JADE COMPUTER NOTE 66

INNER DETECTOR SMEARING AND TRIGGER SIMULATION IN JADE MONTECARLO EVENTS

J. HAGEMANN, J. OLSSON, R. RAMCKE 24.8.1983

This note contains detailed information about the software handling of inner detector resolution and trigger simulation in Monte Carlo events in JADE. It is mainly intended to be a help for those who will have to change or add details in future developments. But it also gives overall information about the structuring of Monte Carlo events and Monte Carlo data files in JADE and should therefore be a useful introduction for the general user. The list of program errors and logical mistakes which is given below, as well as the notes on Muon Monte Carlo and TP-programs should also be of general interest.

The details refer to the proposed new scheme for smearing and trigger simulation, which does away with the many known program bugs which plague the up to now current version. A list of these bugs and the affected routines is provided below and where relevant, commented upon in the text.

The simulation of the JADE detector response is done in so called "TRACKING" programs. For the inner part of the JADE detector, (i.e. inner detector, scintillation counters and lead glass), the general programs reside on F22ELS.JMC.S and F22ELS.JMC.L. For the Muon filter simulation, special programs exist on F22RJB.RLMC.S and F22RJB.RLMC.L. For details of these programs, see JADE COMPUTER NOTES nr 26,40,54 and 55.

Here it is only noted that the input to these tracking programs are 4-vector events in the CPROD-format (see JADE COMPUTER NOTE 10) and the output are simulated JADE events in BOS-format. These output events are often referred to as "TRACKED" events.

For the inner detector, the tracking is done with a fine resolution, typically 20 micron (in recent developments, 5 micron has been used and is proposed for standard use in the new version). This resolution is kept in the output events and for this reason the output events are also known as " UNSMEARED, TRACKED " events. The simulation of the real resolution is known as "SMEARING" and is done when unsmeared events are read as input data. For the understanding of this process, the following general information is needed:

GENERAL INFORMATION:

A file with JADE MC events, fully simulated and in BOS format, always start with two "CALIBRATION and DETECTOR STATUS" events. These two "events" are automatically read and processed by the standard read routine(s), or written out by the standard write routine(s):

```
F11LHO.JADEGS(EVREAD)
F11LHO.JADEGS(EVWRIT)
```

Thus the normal user never sees these initial events and the first call to EVREAD will return the first simulated JADE event on the file, although this event is only the third logical event on the file.

The tracking programs create these initial CALIBRATION and STATUS events automatically, in the output routine

```
F22ELS.JMC.S(WRTMCB)
```


The logical event nr 1 contains the following banks:

MTCO:1, MJET:2, MGEO:3.

When written out by subr. WRTMCB, these banks contain:

```
MTCO:    empty
MJET:    first 34 words of COMMON /CJDRCH/
MGEO:    COMMON /CGEO1/  66 words
```

Thus the bank MGEO contains all the geometry constants of the JADE detector (beampipe, pressure vessel,inner detector, counters, magnet, lead glass), which have been used in the tracking process. The bank MJET contains detailed geometry constants for the inner detector (words 1-19) and resolution constants (words 20-34), which were used in the tracking process. Here the information about the fine resolution, e.g. the time bin constants, are kept. The bank MTCO is empty, but will later contain details of smearing and trigger simulation, which are not provided in the tracking program step.

Event nr 2 contains the following banks:

MUCO MUCD, MUOV, MFFI, MCFI, MFSU,
MCSU, MCEL, MCST, MUFI, MUYO, MUEN.

When written out by subr. WRTMCB, these banks contain various Muon filter information. WRTMCB calls subr. MUCONW for this purpose. Note however that MUCONW takes this information from the COMMON /CALIBR/, which is not properly set in the standard tracking program. Thus these banks have dummy content. Only when the complete Muon filter tracking program is used do these banks contain relevant information (see also the note below on Muon Monte Carlo).

To summarize: WRTMCB creates and writes, on its first call, the two calibration events and then, as the third logical output event, the first simulated event. Subsequent calls to WRTMCB will only write out the corresponding simulated event, in the normal way. All information about the tracking status and resolution is kept in the first two events on the output file.

What happens when reading such an unsmeared, tracked event file? This should be done with the standard routine EVREAD (or any other routine equipped in the same way). A flow chart of EVREAD is provided in Fig.1. Some details were already given in JADE COMPUTER NOTE 25.

EVREAD distinguishes three event types:

```
IEVTP = 0       Real data.
IEVTP = 1       Unsmeared Monte Carlo data.
IEVTP = 2       Smeared Monte Carlo data.
```

The variable IEVTP is set by EVREAD and passed on to other routines via the COMMON /CADMIN/ IEVTP,... The COMMON /CADMIN/ is BLOCK DATA set in subr. EVREAD.

If the first event on the file contains the bank HEAD, EVREAD assumes that this is Real data, IEVTP = 0 and immediate RETURN is done.

If the first event on the file contains the bank MTCO, MC data is assumed. Smeared on Unsmeared data is decided upon from the value of the Smearing Flag: The latter is contained in the bank MTCO:

```
IDATA(IDATA(IBLN('MTCO'))+1)
```

Its value is 0 for unsmeared, 1 for smeared data. IEVTP is set accordingly. To copy the information in the banks MTCO,MJET and MGEO into the relevant commons, EVREAD calls subr. RDMTCO. A flow-chart of this routine is provided in Fig.2.

1. Event
→ MTCO

RDMTCO copies the bank MJET into COMMON /CJDRCH/ and the bank MGEO into COMMON/CGEO1/; thus the information on the data file will override the BLOCK DATA setting of these commons. If the file contains already smeared data and if it was written with a modern version of RDMTCO and output routines, the bank MTCO will contain information about smearing and trigger constants. In this case, the second and third words in bank MTCO are non-zero. If so, the information in MTCO is copied by RDMTCO into the relevant commons: /CBIN/, /CTRIGG/ and /CRDSTA/.

If the first event contained the bank MTCO, the second event is expected to contain the bank MUCO. To unpack this Muon filter information, EVREAD calls subr. MUCON.

2. Event → MUCO

The third event should now contain a HEAD bank. After reading this event, EVREAD returns to the calling program and the first simulated event is available in COMMON /BCS/, with corresponding constants in various commons.

3. Event → HEAD

RDMTCO prints relevant information when it is called. The sequence MTCO, MUCO and HEAD can be repeated any nr of times and proper updating is always done, with accompanying informing print.

If EVREAD decides that the data are unsmeared, i.e. IEVTP = 1, it will smear them before returning. This is done for each event with a call to subr. RDJETC, which is an entry in RDMTCO. RDJETC calls in turn a number of subroutines to do the smearing of inner detector data in the bank JETC. This process will be described in detail below. RDJETC also calls the subroutines RDTRIG, RDTRG1 and RDTRG2, which simulate the trigger and create the banks LATC, TRIG:1 AND TRIG:2, as well as update the bank HEAD for TRIGGER ACTION and LOGICS CONDITIONS (TALC, HEAD word 22). Details of the trigger simulation will be given below. See also JADE COMPUTER NOTE 55.

Note here that the up to now current version of RDJETC calls the subr. RDTRG instead of RDTRIG. RDTRIG is essentially the same as RDTRG, but fills the COMMON /CTRIGG/ instead of COMMON /CTR123/ and is moreover extended to include 1982 trigger modes. It does not call RDTRG1 and RDTRG2 as RDTRG does, avoiding the logical error described below (see the section ERRORS).

Finally some words about the writing of Monte Carlo data to an output file. For this, the standard subr. EVWRIT should be used (or any other routine equipped in the same way). If called for real data, EVWRIT will only write the event on the output unit. If the event type IEVTP is not 0, EVWRIT will create the two CALIBRATION and STATUS events in the same way as described for WRTMCB above, and it will set the smear flag in the first word of bank MTCO, to the value 1. It creates the banks MJET and MGEO and fills them with the content of COMMONS /CJDRCH/ and /CGEO1/. The bank MTCO is also created and the content of the COMMON /CBIN/ (Smearing constants), as well as the COMMONS /CTRIGG/ and /CRDSTA/ (Trigger constants) is copied into MTCO. For the second event with Muon filter information, the subr. MUCONW is called. Finally, the simulated event is written, as the third logical event on the file. Subsequent calls will only write the MC events, just as is done for real events.

To summarize: EVWRIT will assure that the current smearing parameters are stored on the same file as the output events, keeping the structure of two CALIBRATION and STATUS events preceeding the actual MC events. If unsmeared events were read, the parameters of the just performed smearing are remembered. If smeared events were read, the smearing constants from the input file are transferred to the new output file.

OBS: It is not possible to smear already smeared events.

OBS: If unsmeared data should only be read and written, without smearing, a fast BOS READ and WRITE program should be used, since the use of EVREAD and EVWRIT automatically will invoke smearing and the original fine resolution is then destroyed.

A ready program for such fast READ and WRITE is provided in F22ELS.JMC.S(#COPY).

NOTE: The earlier version of EVWRIT did not copy any information from trigger simulation into MTCO and also not the dead cell status, nr of random hits, double hit resolution or wire efficiencies. Only the smearing variables which are situated in COMMON /CJDRCH/, like the time binning or z-resolution, were properly copied into the bank MJET.

DETAILS OF SMEARING:
==========

The purpose of smearing is to worsen the fine resolution provided in the tracking program, so as to obtain a resolution matching the real data. The resolution in the real data depends on the intrinsic resolution as well as on a number of more or less well understood systematic effects, like chamber position uncertainties, field inhomogeneities, etc.. Most of these systematic effects are not simulated in MC events and the resolution has to be artificially imposed in other ways. In earlier versions of the smearing routines, worsening of resolution beyond the intrinsic resolution has been involuntarily obtained by the presence of several program bugs. In the momentum determination of high energy tracks these bugs cause severe systematic effects, which are not seen in real data. The removal of these bugs therefore necessitates a better and controllable smearing of the resolution. In the following, a method for such smearing is presented and suggested for standard use. First however, a short introduction to the flow of subr. RDJETC:

```
RDJETC
   ---> RDHEAD          (Save date, HEAD words 6-8, into words 96-98)
   ---> RDTRIG          (Set version date with call to RDDATE,      )
                        (set /CTRIGG/ and /CRDSTA/ accordingly.     )
   ---> RDRESO          (Smear time and Z-coordinate values.        )
   ---> RDMODN
          ---> RDRDMH   (Generate random hits.                      )
          ---> RDMERG   (Merge random hits into JETC data.          )
          ---> RDINEF   (Kill some hits, acc. to wire inefficiency. )
          ---> RDDOUB   (Kill hits too close, double hit resolution.)
          ---> RDDDCL   (Kill hits in dead cells.                   )
          ---> RDPOIN   (Adjust pointers and data in JETC bank.     )
   ---> RDPATR          (Adjust some values in bank PATR:12.        )
   ---> RDTRG1          (Create banks LATC and TRIG:1.              )
   ---> RDTRG2          (Create bank TRIG:2.                        )
   ---> RDALGN          (Delete LG-blocks below readout threshold,  )
                        (specified by parameter IPHALG, default 0.  )
```

Here RDTRIG sets the COMMON /CRDSTA/ with information about dead cells in the inner detector. This information is used in the subsequent smearing (subr. RDDDCL). In an earlier version of RDJETC, which called subr. RDTRG after the smear routines, the first event was treated with a sometimes wrongly set COMMON /CRDSTA/.

After the proper smearing the hardware trigger is simulated in subr. RDTRG1 and RDTRG2. Relevant constants have been set by RDTRIG in COMMON /CTRIGG/.

The constants used in the smearing are stored in COMMON /CBIN/:

```
      COMMON/CBIN/TIME(6),ZOF,ZRS,ZL,ZSC,EPSI(3),DOUB(3),IRN(3),
     *  BINDL8(6),RJITT
```

TIME:	Bin width for drift time in mm, wires are put together in groups of 8.
ZOF:	Offset in z-coordinate, in mm.
ZRS:	Standard deviation for z resolution, in mm.
ZL:	Effective wire length in mm.
ZSC:	Scaling factor for z amplitudes.
EPSI:	Wire efficiency in each ring.
DOUB:	Double track resolution in mm in each ring.
IRN:	Nr of random hits inserted in each ring.

```
C---------- BINDL8 and RJITT are used in the new RDRESO for smearing.

           BINDL8:  Bin width for drift time in mm.
           RJITT:   Standard smearing of the drift coordinate.

   Default values are (BLOCK DATA set in RDMTCO):

      DATA TIME / 6*0.380/
      DATA ZOF, ZRS, ZL, ZSC / -10., 20., 2687., 1. /
      DATA EPSI / 3*.98 /
      DATA DOUB / 3*7.5 /
      DATA IRN  / 20, 20, 20 /

      DATA BINDL8 / 6*0.380/
      DATA RJITT / .270 /
```

The array BINDL8 is actually a help array and used together with the array TIME; it holds the actual binning used in the smearing, i.e. the values stored in time. This help array has been introduced to keep backward compatibility.

OBS: The TIME and RJITT values are designed to simulate the actual momentum resolution in real data. The values should not be directly compared to the space resolution in the real data.

RDRESO:

Drift-times and amplitudes are smeared in RDRESO. The drift-time, here called IDRI, is originally given in units of fine resolution, BINMC = .02 mm (or .005 mm). It is treated as follows:

```
      DRIFT = FLOAT(IDRI)*BINMC
      IDRIFT = IFIX(DRIFT/BINDL8)
      DRIINT = (FLOAT(IDRIFT) + .4999)*BINDL8
C-----------------------------------------------------------------------
C         SMEAR DRIFT COORDINATE
C
      Z1 = RN(DUM)
      SQLOG=SQRT(-2.*ALOG(RN(DUM)))
      G1=SIN(PI*2.*Z1)*SQLOG
C G1 IS RANDOM AND GAUSSIAN
      DRIINT = DRIINT + G1*RJITT
C-----------------------------------------------------------------------
      IDRI  = IFIX(DRIINT/BINMC)
```

Thus all smearing beyond the pure binning is governed by the parameter RJITT. The smeared coordinate is returned in units of the fine binning BINMC.

The difference between the old smearing scheme and the new one given by the above code, is shown schematically in Fig.3. Note the serious systematic error in the old scheme, coming from the adding of half a time bin to the drift coordinate. This bug gives rise to strong systematic effects for high momentum tracks. For lower momenta the effect is a worsening of resolution, which agrees roughly with the real data resolution. See also below in the section ERRORS; the errors in subr. JINIT and BLDAT also contribute to the artificial worsening of resolution in Monte Carlo data.

It is clear that in the proposed new scheme there is an interplay between the TIME bin width and the smear parameter RJITT. Several different combinations could be envisaged, that would give a similar momentum resolution in the data.

To change momentum resolution of the inner detector, the variables TIME(1-6) and RJITT should be changed. The present default values correspond approximately to a resolution dpt/pt = .02 * pt, with pt being the transverse momentum. A larger RJITT value means a worse momentum resolution. For RJITT < .1 mm, also the array TIME should

be changed to smaller values, to avoid binning effects. The following table may serve as a rough guide to such changes.

TIME (micron)	RJITT (micron)	dpt/(pt**2) (%)
380	270	2.1
380	100	1.7
200	200	1.5
150	150	1.0
80	80	0.7

For the z-resolution, the following algorithm is used:

```
   IAL = HDATA(J+1)
   XL = FLOAT(IAL)
   CALL NVERT( ZRS, XL, XLL )
   XL = XLL / ZL
   IAR = HDATA(J+2)
   XR = FLOAT(IAR)*ZSC
   HDATA(J+2) = IFIX((XL+.5)*XR)
   HDATA(J+1) = IFIX((-XL+.5)*XR)
```

Thus the first amplitude, which contains the true z-coordinate to 1 mm precision, is rescaled in terms of the second amplitude, which contains the dE/dx pulse height. The chosen algorithm assures that the z-coordinate calculation agrees with the calculation used for real data in various analysis routines.

The treatment of the z-coordinate has not changed, but in the earlier version the offset of 10 mm, ZOF, was subtracted before smearing. This subtraction was not compensated in later reconstruction of the z-coordinate, giving rise to a systematic shift of 10 mm in z-coordinates in MC data. In the new version, the z offset parameter is not used.

RDMODN:

RDMODN and the subroutines it calls are all concerned with adding or removing hits. The /CBIN/ variables to steer these actions are EPSI (wire efficiencies), DOUB (double hit resolution) and IRN (nr of random hits, e.g. from synchrotron radiation or electronic noise). The dead cell hit removal was already mentioned above. These routines are all straight forward and need no special comment. One change has been made in the new version: RDRDMH gives in the new version only random hits with coordinates inside the corresponding cell. The earlier version allowed maximum drifttimes everywhere and hits could have coordinates in the next or overnext cell.

NOTE: In the early versions of RDMTCO an elaborate scheme for changing COMMON /CBIN/ was foreseen, using the argument HOPT in
CALL RDMTCO(HOPT).
The sense of this has since been lost and the smearing occurs with the parameters in /CBIN/, independent of HOPT='DE' or HOPT='SE'. The argument in RDMTCO is kept however, for backward compatibility.

DETAILS OF TRIGGER SIMULATION:
==========

There are many different triggers in the JADE detector and their conditions vary over the years. Some of these triggers are directly linked with the status of the inner detector (dead cells, random hits) and the software simulation of trigger conditions is therefore done together with the smearing of MC data. This simulation is an important aspect in some physics analyses, e.g. 2-photon physics, where trigger efficiencies may vary considerably.

The routines for trigger simulation have also developed over the years, with the addition of new triggers or modification of existing

ones. The routines to be discussed below are the most complete so far, in that trigger conditions for 1982 and later are included (the so far standard versions only include conditions for 1979-1981) and in addition several bugs have been removed. The scheme however follows closely the earlier scheme described in JADE COMPUTER NOTE 55.

The trigger simulation involves a number of thresholds for latch settings and limits for various veto conditions. Over the years these thresholds and limits have been given in commons with varying names and lengths. The version described here introduces the COMMON /CTRIGG/ which is structured in a logical way and has additional dummy variables for later extensions. COMMON /CTRIGG/ is set by subr. RDTRIG and has the following content:

```
      COMMON /CTRIGG/ IHIST(3),NBPTFW,IDUM1(5),HDUM1,
     *  HLGBRT,HLGBST,HLGQT,HLGTOT(4,2),HECAPT(4),HLGTL,HLGTH,HDUM2(10),
     *                IWIDBS,NRNBSL,NRNBSH,NTOFBS,IDUM2(10),
     *                NTOFC,NTOFC1,NTOFC2,NTBGC,NTBGC1,NTBGC2,
     *                IWCOLL,IWCOLN,IWMPRG,HFMPRL,HFMPRH,HWCOLN,HMPRON,
     *                IWCTBG,NTFCOL,NTFCLN,IDUM3(10),
     *  HITCLL(3),HITWLL(3),HITSUM(3),HCHAMB(3),HMASK(16,3),HDEADC(10),
     *  HACC1,HACC2,HACC3,HACC4,HACCM,HDUM6,IWIDT2,IDUM4(10)
C
C DATE OF VERSION
      DATA IHIST /1,1,1982/
C
C NBPTFW = WIDTH OF BP-TOF MATRIX; OBSOLETE, NEVER USED IN TRIGGER
      DATA NBPTFW / 6 /
C
C DUMMY SPARE VARIABLES
      DATA IDUM1 /5*-1/, HDUM1/-1/, HDUM6/-1/
      DATA IDUM2 /10*-1/, HDUM2/10*-1/
      DATA IDUM3 /10*-1/, IDUM4/10*-1/
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     ****   THRESHOLDS FOR LATCHES ****
C
C     BARREL GROUPS,        BARREL SEPTANCES, ENDCAP QUADRANTS
      DATA HLGBRT/ 80/,        HLGBST/180/,        HLGQT/50/
C                  TOTAL ENERGY                      BARREL ENERGY
      DATA HLGTOT/ 6000,2500,2000,1500,        6000,2500,2000,1500/
C                  ENDCAP ENERGY
      DATA HECAPT/ 4000,1000, 500, 300/
C                  TAGGING ENERGIES
      DATA HLGTL/ 1500/,    HLGTH /6000/
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     ****   DATA FOR T1 ACCEPT  ****
C
C           NEUTRAL COPLANAR TRIGGER
      DATA NRNBSL/2/,  NRNBSH/3/, IWIDBS/3/, NTOFBS/1/
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     ****   DATA FOR T1 POSTPONE  ****
C
C           COPLANAR TOF TRIGGER
      DATA IWCOLL/3/, NTFCOL/5/
C           NARROW COPLANAR TOF TRIGGER     HWCOLN=1 --> TRIGGER ACTIV
      DATA IWCOLN/1/, NTFCLN/7/, HWCOLN/1/
C           MULTIPRONG TRIGGER              HMPRON=1 --> TRIGGER ACTIV
      DATA IWMPRG/6/, HFMPRL/3/, HFMPRH/6/, HMPRON/0/
C           COPLANAR TBG TRIGGER
      DATA IWCTBG/2/

C           LIMITS FOR NR OF TOF'S IN POSTPONE TRIGGERS
C           NTOFC  =  LIMIT FOR NTOF IN TOTAL ENERGY TRIGGER
C           NTOFC1 =  LIMIT FOR NTOF IN ENDCAP TOTAL ENERGY TRIGGER
C           NTOFC2 =  LIMIT FOR NTOF IN NRTOFBG-NS TRIGGER
      DATA NTOFC/2/,NTOFC1/2/, NTOFC2/7/

C           LIMITS FOR NR OF TBG'S IN POSTPONE TRIGGERS
C           NTBGC  =  LIMIT FOR NTBG IN TAG TRIGGERS
C           NTBGC1 =  LIMIT FOR NTBG IN BARREL ENERGY TRIGGER
```



```
C           NTBGC2 =  LIMIT FOR NTBG IN NRTOFBG-NS TRIGGER
      DATA NTBGC/0/,NTBGC1/0/, NTBGC2/2/
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     ****    T2 DATA    ****
C
      DATA HITCLL/8,7,8/,  HITWLL/2,2,2/  ,HITSUM/10,9,10/
      DATA HCHAMB/24,24,48/,  HMASK/16*1,16*1,16*1/,HDEADC/10*0/
C
C  HACC1:  NR OF ALL TRACKS IN T1 POSTPONE: TOTAL LG ENERGY + TOF
C  HACC2:  NR OF ALL TRACKS IN T1 POSTP.82: TBGNS + TOF
C  HACC2:  NR OF FAST TRACKS IN T1 POSTP.79-81: WIDE COPLANAR TOFS
C  HACC3:  NR OF FAST TRACKS IN T1 POSTP.82: TOTAL LG ENERGY AND TBG
C  HACC3:  NR OF ALL TRACKS IN T1 POSTP.79-81: TAG
C  HACC4:  NR OF FAST TRACKS IN T1 POSTP.82: TAG + TBG
C  HACC4:  NR OF FAST TRACKS IN T1 POSTP.79-81: NARROW COPLANAR TOFS
C  HACCM:  NR OF ALL TRACKS IN T1 POSTPONE: MULTIPRONG COPLANAR TOFS
C
      DATA HACC1/2/,  HACC2/2/,  HACC3/1/, HACC4/1/, HACCM/3/
C   IWIDT2 GIVES WIDTH OF T2 COPLANARITY
      DATA IWIDT2 /4/
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     WIRE DISABLE SECTION FOR JETC BANK
C
      COMMON / CRDSTA / NDEAD, NCDEAD, HITD(10), HCELLD(10), IPHALG
C
C                                          DEAD WIRES
      DATA NDEAD / 0 /
      DATA HITD / 10*0 /
C                                          DEAD CELLS
      DATA NCDEAD / 0 /
      DATA HCELLD / 10*0 /
C
C                  LEAD GLASS BLOCK READOUT THRESHOLD
      DATA IPHALG / 0 /
C
```

The BLOCK DATA setting (in RDTRIG) corresponds to 1982 trigger conditions. To get the status for earlier years or periods, the date in the HEAD bank has to be overwritten. This is done by the call to subr. RDDATE, which then has to be provided by the user. The original date in bank HEAD, which is of some interest, since it tells the date of the original tracking (and thereby the program version), is lost here; therefore a copy of this original date is done in subr. RDHEAD. Thus the date in words 6-8 in HEAD is copied into words 96-98 in HEAD.

According to the date in HEAD words 6-8, RDTRIG decides which status version to use; if it is different from the default of 1982, the corresponding updates in /CTRIGG/ and /CRDSTA/ are made:

```
C                                          **********************
C                                          *  CHANGES 1979-81    *
C                                          **********************
            HLGTOT(1,1) = 4000
            HLGTOT(2,1) = 2000
            HLGTOT(3,1) = 1000
            HLGTOT(4,1) = 500
            HACC1 = 1
C                                          **********************
C                                          *  CHANGES 1979-80    *
C                                          **********************
            HWCOLN = 0
            HITCLL(1) = 12
            HITCLL(2) = 11
            HITCLL(3) = 12
            HITWLL(1) = 1
            HITWLL(2) = 1
            HITWLL(3) = 1
            HITSUM(1) = 12
            HITSUM(2) = 12
            HITSUM(3) = 12
```

```
C                                       ************************
C                                       *  CHANGES 1979 ONLY   *
C                                       ************************
            HDEADC(1) = 17
            NCDEAD = 1
            HCELLD(1) = 17

C                                       ************************
C                                       *  CHANGES 1980 ONLY   *
C                                       *  PERIOD 1: JAN.-MAR. *
C                                       ************************
            NCDEAD = 8
            HCELLD(1) = 17
            HCELLD(2) = 37
            HCELLD(3) = 65
            HCELLD(4) = 66
            HCELLD(5) = 73
            HCELLD(6) = 74
            HCELLD(7) = 81
            HCELLD(8) = 82
C                                       ************************
C                                       *  CHANGES 1980 ONLY    *
C                                       *  PERIOD 2: APR.- DEC. *
C                                       ************************
            NCDEAD = 6
            HCELLD(1) = 17
            HCELLD(2) = 37
            HCELLD(3) = 65
            HCELLD(4) = 66
            HCELLD(5) = 81
            HCELLD(6) = 82
C                                       ********************
C                                       *  CHANGES 1981    *
C                                       ********************
            HACC3 = 2
            HACC4 = 2
```

After smearing the inner detector data, the trigger banks are set up in the calls to subr. RDTRG1 and RDTRG2.

Note: If you want 1979 trigger conditions, use a date after 1.7.1979, since an earlier date is not accepted by subr. KLREAD, in later reading of the smeared data.

RDTRG1:

Bank LATC is first created and filled, according to the thresholds given for the period in /CTRIGG/. Then bank TRIG:1 is created. T1 ACCEPT and T1 POSTPONE conditions are checked and set in the corresponding words in TRIG:1. In the subr. RDTRG2 the track situation in the inner detector is simulated, with help of the nr of hit wires (in bank JETC) and the arrays HITCLL,HITWLL and HITSUM. Dead cells which are permanently on in the track trigger are given by array HDEADC. The bank TRIG:2 is created and filled with corresponding T2 information.

The actual nr of "fast" and "all" tracks (JADE NOTE 31) are compared to the requirements of the various T1 POSTPONE conditions and the "TRIGGER ACTION and LOGIC CONDITION" (TALC) word (HEAD bank word 22) is filled accordingly.

Since a major change in trigger conditions occurred between 1981 and 1982, with different structuring of the trigger banks, separate subroutines are used for the trigger simulation in 1979-81 and 1982 - : RDTRG1 <--> RD82T1 and RDTRG2 <--> RD82T2, respectively. The 82 versions are called from RDTRG1 and RDTRG2; this is regulated by the value of IHIST(3), which gives the year of the status version.

Not all triggers in 1982 are simulated however; the following are not yet provided:

```
    T1 ACCEPT:
      FWMU(COPL+-1) * ECAP(COLLIN.)                    Trigger bit 4
```



```
      "ZORN" TRIGGER: >= 1 SEPT. * TOF<1 * TAG         Trigger bit 15
      RANDOM TRIGGER                                   Trigger bit 16

    T1 POSTPONE:
      FWMU(5) * 3 TRACKS(ID) * 1 MUTRACK(FORW.)        Trigger bit 8
```

NOTE: The forward muon scintillation counters have so far not been included in the detector simulation.
Tagging simulation uses the counter set up from 1979-80 and the conditions for 1981-82 and the present set up for 83, have not yet been simulated.
The dimensions and positions of tagging counters which are used in the tracking program, are not saved in any way on the output data set, since they are not included in the COMMON /CGEO1/.

WARNING: ALL TRIGGERS WHICH INVOLVE THE LEAD GLASS ENERGY THRESHOLDS are treated as step functions, i.e. the LG energy is either below or above a fixed threshold. In reality the threshold behaviour of energy triggers is more complicated and requires careful study, mostly involving shower program calculations of energy deposits.

WARNING: Earlier versions of RDTRG1 and RDTRG2 have bugs. These are described below, in section ERRORS.

RDALGN:

RDJETC finally calls subr. RDALGN. The tracking programs register all energy deposits in lead glass blocks down to 1 MeV. The read out threshold in the real data is however higher (25-30 MeV) and varies for different periods. RDALGN provides the possibility to kill all LG blocks in the bank ALGN (there is no bank ALGL in Monte Carlo), which have energies below a specified threshold, given by the variable IPHALG. Default value is zero, however, no killing.

Note in this connection that the really preferrable procedure is to transform the block energy into digitized counts, each count corresponding to 5-6 MeV, and then kill blocks which are below the read out threshold (5-6 counts in hardware read out). Finally remaining blocks should be transformed back into MeV values again. This really means that the energy is measured in units of 5-6 MeV and such a procedure would come closest to a realistic simulation. This is not yet active in the standard smearing process.

Note also that RDALGN is called after trigger simulation. The energy sums which determine the trigger are not dependent on the read out threshold.

ERRORS:

It follows below a list of known errors in the earlier versions of the smearing and trigger simulation routines. These errors are all corrected in the new versions presented in this note. For the sake of completeness, also errors in the tracking routines are listed. The correction of the latter is proposed to take place simultaneously with the instalment of the new smearing and trigger simulation routines.

RDRESO: .5 time-bin added to drift-time value; causes large systematic errors.
10 mm offset subtracted from z-coordinate, never corrected for in later software reconstruction.

RDRDMH: Random hits outside cell boundary possible.

RDTRG: First event will have dead cells in accordance with default values and not according to specification.

RDTRG1: TOF veto in wide coplanar trigger <=3 instead of <=4.

RDTRG2: Coplanarity in T2 tracks wrong.

RDMTCO: Copying of trigger constants and smear status into bank MTCO meaningless, since MTCO is deleted before reading the next event (with Muon constants). See however the note below, concerning TP-programs. To keep compatibility with those programs, the new version of RDMTCO still does the copying of COMMONS /CTRIGG/,/CRDSTA/ and CBIN/ into the bank MTCO.

EVWRIT: Creates MTCO bank and writes it out without filling it.

JINIT: Radius of radial-symmetric electric field at signal wires is different to radius used in later reconstruction programs. Systematic errors <= 200 micron in hit positions may result.

BLDAT: Following DATA values should be changed. Note that they are also BLOCK DATA set in subr. SUPERV, on F11LHO.JADEGS.

	VARIABLE	NEW VALUE	OLD VALUE
COMMON /CJDRCH/	YSUSPN	0.0	0.070
	TIMDEL	0.005	0.020
	DRIDEV	-0.349066	0.349066
	DRISIN	-0.342020	0.342020
COMMON /CJIONI/	POTBEA	160.3	213.0
	POTIVE	160.3	213.0
	POTOVE	160.3	213.0
	POTVES	160.3	213.0
	POTZJL	160.3	213.0
	POTZJR	160.3	213.0
COMMON /CGEO1/	RITNK	167.0	165.0
	DZJM	-30.0	-5.0
	DZJP	30.0	5.0
	XRZJM	0.337	0.060
	XRZJP	0.337	0.060
	ZTKM	-1350.	-1470.
	ZTKP	1350.	1470.
	DZTKM	-74.	-7.
	DZTKP	74.	7.
	XRZTKM	0.832	0.080
	XRZTKP	0.832	0.080

EVTINI, JIPATR: Call ro subr. DAY2 made for every tracked particle. the corrected versions make one call per event.

MCJADE: Event count sometimes wrong by one unit. This error will be taken care of later, it is not vital to the other changes discussed here.

NOTE ON MUON MONTE CARLO:
==========

A complete tracking program, which includes the Muon filter simulation, exists on F22RJB.RMLC.S and L. It is described in JADE COMPUTER NOTE 54 and more details are given in another note, which will soon be distributed by R. Barlow. In connection with the present note the following should be stressed:

The program decides upon which detector status to use in a similar fashion as described above, with the user providing a date. The proper detector status parameters are taken from the COMMON /CALIBR/, which is first filled by a call to subr. KALIBR (and KLREAD). This also means that in this program, the second event on the file, with the muon constants banks, really contains relevant information. Note also that the tracked events will contain an already "smeared" and realistic simulation of resolution as fas as the muon filter is concerned. This is because the geometry constants also involve the

varying positions of the muon filter with respect to the inner detector. For this reason the tracked events can not be used for simulation of different muon filter status in different periods, which is possible for the inner detector, as described in the present note. Thus the program produces tracked events which have "smeared" Muon filter but "unsmeared" inner detector data. Smearing of the inner detector occurs as soon as the events are read again, and it occurs with the detector status as specified by the user in subr. RDDATE.

----------->>>>>>>>>>> obs obs obs obs obs <<<<<<<<<<<----------

It is the responsibility of the user to assure that the inner detector status and the muon filter status agree, since all combinations are possible. There are no programs which will notice inconsistencies and print warnings.....
----------->>>>>>>>>>> obs obs obs obs obs <<<<<<<<<<<----------

Note also that the action of subr. RDHEAD in this case will be to save the version date of the Muon filter status, into HEAD words 96-9800082800 before HEAD words 6-8 are overwritten by subr. RDDATE, with the date of the specified inner detector status.

NOTE ON MONTE CARLO TP PROGRAMS:
==========

A version of the TP program is available on the tracking library:

F22ELS.JMC.S(#TPMAINC and @TPMAIN)

The main program @TPMAIN contains an event loop which does not use the subr. EVREAD and EVWRIT, but calls instead directly the BOS routines BREAD and BWRITE. The handling of MTCO and MUCO events is done directly in this loop. After the calls to subr. RDMTCO or MUCON, respectively, the MTCO and MUCO events are written out immediately on the output file. In this case the copying of trigger and status commons into the bank MTCO, which is listed as logical error above, makes sense; the stored information is really written out on the output file and not immediately deleted, as is the case when using subr. EVREAD. Unfortunately this copying and writing out takes place before the first event has been read and before the first call to RDJETC has taken place. Thus all the updates of the status commons, which will be done by RDTRIG (or RDTRG in the old version) will never be registered on the output file.

This is of course a logical error and people who used the TP-program in this way should be aware that there is no guarantee that the smeared events agree in status commons content and HEAD bank date.

The new version of subr. RDMTCO still does the copying of /CTRIGG/, /CRDSTA/ and /CBIN/ into MTCO, to keep compatibility with the present TP-program. It prints also a warning, to be aware of the nonsense. But for the future it should be considered to change the TP-program to the use of EVREAD and EVWRIT and get rid of the logical errors in the present version.

NOTE ON THE TRACKING PROGRAM:
==========

Minor changes are also being made to the tracking program. These changes are only local and should not affect anybody. Such changes are e.g.

WRTMCB: A new bank HITL is created in the output events. It will be used in connection with the new scheme for hit-track association in the later history of the event. This will be described in a later computer note.

WRTMCB,TRCDET,JHITIN,EVTINI,JSPOIN,JRING,..... The present capacity for registering inner detector hits, 1500, will be extended to 4000. This means enlarging a common, occuring in the above routines.

$HISTORY: This member holds the history of the library

F22ELS.JMC.S and it should be consulted for more information on these minor changes to the tracking program.

CONCERNING INSTALMENT OF NEW ROUTINES ON F22ELS.JMC.S AND L:
==========

The instalment of the corrections is proposed to take place in two steps:
Step 1 starts with the issue of this note; a complete version of the new tracking, smearing and trigger simulation routines is prepared on the libraries:

F22ELS.JMC.S1 and L1.

In order to get hold of any remaining incompatibilities with existing programs or datasets, all MC users and producers are urged to replace the present library F22ELS.JMC.L in their programs with the new version F22ELS.JMC.L1, and see if anything unusual or undesirable happens. Note that the corrected version of EVWRIT is situated on F22ELS.JMC.L1 and therefore this library should be linked before the standard F11LHO.JADEGL.

Step 2: After a successful Step 1 (estimated time c:a 20 days) a renaming will take place:

```
F22ELS.JMC.S   ----->  F22ELS.JMC.S0
F22ELS.JMC.L   ----->  F22ELS.JMC.L0
F22ELS.JMC.S1  ----->  F22ELS.JMC.S
F22ELS.JMC.L1  ----->  F22ELS.JMC.L
```

Simultaneously EVWRIT and RDMTCO on the standard libraries F11LHO.JADEGS and JADEGL will be updated.

Step 2 will take place at a fixed time which will be announced before.

* * * * * * * * * * *

This note can be printed by submitting the member

JADEPR.TEXT(JBJCN66)

An addendum is found in the member JADEPR.TEXT(JBJCN66A)

Figs.1,2 and 3 can be obtained from Mrs. Platz or from J. Olsson.

```
  ********************************************************
****                                                    ****
****      J A D E  C O M P U T E R  N O T E   6 6       ****
****                                                    ****
****                  A D D E N D U M                   ****
****                                                    ****
  ********************************************************
```

J. HAGEMANN, J. OLSSON, R. RAMCKE 07.2.1984

* * * * * * * * * *

In the following, recent changes to some of the smearing subroutines are described.

RDALGN:

The preferrable procedure described in the original note (JCN 66) has now been implemented. The variable IPHALG:

```
COMMON /CRDSTA/ NDUM(12),IPHALG
```

is now used as a flag: IPHALG = 0 no deletion of lowenergy blocks
IPHALG <> 0 deletion of lowenergy blocks

BLOCK DATA setting is now IPHALG=1, i.e. low energy lead glass blocks are deleted when the MC data undergo the normal smearing process. The read out threshold is automatically set to the value which corresponds to the period (given by IHIST(3) in COMMON/CTRIGG/). The pulseheights (in MeV) in ALGN are first converted into ADC-counts with help of the average calibration constant (5.32 MeV/count in 1979-80, 4.94 MeV/count in 1981-82) and then all blocks below the readout threshold (5 counts in 1979-80, 6 counts in 1981-82) are rejected. Surviving blocks get their energy in MeV back by a multiplication, count * calibrationconstant. In this way the electronics of the readout is closely simulated.

Note that the 1983-84 lead glass set up is not yet well simulated, the mixture of SF5 and SF6 has not yet been seriously considered in the MC simulation. Thus the 1982 status is still obtained for these latter periods.

Note also that jobs which standardly set IPHALG=28 are not affected, the low energy block cuts are almost the same as with the old algorithmus.

RDRESO:

The smearing process described in the original note gives a momentum resolution comparable to that of the real data. It was however found that the chisquare distribution of the fitted tracks is about a factor 2 higher than in the real data; also the vertex distribution of fitted tracks is not well simulated. This is partly due to the binning with .380 mm, which makes it difficult to tune the smearing with the parameter RJITT. In the new version, RJITT is used without this binning and in addition, RJITT is made dependent on the distance from the drift-wire (the proportionality constant was given by J. Spitzer). With this change, RJITT can be set lower and is now by default .230 mm (previously .270 mm). This is still larger than the single hit resolution in the real data; the chisquare distribution of fitted tracks is therefore still somewhat worse than for real data, although momentum resolution and vertex distribution are now both reasonably described.

WARNING:
****
Until a still better recipee for inner detector smearing has been found, great caution has to be exercised if one wants to make a cut in the chisquare distribution of fitted tracks, in the course of a physics analysis.

The following code is now used in RDRESO:


```
      DRIINT = FLOAT(IDRI)*BINMC
c
      Z1 = RN(DUM)
      SQLOG=SQRT(-2.*ALOG(RN(DUM)))
      G1=SIN(PI*2.*Z1)*SQLOG
C G1 IS RANDOM AND GAUSSIAN
      RJITT1 = DRIINT*.677E-3 + RJITT
c
      DRIINT = DRIINT + G1*RJITT1
      IDRI = IFIX(DRIINT/BINMC)
```

This note can be printed by submitting the member

JADEPR.TEXT(JBJCN66A)

JADE COMPUTER NOTE 66

ADDENDUM

J. HAGEMANN, J. OLSSON, R. RAMCKE 07.2.1984

In the following, recent changes to some of the smearing subroutines are described.

RDALGN:

The preferrable procedure described in the original note (JCN 66) has now been implemented. The variable IPHALG:

```
COMMON /CRDSTA/ NDUM(12),IPHALG
```

is now used as a flag: IPHALG = 0 no deletion of lowenergy blocks
IPHALG <> 0 deletion of lowenergy blocks

BLOCK DATA setting is now IPHALG=1, i.e. low energy lead glass blocks are deleted when the MC data undergo the normal smearing process. The read out threshold is automatically set to the value which corresponds to the period (given by IHIST(3) in COMMON/CTRIGG/). The pulseheights (in MeV) in ALGN are first converted into ADC-counts with help of the average calibration constant (5.32 MeV/count in 1979-80, 4.94 MeV/count in 1981-82) and then all blocks below the readout threshold (5 counts in 1979-80, 6 counts in 1981-82) are rejected. Surviving blocks get their energy in MeV back by a multiplication, count * calibrationconstant. In this way the electronics of the readout is closely simulated.

Note that the 1983-84 lead glass set up is not yet well simulated, the mixture of SF5 and SF6 has not yet been seriously considered in the MC simulation. Thus the 1982 status is still obtained for these latter periods.

Note also that jobs which standardly set IPHALG=20 are not affected, the low energy block cuts are almost the same as with the old algorithmus.

RDRESO:

The smearing process described in the original note gives a momentum resolution comparable to that of the real data. It was however found that the chisquare distribution of the fitted tracks is about a factor 2 higher than in the real data; also the vertex distribution of fitted tracks is not well simulated. This is partly due to the binning with .380 mm, which makes it difficult to tune the smearing with the parameter RJITT. In the new version, RJITT is used without this binning and in addition, RJITT is made dependent on the distance from the drift-wire (the proportionality constant was given by J. Spitzer). With this change, RJITT can be set lower and is now by default .230 mm (previously .270 mm). This is still larger than the single hit resolution in the real data; the chisquare distribution of fitted tracks is therefore still somewhat worse than for real data, although momentum resolution and vertex distribution are now both reasonably described.

WARNING:
****

Until a still better recipee for inner detector smearing has been found, great caution has to be exercised if one wants to make a cut in the chisquare distribution of fitted tracks, in the course of a physics analysis.

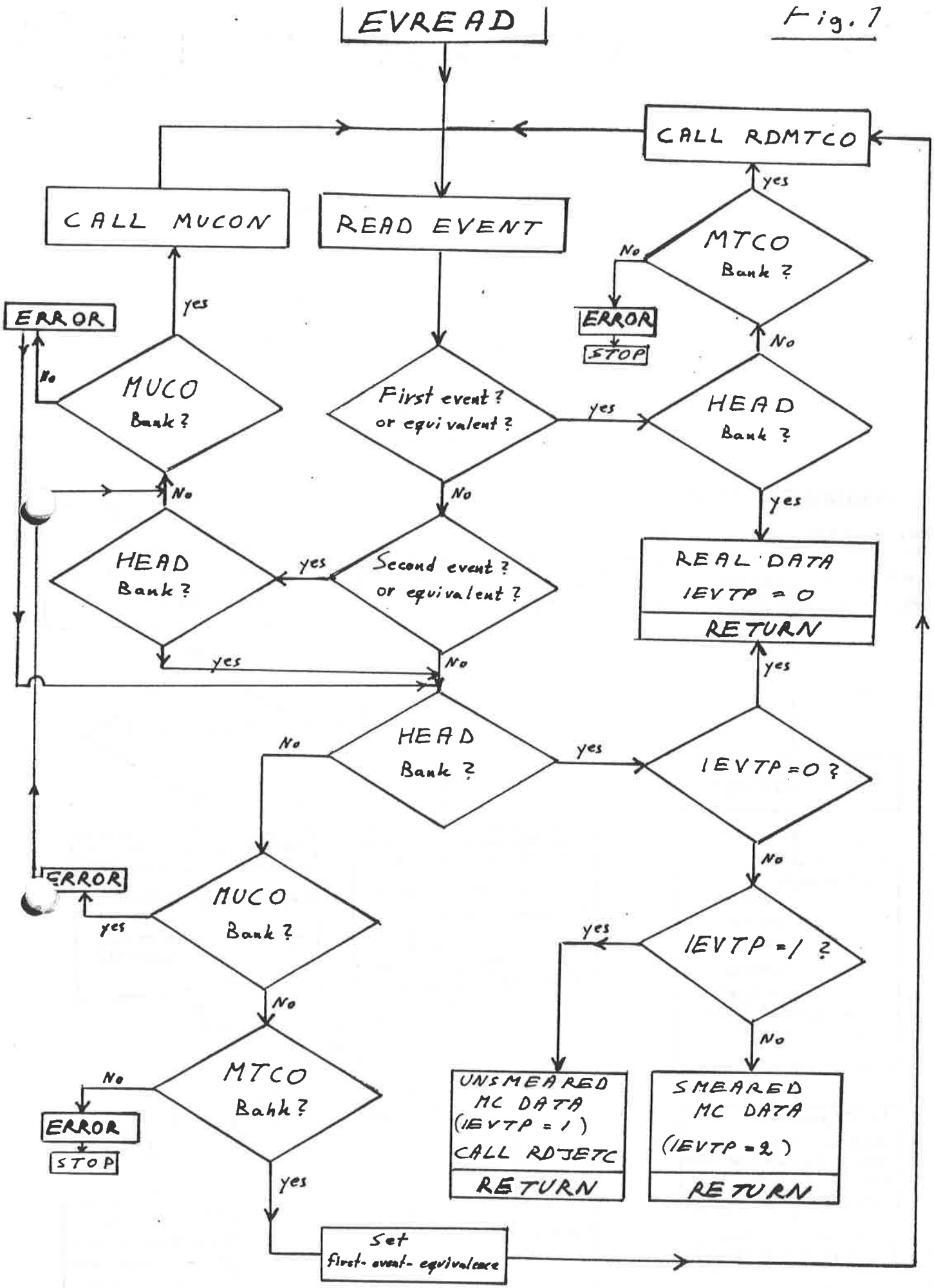
The following code is now used in RDRESC:

```
      DRIINT = FLOAT(IDR1)*BINMC
C
      Z1 = RN(DUM)
      SQLOG=SQRT(-2.*ALOG(RN(DUM)))
      G1=SIN(PI*2.*Z1)*SQLOG
C G1 IS RANDOM AND GAUSSIAN
      RJITT1 = DRIINT*.677E-3 + RJITT
C
      DRIINT = DRIINT + G1*RJITT1
      IDRI = IFIX(DRIINT/BINMC)
```

This note can be printed by submitting the member

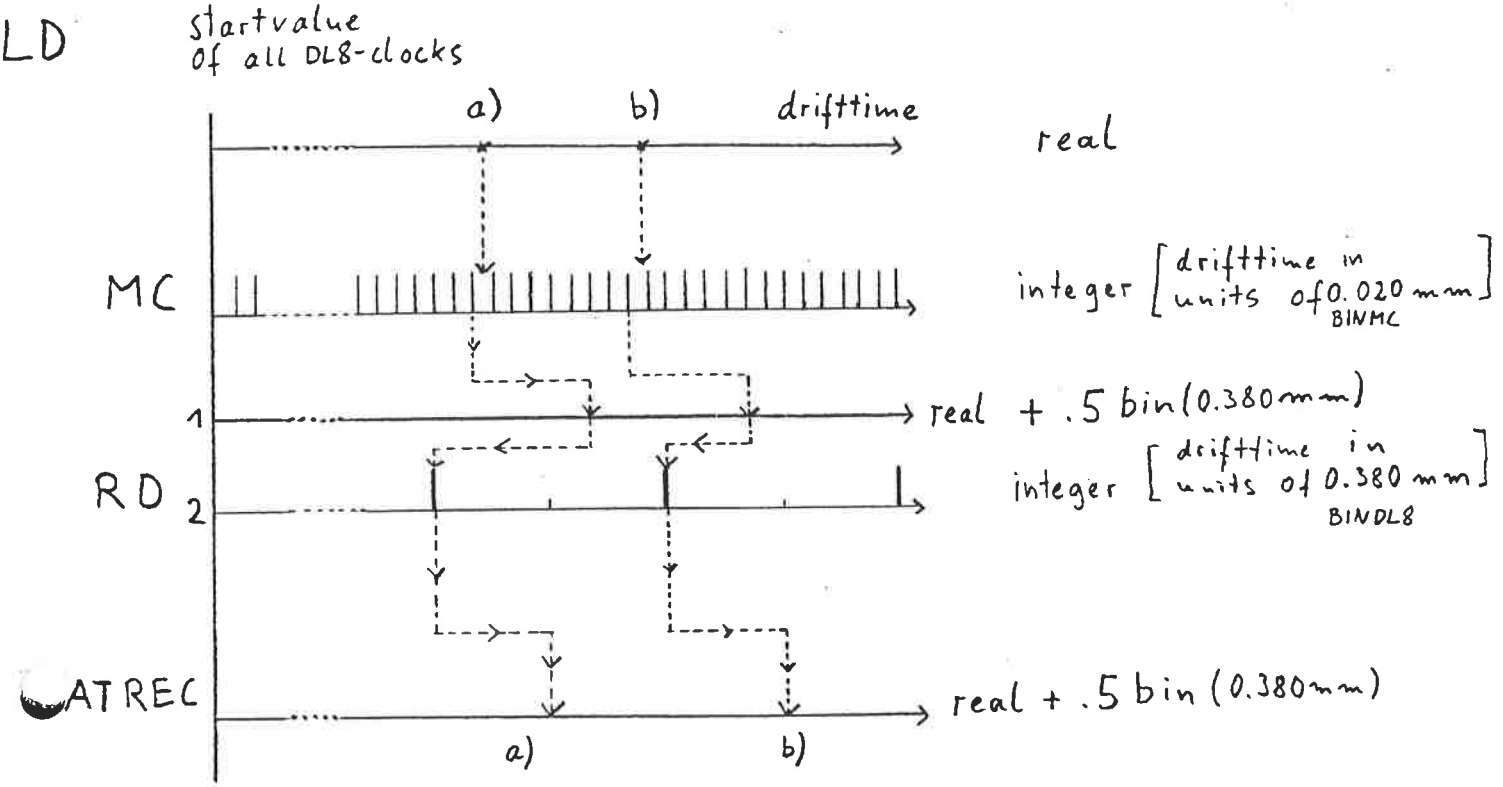
JADEPR.TEXT(JBJCN66A)

Fig. 1


EVREAD
CALL RDMTCO
yes
MTCO Bank ?
No
ERROR
STOP
CALL MUCON
READ EVENT
ERROR
yes
No
MUCO Bank ?
First event ? or equivalent ?
yes
HEAD Bank ?
No
No
yes
HEAD Bank ?
yes
Second event ? or equivalent ?
REAL DATA
IEVTP = 0
RETURN
yes
No
yes
No
HEAD Bank ?
yes
IEVTP = 0 ?
No
ERROR
yes
MUCO Bank ?
yes
IEVTP = 1 ?
No
No
MTCO Bank ?
No
yes
ERROR
STOP
UNSMEARED MC DATA
(IEVTP = 1)
CALL RDJETC
RETURN
SMEARED MC DATA
(IEVTP = 2)
RETURN
Set first-event-equivalence

The Smearing Algorithm in Subroutine KUKESU

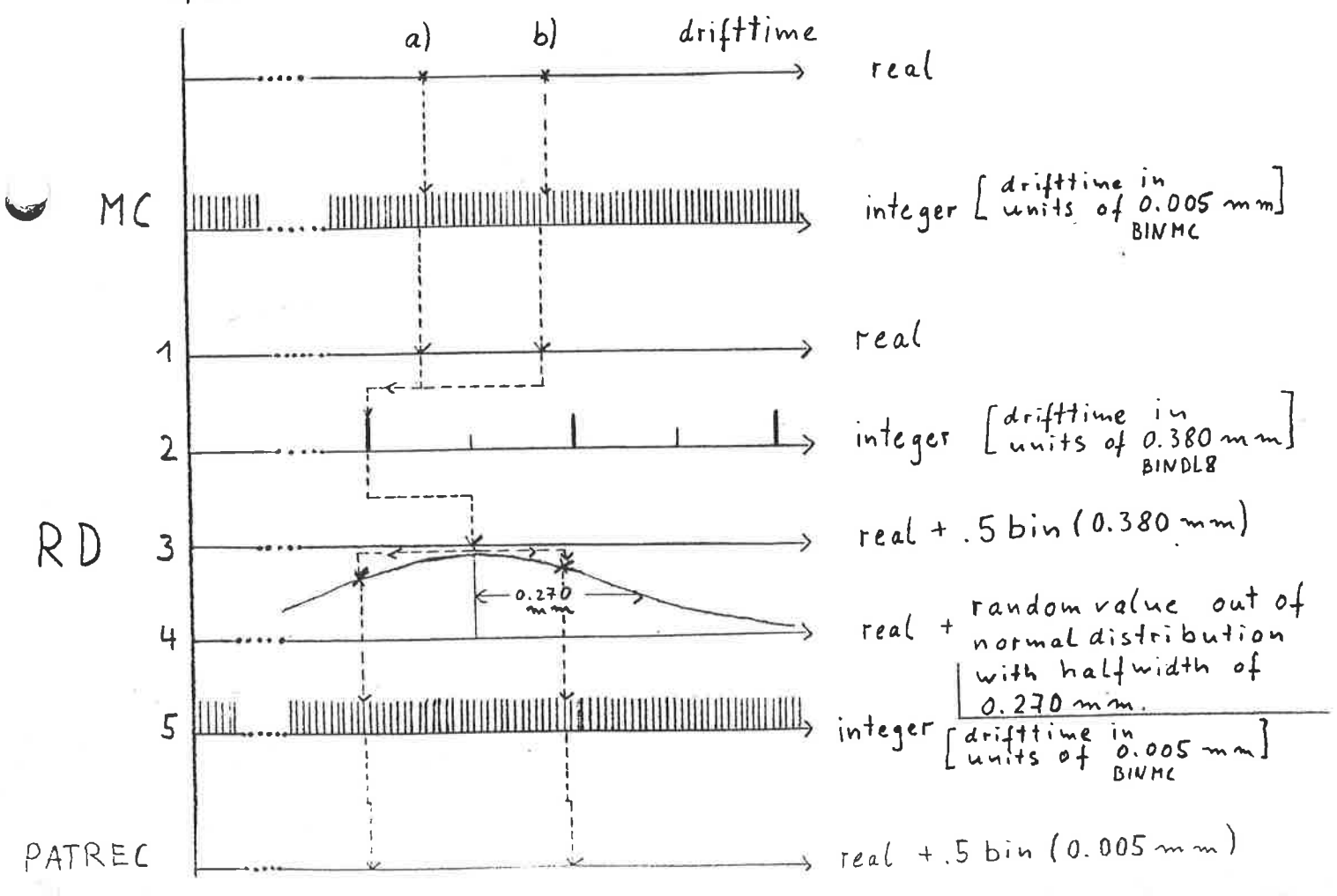
OLD



a) o.k.

b) systematic error

NEW



JADE Computer Note 67

THE JADE MUON MONTE CARLO

Roger Barlow
August 1983

Contents:

Introduction

This describes the routines used to describe nuclear interactions and other processes in the "Muon Monte Carlo".

The routines are in F22RJB.RLMC.S/L at DESY
JADE.LIBRARYS.CASCADE at Manchester (Source)
ULIB.JADEGL at RAL (Load)

At DESY, to use the program you have to put the load library F22RJB.RLMC.L in the SYSLIB DD cards before the standard F22ELS.JMC.L. At RAL this version is the default. In both cases, calibration data files must be given in the standard way.

The original code was taken from Grant [N.I.M. 131 p 167 (1975)], but has been much altered since. The note is for the benefit of (1) people who want to do muon physics and (2) people who might like to use the routines for other purposes (beam pipe, lead glass, etcetera). I have tried to describe in full detail both the program mechanism and also the underlying physics. There are several major areas where both can be improved, especially in the different units used by different routines (GeV-MeV; cm-mm). However, if one waited till the program were perfect before documenting it one would wait forever. Later improvements will be described in addenda to this note, and the existence of these will be advertised by a message in the printout from the program.

ROUTINES

DATEMC(H)

Called by: MUCONM
Calls: none
Arguments/value: H is an array of 6 halfwords set to time and date
Input Common(s): TODAY
Output Common(s): none

Moves the Time and date on which this run is supposed to have happened from the common TODAY where it has been set up by the user, to an array for BOS purposes.
This is separate from the date set up for RDMTCO etc. in the analysis step, as described in JADE Computer note 66. It is up to the user to make sure that the two dates are the same.

MUBREM

Called by: MURTNE
Calls: MUTRAK,MUSCAT
Arguments/value: None
Input Common(s): RESULT,SUN,MAT
Output Common(s): RESULT,SUN,EVENT

Called only if this is an electron. For a particle of momentum P generates a photon of energy uniformly distributed between P/10 and P. Probability of radiating this photon is then taken as 10/9 * P/P(photon) * (step size/radn length). Electron is taken a random distance through the step size before radiating.

Status: OK. Formula is a good approximation above a few hundred Mev - see Perkins section 2.5.2. The low energy cutoff at P/10 is not true, of course, and is only there to avoid generating many time-consuming low energy photons. Anyone interested in electromagnetic showers should move it to a lower value.

MUCONM

Called by: MCJADE
Calls: DATEMC,KALIBR,MUFIX
Arguments/value: None
Input Common(s): CIOUNI
Output Common(s): Bos Common

Initial stage for calibration data etc. It creates a HEAD bank with date given by DATEMC and run number -1 (!), and calls KALIBR which then gets the calibration data for the requested date, including the muon calibration (as the run number is not 0) but does not re-evaluate the Lorentz angle corrections (as the run number is less than 1). The mendacious HEAD bank is then deleted. MUFIX is called to set up muon chamber parameters.

MUDEDX(BETA)

Called by: MUTRAK
Calls: none
Arguments/value: Beta is particle velocity.
Function returns real value of dE/dx
Input Common(s): MAT,RESULT

Computes dE/dx according to Bethe-Bloch formula.
Status: OK
Output Common(s): none

MUDKBG

Called by: MURTNE
Calls: MUROT
Arguments/value: None
Input Common(s): RESULT,SUN
Output Common(s): RESULT,DISPL

Particle decays. Charged pions decay to muons. Charged Kaons decay to muons or to pi pi0.

Status: OK, except that the pi0 is dropped and not tracked.

MUFIX(REST, RESL, EFF, RNOISE, DEBUG, RESEND, PPULSE)

Called by: MUCONM
Calls: none
Arguments/value: See below
Input Common(s): none
Output Common(s): MUPARS

Called by MUCONM at the start of the event. It sets values in MUPARS control common:

REST and RESEND are the transverse resolutions in the barrel and endwall chambers - standard values 5.0 and 10.0 mm
RESL is the longitudinal resolution - standard value 500mm
EFF is the chamber efficiency - standard value 95%
RNOISE is the probability of a random hit in each chamber - standard value 1%
DEBUG is a debug flag (logical) usually set to FALSE unless you want piles of paper.
PPULSE is the probability that a chamber will multi-pulse - standard value 0.75 .

Many people find it useful to have their own version of this routine. If you want to change the calibration data - e.g. to switch chambers on or off - then this is the place to do it.

MUINTA(IND)

Called by: MURTNE
Calls: MUSIGM
Arguments/value: IND is the material index - checked for zero
value returned is 1 if the particle interacted, else 0
Input Common(s): RESULT, MAT, SUN, MUABSL
Output Common(s): none

Decides whether a hadron interacted in this step.
For each element in the material, it finds the absorption length from the density and the cross section. Adds these reciprocally, then compares the absorption length with the step size and decides randomly whether or not an interaction occured. If it did, it then decides what element the struck nucleus was by comparing the absorption lengths.

MUORDR

Called by: MCJADE
Calls: MUSTOR
Arguments/value: None
Input Common(s): CJTCDC,CMUREG,CMUCALIB,CMUSIG,MUPARS
Output Common(s): Bos Common
Input Bank(s): MUX1:1, MUX1:2, MUX1:9
Output bank(s): MUEV:0, MUHC:1

Called at the end of the event to unpack the muon hits and form the MUEV bank for output. First it adds any desired random hits, then sorts the hits into order, then makes MUEV. Hit 4 overwriting hit 1 is done here. Also makes the MUHC bank. Uses MUX1:3,4, and 5 as temporary work space.

MUPAIR

Called by: MURTNE
Calls: MUTRAK
Arguments/value: None
Input Common(s): RESULT,SUN,MAT
Output Common(s): EVENT,SUN,RESULT

For photons, converts to an e+ e- pair with probability exp -(7/9 *step/Rad length). Moves the photon an arbitrary distance along the step before conversion.

Status: OK for photons above 1 Gev. Again, anyone doing EM shower studies should improve it. They should be using EGS anyway.

MUPCN(P)

Called by: MUCAS
Calls: None
Arguments/value: P is momentum of incident particle
Real value returned is probability of charge exchange
Input Common(s): None
Output Common(s): None

This interpolates from a table the probability of charge exchange. This falls from 32% to 1%.

Status: Very insecure. I don't know where these numbers came from or how good they are.

The MUCH bank is created, and the information about the particle is stored there. The routine has to spot the first decay/interaction/rangeout/escape of this particle.
"Rangeout" happens when the particle momentum falls below PCUT.

The routine steps the particle through the detector, at each step offering the possibilty of decay, interaction, etc, and losing energy. When the tracking of this particle stops (for one of the 4 above reasons) then the next particle is taken from EVE and tracked (n.b. this will have been created in the cascade; the next particle from MCJADE doesn't appear till the next MURTNE call). This continues until there are no particles left.

There is an entry MUSEED(ISEED) which can be used to set the seed used for the random number generator.

MUSCAT

Called by: MURTNE
Calls: None
Arguments/value: None
Input Common(s): RESULT,SUN,MAT,KUT
Output Common(s): RESULT

Coulomb scattering a la Particle data book.
theta rms = 15 sqrt(x/X) (1+1/9 log(x/X)) / p beta

status: OK

MUSEED(ISEED)

Called by: The user, if desired.
Calls: None
Arguments/value: ISEED is the seed to be used
Input Common(s): None
Output Common(s): None

This is an entry in MURTNE. It sets the value of the seed for the Random Number generator.

MUTRAK

Called by: MURTNE,MUPAIR,MUBREM
Calls: MUDEDX,MUMAGF
Arguments/value: None
Input Common(s): RESULT,SUN,CGEO1
Output Common(s): RESULT

Moves particle a step through the detector; energy loss
and magnetic field bending are applied

MUTRYN(N,NZ,INSIST)

Called by: MUCAS
Calls: None
Arguments/value: N is the number of charged particles
NZ is the number of pi zeros
INSIST is the minimum no of particles needed
Input Common(s): RESULT
Output Common(s): None

Decides on the Multiplicity of a generated event
if the beam energy is below 1 Gev, essentially nothing happens.
Otherwise the mean and dispersion of the charged multiplicity
distribution are given by a linear dependance on log s

For pi+ and proton beams
$\langle N \rangle = 1.04 + 0.91 \log s$ $\langle D \rangle = -0.405 + 0.497 \log s$
For pi- and neutron beams
$\langle N \rangle = -0.81 + 1.474 \log s$ $\langle D \rangle = 0.07 + 0.493 \log s$

If the energy is between 1 and 4 Gev, then the prong number is
given by a gaussian using the values of <N> and <D>
Above 4 Gev, the (well known) Czyzewski-Rybicki formula
is used [Nuclear Physics B 47 p 633 (1972)]

The expected number of pi zeros is taken as <N>/3
the actual number is then given by a poisson distribution.

Status: I have not checked the behaviour of <N> and <D>.
The C-R formula is fine. The pi zero assumptions are unchecked
but seem not wildly unreasonable.

MUSGRD(A1,A2)

Called by: MUSGNS
Calls: MUSGRN
Arguments/value: A1,A2 are momenta generated
Input Common(s): COUNT
Output Common(s): NONE

Generates an exponential Pt and a uniform azimuth, transforms to x and y, and returns the values.

MUSGRN(R,N)

Called by: MUSGRD
Calls: None
Arguments/value: R contains N random numbers
Input Common(s): PSRAND
Output Common(s): None

Gemerates random numbers - uniformly between 0 and 1

MUSGWT(W)

Called by: MUCAS
Calls: None
Arguments/value: W(2) is a real array
Input Common(s): SAGELL,SAGEWT
Output Common(s): SAGEWT

Sage routine. Fills W from commons produced by the last event generated by MUSGNS. W(2) is the weight

MUZAP(JREG)

Called by: MUFIND
Calls: MUSTOR
Arguments/value: JREG is the region in the filter
Input Common(s): RESULT,CMURJB
Output Common(s): CMURJB

Called when the particle is in a muon chamber region. Checks that it hasn't already just hit this chamber, then works out the number of the chamber it would hit, the transverse and longitudinal coordinates, and calls MUSTOR to register the hit.

COMMON/JEVENT/JEVT

Event number. Set in the standard part of the MC

Used by: MURTNE

COMMON/XUT/ PCUT

PCUT Low momentum cutoff.

Used by: MUNUCL,MURTNE,MUSCAT

COMMON/MAT/DEN,RADLTH,ATWT,ABSL,DDXA,DDXB

Information on material properties

DEN	Density (units Nuclei/gram times 10**-20)
RADLTH	Rad length (cm)
ATWT	Average atomic wt for this material
ABSL	Not used.
DDXA	Energy loss parameters
DDXB	used by MUDEDX

Filled by MUMAT

Used by: MUBREM,MUDEDX,MUEXNU,MUINTA,MUPAIR,MUSCAT

COMMON/MUABSL/NMUABS,ATWTS(20),DENUCS(20),RLENGT(20)

Info about the materials in this composite

NMUABS	Number of elements
ATWTS	their Atomic weights
DENUCS	their densities (Units N/g *1.0E-20)
RLENGT	The reciprocal absorption length of each

Filled by: MUMAT

Used by: MUINTA

COMMON/MUSIGA/ SIG(162,5)

Nucleon target cross sections

See listing of MUSIGM for exact details

used by: MUSIGM

COMMON/MUTYPE/ITYPE,IREGIO

ITYPE	Region type a la MUREG
IREGIO	Muon region number

Filled by MUFIND

Used by MUMAGF

BANKS

MUCH:n Information on track n.
5 words long

WORD	1	MASS
	2	FATE
		1=INTERACTED
		2=DECAYED
		3=RANGEOUT
		4=LEFT
	3	}
	4	} x,y,z of position where fate struck
	5	}
	6	Number of muon chamber hits made by this track

Made by: MURTNE
Output

MUEV:0 Standard MUEV bank.
Made by: MUORDR
Output

MUHC:1 3 integer*4 words/hit: chamber number
transverse co-ordinate
track number
Made by: MUORDR
Output

MUX1:1 Contains transverse values of hits
Made by MUSTOR, read by MUORDR
Temporary

MUX1:2 Contains Longitudinal values of hits
Made by MUSTOR, read by MUORDR
Temporary

MUX1:3 Temporary work bank used by MUORDR

MUX1:4 Temporary work bank used by MUORDR

MUX1:9 Contains track info.
Made by MUSTOR, read by MUORDR
Temporary

Addendum to JADE Computer Note 67

Hugh McCann

October 1983

Since the Muon Tracking Monte Carlo uses the actual calibration which was valid on the date set by the user, it is necessary for the user to supply a date which complies with his requirements. In general, the user requires a calibration version which reasonably represents the conditions under which his real data were recorded. To that end, the following dates can be recommended (the time is always to be taken as 00 : 00 : 00):

Date	$\sqrt{s}$ (GeV) at that Date	# inoperative μ chambers / # installed
20.8.1980	> 30 (scan)	∿ 30/622
25.3.1981	∿ 34	∿ 20/622
21.7.1981	14 (22)	∿130/622
		Use this date also for 22 GeV simulation.
1.11.1981	∿ 35	∿ 70/622
1. 7.1982	34.6	53/618
1.11.1982	∿ 39 (scan)	86/618
		Of these, 74 were in the layer inside the magnet return yoke.
1. 6.1983	> 40 (scan)	8/618

On these dates, the situation represents a reasonable average for the given run period. For an average over our entire data sample at present, 1.7.1982 is a reasonable choice. The user should also take careful note of the comments in J.C.N. 67 page 3 regarding RDMTCC.

JADE COMPUTER NOTE 68 THIS IS JADEPR.TEXT(NOTE68)

P. STEFFEN, 83/09/21

THE JADE CALIBRATION SCHEME

The jade calibration constants are available to off line programs in COMMON /CALIBR/. THE constants are updated whenever KALIBR/KLREAD is called (for each run). As a consequence one has always the current constants in /CALIBR/ available, which are relevant to the event beeing processed.
The different sets of constants are on the disk-file 'F11LHO.AUPDAT1' (, or on the two files 'F11LHO.BUPDAT0', 'F11LHO.BUPDAT1').
The status and the recent changes of the calibration files are described in 'F11PST.LHOLIB.S(#CALNEWS)'.

A. Structure of COMMON /CALIBR/.

1. COMMON /CALIBR/ ACALIB(1000)
   DIMENSION HCALIB(2000), ICALIB(1000)
   EQUIVALENCE (ACALIB(1),ICALIB(1),HCALIB(1))
   The actual length of /CALIBR/ words is set in the SUBROUTINE KLREAD.

2. The first 100 locations are foreseen for pointers and administration:
   IDATA(1): POINTER TO MUCA-constants
   IDATA(2): POINTER TO LGMA-constants
   IDATA(3): POINTER TO TAGS-constants
   IDATA(4): POINTER TO JTPL-constants
   IDATA(5): POINTER TO JTAB-constants
   IDATA(6): POINTER TO TOFC-constants
   IDATA(7): POINTER TO LGST-constants
   IDATA(8): POINTER TO DEDX-constants
   IDATA(9): POINTER TO SPTG-constants
   IDATA(10): POINTER TO RVTX-constants
   IDATA(11): POINTER TO RCON-constants
   IDATA(12): POINTER TO TAGF-constants

   e.g. IPVRVTX = IDATA(10) : pointer to run-vertex coordinates
   XV = ADATA(IPRVTX+1): 1. constant = x(vertex)

   If the constants consist of half-words one has
   e.g. IPJTPL = IDATA(4)*2 : pointer to jet chamber constants
   IT0 = HDATA(IPJTPL+1) : 1. constant = T0(1.wire)

3. It is JADE convention that the MUCA-constants are the first set of constants and they always start at ADATA(100).

4. The following different sets of constants are at present available:

MUCA : Mu-chamber constants	H. McCann	(Man.)
LGMA : lead glass constants	M. Minowa	(Tok.)
TAGS : tagging constants (obsolete)	H. Wriedt	(Lan.)
JTPL : jet chamber wire constants	R.D. Heuer	(Hei.)
JTAB : jet chamber cell constants	P. Steffen/J.Spitzer	(Hei.)
TOFC : time of flight constants	B. Naroska	(Desy)
LGST : lead glass "spinning blocks"	M. Minowa	(Tok.)
DEDX : dE/dx calibration constants	S. Bethke	(Hei.)
SPTG : tagging "spinning blocks"(obsolete)	H. Wriedt	(Lan.)
RCON : not jet established		
RVTX : run dependent event vertex	S. Komamiya	(Hei.)

TAGF : tagging constants (1982 ...) A. Finch (Lan.)

B. Structure of the Calibration Files

1. Calibration data for different periods are stored on the files:
   F11LHO.BUPDAT0 : constants up to run 10 000
   F11LHO.BUPDAT1 : constants from run 10 000on
   A compressed version of both files is on F11LHO.AUPDAT1, which contains no LGST and SPTG constants ("spinning block" constants). This file is commonly used. The constants, which are left out, are used in general only in the REDUC1-step.

   A record on the UPDAT-file may contain a complete set of constants as well as update values for a limited number of constants.

2. The record of the calibration file have the folllowing format:
   1. word : LENGTH = number of following words
   2. word)
   3.)
   :) : data
   :)
   (length+1))

   The records can be read and written with the statements
   DIMENSION IBUF(2009)
   READ (22) LENGTH, (IBUF(I1),I1=1,LENGTH)
   WRITE(22) LENGTH, (IBUF(I1),I1=1,LENGTH)

3. The first record on a calibration file has only one data-word:
   LENGTH = 1
   IBUF(1) = time at which the data on the calibration file start to become valid.
   According to this time the KLREAD-subroutine selects the calibration file (e.g. BUPDAT0 or BUPDAT1).

4. The following records have a LENGTH >= 9.
   The first 9 data words contain a header. The following words contain the calibration constants:
   IBUF(1) = name of constants (e.g. MUCA, DEDX, see A.4.)
   IBUF(2) = current number of records for the same set of const.
   IBUF(3) = total number of records for the same set of const.
   IBUF(4) = unused
   IBUF(5) = time at which the constants have been established
   IBUF(6) = time from which on the constants are valid
   IBUF(7) = 0, if record contains updates of selected words within a set of constants.
   = I, if record contains updates of consecutive words within a set of constants;
   I = 1.location within the set of constants which shall be replaced by the constants of the record.
   Exemples are given in part B.7.
   IBUF(8) = run number from which on the constants are valid
   IBUF(9) = unused
   IBUF(10) = data
   IBUF(11) = ...
   ...
   ...

5. The last record has LENGTH = 9 and consists only of a header. The time (IBUF(6)) and the run number (IBUF(8)) are set to a very large value.

6. The records are limited in lenghth to 2000 words. A longer set of constants is split in two or more records.
   E.g. a set of 3000 constants(full words)
   1. record: LENGTH = 1509

```
                    words 1-9  : header (word 7(1.location) =    1)
                    words 10...: constants    1 - 1500
          2. record: LENGTH = 1509
                    words 1-9  : header (word 7(1.location) = 1501)
                    words 10...: constants 1501 - 3000
     Or a set of 6000 constants(half words)
          1. record: LENGTH = 1509
                    words 1-9  : header (word 7(1.location) =    1)
                    words 10...: constants    1 - 3000
          2. record: LENGTH = 1509
                    words 1-9  : header (word 7(1.location) = 3001)
                    words 10...: constants 3001 - 6000
```

7. Updates of subsets of constants.
   There is a complete set of constants for the different periods of data taking. Within a period it is in general only necessary to update a subset of the complete set of constants. There are two possibility for replacing such a subset of constants.
   Exemples:
   1. Replace a subset of 6 consecutive constants starting with the 27th-word of the complete set of constants:

```
      record: word 7     = 27
              word 10-15 = constants to replace
                           words 27-32 of the complete set.
```

   2. Replace the 4th, 26th and 138th constant(full words)

```
         record: word 7     = 0
                 word 10    =   4 : location within the complete set
                                    to be replaced
                 word 11    = replacement of   4th constant
                 word 12    =  26
                 word 13    = replacement of  26th constant
                 word 14    = 138
                 word 15    = replacement of 138th constant
                  ...          ...
      Replace the 4th, 26th and 138th constant(half words)
         record: word 7      = 0
                 halfword 19 =  4 : location within the complete set
                                    to be replaced
                 halfword 20 = replacement of   4th constant
                 halfword 21 =  26
                 halfword 22 = replacement of  26th constant
                 halfword 23 = 138
                 halfword 24 = replacement of 138th constant
                  ...          ...
```

C. Correlation Files of Run Number and Time of Data Taking

1. There exist two files, which contain for each run the time used for updating the calibration constants. The two files are
   F11LHO.DSKTI000 : runs 539 ... 9728
   F11LHO.DSKTI001 : runs 1000 ...
   The records are ordered according to the calibration time.

2. The records have a fixed length of 9 words:

```
word 1 : run number
word 2 : calibration time
word 3 : second      )
word 4 : minute      )
word 5 : hour        )  time at run start
word 6 : day         )
word 7 : month       )
word 8 : year        )
word 9 : calibration time of REFORM step
```

3. The second file F11LHO.DSKTI001 is continuously updated by the REFORM-job. The new records are added to the end of the file. As the runs do not always come in the proper order the file must be reordered regularly (about once/month during data taking and after the last REFORM-job of a running period) (see C.5).

   If new constants for the current running period are installed it is always recommended to reorder the correlation file before installing the new constants.

4. With the help of the correlation file the subroutine CVDTRN(IRUN,ITIME) converts run# into calibration time and vice versa. If one of the arguments (IRUN or ITIME) is zero the proper value will be returned.

5. The reordering of the correlation file F11LHO.DSKTI001 is a rather delicate operation, which must be left to specialists. For completeness of this note some gross details of this operation are descibed:
   In order to avoid any interference between the REFORM-job, which adds new records to the end of the file, and the ordering we pursue the following procedure:
   1. Produce a backup copy of F11LHO.DSKTI001 for safety.
   2. Allocate F11LHO.DSKTI001 to your NEWLIB-session:
      ALLOC F(SAFE) DA('F11LHO.DSKTI001').
   3. Run the job JBLORDER with high priority:
      writes ordered correlation file onto intermediate file.
   4. Check the printout of JBLORDER (no fatal error occurred).
   5. Submit the job JBLCOPY with high priority.
   6. Wait until FREE-request of F11LHO.DSKTI001:
      F11LHO** (....) needs F11LHO.DSKTI001,
      F11LHO** is the name of the submitted JBLCOPY-job.
   7. If any other job (e.g. a REFORM-job) request a FREE, one must
      CANCEL F11LHO**, the JBLCOPY-job,
      FREE DA('F11LHO.DSKTI001') and
      try later again starting from 1.
      In this case a destructive interference has occured.
   8. Otherwise one follows the FREE-request:
      FREE DA('F11LHO.DSKTI001')
   9. Check the printout of JBLcopy (no fatal error occurred).
   10. Don't forget to update #CALNEWS.

D. Service of the Calibration Files

0. Also this operation is rather delicate and must be left to specialists. Nevertheless the details are descibed here for completeness of this note.

1. One must avoid complications arising from jobs, which update the calibration constants, and other jobs, which merely read the calibration data.
   Therefore we use the file F11LHO.KALWRK0 for updating constants and as the master file for F11LHO.BUPDAT1 (AUPDAT1). KALWRK0 is copied frequently to BUPDAT1 (AUPDAT1) when constants are added or changed.

2. Updating of the calibration files is performed by two different jobs:
   - The REFORM-job, which adds LGST-constants ("spinning block" constants.
   - The so-called KADD-jobs, which add or change all other constants.

3. In order to avoid any interference between the REFORM-job and the KADD-job one procedes as follows for the KADD-job:
   1. Produce a backup copy of F11LHO.KALWRK0 for safety.
   2. Allocate F11LHO.KALWRK0 to your NEWLIB-session:
      ALLOC F(SAFE) DA('F11LHO.KALWRK0').
   3. Run the job JBKADD* with high priority:
      write updated calibration file onto intermediate file.
   4. Check the printout of JBKADD* (no fatal error occurred).
   5. Submit the job JBKALCOP with high priority.
   6. Wait until FREE-request of F11LHO.KALWRK0:
      F11LHO** (....) needs F11LHO.KALWRK0,
      F11LHO** is the name of the submitted JBKALCOP-job.

```
 7. If any other job (e.g. a REFORM-job) request a FREE, one
    one must CANCEL F11LHO**, the JBKALCOP-job,
             FREE DA('F11LHO.KALWRK0') and
             try later again starting from 1.
    In this case a destructive interference has occured.
 8. Otherwise one follows the FREE-request:
             FREE DA('F11LHO.KALWRK0')
 9. Check the printout of JBKALCOP (no fatal error occurred).
10. Copy KALWRK0 --> BUPDAT1  (JBKALCOP-job).
11. Produce new AUPDAT1- file from BUPDAT0+KALWRK0 (JBCOMB1-job).
12. Check printouts.
13. Don't forget to update #CALNEWS.
```

SCN 69

MONTE CARLO TRACEBACK 28/10/83
@
JADE COMPUTER NOTE 69 C. BOWDERY

MONTE CARLO TRACEBACK

Introduction

In some fields of analysis, it is very important to be able to trace the history of a given Monte Carlo-generated Jet Chamber track. This JADE Computer Note will explain how this is now possible. Section A will explain an easy way to access the history information in an analysis program. Unfortunately this only applies to Monte Carlo events! For those interested in the implementation details, section B will explain the traceback from a PATR track to a VECT bank particle (a '4-vector') and section C will explain the traceback from a VECT bank particle to its origins in an e+e- collision and the conventions used in the new PALL bank. Please note that the word 'track' always means a PATR track and the word 'particle' always means a '4-vector' in this note.

IMPORTANT: The PATR Traceback will only work with events tracked since the 20th October 1983. Old events however will not cause the programs to crash. Also the 4-vector History Traceback will only work for events conforming to the PALL conventions that are explained in Section C.

@
A) Accessing the Monte Carlo Traceback

Two easy-to-use subroutines have been written to provide the user with the essential history information compiled during event generation, tracking and basic analysis. The first subroutine links a given PATR track to one or more 4-vector particles in the VECT banks.

```
DIMENSION  IPART(3), IVECT(3), FRACT(3)
.......
.......
CALL MCTRCB( ITRACK, NPART, IPART, IVECT, FRACT )
```

```
Input:  ITRACK =  Track number in the most recent PATR bank
Output: NPART  =  Number of 4-vectors associated (usually 1)
        IPART  =  Array of associated VECT particle numbers
        IVECT  =  Array of VECT bank numbers corresponding to IPART: 0/1
        FRACT  =  Array of hit fractions for each associated particle
```

If ITRACK is less than 1 or greater than the maximum number of tracks, MCTRCB will return with NPART set to -2 after printing a message on unit 6 (maximum of 10 messages). If NPART is -1 then there was a problem finding the TR4V bank (see Section B). This means that no traceback information was available which could mean that old Monte Carlo events are being processed. Otherwise NPART can take any value between 0 and 3. IVECT elements can be 0 or 1 signifying the traced 4-vector belongs to VECT/0 and VECT/1 respectively. FRACT values lie in the range 0.0 to
@
1.0 and signify the fraction of hits associated with ITRACK that were caused by the particular 4-vector. The sum of all FRACT's can exceed 1.0 if hits are 'claimed' by more than one 4-vector.

A second or third 4-vector will be associated with ITRACK if at least 8 hits belong to a second (or third) 4-vector and that 4-vector is not associated with another PATR track with a higher hit fraction. (Only 5

hits are required if the total number of hits on the PATR track is less than 20.) There will probably be some cases when MCTRCB only returns one associated 4-vector when there should have been 2 and vice versa but this should not be a big problem.

Examples

a) PATR track 5 was caused by 4-vector particle 7 in VECT/0.

```
==> NPART = 1  ,  IPART(1) = 7  ,  IVECT(1) = 0  ,  FRACT(1) = 1.00
```

b) PATR track 3 is a fit through the hits of 4-vector 2 in VECT/0 which decayed to 4-vector 9 in VECT/1.

```
==> NPART = 2  ,  IPART(1) = 2  ,  IVECT(1) = 0  ,  FRACT(1) = 0.39
                  IPART(2) = 9  ,  IVECT(2) = 1  ,  FRACT(2) = 0.61
```

c) PATR track 10 is the result of two very close particles, 7 in VECT/0 and 8 in VECT/0.

```
==> NPART = 2  ,  IPART(1) = 7  ,  IVECT(1) = 0  ,  FRACT(1) = 1.00
                  IPART(2) = 8  ,  IVECT(2) = 0  ,  FRACT(2) = 0.98
```

@ The second routine, MCHTRB, provides a traceback from a 4-vector in VECT/0 to its ancestors in the PALL bank (if filled according to the conventions given in Section C).

```
DIMENSION  P9VECT(9,30) , IPALL(30)
.......
.......
CALL MCHTRB( I4VECT, NFOUND, P9VECT, IPALL, IFLAVR, IQG, IPN )
```

```
Input:  I4VECT  = Particle number in VECT/0
Output: NFOUND  = No. of ancestors found including original particle
        P9VECT  = Array of 9-vectors for each of the NFOUND particles
        IPALL   = Array of pointers to PALL for each found particle
        IFLAVR  = Flavour of the event from PALL header (if relevant)
        IQG     = Index pointer to P9VECT for the parton ancestor
        IPN     = Parton parent order number (see Section C)
```

For each particle there is a 9-vector of information in P9VECT:
px, py, pz, E, m, charge, type, parent_number, parton_parent_number

The order of the particles in P9VECT is as follows:

```
1   original particle corresponding to VECT/0 entry
2   parent particle
3   grandparent particle
.       .......
.       .......
.   parton ancestor ( quark or gluon)              -- IQG points to here
.   virtual spin 1 boson ( photon or Z )           -- if present
.   initial particles ( e+,e-,radiated photon )    -- if present
```

@
Please note that ALL the components of P9VECT are REAL*4 (even the type, charge and pointers which are usually INTEGER*4) in order to simplify the interface. Full details of the 9-vectors are given in Section C. Please also note that the parent_numbers here are the pointers to the parents in the PALL bank. Obviously the position of a parent in P9VECT is just the next particle down in the list. The type values depend on which 4-vector generator was used to produce the Monte Carlo events.

N.B. If NFOUND is 0 then this may mean that the PALL bank does not conform to the convention defined in Section C. This will be the case for MC events which only have 'produced' particles stored in the first part of the event record (see Section C).

```
NFOUND = -1     no VECT bank  (no warning message printed)
       = -2     no PALL bank  (no warning message printed)
```

```
            = -3     I4VECT particle number illegal (message printed)

Both these routines reside on 'F11LHO.JADEGS/L'.  MCHTRB needs s/r VZERO
                                                  from the CERN library.

B)  PATR Traceback
    --------------

   In order to systematically traceback a PATR track to its 4-vector
origin, it is first necessary to store the association of the jet
chamber hits with the 4-vector particle that produced them. Since the
hit/track assignment of hits made in PATREC is stored in the JHTL (hit
label) bank, it is then possible to associate PATR tracks with
4-vectors. However, this simple outline is complicated by jet chamber
smearing, random hits, measuring inefficiencies and overlapping tracks.
We will consider the problem and its implemented solution in 3 stages.
@   Stage 1 :      4-vector particle / JETC hit association during
                   the Detector Simulation ("Tracking")
    Stage 2 :      Maintaining the association during JETC "Smearing"
    Stage 3 :      After PATREC, association of PATR tracks to 4-vector
                   particles in the VECT banks.

  Stage 1  (During Tracking)
  --------------------------

   The basic idea at stage 1 is to record the 4-vector particle number
in a BOS bank ('HITL') whenever a jet chamber hit is stored in the JETC
bank during tracking. The implementation of this scheme was slightly
complicated by the fact that the jet chamber hits are first created and
then sorted (by wire numbers and drift times). To overcome this
problem, the hit label information is simultaneously sorted with the
hits.
          HITL Bank Structure
          -------------------

     Name of bank    :   HITL                          (= Hit Label)
     No. of bank     :    8                            (= JETC bank number)
     Length          :  No._of_JETC_hits + 1   half-words    (+1 if odd)

     Half-word       Contents                Comment
     ---------       --------                -------

         1             HITS          Number of (unsmeared) JETC hits
         2             HWORD1        Encoded 4-vector info for hit 1
         3             HWORD2           "       "      "   "   "  2
         .               .                        .
         .               .                        .
         .               .                        .
     HITS + 1          HWORDn        Encoded 4-vector info for last hit
@
 The code used for the HWORDs is:   2 * particle_number + VECT_bank_no.

      e.g.   Particle  5 in VECT/0  would be code 10
             Particle 11 in VECT/1  would be code 23

 The following subroutines were modified to implement stage 1:

   JRING   :    Hit/particle association recorded here when hit created
   JHITIN  :    Hit/particle association information sorted here
   WRTMCB  :    HITL bank created and filled with the association info.


  Stage 2  (Reading Unsmeared Tracked Events)
  -------------------------------------------

   When the jet chamber hits are smeared in RDMTCO, the corresponding
hit label information has to be updated. Basically, stage 2 creates a
new BOS bank ('HTSL') which links the unsmeared hits with the new,
smeared ones. The following features had to be considered:

   Double hit resolution:   Hits lost due to the double hit resolution
```



```
                            cut are linked to the hit that absorbed them
                            but marked with a minus sign.
   Inefficiency losses:     Hits killed for this reason are marked with
                            a zero.
   Random hits:             These are not associated with any unsmeared
                            hits. They just alter the hit numbering.

@          HTSL Bank Structure
           -------------------

     Name of bank    :   HTSL                          (= Smeared Hit Label)
     No. of bank     :    8                            (= HITL bank number )
     Length          :  No._of_JETC_hits + 1   half-words    (+1 if odd)

     Half-word       Contents                Comment
     ---------       --------                -------

         1             HITS          Number of (unsmeared=old) JETC hits
         2             HLINK1        Pointer to smeared hit for old hit 1
         3             HLINK2           "       "      "   "   "  "  2
         .               .                        .
         .               .                        .
         .               .                        .
     HITS + 1          HLINKn     Pointer to smeared hit for last old hit


  Thus:  HLINKi = 0     means unsmeared hit 'i' has been 'killed'
                > 0     means that the unsmeared hit 'i' is linked to a
                        smeared hit with number HLINK
                < 0     means that the unsmeared hit 'i' was lost due to
                        the double hit resolution. Smeared hit | HLINK |
                        was the hit that absorbed it.

 The following subroutines were modified to implement stage 2:

   RDMERG  :    Re-orders hits after creation of random hits in RDRDMH.
   RDDOUB  :    Double hit resolution cut applied here.
   RDPOIN  :    Edits out 'killed' and 'absorbed' hits.
   RDMODN  :    Now includes code to create the HTSL bank.
@ Stage 3  (Analysis of Smeared Data)
  -----------------------------------

   After PATREC has created the PATR and JHTL banks containing the
results of the track search, stage 3 of the traceback scheme can be
carried out. This will normally be done in the SUPERVISOR but users can
perform this task themselves by calling subroutine MCTR4V. This
processes the PATR, JHTL, HITL and HTSL banks in order to create a new
BOS bank ('TR4V') containing the desired traceback from every PATR track
to a 4-vector particle. Additionally, MCTR4V is called in MCTRCB (see
Section A) to create the TR4V bank there if it does not already exist.

 The following FORTRAN statement invokes MCTR4V:
      ....
      ....
      CALL MCTR4V( IOPT, IERROR )
      ....
      ....

 where:   IOPT  is an input parameter:  0 = perform stage 3 if not done
                                           already for the latest PATR
                                           bank, i.e. create TR4V bank
                                        1 = delete old stage 3 results
                                           for latest PATR bank and redo
          IERROR is a return code:      0 = no errors
                                        1 = BOS error. TR4V not created
                                        2 = HITL bank not same length
                                           as HTSL bank. No TR4V bank
                                           was created.

   The information from the HTSL and HITL banks are unpacked into an
 array holding up to 4 associated 4-vector particles for each smeared jet
```

chamber hit. Then the JHTL bank is decoded to determine the associated
@
PATR track (or 2 tracks). Finally each PATR track is considered in turn; its hits are examined and their 4-vector origins are histogrammed. The particle with the largest number of histogram entries is taken as the most likely cause of the PATR track. The particle number (encoded as 2 * particle_number + VECT_bank_number) is stored in the TR4V bank along with the number of hits that were associated with it.

Additionally, if there exists a second (or even a third) candidate 4-vector particle that has at least 8 hits associated (5 if the total number of hits for the track is less than 20), a second (and third) entry is made in TR4V for that PATR track.

TR4V Bank Structure

Name of bank : TR4V (= Track_to_4-Vector Link)
No. of bank : = latest PATR bank number
Length : 8 * (No._of_PATR_tracks) + 1 words

Full-word	Contents	Comment
1	NTRKS	Number of PATR tracks
2...9	NRES1	8 words with results for track 1
10...17	NRES2	8 words with results for track 2
18...25	NRES3	" " " " " " 3
.	.	. . .
.	.	. . .
NTRKS*8-6...NTRKS*8+1	NRESn	8 words with results for last track

@
where NRESi stands for 8 words as follows:

1	NAPART	No. of associated 4-vector particles
2	NTOTHT	Total no. of hits along the PATR track as recorded in the JHTL bank by PATREC
3	IPART1	Encoded word for most probable 4-vector
4	NHITS1	No. of smeared hits belonging to IPART1
5	IPART2	Encoded word for 2nd most probable 4-V
6	NHITS2	No. of smeared hits belonging to IPART2
7	IPART3	Encoded word for 3rd most probable 4-V
8	NHITS3	No. of smeared hits belonging to IPART3

The code used for the IPARTi words is (as for the HITL bank):

2 * particle_number + VECT_bank_number

Subroutine MCTR4V resides on 'F11LHO.JADEGS/L'.

C) 4-Vector History Traceback

The PATR Traceback described in section B allows one to trace the origin of a PATR track to the VECT banks. (Pointers inside VECT/1 allow traceback to VECT/0.) However, it is often vital to know the 4-vector history of a particle in the VECT/0 bank. This is now possible using the new 'PALL' bank.

Changes have been made in the JADE Tracking Program MCJADE in order to create the PALL bank from information stored in the input 4-vectors.
@
This information is only meaningful if set up correctly by the 4-vector event generator. At the time of writing (October 1983), only the LUND 4.3 and 5.1 generators maintained by Alfred Petersen are able to output the required data but other generators could be modified to conform with the scheme to be outlined below.

JADE Computer Note 10 describes the format of JADE 4-vector events

that are acceptable to the tracking program. This is the so-called CPROD format, named after the COMMON block used in the tracking program. (Actually the /CPROD/ common in the tracking program defines the 'shape' and maximum size of the 4-vector records; the events themselves are always smaller, with no trailing zeros. The tracking routine BRVECT unpacks this compact structure into the /CPROD/ common.) The CPROD format is split into 2 parts, the first part for 'produced' particles and the last part for 'final' particles that are stable or meta-stable. It is the 'final' particles from an event generator that are tracked by MCJADE. Up till now, the 'produced' particles have simply been thrown away by the tracking program but, for the purposes of what might be called '4-vector history traceback', this is no longer so.

The 'produced' particles list of the CPROD format will now be known as the 'all' particles list and expanded from 30 particles to 500. As its name suggests, this list now provides space for storing information about every particle produced in the event simulation. When the events are tracked, this information is copied into a PALL bank in the same way as the 'final' particle information is copied into the VECT/0 bank. Since there is an (almost) identical entry in PALL for every entry in VECT/0, it is possible to associate a particle in the VECT/0 bank with one in PALL. Since it is also part of the scheme that each PALL particle has a pointer to its parent, a complete traceback of the history of any VECT/0 particle is thus possible.
@

```
      COMMON / CPROD / NEV, BEAM, PT, PHI, THETA, IFLAVR,
     +                 NP, NC, NN, PP(4,500), XM(500),
     +                 JCH(500), JTP(500), JP(500,2),
     +
     +                 NF, NCF, NNF, PF(4,300), XMF(300),
     +                 ICF(300), ITF(300), PSTRT(3,300)
```

NP = total number of 'all' particles
NC = total number of 'all' charged particles
NN = total number of 'all' neutral particles

PP(i,k) = i'th component of the 4-vector for the k'th particle
XM(k) = mass in GeV of the k'th particle
JCH(k) = charge of the k'th particle
JTP(k) = type of the k'th particle (see below)
JP(k,1) = pointer to PARENT of the k'th particle in the 'all' list
JP(k,2) = order no. of parton parent for the k'th particle (see below)

Thus for each particle k, there is a so-called 9-vector of data incorporating the energy-momentum 4-vector.

Particle Types

Where defined, JADE types are used as in JADE Computer Note 10. Otherwise private types are used such as the LUND table. Although this is unsatisfactory in principle, no other solution yet exists. In practice this may not matter as particle masses are also available and the LUND table may well last for the lifetime of JADE at PETRA. (Currently the LUND 4.3 and 5.1 generators held by Alfred Petersen use (1000 + LUND_type) * ISIGN or (JADE_type) * ISIGN, whichever is
@
appropriate. ISIGN is +ve for particles and -ve for anti-particles. Since the type is signed, only the absolute values should be used when comparing entries in the VECT/0 and PALL banks.)

Parent Pointers

JADE Computer Note 10 says that JP(k,1) is a pointer to the daughter particle in the 'final' list. In this new scheme, it is now a pointer to the parent particle in the 'all' list. This follows the LUND simulation scheme and is more useful in practice anyway. It is a convention that the parent number of the 'first' particle(s) is zero. There could be more than one 'first' particle of course but what type this particle(s)

has is not fixed in this scheme. It could be an initial electron or positron (as in the LUND generator) but does not have to be.

Parton Parent (Order Number)

JP(k,2) used to be the decay multiplicity but has now been redefined to be the parton parent number. However note that this does not give the flavour of the parton. It can however be used to make a fast 'jet' assignment. The first parton is assigned number 1, the second (which may be a gluon) is assigned number 2 and so on. The flavour of an event (where meaningful) can be found in IFLAVR in /CPROD/ and in the 9th word of the PALL and VECT/0 banks. It is another convention that the parton number of the partons is -100 and the initial e+, e-, virtual photon (or virtual Z) have a 'parton number' of -1000. Radiated photons in the initial state have a 'parton number' of -2.
@

PALL Bank Structure

Name of bank : PALL (= All Particles)
No. of bank : 0
Length : L0 + L1 * NP words

Full-word	Contents	Comment
1	L0	Length of header = 9 words
2	L1	Number of words per particle = 9
3	IEVNT	Event number
4	NP	Total number of 'all' particles
5	NC	Number of 'all' charged particles
6	NN	Number of 'all' neutral particles
7	PHI	Phi of event axis (if relevant)
8	THETA	Cos Theta of event axis(" ")
9	IFLAVR	Flavour of event (" ")
L0 + 1	9VECT1	9-vector of 1st particle (9 words)
L0 + L1 + 1	9VECT2	9-vector of 2nd particle (9 words)
L0 + 2*L1 + 1	9VECT3	" " " 3rd " (" ")
.	.	. . . . .
.	.	. . . . .
L0 + L1*(NP-1)+1	9VECTn	9-vector of last particle (9 words)

where 9VECTi stands for the 9-vectors as explained above:
px, py, pz, E, m, charge, type, parent_number, parton_parent_number
@
Notes
---

a) The first 5 components of P9VECTi are REAL*4 while the other 4 are INTEGER*4. This is different from the output of MCHTRB.

b) If a quark or anti-quark appears in the chain, its electric charge value will probably be undefined.

c) The PALL bank should not be confused with the subroutine PALL in the BOS library.

d) For Multi-hadronic events, there has been an unofficial practice of storing the partons at the end of the VECT/0 bank with an illegal type code, e.g. -100,-101 etc. This is now obsolete with the new PALL bank but may still happen for the time being. However this practice may cease in the near future. Another unofficial feature which may persist is the allocation of a sign to the IFLAVR word in VECT/0 to indicate whether the quark or anti-quark is stored first in the bank. A negative value implies the anti-quark comes first.

e) Subroutine MCTRCB calls subroutine SCTR4V. Both are stored on the member MCTRCB on F11LHO.JADEGS/L.

I would like to acknowledge the contribution of Karl-Heinz Hellenbrand to the design and testing of the traceback scheme outlined in this note.

Copies of this JADE Computer Note can be obtained by the following NEWLIB command: SUB 'JADEPR.TEXT(JADECN69)'

▶ ausschließlich relevant für MCJADE (siehe MCVAL1)

1 = γ
2 = e
3 = μ
4 = π
5 = K
6 = N

JADE Computer Note 69

28.10.1983

Karlheinz Meier

Monte Carlo Simulation of Electromagnetic Showers in the Lead-Glass

The standard JADE LG-tracking does not properly simulate electromagnetic showers. In particular the longitudinal and tranversal energy spread is only poorly reproduced. One method to overcome these problems is the well known EGS-algorithm which however is to slow in generating large amounts of MC events as it is required for efficiency calculations.

The scheme being presented in this note is based on a parametrisation of electromagnetic showers in SF5 lead glass (E. Longo et. al.,Nucl. Instr. a. Meth.,128(79),283).

The parametrized energy density function ρ allows to calculate the energy fraction deposited in a single block by

$$F_i = \int_{Block} \rho \cdot dr \cdot d\Theta \cdot d\varphi$$

This integral has to be evaluated in detector coordinates which can only be done with the use of numerical methods. The processing time for an average multihadron event is therefore as large as 5 sec which is to much for an implementation into the standard JADE Monte Carlo chain.

It is however easy to run the new program instead of the standard LG tracking by changing the tracking job according to one of the two following methods.

- insert the source program as a macro

```
%MACRO 'F1IMEI.MCSHOWS(TRLG3)'
```

or

- include the compiled version

```
INCLUDE SYSLIB(TRLG3)
```

L. Becker, G. Heinzelmann 25.1.84

JADE COMPUTER NOTE 71

New dE/dx Calibration

Based on the z calibration P. Dittmann has started to develop a new calibration for the measurement of the energy loss dE/dx. His work has been continued and a final calibration is now available for the period 1979 - 1982.
The resulting overall rms resolution (compared with the old one) is:

	old	new
Bhabhas	9 %	6.5 %
Pions in multi-hadronic events 0.45 < p < 0.6 GeV	11 %	8.0 %

Here, only tracks with more than 36 hits and cos(theta) < 0.75 were taken into account.

For the data of 1983 a preliminary calibration is available.

The old program DEDXBN was replaced by a new one which performs:

a) Calculation of dE/dx and sigma(dE/dx)
b) Comparison with the theoretical value (J.A.J. Skard, K. Ambrus)

The program is on the general library F11LHO.JADEGL and is called by:

```
CALL   DEDXBN
```

The results are stored in the

```
COMMON /CWORK1/ IER,NTR,TRES(10,60).
```

```
IER             =  ERRORFLAG:
                   IER=1000  IF BANK POINTER = 0
                   IER=4000  IF # OF TRACKS .LE. 0
                                OR .GT. 60
NTR             =  # OF TRACKS

ITRES(1,ITR)    =  NHIT
TRES(2,ITR)     =  DEDX
TRES(3,ITR)     =  SIGMA(DEDX)
TRES(4,ITR)     =  CHISQ(ELECTRON)
TRES(5,ITR)     =  CHISQ(PION)
TRES(6,ITR)     =  CHISQ(KAON)
TRES(7,ITR)     =  CHISQ(PROTON)
ITRES(8,ITR)    =  JMIN, NUMBER FOR MINIMUM CHISQUARE
                   1 = P, 2 = K, 3 = PI, 4 = E, 0=NO DEDX
TRES(9,ITR)     =  MOMENTUM (GEV)
TRES(10,ITR)    =  MOMENTUM ERROR
```

The program DEDXBN has to be used in the SUPERVISOR. In this way the dE/dx calibration constants are given automatically by KLREAD from the general calibration files

```
F11LHO.AUPDAT1
```

or

```
F11LHO.BUPDAT0
F11LHO.BUPDAT1
```

The results of DEDXBN can be saved by creating a BOS bank 'DEDX' via:

```
IPATR = IDATA( IBLN('PATR') )
CALL DEDXBK(IPATR)
```

(The bank number is the same as for the 'PATR' bank.)

The 'DEDX' bank contains: