Title: Finalise the TPVX/1 Bank

Action: s/r TPBNKS is called to cut the tail of the TPVX/1 bank. For the first event only, a list of banks that will be output is printed using s/r EVBKPR.

TP Error Messages: There is one TP error message possible from this step, originating from a BOS error return.

- **** Error in TPBNKS **** Error return from BOS routine BCHM

Title: TP Mass Assignment, Decays and V^0 Finding

Action: s/r TPMASS is called to identify as best as possible the particles in the event. A mass code is written into every TPTR bank even if it is 0 (= unknown type). In addition a mass value is also stored, which is that of a pion for unidentified particles for the purposes of calculating total energies, etc. Decaying tracks in the Jet Chamber are searched for by s/r TPCH2V and converted photons and V^0 particles are searched for by s/r TPVEE. If any are found, additional TPTR banks and TPVX banks are created. Finally s/r TPTCNT is called to count the number of tracks found and update the event summary bank TPEV.

TP Error Messages: There are two TP error message, the first of which could indicate that Step 3 was not done and the second, that Step 9 ended prematurely.

- `**** Error in TPCH2V ****  'TPVX/1' bank is missing`
- `**** Error in TPTCNT ****  'TPVX/I' bank is missing`

Title: Store the Muon Results Summary

TP Flag 12: 0 ⇒ omit, 1 ⇒ store results, −1 ⇒ +1 but MUR2/4,5,6 deleted at end, −2 ⇒ −1 but MUR2/2,3 deleted as well.

Action: s/r TPMUR is called if TP flag 12 is non-zero. This copies muon analysis information from the MUR2/1 bank to the TPTR summary banks. At the end of the step, some MUR2 banks are deleted if requested by TP flag 12 in order to save space. These are essential for graphical display purposes though so it is not advisable to delete them if muon results scanning is to occur later.

TP Error Messages: None.

Title: Muon Analysis

TP Flag 10: 0 ⇒ omit, 1 ⇒ do analysis if not done, −1 ⇒ delete old results and re-analyse

Action: If TP flag 10 is set to −1 or either of the 2 banks MUR1/0 or MUR2/0 is missing, all MUR1 and MUR2 banks are deleted. Then s/r MUANA is called to carry out muon finding using as input the PATR and MUEV banks (see JCN 22). The output banks are MUR1 and MUR2 (two of the former, 7 of the latter).

TP Error Messages: None but there may error messages from the muon routines, particularly s/r MUCOOR. These are explained in a short table printed out during processing of the first event. A summary of errors and statistics is printed at the end of the job before the TP histogram printout.

Title: LG Clusters Matched to I.D. Tracks

TP Flag 8: $0 \Rightarrow$ omit, $\neq 0 \Rightarrow$ carry out step

Action: s/r LGCDIR is called if TP flag 8 is non-zero. This performs energy corrections to the clusters, determines the cluster directions and associates clusters to inner detector tracks by impact position. Unassociated clusters are assumed to be photons.

TP Error Messages: No TP error messages are printed in this step but an error message from s/r ILCTRC about tracks being out of the Jet Chamber is not uncommon. This is due to a 'harmless' but annoying bug in PATREC for some tracks which leave the Jet Chamber through the end walls.[1]

[1]If a track has a hit associated, by chance, at a radius greater than the exit point of the track, it is often kept. When the r-z fit is done, the z position of this hit is ignored but the stored end point of the track is taken from the radial position of this hit and its predicted z by extrapolation, which puts it outside the chamber.

Title: dE/dx Analysis and TP Summary

TP Flag 6: 0 ⇒ omit, ≠ 0 ⇒ do analysis & TP

Action: s/r DEDXAN called if TP flag 6 is non-zero. This first calls for inner detector z recalibration and hit cleaning without altering the PATR bank (ZSFIT(3)). Then DEDXBN is called to do the dE/dx analysis (see JCN 71). Finally the results are placed in the appropriate TPTR track summary banks.

TP Error Messages: No TP error messages are produced in this step but error messages may be printed from s/r ERRMON called by s/r DEDXBN. These indicate a rare error in the PATR bank which has not been tracked down and corrected yet. They can usually be ignored.

Title: TOF Analysis

TP Flag 4: $0 \Rightarrow$ omit, $\neq 0 \Rightarrow$ do analysis

Action: s/r TOFINT called if TP flag 4 is non-zero. This creates a TOFR bank if an ATOF bank exists.

TP Error Messages: No TP error messages are produced in this step but error messages may be printed by s/r TOFINT including a short bank dump. Recovery action is taken.

Title: Jet Chamber Processing

TP Flag 2: Pattern recognition: 0 ⇒ no, 1 ⇒ yes if not done, −1 ⇒ yes, after deleting all PATR,[1] JHTL and ZVTX banks, 2 ⇒ add new PATR and JHTL banks.

TP Flag 14: JETC calibration: 0 ⇒ no, ≠ 0 ⇒ yes

TP Flag 16: z recalibration[2]: 0 ⇒ no, 1 ⇒ yes, 2 ⇒ yes plus PATR z refit.

TP Flag 17: Circle $r\phi$ refit: 0 ⇒ no, 1 ⇒ yes, −1 ⇒ yes[3]

TP Flag 18: Parabola $r\phi$ refit with vertex constraint: 0 ⇒ omit, 1 ⇒ without vertex, 2 ⇒ weak, 3 ⇒ strong; +10 ⇒ plus common rz fit; −ve options ⇒ +ve[3].

TP Flag 19: TP Tracks: 0 ⇒ omit, 1 ⇒ from circle fit PATR , 2 ⇒ from parabola fit PATR, −ve options ⇒ as for +ve plus only latest JHTL bank kept.

Action: s/r TPJETC is called. This calls s/r TPJTCA, if TP flag 14 is non-zero, to calibrate the Jet Chamber. S/r TPJTCA calls s/r JETCAL if no calibrated JETC bank exists, s/r JRECAL if it does. If TP flag 16 is positive, s/r ZSFIT is called to do z recalibration and optionally a PATR rz refit. Then s/r PATRCO is called according to TP flag 2 with prior deletion of banks if requested. Next there is an optional call to s/r RFEVFT[4] to perform a circle $r\phi$ refit, creating new PATR and JHTL banks. Then there is the optional call to s/r FITEVR[5] to perform a parabola $r\phi$ refit, creating new PATR and JHTL banks. If the current event is Monte Carlo, s/r MCTR4V is called to trace the origin of PATR tracks. If no ZVTX bank exists, s/r ZVERTF is called to find the 'fast' z vertex. The PATR banks are then re-arranged so that the bank with the chosen fit is the one with the lowest number. Finally the s/r TPPATR is called to create and fill TPTR banks for every charged track in the PATR bank with the lowest number except for those tracks determined to be continuations of other tracks.

TP Error Messages: None.

---

[1] Except MC PATR/12

[2] No hit cleaning

[3] Newest unrefitted PATR bank and JHTL bank are deleted at the end of the step.

[4] See JADE Note 48

[5] JCN 61 needs updating

5. What JCL is needed to use the TP program?

There are 2 standard jobs available on F22YAM.TPSOURCE, which is also the dataset with the TP source routines. The jobs execute the TP program in the form of a load module thus avoiding linking the routines every time. The simplest job is in member #RUNTP. If tapes are to be used for input and output, #RUNTPC is to better as it includes a step to copy the input data to temporary disk before the TP program is executed. If the load module has to be relinked, #LINKTP exists to do it.

It is recommended to check that your copies of the above JCL jobs are still up-to-date before you use them. This is because it is possible that library changes may necessitate alterations in the JCL statements from time to time.

6. The TP Program Structure.

The main program, in member @TPMAIN, calls s/r TPINIT to initialise the analysis packages and read the 'data cards'. Then an event is read using s/r EVREAD and processed by s/r TPSPRV and written out by s/r EVWRIT. (This is a simplified description as other routines are also called by the main program.) Some TP error messages may be printed but these should be self-explanatory. The actions of s/r TPSPRV are described in the following pages.

3. What does the TP program do?

The program consists of a number of a steps which are executed in strict order. Each step performs a particular task. A short list of the tasks is given below. A more complete description is given on the following pages. As changes are made, the appropriate page will be issued as an update to this note.

a) Jet chamber processing (incl. pattern recognition and refitting).

b) Vertex finding and fine momentum determination.

c) TOF analysis.

d) dE/dx analysis.

e) LG analysis.

f) Muon chamber analysis.

g) Tagging system analysis.

h) Searches for V^0 particles and charged decays.

i) Sphericity and thrust calculations.

j) Production of the summary banks (TPEV, TPTR, TPVX).

4. What about the 'data cards'?

There are, as stated above, 20 'data cards' which can be used to steer the program. They are read by the program from FORTRAN unit 5 (SYSIN). The first one is special in that it has a different format from the others. Its function is to specify the maximum number of events to be processed, the run number range and the time-out safety margin. The format is (10X, I10, 10X, 2I10, 10X, I10) but it is easier to modify a copy of a model job than to create your own JCL. The layout is shown below with an actual example beneath it.

```
MAX EVNTS=nnnnnnnnnn, RUNS   =ssssssssssssssssssss, /!.NTOUT=tttttttttt
MAX EVNTS=    999999, RUNS   =          0  99999999,   NTOUT=       500
```

Note that the run range is given as a START run number and an END run number with both START and END being part of the range. The format is a little strange in order to include the experiment number, being 100,000 * Experiment No. + Run Number. Also the time-out units are 1/100 *s*.

The next 19 'data cards' control the various options. Their format is (27X,I3). The model jobs contain sample 'data cards' with comments on them to assist in choosing the right value for your application. However because of lack of space, not all options can be mentioned. The list of the TP flags on the next page, in association with the descriptions of the various TP steps, should be definitive.

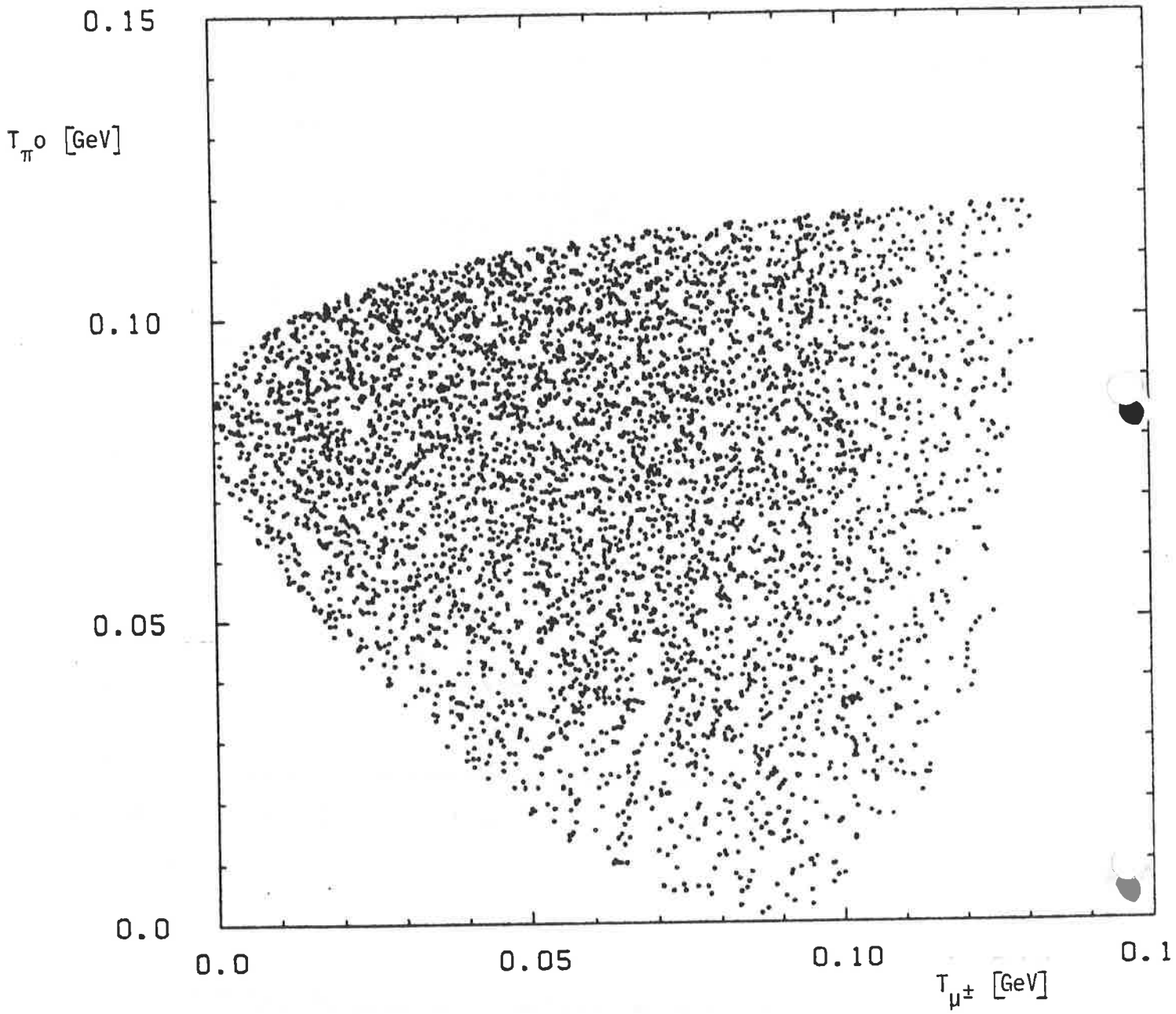


Fig. 2a): Dalitz plot of the decay $K^\pm \to \pi^0 \mu^\pm \nu$.
T_{π^0} and $T_{\mu^\pm}$ are the kinetic energies of the π^0 and the $\mu^\pm$ respectively.

Old scheme for neutral kaons: The routine JTRNTR 'tracks' kaons to the surface of the lead glass without any interaction. (This routine is called for neutrons as well.) Then the particles are taken over by the lead glass routines TRLGL if they hit the barrel or ENDCLG if they hit the endcap. These routines determine the particles energy loss in the lead glass.

New scheme for neutral kaons: JTRNTR is modified slightly. For each kaon a decay point is determined. If this point is inside the space surrounded by the lead glass the routine TRKADC is called, proceeding like step 3) in the new scheme for charged kaons.

Final remarks

1) In the Tracking Monte Carlo running at the Heidelberg IBM since 1984 the new routines have been used. Unfortunately for the KØL the same decay length as for the $k^{\pm}$ was put in, namely 3709 mm instead of 15540 mm. I studied the consequences on the 4 vector level with the LUND generator. In 20 000 events with 14600 KØL's I found with:

3709 mm decay length 2160 KØL-decays and with
15540 mm decay length 610 KØL-decays in the Inner Detector.
The error is corrected since 20/10/84.

2) The new routines will become standard on F22ELS.JMC.S and F22ELS.JMC.L at the date of appearance of this JADE NOTE.

3) If somebody wants to have kaon decay at the 4-vector level in the LUND generator it is recommended to use the routine F11HEL.LU52.S (LUDECY) instead of F22PET.LUND52.S(LUDECY). At the first call part of some arrays are overwritten in BLOCK DATA LUDATA. These are DECLEN(18), DECLEN(38), IDB(18) IDB(38), CBR(203...214), KDP(809...856). DECLEN $\neq \emptyset$. and IDB $\neq \emptyset$. give rise to the decays of KØL and $k^{\pm}$ in the LUND generator. Take care that no other data are stored in the above mentioned arrays.

Decay Mechanisms

(6), (7) and (8) two body decay

(5), (9) and (10) relativistic phase space decay

As an example the triangular Dalitz plot for the decay $K^o \to \pi^o\pi^o\pi^o$ is shown in Fig. 1.

(1) to (4)

(11) and (12) $K_{\ell 3}$ decays using the matric elements

$$f_+^2(t) * \{2*(p_K p_\ell)(p_K p_\nu) - m_\ell^2 (p_K p_\nu) + (\tfrac{1}{4} m_\ell^2 - m_K^2)(p_K p_\nu)\}$$

$$+ f_+(t) f_-(t) * m_\ell^2 * \{(p_K p_\nu)\}$$

$$+ f_-^2(t) * \tfrac{1}{4} m_\ell^2 * (p_\ell p_\nu)$$

m_ℓ and m_K being the masses of the leptons and the kaons respectively, and p_K, p_ℓ, p_ν the 4 vectors of the kaons, leptons and neutrinos respectively. f_+ and f_- are the form factors of the $K_{\ell 3}$ decays. They depend only on $t = (p_K - p_\pi)^2$, the square of the four-momentum transfer to the leptons. Several parametrizations for f_+ and f_- are possible. Here the λ_+, λ_o parametrization is used because of the small correlation between both parameters (see PDG 1982 pp. xii et seq. and pp. 73 + 74).

$$f_+(t) = f_+(0) * \left[1 + \lambda_+ \frac{t}{m_\pi^2}\right]$$

$$f_-(t) = (\lambda_o - \lambda_+) * \frac{(m_K^2 - m_\pi^2)}{m_\pi^2}$$

The values for λ_+ and λ_o are taken from PDG 1982 p. 76.

	λ_+	λ_o
$k^{\pm}_{\mu 3}$	0.032 ± 0.008	0.004 ± 0.007
$k^{o}_{\mu 3}$	0.034 ± 0.005	0.025 ± 0.006
$k^{\pm}_{e 3}$	0.0285 ± 0.0043	
$k^{o}_{e 3}$	0.0300 ± 0.0016	

Appendix B:

How to read records of unknown length in FORTRAN 77:

Declare an array which is certainly longer than the longest record, e.g.

```
      INTEGER*2 array(4096)
```

Before the first READ:

```
      CALL ERRSET(213,256,-1,0,0,0)
```

to suppress error msges and error branchings if the record is shorter than the array.

In the READ-Loop:

```
      READ(n,END=stend,ERR=sterr) array
stx   CALL REDLEN(lenblk,iret)
      IF ( iret .NE. 0 ) GO TO stx
```

"iret" should be zero. If not, an error has occured. Try again the CALL, maybe with an error count to avoid dead loops. If o.k. "lenblk" contains the record length in bytes. Grind the array down!

The errors 213 are however reported in the FORTRAN 77 error summary.

When all files are copied, proceed:

(m-line)
CHANGE↓↓

GIVE MESSAGE

(m-line)
START JADEØ↓↓

START JADEØ REQUEST ACCEPTED

(m-line)
Proceed as in the center of the previous page. After the response
GIVE MESSAGE

(m-line)
press <ctrl>+Z

---- THE END ---

Press <esc> and log on again under user RT
ΘUNFIX 63↓

ΘLOG↓

That's it !!

3) The COPY Commands

When the F58 service module is loaded a typical command list to copy source file can be entered like this:

```
FROMEXP (directry.user)source:SYMB↓
TOIBM userid.libname(membname)↓
COPY↓↓
```

or

```
FROMIBM ......
TOEXP ......
COPY↓↓
```

For the IBM-side no apostrophes! The library "libname" must already exist. For TOIBM "membname" can be a new or an old member, which will then be overwritten. The members copied to the IBM are directly NEWLIB-compatible.
On the EXP-(NORD-)side all file conventions hold, i.e. abbreviations allowed, or "directry." may be omitted for disc-resident files. However it is not possible to create a file with quotation marks. Furthermore, to copy files from IBM to NORD, the user has to establish a "friend-write-access" for the user RT. Do this sometime at some terminal and once for good:

Logon under the desired user name and proceed:

```
ΘCREATE FRIEND↓
FRIEND NAME: RT↓

ΘSET-FRIEND-ACCESS
FRIEND NAME: RT↓
ACCESS (R..... OR N): RWA↓

ΘLOG↓
```

For copying data files the conditions are different. To avoid complications the user should only copy data files with INTEGER data, because the floating-point representations on the NORD and on the IBM are different. The typical message list is

```
FROMEXP (directry.user)dlist:DATA↓
TOIBM userid.dsname.subname↓
COPY/DATA↓↓          (optl.)
```

On the NORD side the conventions are the same. On the IBM the dataset must be allocated before as a sequential dataset of sufficient size. All other parameters are overwritten during the Copy-action. Old data in the dataset will be overwritten. The copied data are written in records of length 4096 bytes plus an odd record at the end if the original dataset does not consist of a multiple of 4096 bytes. See App. B about how to read the copied data in a FORTRAN77 program.

```
-----------------------------
-  TMS 9900 SERVICE PROGRAM  -
-----------------------------

         ( some lines )

1 : COMMUNICATIONS WITH IBM          <------  Reentry after timeout.
2 : COMMANDS WITHOUT USING IBM                (see below)

--->      1   > 1↓
```

After entering the 1 as indicated above a line appears which in the following will be called the NORD m-line:

MESSAGE ("CTRL+C" : TAKE LAST COMMANDS):

After the NORD m-line has appeared the program is always ready to accept messages (abbr. msge's) for the IBM. Such a message consists of one or several lines of text (with one ↓ to terminate a line) and an empty line, or equivalently an extra ↓, to send the message off. A one-line simple command looks like this:

command↓↓

A single ↓ entered after the NORD m-line sends an empty msge. It can be used to tickle the IBM for a return msge. Note that the IBM never sends a msge by its own, but only in response to a msge from the NORD. Any return msge is output at the terminal, followed by the NORD m-line.

After a message text is sent offthe terminal input is blocked until the response from the IBM arrives.

Ending the session at the NORD10:
When everything is said and done, press Ctrl+Z to stop the program. The response is something like THE END. Afterwards (don't forget!) log on again under RT and proceed:

ΘUNFIX 63↓
ΘLOG↓

When no msge is sent for more than 5 minutes, the IBM link is released, a Timeout message is output, and the NORD program starts again almost from the beginning (see arrow near top of this page). Proceed with entering a 1↓ . Nothing is changed however on the IBM side. The module from before the timeout is still active.

of the JADE-detector from an electromagnetic shower with a given impact point, a vertex and an energy. This is done by integrating the theoretical shower function SF (contained in the member EXPECT) with the use of the VEGAS integration method.

Input variables :

IPART	:	detector part
E	:	Energy of showering particle
VAR(2)	:	Impact coordinates
VERT(3)	:	vertex coordinates
NBLOCK	:	number blocks, for which the energy fraction is calculated
N(BLOCK) IBLIST	:	list of block addresses, for which the energy fraction is calculated.

Output variables :

(NBLOCK) BLCFRC	:	list of energy fractions for all required blocks
AVGSUM	:	sum of all calculated energy fractions.

The photon 4-vectors are stored in the array

COMMON /CPARTC/ GANMAT (40,10).

GANMAT contains the following information :

GANMAT(N.1) = E_γ
(N.2) = $E_{\gamma,x}$
(N.3) = $E_{\gamma,y}$
(N.4) = $E_{\gamma,z}$
(N.5) = detector part
(N.6) = # blocks
(N.7) = X^2/D.O.F.
(N.8) = not used
(N.9) = not used
(N.10) = Pointer to LGCL

All three following routines are used in the main analysis program but might also be of interest for special applications.

The Subroutine SHWCPR

The subroutine

SHWCPR (VERT,CHI22)

compares a single electromagentic shower profile with a measured block topology and calculates a X^2-value. The properties of the measured clusters have to be loaded into the

COMMON /CLOAD/

which can be done for the LGCL-Cluster IC by calling

CLUSIN(IC,IFIND)

(IFIND=0 → Cluster not existing,
IFIND=1 → everything o.k.)

The 3-dim array VERT(β) has to be filled with the vertex of the photon. this can e.g. be (0.0.0) or the run-vertex. The output variable CHI22 contains the X^2 for the agreement between the measured block topology and the single-photon hypothesis.

The 'GAMR'-bank shows a similar structure as the 'PATR'-bank. It starts with a header containing general event information, followed by the properties of the single photons.
The double-shower information is only available on special request in the subroutine SHWFIT (see description below).
All given energies are in MeV units.
Cluster coordinates are (x,y) in case of the barrel and (φ,z) in case of the endcaps.

Location of words in 'GAMR'

Header :

word	content
1	# words in header
2	# photons in 'GAMR'
3	# words/photon without double shower fit
4	# words/photon for the double-shower fit only
5	# blocks for all photons
6	γ-Energy of all photons (ΣE_γ)
7	used vertex : 1 → event vertex 0 → (0.0.0)

Single particle information :

word	content
1	photon number
2	detector part (-1.0.1)
3	# blocks
4	analysis flag : 0 : no fit 1 : single-shower fit 2 : double-shower fit
5	date of cluster analysis
6	Energy
7	Impact point (1. coordinate)
8	Impact point (2. coordinate)
9	dx
10	dy
11	dz

Examples

The pointers KP1 and KP2 to the electron tracks K1 and K2 in the "PATR" - bank are given by the statements:

```
IP  = IDATA( IBLN( 'PATR' ) )
L0  = IDATA( IP + 1 )
LTR = IDATA( IP + 3 )
KP1 = IP + L0 + (K1-1) * LTR
KP2 = IP + L0 + (K2-1) * LTR
```

A loop over all "PHOT" - banks of an event could be (I is the pointer to the current bank):

```
      I = IDATA( IBLN( 'PHOT' ) )
101   CONTINUE
      IF( I .EQ. 0 )  GO TO 199
      .......
      .......
      I = IDATA( I - 1 )
      GO TO 101
199   CONTINUE
```

c. For each conversion:

1) Refit of electron - tracks in r - z with conversion vertex - constraint

2) Cuts

3) Photon fit with vertex - and invariant mass - constraint

4) Creation of a "PHOT" - bank

2. Options

Pattern recognition (rather time consuming), z - refit and photon fit are subject to the user's wishes and can be switched on/off in USCOGAM with the flags LPAT, LZF, LFIT. Default is LPAT, LZF, LFIT / 0, 1, 1 /.

```
LPAT = 1 / 0     perform modified PATREC  /  NO new PATREC
LZF  = 1 / 0     do the z - refit  /  do NOT the z - refit
LFIT = 1 / 0     do the photon fit  /  do NOT the photon fit
```

In addition to the parameters for the vertex search on BLGEO, there are cuts made in the user routine USCOGAM, which the user may change according to his needs.

The standard cuts applied are (for a description of the variables see "PHOT" - bank):

```
| DXY   | < 12.0 mm
| SDXY  | <  6.0
| SDPHI | <  4.0
| SDZ   | <  3.0
| SDTH  | <  3.0
        R < 50.0 mm
       SR > -5.0
      APV <  0.2
       XM < 30.0 MeV + EC / 50.0
momentum dependent cuts on # hits for electron tracks:
  # hits >  8    for                P <   75.0 MeV
  # hits > 16    for   75.0 MeV < P < 1000.0 MeV
  # hits > 25    for 2000.0 MeV < P
```

3. CPU time: 600 multihadron - events / minute with and 1500 multihadron - events / minute without pattern recognition.

repeat LOOP1 for +Z		
Calculate angle between clusters found above.Flag colinear pairs of clusters in TAGG2		
Create output banks	TAGSTO	
Return to TAGAN		

```
                              carlo data if simulation didn't set  flag
                              in 'ATAG

          CALL TAGINT(&100)  - Initialisation - to be called once  per
                               event ; RETURN 1 if can't work  out  which
                               tagger this is ( due to no head bank).

          CALL TAGADC(IWRITE,&100)     - Gets the ATAG data,if  IWRITE=
                                         1 writes out some debugging info
                                         Applies  nominal calibration to
                                         convert channel number to MeV.

          CALL TAGPED    - pedestal fixing - optional
                                          (no disaster if not done)

           CALL TAGKAL(IWRITE)     -    Calibration - optional
                                          (no disaster if not done)

           CALL TAGSUM(-1,SUMM,&100) - work out sum of -z and + z
           CALL TAGSUM(+1,SUMP,&100) - return 1 if sum has 'impossible'
                                                 value
C
C SUMM,SUMP,THRESH are in MeV
C

           IF((SUMM.GT.THRESH).OR.(SUMP.GT.THRESH))....
 100       CONTINUE


 ! For more detailed information about these and other routines
 ! in this package see 'F11LHO.TAGG.S(#TAGDOC)'
```

For completeness there now follows a brief description of the procedure adopted in the routine 'TAGGTP' for analysing all data from 1981 onwards.

PROCESS	ROUTINE NAME (if not done in TAGGTP)	NOTES
Initialisation	TAGINT	
Read data in 'ATAG'	TAGADC	1) An overall calibration that converts adc channel number to MeV is applied. 2) Software addresses are used from here on.
Subtract pedestals	TAGPED	These are caused by fluctuating pedestals due to 50HZ AC pickup on signal cables.It is only treated in those events where

4) For 1979/80 data only :
He must also have the private calibration file 'F22HOW.PEDESTAL.ALLSP80' attached to fortran stream 19.

The routine TAGAN

TAGAN(IERTAG,NRUN)

Arguments:

IERTAG – OUTPUT RETURN CODE

NRUN – INPUT DUMMY RUN NUMBER TO OVERIDE CONTROL OF WHICH ANALYSIS ROUTINE IS CALLED FOR MONTE CARLO DATA

```
++++++++++++++++++++++++++++++++++++++++++++
+ IN 99% OF CASES THIS CAN AND SHOULD BE +
+ SET TO ZERO                             +
++++++++++++++++++++++++++++++++++++++++++++
```

Description:

This routine controls the analysis routines. Its first job is to check that the input event can be analysed. Classes of events that can not be analysed are :

a) Pedestal events.
b) Events with no 'HEAD' bank.
c) Events with no 'ATAG' bank.

The routine must then decide which of the three possible sets of routines to call (one for each version of the tagging sytem). It does this by using the information in the HEAD bank. For real data it simply uses the run number. For Monte Carlo events it is neccesary to tell the program which Mark of tagging system to expect this is done by one of two methods:–

On encountering Monte Carlo data, the routine looks at the input argument 'nrun' . If this number is not zero it takes, and uses, it as the run number, so –

Use of NRUN:

NRUN	ASSUMED SIMULATION
<6000	Mark 1
6000> <12947	Mark 2
>12948	Mark 3

If nrun is set to zero there is a second line of attack which is to look at the 2nd half word of the bank 'ATAG'. The value of this determines which simulation was done according to following scheme.

Value of word	Simulation

Most of the tests shown on page 1 should be self-evident but some explanation may be helpful here. Test number 4, for example, checks whether the mass of a given particle agrees with the mass given in the particle data booklet for the given particle type code. The agreement has only to be better than 10 MeV/c^2. Test 5 checks that the Energy, Momentum and Mass are consistent to 5%. Tests 6 and 7 ensure that the production vertex for a given particle is reasonable.

Tests 1 and 13 to 17 are performed on the general event parameters. All the others are performed on every particle that is to be tracked, that is, on every particle that is placed in the VECT/0 bank with the exception of any partons that may be stored at the end.[1] The particles that are placed in the PALL bank are not checked.

Even if only a single particle fails any of the tests, the event is rejected.[2] This is necessary for safety reasons. For every error detected, a 1 line error message will be printed out which indicates which test was being performed and for which event and particle[3] sequence number. The event sequence number might not be the same as the event number created by the 4-vector generator incidentally. After the basic error message, additional information will be printed where appropriate. This will usually be a complete printout of the 10-vector of the bad track.

3. Disabling Tests in Special Circumstances.

When some non-standard data are to be tracked, for example, heavy stable particles, then it will be necessary to disable 1 or more of the tests. This is very easy to do. COMMON / CVFLAG / contains an array with 20 LOGICAL*4 flags which are block data set to .TRUE. . The n'th flag is associated with the n'th test. Thus to disable test 5, do the following in the main program that calls MCJADE:

```
      LOGICAL*4 VTEST
      COMMON / CVFLAG / VTEST(20)
            :
C       Other declarations etc
            :
      VTEST(5) = .FALSE.
            :
      CALL MCJADE(0,1)
            :
```

Please note that there are no tests with numbers 10,18,19 and 20 so manipulating these flags will have no effect. In order to avoid accidents, a warning message is printed before tracking commences if any of the flags are found to be set to .FALSE..
It should be stressed here that the tests performed by MCVALI are very fast and should not normally be disabled.

The TEX source of this note is in 'JADEPR.TEXT(JADECN72)'.

[1] See JADE Computer Note 69, last page.

[2] Except when the only error is a photon of energy $E \leq 200\ MeV$ failing test 5. In this case the event is kept.

[3] For tests 1 and 13 to 17, the particle sequence number is printed as zero ('0')

The program requires the following additional libraries to be linked

F11MEI.MCSHOWL

F22KAN.LGMTC.L

F22KAN.KZYLIB.LOAD

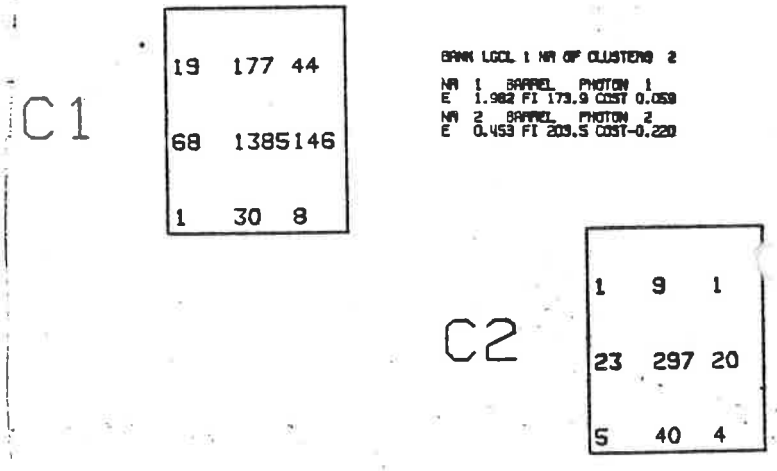
F22KAN.ANAL.L

some remarks

- the program includes the simulation scheme for muon and hadron energy deposition written by Junichi Kanzaki.

- it is essential to apply a cut on the single block energy to simulate the readout threshold of the LG ADC's. This can be done in the TP-step by changing the variable IPHALG in COMMON /CRDSTA/ to e.g. 28 (MeV).The effect of the readout threshold is demonstrated in the example below.

no readout threshold applied



readout threshold of 28 MeV