" is redundant; see 4.1.2.

FILO 'mydata.epio | EPIO | DISP' The output EPIO file will be sent back to the computer where you run alpharun.

FILO 'ALDATA | EPIO | CART AT1234.45.AL' The output EPIO file will be written on file 45 of the DLT tape AT1234 with ANSI label ALDATA. This is valid at CERN only. Please see with your system administrator if this syntax for tapes/DLTs works on your computer, if you are working in a Homelab.

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

The output event type (POT, DST, MINI – see 4.1.1) is the same as the input event type unless the MINI card is given (see below). Event directories can be created from any input event type (see 4.1.5).

MINI: Select Mini-DST for output file

If the MINI card is given, the output file specified with the FILO card will be written in Mini-DST format; see Appendix D and the Mini-DST User's Guide for details. The user must put a CALL QWRITE statement (13.2 on p. 130) in his code to write out the events. To be used only for private productions.

Beware: the banks written by default on output are the LEP2-type ones (see Appendix D).

MINP: Write Mini-DST with official content

If the MINP card is given, the output file specified with the FILO card will be written in the same Mini-DST format as official ALEPH MINI productions. All important ALPHA packages used to build the MINI are automatically initialised with the right constants (especially ENFLW, QIPBTAG and QSELEP). The event writing is done for ALL events, the user must NOT have a CALL QWRITE in his code, otherwise the events will be written twice.

Official ALEPH MINIs are done using this MINP data card and reading POTs, not DSTs, in order to have the informations needed to build the pad dE/dx information with the good calibration. For that purpose, the MINP card sets automatically the unpacking (see 4.3 on p. 17) to UNPK 'AL SO' .

With the MINP card, the programs recognises which kind of input is used and writes the right kind of MINI for LEP1 or LEP2 data/ Monte Carlo (see Appendix D).

COMP: Data compression

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

COMP 'NONE' suppresses the compression.

NWRT: Number of events to write out

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

4.1.5 Event Directories

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

Creating Event Directories

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

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

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

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

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

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

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

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

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

Reading data with event directories

The event directory must be specified in the FILI card:

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

All of the run / event selection cards (Sec. 4.1.3) can be used with event directories. If the CLAS card (described below) is given in the card file, only events with certain classification words will be read from the input file.

CLAS: Select events with certain classification word

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

It is also possible to make more complicated selections based on the event classification word by supplying a new version of the routine BSELEC. This routine should be extracted from the BOS77 library and modified. The default version of BSELEC, shown below, checks to see if a MASK has been supplied with the CLAS card. If so, it checks to see if the event classification word IWORD and MASK have any bits in common. Events are read in only if BSELEC is .TRUE. The line KCLASW=IWORD should not be changed; this line allows access to the event directory classification word inside of ALPHA (*e.g.*, inside QUEVNT).

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

4.1.6 FDBA: Select an ADBSCONS database

In the absence of any FDBA card, the officially released ALEPH database (with name adbscons.daf) is used by default in ALPHA.

The FDBA card directs ALPHA to use a database which is not the default one. This may be useful to use the database under test and not yet officially released, or to use an old version.

On VAX/VMS: *FDBA 'dbase:name.daf'*

On UNIX: *FDBA '$ALEPH/dbase/name.daf'*

If name = adbstest, the current test database under developement will be selected.

To get an old version: give explicitely its name , e.g. adbs217.

Important remark:

In order to analyze 1989/1990 data, it is **mandatory** to ask for a special database with name = adbs8990.

4.1.7 COPY: Copying events

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

Format *COPY* (no parameters)

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

4.2 ALPHA Process cards

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

NOMC no Monte Carlo "truth"

NOCH no CHarged tracks (also excludes V0s)

NOEM no Error Matrix for charged tracks . Be careful : if you use this card , you cannot use any fitting routine (KVFITx) , and you cannot use QIPBTAG .

NOV0 no V0s . Do no use it if you call QIPBTAG !

NOCO no CalOrimeters

NOPC no NEutral OBjects (from PCPA)

NOGA no GAmpec objects (from EGPC or PGPC or PGAC) . Do not use it if you need QPI0DO !

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.

4.3 UNPK: POT / DST / MINI unpacking

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

Format *UNPK 'ab cd ef ... '*

The two−character options have the following meanings:

AL all banks are unpacked but no coordinate sorting is done

VD VDET coordinates

IT ITC coordinates

TP TPC coordinates

TE dE/dx

EC ECAL (electron id.)

HC	HCAL
MU	Muons
FI	track fits
SO	to sort coordinates in phi to redo pattern recognition
CR	cal. object relationship banks
' '	NO unpacking

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

4.4 READ: Input from different card files

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

Format *READ 'card–file–name'*

The default file format is CARDS.

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

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

4.5 DEBU: Debug output

There are two debug levels:

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

The debug level is available as the variable KDEBUG.

4.6 TIME: Job time control

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

Remarks If no TIME card is given, 65 seconds is assumed by default on UNIX systems , 15 seconds on other computers . The number on the TIME card must be given WITHOUT a decimal point. In ALPHA, it is converted to a floating point number and is available as the variable QTIME (see 6.7.5). On all CERN computers, time is counted in IBM 370/168 seconds.

4.7 Histograms

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

4.7.1 HIST: Write histogram file

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

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

data set name	see 4.1.1.
Default file format	HIS
parameters:	(optional − described in 5.2.1)
UPDA	
NOOV	
DISP	
NREC	
RECL	

4.7.2 HTIT: General histogram title

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

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

4.7.3 NOPH: Histogram Printing

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

Format: *NOPH*

4.8 FIEL: Magnetic field

Magnetic field can be set to a given value (only for Monte Carlo studies):

FIEL 15. Set magnetic field to 15 KGauss.

4.9 FRF0: Use track fit without vertex detector (POT/DST only)

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

4.10 FR10 or FR12: Use unsmeared track hits (Monte Carlo POTS only)

To be used by experts only.

Monte-Carlo only: for datasets produced with JULIA 306 and after, the TPC hits are smeared to better reproduce the data. The track banks FRFT/0 and FRFT/2 contain the parameters of the tracks fitted from the smeared hits and are transmitted to ALPHA variables.

If the data card FR12 (resp. FR10) is given, the ALPHA variables are filled with parameters of tracks fitted from unsmeared hits with (resp. without) VDET. This allows to make some comparisons w.r.t the smeared hits.

FR10 or FR12 is not compatible with FRF0. Cannot work on MINIs.

4.11 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).

4.12 EFLW and EFLJ: Energy Flow

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

Replacing the EFLW card with the EFLJ card causes ALPHA to store jets based on energy flow objects in addition to the energy flow objects themselves. See Ch. 11 for details on energy flow packages.

4.13 Particle table

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

4.13.1 PMOD: Modify particle attributes

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

Parameters:

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

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

4.13.2 PNEW: New particles

Modify attributes of an existing particle.

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

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

4.13.3 PTRA: Modify particle names in the MC particle table

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

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

Parameters:

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

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

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

4.14 SYNT: Syntax Check

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

SYNT is given, then

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

4.15 BPER,BPWT,BSIZ,SIBE : chunk-by-chunk beam position for MCarlo

The ALPHA variables XGETBP, QVTXBP(I), QVTEBP(I), and QVTSBP(I) (see chapter 6.2) provide chunk-by-chunk beam position information for real data. They also provide effective beam positions for correctly simulating the size of the luminous region in Monte Carlo. For Monte Carlo, the default behavior is to use the luminous region size parameters (database bank ALRP) for the geometry year of the Monte Carlo file.

The BPER card allows the user to override the default. The parameters from the year(s) specified on this card are then used; each event is assigned at random to a particular year. The years (in the LEP 1 era) are weighted according to the numbers of qqbar events in the MAYB/PERF runs in the VD run selection.

Different weights may be specified by means of the BPWT card . This card is mandatory for specifying the relative weights of LEP 2 periods, if two or more periods are selected with BPER .

Arbitrary beam sizes may be simulated by means of the BSIZ card.

The x,y beam position uncertainties for Monte Carlo datasets may be smeared using the SIBE data card.

The BPER, BPWT, BSIZ and SIBE cards have no effect when real data is analyzed.

BPER [per1 [per2...]] per1,per2... are run periods as specified in the first column of ADBR. Years may be given as "1992" or "92" or "9200". Other run periods have the form "yymm", such as "9510". A list of valid run periods is printed (and execution terminates) if the BPER card is given with invalid or no run periods.

BPWT [iwt1 [iwt2...]] iwt1,iwt2,... are *integer* weights for the periods listed on the BPER card. An error results if there is no accompanying BPER or if it has a different number of tokens. The weights may have an arbitrary normalization.

BSIZ σ_x σ_y $[\sigma_z]$ σ_x and σ_y are floating point numbers giving the desired sizes of the luminous region in cm, transmitted to QVTXBP(1) and (2). The BSIZ card overrides all values in ALRP. The value of σ_z , if given, is returned in QVTXBP(3) but has no other effect. If σ_z is not given, the value 1 is returned in QVTXBP(3).

Notes: the beam position uncertainty is set to zero if BSIZ is given. BPER/BPWT cards are ignored if BSIZ is given.

SIBE meanx σ_x **meany** σ_y Allows to perform a smearing of the x and y beam position uncertainties for Monte Carlo datasets. meanx (resp. meany) is the central value of the error, σ_x (resp σ_y) is the gaussian dispersion around this mean value. Units are in 10^{-3} cm. The results are returned in QVTEBP(i).

Examples :

1. To simulate 1993 and 1994 beam size conditions with natural relative weights:

BPER 1993 1994

2. To simulate 1993 and 1994 with equal weights:

BPER 1993 1994

BPWT 50 50

3. To simulate a luminous region size of 120 x 7 microns:

BSIZ 0.0120 0.0007

4. To smear the beam position uncertainties as follows:

error in x 27.4 microns with 2 microns dispersion,

error in y 23.3 microns, 4.7 microns dispersion:

SIBE 2.74 0.2 2.33 0.47

4.16 BOBS: Beam spot information using LEP BOMs

If this data card is given, the ALPHA variables XGETBP, QVTXBP(I), QVTEBP(I), and QVTSBP(I) (see chapter 6.2) are filled with the values obtained from the LEP BOMs for data in 1996 and after. The data card has no effect for LEP1 data or for Monte Carlo data.

This data card cannot work on datasets produced with a JULIA version before 284, which build the bank BLQP containing the necessary information.

Format *BOBS sigmax sigmay ibomonly idebug*

Parameters:

sigmax:	Real number equal to the LEP BOM+QSO x intrinsic resolution in cm (default : 0.0040, i.e. 40 microns)
sigmay:	Real number equal to the LEP BOM+QSO y intrinsic resolution in cm (default : 0.0010, i.e. 10 microns)
ibomonly:	integer flag = 1 if one wants to use only the LEP BOM + QSO information = 0 to use the weighted average between the LEP BOM + QSO information and the VDET chunk-by-chunk information
idebug:	integer flag = 1 to get a debug printout of the BOM informations, = 0 otherwise.

To know the status of the ALPHA beam spot position for a given event when using this card, one has to use the subroutine QFILBP_STATUS (see 12.18 on p. 127).

4.17 QFND: Calling the QFNDIP package

The QFNDIP package of D. Brown et al. is described in the ALEPH Note 92−047, PHYSIC 92−042. It performs a precise determination of the interaction point on an event−by−event basis. It needs the beam constraint from the event−chunk beam position computed in the routine QFGET_BP (see sec. 6.4).

All necessary input banks are read by ALPHA at initialisation time. The results of QFNDIP are available as the main vertex variables QVX, QVCHIF and QVEM described in sec. 8.4 on p. 62.

You should NOT call yourself neither QFGET_BP nor QFNDIP. QFGET_BP is called automatically for each event; QFNDIP is called when you put the QFND card.

4.17.1 QFND card:

If the data card:

QFND is given, QFNDIP is called automatically and the results are stored in the main vertex variables (sec 8.4 on p. 62).

Warnings: The event−chunk beam position and therefore the interaction point determined by QFNDIP is NOT available for real data taken in 1989 and 1990. It is available for all Monte-Carlo data.

QFNDIP must be called before calling the b−tagging package QIPBTAG (sec 12.5).

4.17.2 DWIN card:

This data cards allows to use in QFNDIP a selection of charged tracks fulfilling specified cuts on their longitudinal components.

Format *DWIN ilong window*

Parameters:

ilong if = 1 use tracks with : Distance(track-jet) < window
if = 2 use tracks with : Abs(Distance(track-jet)) < window

window cut value in cm

If DWIN is absent, or if ilong = 1 and window=0., the results are the standard QFNDIP ones.

Recommended values for LEP 2 data:

DWIN 1 0.035

4.18 Special Cards:

Hereafter are described special data cards, to be used only by specialists.

MEXT Forces the muon track extrapolation through HCAL and muon chambers. This works only on POTs/DSTs.

EFOU Forces the writing of the Energy Flow bank EFOL on the output tape. See sec 11.1 for details.

REV0 Forces the V0s to be redone with VDET coordinates, the result being put in the bank YV0V, NR=3, even if old V0s are present (YV0V NR = 0 or 1). This works only on POTs/DSTs. Please note that all MINIs produced since June 1993 have been done with this option. See sec 8.2.12 for details.

REPG Calls automatically the JULIA routine GAPGAC to rebuild from scratch the photon bank PGAC. This works only on POTs/DSTs, and if JULIA has been linked. Please note that all MINIs produced since December 1994 have been done with this option. See sec 8.3.3 for details.

DRGA Forces the dropping of the PGAC bank read from the data. Mandatory if one wants to rebuild properly this bank using the REPG card.

MINA 'list' add non-standard banks to a private MINI, see Appendix D for details.

NOSC Removes the SICAL clusters from the energy flow objects created by ENFLW. See sec 11.1 for details.

NOPX Do not rebuild the pad dE/dx bank PTPX when reading a POT.

NSID Suppresses the call to the SICAL cleaning routine SICLID in the Energy Flow package. Valid only for data processed after May 12th, 1999. See sec 11.1 for details.

NQVJ Suppresses the internal jet finding of QVSRCH, to use user-defined jets. See sec 12.8 for details.

PCOR 0 Get the "sagitta correction" made for all charged tracks - see QPCORR (Sect. 12.20) for details.

PCOR 1 Get the "sagitta correction" made only for charged tracks having VDET coordinates - see QPCORR (Sect. 12.20) for details.

Chapter 5

Creating Histograms and Ntuples

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

5.1 Booking and Filling Histograms/Ntuples

All of these routines call standard HBOOK4 routines.

5.1.1 Book a 1−dimensional histogram

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

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

Input arguments:

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

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

5.1.2 Book a 2−dimensional histogram

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

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

Input arguments:

ID histogram ID number – nonzero integer

CHTITL histogram title – character variable up to 80 characters

NX number of bins in X

XMI lower edge of lowest X bin

XMA upper edge of highest X bin

NY number of bins in Y

YMI lower edge of lowest Y bin

YMA upper edge of highest Y bin

VMX normally set equal to 0.– see HBOOK manual for details.

5.1.3 Book a Profile histogram

CALL QBPROF (ID, CHTITL, NX, XMI, XMA, YMI, YMA, CHOPT)

QBPROF includes the same features as QBOOK2. The arguments are the same as for CALL HBPROF (...): – see HBOOK manual .

5.1.4 Book an Ntuple

CALL QBOOKN (ID, CHTITL, NVAR, TAGS)

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

Input arguments:

ID Ntuple ID number – nonzero integer

CHTITL Ntuple title – character variable up to 80 characters

NVAR number of variables

TAGS name of character array of dimension NVAR containing names for variables to be stored.

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

corresponds to :

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

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

5.1.5 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.

5.1.6 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).

5.1.7 Fill Ntuple with many variables

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

Fills the Ntuple ID with the variables A1 ... An ($n < 80$). CALL QHFN (ID, A1, A2) corresponds to

```
DIMENSION A(80)
A(1) = A1
A(2) = A2
CALL HFN (ID, A)
```

5.1.8 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 ($n < 80$; see QHFN) and with run / event number as variables n+1 and n+2 (see QHFR).

5.1.9 Sample ALPHA program to book and fill histogram, Ntuple

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

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

5.1.10 Limitations to the ALPHA histogram facilities

The default length (100000 words) of the /PAWC/ buffer of HBOOK may be too short for special uses. It may be increased by modifying the parameter LQPAW in the QUIHIS routine.

There are no ALPHA facilities to book of fill column-wise Ntuples.

If the user wants to use Ntuples with more than 80 variables, he has to use the standard HBOOK calls. The output file must be open using HROPEN and written using HROUT/HREND. See the HBOOK manual for more details.

If the user wants write very large Ntuple files which exceed the default maximum length of 64 Mbytes , he has to use the NREC and RECL parameters in the HIST data card , as described in 5.2.1 .

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 printed out to the log file in the program termination phase (i.e., after return from QUTERM; see 3.4 and 3.5.4).

The HIST data card is necessary for writing histograms and Ntuples to a histogram file which can be used in a subsequent interactive session (PAW). Users who want to write files larger than the maximum default size of 64 Mbytes must give appropriate NREC and RECL parameters in their HIST Card , see below .

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

data set name see 4.1.1.

Default file format HIS

parameters (optional) :

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

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

NOOV Overwrite protection (see 4.1.4). Cannot be used with UPDA.

On DEC/VMS : Unnecessary.

On UNIX : Strongly recommended.

DISP Returns the resulting Histogram File to the original computer from which the ALPHA job was submitted. Must be used if you are running ALPHA in batch on a UNIX platform using **alpharun**,see App. C on p. 148.

NREC nmax Sets the maximum number of records in the file to 'nmax'. If parameter NREC is missing, or if 'nmax' is missing or not in the range 100-100000, the default of 16000 records is used. The NREC parameter cannot be used with the UPDA parameter.

RECL nwords Specifies the record length in the file to be 'nwords'. If parameter RECL is missing, or if 'nwords' is missing or not in the range 1-999999, the default record length is used. The default record length is 1024 if the UPDA parameter is missing. If the UPDA parameter is present, the default record length is 0 to let ZEBRA try to determine the record length from the file itself (if ZEBRA fails to determine the record length, ALPHA retries with RECL=1024; if the file can still not be opened the user must specify explicitely the record length that was used to create the file).

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

5.2.2 NOPH: Histogram Printing

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

Format : *NOPH*

5.2.3 HTIT: General histogram title

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

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

Chapter 6

Mnemonic symbols

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

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

6.1 Mathematical and physical constants

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

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

6.2 Run information

KRINNE	number of events in run (with HV on)
KRINLF	LEP fill number
KRINDQ	data quality (see bank description for RLUM variable RQ)
XIOKLU	Logical flag, .TRUE. if Luminosity obtained from LCAL (for LEP I runs before 16600 , and all LEP II runs)
QRINLU	Luminosity from LCAL in nb**-1 , from database , if XIOKLU = .TRUE.

XIOKSI Logical flag, .TRUE. if Luminosity obtained from SiCAL (for LEP I runs after 16600 ; always .FALSE. for LEP II runs)

QRSLLU Luminosity from SiCAL in nb**-1 , from database , if XIOKSI = .TRUE.

KRINNZ number of Z $\rightarrow$ hadrons (from database if available) . This is only an estimation obtained using CLAS 16 events with XLUMOK = .TRUE.

KRINNB number of Bhabhas (from database if available)

QMFLD magnetic field (best estimate), in kGauss. Taken from data card FIEL (if given), run header bank RALE, or run header bank AFID for MC events. If ABS(QMFLD) > 20 QMFLD is set to 15.

QVXNOM, QVYNOM, QVZNOM run-by run mean position of interaction point; taken from the database bank LFIL (see KBPSTA below).

Important warning: These quantities are to be considered as obsolete, except for 1989-1990 data.

For all other years of data taking, and for all Monte Carlo datasets, they are strictly equivalent to QVTXBP(I) described in the next paragraph, 6.4 on p. 35.

QVXNSG, QVYNSG, QVZNSG (Statistical error) ^{2} on QVXNOM etc.;

Obsolete, kept only for backward-compatibility.

KBPSTA Beam Position Status: method used to determine QVXNOM, QVYNOM, etc.

Obsolete, kept only for backward-compatibility.

- KBPSTA = 0: No mean beam position for the run
- KBPSTA = 1: Mean beam position computed with VDET, per RUN (most of runs taken from 1991 to 1994)
- KBPSTA = 2: Mean beam position computed with VDET, per FILL (this happens when the run has too few events)
- KBPSTA = 3: Mean beam position computed WITHOUT VDET, per FILL (this is the status of all runs in 1989/1990 and of the runs in 1991 which do not have a working VDET)

QDBOFS average systematic offset of D0.

Obsolete, kept only for backward-compatibility.

6.3 LEP c.m.s. energy QELEP

The ALPHA variable **QELEP** gives the best available value of the LEP c.m.s. energy, in GeV.

This is intermediate between a Run information and an event-by-event information.

Since ALPHA 123.08 (October 1998) this value is obtained as follows:

- For Monte Carlo events, QELEP is the generated LEP energy.
- For about 99% of LEP2 events QELEP is obtained from the 15 minutes time-chunk determinations of the LEP c.m.s energy in ALEPH, as given by the LEP Energy Group (values stored in the database bank RNL2). See the note ALEPH 97-029 (PHYSIC 97-24) for more details.
- For LEP1 events, or for a tiny fraction of LEP2 events without time-chunk, QELEP is obtained from run-averaged energies (values stored in the database bank RNR2).
- If none of the above values is available (in general because the run is too short), QELEP is taken as the average LEP energy over the corresponding LEP fill (values stored in the database bank RNF2).

For a given event, to know how the LEP energy was obtained, please use the subroutine QWHICH_EN described in 12.16 on p. 126.

Important: Please DO NOT call any more the subroutines QELEP2 or QELEP1.

6.4 Chunk-by-chunk luminous region information

The quantities described below are used internally by the QFNDIP package (sec. 4.17 on page 24). They give the best possible precision and should therefore be preferred, when available, to QVXNOM,QVYNOM, QVZNOM described above.

XGETBP = .TRUE. if the event-chunk beam information from QFGET_BP is available. The quantities listed below are defined only if: XGETBP = .TRUE.

QVTXBP(I) I=1,2,3: x,y,z position of luminous region centroid (cm). Notice: QVTXBP(3) is always equal to zero.

QVTEBP(I) I=1,2,3: Estimated uncertainty on QVTXBP(I) (cm). Notice: QVTEBP(3) is always equal to 1.

QVTSBP(I) I=1,2,3: x,y,z rms size of luminous region (cm), averaged over the year (or run period). The routine QBEAMX (see section 12.10) provides access to the metachunk-by-metachunk values of sigma_x.

Real data: The beam position x and y and their uncertainties returned in QVTXBP and QVTEBP are taken from the ALPB bank in the run record, or from the beam.position file if the dataset was produced by JULIA version < 275.03 . The sizes of the luminous region returned in QVTSBP are taken from the ALRP database bank and are average values for the year (or run period).

Monte Carlo: The beam position x and y returned in QVTXBP are generated so as to simulate the desired beam size distribution. The primary vertex position of the event is extracted from truth information and an offset is added according to the size of the luminous region. The smearing corresponding to the average chunk-by-chunk beam position uncertainty (QVTEBP) is included.

The "desired" beam size distribution is by default taken from measurements of the actual LEP conditions for the geometry year of the Monte Carlo file. This default may be changed by means of data cards (BPER,BPWT,BSIZ,SIBE) as explained in section 4.15.

The horizontal size of the luminous region σ_x is thrown at random from the measured distribution, which is parametrized as the sum of two Gaussians (not a delta function). The uncertainties in x and y returned in QVTEBP, as well as the three sizes returned in QVTSBP, are taken from the ALRP database bank and are average values for the year (or run period).

Important remarks:

To know how the beam position information was obtained for a given event, use the subroutine QWHICH_BP (see Sec 12.17 on p. 126).

Another way of obtaining the beam position is to use the LEP BOMs (for LEP2 data only). See the description of the data card BOBS in Sec 12.17 on p. 126) and the subroutine QFILBP_STATUS to know how the beam spot was found for a given event in Sec 12.18 on p. 127.

6.5 Event information

6.5.1 Event header: from bank EVEH

KRUN Run Number

KEVT Event Number

KEVEDA DAte

KEVETI TIme

KEVEMI(I) trigger Mask I, I = 1 to 4

KEVETY event TYpe

KEVEES Error Status

6.5.2 Event directory information

KCLASW EDIR class word. See App. G for the definition of event classes.

6.5.3 Event generator status: from bank KEVH

KKEVID process ID

QKEVWT WeighT

6.5.4 Detector HV status: from banks REVH, LOLE

XHVTRG =.TRUE. if XLUMOK checks are satisfied. (To save time, XHVTRG should be used rather than calling the function XLUMOK.)

KREVDS Detector status word from REVH bank

XVLCAL = .TRUE. if the LCAL is OK (i.e., the LOLE bank is present and there is no error condition)

XVSATR = .TRUE. if SATR HV is OK

XVITC = .TRUE. if ITC HV is OK

XVTPC = .TRUE. if TPC HV is OK (bit 15)

XVTPCD = .TRUE. if all TPC HV is OK (dE/dx bit − bit 4)

XVECAL = .TRUE. if ECAL HV is OK (i.e., all ECAL HV bits are on)

XVHCAL = .TRUE. if HCAL HV is OK (i.e., all HCAL HV bits are on)

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

6.5.5 Trigger Information: from XTEB or XTRB, XTCN

KXTET1 Level 1 trigger bit pattern

KXTET2 Level 2 trigger bit pattern

KXTEL2 Level 2 bit pattern after applying the enabled trigger mask

KXTCGC Number of GBXs since the last event readout

KXTCLL Number of level 1 yes conditions since the last event readout

KXTCBN e^- bunch number

KXTCCL level 1 control word

KXTCHV HV status word (equivalent to KREVDS above)

KXTCEN mask of enabled triggers

6.5.6 Beam position from BOM system: from bank BOMB

QVXBOM x beam position from BOM

QVYBOM y beam position from BOM

KERBOM Error code for BOM

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

6.6 ECAL Wire Energies

QEECWI(IMOD) ECAL wire energy summed over the whole module IMOD in GeV. Modules $1-12$ refer to endcap A, $13-24$ to the barrel, and $25-36$ to endcap B.

Please refer to subroutine QEWSUM (see 12.19 on p. 127) to get the separate wire energies on even/odd planes of the ECAL modules.

6.7 ALPHA Internal Constants, Variables

6.7.1 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

6.7.2 Program status

KSTATU -1: program initialization; 0: event processing; 1: program termination

KDEBUG debug level (see 4.5)

6.7.3 Event status

XMINI = **.TRUE.** if event read from Mini-DST = **.FALSE.** if event read from POT or DST

XNANO = **.TRUE.** if event read from Nano-DST = **.FALSE.** if event read from POT or DST or MINI

XMCEV = **.TRUE.** if MC truth available for the event

6.7.4 Input / output units

KUINPU	event input = 20, 21
KUOUTP	event output = 50
KUEDIN	EDIR (event directory) input = 30
KUEDOU	EDIR (event directory) output = 60
KUCONS	data base = 4
KUPRNT	log file = 6
KUPTER	terminal = 76 or 0 (see Ch. 2)
KUCARD	card input = 7
KUCAR2	second card input = 8
KUHIST	histogram output unit = 15
KURTOX	EXCH format histogram output on UNIX machines = 16

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

6.7.5 Timing

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

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

6.7.6 Character variables

CQVERS	ALPHA version number e.g. 121.01 (6 char)
CQDATE	date at start of job (8 char)
CQTIME	time at start of job (8 char)
CQFOUT	data set name of event output file = ‘ ’ if no output file given

Chapter 7

ALPHA "Tracks" and "Vertices"

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

The properties of the tracks and vertices are found using functions which refer to the ALPHA "track" and vertex numbers. For example, the energy of ALPHA "track" ITK is QE(ITK). The properties available for each tracklike object and each vertex are described in sections 8.1 and 8.4, respectively.

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

7.1 Access by Fortran DO loops

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

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

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

KFCHT,KLCHT	number of first (last) charged track.
ITK	loop index = ALPHA track number
QP(ITK)	momentum of track ITK (see 8.1.1)

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

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

7.1.1 ALPHA "TRACKS" :

ALPHA "tracks" are stored in the internal ALPHA bank QVEC.

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

Calorimeter Objects: KFCOT, KLCOT, KNCOT For physics analysis with optimised energy resolution, one should better use the Energy Flow objects (see below).

Calorimeter objects can be any of the following:

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

Calorimeter objects can be further divided into:

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

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

"REconstructed" objects: KFRET, KLRET, KNRET "REconstructed" objects are:

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

Reconstructed Standard V0s from YV0V: KFV0T, KLV0T, KNV0T See Section 7.5.1 for comments on the daughters of standard V0s.

Tracks from standard V0 Vertices: KFDCT, KLDCT, KNDCT Charged tracks outgoing from reconstructed DeCay vertices. The momenta for these tracks are calculated relative to the secondary vertex position.

[1] For ECAL wire energies, see 6.6.

Caution: this section includes only the daughter tracks from reconstructed standard V0s from YV0V, and not from the "Long V0s" coming from the new tracking with JULIA version 302 and after (see below).

Reconstructed Long V0s: KFLVT, KLLVT "Long V0s" (coming from the new tracking with JULIA version 302 and after). One gets the loop limits for them through the following subroutine call:

CALL QLV0T(KFLVT,KLLVT)

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

Energy Flow objects made with the ENFLW package (see Chapter 11) should be used for all analyses which need an optimised calorimeter energy resolution.

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

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

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

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

7.1.2 ALPHA "VERTICES" :

ALPHA "vertices" are stored in the ALPHA internal bank QVRT.

Monte Carlo vertices ("truth"): KFMCV, KLMCV, KNMCV These vertices come either from the KINGAL generator or from the interactions in GALEPH.

REconstructed Vertices: KFREV, KLREV, KNREV For datasets coming from the "old" tracking (JULIA version 285 and before) this includes: the Main vertex (which is the first vertex, KFREV, when it exists), Standard V0s from the YV0V package, and Secondary vertices fitted by any of the KVFITx routines, see sec. 9.3 on p. 74.

It is possible that there is no Main vertex in an event.

For datasets coming from the "new" tracking of 1997 (JULIA version 302 and after): in addition to the previous vertices one may get three other vertex categories described below. To access them by DO loops, one has to call specific subroutines which give the corresponding vertex numbers:

- "Long V0s":
  CALL QLV0V(KFLV0,KLLV0)
  KFLV0, KLLV0 are the first and last "Long V0" vertex numbers, if any.
- Nuclear Interactions:
  CALL QNUCL(KFNIV,KLNIV)
  KFNIV, KLNIV are the first and last nuclear interaction vertex numbers, if any
- Kink Vertices:
  CALL QKINKV(KFKIV,KLKIV)
  KFKIV, KLKIV are the first and last kink vertex numbers, if any

Please take note of the above subroutine calls: they are mandatory to get the loop limits for these new vertex categories.

7.2 Loops over ECAL and HCAL objects

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

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

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

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

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

7.3 Relationships between objects in different subdetectors

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

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

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

For example,

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

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

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

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

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

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

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

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

7.4 Direct access to particles

7.4.1 Particle name and class

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

- The particle name (example: ‘E+’ or ‘GAMMA’); see 15.1.
- The object (= track = particle) class which distinguishes between reconstructed tracks, the Monte−Carlo truth, and any Lorentz frame derived from one of them:
  - Class KRECO: Reconstructed objects read from the event input file and everything derived from them except Lorentz boosted objects.
  - Class KMONTE: Monte−Carlo truth.
  - Each Lorentz frame is considered as its own class (see 9.6). These classes are denoted by the number of the object which defines the Lorentz rest frame.

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

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

7.4.2 Example: Loop over all MC generated positrons

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

KPDIR (’particle−name’, ICLASS) *’particle−name’:* Character string (1 to 12 characters). *ICLASS:* Track class (see 7.4.1): KRECO or KMONTE or a track number ITKRST if ITKRST has been used before to define the rest frame for a Lorentz boost (see 9.6).

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

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

7.4.3 Particle name versus integer particle code – time consumption

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

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

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

 90   CONTINUE ...
```

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

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

7.4.4 Loops over a particle and its antiparticle

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

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

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

 90   CONTINUE ...
```

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

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

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

7.4.5 Analysis of particle systems: Examples

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

Combinations of the same particles: π^+ π^+

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

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

$\Delta^{++} \rightarrow$ p π^+

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

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

```
      IPIPLU = KFOLLO (IPIPLU)
      GO TO 20
    ENDIF
    IPROT = KFOLLO (IPROT)
    GO TO 10
  ENDIF
```

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

7.5 Mother – daughter relationships

7.5.1 Mother to daughters

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

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

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

KDAU (ITK,I) track number of Ith daughter

Note for standard V0s from YV0V: The daughters of a standard V0 (section V0T) are stored in the DCT section (see 7.1). These tracks are copies of tracks in the CHT section, but their momenta are recalculated relative to the secondary vertex position. The function KCHT (see 7.6.2) returns the CHT track number corresponding to a track in the DCT section.

Example:

```
      DO 10 IV0=KFV0T,KLV0T
c---  First daughter of V0 (in DCT section)
       I1DCT=KDAU(IV0,1)
C --- Corresponding track in CHT section.
       I1CHT=KCHT(I1DCT)
    10 CONTINUE
```

The above note is **not valid** for the "Long V0s" from the new tracking (vertices in the range KFLV0,KLLV0).

7.5.2 Daughter to mother(s)

The connection from a daughter to its mother(s) is available for MC particles and for daughters of "saved" composite particles (see KVSAVE in 9.2.9).

The QVADDx routines (9.2.2) and the jet / event topology routines 10) do NOT set up this relation.

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

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

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

7.6 Access to the "same" object

The "same" object means:

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

7.6.1 Loops over copies of the "same" object using KSAME

Example:

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

Remarks: This loop is terminated if it arrives at the original track. KSAME never returns 0. The same particle can be boosted several times into the same Lorentz frame provided that the boosts are performed with different mass or other hypotheses (see 9.6.1); if you start with the original track ITK, the most recently boosted hypothesis is reached first.

7.6.2 Find original copy of a charged track

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

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

7.7 Match reconstructed tracks and MC truth

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

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

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

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

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

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

KBESTM (ITK) Best match to track ITK.

Remarks:

- The match is performed on the basis of shared hits in the TPC and ITC.
- A much better matching is provided for charged tracks since ALPHA 122 by a set of new routines: MCMATCH, JULMATCH, VDHMATCH which are described in Section 12.13 on p. 124.
- A specific matching is available for Energy Flow objects (charged and neutral); it is described in Section 11.1.5 on p. 96.

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 Ith outgoing track

Chapter 8

ALPHA Track and Vertex Attributes

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

The units used throughout ALPHA are cm, sec, GeV, GeV/c, GeV/c^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) P = momentum of vector I.

QX (I) x momentum component

QY (I) y momentum component

QZ (I) z momentum component

QE (I) Energy

QM (I) Mass (use QMASV0 for V0 mass; see below)

QCH (I) CHarge

KCH (I) NINT (QCH(I)) (be careful with quarks)

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

8.1.2 V0 Mass

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

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

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

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

The argument 'name' in QMASV0 can be given in lower or upper case .

8.1.3 Track error covariance matrix

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

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

8.1.4 Distance to the beam position:

Available for charged reconstructed tracks.

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)	χ^2 due to QDB and QZB.

The coordinates of the beam position used to compute these values are QVTXBP(I) (see 6.4 on p. 35).

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

8.1.5 Stability code

KSTABC (I) Stability code

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

1 Particle does not decay.

2 Neutral particle that decays in the calorimeter volume. Charged particle that decays in the TPC or calorimeter volume. Here, TPC and calorimeter volumes are full cylinders (including the beam pipe region).

3 One of the ancestors of this stable particle has interacted with matter. Energy and momentum are NOT conserved.

0 Decay products of "stable" particles including all garbage in the calorimeter.

−1 Particle decays immediately (resonance etc.).

−2 Particle decays with finite decay length but before reaching the detector volume (see above).

−3 Particle interacts with matter before reaching the detector volume. The decay products do not conserve energy and momentum.

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

8.1.6 Test a particle's name

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

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

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

Important remark : These functions MUST be called with 'part−name' being a member of the official list of ALPHA particle names given in Appendix E. They are CASE-SENSITIVE and do not work if you supply the wrong particle name .

The same functions exist for integer particle codes IPC = KPART ('part−name') instead of the particle names (see 7.4.3):

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

8.1.7 Test if particles are based on the same object

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

8.1.8 Flags, pointers, etc.

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

KTN (I) GALEPH/ JULIA/ ENFLW track number. This means:

- for MCarlo "truth" particles (I=KFREV to KLREV) KTN(I) is the row number of ALPHA track I in the FKIN bank;
- for reconstructed tracks (I=KFCHT to KLCHT) KTN(I) is the row number in the FRFT bank;
- for CAL Objects it is the row number in the PECO or PHCO banks;
- for ENFLW objects it is the row number in the EFOL bank.

KCLASS(I) Track class:

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

KTPCOD (I) track's Particle Code

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

KLUNDS (I) LUND status code (MC particles only!); defined only if the event generator is JETSET or is interfaced with JETSET.

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

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

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

8.2 "Track" related detector data

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

8.2.1 Global geometrical track fit: Bank FRFT

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

XFRF (I) **.TRUE.** if track fit data are available for track I

QFRFIR (I) Inverse radius of curvature in x−y projection Signed positive if track bends counterclockwise, negative 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) 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 to 6 in the order IR,TL,PH,D0,Z0,AL. Note that for data processed before April 1993 the error matrix is valid at the innermost point used in the track fit, and therefore does not include multiple scattering in material before the tracking chambers. For data processed - or reprocessed - after April 1993 the error matrix is valid at the interaction point and therefore includes the effect of multiple scattering .

QFRFC2 (I) χ^2 of helix fit

KFRFDF (I) Number of degrees of freedom

KFRFNO (I) Option flag for track fit

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
KFRTNT (I)	Number of coordinates in TPC
KFRTNR (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. To be used with care: the TPC bits were not filled properly until JULIA 281 (July 1996) and the VDET bits are always 0 for all datasets since Summer 1995.
KFRIBC (I)	Bit pattern for calorimeters
KFRIDC (I)	Dead zone pattern for calorimeters
QFRIPE (I)	Electron probability
QFRIPM (I)	Muon probability
QFRIPI (I)	Pion probability
QFRIPK (I)	Kaon probability
QFRIPP (I)	Proton probability
QFRINK (I)	No Kink probability
KFRIQF (I)	Track Quality Flag from UFITQL
XFRIQF (I)	**.TRUE.** if KFRIQF(I) = 1 or 3

8.2.4 dE/dx data: Bank TEXS

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

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) **.TRUE.** if HCAL data are available for track I

KHMANF (I) Number of Fired planes

KHMANE (I) Number of Expected fired planes

KHMANL (I) Number of Fired planes within Last ten planes

KHMAMH (I) Mult Hits: number of clusters in last ten planes

KHMAIG (I) IGeomflag: flag of possible dead zone

QHMAED (I) Energy Deposit in corresponding HCAL storey

QHMACS (I) χ^2

KHMAND (I) Number of Degrees of freedom

KHMAIE (I) Expected bit map

KHMAIT (I) True bit map

KHMAIF (I) Preliminary identification flag

8.2.7 Muon chamber data: Bank MCAD

XMCA (I) **.TRUE.** if muon chamber data are available for track I

N = 1,2: Int/Ext chambers

KMCANH (I,N) Number of associated hits

QMCADH (I,N) Minimum distance hit−track

QMCADC (I,N) Cutoff on hit−track distance

QMCAAM (I) Min. angle between extrapolated and measured (in muon ch.) track

QMCAAC (I) cutoff on minimum angle

8.2.8 QMUIDO Muon Identification: Bank MUID

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

8.2.9 ECAL objects: Bank PECO

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

8.2.10 ECAL objects: Bank PEPT

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

8.2.11 HCAL objects: Bank PHCO

XPHC (I)	**.TRUE.** if HCAL data (PHCO) are available for calorimeter object "track" I
QPHCER (I)	Raw energy
QPHCTH (I)	Theta
QPHCPH (I)	Phi
QPHCEC (I)	Energy corrected for geometrical effects

KPHCKD (I)	Region code − see ALEPH 88−134
KPHCCC (I)	Correction code − see bank description
KPHCRB (I)	Relation bits − see bank description
KPHCPC (I)	PCOB number of associated cal. object

8.2.12 Reconstructed V0s: Bank YV0V

XYV0 (I)	**.TRUE.** if V0 data are available for track I
KYV0K1 (I)	JULIA/FRFT track number of positive track from V0 (NOT the ALPHA track number !)
KYV0K2 (I)	JULIA/FRFT track number of negative track from V0 (NOT the ALPHA track number !)
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)	χ^2 of V0 vertex fit
KYV0IC (I)	Fit hypothesis (see YV0V bank description)
QYV0DM (I)	Minimum distance between helices
QYV0S1 (I)	Psi angle for + track from V0
QYV0S2 (I)	Psi angle for − track from V0

8.2.13 Energy Flow: Bank EFOL

XEFO (I)	**.TRUE.** if energy flow (EFOL) data are available for “track” I (of the EFT section)
KEFOTY (I)	Type of energy flow object (see Sec. 11.2)
KEFOLE (I)	PECO number of associated ECAL object
KEFOLT (I)	FRFT number of associated charged track
KEFOLH (I)	PHCO number of associated HCAL object
KEFOLC (I)	PCOB number of associated calorimeter object
KEFOLJ (I)	EJET number of associated jet

8.2.14 Neutral objects from PCPA: Bank PCQA

XPCQ (I)	**.TRUE.** if PCQA data are available for track I
KPCQNA (I)	NAture of neutral object (see Sec. 11.3)

8.3 Photon attributes

The photon-finding package has been introduced in 1992 in the ALEPHLIB and has been deeply modified many times, getting each time a new name and giving different output banks.

The GAMPEK photon algorithm is described in the reports ALEPH 93−107 and ALEPH 94−057.

Users who want to study only π^0 production may use directly the QPI0DO algorithm described in 12.9 , page 114.

In the descriptions below , the index I refers to the "tracks" of the GAT section and runs from KFGAT to KLGAT.

8.3.1 Photons from GAMPEC: Output bank EGPC (Obsolete since July 1993)

This bank was written on POTs, DSTs and MINIs before July 1993 (JULIA versions 264 and before, MINI version 8 and before). It should not be used any more. Please refer to previous ALPHA manuals for the description of the related variables.

8.3.2 Photons from GAMPEX: Output bank PGPC (Obsolete since December 1994)

This bank was written on POTs , DSTs and MINIs between July 1993 and December 1994 by JULIA versions 265 to 271 or MINI 9. Please refer to previous ALPHA manuals for the description of the related variables.

8.3.3 Photons from GAMPECK: Output bank PGAC

This bank is written on POTs , DSTs and MINIs since December 1994 by JULIA version 272 and after, or MINI version 10 and after.

On these datasets, the photons from the old GAMPEC package (bank EGPC, see above) or GAMPEX (bank PGPC, see above), are no more available.

XPGAC (I)	**.TRUE.** if GAMPECK data are available for "track" I of the GAT section
QX (I)	Momentum of photon , x component
QY (I)	Momentum of photon , y component
QZ (I)	Momentum of photon , z component
QE (I)	Energy of photon
QPGPR1 (I)	Energy fraction in stack 1

QPGPR2 (I) Energy fraction in stack 2

QPGPF4 (I) Energy fraction in 4 central towers

QPGPDM (I) Distance to the closest track (cm)

KPGAST (I) NST1+100∗NST2+10000∗NST3, NSTi=number of storeys in stack i

KPGPQU(I) QUality flag

- CRCK + 10*DST1 + 100*DST2 + 1000*DST3
- DSTi = 1 if dead storey(s) in stack i
- CRCK = 1 if photon in crack region

QPGPQ1 (I) Quality estimator 1 for photon

QPGPQ2 (I) Quality estimator 2 for photon

QPGPM1 (I) 1st moment from CLMOMS analysis

QPGPM2 (I) 2nd moment from CLMOMS analysis

QPGPMA (I) Pi0 mass estimated from CLMOMS analysis

QPGPER (I) Raw energy of photon

QPGPTR (I) Raw Theta of photon

QPGPPR (I) Raw Phi of photon

QPGAEF (I) Expected fraction in 4 towers

QPGAEF (I) Geometrical correction

QPGAZS (I) Zero suppression

QPGAPL (I) Probability to be a fake photon of electromagnetic origin

QPGAPH (I) Probability to be a fake photon of hadronic origin

KPGAPN (I) Row number , in PGAC , of parent giving a fake photon

KPGAFA (I) Flag for fake discrimination

KPGAPE (I) Row number of corresponding PECO cluster

8.4 Vertex attributes

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

Warning : JULIA or QFNDIP Main vertex ?

For the Main vertex (IVX = KFREV and KVTYPE(IVX) = 1, see below) the quantities described below may have been obtained either from the JULIA vertexing or from the QFNDIP package if you asked for it through the data card QFND (sec 4.17). See the entry QVCHIF below to know which package was used to give the Main vertex positions .

QVX (IVX) x position

QVY (IVX) y position

QVZ (IVX) z position

KVN (IVX) JULIA/GALEPH vertex number

KVTYPE(IVX) vertex type

- = 1 for the Main vertex;
- = 0 for standard reconstructed V0s from YV0V (IVX = KFREV,KLREV)
  or for MC "truth" vertices (IVX = KFMCV,KLMCV)
- = 2 for a secondary vertex fitted by any of the KV-FITx routines, see sec 9.3 on p. 74.
- = 3 for a "Long V0" vertex from the new tracking
- = 4 for a Nuclear Interaction vertex from the new tracking
- = 5 for a Kink vertex from the new tracking

Vertex types 3,4,5 are defined only for datasets produced with JULIA version 302 and after.

QVCHIF(IVX) Chisquare of the vertex fit.

Special meaning for the Main vertex:

Always 0 if the Main vertex was found by JULIA.

If the Main vertex was found by QFNDIP, this chisquare is positive.

If QFNDIP was called and was unable to find a Main vertex, this Chisquare is set to a huge negative value.

QVEM (IVX,N,M) element (N,M) of the covariance matrix N,M = 1,2,3 in the following order: QVX,QVY,QVZ.

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

See Section 7.8 for pointers between ALPHA tracks and vertices.

8.5 Special attributes for a Kink Vertex

```
CALL QVKINK(IVX,PKINK,PTOUT,PLOUT,IHYPK,IER)
```

This routine allows to get special attributes defined only for Kink vertices produced by JULIA version 302 and after. These quantities are available from POTs, DSTs and MINIs.

Input arguments:

IVX ALPHA vertex number of a kink vertex (IVX must be between KFKIV and KLKIV, see sec 7.1.2).

Output arguments:

IER = 0 if IVX is really a kink vertex, = 1 if not.

PKINK Momentum of parent track at the kink position

PTOUT Pt of outgoing charged particle relative to parent

PLOUT Pl of outgoing charged particle relative to parent

IHYPK Hypothesis of mass cuts satisfied by this decay:

- = 1 $\pi \to \mu\nu$
- = 2 $K \to \mu\nu$
- = 3 $K \to \pi\pi^0$
- = 4 $\Sigma^+ \to p\pi^0$
- = 5 $\Sigma^+ \to n\pi^+$
- = 6 $\Sigma^- \to n\pi^-$
- = 7 $\Xi^- \to \Lambda\pi^-$
- = 8 $\Omega^- \to \Lambda K^-$
- = 9 $\Omega^- \to \Xi^0\pi^-$
- = 10 $\Omega^- \to \Xi^-\pi^0$
- = 0 if none of the above hypotheses are satisfied

CALL QKINKT(NKINKS,LISTKINKS)

This routine allows to get the number and list of charged tracks ending in Kink vertices produced by JULIA version 302 and after.

Output arguments:

NKINKS = number of kink vertices in the event.

LISTKINKS(I) = list of charged tracks ending in the I=1,NKINKS kink vertices; must be dimensioned to 50 in the calling routine.

Chapter 9

Kinematics and Track Operations

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

9.1 Scalar quantities

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

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

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

QMSQ2 (I,J)	(invariant mass)^2 of particles I and J
QM2 (I,J)	invariant mass of particles I and J
QMSQ3 (I,J,K)	(invariant mass)^2 of particles I, J, and K
QM3 (I,J,K)	invariant mass of particles I, J, and K
QMSQ4 (I,J,K,L)	(invariant mass)^2 of particles I, J, K, and L
QM4 (I,J,K,L)	invariant mass of particles I, J, K, and L
QDMSQ (I,J)	mass^2 of the 4−momentum difference p(I) − p(J). In a decay I −−> J + x, QDMSQ(I,J) gives the mass^2 of x.
QPPAR (I,J)	momentum component of particle I parallel to particle J
QPPER (I,J)	momentum component of particle I perpendicular to particle J
QDOT3 (I,J)	scalar product of momentum vectors I and J (3−vectors)

QDOT4 (I,J) scalar product of 4−vectors I and J = QE(I) * QE(J) − QDOT3(I,J)

QCOSA (I,J) cos (angle between tracks I and J) (lab frame)

QDECA2(I,J) cos (decay angle): In a two−body decay x → I + J, the decay angle is the angle between particle x and particle I, measured in the rest frame of particle x (i.e., the angle between the boost direction and particle I).

QDECAN(I,J) extension of QDECA2 for the n−body decay I → J + any. Note the different meaning of the first argument in QDECA2 and QDECAN.

QMDIFF(I,'part') mass difference between I and particle table mass of 'part'.

QMCHI2(I,'part') χ^2 resulting from mass difference between I and particle table mass of 'part'. This function is equivalent to

$$(\mathrm{QM(ITK)}{-}\mathrm{QPMASS}('\mathrm{part-name}'))^2/\mathrm{QSIGMM(ITK)}.$$

QMCHIF(I) χ^2 of mass-constrained fit (KVFITM or KVFITA – see 9.3). QMCHIF(I)=−1 if track I was not the result of a fit.

QVDIF2(IV1,IV2) distance between vertices IV1 and IV2 in $r{-}\phi$ (see 9.4).

QVDIF3(IV1,IV2) distance between vertices IV1 and IV2 in 3 dimensions (see 9.4).

QVCHIF(IV) χ^2/NDF of vertex fit for vertex IV (KVFITN or KVFITV – see 9.4).

9.2 Vector quantities

9.2.1 General Remarks

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

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

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

The function KVNEW (DUMMY) creates a new track (ISUM) in the system area which is needed as working space for most of the subroutines described here (see 9.2.8). New "tracks" can be created whenever necessary , however users have to be warned that if they do it inside loops , *they may quickly exceed the size of the BOS array.* A warning is issued if an "input" track is used as working space (*i.e.*if an "input" track is given as the output track of a routine).

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

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

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

9.2.2 Add 4−momenta of particles

Add two particles

CALL QVADD2 (ISUM, ITK1, ITK2)

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

Add three particles

CALL QVADD3 (ISUM, ITK1, ITK2, ITK3)

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

Add four particles

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

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

Add N particles

CALL QVADDN (ISUM, ITK)

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

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

The sum of all track momenta is stored in ISUM.

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

9.2.3 Recalculate 4−Vector of V0

CALL QIDV0 (ITK, 'PI+', 'PI−')

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

```
       DO 10 ITK=KFVOT,KLVOT
         CALL QIDVO (ITK, 'P', 'PI-')
         IF (QMCHI2 (ITK, 'LAMO') .LE. 9.)
      &    ISAVE = KVFITM (ITK, 'LAMO', IER)
         CALL QIDVO (ITK, 'pi+', 'p#')
         IF (QMCHI2 (ITK, 'LAMO') .LE. 9.)
      &    ISAVE = KVFITM (ITK, 'LAM#O', IER)
    10 CONTINUE
```

The particle names in the QIDV0 arguments may be given either in upper or lower case .

9.2.4 Copy a track

CALL QVCOPY (ITO, IFROM)

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

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

9.2.5 Cross product

CALL QVCROS (ICROSS, ITK1, ITK2)

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

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

9.2.6 Drop tracks

CALL QVDROP ('part−name', ICLASS)

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

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

will drop tracks with any track in class ICLASS. The main application of this subroutine is to drop all tracks in a specific Lorentz frame. See the example in section 9.6.3.

If ICLASS = KRECO or ICLASS = KMONTE: Only tracks created in the analysis program are dropped; tracks coming from the event input file cannot be dropped.

It should be noted that , as no BOS garbage collection takes place in ALPHA , the memory space corresponding to dropped tracks is NEVER released inside an event . If one exceeds the size of the BOS array , the only way to solve the problem is to increase this size in the QUIBOS subroutine (sec 3.5.5 on p. 8) .

9.2.7 Copy track attributes into a Fortran array

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

Copy 3−momentum of a track

CALL QVGET3 (ARR, ITK)

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

Copy 4−momentum of a track

CALL QVGET4 (ARR, ITK)

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

Copy covariance matrix of a track

CALL QVGETS (ERRMAT, ITK)

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

9.2.8 Create a new track

INEW = KVNEW (DUMMY)

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

9.2.9 Save a track

ISAVE = KVSAVE (ITK, 'part−name')

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

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

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

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

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

```
      ISUM = KVNEW (DUMMY)
      ITK1 = KPDIR ('E+', KRECO)
   10 IF (ITK1 .NE. 0)  THEN
        ITK2 = KPDIR ('E-', KRECO)
   20   IF (ITK2 .NE. 0)  THEN
C ... all e+ e- combinations:
          CALL QVADD2 (ISUM, ITK1, ITK2)
```

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

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

9.2.10 Save a track inside particle/antiparticle loop

ISAVE = KVSAVA (ITK, ‘part−name’, IANTI)

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

9.2.11 Save a track and set its mass

ISAVE = KIDSAV (ITK, ‘part−name’)

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

9.2.12 Save a track with class ICLASS

ISAVE = KVSAVC (ITK, ‘part−name’, ICLASS)

Save (see 9.2.9) track ITK in track class ICLASS independent of the track history. Track classes are described in 7.4.1. If class ICLASS does not yet exist, a new class is created. Note that the maximum number of new classes is six (see 9.6).

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

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

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

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

9.2.13 Modify track parameters

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

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

Scale track momentum

CALL QVSCAL (ITK, FACTOR)

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

Set mass of a track

CALL QVSETM (ITK, AMASS)

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

Set 3−momentum of a track

CALL QVSET3 (ITK, ARR)

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

Set 4–momentum of a track

CALL QVSET4 (ITK, ARR)

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

Set covariance matrix of a track

CALL QVSETS (ITK, ERRMAT)

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

9.2.14 Set User Track Flags

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

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.

9.2.15 Subtract track momenta

CALL QVSUB (IDIFF, ISUM, ISUB)

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

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

9.2.16 Zero track attributes

CALL QVZERO (ITK)

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

9.3 Kinematic fitting

None of the functions described below can work when reading a NANO-Dst
,

$\boxed{\text{IFIT = KVFITM (ITK, ‘part−name’, IER)}}$

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

$\boxed{\text{IFIT = KVFITA (ITK, ‘part−name’, IER, IANTI)}}$

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

The χ^2 for the track fit may be accessed with the statement function QMCHIF(IFIT) .

9.4 Vertex fitting with YTOP

The following functions provide an interface to the YTOP package in ALEPHLIB. This package is described in the note ALEPH 91−132 , SOFTWR 91−005 . All these functions build a new vertex available in QVRT with its full error matrix .

None of the functions described below can work when reading a NANO-Dst ,

$\boxed{\text{IFIT = KVFITN (ND, ID, ‘part−name’)}}$

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

$$IVX = KENDV(IFIT).$$

IFIT = -1 if the fit fails.

$\boxed{\text{IFIT = KVFITV (IV,ND, ID, ‘part−name’)}}$

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

Both functions refit the track parameters of the input tracks and calculate the 4-vector and error matrix of the new track (IFIT) at the fit vertex. The fit vertex position and error matrix are stored in the end vertex of IFIT:

KENDV(IFIT). There can be any number of input tracks, but if NTR > 10, KVFITx will first vertex tracks 1-10 and then add the following tracks to this vertex. Input tracks can be either charged tracks, V0s previously refitted using KVFITN (otherwise the error matrix is not available), or "tracks" resulting from a previous fit.

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

NDF = 2 * ND -3 for KVFITN

NDF = 2 * ND for KVFITV

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

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

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

Example:

Assume that you have a $D^0 \to K\pi$ candidate (ID0) and a lepton (ILEP) from a B-meson decay. The following code finds the vertex of the B decay.

```
      CALL QVSETM(IPION,QPMASS('PI+'))       !pion mass
      CALL QVSETM(IKAON,QPMASS('K+'))        !kaon mass
      ITL(1) = IPION
      ITL(2) = IKAON
      ID0    = KVFITN(2,ITL,'D0Kp')
      IF(ID0.GT.0) THEN
        RMD0 = QM(ID0)                ! vertex refitted D0 mass
        IVD0 = KENDV(ID0)             ! D0 vertex
        CHI2 = QVCHIF(IVD0)           ! chi**2 of the D0 vertex
C
        ITL(1) = ILEP
        ITL(2) = ID0
        IB     = KVFITN(2,ITL,'Blep')    ! fit B vertex
        IF(IB.GT.0) THEN
          IVB = KENDV(IB)                ! B vertex
          CHI2 = QVCHIF(IVD0)            ! chi**2 of B vertex
          DIST = QVDIF3(IVB,IVD0)        ! distance between B and D0
                                         ! vertex
        ENDIF
      ENDIF
```

IFIT = KVFITC (NTKD,ITKD,RMAS,DRMAS,'part−name')

Fits NTKD tracks in array ITKD to a common vertex with mass constraint RMAS in the fit. The new track is stored with the name 'part−name' and can be accessed as described above for KVFITN or KVFITV : The vertex number is the end vertex of track IFIT:

$$IVX = KENDV(IFIT).$$

IFIT = -1 if the fit fails.

NTKD	Number of input tracks to be fitted
ITKD	Array of input ALPHA "track" numbers
RMAS	Mass to which the tracks are constrained (in Gev)
DRMAS	Error on Mass**2 = 2*RMAS*Sigma(DMAS)
'part−name'	Name which will be given to the new track

All above quantities are input arguments to KVFITC .

This routine may give problems , or wrong results , if DRMAS is not properly tuned . It is strongly recommended to use KVFTMC instead (described just below) which doesn't need any error on the mass as input .

IFIT = KVFTMC (NTKD,ITKD,'part−name',USE,RMAS)

Fits a subset of NTKD tracks in array ITKD to a common vertex with mass constraint RMAS in the fit. Same remarks as for KVFITC for the fitted vertex .

NTKD	Number of input tracks in array ITKD
ITKD	Array of input ALPHA "track" numbers
'part−name'	Name which will be given to the new track
USE	Array of logicals of size NTKD defining which tracks to use in the fit
RMAS	Mass to which the tracks are constrained

All above quantities are input arguments to KVFTMC .

9.5 Auxiliary routine for V0s

The following routine is an interface to the YCHIV2 routine in the ALEPHLIB .

CALL QV0CHK(JULTR,CHI2,IER)

Computes the chisquare of a charged track belonging to a V0 with the constraint of the primary vertex . This routine allows to do subsequent cuts on this chisquare to get a given efficiency and purity of a V0 sample . The routine works without restriction on POTs or DSTs .

On MINIs , it works only if the QFNDIP package has been called , using the QFND data card in the input CARDS file .

JULTR JULIA “track” number .
BE CAREFUL ! It’s not the ALPHA track number !
If you know only the ALPHA track number ITK , you get the JULIA track number by the function KTN : JULTR = KTN(ITK) .

CHI2 Chisquare of track ITK with main vertex constraint , set to -999999. if no calculation possible .

IER = 0 if all OK , = 1 if no CHI2 calculation possible

Example of use of QV0CHK :

```
      DO IVTK = KFVOT,KLVOT           ! loop on all V0 tracks
         IF (XYV0(IVTK)) THEN            ! information on daughters exists
            IT1 = KYV0K1(IVTK)           ! 1st V0 daughter ( JULIA track number)
            IT2 = KYV0K2(IVTK)           ! 2nd V0 daughter ( JULIA track number)
            CALL QV0CHK(IT1,CHI1,IER1)
            CALL QV0CHK(IT2,CHI2,IER2)
C  here make checks or cuts on CHI1 , CHI2
        ENDIF
     ENDDO
```

Optimised cuts for K0s selection :

The suggested cuts to be used after calling QV0CHK in order to select K0s are as follows :

Cut1 For each track the chi**2 value returned by QV0CHK should be greater than 13.

Cut2 The sum of the TWO chi**2 for both V0 tracks should be greater than 80.

Both conditions should be satisfied . Please notice that the above cuts are for K0s only and cannot be applied to Lambdas.

9.6 Lorentz transformations

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

9.6.1 Boost a track and its daughters

IBOOST = KTLOR (ITK, IREST)

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

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

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

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

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

The track to be boosted (ITK) and the track which defines the rest frame (IREST) may belong to different track classes. No check is done that the boost makes sense. Note, however, an important restriction: If more than one track is boosted into a frame, all of them must come from the same class. This restriction prevents putting reconstructed tracks and MC truth into the same track class; see example in 9.6.3.

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

9.6.2 Boost a track

IBOOST = KTLOR1 (ITK, IREST)

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

9.6.3 Boost all tracks of a given class

CALL QTCLAS (ICLASS, IREST)

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

Example:

```
IREST = ... this momentum vector defines the rest system.
```

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

Chapter 10

Event Topology Routines

All of the subroutines described in this chapter perform loops over tracks or particles. The arguments and loop algorithms are similar for all of these subroutines, and are described in detail in Section 10.3. The "tracks" to be considered are selected with the routines QJOPTR (for reconstructed tracks) and QJOPTM (for Monte Carlo tracks); these routines also specify tracks to be used by the Lorentz transformation routine QTCLAS (see 9.6.3). In addition, the LOCK routines described in Section 10.2. can be used to exclude tracks from analysis by the QJxxxx routines described in this chapter.

10.1 Options for "QJxxxx" routines

10.1.1 Set option for reconstructed objects

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

Input arguments:

'reco−option' One of the following options:

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

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

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

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

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

10.1.2 Set option for MC particles

CALL QJOPTM ('MC−option', 'additional')

'MC−option' One of the following options:

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

'additional' Same as for QJOPTR.

10.2 Lock tracks / subsamples of tracks

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

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

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

10.2.1 Lock a single "track"

CALL QLTRK (ITK)

ITK ALPHA “track” number

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

10.2.2 Unlock a single “track”

CALL QLUTRK (ITK)

ITK ALPHA “track” number

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

10.2.3 Lock a track “family”

CALL QLOCK (ITK)

ITK ALPHA “track” number

The family of track ITK consists of:

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

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

10.2.4 Unlock tracks (locked with QLOCK)

$\boxed{\text{CALL QLZER (IREMC)}}$

IREMC = KRECO for reconstructed tracks and KMONTE for MC truth

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

10.2.5 Reverse the lock state (corresponding to QLOCK)

$\boxed{\text{CALL QLREV (IREMC)}}$

IREMC (see 10.2.4).

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

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

10.2.6 Second Lock

$\boxed{\text{CALL QLOCK2(ITK)}}$

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

$\boxed{\text{CALL QJADDP (SCALAR, ‘vector−name’, ICLASS)}}$

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

10.3.1 Input argument

ICLASS Class = KRECO or KMONTE or a Lorentz frame identifier (see 7.4.1). If ICLASS is KRECO, note that initially all charged particles have the pion mass and all neutral objects have mass = 0. This can be modified by CALL QVSETM (see 9.2.13). If ICLASS refers to a Lorentz frame, particles not boosted into the frame are ignored without notification. The routine QTCLAS (see 9.6.3) performs a Lorentz transformation of all tracks belonging to a class. If a particle has been boosted several times into the same frame, the most recently boosted hypothesis will be used (see remarks in 7.4.5).

10.3.2 Results

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

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

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

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

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

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

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

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

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

10.4 Momentum tensor eigenvalues and eigenvectors

CALL QJEIG (EIGVAL, 'eigenvector', ICLASS)

See also QJSPHE in 10.6 for sphericity value and axis.

Input argument:

ICLASS described in 10.3.

Results:

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

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

'eigenvector' Three eigenvectors:

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

10.5 Linearized momentum tensor eigenvalues and eigenvectors

CALL QJTENS (EIGVAL, 'eigenvector', ICLASS)

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

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

$$j, k = 1, 2, 3 \tag{10.2}$$

Input arguments and results are as described for QJEIG.

10.6 Sphericity

CALL QJSPHE (SPHERI, 'spheri−axis', ICLASS)

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

ICLASS described in 10.3.

Results:

SPHERI Sphericity value

'spheri−axis' Sphericity axis.

Error conditions:

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

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

10.7 Thrust

CALL QJTHRU (THRUST, 'thrust−axis', ICLASS)

Input argument:

ICLASS described in 10.3.

Results:

THRUST Thrust value.

'thrust−axis' Thrust axis.

Error conditions:

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

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

When reading MINIs (versions ≥ 10) :

The thrust calculation is very CPU-time consuming. To save time when reading a MINI, from MINI 10 onwards (written since December 1994) the thrust axis and value are calculated using the 'EF' option in QJOPTR (see 10.1.1). The results are stored in a special bank on the MINI. When reading these MINIs, the thrust axis and value are not computed from scratch, but read directly from this bank.

You don't need to change anything to your program if you want to use the 'EF' option. If you want to compute the thrust with another option, you MUST give the following sequence of calls:

```
XJTHRU=.TRUE.
CALL QJOPTR ('your reco-option' , 'additional' )
CALL QJTHRU(THRUST,'thrust-axis',ICLASS)
```

The logical variable XJTHRU belongs to the QCDE include file of ALPHA, and is set to .FALSE. at the beginning of each event.

10.8 Fox−Wolfram Moments

CALL QJFOXW(FOXWOL, ICLASS)

Input argument:

ICLASS described in 10.3.

Result:

FOXWOL Fox−Wolfram moments H0 − H4; DIMENSION FOXWOL(5).

10.9 Divide event into two hemispheres

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

Input arguments:

ICLASS described in 10.3.

IVEC Track number of vector which defines the "hemi"spheres.

COSCUT The cosine of the opening angle of a cone around IVEC. Tracks inside this cone belong to the same side, and all other ones belong to the opposite side. The word "hemisphere" is correct if COSCUT = 0.

Results:

'same−s' The 4−momentum sum of tracks on the same side as IVEC.

'opp−s' The 4−momentum sum of tracks on the side opposite to IVEC.

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

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

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

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

10.10 Missing energy, mass, momentum

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

Input arguments:

ICLASS	described in 10.3.
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.

Results:

PMISS	Missing momentum.
'miss−vector	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 Algorithms

There are two algorithms available, each with the same 3 variants (or schemes). The first algorithm is known as the JADE algorithm, which defines the invariant mass as:

$$M^2 = 2E_1E_2(1 - \cos\theta_{12}).$$

The second is known as the DURHAM algorithm and uses a proposal by Dokshitzer for invariant mass, which is less sensitive to soft gluons:

$$M^2 = 2(MIN(E_1, E_2))^2(1 - \cos\theta_{12}).$$

Both algorithms combine particles into jets using one of the following combination schemes:

E scheme	$P_{ij} = P_i + P_j$ $E_{ij} = E_i + E_j$
P scheme	$P_{ij} = P_i + P_j$ $E_{ij} = \vert P_{ij}\vert$
E_0 **scheme**	$E_{ij} = E_i + E_j$ $P_{ij} = E_{ij}(P_i + P_j)/\vert P_i + P_j\vert$

For more details, please read the internal reports ALEPH 90−149 (SOFTWR 90−015) and ALEPH 91−151 (SOFTWR 91−006).

JADE Algorithm with E scheme:

`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.

Input arguments:

ICLASS described in 10.3.

YCUT Cut on the scaled invariant mass of 2 tracks. Pairs of tracks are merged if their scaled invariant mass is smaller than YCUT.

EVIS The visible energy of the event. If EVIS equals 0, the visible energy is computed as the sum of the input particle energies.

Results:

NJETS is the number of "jets" .

- > 0 = number of jets found
- -1 = input error with EVIS , or no particles
- -2 = error in one of the particles
- -3 = too many jets found
- -4 = unknown combination scheme requested
- -5 = unknown algorithm requested
- -99 = Not enough BOS workspace to do jet finding

'name' Vectors containing 4−momenta of the jets.

EXAMPLE:

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

DURHAM Algorithm with E scheme:

$\boxed{\text{CALL QDMMCL (NJETS, `name', ICLASS, YCUT, EVIS)}}$

The arguments and usage are identical to that for QJMMCL above.

Generalised Version : JADE or DURHAM or GENEVA or INVMAS algorithm

$\boxed{\text{CALL QGJMMC (NJETS, `name', ICLASS, YCUT, EVIS,SCHEME,VERSN)}}$

The arguments and usage are identical to that for QJMMCL above , except for the extra input arguments : **Input arguments:**

SCHEME	CHARACTER*2 : 'E' or 'P' or 'E_0' (see above)
VERSN	CHARACTER*6 : 'JADE' or 'DURHAM' or 'GENEVA' or 'INVMAS' to select the corresponding algorithm
YCUT	- if > 0. : Cut on the scaled invariant mass of 2 tracks. Pairs of tracks are merged if their scaled invariant mass is smaller than YCUT. - if < 0. : fixed number of jet asked for, i.e. you can request that the event is clustered until a certain number of jets is found. Example: setting YCUT=-4. will force the clustering to get 4 jets and only 4 jets as output.

Note that 'NORMAL' and 'BETTER' are alternatives for 'JADE' and 'DURHAM' for historical reasons.

If the 'INVMAS' algorithm is selected, the 'E' scheme is automatically switched on, whatever has been put in the input argument SCHEME.

QJMMCL and QDMMCL both call this routine , which itself calls the ALEPHLIB routine FJMMCL , which actually does the jet finding.

$\boxed{\text{CALL GETYIJ (MyValues,Nsteps)}}$

If you have called QGJMMC with a negative value of YCUT to obtain a fixed number of output jets, this routine returns a vector of all Yijs computed, and the number of steps needed to get 4 jets.

Example: In your user routine you define a vector:

```
   Real*4  MyYvalues(1000)
   Integer Nsteps
   Real EVIS,Y34,Y45,YCUT,EVIS
   Character*2 Scheme
   Character*6 VERSN

.... then at some stage you call the clustering ...

   call QJOPTR('EF',' ')
```

```
      EVIS=0.
      YCUT=-4.
      SCHEME = 'E'
      VERSN  = 'DURHAM'
      call QGJMMC(NJETS,'myjets',KRECO,YCUT,EVIS,SCHEME,VERSN)
      IF (NJETS.LE.0) RETURN  ! Error

   ... and afterwards you call ...

      call GETYIJ(MyYvalues,Nsteps)

   ... this returns a vector of all yijs computed and
       the number of steps needed to get to 4 jets. Therefore

      Y34 = MyYvalues(Nsteps)
      Y45 = MyYvalues(Nsteps-1)
```

10.11.2 Scaled Minimum Distance Algorithm

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

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

Input arguments:

ICLASS described in 10.3.

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 $2Evis$); only jets with energies $> 2\eta Evis$ 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.

Results:

NJETS is the number of "jets".

'name' Vectors containing 4−momenta of the jets.

10.11.3 JETSET algorithm LUCLUS from LUND

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

Input arguments:

ICLASS described in 10.3.

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=0.25GeV)

DMAX2 Max. distance to join 2 clusters (usually=2.5 GeV)

MULSYM
- = 1 for symmetric distance criterion (usual)
- = 2 for multicity distance criterion

Results:

NJETS is the number of "jets"
- = −1 if not enough particles
- = −2 if not enough working space (KTBOMX)

TGEN Generalized thrust

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

'name' Vectors containing 4−momenta of the jets.

10.11.4 PTCLUS: Jet-finding algorithm

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

The PTCLUS jet-finding algorithm is described in the note ALEPH 89 – 150 (PHYSIC 89-060) by M. Scarr and I. TenHave.

Input arguments:

ICLASS described in 10.3.

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

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

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

Results:

NJETS is the number of "jets". (−1 if algorithm fails)

TGEN Generalized thrust

'name' Vectors containing 4−momenta of the jets.

Chapter 11

Energy Flow

Historically, three energy flow packages have been used in ALEPH: the mask algorithm of M-N. Minard and M. Pepe-Altarelli, the PCPA−based energy flow of A. Bonissent, and the ENFLW package of P. Janot. Since ALPHA 115 (May 1993), the ENFLW package is fully integrated in ALPHA; all features of the other two packages are not maintained any more. Therefore, users are strongly advised to use the ENFLW energy flow.

11.1 ENFLW Energy Flow

The ENFLW algorithm is described in the Section 10 of the paper:

“ Performances of the ALEPH detector at LEP ”

Nucl. Inst. and Meth. **A 360** (1995), pp. 481−506 .

To use the ENFLW energy flow analysis, the EFLW card must be given in the ALPHA card file. If the EFLW card is present, the EFT section of ALPHA is filled with selected charged tracks and neutral ECAL and HCAL clusters. These objects can be accessed with DO loops (KFEFT, KLEFT, KNEFT – see 7.1.1) or with the particle name ‘EFLW’ using the functions KPDIR and KFOLLO (described in 7.4).

The charged tracks that appear in the EFT section are copies of standard ALPHA charged tracks from the CHT section. Therefore, if a charged track in the CHT section is locked (using QLTRK or QLOCK), the corresponding track in the EFT section will be locked also (and vice versa).

All statement functions providing information about charged tracks can be used directly with charged tracks in the EFT section .

11.1.1 Access to ENFLW informations

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

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

KEFOTY (I) Type of energy flow objects (exclusive list, no double counting):

- 0 = Charged Track (Pion assumed, not identified either e or mu)
- 1 = Electron
- 2 = Muon
- 3 = Track from a standard V0 (either Λ, K_s^0 or γ conversion) from the YV0V package
- 4 = Electromagnetic (γ or π^0)
- 5 = ECAL hadron/residual
- 6 = HCAL element
- 7 = LCAL element (No Particle Identification available for LCAL)
- 8 = SICAL element (No Particle Identification available for SICAL)

KEFOLE (I) PECO number of associated ECAL object

KEFOLT (I) FRFT number of associated charged track

KEFOLH (I) PHCO number of associated HCAL object

KEFOLC (I) PCOB number of associated calorimeter object

KEFOLJ (I) EJET number of associated jet

11.1.2 Event topology routines and ENFLW

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

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

Example:

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

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

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

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

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

11.1.3 Removing of SiCAL clusters in ENFLW

A special data card: NOSC is foreseen to remove the SiCAL clusters from ENFLW objects. This works on POT/DST/MINI.

However, be careful: when writing MINIs from DST/POT with NOSC, the SiCAL clusters are NOT saved on the MINIs and will not be recovered when reading these MINIs even without NOSC. This is OK for private productions; the official MINIs are produced with the SICAL clusters in energy flow objects.

11.1.4 Cleanup of bad SiCAL clusters in ENFLW

Since ALPHA 124.09 (May 1999) all bad SiCAL clusters are removed from the energy flow objects. This is done by calling the ALEPHLIB subroutine SICLID. This works for all POTs, but works only for DSTs and MINIs processed with JULIA version 309 and after.

This cleanup may be inhibited by a special data card, NSID.

11.1.5 Matching ENFLW informations and true information

The code providing this information is available since ALPHA 122.30 (June 1997).

For charged tracks, the matching between ENFLW objects and Monte Carlo true information (and *vice-versa*) is done by the standard KMTCH function (7.7 on p. 50).

For ECAL objects, it is done using the PEMH, PECO and PCOB banks. For HCAL objects, it is done using the PHMH, PHCO and PCOB banks for datasets generated with JULIA version 285.1 and after (March 1996 and after); for previous datasets a tight angular matching (100 mrad) is used as a substitute when PHMH is not available. For ECAL hadronic residuals (KEFOTY = 5) and SICAL clusters (KEFOTY = 8) the matching is not fully efficient, since the FKIN/PECO information is not stored for SICAL clusters.

The matching information is provided through the following functions:

KNTRU (ITK) Number of matches of ALPHA "track" ITK

The matching is ENFLW → Monte Carlo if ITK is between KFEFT and KLEFT

The matching is Monte Carlo → ENFLW if ITK is between KFMCT and KLMCT

As for charged tracks, any ENFLW object may have more than one match, therefore additional information is provided through the following functions:

KTRU (ITK,I) Ith match to ALPHA "track" ITK (I=1,.. KNTRU(ITK))

QSTRU(ITK,I) Matching quantity for the Ith match to ALPHA "track" ITK

- if QSTRU < -1.0: (-QSTRU) is the number of shared VDET + ITC + TPC hits
- if $-1.0 <$ QSTRU $< 0.$ (-QSTRU) is the matching angle (in radians) for HCAL/SICAL
- if QSTRU $> 0.$ QSTRU is the shared energy (from PEMH or PHMH banks)

11.2 Mask Energy Flow

This algorithm is described in the report ALEPH 89−110 (PHYSIC 89−043). Its results are not written any more in ALEPH datasets.

11.3 PCPA-based Energy Flow

This algorithm is described in the report ALEPH 92−055 (PHYSIC 92−048).

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

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

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

To use only NEOB objects:

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

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

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

KPCQNA (I) NAture of neutral object (see Sec. 11.3)

- 1 Isolated gamma

- 2 Gamma from multi-gamma neutral cluster
- 3 Gamma from identified π^0
- 4 Gamma from electron bremsstrahlung
- 5 Gamma from electromagnetic charged cluster
- 10 Unresolved gamma-gamma
- 12 Residual electromagnetic energy from neutral cluster
- 13 Residual electromagnetic energy from charged cluster
- 17 Neutral hadron
- 18 Residual hadronic energy from neutral cal object
- 19 Residual hadronic energy from charged cal object with no HCAL component
- 20 Residual hadronic energy from charged cal object with HCAL component
- 21 contribution from an ECAL cluster for which EBNEUT was in error
- 22 contribution from an LCAL object

Example:

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

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

Chapter 12

Other ALPHA Physics Routines

12.1 dE/dx Analysis

12.1.1 Calculate dE/dx for Track ITK from TPC wires

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

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

Input arguments:

ITK ALPHA track number of a charged reconstructed track.

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

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

Q(nhyp) 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.

TL Useful length of the track (cm).

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

SIGMA(nhyp) 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 χ^2 with 1 d.o.f. as: $\chi^2 = ((\mathrm{RI}-\mathrm{RIEXP})/\mathrm{SIGMA})^2$.

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
- =6: there is no valid dE/dx calibration for this run

12.1.2 Calculate dE/dx for Track ITK from TPC pads

```
CALL QPADX(ITK,NHYP,RMASS,Q,RI,XNS,TL,RIEXP,SIGMA,IER)
```

This routine is an ALPHA interface to the ALEPHLIB routine TPDHYP. Note that the user must check the return code IER before trying to use any of the output arguments.

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, RIEXP, and SIGMA may be scalar variables.
RMASS(nhyp)	Array of masses, one for each hypothesis.
Q(nhyp)	Array of charges, one for each hypothesis.

Output arguments:

RI	Measured ionization, normalized such that RI=1 corresponds to minimum ionizing.
XNS	Number of wires on the track (NOT integer!).
TL	Useful length of the track (cm).
RIEXP(nhyp)	Expected ionization for each mass hypothesis, normalized such that RIEXP=1 corresponds to minimum ionizing.
SIGMA(nhyp)	One standard deviation resolution error for each hypothesis. This is the expected dE/dx resolution, given XNS, TL, RIEXP, and the momentum resolution. **Note** that one can calculate a χ^2 with 1 d.o.f. as: $\chi^2 = ((\mathrm{RI}-\mathrm{RIEXP})/\mathrm{SIGMA})^2$.
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

- =3: input charged track has no dE/dx information.
- =4: cannot find the necessary database calibration banks, TC1X, TC2X, TP3X and/or TC3X.
- =5: cannot find RUNH or EVEH bank
- =6: cannot find TCSX or TCGX banks

12.1.3 Combined dE/dx estimation using both wires and pads

`CALL QDDX(ITK,NHYP,RMASS,Q,NS,XNPAD,EXTIM,IER)`

Provided by Tommaso Boccali, 19 April 1999.

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, and EXTIM may be scalar variables.
RMASS(nhyp)	Array of masses, one for each hypothesis.
Q(nhyp)	Array of charges, one for each hypothesis.

Output arguments:

NS	Number of wires on the track, -1 if none.
XNPAD	Number of pads on the track (Floating !), -1. if none.
EXTIM(nhyp)	(ri-riexp)/σ for each particle hypothesis.
IER	Error return code =0: an estimator was computed; =1: no information available.

12.1.4 Modified QDEDX for Monte Carlo (Obsolete ! Kept only for backward-compatibility).

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

To be called only for Monte Carlo datasets produced before December 1993; for more recent datasets one has to use QDEDX (see the use of the CHTSIM function just below).

This routine serves the same purpose as QDEDX, but treats Monte Carlo differently. QDEDX takes the dE/dx from the detailed simulation program TPCSIM. QDEDXM, however, only takes the number of samples and the track length from TPCSIM, from which a prediction for the resolution is obtained. The measured momentum and the Monte Carlo true mass then are used to predict the mean dE/dx, which is smeared by a gaussian random number to give the simulated dE/dx.

The advantage of this approach is that it is easy to adjust the parameterization to give agreement with data, whereas to do so with TPCSIM is nontrivial and would require regeneration of the Monte Carlo data set. The disadvantage is that the non−gaussian tails (which are small and arise primarily on the high side, due to unresolved track overlap) are not simulated. An option does exist to try to get the best of both worlds: by calling QMTAIL one can set a parameter to tell QDEDXM to retain the tail simulated by TPCSIM beyond a specified number of standard deviations. The distribution below that number of standard deviations then is obtained from the gaussian random number generator. Clearly this solution is not perfect, since the distributions generally will not match at the chosen cut value.

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

```
CALL QMTAIL(CUT)
```

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

12.1.5 QDEDXM or not QDEDXM ? (Obsolete ! Kept only for backward-compatibility).

```
LOGICAL FUNCTION CHTSIM(IVERS)
```

Function defined only for MCarlo datasets .

Since the release of TPCSIM version 216 in December 1993 , it is much better to use QDEDX for MCarlo data as well as for real data . The logical function CHTSIM is .TRUE. if the Monte−Carlo dataset currently read has been produced with a version of TPCSIM , IVERS , which allows to use QDEDX . If CHTSIM is .FALSE. , one has to use QDEDXM .

CHTSIM needs to be called only once for a given Monte-Carlo run .

Output argument:

IVERS	The TPCSIM version number for the current dataset .

12.1.6 Check TPC High Voltage for dE/dx

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

The function TCHKHV, from the ALEPHLIB, is used to check the TPC high voltage before using the dE/dx information from the TPC. It checks the data base bank TDBS to find whether any sectors were being intentionally operated at reduced voltage during the run in question. If so, then only the normal TPC HV bit is checked. Otherwise, the "dE/dx" HV bit is checked.

Input arguments:

KRUN ALEPH run number.

KEVT ALEPH event number.

Output arguments:

IFLG What kind of test was made?

- =0: test was made on dE/dx HV bit.
- =1: test was made on TPC tracking HV bit.
- =2: no test was made (banks not found). TCHKHV=.FALSE.

TCHKHV = .TRUE. if HV is on; .FALSE. otherwise.

12.1.7 Check Existence of dE/dx Calibration for Run

```
LOGICAL FUNCTION TCHKEX(KRUN)
```

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

12.2 Photon conversions

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

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

Input arguments:

I1 ALPHA track number of a charged track.

I2 ALPHA track number of a another charged track.

Output arguments:

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.

12.3 Muon Identification: QMUIDO

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

This routine is an ALPHA interface to the ALEPHLIB routine AMUID. It collects useful information from the banks HMAD, MCAD, and MUID.

For most users, who only look at the identification flag IDF, this routine is useless since this flag can be accessed with the statement function KMUIIF (see section 8.2.8). The only purpose of calling this routine is to look in more detail at the muon identification.

QMUIDO cannot be called when reading a Nano-Dst .

Input argument:

ITK ALPHA track number of a charged reconstructed track.

Output arguments:

IRUN No longer filled (needed for backwards compatability)

IBE Bitmap of the planes EXPECTED to have fired in the HCAL

IBT Bitmap of the planes which have fired in the HCAL

IM1 Number of associated muon chamber hits in the inner layer

IM2 Number of associated muon chamber hits in the outer layer

NEXP Number of planes expected to have fired in the HCAL

NFIR Number of planes fired in the HCAL

N10 Number of planes fired in the last ten expected HCAL planes

N03 Number of planes fired in the last three expected HCAL planes

XMULT Excess hit multiplicity in the last ten planes on the HCAL

RAPP Distance between track extrapolation and closest muon chamber hit in standard deviations (the distribution is only approximately normal)

ANG Angle between track extrapolation and closest muon chamber hits in standard deviations (the distribution is only approximately normal). Only available for tracks with at least one muon chamber hit in each layer

ISHAD Shadowing flag = 0 if track is not shadowed; otherwise it is the JULIA track number of the shadowing track.

SUDNT Sum of HCAL hit to track residuals in the last 10 planes.

IDF Official muon identification flag.

- = 1 if muon flagged only by HCAL
- = 2 if muon flagged only by MUON
- = 3 if muon flagged by both HCAL and MUON 3 is the .AND. of 1 and 2
- = 10 is one hit in each layer of MUON chambers but failing tight matching criteria
- = 11 is good HCAL pattern
- = 12 is one and only one MUON hit
- = 13 is good HCAL + one and only one muon
- = 14 is good HCAL + one hit in each layer
- = 15 is one hit in each layer of MUON chambers passing tight matching criteria
- = 0 not a muon
- = -1 to -15 as above but lost shadowing contest

IMCF Monte Carlo true source of this track

- = 0 ambiguous or data
- = 1 primary b
- = 2 secondary c
- = 3 primary c

- = 4 b to tau
- = 5 other muon
- = 6 non decaying hadron or electron
- = 7 decay hadron

IER No longer filled (needed for backwards compatability)

12.4 Utility Routines for VDET Analysis

12.4.1 Number of VDET hits per layer for track ITK

`CALL QVDHIT(ITK,IVHIT)`

Input argument:

ITK ALPHA track number of a reconstructed charged track.

Output argument:

IVHIT

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

12.4.2 VDET HV status

`LOGICAL FUNCTION XVDEOK(dummy)`

A detailed description of this function can be found in the report ALEPH 91−161 (PHYSIC 91−137) .

The function XVDEOK returns .TRUE. if the VDET HV is on for the current event and .FALSE. otherwise. XVDEOK uses the HV bits and also calls KVGOOD (see below).

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

XVDEOK cannot work when one is reading a Nano-Dst.

12.4.3 VDET Readout Status

INTEGER FUNCTION KVGOOD(dummy)

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

KVGOOD readout and HV status:

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

12.5 B-Tagging routine QIPBTAG

12.5.1 General considerations

QIPBTAG is an algorithm for tagging the presence of long-lived particles written by D. Brown and M. Frank.

It computes the impact parameter of tracks relative to the primary vertex reconstructed using the QFNDIP subroutine (4.17 on p. 24). After calibration, the algorithm converts the track signed impact parameter significance into a probability that the track came from the primary vertex found by QFNDIP. Track probabilities are combined to give jet, hemisphere and event probabilities.

Several important modifications have been done to the original QIPBTAG, especially to take into account the new VDET for data from 1996 and after (VDET96 option, J.Carr and M. Thulasidas, June 1997), to have a consistent b-tag for LEP2 data, and to get from the database bank QIPC the calibration constants which are dependant on the period of data taking.

The last version released with ALPHA124.05 (thanks to I. Tomalin, April 1999) recognises automatically the type of data being read and gives the best possible b-tagging efficiency. See ALPHA124.NEWS for more details.

To use QIPBTAG properly, one must read the complete description of the method and of the package in the Aleph Note 92−135 (August 1992).

12.5.2 Calling the QIPBTAG routine

CALL QIPBTAG(IRET,NTRK,NJET,ITRK,IFRF2,PRTRK,PRJET,PRHEMI,PREVT)

Input argument: none

Output arguments:

IRET
- = 0 good event , analysis performed
- = 1 Number of jets less than 2
- = 2 Jets outside solid angle cuts
- = -2 No beam spot information
- = 3,4 No tracks suitable for btag analysis
- = -3 Error in interaction point finding

NTRK Number of tracks used to calculate the probabilities (maximum 400)

NJET Number of jets found (maximum 40)

ITRK Array (dim 40) of ALPHA track numbers of the NJET jets found, sorted by energy

IFRF2 Array (dim 400) of row numbers in FRFT 2 of the NTRK tracks used in the analysis

PRTRK Array (dim 400) of the probabilities of each track used in the analysis

PRJET Array (dim 40) of the probabilities of each jet of ITRK

PRHEMI Array (dim 2) of the probabilities of each hemisphere

PREVT Probability of the event

All the above arrays must be properly dimensioned in the calling routine.

Additional QIPBTAG informations are available in the ALPHA include file BTAGRAW, which may be found in $ALROOT/alphavsn/inc/btagraw.h

12.5.3 Data cards for QIPBTAG

Mandatory data card :

QFND QFNDIP is needed to find the event interaction point

Optional data cards :

BHIS nnnn Books default diagnostic histograms, offset by nnnn

CALB Create a private calibration. The user must also CALL BTAG_FIT('filename') in QWTERM, and the output calibration cards will be written to the file 'filename'.

NUMJ nn with nn> 3, will force QIPBTAG to calculate the b-tagging probability for the first nn jets of the event, ordered by decreasing energy. If this data card is not provided, nn=3 by default.

BTRK nn 'name' Define a user track type with ALPHA track name 'name'. Up to 3 track types can be defined per job, nn must be ≤ 3. These tracks will never be removed by QIPBTAG selection cuts. Tracks should be selected or created in QUEVNT and copied to name 'name' before calling QIPBTAG. If the track is a mother track created by vertexing (see KVFITN), its daughter tracks are excluded from QIPBTAG, and the FRFT 2 track number is set to 20000 + ALPHA track number. If the CALB card is present, user defined track types will be calibrated along with the rest.

BNEG Use negative impact parameter tracks in the definition of jet, hemisphere, and event probabilities. This slightly enhances efficiency at the cost of the negative impact parameter control sample.

NOBG Forces BNEG option off, even if VDET96 option is ON. (Of course, must not be used together with BNEG !).

HJET Define the hemispheres according to the leading jet direction. By default, the event is split by the thrust direction (hemi 1 being positive). The hemisphere axis can also be supplied externally for each event as the ALPHA track of name 'THRUST'.

JRES Overwrite the default jet definition and corresponding angular resolution parameterization. User-supplied jets should have ALPHA track name 'QIPBJETS'.

NQIP Include VDET 96 modifications even if one is reading LEP 1 data or MCarlo (VDET description of years 1991 to 1995 included).

OQIP Do NOT Include VDET 96 modifications even if one is reading LEP 2 data or MCarlo (VDET description of 1996 and after). Strongly recommended if you are reading LEP1 data reprocessed in 1998.

TRA2 Overwrite the default track selection cuts. This card is for use by experts only .

QITI Calls the QIPBTCPU routine, see section 12.5.5.

12.5.4 Remarks on QIPBTAG

QIPBTAG cannot work on 1989 and 1990 real data or Monte Carlo (no VDET).

QIPBTAG works properly only on real data for which the VDET is OK (runs given by the SCANBOOK selection VDET OK), and on events for which XVDEOK = .TRUE. and for which QFNDIP has found a good interaction point.

To use QIPBTAG only for events with a good interaction point and VDET OK you should call it after performing some tests like in the example below:

```
IF (XVDEOK(DUMM).AND.XGETBP.AND.QVCHIF(KFREV).GT.0.) THEN
   CALL QIPBTAG(.....)
ENDIF
```

12.5.5 Auxiliary routine QIPBTCPU

If you want to know where QIPBTAG is spending CPU time , you can add a

```
CALL QIPBTCPU('SUMMARY',2)
```

at the end of your program .

12.6 Smear 3-d impact parameter in Monte Carlo: QSMEAR

The routine QSMEAR may be used AFTER a call to QIPBTAG.

Its purpose is to smear the three dimensional impact parameter significance and delete extra tracks in the Monte Carlo. It includes options for doing this work dependently on the polar angle and momentum of the tracks.

Calling sequence:

CALL QSMEAR(FIRST,QIPVER,SMBIN,RTBIN, NTRK,NJET,IFRF2,PREVT,PRHEMI,PRJET,PRTRK)

Input Arguments:

FIRST	LOGICAL = Is this the first call to QSMEAR since calling QIPBTAG (or QIPBTAG2) ?
QIPVER	CHAR*1 = '1' if being used after call to QIPBTAG = '2' if being used after call to QIPBTAG2.
SMBIN	CHAR*1 = '0' no smearing correction at all. = 'N' no binning in theta/p for smearing. = 'T' binning in theta for smearing. = 'P' binning in p for smearing. = 'B' binning in both theta/p for smearing.
RTBIN	CHAR*1 = '0' no track deletion at all. = 'N' no binning in theta/p for deletion. = 'T' binning in theta for deletion. = 'P' binning in p for deletion. = 'B' binning in both theta/p for deletion.
NTRK,NJET,..	Same arguments of call to QIPBTAG.

Output Arguments:

Same arguments overwritten with smeared values.

The -ve tag quantities in /BTAGRAW/ are also smeared.

It is possible to smear Monte Carlo from a different year than the data by including the card QPYR in the ALPHA card file with a YYMM code as an argument. Example:

12.7 Invariant mass b tag: QBMTAG

The routine QBMTAG can be used in conjunction with QIPBTAG to provide extremely pure (99%) b hemisphere tagging.

In addition to providing a btag for each event hemisphere, it also does so for each jet in the event.

12.7.1 General considerations

Invariant mass b tag.

Can be called after QIPBTAG or QVSRCH, to further increase b purity.

For each event-half QBMTAG loops over tracks in order of decreasing inconsistency with primary vertex, and stops when their combined invariant mass exceeds 1.8 GeV. (i.e. The mass of a typical charmed hadron). It then returns a tag, CLMASS, indicating the probablity that the "last track" used came from primary vertex.

For a very high purity tag, CLMASS should be combined with the QIPTBAG or QVSRCH tags. The optimal linear combination (together with suggested cuts) is:

```
        0.7*CLMASS - 0.3*log10(QIPBTAG) > 2.4
    or   0.7*CLMASS + 0.3*QVSRCH > 8.3
```

12.7.2 Calling the QBMTAG routine

CALL QBMTAG(METHOD,NTRACK,NJET,TRKJET,PROBTRK, FRF2TRK,PVTX,SVTX,DJET,CLMASS,CTSTAR,NSEC,CLMASJ, CTSTAJ,NSECJ,LISTTRK,LISTEH,LISTJET)

INPUT ARGUMENTS:

METHOD INTE : =1 if using QIPBTAG, =2 if using QVSRCH.

Following arguments only needed if METHOD=1 (dummy otherwise):

NTRACK INTE : Number of tracks used by QIPBTAG.

NJET INTE : Number of jets considered by QIPBTAG.

TRKJET(*) INTE : ALPHA track number of QIPBTAG jets.

PROBTRK(*) REAL : QIPBTAG track probabilities.

FRF2TRK(*) INTE : Row in FRFT bank of QIPBTAG tracks.

Following arguments only needed if METHOD=2 (dummy otherwise).

PVTX(3) REAL : Position of primary vertex from QVSRCH.

Only needed if METHOD=2 (dummy otherwise), same for following arguments.

SVTX(3,2) REAL : Positions of two secondary vertices in jet coordinate system from QVSRCH.

DJET(3,2) REAL DJET(3,2) : Unitized jet directions from QVSRCH.

OUTPUT ARGUMENTS: Dimensioned 2 to correspond to two event-halves:

CLMASS(2) REAL : For METHOD=1, = -log10(confidence level) for "last track" to come from primary vertex (c.l. in range 0-1).
For METHOD=2, = QSQT(CHI**2 (primary) minus CHI**2 (secondary)) for "last track".

CTSTAR(2) REAL : Decay angle of "last track" in "b" rest frame.

NSEC(2) INTE : Number of tracks used to reach 1.8 GeV mass. (zero if couldn't reach 1.8 GeV).

Following jet info available only if METHOD=1:

CLMASJ(*) REAL : Same variables as CLMASS for jets, ordered as in TRKJET

CTSTAJ(*) REAL : Same variables as CTSTAR for jets, ordered as in TRKJET

NSECJ(*) INTE : Same variables as NSEC for jets, ordered as in TRKJET

LISTTRK(*) INTE : List of all ALPHA charged tracks and V0s which were considered by QBMTAG. If the track is neutral, it is a V0. If a V0 is not in the range KFV0T-KLV0T, then it has been created using YTOP, and its daughters are (refitted) copies of tracks from the KFCHT-KLCHT section.

LISTEH(*) INTE : If track comtributed to 1.8 GeV hemisphere mass then =1 or =2 to show which hemisphere it's in. If =0, then track was rejected.

LISTJET(*) INTE : If track comtributed to 1.8 GeV jet mass, then =ALPHA track number of jet. If =0, then track was rejected.

12.7.3 Remarks on QBMTAG

If Method=1, then one can obtain higher purities by using the BNEG card. This allows QIPBTAG and QBMTAG to use information from both +ve and -ve impact parameter tracks. (See ALNEWS 1389 in OFFLINE folder). If one omits this card, however, then in the same way as one can measure the uds efficiency of QIPBTAG using the -ve hemisphere tag NPROBHEMI, it is possible to create a -ve tag from QBMTAG for the same purpose. This can be done by calling it with argument NPROBTRK instead of PROBTRK.

Please send all remarks concerning this package to : Ian.Tomalin@cern.ch

12.8 QVSRCH : Secondary vertices and b-tagging

```
CALL
QVSRCH(BPOS,BSIZ,PVTX,EPVTX,DJET,BTAG,SVTX,ESVTX,
AVTX,CAVTX,JERR)
```

This routine from T. Mattison performs a primary vertex finding , a 2-jet finding and a secondary vertices finding with an estimation of b quark tagging variables .

To use it properly , one must read the complete description of the method and of the package in the Aleph Note 92−173 (December 1992).

QVSRCH cannot be used when reading a Nano-Dst, since it needs the track error matrices.

Input arguments:

BPOS(3) Beam position. BPOS(2) must be an accurate beam y coordinate, i.e., QVYNOM or QVTXBP(2) (output from GET_BP). BPOS(1),BPOS(3) are used for X and Z track cuts and as additional constraints on primary vertex

BSIZ(3) Size of luminous region (1 sigma). Fixed values of .0200, .0020, 2.0 will work OK. Any negative element of BSIZ will suppress internal primary finding, in which case PVTX() and EPVTX() become input variables.

Output arguments:

PVTX(3) Primary vertex in Aleph XYZ coordinates

EPVTX(3) Error (not variance) in PVTX

DJET(3,2) Two normalized XYZ direction vectors for two jets

BTAG(2) B-tagging variable for the two hemispheres it is basically the half the chi-square difference between the hypothesis that all tracks in the hemisphere come from primary, or that some tracks come from the secondary and some from primary a cut at 10, or at 20 on the sum, is reasonable

SVTX(3,2) Two secondary vertices in coordinate systems that are used internally, (one system per jet).

- Origins are at PVTX(), and are oriented by DJET(,)
- The third coordinate is the decay length in the jet direction
- The first coordinate is R-Phi-like, the second is Z-like

ESVTX(3,2) Errors on SVTX() in the internal rotated system where geometrical correlations are small and ignored . May be negative, indicating a problem, not necessarily severe, in finding the vertex

AVTX(3,2) Two secondary vertices in ALEPH XYZ coordinates

CAVTX(3,3,2) Two covariance matrices for AVTX()

JERR Error flag

- 0 All OK
- +1 for secondary vertex 1 errors questionable
- +2 for secondary vertex 2 errors questionable
- +3 for questionable errors on both secondary vertices
- +4 for no good tracks in jet 1
- +8 for no good tracks in jet 2
- +12 for no good tracks in either jet
- +16 for questionable primary vertex
- +32 for not enough tracks for primary vertex
- +64 for no jets found

QVSRCH performs an internal jet finding. If you don't want it, a data card:

NQVJ

will suppress the internal jet finding, in which case DJET becomes an input variable which must be filled in by the user.

12.9 QPI0DO: π^0 finding routine

The π^0 finding routine QPI0DO is described in the report ALEPH 93−095 / PHYSIC 93−078. It uses the photons from the ALPHA GAT section (8.3.2 on p. 61).

To use QPI0DO, one has simply to put in QUEVNT:

```
CALL QPI0DO
```

Input argument: none

Output arguments:

All output arguments are returned in the ALPHA common deck GAMPI0:

```
   PARAMETER (MXPI0=200)
   COMMON / GAMPI0 / IQPI0, NTPI0, PI0MOM(4,MXPI0),ITYPI0(MXPI0),
  +                  IPI0GAM(2,MXPI0),CHIPI0(MXPI0)
```

IQPI0 Return code

- $= 0$: All OK
- $= 1$: No π^0 found
- > 1 : More than MXPI0 π^0 found

NTPI0 Number of π^0 found

PI0MOM Refitted 4 momentum of the NTPI0 π^0

IPI0GAM Number of photons 1 and 2 giving the π^0 , in the GAT section

CHIPI0 Chi2 value after refit. Set to -999. if no convergence.

ITYPI0 π^0 type, see below:

- =1 : 2 photons in same PECO, with N=2 photons in the PECO
- =2 : 2 photons in same PECO, with N>2 photons in the PECO
- =3 : 2 photons in different PECO, with N=1 photons in each PECO
- =4 : 2 photons in different PECO, with N>1 photons in one PECO

QPI0DO uses automatically the best photons from the GAT section. For POTs, DSTs and MINIs ≥ 10 the PGAC photons (sec 8.3.3) are used. For MINIs ≤ 9, these are the PGPC photons (sec 8.3.2).

Beware that the mass cut used to find the π^0 candidates in QPI0DO photons is quite wide: this mass window is $\pm 2\sigma$ around the mean GAMPEX reconstructed π^0 mass and is parametrised in the internal subroutine PI0LIM. For some specific analyses, a tighter cut might be needed, in that case one has to come back to the original GAMPEX photons and check their invariant mass.

Five control histograms 9001 to 9005 can optionally be filled. They give a display of the $\gamma - \gamma$ invariant mass versus γ energy for all photon pair types ITYPI0 (see above), in scatter plot 9000+ITYPI0 +1 , plot 9001 giving the sum over all types.

To get these histograms , you need simply to put:

CALL QPI0BK

in your QUINIT subroutine.

A debug printout routine is also provided to print all π^0s found in an event; to call it, put:

CALL PI0DEB

in your QUEVNT subroutine.

12.10 QBEAMX : Size of luminous region (LEP 1 only)

CALL QBEAMX(KRUN,KEVT,IRET,SIGX)

The ALPHA array QVTSBP (section 6.4) provides an average horizontal beam size sigma_x for each year. QBEAMX is designed to provide run/event-dependent sigma_x values for the real data and to provide corresponding information for Monte Carlo. The data is divided into "metachunks" of typically 275 qqbar events (about four beam position "chunks") for the measurements. The numerical values are read automatically by ALPHA at initialisation time and put in internal banks VBWP,WIDE and WIDN .

To use QBEAMX, one must make the above subroutine call during the event analysis stage .

KRUN	Input argument : run number
KEVT	Input argument : event number
IRET	Return code (=0 if successful). Notice that QBEAMX will not succeed if XGETBP (section 6.4) is .FALSE..
SIGX	Output rms size (in cm) of the luminous region in the x direction.

Real data: The measured value of SIGX is taken from the WIDE bank. A bias correction (from the database bank VBWP) is applied.

Monte Carlo: The value of sigma_x thrown in the subroutine QFGET_BP is recovered by means of the entry QFMCSX. NEVE, the number of events in the metachunk, is thrown according to the distribution in the real data, as found in the WIDN bank. The value of sigma_x is smeared according to the resolution (as a function of NEVE, with parameters in VBWP) and the result is returned in SIGX. Thus the data and Monte Carlo should be directly comparable as far as SIGX is concerned.

12.11 QSELEP : Lepton Identification for Heavy Flavours

QSELEP is a general lepton identification package for Heavy Flavour Physics. The corresponding physics results are described in the paper:

" Heavy quark tagging with leptons in the ALEPH detector "

Nucl. Instr. Meth. **A 346** (1994) pp 461 – 475.

QSELEP is built basically from the LEPTAG package (note ALEPH 92-101 / PHYSICS 92-090) written by Mark Parsons , adapted to ALPHA by Ingrid Ten Have and M-N Minard in November 1994. However it contains also most features of HEVLEP, the physics results and properties of which are described in detail in the note ALEPH 94-055 / PHYSIC 94-049. Due to technical reasons, the CALPOIDS package, auxiliary to HEVLEP, is not included into QSELEP and is still on UPHY.

The Monte-Carlo part of QSELEP: QTRUTH, is described in the next section (12.12).

12.11.1 How to call QSELEP

You don't have to call explicitely the QSELEP routine: either you get the results directly from MINI banks, or you trigger its execution through data cards as explained in 12.11.4.

When reading a MINI version ≥ 10:

If you are happy with the standard cuts used to build the MINIs, you have nothing to do: the QSELEP results are available directly through the ALPHA variables described afterwards in sec 12.11.6 to 12.11.8.

If you want to use tighter cuts, please refer to sec 12.11.4. This may be useful e.g. to use the CALPOIDS weighting (in UPHY).

When reading a POT or a DST or an old MINI:

If you want to call QSELEP yourself on a POT or DST (or on old MINIs made with MINILIB version 9.0 and before): you have to put in your CARDS file the data cards described in sec 12.11.4. You don't have to call QSELEP in your Fortran program: the call and all the necessary initialisations will be triggered by the QLID data card.

Then you may access the tagged leptons informations by the new ALPHA statement functions described below in sec 12.11.6 to 12.11.8.

12.11.2 Important warning if you call QSELEP yourself:

If you call QSELEP when reading a POT/DST, or when reading a MINI with "non-standard" cuts (see sec 12.11.4) be aware that:

QSELEP RESETS ALL ALPHA TRACK LOCKS and uses them internally. It uses also internally ALL ALPHA TRACK USER WORDS. Therefore you must be very careful if your analysis needs these locks and/or user words, and if you call QSELEP through the QLID card and user's cuts.

12.11.3 Output of QSELEP

The output of QSELEP consists of two banks PDLT,PLJT for real data, plus an additional one PMLT for MCarlo data. These banks are made using the default cuts described in 12.11.4 and are written on MINIs versions ≥ 10. They are also available on NANOs versions ≥ 115.

User access to these banks is done through the ALPHA variables and statement functions described below in 12.11.6 to 12.11.8.

When a MINI version ≥ 10 is written, all the cuts used by QSELEP are written onto a run header bank, PLSC. This bank is used to set the default cuts for QSELEP when reading this MINI with ALPHA.

12.11.4 Default cuts and users cuts

If you are not happy with the standard cuts used when running QSELEP to make a MINI (version ≥ 10) , or if you want to create the QSELEP information from a POT/DST/old MINI: you have first to put in your CARDS file the two data cards

QLID

EFLJ

If you don't provide any of the other data cards below, QSELEP will not be called when reading MINIs ≥ 10 since the cuts are the same as used to build it. If you run on a POT/DST/old MINI, QSELEP will be called using the default values described below.

If you want to use other cuts, you have to put in your CARDS file any combination of the data cards described below. The data cards described below correspond to the default values used to produce MINIs. Comments are self-explanatory.

```
*
*--  User card for general charged track selection
*
*-- * #tracks e_frac #tpc   d0    z0     cos(theta_max)
LSHD    5      0.10    4    2.0   10.0      0.95
*
*--   User card to select electron candidates in charged tracks
*--
*-- *  #tpc   d0    z0    cos(the_max)   pcut  p_tcut
LSET    5    2.0   10.0      0.95         2.0    0.0
*--
*--   User card electron identification cuts
*--
*-- *  rt_min rt_max rl_min rl_max #wires ri_min ri_max
LSEC   -3.0   999.0  -3.0   3.0      0    -999.  999.
*--
*--   User card rejection of photon conversions and Dalitz decays
*--
```

```
*-- * XY_max Z_max  M_max   W_conv(1=write,0=reject)
LSCV   1.0    1.0   0.020   1
*--
*--   User card to select muon candidates in charged tracks
*--
*-- * #TPC  d0    z0    cos(the_max)  pcut   p_tcut
LSMT    5   2.0  10.0      0.95       3.0     0.0
*--
*--   User card muon identification cuts
*--
*-- *  QMUIDO return flags accepted
LSMC   13  14
*--
*--   User card jet cuts
*--
*-- * #objects/jet  Max P(lepton)/E(jet) #jets   Ycut      Jrej
*--                                                      1=keep,0=cut
LSJT      5                0.9             2     0.004       1
```

12.11.5 Access to QSELEP results

In all circumstances :

- either if you have called QSELEP yourself through QLID + cards

- or if you read a MINI / NANO containing the QSELEP output banks

you can access the results of QSELEP through the ALPHA variables described in the following subsections .

12.11.6 Properties of selected tagged Leptons

When doing a loop on all reconstructed charged tracks :

DO ITK=KFCHT,KLCHT

you can access the QSELEP results as follows :

ALPHA Track ITK is a Lepton tagged by QSELEP if :

XLEPTG(ITK) = .TRUE.

If yes , then :

KLEPPA(ITK) QSELEP lepton type :

- 2 = e+
- 12 = e+ in crack region
- 22 = e+ in overlap region
- 3 = e-

- 13 = e- in crack region
- 23 = e- in overlap region
- 5 = mu+ QMUIDO IDF=14
- 15 = mu+ QMUIDO IDF=13
- 25 = mu+ QMUIDO IDF=11
- 35 = mu+ QMUIDO IDF=15
- 45 = mu+ all other QMUIDO flags
- 6 = mu- QMUIDO IDF=14
- 16 = mu- QMUIDO IDF=13
- 26 = mu- QMUIDO IDF=11
- 36 = mu- QMUIDO IDF=15
- 46 = mu- all other QMUIDO flags

KLEPJT(ITK) ALPHA "track" Number of nearest jet in LJET section of ALPHA The properties of this jet can be accessed directly by standard ALPHA functions

e.g. PXJET = QX(KLEPJT(ITK))

QLEPPI(ITK) Transverse momentum respect to the jet , lepton included

QLEPPE(ITK) Transverse momentum respect to the jet lepton excluded

KLEPVP(ITK) Flag giving the validity of the Pt calculation :

- 0 = OK
- 10 = No E-flow track found
- 20 = No valid jet
- 30 = Not enough objects in jet

12.11.7 QTRUTH flags for recontructed leptons (Mcarlo only)

When reading MCarlo events, results from the QTRUTH part of QSELEP can be accessed as follows :

When doing a loop on all reconstructed charged tracks :

DO ITK=KFCHT,KLCHT

ALPHA Track ITK is a MCARLO tagged lepton with QTRUTH results if :

XLEPTH(ITK) = .TRUE.

If yes , then :

KLEPFL(ITK) Primary quark Flavour of the event

1=d, 2=u, 3=s, 4=c, 5=b

KLEPPO(ITK) Flavour of the quark from gluon splitting

4 or 5 if the chain is originating from a c or a b quark respectively, which are coming from gluon splitting

0 otherwise

KLEPCH(ITK) H.F. process originating the decay chain

- -2 b → cbar u ; cbar → X
- -1 b → u X
- 1 b → c X
- 2 b → cbar c ; cbar → X
- 3 b → c → X
- 4 c → X
- 5 fragmentation or uds

KLEPSP(ITK) End of the decay chain

- 0 lepton from H.F. hadron
- 1 K → l
- 2 pi → l
- 3 gamma → l
- 4 J/psi → l
- 5 psi' → l
- 6 pi0 → l
- 7 other → l
- 8 K
- 9 pi
- 10 other

KLEPLE(ITK) Track identity

- -1 unidentified
- 0 non lepton
- 1 mu
- 2 tau → mu
- 3 electron
- 4 tau → e

KLEPME(ITK) Identity of the decaying b particle (if any)

- -2 "oscillated" Bs
- -1 "oscillated" Bd
- 0 b baryon or non b particle
- 1 Bd
- 2 Bs
- 3 B+

KLEPKT(ITK) FKIN track number of the lepton

12.11.8 Jets built by QSELEP : "LJET" section

As QSELEP builds its own jets , a new ALPHA section is built to access them through standard ALPHA calls . These jets are stored as standard ALPHA "tracks" in a new ALPHA "track" section .

Name of the Section : 'LJET'

- KLFJET = ALPHA "track" number of 1st QSELEP jet
- KLLJET = ALPHA "track" number of last QSELEP jet
- KLNJET = number of QSELEP jets
- KLJTNO(I) = number of objects inside QSELEP jet I

The access to the jet properties can be done either by a DO LOOP :

```
          DO IJET = KLFJET,KLLJET
             PXJET = QX(IJET)
             PYJET = QY(IJET)
             PZJET = QZ(IJET)
              EJET = QE(IJET)
              NOBJ = KLJTNO(IJET)
          etc..
```

```
or by an implicit loop on "particles" with name 'LJET' of class KRECO :
```

```
          IJET = KPDIR('LJET',KRECO)
    10    IF (IJET.EQ.0) GO TO 100
             PXJET = QX(IJET)
             .....
          IJET = KFOLLO('LJET')
          GO TO 10
    100   CONTINUE
```

12.11.9 New EDIR classes from QSELEP

When an output file is written in an ALPHA job which called QSELEP , two new EDIR classes are defined on the output class word in bank REVH

Class 27 : Electron selected (Rt $\geq -3.$, $-3. \leq$ Rl $\leq 3.$)

Class 28 : Muon selected (QMUIDO flag = 13 or 14)

Events with these EDIR classes may be accessed directly through the standard CLAS data card (4.1.5 on p. 15) when reading a MINI version ≥ 10 .

12.12 QTRUTH : History of a recontructed MCarlo track:

CALL QTRUTH(ITK,IFLA,IPOP,ICHAIN,ISPLIT,LEPID,IBMES)

QTRUTH is part of the QSELEP package described in the previous section. However it may be called independently, outside the context of QSELEP.

QTRUTH gives information about the "history" of a RECONSTRUCTED MCarlo track ITK. It works on Monte Carlo hadronic events generated with JETSET (or coming from an event generator interfaced with JETSET). The information provided is aimed to the classification of candidate leptons for heavy flavour tagging, but may be useful also in many other cases.

ITK Input argument : Alpha index of the reconstructed track

IFLA Primary quark Flavour of the event

1=d, 2=u, 3=s, 4=c, 5=b

IPOP Flavour of the quark from gluon splitting

4 or 5 if the chain is originating from a c or a b quark respectively, which are coming from gluon splitting

0 otherwise

ICHAIN H.F. process originating the decay chain

- -2 b → cbar u ; cbar → X
- -1 b → u X
- 1 b → c X
- 2 b → cbar c ; cbar → X
- 3 b → c → X
- 4 c → X
- 5 fragmentation or uds

ISPLIT End of the decay chain

- 0 lepton from H.F. hadron
- 1 K → l
- 2 pi → l
- 3 gamma → l
- 4 J/psi → l
- 5 psi' → l
- 6 pi0 → l
- 7 other → l
- 8 K
- 9 pi
- 10 other

LEPID Track identity

- -1 unidentified
- 0 non lepton
- 1 mu
- 2 tau → mu
- 3 electron

- 4 tau → e

IBMES Identity of the decaying b particle (if any)

- -2 ”oscillated” Bs
- -1 ”oscillated” Bd
- 0 b baryon or non b particle
- 1 Bd
- 2 Bs
- 3 B+

12.13 MCMATCH : Matching of a Monte Carlo truth track with a reconstructed track

CALL MCMATCH(IMCTRU,IRECTR,IFLAG)

The routine MCMATCH checks if a “truth” track at the FKIN (generator) level is reconstructed or not after GALEPH and JULIA. See the note of D.Brown ALEPH 94-135 (PHYSIC 94-118) for more details.

IMCTRU Input argument : Alpha index of a “truth” track (between KFMCT and KLMCT)

IRECTR Alpha index of the matching reconstructed charged track (between KFCHT and KLCHT)

= 0 if no matching found

IFLAG Flag describing the matching status:

- -3 No match (lost “truth” track)
- -2 Track decays before making hits
- -1 No well matching track
- 0 MC track double counted
- 1 Normal good match from primary vertex
- 2 Normal good match from secondary vertex
- 2 Normal good match from V0 decay
- 4 Kink fit as 2 tracks
- 5 Kink fit as 1 track
- 6 Normal good match from material interaction

12.14 JULMATCH : Matching of a reconstructed charged track with a MC “truth” track

CALL JULMATCH(IALTRK,IMCTRU,IFLAG)

The routine JULMATCH checks if a reconstructed track comes from a “truth” track at the FKIN (generator) level. See the note of D.Brown ALEPH 94-135 (PHYSIC 94-118) for more details.

IALTRK Input argument : Alpha index of a reconstructed charged track (between KFCHT and KLCHT)

IMCTRU Alpha index of the "truth" track (between KFMCT and KLMCT)

= 0 if no matching found (IFLAG described below is -2 or -3)

IFLAG Flag describing the matching status:

- -3 Hits not related to MC true hits
- -2 Hits mixed from several MC true particles (not decays)
- -1 Spiral branch or calorimetric reflection
- 0 Double counted particles (2 tracks for 1 true particle)
- 1 True track comes from true primary
- 2 True track comes from HF decay
- 3 True track comes from neutral decay
- 4 True track comes from charged decay (split kink)
- 5 Kink fit as 1 track
- 6 True track comes from material interaction
- 7 Unknown

12.15 VDHMATCH : Counting of matching VDET hits for a given track

```
CALL VDHMATCH(IALTRK,IMCTRU,NTRUE,NWRONG,NMISS,NDOUBLE)
```

The routine VDHMATCH must be called only AFTER a call to JULMATCH. See the note of D.Brown ALEPH 94-135 (PHYSIC 94-118) for more details.

This routine cannot be called when running on a MINI, since it needs the detailed VDET hits which are not stored on the MINIs.

IALTRK Input argument : Alpha index of a reconstructed charged track (between KFCHT and KLCHT)

IMCTRU Input argument : Alpha index of the matching "truth" track as given by JULMATCH (must not be 0)

NTRUE Number of true VDET hits on this track

NWRONG Number of true hits on the reconstructed track which truly came from a track other than IMCTRK (split z/rϕ)

NMISS Number of true VDET hits not used in reconstruction (split z/rϕ)

NDOUBLE Number of true hits on the reconstructed track which are associated with 2 or more true Monte Carlo tracks (split z/rϕ)

12.16 QWHICH_EN: to know how the LEP energy QELEP was found

CALL QWHICH_EN(ISTAT)

The output argument ISTAT may be used to know how the center-of-mass LEP energy QELEP was found. For most LEP2 events, this energy is defined in 15 minutes time slices by the LEP Energy Working group. It is available only for high energy runs of 1996 and after. See the note of P.Bright-Thomas ALEPH 97-029 (PHYSIC 97-024) for more details.

ISTAT Output argument: flag telling how QELEP was obtained.

- = 0 No QELEP energy found! This should NEVER happen !
- = 1 QELEP is obtained from the 'RLEP' bank. This is always the case for Monte Carlo events. For real data, this should be very rare, and only if all other methods described below have failed.
- = 2 QELEP is an averaged value over the current LEP fill. This is the case for 1989 to 1992 real data (from database bank LFIL), or for events in some very short runs taken in 1993 and after (from database bank RNF2).
- = 3 QELEP is an average value over the current run, taken from the database bank RNR2. This is the case for LEP 2 data if no time slice value was available for the event (mostly events taken before or after the official start/stop time of a LEP fill, as recorded in the LEP Energy working group tables).

  For LEP 1 data from 1993 to 1995 included, the values are run averages computed from the LEP Working Group values.
- = 4 QELEP was obtained from the 15 minutes time-slice energy determinations for ALEPH, from the LEP energy working group, stored in the database bank RNL2. These time-slices exist only for LEP2 High Energy data for 1996 and after.

12.17 QWHICH_BP: to know how the beam spot was found

CALL QWHICH_BP(ISTAT)

The output argument ISTAT may be used to know how the beam spot was found (see 6.4 on p. 35).

ISTAT Output argument : flag telling how the beam spot parameters were obtained.

- = 0 No beam position found! This should NEVER happen !
- = 1 obtained from the run header bank JSUM. This should be very rare, and only if all other methods described below have failed.
- = 2 obtained from an averaged value over the current LEP fill (database bank LFIL). This is the case only if the two methods described below have failed.
- = 3 obtained from an averaged value over the current run (database bank RXYZ). This happens only for LEP1 data in old processings (JULIA version below 300) if no event-chunk position exists.
- = 4 obtained from the event-chunk beam positions (run header bank ALPB).

12.18 QFILBP_STATUS: to know how the LEP beam position with BOMS was found:

```
CALL QFILBP_STATUS(ISTAT)
```

The output argument ISTAT may be used to know how the LEP beam position with BOMS was found. This implies that the data card BOBS was used (see 4.16 on p. 24).

ISTAT Output argument : flag telling how the LEP beam position with BOMS was found.

- = 0 No standard chunk-by-chunk beam position (XGETBP is FALSE.)
  All positive values below are obtained for XGETBP = .TRUE.
- = 1 VDET (x) VDET (y)
- = 2 BOM+QSO (x) VDET (y)
- = 3 VDET (x) BOM+QSO (y)
- = 4 BOM+QSO (x) BOM+QSO (y)
- = 5 Average (x) VDET (y)
- = 6 VDET (x) Average (y)
- = 7 Average (x) Average (y)

12.19 QEWSUM : ECAL Wire Energy on even/odd wire planes

```
CALL QEWSUM(EWEVEN,EWODD,IER)
```

The routine QEWSUM gives the energy on ECAL wires summed over even/odd planes. These quantities are available from POTs/DSTs but cannot be obtained from MINIs version 102 and before; on MINIs they are available only for datasets written with MINI version 103 and after, since May 1997.

EWEVEN(36)	Output array: sum of wire energies on even wire planes for the 36 ECAL modules
EWODD(36)	Output array: sum of wire energies on odd wire planes for the 36 ECAL modules
IER	Output argument : = 1 if no banks of wire energies for this event = 0 if all OK

In the arrays EWEVEN and EWODD above, ECAL modules $1 - 12$ refer to endcap A, $13 - 24$ to the barrel, and $25 - 36$ to endcap B.

12.20 QPCORR : Correction of charged particle momenta ("Sagitta correction")

This function has been provided by I. Tomalin and introduced in ALPHA 122.44 in March 1998. This is the so called "sagitta correction". It corrects particle momenta for the effects of residual distortions in the central tracking detector. It depends upon $\cos(\theta)$ and the year of data taking.

Every year, even after the detector alignment is finished and corrections have been made for field distortions etc., it is found that Ebeam/P in $Z0 \rightarrow \mu^+\mu^-$ events is not precisely 1, presumably because of residual distortions. (Typically, in the region $|\cos(\theta)| > 0.9$, Ebeam/p is about 0.94 for +ve tracks and 1.06 for -ve tracks. Elsewhere, Ebeam/p is usually consistent with 1 to within a percent or so. The effect is not quite forward-backward symmetric).

The relative bias in momentum is proportional to P, so most people analysing hadronic events can ignore it. Exceptions include analyses using the ECAL electron identifiers in the region $|\cos(\theta)| > 0.9$ or analyses which are very sensitive to systematic biases in the momenta (e.g. jet charge, τ polarization).

QPCORR provides a correction for the momenta based upon Ebeam/P measurements in $Z0 \rightarrow \mu^+\mu^-$ events. It assumes that the corrections for -ve and +ve particles are equal in size, but of opposite sign. This is observed to be true, apart from a constant offset, Ebeam/p = 1.002, which is also present in the MC and so not corrected for.

In principle, the correction depends upon your track selection cuts, but providing that the corrections have a small effect on your analysis, you can ignore this. Arguement NVDET does correct for this to first approximation however.

FUNCTION QPCORR(ITRK,ASCALE,NVDET,ERCORR)

Input arguments:

ITRK Integer: ALPHA track number.

ASCALE Logical: Set .TRUE. if ALPHA variables like QX,QY,QZ are to be rescaled by calling QVSCAL.
N.B. Don't call this routine twice for the same track with ASCALE = .TRUE. !

NVDET Integer: If your analysis only uses tracks with at least 1 VDET hit, set NVDET=1; otherwise =0.

Output arguments:

QPCORR Real : Scale factor applied/to be applied to momentum.

ERCORR Real : Statistical error on this factor.

Important remark: data card PCOR allows to call QPCORR automatically:

PCOR 0 Call QPCORR for all charged tracks

PCOR 1 Call QPCORR only for charged tracks having VDET coordinates

Chapter 13

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

13.1 Program termination

CALL QMTERM ('any message')

Can be called from anywhere.

Calls QUTERM, QUTHIS, QWMESS.

Input argument:

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

13.2 Write the current event on the output file

CALL QWRITE

The file name is specified on the FILO card (see 4.1.4). This routine can be called from user routines; it is called automatically from QMEVNT if the COPY option (4.1.7) is selected. If QWRITE is called more than once for the same event, the event will be written only once.

13.3 Set classification word written to event directory

CALL QWCLAS (IBIT)

Input argument:

IBIT	Turn on bit IBIT in classification word. IBIT = $1 - 30$.

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

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

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

13.4 Timing

13.4.1 Print job time consumption

CALL QWTIME

Called automatically from QMTERM.

13.4.2 Measure time consumption of part of program

CALL QTIMED(INUM)

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

Example:

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

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

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

13.5 Print routines

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

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

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

More options can be given for specific subroutines.

13.5.1 Print a message

```
CALL QWMESS ('any message')
```

Input argument:

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

13.5.2 Print a message plus run, event number

```
CALL QWMESE ('any message')
```

13.5.3 Print full event summary (many pages)

```
CALL QWEVNT
```

Calls QWHEAD, QWSEC, QWTREE

13.5.4 Print event header (one line)

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

Input arguments:

- **'option'** one of 'H', '0', or '1' (see 13.5)
- **'any text'** message; may be blank space: ' '
- **Output** see printer output of QWHEAD called with option 'H'. Here, as in many other print routines, it's a matter of taste which data are important enough to be printed, and comments are welcome. For better readability, the output should always fit onto one printer line.

13.5.5 Print full event header (many lines)

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

Subroutine arguments are the same as for QWHEAD.

13.5.6 Print information for "track"

CALL QWITK (ITK, 'option')

Input arguments:

ITK ALPHA track number.

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

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

Meaning of column "det. data":

F	general track fit data are available
T	dE/dx data are available
H	HCAL data are available
M	muon chamber data are available
E	Ecal data are available
H	Hcal data are available
... rightmost characters:	
C	object is associated to one or several charged tracks
E	object is associated to one or several Ecal objects
H	object is associated to one or several Hcal objects

13.5.7 Print information for vertex

CALL QWIVX (IVX, 'option')

Input arguments:

IVX ALPHA vertex number.

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

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

13.5.8 Print summary for categories of tracks or vertices

CALL QWSEC (ISEC, 'option')

Calls QWITK, QWIVX

Input arguments:

ISEC section number = section in QVEC and QVRT:

KSOVT	Overlap objects
KSCHT	Charged tracks
KSIST	Isolated = neutral cal objects
KSAST	Cal objects associated to charged tracks
KSV0T	Neutral tracks pointing to reconstructed vertices
KSDCT	Tracks outgoing from reconstructed vertices
KSEFT	Energy flow objects
KSNET	Neutral objects from PCPA
KSGAT	Photons from GAMPEC
KSJET	Jets from energy flow objects
KSMCT	MC particles
KSREV	Reconstructed vertices
KSMCV	MC vertices.

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

13.5.9 Print decay tree of track ITK.

CALL QWTREE (ITK, 'option')

Input arguments:

ITK Track / particle number.

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

Output: Similar to CALL QWITK.

13.5.10 Create LaTeXsource of the decay tree of track ITK.

CALL TXTREE (MOTHER,option)

Input arguments:

MOTHER ALPHA Track number where the tree will start.

option if option='TRACK#' the ALPHA track number will be displayed as well as the particle names; otherwise only the particle code is displayed.

Output: The output consists of a file with name txtree.tex, in the user's working directory on UNIX and on the SCWEEK area on AXAL.

Example of use of this routine :

```
C Find the ALPHA track number of a J/PSI :
      NPSI=KPDIR('JPSI',KMONTE)
C  Find the mother of this J/Psi :
      MOTH=KMOTH(NPSI,1)
C Produce the tree with labels only :
      CALL TXTREE(MOTH,' ')
C Produce the tree with labels and track numbers :
      CALL TXTREE(MOTH,'TRACK#')
```

Chapter 14

Modifying ALPHA banks

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

14.1 User track / vertex sections

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

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

$\boxed{\text{CALL QSUSVX (NUSVX)}}$

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

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

14.2 Modifying track / vertex attributes

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

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

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

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

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

Chapter 15

Particle Table

15.1 Description

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

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

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

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

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

15.2 Particle name, particle code

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

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

15.3 How to spell particle names

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

Example

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

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

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

15.4 Data cards for particle table

15.4.1 PMOD: Modify particle attributes

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

Parameters:

'part−name antipart−name' see 4.13.1. The attributes of a particle and its antiparticle are modified at the same time. If a particle is its own anti− particle, the same name has to be given twice.

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

Example:

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

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

Mistake:

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

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

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

- new table entries are created;

- a warning is issued;
- the program execution continues.

15.4.2 PNEW: New particles

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

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

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

program execution continues in either case.

15.4.3 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:

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

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

Parameters:

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

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

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

The routine QCPTRA is equivalent to the PTRA card and can be called from QUNEWR 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.

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

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

Warning : All above functions which have a particle name as input argument are case−insensitive .

Appendix A

Program Structure

```
      QMAIN                 main program
      |
      +---- QMINIT          PROGRAM INITIALIZATION
      |       |
      |       +--  QUIBOS   init BOS
      |       +--  QUIHIS   init histogram package
      |       +--  QMALPH   initialize ALPHA
      |       |
      |       +--  QUINIT   user initialization             <---
      |
  +->-+---  QMREAD          READ EVENTS
  |   |       |
  |   |       +-- ABRSEL    open input files; read next record
  |   |       |
  |   |       +-...-> QUNEWR called for every new run
  |   |       |
  |   |       +-- QMTERM    terminate job if eof or time limit or ...
  |   |
  |   +---   QMEVNT         PROCESS ONE EVENT
  |   |       |
  |   |       +-- QFILL     fill ALPHA variables
  |   |       |
  +-<-+       +-- QUEVNT    user event analysis             <---
```

```
called from anywhere :

            QMTERM          PROGRAM TERMINATION
              |
              +--  QUTERM   user termination                <---
              |
              +--  QUTHIS   output histograms
              |
              STOP          Fortran STOP
```

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

Appendix B

Bank description

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

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

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

Basic attributes

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

Flags

 8   TN   I   JULIA / GALEPH Track Number
 9   SC   I   stability code
10   KS   I   LUND status code
11   CL   I   track CLass
12   PA   I   ALEPH particle code
13   QD   I   offset for corresponding row in QDET (NOT the row number !)

14   NP   I   pointer to Next Particle / same particle code / same class
15   SP   I   Same Particle, different hypothesis or Lorentz frame
16   OV   I   Origin Vertex no
17   EV   I   End Vertex no


18   ND   I   Number of Decay particles
19   DL   I   offset of 1st daughter in particle List QLIN
20   NO   I   Number of mother particles
21   OL   I   offset of 1st mother in particle List QLIN
22   NM   I   Number of Matches
23   ML   I   Match list = row offset in banks QMTL and QMTS

24   BM   I   Lock Bit masks ( up to 448 tracks allowed )
```

```
38   LK   I   QLITK flag

39   DB   F   Distance of closest approach to beam axis (if track pointing
              to the main vertex) or to the corresponding secondary vertex.
40   ZB   F   Z coordinate of point where DB is measured.
41   SD   F   Error**2 on DB
42   SZ   F   Error**2 on DZ
43   CB   F   Chi**2 corresponding do DB and DZ.


Error Matrix

44-53 EM  F   triangular covariance matrix

54    CF  F   chi**2 from last kinematical fit
55    EW  F   weight from energy flow analysis

56-73 US      User space


*------*
| QDET |              DETECTOR INFORMATION
*------*

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

 1   AF   I   offset for corresponding row in bank FRFT (NOT row number !)
 2   AL   I   offset for corresponding row in bank FRTL
 3   NT   I   number of segments in bank 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 MCAD

12   CF   I   calorimeter flag (<0 : ass, > 0 : isol, abs=1 : ECAL; =2 : HCAL)
13   EC   I   offset for corresponding row in bank PECO
14   HC   I   offset for corresponding row in bank PHCO
15   ET   I   offset for corresponding row in bank PEPT
16   FI   I   offset for corresponding row in bank FRID
17   NF   I   number of associated charged tracks
18   FL   I   offset of 1st ass. ch. track in list QLIN
19   NE   I   number of associated ECAL objects
20   EL   I   offset of 1st ass. ECAL object in list QLIN
21   NH   I   number of associated HCAL objects
22   HL   I   offset of 1st ass. HCAL object in list QLIN
23   LH   I   overlap - associated
24   EF   I   offset for corresponding row in bank EFOL
25   PC   I   offset for corresponding row in bank PCQA
26   EG   I   offset for corresponding row in bank EGPC
27   MU   I   offset for corresponding row in bank MUID
```

```
28   DX   I   offset for corresponding row in bank
29   PG   I   offset for corresponding row in bank PGPC
30   PD   I   offset for corresponding row in bank PDLT
31   PM   I   offset for corresponding row in bank PMLT


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

 1   number of words / vertex
 2   maximum allowed number of vertices

 1   VX   F   XPosition
 2   VY   F   YPosition
 3   VZ   F   ZPosition
 4   VN   I   JULIA /GALEPH Vertex number
 5   TY   B   vertex TYpe


 6   IP   I   track number of Incoming Particle
 7   ND   I   Number of Decay particles
 8   DL   I   offset for decay particle list
 9   AY   I   offset for corresponding row in YV0V
10   AF   I   offset for corresponding row in FVER

11-16 EM  F   triangular error matrix

17    CF  F   chi**2 for vertex fit -- filled by KVFITN, KVFITV
18-25 ET  F   track-vertex error matrix
26-30         User space


*------*
| QFPA |              FIRST PARTICLE (FOR DIRECT ACCESS)
*------*

 1   number of track classes = 8
 2   maximum number of rows

 1   xx   I   ALPHA track number
              each row contains the 1st particle for a given class
              in order KRECO , KMONTE , Lorentz frames
              1 row per particle , pointed by QPLI


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

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

 1   xx   I   ALPHA track number
```

```
*------*
| QMTL |              MATCH LIST
*------*

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

 1   xx   I   ALPHA Track number


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

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

 1   xx   I   Number of shared hits


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

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

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


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

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

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


*------*
| QPLI |              POINTERS FROM QPAR TO QFPA
*------*

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

```
 1   xx   I   pointer to QFPA row for this particle


*------*
| 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

All the ALPHA code and documentation (the present user's guide) may be found on www in the ALEPH home page, (http://alephwww.cern.ch/): click on Offline Software, then on ALPHA Source Code Index (for the source code of ALPHA subroutines) or on ALPHA User's Manual (the present user's guide).

Important warning: It is strongly recommended to use always the current ALPHA production version, as given automatically by the alpharun facilities (see below). Using old versions may be dangerous: there may be incompatibilities, bugs or obsolete features which may be difficult to discover at first glance.

C.1 ALPHA on ALWS

Since version 122, ALPHA is not maintained any more on type VAX machines. All what follows refers to AXP machines only. Since the same version, there is no correction file any more: all modified routines are directly put into the binary library.

Most files needed to run ALPHA on ALWS are in the PHY directory:

ALEPH$GENERAL:[PHY]

- ALPHA source code:

  Since August 1996, ALPHA is maintained under CVS. The reference source code of ALPHA subroutines can be found as CVS files with name XXXXX.F in subdirectories of:

  ALIBROOT:[000000.PHY.ALPHAvsn]

  The relevant subdirectories for subroutines are: PACK, UTIL, QFN, ENFL, QVS, QLEP, QPI0, NANO.

  For the sample user routines QUINIT, QUEVNT, QUTERM, the subdirectory is PACK.

  For include files (QCDE,QMACRO,QDECL,QHAC) the subdirectory is INC.

- Fortran files, if you still want to use them:

  PHY:QUUSER.FOR Fortran code of the routines QUINIT,QUEVNT,QUTERM (model routines which have to be filled)

- Include files for Fortran:

  PHYINC:QCDE.INC
  PHYINC:QDECL.INC
  PHYINC:QHAC.INC
  PHYINC:QMACRO.INC

- ALPHA binary libraries:

  PHY:ALPHAvsn.OLB ALPHA binary library.
  PHY:ALPHAvsn_D.OLB with /DEBUG option.

- ALPHA card file (sample)

  PHY:ALPHA.CARDS

- ALPHARUN command file (see Ch. 2)

  PHY:ALPHARUN.COM options stored in ALFPARAM.OPTB (by default)

  ALPHARUN is invoked directly by typing alpharun. It allows to build an ALPHA job, either interactive or BATCH, and links automatically the user's program with all necessary libraries (ALPHA, BOS, ALEPHLIB, JULIA, CERNLIB).

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

  Examples

EXAMINE IW(512:515)	Standard VAX debug command; to be used for all Fortran variables and arrays.
EVALUATE LMHLEN	LMHLEN is a parameter and NOT a variable; use EVA instead of EXA.
QP(ITK)	Debugger commands are defined for almost all mnemonic symbols which have one or more arguments (see Ch. 6). In this context, "QP" is a debugger command which has to be followed by the same argument(s) (given as numbers or variable names) as the mnemonic symbol QP in Fortran.

- ALPHA documentation and News

  ALEPH$GENERAL:[PHY.DOC]ALGUIDE.PS PostScript file to print the present document

  ALEPH$GENERAL:[PHY.DOC]ALGUIDE.TEX LaTeXsource for this document

  PHY:ALPHAvsn.NEWS Description of changes in ALPHA version vsn

  PHY:ALPHA.VERSIONS Short description of the various versions of ALPHA and related ALEPHLIB / MINI versions

C.2 ALPHA on SHIFT,CSF,ALPHA OSF

Reminder :

1 - since ALPHA version 122, all corrections to the current default version are put directly in the ALPHA library; there is therefore no source or compiled correction file.

2 - since ALPHA version 124 (May 1999) the MINI code is fully integrated within ALPHA, both for reading and for writing MINIs.

To run ALPHA on UNIX machines, it is **strongly** recommended to use **alpharun**, which is described below (Sec C.3). It allows either to run ALPHA interactively or to send it in Batch on a given machine, as did previouly SFALPHA which is now obsolete.

The following files refer to the current ALPHA version.

- ALPHA source code:

  Since August 1996, ALPHA is maintained under CVS. The reference source code of ALPHA subroutines can be found as CVS files with name xxxxx.F (Caution! xxxxx must be lower case!) in subdirectories of:

  $ALROOT/alphavsn (e.g. for ALPHA 124 vsn=124 and the directory is $ALROOT/alpha124)

  The relevant subdirectories are: inc,news,pack,util,qfn,enfl,qvs,qlep,qpi0,mini,nano.

- ALPHA include files:

  The reference source code of ALPHA include files is available as CVS files with name yyyy.h (Caution! yyyy must be lower case!) in $ALROOT/alphavsn/inc (for ALPHA 124, vsn=124)

  To find e.g. the path to the source code of QMACRO for ALPHA 124:

  cd $ALROOT/alpha124

  find . -name "qmacro.h"

- Binary library:

  $ALEPH/phy/libalpha.a Since May 1999 (ALPHA124) this library contains all MINI routines.
  There is no ALPHA debug library any more on UNIX machines (too big!).

- ALPHA card file (sample)

  $ALEPH/phy/alpha.cards

- alpharun (see Ch. 2)

  alpharun shell file on all UNIX machines
  See more details on alpharun just below.

- ALPHA documentation and News

  $REFERENCE/phy/doc/alguide.ps PostScript file to print the present document

$REFERENCE/phy/doc/alguide.tex LaTeXsource for this document, and all related files

$ALEPH/phy/alpha.news Description of changes in the current version of ALPHA

$ALEPH/phy/README Short description of the various versions of ALPHA and related ALEPHLIB / MINI versions

C.3 alpharun: run ALPHA on UNIX platforms:

alpharun is a facility intended for people who want to run ALPHA on UNIX machines. It compiles your program with the adequate compiler options, generates automatically the appropriate links to all needed libraries (BOS, ALEPHLIB, JULIA, MINI etc.), and builds the executable through a makefile which you can edit afterwards. Then **alpharun** executes your ALPHA job either interactively or in batch.

To execute **alpharun**, you have just to prepare your Fortran analysis program and your Cards file, then to type **alpharun** and answer the questions.

If you choose to run in Batch, your job will be sent to the machine you have chosen; the OUTPUT logfile is returned to the original computer in your working directory. If you create an output EPIO file through a FILO data card, it will be also returned to the original computer if you have put the DISP parameter in the FILO card (see description of the FILO card, sec 4.1.4 on p. 12). Similarly, if you create an output .HIS histogram file through a HIST data card, it will be also returned to the original computer if you have put the DISP parameter in the HIST card (see description of the HIST card, sec 5.2.1 on p. 31).

Warning! If you write yourself a private output file, you must take care yourself of getting it back from the computer where your job has run in Batch.

It is also possible to run **alpharun** with options, which are described in the **man alpharun** and are summarised below:

```
alpharun [-F filename]
alpharun [-f fortran_file] [-c c_file] [-C card_file] [-g] [-j]
 [-v version] [-o obj_file] [-s file_opt] [-p] [-r]
 [-I include_path] [-m machine] [-t time_limit] <binary_name>
```

```
Where:
    -help            get this (h)elp
    -F <string>      expected (F)ilename which alpharun options
    -f <string>      expected (f)ortran file  (my_fortran.f)
    -c <string>      expected (C) file  (my_cfile.c)
    -o <string>      expected (o)bject file  (my_objfile.o)
    -C <string>      expected  C ard file  (my_cards.cards)
    -g               for debug options to compile
    -j               (j)ulia is used
    -I <string>      expected (I)nclude path
```

```
-b                to run in (b)atch
-m <string>       expected machine to submit your job to
-t <number>       (t)ime limit of your batch job in seconds
-p                (P)VM is used
-v <number>       (v)ersion of alpha
-s <string>       expected name of the file to (s)ave the options
-r                (r)uns the binary after creating it
  <binary_name>   expected name for the binary
```

Appendix D

Using the Mini-DST with ALPHA

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

Since ALPHA version 124 (May 1999), the MINI code is fully integrated within ALPHA, both for reading and for writing MINIs. All MINI routines are now in the ALPHA library.

The source code for all MINI routines can be found in the directory:

$ALROOT/alphavsn/mini (vsn: current ALPHA version number, e.g. 124).

An executable to make MINIs with all standard banks is available in:

$ALEPH/phy/miniprod

Important warning: Do not use old MINI libraries any more. Please use now only the ALPHA library which contains all the necessary MINI features.

D.1 Doing analysis with the Mini

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

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

- XTCN: all
- FRFT: KFRFNO.
- FRTL: KFRTNE, KFRTNR.
- FRID: KFRIBC, KFRIBC, KFRIPE, KFRIPM, KFRIPI, KFRIPK, KFRIPP, KFRINK.
- EIDT: QEIDEC, QEIDRI(I,N=1,4,5).

- PECO: QPECEC, KPECKD, KPEPPC.
- PEPT: all.
- PHCO: QPHCER, KPHCKD, KPHCCC, KPHCRB, KPHCPC.
- YV0V: KYV0IC.
- EGPC: QEGPR1, QEGPR2, QEGPF4, QEGPDM, KEGPST. Note that EGPC is obsolete since MINI 9.0 .
- EFOL: KEFOLC.
- Vertex: QVEM.

The energy flow stored on the Mini-DST corresponds to the ENFLW algorithm; PCPA was available also until MINI 8.6 .

All of ALPHA's event topology routines work with the Mini-DST. Also, the routines described in Ch. 12 may be used. In particular, QMUIDO, QDEDXM, QPAIRF, QVDHIT, and XVDEOK will work.

A few packages cannot run on the MINI, e.g. QEWSUM for datasets written with MINI 10.2 and before, or VDHMATCH for all MINIs.

D.2 Differences between POT/DST and Mini-DST

In addition to the absence of some information, other differences may be observed when using the Mini. The Mini-DST has limited precision, so some quantities will have slightly different values from those found when reading the POT or DST. Non-trivial differences are:

- FRFT/0 is generally not available on the Mini-DST.
- For ITC tracks, the z_0 error is limited to 25cm, which increases QBC2 substantially.
- If the total number of vertices is greater than 30, only the main vertices and the best V^0 vertices (ordered by chi-squared) up to a total of 30 will be available when reading the Mini-DST.
- Further, the V^0 quantities available are the result of swimming the tracks, and if greater precision is required, a refit must be performed. Infrequently, problems arise when the charge of fitted track (FRFT/2) is different from that of the original track used (FRFT/0), and hence the two V^0 tracks appear to have the same charge.
- The chi-squared and NDF for the helix fit for charged tracks may not be the same as those found for the DST, but the chi-squared per NDF should be correct within the foreseen precision.
- The values of R2 and R3 are truncated in the range [-10.23,+10.23], but where there are indications of problems, the values are set to +1000.
- For old MINIs (8.6 and before) with GAMPEK photons, only the presence of dead stories is recorded, not information for each storey. Hence DST1,2,3 of the variable KEGPQU are all 0 or all 1.

D.3 Differences between LEP1 and LEP2 MINIs

As requested by the Physics Analysis groups, the MINIs for LEP2 datasets (both real data and Monte Carlo) contain several banks in addition to the ones written on LEP1 MINIs:

- used by EFLOW:
  PHCO,HTUB,PEWI,PWEI,EFOL,EIDT
- muon identification:
  MHIT,MCAD,HMAD,MUID,MUEX,MTHR,HPDI,HROA
- related to BCAL^{++}:
  BCTR,BCSC,BCSL,BCGN,BCHG
- HCAL calibration:
  HT0C,HCLB,HCCV
- HCAL slow control:
  HSSR,HCNF
- ITC coordinates:
  PIDI ($\rightarrow$ ITCO)
- LCAL crack PMs :
  LCRA
- SiCAL cluster identification:
  SIID
- several specific KINGAL banks:
  KWGT,KWTK,KSEC

The MINI-making program recognizes automatically if it reads LEP1 or LEP2 datasets and adds the above banks if the dataset is a LEP2 one.

D.4 Writing a Mini-DST

To write a private Mini-DST file from a DST (or POT), two options are possible. You may:

- either use the $ALEPH/phy/miniprod executable and the MINP data card:

If you do so you will obtain MINIs with the official ALEPH banks, especially those for LEP2 datasets. If you want the correct pad dE/dx for data, you must read input POTs, not DSTs.

Beware ! Using miniprod, you can't select your output events (**all** input events are written on the output file) and you can't modify the cuts for QSELEP,QIPBTAG, etc.. However, you may add standard ALEPH banks which are not foreseen in the official MINIs, using the MINA data card (see below).

- or write your own customised MINI:

In that case you can select your events and put (almost) all what you want on your MINI. The MINI-making program is an ALPHA job that you can submit with **alpharun**; you have to proceed as follows:

1. Read a POT (for Monte Carlo) or a DST (preferable and much faster for real data if you don't need the special pad dE/dx bank PTPX), or even a MINI, declared with a standard FILI card.
2. Declare an output file which will contain your Mini-DST with a FILO card.
3. ALPHA must be informed that you wish to create a Mini-DST with the data card:

   MINI
4. Invoke integer compression with COMP 'INTE' to save space (about a factor of two) for the output file.
5. If you want to write on your MINI some additional banks which are not ALEPH standard, you must define their format.

   Banks which are not ALEPH standard are banks which are not described in the documentation file:

   al$general:[doc.bankdoc]bankal.fmt on ALWS,

   $ALEPH/dbase/bankal.fmt on UNIX.

   To define the bank formats: in QUINIT, call the BOS routine BKFMT, one call per bank, see the BOS documentation.

   If you don't do so, these banks will be **irrecoverably unreadable** when you will try to read your private MINI afterwards.
6. you have to **give the list of all your non-standard banks, if any,** to the MINI-making program through a data card:

   MINA 'ban1ban2ban3...' where "bani' are the names (4 letters, as usual) of your private banks.

   MINA may be used also to add to your MINI standard ALEPH banks which are not put in the standard MINIs.
7. In your QUEVNT subroutine, after your analysis and selections, call QWRITE whenever you wish to output an event according to your selection criteria.
8. If you wish to create the ENFLW information on the MINI file, you must put an EFLJ data card.
9. You may define your own QSELEP cuts with the appropriate set of LSxx data cards (see 12.11.4 on p. 118).

When you have written a small sample of events, it is worth **checking** that these are readable before creating many events.

Appendix E

Using the NanoDst with ALPHA

The NanoDst is not any more an officially supported ALEPH dataset. Please refer to the descriptions given in previous editions of this User's guide (Appendix E) if you still want to use NanoDsts.

Appendix F

Standard particle table

1	gamma	2	e+	3	e-	4	nu	5	mu+
6	mu-	7	pi0	8	pi+	9	pi-	10	K0l
11	K+	12	K-	13	n	14	p	15	p#
16	K0s	17	eta	18	Lam0	19	Sig+	20	Sig0
21	Sig-	22	Xi0	23	Xi-	24	Ome-	25	n#
26	Lam#0	27	Sig#-	28	Sig#0	29	Sig#+	30	Xi#0
31	Xi#+	32	Ome#+	33	tau+	34	tau-	35	D+
36	D-	37	D0	38	D#0	39	Ds+	40	Ds-
41	Lamc+	42	W+	43	W-	44	Z0	45	Deuteron
46	Tritium	47	Alpha	48	Geantino	49	GeantMino	50	GeantBino
51	GeantAbino	52	GeantPhino	53	Lamc#-	54	Higgs	55	etab
56	K0	57	K0#	58	nue	59	nue#	60	numu
61	numu#	62	nutau	63	nutau#	64	a1+	65	a1-
66	a10	67	f0_975	68	a0_980	69	etapr	70	chic0
71	chic1	72	chic2	73	Jpsi	74	psipr	75	etac
76	rho+	77	rho-	78	K*+	79	K*-	80	K*0
81	K*#0	82	D*0	83	D*#0	84	D*+	85	D*-
86	Ds*+	87	Ds*-	88	rho0	89	omega0	90	Phi
91	chib0	92	chib1	93	chib2	94	Upsilon	95	Ups2s
96	etat	97	chit0	98	chit1	99	chit2	100	gluon
101	uquark	102	dquark	103	squark	104	cquark	105	bquark
106	tquark	107	uquark#	108	dquark#	109	squark#	110	cquark#
111	bquark#	112	tquark#	113	Theta	114	Thetapr	115	B-
116	B+	117	B0	118	B#0	119	Bs0	120	Bs#0
121	Bc-	122	Bc+	123	B*-	124	B*+	125	B*0
126	B*#0	127	Bs*0	128	Bs*#0	129	Bc*-	130	Bc*+
131	T0	132	T#0	133	T+	134	T-	135	Ts+
136	Ts-	137	Tc0	138	Tc#0	139	Tb+	140	Tb-
141	T*0	142	T*#0	143	T*+	144	T*-	145	Ts*+
146	Ts*-	147	Tc*0	148	Tc*#0	149	Tb*+	150	Tb*-
151	Delta++	152	Delta#--	153	Delta+	154	Delta#-	155	Delta0
156	Delta#0	157	Delta-	158	Delta#+	159	Sig*+	160	Sig*#-
161	Sig*0	162	Sig*#0	163	Sig*-	164	Sig*#+	165	Xi*0
166	Xi*#0	167	Xi*-	168	Xi*#+	169	Sigc++	170	Sigc#--
171	Sigc+	172	Sigc#-	173	Sigc0	174	Sigc#0	175	Xic+
176	Xic#-	177	Xic0	178	Xic#0	179	Omec0	180	Omec#0
181	Xi0c+	182	Xi0c#-	183	Xi0c0	184	Xi0c#0	185	Sigc*++
186	Sigc*#--	187	Sigc*+	188	Sigc*#-	189	Sigc*0	190	Sigc*#0
191	Xic*+	192	Xic*#-	193	Xic*0	194	Xic*#0	195	Omec*0
196	Omec*#0	197	Xicc++	198	Xicc#--	199	Xicc+	200	Xicc#-
201	Omecc+	202	Omecc#-	203	Xicc*++	204	Xicc*#--	205	Xicc*+
206	Xicc*#-	207	Omecc*+	208	Omecc*#-	209	Omeccc++	210	Omeccc#--

211 Sigb+	212 Sigb#-	213 Sigb0	214 Sigb#0	215 Sigb-
216 Sigb#+	217 Xib0	218 Xib#0	219 Xib-	220 Xib#+
221 Omeb-	222 Omeb#+	223 Xibc+	224 Xibc#-	225 Xibc0
226 Xibc#0	227 Omebc0	228 Omebc#0	229 Omebcc+	230 Omebcc#-
231 Xibb0	232 Xibb#0	233 Xibb-	234 Xibb#+	235 Omebb-
236 Omebb#+	237 Omebbc0	238 Omebbc#0	239 Sigt++	240 Sigt#--
241 Sigt+	242 Sigt#-	243 Sigt0	244 Sigt#0	245 Xit+
246 Xit#-	247 Xit0	248 Xit#0	249 Omet0	250 Omet#0
251 Xitc++	252 Xitc#--	253 Xitc+	254 Xitc#-	255 Ometc+
256 Ometc#-	257 Ometcc++	258 Ometcc#--	259 Lamb0	260 Lamb#0
261 Xi0b0	262 Xi0b#0	263 Xi0b-	264 Xi0b#+	265 Xi0bc+
266 Xi0bc#-	267 Xi0bc0	268 Xi0bc#0	269 Ome0bc0	270 Ome0bc#0
271 Lamt+	272 Lamt#-	273 Xi0t+	274 Xi0t#-	275 Xi0t0
276 Xi0t#0	277 Xi0tc0++	278 Xi0tc0#--	279 Xi0tc+	280 Xi0tc#-
281 Ome0tc+	282 Ome0tc#-	283 Sigb*+	284 Sigb*#-	285 Sigb*0
286 Sigb*#0	287 Sigb*-	288 Sigb*#+	289 Xib*0	290 Xib*#0
291 Xib*-	292 Xib*#+	293 Omeb*-	294 Omeb*#+	295 Xibc*+
296 Xibc*#-	297 Xibc*0	298 Xibc*#0	299 Omebc*0	300 Omebc*#0
301 Omebcc*+	302 Omebcc*#-	303 Sigt*++	304 Sigt*#--	305 Sigt*+
306 Sigt*#-	307 Sigt*0	308 Sigt*#0	309 Xit*+	310 Xit*#-
311 Xit*0	312 Xit*#0	313 Omet*0	314 Omet*#0	315 Xitc*++
316 Xitc*#--	317 Xit*+	318 Xit*#-	319 Ometc*+	320 Ometc*#-
321 Ometcc*++	322 Ometcc*#--	323 Xitb*+	324 Xitb*#-	325 Xitb*0
326 Xitb*#0	327 Ometb*0	328 Ometb*#0	329 Ometbc*+	330 Ometbc*#-
331 Ometbb*0	332 Ometbb*#0	333 Xitb+	334 Xitb#-	335 Xitb0
336 Xitb#0	337 Ometb0	338 Ometb#0	339 Ometbc+	340 Ometbc#-
341 Ometbb0	342 Ometbb#0	343 Xi0tb+	344 Xi0tb#-	345 Xi0tb0
346 Xi0tb#0	347 Ome0tb0	348 Ome0tb#0	349 Ome0tbc+	350 Ome0tbc#-
351 Xibb*0	352 Xibb*#0	353 Xibb*-	354 Xibb*#+	355 Omebb*-
356 Omebb*#+	357 Omebbc*0	358 Omebbc*#0	359 Omebbb-	360 Omebbb#+
361 beame+	362 beame-	363 Charged	364 Ecal	365 Hcal
366 CALobj	367 last_st_par			

Appendix G

Definition of Event Directory Classes

EDIR classes for LEP1 and LEP 1.5 Data and Monte Carlo:

```
Class # 1:  More than 2 Ecal modules with energy(wires) >=2.5 GeV in each

Class # 2:  Hcal energy(pads) + Ecal energy(wires) > 15 GeV

Class # 3:  End cap A and End cap B both with energy(wires) > 2. GeV
            or Barrel with energy > 6 GeV

Class # 4:  Hcal energy(pads) > 3 GeV + HCW(4 planes) * ITC trigger

Class # 5:  1-->7 tracks with >=4 TPC hits, D0 <5 cm and Z0 <20 cm

Class # 6:  >=8 tracks with the same cuts

Class # 7:  LUM A and LUM B, both with E >15 GeV

Class # 8:  LUM A or LUM B, with E >15 GeV

Class # 9:  Muon (HMAD flag) with energy >3 GeV

Class #10:  Electron with momentum >=2 GeV
            electron candidates based on (-3.5>R2 , -3.5<R3<4.0)

Class #11:  ECAL high voltage ON

Class #12:  TPC high voltage ON (Logical or between bits 4 and 15
            of the high voltage word)

Class #13:  ITC  high voltage ON

Class #14:  LCAL high voltage ON

Class #15:  Dilepton candidates
---------
                    Selections based on TPC tracks only

    The requirements are less strict than those used in the common
    lepton analysis.

1. Only tracks with >= 4 TPC hits, |z_0| < 10 cm, |p| > 0.1 GeV/c are used.
   Their numbers are counted separately for |d_0| < 5 cm (including |d_0| <
```

2 cm), and for |d_0| < 2 cm.

2. a) If there are exactly 2 such tracks with |d_0| < 5 cm, they are declared as good tracks, and the selection continues.
   b) If there are between 2 and 8 such tracks with |d_0| < 2 cm, they are declared as good tracks, and the selection continues. Any track with 2 cm < |d_0| < 5 cm is declared as bad.

3. The thrust axis is calculated using a routine from JETSET. Then, each of the two hemispheres as defined by the thrust axis is required to contain at least one good track.

4. It is required that at least one track with |d_0| < 2 cm has a momentum exceeding 3 GeV/c. IT IS HERE WHERE 2 TRACK EVENTS, BOTH TRACKS HAVING 2cm < |d_0| < 5 cm, ARE LOST. It seems to me that this was not the original intention.

5. If there are more than 4 good tracks, each of them is required to have an opening angle eta with respect to the axis of the corresponding jet which fulfills cos(eta) > 0.85.

Events surviving all these requirements are flagged as class 15.

Class #16: QQbar candidates (based on TPC tracks information)

Selections based on TPC tracks only

Class 16 contains events with :

- At least 5 TPC tracks satisfying the following cuts :
  |D0| < 2 cm
  |Z0| < 10 cm
  TPC coordinates >= 4
  |cos(theta)| < 0.95
- in addition the total energy of all TPC tracks satisfying the cuts above, should have more than 10% of C.M. energy.

Class #17: QQbar candidates (based on calorimetry information)

Selections based on Calorimetry information

Class #17 candidats are:
- Events satisfying the Total Electromagnetic trigger:
  3 GeV in odd and even wire planes of Barrel or 0.75 GeV in odd and even wire of each electromagnetic End cap.
- They should be in time with beam crossing :
  T0 from Ecal wires used (abs(T0)<120 ns in End caps, abs(T0)<100 ns in Barrel)
- They should satisfy the total Energy cuts :
  Ecal wire energy + Hcal analog energy (matching with digital) being defined as total energy should be > 20 GeV
- Cosmics are rejected using pattern of fired modules :

2 modules distant by at least 1 should have wire energy > 3 GeV
- Bhabhas are rejected using fraction of energy deposited in the 2 most energetic clusters.

Class #18: Events in time with beam crossing:
selection based on T0 information from ECAL wires:
abs(T0)<120 ns in End caps and abs(T0)<100 ns in Barrel

Class #19: Muon candidates of all energies:
based on HMAD or MCAD flags or Mucalo(proba.>.80)

Class #20: Bhabha candidates (selection based on Ecal only):
-at least 2 non adjecent modules with E(wires)> 35 GeV each
-2 Ecal clusters with E(pads)> 35 GeV and |cos(theta)|<0.95

Class #21: Single photon candidates

This class flags events with at least one photon with an energy > 1.0 GeV and no charged tracks. To be flagged the event must have TPC HT bit ON, no good (i.e. 4 or more space points) TPC tracks and the SNG_N_EM trigger bit must be set in the Level 1 trigger word. In addition, the event must have at least one cluster with pad energy greater than 1.0 GeV, stack 2 pad energy greater than 0.1 GeV, more than 0.1 GeV in either of the other two stacks and the theta of the cluster must lie in the range 13 to 167 degrees.
The final requirement is that if such a cluster exists then there must be 1.0 GeV or more on the wires for the module in which the cluster lies.

Class #22: Coincidence in Sical A AND B (E >= 20 GeV)

Class #23: Single arm in Sical A OR B (E >= 20 GeV)

Class #24: Di-Lepton Extended Class 15 candidates

Selections based on TPC and GAMPEK photons

The requirements are less strict than those used in the common lepton analysis.

1. Only tracks with >= 4 TPC hits, |z_0| < 10 cm, |p| > 0.1 GeV/c are used. Their numbers are counted separately for |d_0| < 5 cm (including |d_0| < 2 cm), and for |d_0| < 2 cm.

2. a) If there are exactly 2 such tracks with |d_0| < 5 cm, they are declared as good tracks, and the selection continues.
   b) If there are between 2 and 8 such tracks with |d_0| < 2 cm, they are declared as good tracks, and the selection continues. Any track with 2 cm < |d_0| < 5 cm is declared as bad.

3. It is required that at least one track with |d_0| < 2 cm or at least

```
   one GAMPEK photon has a momentum exceeding 2 GeV/c.

4. If there are more than 4 good tracks, each of them is required to have an
   opening angle eta with respect to the axis of the corresponding jet
   which fulfills cos(eta) > 0.85.

      Events surviving all these requirements are flagged as class 24.

 Class #25:  Slow control records

 Class #26:  Events to be used for alignement and calibration purposes
             Muon events selected by mean of trigger bits
             Bhabha events selected from two ecal modules with wire
             energies above 30 GeV

 Class #27:  Two different possible definitions but never on the same dataset :
     POT     VDET Laser events for VDET alignment / calibration
     MINI    electron selection from QSELEP (Rt>-3, -3>Rl>3)   MINI >= 10 only

 Class #28:  Heavy Flavour muon flag , defined on MINIs only
     MINI    muon selection from QSELEP (QMUIDO flag 13 or 14) MINI >= 10 only

 Class #29:  Random trigger events

 Events which do not satisfy any of the preceeding selections are put in
 class #30
```

EDIR classes for LEP 2 Data and Monte Carlo:

Class # 1: >= 1 Ecal cluster with energy(wires) >= 1.5 GeV
+ 1 module with Ewire > 0.5 Gev, |T0| < 200ns.

Class # 2: Hcal energy(pads) + Ecal energy(wires) > 15 GeV

Class # 3: Cosmics passing through VDET

Class # 4: Hcal energy(pads) > 3 GeV + HCW(4 planes) * ITC trigger

Class # 5: 1-->7 tracks with >=4 TPC hits, D0 <5 cm and Z0 <20 cm

Class # 6: >=8 tracks with the same cuts as in Class 5

Class # 7: LUM A and LUM B, both with E >30 GeV

Class # 8: LUM A or LUM B, with E >30 GeV

Class # 9: 2-photon: >= 3 tracks, Ecal/ELEP < 0.5, Echarged/ELEP < 0.4

Class #10: Spare

Class #11: Low multiplicity W W A: >= 1 track of each sign

Class #12: Low multiplicity W W B: all tracks of same sign

Class #13: Spare

Class #14: Spare

Class #15: Dilepton candidates (e+e-, mu+ mu-, tau+ tau-).

Selections based on TPC tracks only

Class #16: QQbar candidates (based on TPC tracks information)

Class #17: QQbar candidates (based on calorimetry information)

Class #18: Spare

Class #19: Muon candidates of all energies:
based on HMAD or MCAD flags or QMUIDO

Class #20: Bhabha candidates (selection based on Ecal only):
-at least 2 non adjacent modules with E(wires)> 35 GeV each
-2 Ecal clusters with E(pads)> 35 GeV and |cos(theta)|<0.95

Class #21: Single photon candidates

Class #22: Sical A (E >= 20 Gev) AND Sical B (E >= 20*ELEP/91.2 GeV)

Class #23: Spare

Class #24: Di-Lepton Extended Class 15 candidates

Class #25: Slow control records

Class #26: Events to be used for alignement and calibration purposes
Muon events selected by mean of trigger bits
Bhabha events selected from two ecal modules with wire
energies above 30 GeV

Class #27: Two different possible definitions but never on the same dataset :
POT VDET Laser events for VDET alignment / calibration
MINI electron selection from QSELEP (Rt>-3, -3>Rl>3) MINI >= 10 only

Class #28: Heavy Flavour muon flag , defined on MINIs only
MINI muon selection from QSELEP (QMUIDO flag 13 or 14) MINI >= 10 only

Class #29: Random trigger events

Events which do not satisfy any of the preceeding selections are put in
class #30

Appendix H

C- and C^{++} Extended ALPHA

In this appendix, a brief introduction to the use of ALPHA from a C or C^{++} program on UNIX machines is given.

H.1 ALPHA^{++}:

The ALPHA^{++} project has been started in 1998 with the following goals:

- **convert the ALEPH data (preferably LEP2 data) from the BOS bank style into persistent objects and write them to an Objectivity/Db database.**
- **rewrite a miniversion of ALPHA in an object oriented computing language (C^{++}), based on the Objectivity database.**

The project is still in progress.

All relevant informations can be found on the ALPHA^{++} webpage:

http://alephwww.cern.ch/ disserto/objty

H.2 C-extended ALPHA:

This C-Extended ALPHA package was set up by Bob Jacobsen, who is not any more a member of ALEPH.

No modifications have been done to the preliminary version below since May 1997. It is not considered as standard ALEPH software, therefore many features may be unreliable, obsolete or incomplete now.

```
"C Extended ALPHA" allows you to use all the facilities of ALPHA
from a C or C++ program.  It is available on UNIX OSF1
 machines.

There are three new types defined from integers:  AlphaTrack,
```

AlphaVertex and AlphaPartNum. These are used as indices into the existing ALPHA banks. Except for two exceptions defined below, the syntax is as described in the ALPHA manual. Note that everything is upper case only.

An example:

```
  AlphaTrack i,j;
  for (i=KFCHT; i<=KLCHT-1; i++)
    for (j=i+1; j<=KLCHT; j++)
      printf(" %d and %d have mass %f\n",i,j,QM2(i,j));
```

The exceptions are:

1) CQTPN returns a character string, which is complicated and error prone in C. Instead, the C version takes the result string as an additional argument:

```
      char name[80];
      CQTPN(KFMCT,name);
```

2) 'Output arguments' are passed as references. For example, QDEDXM is declared:

```
     void QDEDXM(AlphaTrack,
            int numHypotheses,
            float hypMass[],
            float hypCharge[],
            float *measuredIonization,
            int *numSamples,
            float *length,
            float expectedIonizationForHypothesis[],
            float expectedIonizationSigma[],
            int *error);
```

   and should be called with:

```
      QDEDXM(i,2,MassArray,ChargeArray,&MeasIon,&NumSamp,
          &Length,ExpectArray,SigmaArray,&error);
```

The function prototypes in the header file serve as the exact documentation for the argument types.

ALPHA expects to call FORTRAN versions of QUEVNT, etc. The linker sees these as 'quevnt_' (i.e. FORTRAN routines are moved to lower case and have an appended underscore).

There are three approaches for your code:

1) Provide these routines in FORTRAN, and have them call whatever C routines you want to add.
2) Provide C routines called quevnt_, etc, which will be directly called from ALPHA.
3) Provide C++ routines of the form

```
        extern "C" void quevnt_(...)
```

The include file "$ALEPH/cxa/include/cxa.h" defines
all the necessary wrapper functions and macros, and should
be included at the start of your program. When
used with C++, some of the wrapper macros are redefined
as inline functions to reduce the number of procedure calls.

The library "$ALEPH/cxa/lib/libcxa.a" should be included
before the ALPHA and system libraries. If you include C++
code, linking is complicated by the need to use the C++ linker
(to resolve static constructors) and still get the correct
FORTRAN system libraries linked. The following Makefile fragment
will do this:

```
$(NAME): $(OBJFILES) $(LFILES)
   $(LD)  -o $(NAME) $(OBJFILES) $(LFILES) $(LDFLAGS) $(LIBS) $(FORLIBS))
```

where the symbols LD and FORLIBS have previously been set
to machine specific values, perhaps by lines in the .cshrc
of the form:

```
    switch (${OS})
         case OSF1:
                setenv CCC cxx
                setenv FORLIBS '-lfor -lUfor -lots -lutil -lm -lFutil'
                breaksw
         case IRIX5:
         default:
                setenv CCC CC
                setenv FORLIBS '/usr/lib/libftn.so /usr/lib/libc.so -lm '
                breaksw
          case ULTRIX:
                setenv CCC g++
                setenv FORLIBS '-lfor -lUfor -lots -lutil -li -lm'
                breaksw
```

Known problems and deficiencies:

No HPUX version yet.

Many more inline functions should be coded in C++.

Some of the C wrappers may be missing (please let
me know - I can't actually believe some of these
are being used, but will be happy to add them)

It would make a LOT more sense if the C++ types
AlphaTrack, AlphaVertex, AlphaPartNum where
inheritable classes. Unfortunately, I have not
found a compatible way to do this - ideas are
welcome.

Index

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