List of JADE-Computer-Notes

Date	Author	No.	Topic
		1	doesn't exist
14. 8.77	Mills	2	Graphic Facilities on the JADE NORD-10 Computer
15. 1.77	Bartel	3	Status of Monte-Carlo and Off-Line Programs
1. 12.77	Dittmann	4	Standards to JADE-Library programmers
30. 1.78	Heuer, Nozaki, Olsson, Steffen	5	Conventions of Jet Chamber Data Formats for Pattern Recognition and Related Programs
3. 8.78	O'Neill	6	Proposal for Management of Calibration Data
4. 9.78	Dittmann	7	IBM - NORD, Data transfer via magnetic tape
5. 8.78	Stork	8	The JADE BOS System
6.10.78	Yamada	9	Version No. Identifier for Lead Glass analysis Programs
9.11.78		10	Monte Carlo Formats
17.11.78	Steffen	11	Use of BOS-Banks Generating Subroutine Package in the First Stage Data Reduction Program
1. 2.79	Steffen	11a	Extension of Note No. 11
21.11.78	Steffen	12	Track Bank from Pattern Recognition Program last version 18.4.80 by G.F. Pearce
20.11.78	Steffen	13	A Simple Way to Analyse JADE Data on Tapes
29.11.78	Yamada	14 c	Lead Glass Cluster Bank Structure
23.2.79	Yamada	14a	
21. 9.79	Watanabe	14b	Analysis Program for Lead Glass (LG) Counters
31. 7.79	Dittmann, Yen	15	Data Acquisition System: Physical Record Format on Tape
26. 3.79	Hughes, Wriedt	16	Data Format of the Tagging Banks
1. 2.79	Steffen	17	Bank Created by the Z-Vertex Reconstruction (ZVERTF)
1. 2.79	Allison et al.	18	How to Change Existing Analysis Routines to become fully BOS Compatible
15. 2.79	Mills	19	LIFILE - a Program to Prepare File Statistics on the NORD
21. 2.79	Kawabata	20	Output Format of TOF Program
9. 3.79	"	20a	"
14. 1.82	Steffen	21	Hit Label Bank created by PATREC
1. 2.84	Allison et al.	22	Mu Software Information, Issue 4
9. 3.79	Bartel	23	IBM Data Banks
6. 6.79	Helm, Naroska	23a	IBM Trigger Banks
9. 8.79	Naroska	23b	Trigger Words
14. 3.79	Allison	24	Printing Half-Word Banks
12 8.81	Yamada	24a	Event TP-Banks

Date	Author	No.	Topic
20.10.81	Bowdery, McCann	52	How to use the new muon graphics commands and options
10.11.80	Steffen	53	How to use the JADE volumes on MSS
15. 5.82	Elsen	54	Detector Monte Carlo
Nov. 81	Barlow	55	The Muon Monte-Carlo at DESY (fehlt)
14. 1.82	Steffen	56	Subroutine MCTRCO
		57	omitted
12. 8.82	Steffen	58	New Conventions for Calibration Data
12. 8.82	Steffen	59	Run dependent Event Vertex on Calibration file
12. 8.82	Steffen	60	Refit Tracks with Run Vertex Constraint
18. 8.82	Steffen	61	Refit all Tracks of an Event with Run Vertex Constraint
12.10.82	Bartel/Becker	62	JADE REFORMAT PROGRAM
26. 5.83	Becker/Heinzelmann	63	z-Recalibration
5.12.83	Becker	63a	"
27. 7.83	Glasser	64	Help for JADE (on the Computer)
18. 8.83	Glasser	65	Subroutine GGFIT (EMAS, PROB)
24. 8.83	Hagemann Olsson Ramcke	66	Inner Detector Smearing and Trigger Simulation in JADE Monte Carlo Events + Addendum, + Supplement 2
August 83	Barlow	67	The JADE Muon Monte Carlo + Addendum
21. 9.83	Steffen	68	The JADE Calibration Scheme
28.10.83	Bowdery	69	Monte Carlo Traceback
15. 5.84	Meier	70	Monte Carlo Simulation of Electromagnetic Showers in the Lead Glass
25. 1.83	Becker Heinzelmann	71	New dE/dx Calibration
27.2.84	Bowdery	72	Monte Carlo Data Validation
1. 2.84	Bowdery Olsson	73	The JADE Supervisor Program: An Introduction for Newcomers
22. 3.84	Finch	74	Analysis Routines for Tagging System + Supplement, 16.5.85
16.5.84	Kuhlen	75	Routines for Converted Photons
11. 6.84	Meyer	76	A Collection of Programs used in the Analysis of Inclusive Production
28. 6.84	Krehbiel	77	Copying of Source Files and Data Files from the JADE-NORD10 to the IBM and vice-versa
7.11.84	Hellenbrand	78	Decays of $K^{\pm}$ and KØL in MC
17.12.84	Yamada, Bowdery Elsen	79	The JADE TP program
12.12.84	Bowdery, Yamada	80	Format of the Generation 8 TP banks
17.4.85	Olsson	81	LUMI and RANDOM events on Tape

August 12, 1987

Internal Note

A Short Guide to MC simulation of $\gamma\gamma$ processes in JADE

Version 0

J. OLSSON

Introduction

Monte Carlo simulations of $\gamma\gamma$ physics processes, i.e. of the reactions

$$e^+e^- \rightarrow e^+e^-X \tag{1}$$

and the reactions contained in (1)

$$\gamma\gamma \rightarrow X \tag{2}$$

where X is a hadronic system (leptonic systems, i.e. pure QED processes, are not considered in the following), are done for several reasons:

a) Artificial events of the particular reaction under study can be used to optimize analysis cuts.

b) The artificial events can be used to determine the trigger efficiency for the particular reaction. The trigger efficiency is an essential part of the overall detection efficiency. If the latter is known, the observed number of events will provide a measurement of the cross section of (1).

c) The simulation automatically provides the $\gamma\gamma$ luminosity in some form. This number is needed to turn the cross section measurement of (1) into a corresponding measurement of the cross section for (2). The latter can then, in the case of X being a single resonance, be related to the radiative width of X, $\Gamma_{X\gamma\gamma}$.

In the JADE software for Monte Carlo simulation of reaction (1), two major steps can be distinguished, namely

1) Generation of 4-vector events of reaction (1), including 4-vectors of the decay products of X.

2) Simulation of the response of the JADE detector, by following each of the generated 4-vectors of the event through the JADE detector simulation program. The latter produces events in the same format as the real data events, i.e. in BOS format. These so called Monte Carlo events can be analysed in the same way as real data events.

In the following, each of these major steps will be shortly described. Several good works in the literature will be often referenced and the reader is recommended to get familiar with them, for the deeper understanding.

1. G. Bonneau, M. Gourdin and F. Martin, Nucl.Phys.**B54**(1973)573.

2. V.M. Budnev et al., Phys. Reports**15**(1975)181.

3. J. Field, Nucl. Phys**B168**(1980)477, Erratum Nucl.Phys.**B176**(1980)545.

4. M. Poppe, Intl.J.Mod.Phys.**1**(1986)545, DESY 86–014

5. J.D. Jackson, Nuovo Cim.**34**(1964)1644.

Generation of 4–vector events

The QED expression describing the density of virtual photons in reaction (1) is fairly complicated, although well understood (Budnev, Bonneau). In the early days of experimental $\gamma\gamma$ physics, various approximate (and simpler) formulas were used for its numerical calculation, based on the so called Equivalent Photon Approximation, EPA (Budnev, Field). Nowadays, the complexity of the full formula offers no problems, thanks to modern computer technology, and all large experiments doing $\gamma\gamma$ physics have developed sophisticated computer programs to perform the calculation of this density and to integrate it with the produced hadronic system. In the case of JADE, this work was done mostly by S. Kawabata, an earlier member of the group. His programs are based on a program developed by J. Vermaseren for the pure QED process

$$e^+e^- \rightarrow e^+e^- \; e^+e^- \tag{3}$$

or its analogues with μ pair or τ pair production. The method used is *Importance Sampling*; for a description of this method see e.g. the original write–up by S. Kawabata (also S. Kawabata, Comp.Phys.Comm.**41**(1986)127), or Poppe. The heart piece of these programs is the routine VEGAS, originally developed by P. Lepage. It was modified by Kawabata and renamed to BASIS (GRUND in some versions) or SPRING (QUELLE in some versions).

The numerical integration of (1) proceeds in at least 5 dimensions: the energies and the θ angles of the scattered electrons and also the relative ϕ angle of the electrons. If the mass of the produced system (usually called $W_{\gamma\gamma}$) is not fixed, then also this will be integrated over and the process is 6–dimensional. Even for modern computers this is CPU–time demanding. Kawabata therefore arranged the integration in such a way that the time consuming part i.e. the integration itself, is done only once. The tossing of 4-vector events, which is the final goal of the work, can then be done with great speed and repeated many times with new sets of random numbers. The following program descriptions therefore always come in pairs: one part is doing the integration with the integration results being written on a small file; the second part is a program which reads this integration file and uses its results for the 4–vector event generation.

Below several program examples are described in detail (they can be found on the source library F11OLS.JADEMC, which contains a collection of JCL– members for various $\gamma\gamma$ Monte Carlo jobs):

#SBRW70,#SBRW71: The first program performs the integration of reaction (1) with X being a broad resonance, e.g. the $f_2(1270)$. The second program performs the corresponding event generation.

#SBRW70 contains at the end the following data cards, for steering:

```
//*
//*  E          WS         WIDTH      W-MIN      W-MAX    ITT ICOND IFLAG ISPIN
//*(F10.4)    (F10.4)    (F10.4)    (F10.4)    (F10.4) (I5) (I5) (I5)   (I5)
//*  FMAS1      FMAS2      FERMI
//*(F10.4)    (F10.4)    (F10.4)
//*
//GO.SYSIN    DD *
COMPUTE THE SINGLE BREIT-WIGNER SYS PROD. IN GG PROCESS
   17.3000    1.2740     0.1780     0.2700     3.0000   50    3    0    2
    0.1350    0.1350     1.0000
```

The input variables are now explained in detail:

E: The Beam Energy, in GeV.

WS: The Mass of the produced resonance X, in GeV. For a broad resonance, it should be the nominal mass.

WIDTH: The Width of the produced resonance X, in GeV. For a broad resonance, it should be the nominal width.

W-MIN: Lower end of the mass range of X. WMAX is the upper end of the range. In principal, the mass range is $0 - 2\cdot E_{\mathrm{beam}}$, neglecting electron masses. However, in most cases only a small part of this range is really needed, and it is very ineffective to integrate over the complete range. Besides, this may lead to numerical problems. In the example, the $f_2(1270)$ is integrated between the lower limit of two π° masses (its decay products) and the upper limit of 3 GeV, where the resonance is effectively zero again.

ITT: Number of iterations. The precision of the integration, normally 0.1%, is given by the internal variable BCC. If this precision is not reached, the program will end the integration after ITT iterations.

ICOND: This flag is used for steering cuts on the kinematic variables in the integration. This is sensible in some cases, e.g. if only tagged events are wanted (or only untagged). The actual cuts have to be coded for. In the given example, ICOND is a dummy. Note that cuts affecting the integration can only be done on the primary produced particles, i.e. the outgoing scattered electrons and the produced system X. Decay products of X appear only in the second step, when the 4-vector events are produced (see below).

IFLAG: This flag is used for telling the program to start from scratch or to continue with an already started integration. The integration program is clever enough to recognize if too little time is left to perform one more iteration and, in case of such impending time limit, writes the intermediate results onto the output result file. The user is informed by a print out, saying e.g. that integration ended with flag value 2. It is then possible to resubmit the job, with IFLAG set to 2, and continue the integration. In this case, care has to be taken to define the output file correctly, since it is now assumed already existing and used first as input. When the integration has reached the level of IFLAG=3, the output file will be organized such that it can be used as input for the following 4-vector generation program. Thus it is not necessary to continue the integration to the ITT limit or to the specified precision, since it may happen that the program actually is unable, for numerical reasons, to obtain the wanted precision.

ISPIN: The spin of the produced resonance, in the case of $f_2(1270)$ it is 2.

FMAS1: Together with FMAS2 the masses of the decay products of the resonance, in this case both are π°. 2–body decay is assumed, for the parametrization of the resonance as a Breit–Wigner curve.

FERMI: A parameter used in the parametrization of the mass–dependent width of the resonance. More about this below.

The integrated function is given in the program by the FUNCTION F(X), where X is an array of random numbers supplied by the calling routine BASIS. F(X) calls LMFUNC, which provides the $\gamma\gamma$ luminosity, essentially formula (28) and (29d) in Bonneau. The subroutine BREIT gives the $\sigma_{\gamma\gamma}$, which is just a Breit–Wigner formula, (3.24) in Budnev. Later in F(X) some additional Q^2 dependence is added by ρ–poles. This can be replaced by the Q^2 dependence of GVDM, by use of the FUNCTION FGVDM; presently this is commented out.

Before proceeding with the description of the actual 4-vector event generation, some comments on the code of the subroutine BREIT should be given. The formula (3.24) in Budnev is

$$\sigma_{\gamma\gamma} = 8\pi(2J+1)\cdot\Gamma_{X\gamma\gamma}\cdot\frac{\Gamma}{(W^2-m_X^2)^2+\Gamma^2 m_X^2} \tag{4}$$

Here J is the spin and Γ the total width of the resonance. The total width is normally parametrized as a mass-dependent function for broad resonances (see Jackson):

$$\Gamma = \Gamma_0\cdot\left(\frac{p}{p_0}\right)^{(2J+1)} \tag{5}$$

Γ_0 is the nominal width of the resonance, p_0 is the momentum of the decay products (2–body decay!) at the nominal mass, m_X, and p is the corresponding momentum at the actual mass W. The parametrization (5) leads to an asymmetric shape, the more so with higher spin J, and normally an additional damping factor is needed to supress the Breit–Wigner curve at higher masses. This extra damping is usually parametrized as an Angular Momentum Barrier Penetration Factor (see Blatt&Weisskopf, Theoretical Nuclear Physics), sometimes also called Decay Form–factor. It has the form

$$\begin{gathered}\text{Spin 0:}\ \ D(x)=1\\ \text{Spin 1:}\ \ D(x)=1+x^2\\ \text{Spin 2:}\ \ D(x)=9+3\cdot x^2+x^4\end{gathered} \tag{6}$$

with $x=p\cdot r$. p is the momentum and r is a length parameter, usually taken as 1 Fermi. x_0 is the corresponding quantity at the nominal mass. The total width then takes the form

$$\Gamma = \Gamma_0\cdot\left(\frac{p}{p_0}\right)^{(2J+1)}\cdot\frac{D(x_0)}{D(x)} \tag{7}$$

The above parametrizations are empirical and the theoretical foundations can be discussed. For very broad resonances, like the ρ, still more empirical shaping functions can be added (see Jackson). Moreover, the whole formula (4) is approximative, since it is written for the limit of real photons ($Q^2=0$); More accurate descriptions can be found in e.g. Poppe. See also below the discussion on very narrow resonances.

The second JCL–member, #SBRW71, is very similar to #SBRW70. The steering data cards at the end are the following:

```
//*
//*  E          WS         WIDTH      W-MIN      W-MAX   ICOND IXTYP ISPIN
//*(F10.4)    (F10.4)    (F10.4)    (F10.4)    (F10.4)   (I5) (I5)   (I5)
//*  FMAS1      FMAS2      FERMI      NR.EV
//*(F10.4)    (F10.4)    (F10.4)    (I10)
//*
//GO.SYSIN    DD *
COMPUTE THE SINGLE BREIT-WIGNER SYS PROD. IN GG PROCESS
   17.3000    1.2740     0.1780     0.2700     3.0000     0    20     2
    0.1350    0.1350     1.0000     999999
```

Most of these are the same as for #SBRW70 and it is the responsibility of the user to take care that they are really the same as used in the integration. In principle this could have been arranged in a safer way, but for historical reasons it was not. Two variables are new:

IXTYP: Every resonance that is generated has its type number, in the case of $f_2(1270)$ it is 20. The number is used to steer the selection of decay routines, to be described below.

NR.EV: The number of 4-vector events one wants to generate. Normally set to a large number, since the generation of events will be stopped by TIME LIMIT or disc overflow.

There are two or more steering variables to be set, but for historical reasons they are set directly in the FORTRAN code of the MAIN program, which is the first in the JCL member:

LIMRND: Random number seed. A new seed will generate a new set of 4–vector events. *This number should be an odd number.*

KTYPE: This is an array, sitting in COMMON /CDKTYP/. Each element corresponds to a particle type, IXTYP. The value of a given element decides the decay mode of the corresponding resonance. In the example, $f_2(1270)$ decays into $\pi^\circ\pi^\circ$, and each π° decays into 2γ. More about this below in the description of SAGE.

The function F(X) and its called subroutines are identical to those in #SBRW70. BASIS is however now replaced by the slightly different version SPRING, which produces 4-vector events of weight 1. An event is first only produced as the final state e^+e^- X, and in a second stage X is decayed into stable particles by a call to the subroutine DECAYS. The latter routine, which calls a number of other decay routines, is initialized by the previous call to subroutine MTSTRT. The subroutine ROTAT3 provides a random rotation in ϕ, since only the relative ϕ between the electrons is considered in the integration and generation, and the extra overall rotation in the LAB–system has to be added. WRIT4V finally writes the final 4–vector event out in the special format, the so called CPROD–format (named after the COMMON/CPROD/) adopted in the JADE set–up.

The decay routines use the program package SAGE, which produces decay distributions in LIPS (Lorentz Invariant Phase Space). SAGE was written by Jerry Friedman and a long write–up is available. All the SAGE routines, as well as the special JADE routines for decays, which call SAGE routines, are located on the source library F11OLS.JADEMC2 (member MTSTRT); the load versions are found on F11OLS.JADELD. The original SAGE routines have been copied from the original library F11BAR.EVENTGEN.S. It is worth remarking that this set of SAGE routines is very old, it was used already in 1974-75 and the import route from SLAC to DESY is not quite clear. In the winter 1986-87 the most actual version used presently at SLAC (where the author of SAGE also resides) was imported and checked very carefully against the old version used in JADE, and no disagreement was found, apart from esthetic changes in the code.

No detailed description of SAGE will be given here, the interested reader is referred to the write–up and program examples. It is however probably useful to give here a summary of the JADE specific details:

COMMON/CPROD/

The 4-vector events are kept in a special format, for historical reasons not a BOS–bank. It is given by the following MACRO, sitting on the source library F22ELS.JMC.S, which is the standard JADE Monte Carlo library:

```
C ------------------ MACRO CPROD -----------------
C                                        4-VECTOR FORMAT FOR JADE
C
      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)
C
C ---------------END MACRO CPROD -----------------
```

The variables are as follows:

NEV: Generated event nr, starting with 0. After detector simulation, this is the number in HEAD bank word 11.

BEAM: The beam energy.

PT: This and the following PHI,THETA,IFLAVR have no meaning in $\gamma\gamma$ physics and are not set.

NP: Nr of produced particles, before subroutine DECAYS has been called. In our case this is 3, i.e. e^+e^- X. Of course, NP = NC + NN.

NC: Nr of charged produced particles. In our case 2, i.e. e^+e^-.

NN: Nr of neutral produced particles. In our case 1, i.e. X.

PP: 4–vector array of the produced particles 1–NP.

XM: Masses of the produced particles 1–NP.

JCH: Masses of the produced particles 1–NP.

JTP: Types of the produced particles 1–NP.

JP: Pointers of the produced particles 1–NP, referring to decay daughters in the final particles.

NF: Nr of final particles, after subroutine DECAYS has been called. In our case this is 6, i.e. $e^+e^-$and 4 γ. Note that stable produced particles (e^+e^-) are just transferred to the final particles. Again, NF = NCF + NNF.

NCF: Nr of final charged particles, in our case just the 2 electrons.

NNF: Nr of final neutral particles, in our case 4 photons.

PF: Analogue of PP for the final particles.

XMF: Analogue of XM for the final particles.

ICF: Analogue of JCH for the final particles.

ITF: Analogue of JTP for the final particles.

PSTRT: x-y-z of the start point for a final particle, e.g. this could be the decay point in space of a K^0.

PARTICLE TYPES

These are partly (types 1–12) given already in Jade Computer Note 10, but have been further extended for $\gamma\gamma$ physics use. The original set have been used for general Monte Carlo work in JADE and other special areas may have obtained their own extensions of this numbering scheme; there is no overall standardization of this in JADE. Moreover, other numbering schemes are also used, e.g. the one in the LUND Monte Carlo programs. The numbering used in $\gamma\gamma$ physics can be found in the Commented Header of the main routine DECAYS, which at present is the following:

```
C       LAST MODIFICATION:  10.11.1985     J.OLSSON
C
C     LIST OF PARTICLE TYPES:
C       0   NEUTRINO                 20     F0(1270)
C       1   PHOTON                   21     A2(1310)
C       2   ELECTRON                 22     E(1420)
C       3   MUON                     23     U(2981)     (ETAC)
C       4   PION                     24     F1(1515)
C       5   K                        25     S*(980)
C       6   NUKLEON                  26     JOTA(1440)
C       7   PHI                      27     DELTA(981)
C       8   ETA                      28     X(2100)
C       9   ETAPRIME                 29     TAULEPTON (1784)
C      10   OMEGA                    30     PI+PI- SYSTEM (BORN)
C      11   KSTAR(890)               31     ZERVAS HEAVY PSEUDOSCALAR
C      12   RHO                      32     BARYON-ANTIBARYON SYSTEM
C      13   CHI(3.55)                33     BARYON-ANTIBARYON-PIO SYSTEM
C      14   J/PSI
C      15   A3(1680)
C      16..19   NOT USED             50     KO  (USES KTYPE(5) FOR DECAY!)
```

Each of the particles (i.e. the unstable ones) has its own decay routine, with various options for decay modes. The most important are listed below:

pion: Subroutine PI0DK. Only π° decays in the 4-vector event generation, the charged π's are considered stable and may decay in the detector simulation. For the π°, 2 decay modes are simulated:

`MODE 0:`	`GAMMA GAMMA`	`MODE 1:`	`GAMMA E+E-`

ϕ: Subroutine PHIDK. The following decay modes are simulated:

`MODE 0:`	`KL, KS`	`MODE 1:`	`ETA GAMMA`
`MODE 2:`	`K+, K-`	`MODE 3:`	`PI+ PI- PI0`

$\eta(549)$: Subroutine etaDK. The following decay modes are available:

`MODE 0:`	`GAMMA GAMMA`	`MODE 1:`	`PI0 PI0 PI0`
`MODE 2:`	`PI+ PI- PI0`	`MODE 3:`	`PI+ PI- GAMMA`
`MODE 4:`	`PI0 GAMMA GAMMA`		

$\eta'(958)$: Subroutine ETPRDK. The following decay modes are available:

`MODE 0:`	`GAMMA GAMMA`	`MODE 1:`	`RHO GAMMA`
`MODE 2:`	`PI+ PI- ETA`	`MODE 3:`	`PI0 PI0 ETA`
`MODE 4:`	`OMEGA GAMMA`	`MODE 5:`	`PI+ PI- GAMMA`

ω: Subroutine OMEGDK. The following decay modes are available:

`MODE 0:`	`PI0 GAMMA`	`MODE 1:`	`PI+ PI- PI0`
`MODE 2:`	`PI+ PI-`		

$K^\star(892)$: Subroutine KSTDK decays the neutral K*(892) with decay modes:

`MODE 0:`	`PI0 K(SHORT)`	`MODE 1:`	`PI0 K(LONG)`
`MODE 2:`	`K+- PI-+`		

The charged K*(892) is decayed with subroutine CKSTDK and modes:

`MODE 0:`	`PI0 K+-`	`MODE 1:`	`PI+- K(LONG)`
`MODE 2:`	`PI+- K(SHORT)`		

ρ: Subroutine RHODK with the single mode $\pi^+\pi^-$, or $\pi^\pm\pi^0$, depending on charge.

f$_2$(1270): Subroutine F0DK with the following decay modes:

`MODE 0:`	`PI0 PI0`	`MODE 1:`	`PI+ PI-`
`MODE 2:`	`PI+ PI- PI+ PI-`	`MODE 3:`	`K+ K-`
`MODE 4:`	`PI+ PI- ETA`	`MODE 5:`	`PI+ PI- GAMMA`
`MODE 6:`	`PI0 PI0 GAMMA`	`MODE 5:`	`K0(SHORT) K0(SHORT)`
`MODE 8:`	`PI0 PI0 PI+ PI-`		

a$_2$(1320): Subroutine A2DK with the following decay modes:

`MODE 0:`	`RHO+- PI-+`	`MODE 1:`	`ETA PI0`
`MODE 2:`	`OMEGA PI0`	`MODE 3:`	`K+ K- (OR K0S K0S)`
`MODE 4:`	`PI+ PI- PI0`		

E(1420): Subroutine EDECAY with the following decay modes:

`MODE 0:`	`K0(SHORT) K+- PI-+`	`MODE 1:`	`K+ K- PI0`
`MODE 2:`	`2/3 ETA PI+ PI-, 1/3 ETA PI0 PI0`		

$\eta_c(2981)$: Subroutine UDECAY with the following decay modes:

`MODE 0:`	`K0(SHORT) K+- PI-+`	`MODE 1:`	`K+ K- PI0`
`MODE 2:`	`ETA PI+ PI-`	`MODE 3:`	`ETA PI0 PI0`
`MODE 4:`	`PI+ PI- PI+ PI-`	`MODE 5:`	`PI+ PI0 PI- PI0`

```
MODE 6:  PI+ PI- GAMMA          MODE 7:    ETAPRIME PI+ PI-
MODE 8:  K+ K- K+ K-            MODE 9:    K+ K- KOS KOS
```

$f_2'(1525)$: Subroutine F1DK with the following decay modes:

```
MODE 0:  PIO PIO                MODE 1:    PI+ PI-
MODE 2:  K+ K-                  MODE 3:    KOS KOS
MODE 4:  PI+ PI- ETA
```

$S^*(975)$: Subroutine STARDK with the following decay modes:

```
MODE 0:  PIO PIO                MODE 1:    PI+ PI-
```

$\eta(1440)$: Subroutine JOTADK with the following decay modes:

```
MODE 0:  DELTA PIO
```

$\delta(981)$: Subroutine DELTDK with the following decay modes:

```
MODE 0:  K+ K-                  MODE 1:    KOS KOS
MODE 2:  ETA PIO
```

K_S^0: Subroutine K0DK with the following decay modes:

```
MODE 0:  PIO PIO                MODE 1:    PI+ PI-
```

Simulation of the JADE Detector

The 4–vector events written on the output file by #SBRW71in the program example described above, serve as input to the next major step, namely the detector simulation, or in jargon "tracking". Contrary to the above, this program package is well standardised in JADE and also well described. The following Jade Computer Notes deal with tracking: 54,67,69,70,72,86,87 and 87 addendum.
Unfortunately, some are a bit outdated and none is really written for the newcomer. A simple outline of the tracking program is therefore given below. The example is taken from the authors specially adapted program, but the main flow is the same as in all Monte Carlo simulations in JADE:

```
      COMMON / BCS / IW(40000)
C
      COMMON / CIEVS  / KIEV,IEVMIN,IEVMAX
C
C          IEVMIN : FIRST EVENT TO BE READ FROM INPUT FILE
C          IEVMAX : LAST  EVENT TO BE READ FROM INPUT FILE
C
C                  ==== > DEFAULT IEVMIN = 1
C                                 IEVMAX = 9999999
C
      INTEGER*2 HDATE
      COMMON / TODAY / HDATE(6)
C
      COMMON/CFLAG/LFLAG(10)
      LOGICAL * 1 LFLAG
C
C          LFLAG(1) = SMEAR GAMMA AND ELECTRON ENERGIES
C          LFLAG(2) = GAMMA CONVERSION IN OUTER TANK AND COIL (TRKGAM)
C          LFLAG(3) = ABSORPTION LOSSES
```

```
C          LFLAG(4) = 3 DIM SHOWER PROFILE FIT TO EGS CODE
C          LFLAG(5) = .TRUE.   -->  WITH VERTEX CHAMBER TRACKING
C                   = .FALSE.  -->  WITHOUT VERTEX CHAMBER TRACKING
C                                   BUT OLD BEAM PIPE GEOMETRY AND
C                                 BEAM PIPE COUNTERS (BEFORE MAI 84)
C          LFLAG(6) = 3 DIM TOKYO SHOWER PROGRAM, JADE NOTE 20A
C
C         ===> DEFAULT .TRUE.,.FALSE.,.TRUE.,.FALSE.,.FALSE.,.FALSE.
C
C   NOTE: WITH THE USE OF TOKYO SHOWER PROGRAM, LFLAG(1) AND LFLAG(3)
C         ARE NOT USED. HOWEVER, THEY SHOULD BE TRUE TO ENSURE THAT THE
C         ALGN BANK IS PROPERLY MARKED FOR LATER ENERGY CORRECTION..
C         LFLAG(2) IS TRULY REDUNDANT
C         LFLAG(4) SHOULD BE FALSE, ALTHOUGH TRLGL IS ONLY CALLED FOR
C                  HADRONS AND MUONS
      LOGICAL*1 TEXT
      DIMENSION TEXT(72)
C
      LFLAG(1) = .TRUE.
      LFLAG(2) = .FALSE.
      LFLAG(3) = .TRUE.
      LFLAG(4) = .FALSE.
      LFLAG(5) = .TRUE.
      LFLAG(6) = .TRUE.
C
      WRITE(6,521) (LFLAG(I),I=1,6)
521   FORMAT(' LFLAG:  ',6I3)
C
C  INIT OF TODAY
C
      HDATE(1) = 1
      HDATE(2) = 1
      HDATE(3) = 1
      HDATE(4) = 15
      HDATE(5) = 6
      HDATE(6) = 1986
C
      WRITE(6,2201) HDATE
2201  FORMAT('  DATE: ',6I6)
C                                Initialize BOS
      CALL BINT( 40000,15000,500, 0 )
C
C   READ IN:    EVENT READING LIMITS    ievmin ievmax
C
7608  FORMAT(72A1)
7610  FORMAT(7I10)
7508  FORMAT(' ',72A1)
7510  FORMAT(' ',7I10)
```

```
      READ 7608, TEXT
      WRITE(6,7508) TEXT
      READ 7610, IEVMIN,IEVMAX
      WRITE(6,7510) IEVMIN,IEVMAX
C                               main tracking routine
      CALL MCJADE( 0,  5 )
C                               BOS statistics
      CALL BSTA
      CALL PALL
      STOP
      END
```

For the beginner, nothing in this MAIN program really needs explanation. HDATE is an array, which setting determines the "date" of the simulated detector; this is important, when one goes into details of triggering efficiencies or thresholds of SF5/SF6 Lead glass, etc.. Changing of the logical flags LFLAG needs expertise and the default is recommended as above. An important feature is the ability to stear the start and end event. Since parts of the program are rather slow, and the input file with the 4-vector events contains many thousand events, one may need many shorter jobs, or several longer ones, to work through a larger sample. It is convenient to be able to submit many jobs at the same time, each being set up for a particular part of the input file. If the output file is a disc file, which is the normal case, one has to remember that the ready tracked events are full BOS-format events which take much bigger space on disc than the input 4–vector events.

The main difference between the above MAIN program and the standard one (which is situated on F22ELS.JMC.S/L) is the use of LFLAG(6), to invoke the use of the full lead glass shower program from TOKYO. The corresponding version of the main routine, MCJADE, is situated on F11OLS.JADE56.S/L, where also all other routines involved in this shower program package are situated. They will be fully described in a forthcoming Jade Computer Note.

The library F11OLS.ETAP.S contains 2 program examples, with which tracking can be started: #LINKMC and JBTJ20D. The former creates a linked module of the tracking program, which is then used by JBTJ20D. The many entries in the INCLUDE statements are historical and will be made standard as soon as the mentioned Jade Computer Note is made available. Note that in the tracking job, JBTJ20D, the input file is by default on unit 3, the output on unit 2; this is just the opposite of the normal SUPERV standard. It is a nice example of the rich individualism in HEP collaborations.

The output events are in BOS format, they are by all means real JADE events, although they are in fact raw data events, and need all the standard analysis treatments of pattern recognition and cluster analysis. This can be performed by any standard job, e.g. #TG04. They can also be directly looked at in the JADEZ graphics program. In the latter case, note that the command TRUE will display the original 4-vectors, which are kept in a special BOS bank with name VECT.

```
  **********************************************************************
  *                                                                    *
  *  I M P O R T A N T   N O T E:         STANDARD LIBRARY CHANGES     *
  *                                                                    *
  **********************************************************************
                                         C.Bowdery, J.Olsson   19.10.83
  (Member JBIMPNOT on JADEPR.TEXT)


--->>>  As discussed in the JADE Computer meeting 17/10  (and also in
        JADE Computer Note 66), the many changes to the tracking and
smearing routines will now be implemented as standard. These updated
(and in some cases totally new) routines were up to now residing on the
libraries  F22ELS.JMC.S1 and F22ELS.JMC.L1. They have been subjected
to tests and should now be released. We propose that this change take
place on

           THURSDAY   20 OCT. 1983,  at about 09.00 - 10.00
           ************************************************

The change will consist of renaming the following libraries:

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

In addition, some routines on F11LHO.JADEGS and F11LHO.JADEGL have to
be updated:

EVREAD and EVWRIT:  The common /CADMIN/, which is Block Data set in
                    EVREAD, is extended. A bug in EVWRIT (see JCN 66)
                    has been removed.
RDMTCO:  This is a copy of the smearing routines on JMC.S and JMC.L;
         It is also kept on JADEGS and JADEGL to save the general user
         the trouble of linking JMC.L in standard reading jobs.

JBOVER:  This JCL member does the linking of JADE Graphics Modules. The
         overlay structure has changed in the "RD" branch, some of the
         previous routines do not exist any more and there are others
         which are new. There are also some minor changes in other
         branches. Users with private versions of JBOVER should update
         them. Note that the standard overlay is now kept in a MACRO,
         in such a way that private additions are possible.

SUPERV:  The general Block Data in this routine has been changed to
         conform to the corrected errors in BLDAT (see JCN 66).

 ----------------------------------------

There are a number of changes to routines on the present JMC.S1 and
JMC.L1, which are not mentioned in JADE Computer Note 66. Many of them
will be explained in the forthcoming J.C. Note 69 by C. Bowdery, in
particular those changes which concern the traceback possibilities now
offered. Of immediate importance are the following:

 1.   The subroutine MUCONW is no longer kept on JMC.S and JMC.L. It is
      called by WRTMCB to write out the second "Muon Constants Record".
      Any tracking job must therefore link the library F22ALL.JADEMUL,
      which is the proper library for muon routines.

 2.   COMMON /CPROD/ has been extended in dimensions for produced and
      final particles and now has the appearance shown below. Hopefully
      this will be big enough to accomodate any exotic physics until a
      more satisfactory solution for 4-vector formats has been found (as
      discussed in recent JADE Computer Meetings). The reason for the NP
      dimension being bigger than the NF one is explained in Computer
      Note 69. For tracking jobs that read 4-vector events as input data
      this change of dimensions does not matter. For "merged" jobs, with
      generation and tracking in the same step, adjustment of private
      versions of /CPROD/ may be necessary.
```

```
      COMMON/CPROD/ NEV,BEAM,PT,IPHI,ITHETA,IFLAVR,
     *         NP,NC,NN,PP(4,500),XM(500),JCH(500),JTP(500),JP(500,2),
     *         NF,NCF,NNF,IPF(4,300),IXMF(300),ICF(300),ITF(300),
     *                IPSTRT(3,300)


 3.   In the near future other changes will be implemented in the
      tracking program, as were also discussed in the above mentioned
      meeting. This would e.g. be the instalment of the simulation
      routines for hadronic interactions in leadglass (J.Kanzaki); the
      instalment could not yet be done, due to an unreadable tape.

--->>>  On the present version of JMC.S1 is also kept a copy of the
      main program for the TP-step,  @TPMAIN and the associated JCL-
      member #TPMAINC. The logical error in this routine, which was
      described in JCN 66, has been repaired. The change consists in
      calling the subroutines EVREAD and EVWRIT instead of BREAD and
      BWRITE. Another error has also been repaired: the subroutines
      INPATR and INPATC, which are pattern recognition initialization
      routines, are now called in correct sequence. The same changes
      will be made on these members on the TP-libraries, F22YAM.TPSOURCE
      and F22YAM.TPLOAD. Users with private versions of @TPMAIN are
      strongly urged to update them, since the error in INPATC,INPATR
      calling sequence will cause reduced resolution of charged track
      momentum.
      Furthermore, the subr. TPINIT will be updated. It contains the
      same Block Data as SUPERV and BLDAT. As inspection shows, this
      third version is seriously outdated, with magnetic field and
      Lorentz angle opposite to the tracking program. However, this
      does not matter for MC events, since these values are updated
      from the MTCO event in the beginning of the data set (see JCN 66).
      For real data, these values are updated by KALIBR.


--->>>  Some recent development of software for the Tagging detectors
     will also be made standard at the above mentioned time. This
     affects primarily the graphics program, which now has several
     display options for the 1983 version of the tagging system. For
     this, some new Commons has been installed as Macros on the JADE
     Macro library (F11GOD.PATRECSR) :


C---------   MACRO CGEO2COM       GEOMETRY OF FORWARD DETECTOR    --------
C
C---   LEAD GLASS BLOCKS
C      FENDC:   WIDTH (AND HEIGHT) OF BLOCKS
C      XYHOL1:  DISTANCE FROM BEAM CENTRE TO EDGE OF FIRST
C               HORIZONTAL BLOCK
C      XYHOL2:  DISTANCE FROM BEAM CENTRE TO EDGE OF FIRST
C               VERTICAL BLOCK
C      BLDPFW:  DEPTH OF BLOCKS
C      ZMINBL:  DISTANCE FROM INTERACTION POINT TO FRONT SURFACE
C               OF LEAD GLASS BLOCKS (-Z-DIRECTION, PLUTO/CELLO)
C      ZPLUBL:  DISTANCE FROM INTERACTION POINT TO FRONT SURFACE
C               OF LEAD GLASS BLOCKS (+Z-DIRECTION, MARK J)
C---   LUMONITORS
C      XSC:     LENGTH OF LONG EDGE (1: A-COUNTER, 2: B-COUNTER)
C      YSC:     LENGTH OF SHORT EDGE
C      RSC:     DISTANCE FROM BEAM CENTRE TO CENTRE OF INNERMOST LONG
C               EDGE ON A-COUNTER
C      ZMISC:   DISTANCE FROM INTERACTION POINT TO FRONT SURFACE OF
C               LUMONITOR (-Z-DIRECTION, TOWARDS PLUTO/CELLO)
C      ZPLSC:   DISTANCE FROM INTERACTION POINT TO FRONT SURFACE OF
C               LUMONITOR (+Z-DIRECTION, TOWARDS MARK J)
C      DZSC:    THICKNESS OF LUMONITORS
C
C---   DRIFT CHAMBERS
C
C   CHX(I,J): XPOSITION OF WIRE 0 IN CHAMBER I (1-3) IN PLANE J (1-4)
C   CHY AND CHZ ANALOGOUS.      WLEN IS SENSITIVE LENGTH OF WIRES
```

```
C  PITCH IS WIRE DISTANCE IN XY-PLANE, WZDIS DISTANCE IN Z BETWEEN
C  THE ODD AND EVEN WIRE PLANES
C  WIRE 0 IS CLOSEST TO THE BEAM LINE
C  PRESENT DATA SETTING OF WIRES BASED ON "EDUCATED GUESSES"
C
C     ALL VALUES IN MM       BLOCK DATA SETTING IN SUBR. SUPERV
C
C     DATA FENDC/81./,XYHOL1,XYHOL2/141.,151.5/,BLDPFW/400./,
C    1 ZMINBL,ZPLUBL /-5250.,5250./,
C    2 XSC /230.,150./, YSC /150.,70./, RSC /152.48,192.48/,
C    3 ZMISC/-4235.,-4135./, ZPLSC/4235.,4135./, DZSC/6./,
C    4 CHX/140.,-400.,-140.,140.,-400.,-140.,
C    5 140.,-400.,-140.,140.,-400.,-140./,
C    6 CHY /-400.,140.,-400.,-400.,140.,-400.,
C    7 -400.,140.,-400.,-400.,140.,-400./,
C    8 CHZ /-4770.,-4720.,-4770.,-4020.,-3970.,-4020.,
C    9 4020.,3970.,4020.,4770.,4720.,4770./,
C    A WLEN/800./, PITCH /25./, WZDIS/10./
C
C------ END MACRO CGEO2COM

C-----------------------------------------------------------------------
C              MACRO CGEO2 .... JADE FORWARD DETECTOR GEOMETRY
C-----------------------------------------------------------------------
      COMMON /CGEO2/ FENDC,XYHOL1,XYHOL2,BLDPFW,ZMINBL,ZPLUBL
     1,XSC(2),YSC(2),RSC(2),ZMISC(2),ZPLSC(2),DZSC
     2,CHX(3,4),CHY(3,4),CHZ(3,4),WLEN,PITCH,WZDIS
C------------------------ END OF MACRO CGEO2 --------------------------

C-----------------------------------------------------------------------
C      MACRO CGEO3COM..  COMMENTS TO CONTENT OF COMMON /CGEO3/
C-----------------------------------------------------------------------
C  ZMINM2 AND ZPLUM2 ARE DISTANCES FROM INTERACTION POINT TO FRONT SUR-
C  FACE OF -Z AND +Z FORWARD LEADGLASS, 1981-82 VERSION. THE BLOCK
C  DIMENSIONS ARE THE SAME AS FOR 1979-80, IN CGEO2
C
C  1983- ... TAGGING APPARATUS: LEAD SCINTILLATER UNITS
C  NRPBSC ARE NUMBER OF MODULES ON ONE SIDE IN Z
C  PBSCR1-4 ARE THE RADII FROM BEAMLINE TO SEGMENT JOINTPOINTS
C  PBSCZ1-4 GIVE THE Z COORDINATES, FROM MINUS Z TO PLUS Z
C--
c     DATA ZMINM2,ZPLUM2 /-2950.,2950./
c     DATA NRPBSC/8/
c     DATA PBSCR /104.02,120.02,152.02,264.02/
c     DATA PBSCZ /-3470.,-2950.,2950.,3470./
C------------------------ END OF MACRO CGEO3COM -----------------------

C-----------------------------------------------------------------------
C      MACRO CGEO3 .... JADE FORWARD DETECTOR GEOMETRY, 1981-3 VERSION
C-----------------------------------------------------------------------
      COMMON /CGEO3/ ZPLUM2,ZMINM2,NRPBSC,PBSCR(4),PBSCZ(4)
C------------------------ END OF MACRO CGEO3 --------------------------
```

These two geometry Macros are Block Data set in subr. SUPERV on F11LHO.JADEGS, JADEGL as well as in the subr. BLDAT on F22ELS.JMC.S and JMC.L. Still another Block Data setting is present in the subr. TPINIT on F22YAM.TPSOURCE and TPLOAD.

At the same time two new standard libraries for JADE are introduced. F11LHO.TAGGING.S and TAGGING.L contain the routines for tagging analysis. They will be described in a forthcoming note by A. Finch. Presently one can distinguish between the following steps:

```
Calibration:  Complete routines exist for the 1981-82 apparatus.
***********   For the 1983 version, a preliminary calibration exists.
              For the 1979-80 version the complete calibration is
              being installed (up to now it existed on the libraries
              F22HOW.JADEGAMS,JADEGAML and the file F22HOW.CALIBRAC).
```

OBS!
The JADE graphics program now uses routines on this new library for the calculation of tagging energy and block display. This means

that the load library F11LHO.TAGGING.L must be used in the linking, see member JBOVER on F11LHO.JADEGS.

```
Cluster analysis:  Here routines exist fcr all versions but work on
*****************  them is still required since it is not yet
                   possible for data from different periods to be
                   treated in the same job.
```

--->>> A new routine is installed on F11LHO.JADEGS, JADEGL: MCTR4V. This routine is now called by SUPERV after the Pattern Recognition (level 5) and carries out the last stage of the Monte Carlo traceback scheme. For more information, see Jade Computer Note 69, by C.Bowdery.

--->>> Another facility for detailed study of resolutions is offered in the smearing routines. By setting a flag, it is possible to save the tracked JETC bank (with nr 8), which contains the fine resolution used in the tracking. The smeared JETC bank will then be a separate bank, instead of overwriting the original unsmeared bank. The numbers will then be:

```
              smeared:   JETC 8   (as previously)
            unsmeared:   JETC 9
```

The switch for this is situated in COMMON /CADMIN/:

```
    COMMON /CADMIN/ IEVTP,NRREAD,NRWRIT,NRERR,IDUM(4),IJETCI
```

IJETCI is Block Data set to 0. If set to nonzero, the above operation will take place.

In the graphics program, the command   JETC N   allows the user to switch between the different JETC banks (the default is the one with the lowest number). Note that the unsmeared bank gives sensible coordinate values only in the various R-FI views. The z-amplitudes have to be modified according to the algorithm described in JCN 66, in order to show sensible z values. This algorithm will be implemented also for the unsmeared bank in the near future.

GRAPHICS FACILITIES ON THE JADE NORD-10 COMPUTER

JADE COMPUTER NOTE : 2

H.E.Mills

14 August 1978

INTRODUCTION

Several graphics packages are implemented on the NORD-10. These include the Tektronix PLOT-10 AG2 and TCS packages and the CERN histogram packages. They have been modified in order to remove bugs and improve performance particularly concerning speed.

The most recent modifications have been to enable the library to be used by Real Time programs as well as background (TSS) programs and to include recent modifications to the CERN histogram packages.

A brief description of each package is given here, together with changes made.

TERMINAL CONTROL SYSTEM (TCS 3.3)

This package is described in the PLOT-10 manual. It provides facilities to draw lines, define windows and use the cursor on Tektronix storage screens. Many changes have been made to this package in order to reduce the cpu requirements.

JI4014 & JI4010

These subroutines have been written so that the user does not have to worry about the correct initialisation calls for TCS. At the start of a graphics program either JI4014 or JI4010 should be called depending on the type of Tektronix device in use. The screen is cleared and the cursor is moved to the upper left corner of the screen. Alpha numeric mode is selected. The subroutines have one parameter which is the logical unit number for graphical output. For TSS programs this should be 1. For RT programs the number should be the device number for the Tektronix screen e.g. 9 or 34.

e.g. CALL JI4014(1)

FINITT

Since the logical unit number for graphical output of histograms via the routine ZHPL1 is defined by the call to JI4014, there is no need to call ZGLUN as described on page 21 of the CERN ZHIST document. The second parameter to ZHPL1 is now a dummy.
To plot a histogram on the Tektronix you should use...

```
        CALL JI4014(1)
        CALL ZGINT
           .
           .
           .
        CALL ZHPL1(NHIST,0,IE)
```

SPECIAL NOTES

The AG2 and TCS packages have been compiled in DIRECT-ADDRESSED-CALLS (D-A-C) mode. This results in significantly faster code at the expense of a slightly increased memory requirement The present Fortran compiler fails to compile part of the ZHIST and ZH2DH packages correctly in D-A-C mode so they are compiled in the normal mode. This means that programs which use AG2 or TCS may be compiled in D-A-C mode but that ZHIST and ZH2DH programs should not.

ACCESSING THE GRAPHICS PACKAGE

The compiled versions of the packages are stored on (SYS)GRAPH-LIB:BRF. The NRL load sequence could be :-

```
        @NRL
        $LOAD <APROG>,GRAPH,FTNLIB
           .
           .
```

Only the routines required by the program will be loaded.

Further developments to the library will be noted in the log book and the GRAPH-LIB folder.

J. Olsson

W. Bartel

JADE - Computer Note 3

15.11.1977

STATUS OF MONTE-CARLO and OFF-LINE PROGRAMS

I.1 Monte-Carlo Events

Monte-Carlo generated test events with pions only are available for :

a) jet model

b) phase space model

c) two photon events.

All these events are calculated for beam energies of 15 GeV, about 4000 events of each type are available.

The data format has been described in a note which was distributed on June 6, 1976.

Remark : The data format will be changed at some time, because the present format has turned out to be inconvenient.

I.2 Monte-Carlo tracking routines

Tracking routines for charged particles are available to track through the central detector, central lead glass array and the muon filter. A tracking routine for photons has also been implemented.

I.2.1 Tracking through the drift chambers

Charged particles are tracked through the central detector drift chambers;

a) the wire coordinates are approximately correct;

b) the magnetic field is 5 kGauss;

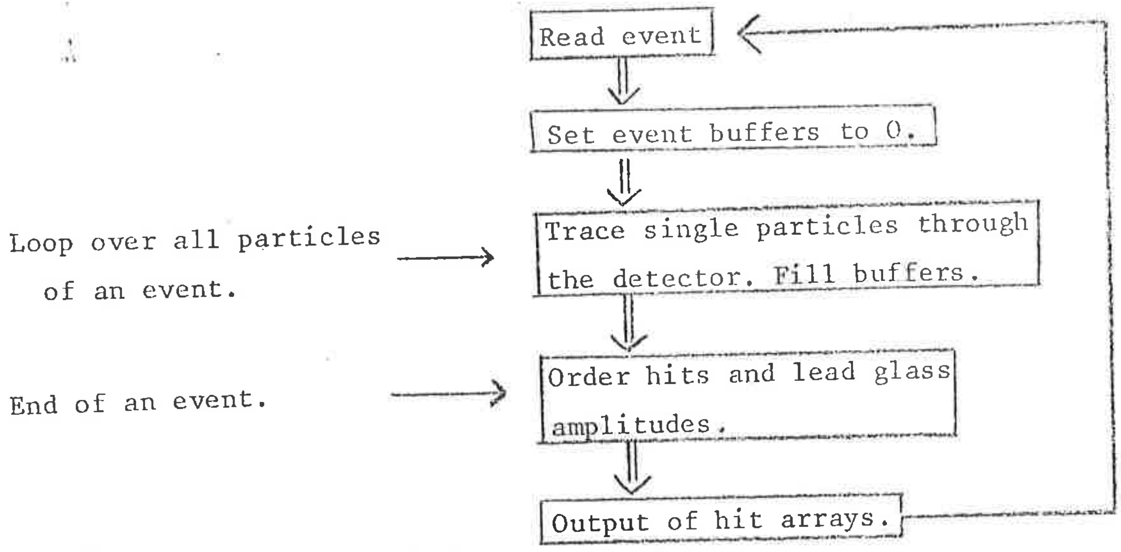
c) a tilt angle of the wire acceptance of 7.5° is assumed;

d) one clock is assumed for a group of 8 wires;

e) no smearing and multiple scattering is assumed.

I.3 Structure of the Monte-Carlo program.

The Monte-Carlo program is structured as follows :


Read event
Set event buffers to 0.
Loop over all particles of an event.
Trace single particles through the detector. Fill buffers.
End of an event.
Order hits and lead glass amplitudes.
Output of hit arrays.

The following output formats are available :

a) Wire arrays :

Pointers (4)	Wire array WARR (max. 6000)

All quantities are INTEGER*2.

Pointer (1)	-	points to the last word in the wire array.
Pointer (2)	-	points to the first word of the first ring (usually 1).
Pointer (3)	-	points to the first word in the second ring.
Pointer (4)	-	points to the first word in the third ring.

WARR (1)	-	number of hits on a wire.
WARR (2)	-	wire number
WARR (3)	-	left amplitude
WARR (4)	-	right amplitude
WARR (5)	-	drift time.

Items 3 - 5 are repeated as many times as there are hits on the specific wire.

These data are also available on direct access files for testing purposes.

d) Muon filter array :

At the moment this contains

Pointers (13)	DATA ARRAY (max. 1680)

Pointers are all I * 4

DATA ARRAY is I * 2

DATA ARRAY = HMUDAT (1680)

Pointer (1)	IWP	points to last word of DATA ARRAY
Pointer (2) . . . (7)	IWDFC1 (1) . . . (6)	Pointer to the first word in faces 1 .... 6 of the muon filter (see JADE-Note No. 9) on IMUDAT. = 0 if no words written for the face.
Pointer (8) . . . (13) .	NFCWDS (1) . . . (6)	No. of (I * 2) words on IMUDAT for faces 1 ..... 6 of the muon filter.

IMUDAT (1)	Chamber No.)
(2)	Drift time (clock pulses)) for firs hit
(3)	Longitudinal time (clock pulses))
. . . .	
(3I - 2)	
(3I - 1)	Ditto for the I^{th} hit
(3I)	
. . . .	

d) Work on pattern recognition.

e) Multiple Coulomb scattering & energy loss and crude interactions will be inserted before the generated data are used in order to test out muon-filter pattern recognition strategies.

Remark :

The interactive removal and addition of chamber hits on the IPS screen with subsequent reanalysis of the tracks and display will be available soon. This scheme will provide a powerful tool to study and develop pattern recognition programs because the results are immediately available and it is not necessary to go through the procedure of submitting special jobs for each change.

Olsson

JADE-Computer-Note 4

Topic: Programming conventions

P. Dittmann

December 1st, 1977

Standards to JADE-Library programmers

Programs which are to be included into the JADE program library must be understandable to other people to assure compatability. The following recommendations are thought to help in optimizing program coding and program execution.

A. General

- Program language must be IBM-compatible FORTRAN
- Each subprogram should contain a comment header which specifies author and date of last update, and also what it does.
- Programs should be tested with the IBM compiler optimization OPT = 2.
- IMPLICIT INTEGER * 2 (H)
  The letter H shall be used as leading letter for I * 2 variables. Other leading letter should be used according to FORTRAN conventions (bad example: REAL MASS).
- The BLANK COMMON should not be used, and be reserved for the HBOOK histogram package.
- Memory space optimization is more important than execution time optimization (which, however, is also important).
- Frequent operations inside loops should be optimized - some examples:

```
           I                              II
X = B(I,1)              . . .      X = B(1,I)
X = A(I+1)              . . .      X = A(1+I)
A(J+1) = A(J+1)+1. . . .         { K = J+1
IF(A(1).LE.0.)GØTØ  .            { A(K) = A(K)+1.
X = C * 0.5                .       IF(A(1).LT.1.)GØTØ
                        . . .      X = C/2.
```

Common //

R.D. Heuer
T. Nosaki
J. Olsson
P. Steffen

JADE Computer - Note No. 5 30.1.1978

Conventions of Jet Chamber Data Formats for Pattern Recognition and Related Programs.

A. Nomenclature

LAYER :	all wires in the central detector that have the same R (distance from the origin).
RING :	16 layers form a ring.

Ring 1 = layer 1 16
Ring 2 = layer 17 32
Ring 3 = layer 33 48

CELL :	The rings are divided in 24 (ring 1,2) or 48 cells. Each cell contains 16 wires of approximately the same ϕ value.
SEGMENT :	One cell of ring 1 and of ring 2 and two cells of ring 3 form a segment.

.../.

C. PACKAGES :

- display on IPS - terminal
- event classification
- cell classification, <h>, <n>, ...
- Z-vertex calculation
- simple track finding within adjacant cells
- track finding for 'complicated cells'
- simple backtracing of tracks through ring 2 and 1
- backtracing of tracks in 'complicated cells'
- track finding for 'complicated segments' using all 3 rings
- simple track fitting
- refined track fitting
- elimination of uncorrelated hits
- others

...../.

E. WORKING AREA

All data necessary for the correspondence of subroutines within one program package shall be stored in a working common :

```
COMMON / CWORK / LWORK , HWORK (10 000)
LWORK = 10 000 = length of working area (INTEGER * 2)
```

The same storage can be used by different packages. The information should be expected to be lost in case of a second call of the same package.

F. INTERMEDIATE RESULTS

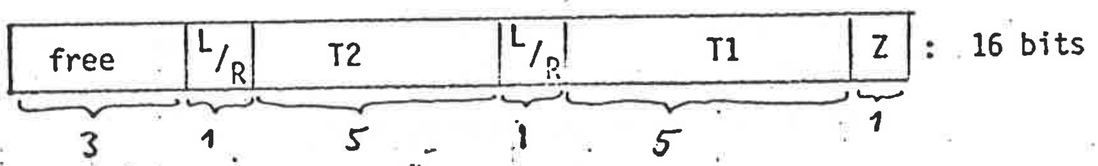
All intermediate results and data necessary for the correspondence between the 'SUPERVISOR' and the different 'PACKAGES' will be stored in a different common. The details are not yet fixed.

G. OUTPUT DATA

The output data consist of the following information :

1. event results : class, No. of tracks, etc.
   details are not yet fixed

2. HLBHIT (I), I = 1, NHIT : Label array (1 label/hit)



..../.

L. O'Neill

3.8.78

JADE - Computer Note No. 6

Proposal for Management of Calibration Data

This note essentially summarizes the discussion at the JADE offline software meeting of July 25, 1978. "Calibration data" as used in this note will refer, as usual, to conversion factors between raw data and physically interesting quantities, for example between ADC channel numbers and energies in MeV, but it will also refer more generally to any data describing the state of the JADE - detector or of PETRA at a given time, for example to survey data on the positions of detector elements or to data on the currents and polarization of the stored beams.

It is clear that a very large number of data describing the state of the experimental apparatus as a function of time and covering all times at which data have been taken must be available to the various analysis programs, since we must assume that events from the entire course of the experiment will continue to be reanalyzed, by more refined programs and for different reasons, long after they have first been processed by the offline software. The management of the necessary calibration data will be complicated by at least two distinct effects : First, some of the physical quantities of interest, photomultiplier gains, ADC pedestals, drift velocities and so on, will actually change with time, second, our estimate of what some of these quantities were at a given time will improve, or at least change, at later times, for example when tracks have been used to estimate wire positions or to check the z coordinate measurements by the inner drift chamber. Thus calibration data will have to be updated not only in the sense of being added to, but also in the sense of being retroactively revised.

The following table shows the various kinds of calibration data for which need has been anticipated. Also shown are the rates at which significant changes are likely to occur, according to current best estimates, and when the data will be available with various degrees of accuracy. Unavoidably the information is still inexact and the definitions vague in many cases. The

the events currently passing through the first data reduction phase were recorded. Old data would be erased from this file and new data added to it at frequent intervals. These data would be used by the program making the first data reduction to write epilogues to the events it accepted. The epilogues would contain reduced data, such as lead-glass energies in MeV, based upon the available calibration data. The old data taken off the cyclic buffer would be written for permanent storage on tape. The rationale for bringing in preliminary calibration data at the data reduction stage will become clear below.

The second disk file would describe the long term time development of each calibration parameter for the course of the whole experiment. The data on this file would be in the form of corrections, as they later become known, to the calibration data used at the first data reduction stage. The time development file could be organized as follows : The variation of each parameter could be fitted with a quadratic polynomial within each of a sufficiently large number of time bins. The sizes of the bins could be tailored to the actual rate of variation of each parameter, and bin bounderies could be adjusted to match abrupt discontinuities. Except where discontinuities occur, the quadratic fits in adjacent bins and their first derivatives could be made to match at the bin bounderies (spline fit). For each parameter and each time bin, four constants would then have to be stored on disk : the three constants specifying the quadratic polynomial, and one giving the time at which the bin in question ends. The entire history of calibration parameter number one would be written first on the disk, then the entire history of parameter two, etc. with time changing in the "inner loop" and parameter number in the "outer loop". A program requiring new calibration data would have to read the entire file, and rewind it if new data were required more than once per job.

This organization has the clear disadvantage of requiring a great deal of input/output. It also requires that when new time bins are added the entire file must be rewritten. This however must be done in any case when calibration data are retroactively revised. The enormous advantage of the proposed organization is that it requires writing to disk only as much information as is really needed. Parameters which are stable can be described in one time bin. In the alternative approach, that of making the parameter number the "inner loop" variable, time bin size would have to be set by the most rapidly varying parameter and there would be no way at all of matching abrupt discontinuities to time bin bounderies parameter

by parameter. If the most unstable parameter could be described by three constants in each of 100 time bins (probably a very optimistic hypothesis) 40 000 parameters would require 40 000 x 300 = 12 million constants, or 24 million bytes, or 1850 tracks. This would be an enormous, probably prohibitively large file, and it would be mainly occupied with repeated data on stable parameters.

Even with the suggested organization the time development file may tend to become very large, depending on how many of the calibration parameters vary with time and on how rapidly they vary. It may be extremely useful, in keeping this file to manageable size, if most of the time variation of most of the calibration parameters can already be factored out on the data tapes. Many parameters, e.g. leadglass photomultiplier gains, may be known as well or nearly as well as they will ever be known as soon as the standard calibration runs are taken. On the other hand it is certainly not possible to have smooth fits to the current calibration data ready for the dump job at the IBM. A logical time at which to bring in current calibration data may be when the first data reduction is done, when obvious background events are rejected. This first reduction will be done by batch jobs running asynchronously with, and some time behind, data acquisition. These circumstances make it clear why the procedure described above for bringing in preliminary calibration data at the first reduction phase is proposed. It also becomes clear, in view of the proposed spline fit scheme for the long term time development file, that the cyclic buffer file must already have smooth fits to the time variation of the calibration parameters for the interval at interest, and not just the most recent set of values. Otherwise the corrected pulse heights etc. written into the tape epilogues will have a "step function" time dependence and the long term time development file will have to have a new time bin for each step. On the other hand no great harm is done if a few of the parameters used in writing the epilogues are very bad, since the development file can correct just these. The corrections on this latter file could be additive rather than multiplicative in case, for example, a few gains were multiplied by zero due to dead pulser channels. Alternatively a program could easily be written to keep unreasonable values from being written to the cyclic buffer file and to replace them with default values.

The long term time development file should probably be split into several disk data sets, each corresponding to a major block of running time. A new time development data set could be started after long shutdowns, JADE Computer Note 7

P. Dittmann

4.9.78

IBM <==> NORD

Data transfer via magnetic tape

Two programs have been written at the NORD to transfer data between the two machines:

IBM-DATA transfers binary data (i.e. events)

IBM-CHAR converts character data (i.e. program code).

The tape units at the IBM and at the NORD are compatible (9 tracks, 1600 bpi (DEN=3)). Also both machines have 8bit-bytes. The problems left are:

- Tape lables. They are not supported at the NORD.
  However, if the programs detect an IBM-Label (starting with 'VOL1' in EBCDIC) a skip to the next EOF is issued.
- EBCDIC vs. ASCII code. The conversion is done using the JADE-LIB routine YASEB. Only the lower 6 bits are used, so parity bits don't play any role. Character data on tape are always read or written in EBCDIC-code, on all other devices at the NORD in ASCII-code.
- Record formats. The record length on tape is assumed to be constant and less than 6400 bytes. This is achieved with the IBM record formats RECFM=F and RECFM=VBS. Details are given in the appendix.
- Multi-file data. Many files on one tape is a problem at the IBM. Therefore tapes are assumed to have only one file. There is a possibility to write more than one NORD-SINTRAN file to the tape, but there will be no EOF's written onto tape in between.
- Floating point numbers. There was no attempt to convert 'REAL's at the NORD. A conversion routine exists at the IBM.

The programs are written in FORTRAN. If somebody wants to write his own tape I/O-routines he may easily consult the source listing.

2. From NORD to IBM

   The example shows how to transfer program coding written at the NORD to NEWLIB at the IBM.

a. At the NORD:

   Load an unlabelled tape, otherwise you run into problems since there will be no trailing labels written.

```
@(J-P)IBM-CHAR
INPUT FILE: file-name
OUTPUT FILE: M-T-1
) see above
END OF FILE. COPY ANOTHER FILE: YES or RETURN
```

   There will be no EOF between the files on the tape.

b. At the IBM:

```
// EXEC NEWLIB, PS='sourcelib', TO=DISK
// SYSINØ DD *
./ ADD name,
// SYSIN DD UNIT=TAPE, DSN=something, VOL=SER=volume#,
// DCB = (DEN=3, RECFM=F, BLKSIZE=80), LABEL=(,NL)
```

B. IBM - DATA Program description

1. From IBM to NORD

   The example shows how to transfer Monte-Carlo events from the IBM to the NORD.

a. At the IBM

```
// EXEC DUPDAT
// PRINT SYSOUT=A,
// INPUT DSN=F11BAR.NEWT, DISP=OLD
// OUTPUT UNIT=TAPE, DSN=anything, VOL=SER=volume#,
// DCB=DEN=3 [, LABEL=(,NL)]
```

   Here it is assumed that the INPUT tape is written with RECFM=VBS.

b. At the NORD (for more details see example A.1.b)

   Mount tape and press ONLINE

- 4 -

Appendix: IBM Record formats F and VBS

Record format F:

FORMAT CONTROL

The following discussion provides information on records written under control of a FORMAT statement.

UNBLOCKED RECORDS: For fixed-length and undefined records, the record length and buffer length are specified in the BLKSIZE subparameter. For variable-length records, the record length is specified in the LRECL subparameter; the buffer length in the BLKSIZE subparameter. The information coded in a FORMAT statement indicates the FORTRAN record length (in bytes).

F ed-Length Records: For unblocked fixed-ngth records written under FORMAT con- l, the FORTRAN record length must not exceed BLKSIZE (see Figure 32).

Example: Assume BLKSIZE=44

```
10  FORMAT(F10.5,I6,2F12.5,'SUMS')
    WRITE(20,10)AB,NA,AC,AD
```

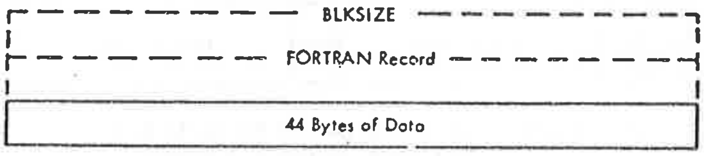


Figure 32. FORTRAN Record (FORMAT Control) Fixed-Length Specification

If the FORTRAN record length is less than BLKSIZE, the record is padded with blanks to fill the remainder of the buffer (see Figure 33). The entire buffer is written.

Example: Assume BLKSIZE=56

```
5  FORMAT(F10.5,I6,F12.5,'TOTAL')
   WRITE(15,5)BC,NB,BD
```

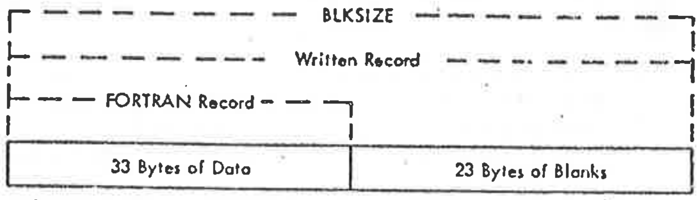


Figure 33. FORTRAN Record (FORMAT Control) Fixed-Length Specification and FORTRAN Record Length Less Than BLKSIZE

Record format VBS:

UNFORMATTED CONTROL

Only variable-length records can be written without format control, i.e., the RECFM subparameter must be VBS. (If nothing is specified, VBS is assumed.)

Records written with no FORMAT control have the following properties:

- The length of the logical record is controlled by the type and number of variables in the input/output list of its associated READ or WRITE statement.
- A logical record can be physically recorded on an external medium as one or more record segments. Not all segments of a logical record must fit into the same physical record (block).
- Two quantities control the manner in which records are placed on an external medium: the block size (as specified by the BLKSIZE parameter), and the logical record (as defined by the length of the I/O list). BLKSIZE is specified as part of the DCB parameter of the data definition (DD) statement. If not specified, FORTRAN provides default values.

Each block begins with a 4-byte block descriptor word (BDW); each segment begins with a 4-byte segment descriptor word (SDW). The SDWs and BDWs are provided by the system. Each buffer begins with a 4-byte block descriptor word (BDW). The SDWs and BDWs are provided by the system.

JADE Computer Note No. 8

Topic: The JADE BOS System

Derek Stork

5th August 1978

Following the decision to adopt the BOS (Bank Organisation System) method of storing output data on tape and disc (see JADE Notes 9(a) and 24), a version of the BOS program has been prepared for the JADE experiment and is now implemented on the three computers. (DESY IBM 370/168; RL IBM 360/195; NORD 10).

This note first describes the philosophy followed in implementation. Details of the FORTRAN calls available are then given and some hints on optimisation set out. The details of the LOAD modules used and how to link with them are finally given.

At all stages more detailed discussion is left to the BOS reference work:

Internal Report
DESY F14-77/01
August 1977
'BOS - Bank Organisation System - Dynamic
Storage Organisation with FORTRAN', by V. Blobel

We will refer to this as 'the BOS report'.

Philosophy

The following guidelines have been followed in adapting the BOS for JADE:-

1) All routines involving IBM FORTRAN syntax in their calls have been omitted and the existing alternative standard - FORTRAN - callable routines have been used instead. This means that the same calls are available on the IBM + NORD machines. Program differences are therefore transparent to the user.

2) The 'low priority' applications subroutines, which involve mainly histogramming and counting statistics, have been omitted entirely. (see section 13 of the BOS report). This was done to save core and because perfectly adequate histogram routines are available and widely-used already by members of the collaboration.

3) Apart from the above items a policy of 'minimum interference' with the source subroutines has been followed.

Routine Call	Purpose	Notes
CALL BDAR (NAME*,N$^+$,INDA$^+$,NLIM*)	Locate all banks with name NAME INDA contains the N output indices (N.LE.NLIM)	NAME must be DOUBLE INTEGER for the NORD 10
CALL BDEF (N*,TEXT*)	Set up a list of N bank names in array 'TEXT'	TEXT must be DOUBLE INTEGER for NORD 10. see BOS Report section 12.
CALL BDLM	Delete a pre-defined set of banks	Banks defined in BMLT are deleted.
CALL BDLS (NA*,NR*)	Delete bank name NA number NR	NA is DOUBLE INTEGER for NORD 10.
CALL BDMP	Print a dump	
CALL BGAR (IGA$^+$)	Perform garbage collection	IGA = 0 No garbage collection done IGA = 1 Garbage collection done
CALL BGAC (IGA$^+$,NW*)	Perform garbage collection if there is no room to add a bank of length NW to the store.	See above.
CALL BINT (NSPACE*,NREC*,NDMP*,NADD*)	Initialise the BOS system.	NSPACE as before NREC = Max No. of words in a bank NDMP = No. of words printed in a dump NADD = 0 Must be your first call to BOS.

Routine Call	Purpose	Notes
CALL CCHL $(NW^{*},IERR^{+})$	Change length of LAST CREATED BANK to NW words.	IERR = 2 if not enough space. See BOS report.
CALL CCRE $(IND^{+},NA^{*},NR^{*},NW^{*},IERR^{+})$	Create bank name NA, number NR, length NW.	IERR is error flag = 1 bank already exists = 2 not enough space NA must be DOUBLE INTEGER for NORD 10
CALL CLOC (IND^{+},NA^{*},NR^{*})	Locate bank name NA, number NR.	Test IND = 0 to see if bank exists NA must be DOUBLE INTEGER for NORD 10.
CALL CMVE $(IND^{+*},IERR^{+})$	Move bank at index IND to end of storage. IND is then returned as new position.	IERR = 2 if not enough space.
CALL CNXT (IND^{+*}) CALL CPOS (NA^{*})	Pair of routines to locate all banks of a given name NA.	NA must be DOUBLE INTEGER for the NORD 10.
CALL CREAD $(IUN^{*},IERR^{+})$	Read in data to IW from unit IUN.	IERR = 1 read error occurred. IERR = 2 end of record hit.
ILOC = IBLN (NA^{*})	IW (ILOC) is set to the index of the first bank with name NA.	NA must be DOUBLE INTERGER on the NORD. IW (ILOC) contains 0 if the bank does not exist.

In addition calls to subroutines BINP, BOUTP and BREADC are available on the IBM machines only. Their use is not recommended

Linking the BOS module with your program

1. NORD

Use the following command (once the NORD relocating loader NRL has beeen invoked).

```
LOAD <YOURPROG>, JADEBOS,FTNLIB.
```

This loads the required parts of BRF file (SYSTEM)JADEBOS-780721:BRF.

This BOS NORD file is compiled in LIBRARY MODE. This means that only the parts of the library actually called by <YOURPROG> will be loaded. In this way memory is economised in the NORD.

2. IBM 370/168 DESY

The BOS exists as two NEWLIB libraries

F11HUG.DS.JADEBOS.S	Fortran source
F11HUG.DS.JADEBOS.L	Load library

To link with the load library the following JCL statement is required

```
//LKED.SYSLIB  DD  DSN = F11HUG.DS.JADEBOS.L,DISP=SHR
```

The NEWLIB structure ensures once again that only those parts of the BOS library invoked by your program will be loaded.

3. IBM 360/195 RUTHERFORD

The BOS exists at Rutherford as an 'Automatic Call' library

USER.XM65.JADEBOS

To link this load library with your program the following JCL statement is required

a) If you are only linking with the BOS library and no others (apart from the standard libraries) you may use the symbolic parameters in the FORTRAN H procedure.
eg:

```
// EXEC FHCLG, <some parameters> , SYSLIB = 'USER.XM65.JADEBOS'
```

[illegible]

JADE - Computer Note No. 9

S. Yamada

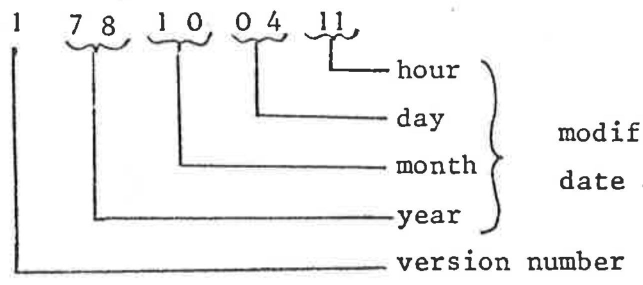
6. 10.1978

Version No. identifier for lead glass analysis programs

The first version of the lead glass cluster finding and processing program is now available for practical use. It has been tested with Monte Carlo simulated data. But work on its improvement is still going on and new versions of the subroutines are expected. When we analyse real data to produce reduced or TP tapes in the future, it may be necessary to identify the version number of each subroutine. A common block

```
. COMMON / CLGVRN / NVRSN (20)
```

is used to store the version numbers of the currently used subroutines. As a subroutine is called, it files its version number code into the corresponding NVRSN. The code is a 9-digit number as shown below.



At the end of a job one can list the version numbers by calling LGVRNL or copying the common block data into his output. An example of the LGVRNL is shown adjacent to this paper.

```
VERSION CODES OF LG-ANAL PROGRAMS USED IN THE JOB
  LADPUS/         0  LGADCD/ 178100509  LGAVRP/         0  LGAVRZ/ 278100421  LGBLCK/ 178090112
  LGCCTL/ 178092322  LGCHGN/         0  LGCLUS/ 278100608  LGCLP3/ 278100422  LGCLPC/         0
 (SPARE)/         0 (SPARE)/         0 (SPARE)/         0  LGNBLS/ 178100509  LGSHCN/         0
  LGSORT/ 178100509  LGSRTH/ 178100509   SHMAX/ 178100520 (SPARE)/         0 (SPARE)/         0
```

[illegible]

JADE - Computer Note No.10 9.11.1978

F11BAR. Sage PS
" PL

F11BAR. EV~~BA~~TGEN

Monte Carlo Formats

I. Four Vector Generation

Library : F11BAR. EVENTGEN.S and F11BAR. EVENTGEN.L

1. Tape format

Monte-Carlo tapes containing four vectors for various types of event classes are written in the following format :

```
NR, BEAM, DUMMY(4),
NP, NC, NN, ((PP(I4,N), I4 = 1, 4), XM(N), ICH(N), ITP(N).
(IP (N,I2), I2 = 1,2), N = 1, NP),
NF, NCF, NNF, ((PF(I4,N2), I4 = 1,4), XMF (N2), ICF (N2), ITF (N2),
(PSTRT (I3,N2), I3 = 1,3), N2 = 1, NF)
```

NR	=	Event No.
BEAM	=	Beam energy in GeV
DUMMY	=	Not yet specified
NP	=	Total number of primary particles
NC	=	Number of charged primary particles
NN	=	Number of neutral primary particles
PP(4,30)	=	four vectors of primary particles
XM(30)	=	Mass of primary particles (in GeV)
ICH(30)	=	Charge of primary particles
ITP(30)	=	Type of primary particles
IP(30,2)	=	Pointer array to decay products IP(N,1) = points to first decay product in PF IP(N,2) = number of decay products
NF	=	total number of final state particles
NCF	=	number of charged final state particles
NNF	=	number of neutral final state particles
PF(4,60)	=	four vectors of final state particles
XMF(60)	=	mass of final state particles

ICF(60)	=	charge of final state particles
ITF(60)	=	type of final state particles
PSTRT(3,60)	=	x, y, z - coordinates of the origin of final state particles (in mm)

The particle types are defined as follows :

JETSET 6.3 : [ ]

Type		Particle			
1		Photon	S		
2		Electron	S		
3		Muon	S		
4		Pion	S		
5		K	S	50	K^0_S
6		Nukleon	S		
7	[35]	Phi			
8	[24]	Eta			
9	[25]	Etaprime			
10	[34]	Omega			
11	[28⁺,29⁰]	K^*(890)	~~K^{*0}~~		
12	[27⁺,33⁰]	Rho	~~K^{*+}~~		
~~13~~		~~X(2.8)~~			
13	[36]	ψ	[30⁰,31⁺]		
14	[ ]	ψ'			
15	[26]	η_c			
16	[20⁰,21⁺]	D			
17	[30⁰,31⁺]	D^*			
18	[22]	F			
19	[32]	~~F*~~ [illegible]			
20		[illegible]			
21		A_2			

2. Programs are available to generate various types of events.

a) Jet events

H.G. Sander's coding of Feynman and Field is used to produce the primary particles of a jet.
SAGE phase space routines are used to decay them.

TAPE :	F11BAR. JETAT 30 contains 8000 jet events at 30 GeV.
DISK :	the first 200 events are available on F11BAR. JETAD 30

b) Phase space

Pion only phase space events generated by SAGE with a Poisson multiplicity distribution

TAPE : F11BAR. POISA 30
5000 events at 30 GeV

c) Beam gas

Use H.G. Sander's coding of FOWL generation

TAPE : F11BAR. BMGS A 30
10.000 beam gas events at 15 GeV beam energy.

d) Fast routines (not worth-while writing tapes) are available for generating QED events (ee, $\mu\mu$, $\gamma\gamma$) and a fair number of two body final states, e.g. $\omega\ \pi^o$, $\gamma\ \eta$, $\gamma\ \eta'$, $\phi\ \eta$, $K\ K^*$, $\pi^+\pi^-$, etc.

II. Tracking

Library : F11BAR. JADE. SOURCE and F11BAR. JADE. LOAD

The particles stored on the four vector tapes are traced through the detector and corresponding output tapes are written.

These tapes however do not have the final event format.
The ϕ-resolution of the jet chambers is set to 20μ, z-coordinates are given in mm and z-amplitudes are normalized. The tapes have to be read by the routine READMC, which inserts experimental resolutions, takes into account inefficiencies and inserts random hits. The default values may be changed by the user. After the call to READMC, the event is available in /CDATA/Length, ID(4000).

Tape Format :

1. record	geometrical const. and chamber const.	
2. record	μ - chamber const.	
3. record	4-vectors)	repeated
4. record	event banks)	

a) The content of the constants records are described by comment cards in BLDAT and READMC.

b) Four vector record : BOS - format

0 word	total length	
1 word	'VECT'	
2 word	1	
3 word	0	
4 word	length	
5 word	event number	
6 word	NF total No. of final state particles	
7 word	NCF total No. of charged particles	
8 word	NNF total number of neutral particles	
9 -ff-	P(7,60)	repeated NF times
	P(1,N) ... P(4,N)	4-vector components
	P(5,N)	Mass
	P(6,N)	Charge) Integer
	P(7,N)	Type)

After a call to READMC the 4-vector data are stored in /C4VECT/VECT(424).

c) Data record :

1. HEAD

Header bank with fixed pointer table as described in JADE-Note No. 24 with one change. Now there is only one μ-filter bank instead of 6 as originally proposed.

BANK 1	HEAD	
BANK 2	TRIG	Pointer on LOC 55 in Head Bank
BANK 3	SCAL	Pointer on LOC 56 in Head Bank
BANK 4	LATC	Pointer on LOC 57 in Head Bank
BANK 5	ATST	Pointer on LOC 58 in Head Bank
BANK 6	ATOF	Pointer on LOC 59 in Head Bank
BANK 7	ALGL	Pointer on LOC 60 in Head Bank
BANK 8	JETC	Pointer on LOC 61 in Head Bank
BANK 9	CONC	Pointer on LOC 62 in Head Bank
BANK 10	MUEV	Pointer on LOC 63 in Head Bank

The first data word in the header bank, i.e. ID(5) now contains the event number.

2. TRIG

The organization of the trigger bank is not yet fixed. At present we work on the following scheme :

```
I*4  Word  1  )
 |         2  )     BOS
 |         3  )
 v         4  )

I*2    4 + 1  )
 |       + 2  )     T1  information
 |       + 3  )
 |
 |       + 4  )
 |        :   )     T2  information
 |        :   )
 |      + 10  )
 |
 |      + 11        bit 0 - 7     BP contr.  1 - 8   )
 |      + 12        bit 0 - 7                9 -16   )
 |      + 13        bit 0 - 7               17 -24   )
 |                                                   )
 |      + 14  )                    TOF       1 - 7   )
 |      + 15  )                              8 -14   )  latches
 |      + 16  )     bit 0 - 6               15 -21   )
 |      + 17  )                             22 -28   )
 |      + 18  )                             29 -35   )
 |      + 19  )                             36 -42   )
 |
 |      + 20  )                    LGRow     1 - 7   )  lead glass
 |      + 21  )                              8 -14   )  row latches
 |      + 22  )     bit 0 - 6               15 -21   )
 |      + 23  )                             22 -28   )
 |      + 24  )                             29 -35   )
 v      + 25  )                             36 -42   )
```

I*2	word 4 + 26	bit 0 - 7	LGQ	1 - 8	lead glass end cap quadrant latches
	+ 27	bit 0 - 3	LGEsum	1 - 4	total lg energy latches
	+ 28	bit 0,1	TAG		tagging latches
	+ 29	bit 0,15	JTRKA	1 -16	Jet Ch. Tracks all ($p > 0.2$ GeV)
	⋮				
	+ 34	bit 0,15	JTRKA	81-96	
	+ 35	bit 0,15	JTRKF	1 -16	Jet Ch. Tracks fast ($p > 1$ GeV)
	⋮				
	+ 40	bit 0,15	JTRKF	81-96	

3. SCAL

empty

4. LATC

I*4	word 1	)
	2	) BOS
	3	)
	4	)
I*2	4 + 1	Pointer to first TOF counter
	+ 2	Pointer to first beam pipe cntr
	+ 3	Pointer to first free position
	+ 4	empty
	+ 6	)
	⋮	) counter numbers
		)

5. ATST

empty

6. ATOF

empty

7. ALGL

I*4	word 1	)
	2	) BOS
	3	)
	4	)
I*2	4 + 1	Pointer to first barrel hit
	+ 2	Pointer to first -z end cap hit
	+ 3	Pointer to first +z end cap hit
	+ 4	Pointer to first free location
	+ 5	Block number) repeated
	+ 6	Amplitude (in MeV))
	⋮	

8. JETC - see JADE Computer Note No. 5

```
I*4     word 1  )
             2  )   BOS
             3  )
             4  )

I*2      4 + 1  )
           :    )
           :    )   Pointers
           +98  )
                )

           + 99    wire no.       )
           +100    amplitude -z   )   repeated
           +101    amplitude +z   )
           +102    drift time     )
```

9. CONC

empty

10. MUEV

```
I*4     word 1 )
             2 )   BOS
             3 )
             4 )

I*2      4 + 1     Pointer to face 1
           + 2     Pointer to face 2
           + 3     Pointer to face 3
           + 4     Pointer to face 4
           + 5     Pointer to face 5
           + 6     Pointer to face 6

           + 7     Pointer to first free position.
           + 8     empty

           + 9     wire No. (4+ CHAMB + Hit No. -1) )
          + 10     drift time                       ) repeated
          + 11     Δt longitud.                     )
             :
             :
```

In case of trouble with programs or libraries contact :

J. Allison

W. Bartel

E. Elsen

JADE Computer Note No. 11
P. Steffen - F11 -
17.11.1978

Use of BOS-Banks Generating Subroutine Package in the First Stage Data Reduction Program.

- In order to avoid the disadvantages of the standard set of 'BOS' subroutines (14 K-storage and additional time consumption) a special subroutine package is available on the library 'F11PRC,JADEGL' (1.5 K-storage). This subroutine generates selected 'BOS-banks' and sets the appropriate pointers in the 'HEAD'-bank.

- The input and output data are stored in the COMMON/CDATA/LENRCD, IDATA (5000).

```
    LENRCD = record length
    IDATA(1) = start of the 'HEAD'-bank.
```

- Initialization must be done once by the CALL JBCRX(NWMAX).

```
    NWMAX = length of array IDATA
```

- Creation of a bank can be done by the CALL JBCRE (IND,NA,NR,NW,IER):

```
 NA = name of the bank
 NR = number of the bank NA
 NW = length of the bank (in 4 byte words)
IND = pointer to first data word -1
```

(this is the same calling sequence and convention as within the BOS-system)

```
IER = error return code
    = 0 if bank has been created
    = 1 if bank with same name existing
    = 2 if not enough space available
    = 3 if name of the bank is unknown to the program
```

- Up to now only the following bank names are allowed and can be created with corresponding pointers in 'HEAD'

name:	pointer:	contents of bank:
JHTL	IDATA(69)	hit label array as defined in JADE Computer Note 5
PATR	IDATA(70)	results from pattern recognition
ZVTX	IDATA(71)	" " ZVERTF program
LGCL	IDATA(72)	" " lead glass programs
MUR1	IDATA(73)	} results from μ-chamber programs
MUR2	IDATA(74)	

- In addition the following calls might be useful.

```
CALL JBCR0 (IND,NA,NR,NW,IER): creation of a bank that is initialized
                               to zero
CALL JBCRA (IND,NA,NR,NW,IER,AR(1)): creation of a bank that is
                               filled with the data array starting
                               at AR(1)
```