JADE Computer Note 103
Sep 20th, 2003

A Guide to the Resurrected JADE Data and Software

Pedro A. Movilla Fernández

Contents

JADE Computer Note 103
Sep 20th, 2003

A Guide to the Resurrected JADE Data and Software

Pedro A. Movilla Fernández
(pedro@mppmu.mpg.de)

Abstract

This software note reviews the currently usable JADE data and the resurrected JADE software and includes a brief instruction manual for the detector simulation, the event reconstruction software and further utility programs.

1 Introduction

The present software note describes those parts of the JADE software and data sets which were resurrected within the framework of the re-analysis of the JADE data [1]. It is mainly intended as a guide for the correct handling of various stand-alone programs: the simulation of the JADE detector (`MCJADE`), the event analysis program formerly known as the JADE "Supervisor" (`SUPERV`), the JADE event display (`JADEZ`), and a data formatting package useful for top level physics analyses of multihadronic data (`ZE4V`). Detailed information about these programs are spread more or less incoherently over various "historical" JADE notes, JADE computer notes, and Ph.D. theses. The present note is an attempt to summarise the most relevant information taken from these sources and the program code itself. It might serve as a first orientation and a starting point to solve still remaining tasks.

The reanimation of the JADE software was mainly motivated by the necessity to perform tests of Quantum Chromodynamics at PETRA energies on the basis of state-of-the art methods. The programs considered here are essential to study the impact of the limited detector resolution and acceptance on the measurement of physical observables like event shapes and particle spectra. Some of the recently performed QCD analyses (e.g. [2–4]) are already based on newly generated detector simulation samples with modern underlying e^+e^- event generators.

A bigger part original software was developed on the former data acquisition computer (NORD-10S/50) of the JADE experiment as well as on the offline analysis computers (IBM/370) of the DESY computer centre [5]. Unfortunately, the documentation of the program libraries is partially rather imprecise and ambiguous. Anyhow, it was possible to adapt the programs on current computer platforms [1]. The most complete and successful tests were performed on IBM RS/6000 AIX machines. An adaption of the complete libraries to other computer platforms should be straight forward.

The note is organised as follows[1]. Section 2 gives an introductory overview of the main actions and functions of the programs, i.e. it sketches the simulated physical processes in the detector

[1] The reader who is not interested in all the technical details described in the note but want to use the programs should proceed to Section 4.1 and Section 4.3 for installation and Section 5.4 for running the programs.

and the main event analyses steps. Section 3 focuses on the structuring and the preprocessing history of the JADE data and describes the currently available data files. Section 4 describes the organisation of the program libraries and explains how to install them. The instruction guide for handling the main programs is presented in Section 5 which also includes a brief technical description of program flow. Section 6 describes some utility programs to handle the input and output data of the programs. The following two sections are intended only for those who want to perform further code development: Section 7 gives some technical details on the adaption adaption of the JADE software on current computer platforms, and Section 8 concludes with a summary of the remaining problems related to the software and the data.

2 Description of the Programs

In the following, the functionality of the reactivated parts of the programs based mainly on the standard settings of the steering parameters is summarised. See the references cited for more details. Some more technical information how to run these programs is given in Section 5. A description of the components of the JADE detector mentioned below can be found in [6].

2.1 The Tracking Simulation

`MCJADE` [7, 8] is the main steering routine for the simulation of the JADE detector response ("tracking simulation"). The program contains a detailed encoding of the geometry and the material composition of the detector. It passes a four-vector particle configuration externally provided by an event generator through the simulated JADE apparatus. Scattering processes through the various components and the probability of secondary meson decays are calculated at regular intervals along the particle trajectory. Monte Carlo methods are used to simulate the response of various subsystems, e.g. hits in the Jet Chamber and the energy deposited in the Lead Glass shower counters. The simulation of hits in the Jet Chamber is performed with 100% accuracy at this stage, with no noise or "dead" drift cells.

The program considers the following particles: photons, electrons, muons, pions, protons and neutrons, charged kaons and K^0_{L}. Charged particles are taken through helix trajectories up to the coil or the end caps of the Lead Glass, respectively. Multiple scattering off the traversed material (i.e. beam pipe, pressure vessel wall, Jet Chamber cell walls, drift gas) as well as the energy loss due to ionisation and bremsstrahlung is taken into account. Also nuclear interactions in the pressure wall and in the solenoid are considered. The position of the responding wires and the drift times are calculated taking magnetic field dependencies into account.

Neutral particles are tracked linearly from the interaction point to the Lead Glass. At each material layer, the photons are tested for conversion, with the resulting $\mathrm{e^+e^-}$ pair tracked through the detector like two charged particles. The simulation of the energy deposition in the Lead Glass is based on the empirical shower profiles in Ref. [9–11], taking light guide effects, possible hadronic interactions and the energy loss between the Inner Detector and the shower counters into account. Hadronic showers as well as low energetic electromagnetic showers are approximated in one dimension w.r.t. the direction of incident particle, whilst the evolution of the shower of high energetic electrons and photons is based on a three-dimensional shower profile [11]. Conversion and absorption losses in the material in front of the Lead Glass surface as well as energy fluctuations due to the varying penetration depths of photons are considered.

In case of minimal ionising particles, the energy may also be deposited directly, i.e. without any shower development. Nuclear interactions of neutral hadrons are neglected. The readout thresholds of the Lead Glass counters are set like in the experiment.

The tracking program contains also a simulation of the tagging system [12].

In the standard version of `MCJADE`, only track hits in the Jet Chamber and, depending on the detector configuration date, in the Vertex Chamber are generated. The program currently neither supports a simulation of the Z Chamber data nor the Muon System response. In the past, also a program version including the simulation of the Muon System was in use [13]. Unfortunately, the corresponding software libraries could not be retrieved as yet.

A more detailed simulation of the Lead Glass showers which is known as the "Tokyo Shower Program" [14, 15] is available but not reactivated as yet. This program considers the yield of Čerenkov photons induced by relativistic particles and also takes various secondary effects (reflexion and absorption effects within the counters and light guides, detection efficiencies of the photo multipliers) into account. This version was used in special JADE studies but never became standard at PETRA times due to the extreme computing time of the showers.

For historical reasons, the simulation of the resolution and inefficiencies of the JADE detector ("smearing simulation") as well as of the triggers is separated from the pure tracking simulation. Currently, these tasks are embedded within the standard JADE Supervisor, see below.

2.2 The JADE Supervisor

The JADE Supervisor `SUPERV` [16] coordinates various standard analysis routines used for the reconstruction of both real and Monte Carlo events. The present version supports an event analysis based on the data from the Jet Chamber, the Lead Glass calorimeter, and the Muon System. For the reasons given above, the latter is not relevant for Monte Carlo data provided by the present version of `MCJADE`. The program also comprises a simulation of the real detector resolution in case of unsmeared Monte Carlo events, as is the case for the standard `MCJADE` output.

Track Finding

The pattern recognition of tracks [6, 17] in the Jet Chamber is performed in the r-ϕ plane because the measurement of the drift time is by two orders of magnitudes more precise than the measurement of the z coordinate. First, triplets of adjacent wires which lie on a straight line are searched for and then connected to form track elements within each cell. Tentative parabola fits to the associated hits are performed, thus resolving most of the left-right ambiguities. Track elements in different cells are combined by extrapolating the resulting curves within given limits, starting in the outermost ring and going inwards. A fit is made to all points by a parabola or, in case of low track momenta, by a circle. Finally, all hits in the vicinity of the trajectory are re-examed and added to the track if they agree with within a given tolerance. For the points selected in this way, a straight line fit is attempted in the r-z plane which is repeated after eliminating badly fitting hits.

Several subroutines are available for refining the track fitting procedure, e.g. by adding vertex constraints or by performing a three-dimensional helix fit. These are not considered in the

standard version of SUPERV, but an implementation is straight forward. An extended version of the Supervisor that also includes additional analysis steps like track refitting, time-of-flight analysis, the calculation of the specific ionisation loss $\mathrm{d}E/\mathrm{d}x$ and more is embedded in the interactive JADE graphics program JADEZ, see next Section.

The present version of the Supervisor does not contain an analysis of the Vertex Chamber and Z Chamber data, since the corresponding software was never included in the standard libraries.

Cluster Finding

The cluster analysis algorithms are described in detail in [9, 18, 19]. The energy deposited in a Lead Glass block is calculated from the photo multiplier signals using conversion constants derived from an individual calibration of each block and a subsequent fine tuning at run time. Clusters are reconstructed by firstly searching for adjacent counters which had registered pulses above a fixed threshold value. Overlapping showers are identified by searching for nearby separable energy maxima within a coherent responding Lead Glass region. The sum of the block energies gives a first approximation for the cluster energy.

In the next step, the tracks found in the Inner Detector are extrapolated into the Lead Glass and then checked for correlation with clusters, taking multiple scattering and measurement errors into account. The cluster energies are corrected for associated tracks. Unassociated clusters are supposed to be photons. In case of associated clusters, a classification scheme depending on cluster energies and track momenta is applied in order to check for electron or photon candidates.

The point of impact of an incident particle is estimated by calculating the energy weighted coordinates of the Lead Glass blocks. In case of electron and photon candidates, the energy dependent location of the shower maximum is taken into account. In a later step, the determination of the barycentre of the clusters is refined by fitting a three-dimensional theoretical shower profile [9].

The raw cluster energies are corrected for various effects which were determined from a detailed simulation of showers as a function of the shower energy and the angle of incidence [19, 20]. Among others things, the starting point of the shower in the material in front of the Lead Glass surface, the point of impact on the Lead Glass surface, the angle dependency of Čerenkov light yield, the read out thresholds, and the leaking of showers are taken into account.

Smearing and Trigger Simulation

Since the Inner Detector tracking is normally done with a fine resolution, a supplementary simulation of the real detector resolution of the MCJADE output events is necessary. At PETRA times it was common practice to handle the tracking and the smearing simulation separately in order to save valuable computing time[2]. Once a tracking simulation of the ideal detector has been performed, the resulting MC data banks are available for further MC processing using the smearing conditions of different data taking periods but with one and the same detector configuration.

[2]For example, the pure tracking of a typical multihadronic event on the historical IBM machines took 5 seconds [10] on the average.

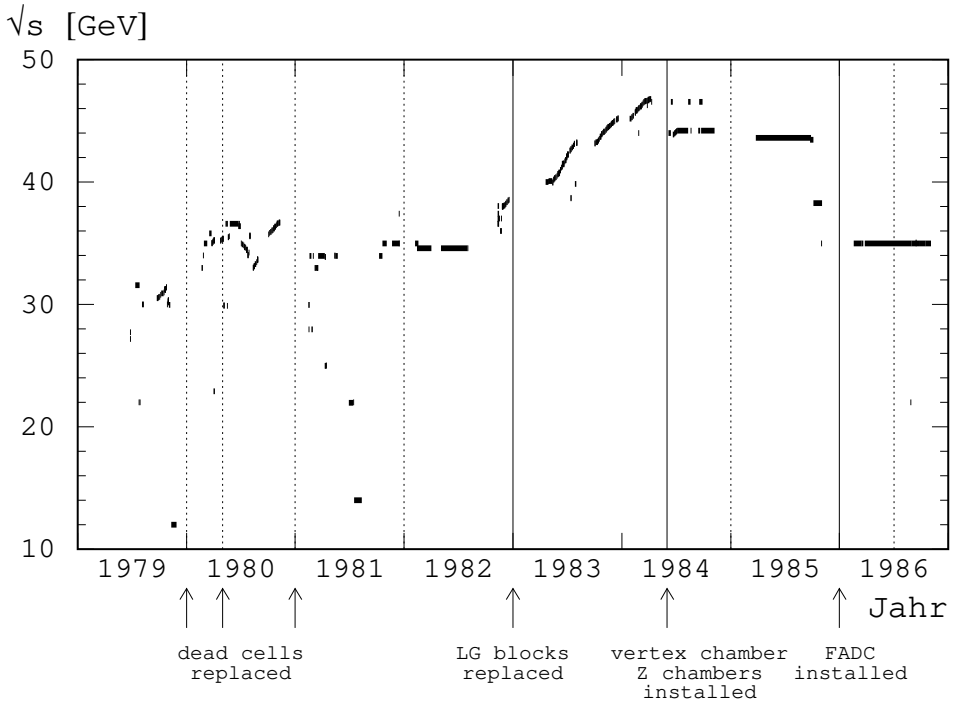


Figure 1: The diagram shows the history of the PETRA centre-of-mass energies and the corresponding detector status relevant for both the tracking and the smearing simulation. The vertical solid lines mark major changes of the hardware configuration, the dashed lines indicate the different data taking periods with constant smearing conditions.

The standard JADE Supervisor version allows to steer the smearing simulation subroutines[3]. The following effects depending on the conditions of the various data taking periods are considered [8, 21]: A smearing of the drift times and z coordinates, the generation of random hits, the elimination of hits according to wire efficiencies and inoperative ("dead") cells, the merging of nearby hits due to the limited double hit resolution. Furthermore, Lead Glass entries below a readout threshold are deleted.

Finally, a simulation of the T1 and T2 triggers [8] is performed subsequently by simply testing Lead Glass thresholds and logical combinations of drift cells with a minimum number of hit wires, both of which are based on the preceding simulation of tracks and Lead Glass energies.

Major changes of both the hardware configuration and the smearing conditions encoded in the JADE simulation as well as the centre-of-mass energies valid for the respective detector status are summarised chronologically in Figure 1.

2.3 The ZE4V Formatting

The program `ZE4V` basically extracts the most relevant event analysis information as e.g. output by the JADE Supervisor (e.g. track momenta from the pattern recognition and cluster energies

[3]It would be straight forward to organise the smearing step in another way, e.g. as a stand-alone program or in the framework of the tracking simulation `MCJADE`.

from the Lead Glass analysis) and converts the data into so-called ZE4V records, see [22] for a detailed description. This data format allows a faster access to the event information and is more suitable for top level multihadron studies than the Supervisor result BOS banks. A description of these data banks and of the ZE4V data is given in Section 3. The program is also able to read information from so-called TP data banks [23], another useful format suited for top level physics analyses.

In addition to the pure formatting, various cuts are imposed to the event information before writing it into a ZE4V record. Each track is required to have at least 20 hits in the r-ϕ plane and 12 hits in the r-z plane. The radial distance $R_{\min}$ between the track and the event vertex and the absolute value of the z coordinate of the track origin must not exceed 50 mm and 350 mm, respectively. The minimum momentum of a track must be greater than 50 MeV/c. A Lead Glass cluster is accepted as a photon candidate and kept in the record, if after subtracting the minimum ionising energy for each connected track, the remaining cluster energy exceeds 50 MeV.

During JADE operation, the ZE4V package never became part of the standard JADE libraries. Several private program versions exist which were adapted for the special needs of individual analyses. The present reactivated program performs additional analysis steps for electron candidates and also includes a simulation of the energy loss $\mathrm{d}E/\mathrm{d}x$ of tracks in the case that Monte Carlo data is processed.

2.4 The JADE Event Display

The interactive graphics program `JADEZ` [24] is an extended version of the JADE Supervisor allowing for detailed graphical representations of the individual analysis steps and of the detector components. The reconstruction details and the event display options can be steered very flexibly by keyboard control and mouse actions. The steering commands are described in detail in Section 5.5 and in Appendix C.

The default view of the event displays the central detectors, the Time-Of-Flight (TOF) hodoscope, and the Lead Glass shower array in the r-ϕ projection. Figure 2 shows a simulated JADE event at a centre-of-mass energy of $\sqrt{s} = 35$ GeV, corresponding to the detector configuration of the year 1986. The top and the bottom of the event display contain some text information like run and event number, data period, trigger information, and a brief summary of the detector response. The energies shown in the Lead Glass blocks are in MeV and represent the sum of all energies in the complete row of blocks along the z direction. The TOF results are displayed in nanoseconds with the counter numbers also shown. The Jet Chamber and the Vertex Chamber hits are drawn with their mirror hits. The right top and right bottom part of the event display shows the orthogonal projections of the same event.

The figure at the bottom is a graphical representation of some of the main event reconstruction results. Obviously, the left-right ambiguities in the Jet Chamber were resolved successfully, random hits were rejected, and track elements found in the individual cells are connected to tracks[4]. Furthermore, the cluster analysis was capable of identifying photon candidates.

The program supports a variety of further analysis steps as well as the manual editing of events and the recalibration of the data,

[4]Note that the present version of the event analysis does not involve the Vertex Chamber data.

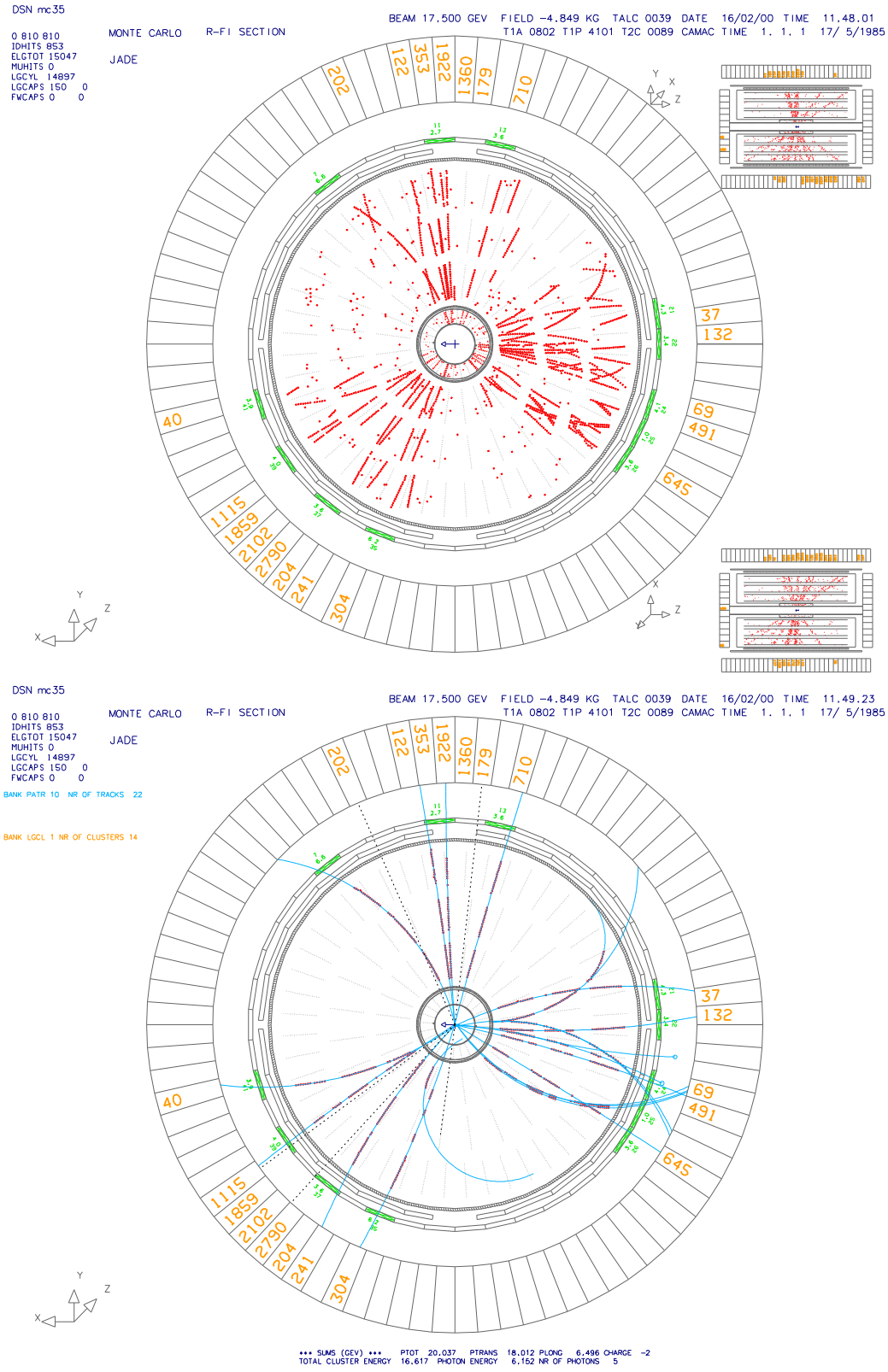


Figure 2: A newly simulated event at $\sqrt{s} = 35\,\mathrm{GeV}$ displayed by the JADE interactive graphics program `JADEZ` in the r-ϕ projection. The upper plot shows the simulated hits in the Jet Chamber and the Vertex Chamber (including left-right ambiguities), TOF entries and the sum of energies deposited in the Lead Glass. The lower plot represents various results of the event reconstruction. The solid lines represent the tracks found by the pattern recognition programs, the dashed lines indicate the results of the cluster analysis.

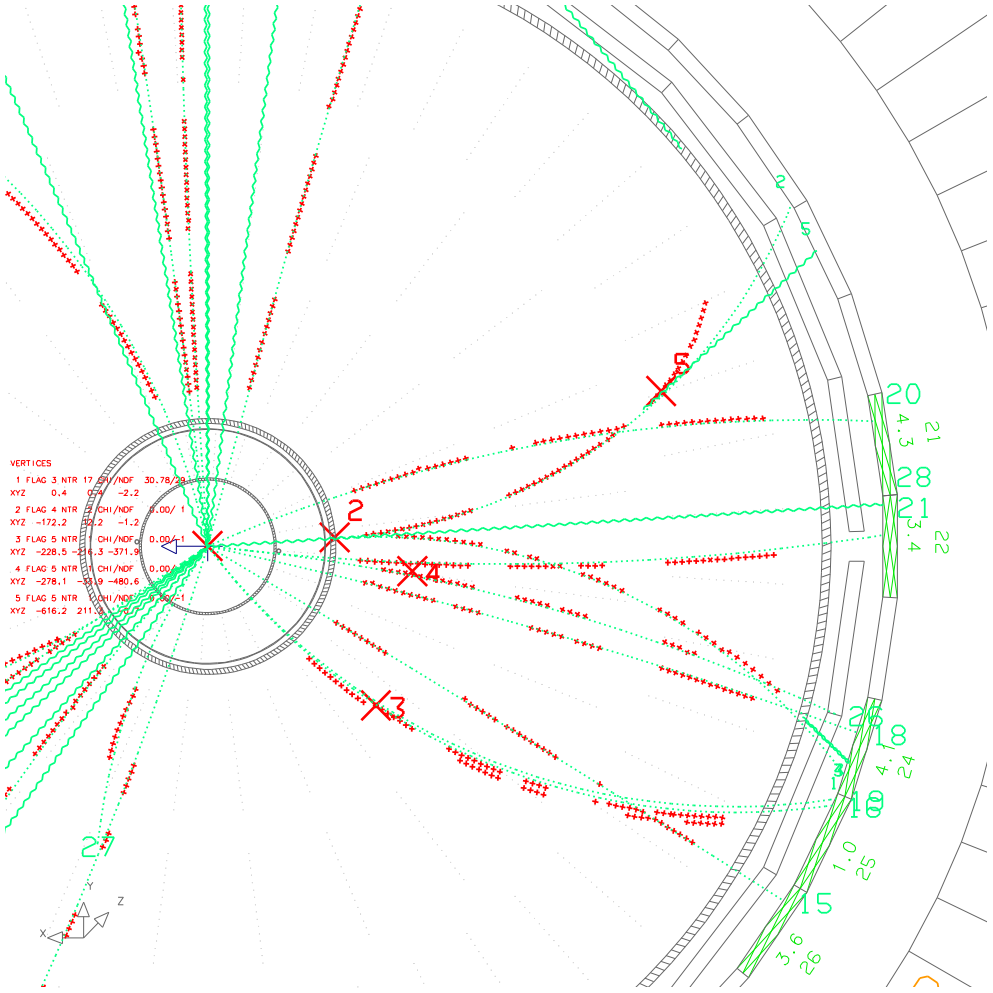


Figure 3: A zoomed section of the reconstructed event shown in Figure 2 compared with the underlying four vector data delivered by an external QCD event generator. The lines represent the extrapolated trajectories of the initial particle configuration and of particles originating from secondary interactions with the detector material (dotted lines are charged particles, wavelike lines are photons).

Figure 3 shows a zoomed section of the same event, with the initial particle four vector configuration supplied by a QCD event generator[5] on top. Obviously, there is a correlation between the simulated detector response and the four vector input. Furthermore, secondary interactions which are also apparent in the real data are visible in the simulation (e.g. photon bremsstrahlung, e^+e^- conversion, and multiple scattering). Various further event display pictures of these type qualitatively demonstrate that the simulation and the analysis software packages work reliably.

Note that the historic version of the JADE graphics provided only a monochromatic view of the event. In the reanimated version, colours were introduced in order to improve the graphical representation of both the detector signals and the analysis information.

[5] PYTHIA 5.7

3 Description of the JADE Data

This section is intended to give a principal idea of how the currently available JADE data are structured. A summary of the processing history and a list of data files is presented.

3.1 Data Formats

All available JADE data, i.e. real data after the REDUC1/REDUC2 reduction steps [25, 26],as well as the Monte Carlo data generated for the present reanalysis, are mainly organised in the framework of the data management system **BOS** [27], version 1979. It was introduced in the 1970ties for the handling of the online and offline data flow of high energy physics experiments. A JADE BOS record contains the full and detailed information about the response of all detector subsystems and the event reconstruction results.

Several compact data formats with compressed JADE event information were used in the past for various physics studies. The already mentioned **ZE4V** [22] format was initiated by a subgroup of the JADE collaboration particularly for the studies of multihadronic final states, allowing for a simplified handling of event information independently from BOS. Another commonly used (but BOS based) data format with reconstructed and summarised JADE events is known as the **TP** [23] format. Corresponding files were not retrieved as yet. However, the software for creating TP data banks is partially available but not reactivated as yet.

3.1.1 BOS Records

Within the framework of BOS, all event data and analysis information are structured in *banks* of variable length. A JADE event record is a dynamically arranged ensemble of standardised BOS banks that represent the subsystems of the detector (the raw Jet Chamber data, the Lead Glass ADC counts, etc.) or event analysis result information (tracks from the pattern recognition, Lead Glass cluster, etc.). A complete JADE event record consists of about roughly two dozen BOS banks.

A BOS bank is identified by a *bank name* and a *bank number*. Each bank has a header of four 32 bit words for administrative purposes (bank name, bank number, pointer to the next bank with the same name, number of data words), followed by the data part. For illustrative purposes, some of the most important BOS banks are sketched in the following:

The Jet Chamber data is supplied by the BOS bank named `JETC` [28] which is a bank of 16 bit words. The first part of `JETC` contains pointers to the first hit in each of the 96 drift chamber cells. In the second part, each measured hit is referred to by a sequence of four 16 bit integer words representing the wire and hit number (combined to one 16 bit integer), the measured pulse amplitudes at the two wire ends (ADC counts) and the drift time (a multiple of a given time unit). The results of the track finding programs are stored in the track bank `PATR` [29] and the hit label bank `JHTL` [30]. Each track in `PATR` is represented by a group of 32 bit integer words and floating point numbers, giving information about the curvature, the fit type, the corresponding fit parameters, the fit quality etc. The hit label bank contains two 16 bit words for each measured hit which encodes the correlation between hits and tracks (e.g. distance between hit and fitted track). The results of different analysis steps are labelled by different numbers of the tracking bank. For example, `PATR/10` (i.e. bank `PATR` number `10`) is usually the

standard pattern recognition result bank, and `PATR/8` may refer to the result of a refined track fitting procedure, see e.g. [31, 32].

The raw Lead Glass data are supplied by the bank `ALGL` [18]. Basically, it contains ADC counts (16 bit words) and the corresponding channel numbers (16 bit words) of hit blocks in the Lead Glass hodoscope. The pulse heights converted to MeV (after calibration) and sorted according to the clusters are stored in the bank `ALGN` [18]. The bank `LGCL` [18] contains the final cluster analysis results, each cluster is represented by groups of 16-/32-bit integer and floating point numbers giving information about cluster energies and directions, shower profiles, track correlation, etc.

The names of the most important data banks and the corresponding references to explanatory JADE Notes and JADE Computer Notes are summarised in Appendix A. The raw data and reconstruction banks are generally organised as compact mixtures of 16 bit integers, 32 bit integers, and floating point numbers, since at PETRA times data acquisition and processing was dictated by the demand of saving memory resources. From the today's point of view this structuring of the data is extremely problematic since the correct accessing of a data words or single bits in BOS banks might depend on the endian convention of the computer platform (Section 7).

3.1.2 ZE4V Records

The ZE4V data format is intended to be administrated autonomously without using BOS, although the ZE4V software itself handles them within the BOS environment. A ZE4V event record is a mixture of 16 and 32 bit words. It starts with a *global header part* containing information about the structuring of the event record and about global event properties. Then follows the particles section with the measured quantities of the reconstructed charged particles and photons. Each particle section begins with a *general part* (identical for charged and neutral particles) containing the four momentum, charge etc. of the particle. This is followed either by a *specific section for charged particles* (containing e.g. the number of Inner Detector hits or the DOCA point of a track) or a *specific section for neutral particles* (containing e.g. the uncorrected cluster energy, the number of responding Lead Glass blocks). A ZE4V record also foresees a Monte Carlo section filled with the four vectors of the initial parton/hadron configuration of a simulated event.

3.2 Data Preprocessing History

The JADE data have passed a multiple level trigger and various filtering programs. The first level trigger T1 [33] was an energy trigger based upon the fast analogue signals of the Lead Glass. The track trigger T2 [34] used the signals from the Jet Chamber, and the third level trigger T3 [35] was based on the Muon Filter drift data. The trigger actions are summarised in [6, 36]. Parallel to the read-out, the events were processed by fast online filtering algorithms [36] (see also [37, 38] for more details) in order to reject background events like cosmic events, beam-gas interaction and electronic noise. The accepted events were directly transferred from the data acquisition computer[6] to the IBM mainframes[7] of the DESY computer centre. The data

[6] A NORD-10S coupled with a NORD-50 by Norsk Data [39].
[7] IBM/370

acquisition system is described in detail in [40]. Both the online and the offline data flow were managed within the BOS framework.

After a general reformatting procedure [6] and before entering any individual analysis chains, the data were passed through a two-staged offline filtering procedure known as REDUC1 [25] and REDUC2 [26], in order to further enrich the fraction of signal events in the data samples. The standard cuts imposed at this stage are partially based on fully reconstructed event information as e.g. supplied by the result banks `PATR` and `LGCL` of the Supervisor.

All currently available BOS raw data banks underwent these processing steps. In case of the ZE4V data, the additional quality cuts for tracks and clusters described in Section 2.3 were imposed.

3.3 Currently Usable Files

In 1997, the raw JADE BOS banks were transferred from the IBM computers of the DESY computer centre onto EXABYTE cartridges [41]. The IBM data has been converted into a machine independent word format using the FPACK conversion program [42] with the default format 'B32', which means that the dynamical BOS bank data were rigidly copied word by word as 32 bit patterns, irrespective of the fact that the BOS data banks are dynamically built up by a mixture of data words of different types and lengths. Up to now, neither the FPACK data were retranslated into the original BOS banks, nor the reactivated software has been extended for handling FPACK records. As a consequence, *the original raw JADE data are not readable* as yet, although the reactivated software should be capable of reading and processing *binary BOS banks with big endian scheme*. In contrast to that, the format of the newly generated BOS banks provided by the present detector simulation is suitable for the JADE Supervisor as long as both programs run on the same computer platform.

Up to now, the following files are *usable* for the purpose of analyses:

- **Real JADE data in ASCII formatted ZE4V:** All usable real data on which the JADE reanalysis is based as yet are available as plain text files logically structured like ZE4V records. When transferring these data from the old IBM tapes to modern data carriers, the memory consuming ASCII format was chosen in order to guarantee a simple and unambiguous access to the information also on later computer platforms.
- **MC JADE data:**
  - **ASCII formatted ZE4V:** The same as mentioned above applies also for some "historical" detector Monte Carlo files preprocessed in the 1980ties.
  - **Binary formatted BOS:** These are newly created BOS data banks from both the detector simulation `MCJADE` and the subsequent event analysis programs `SUPERV` and `ZE4V` described in Sections 2.1, 2.2, 2.3.
  - **Binary formatted ZE4V:** These files were generated after processing the simulated BOS banks with the `ZE4V` program (Section 2.3)

The currently usable files with the corresponding numbers of events are listed in Appendix B.

3.3.1 Available ZE4V Versions of Real Data

Two versions of data samples with enriched multihadronic events packed in the ZE4V format are available as yet, commonly referred to as "9/87" and "5/88" data. The data belonging to the first one were rescued from the original IBM computers of the DESY computer centre [43], the data belonging two the second version were retrieved from magnetic IBM tapes found at the Physics Department of the Heidelberg University [44]. The notations presumably correspond to the *reconstruction version* based on the so-called "TP" event analysis program, see [23, 32, 45, 46]. The information stored in the "general particle sections" of the ZE4V records (Section 3.1) indicate that the currently available ZE4V data were not derived from the standard Supervisor result banks (`PATR`, `LGCL`) but from the TP result banks (Appendix A). As already mentioned, the `ZE4V` program provides also the processing of these banks. In contrast, the newly created Monte Carlo ZE4V records are based on the `SUPERV` result banks.

Unfortunately, no detailed documentation of the exact preprocessing which the present data underwent could be retrieved, so the differences between the two data versions are not clarified up to now. Some information about presumably standard event reconstruction steps is given in various JADE Computer Notes [31, 32, 45, 47–49], of which some of them explicitly refer to TP program options.

Table 1 summarises the number of events after applying the multihadronic selection criteria [1] to both data versions for selected run periods and energy bins. Generally, the 9/87 data sample contains less multihadronic events. In case of the energy bin around 44 GeV, a considerable amount of events is missed in the 9/87 version compared to the 5/88 data.

3.3.2 Available ZE4V Versions of MC Data

The currently available *preprocessed* ZE4V Monte Carlo samples were obviously derived from TP records as well, and not from the Supervisor result banks. They corresponding detector simulation was run at centre-of-mass energies at $\sqrt{s} = 35\,\text{GeV}$ and 44 GeV, with the hardware configurations of 1982, 1985 and 1986, respectively. Apart from the event reconstruction details, the parameter settings of the underlying event generator (mostly JETSET 6.3 parton shower plus string fragmentation model) are not known with absolute certainty. They are presumably given in [50, 51].

In contrast, the processing history of the *newly generated* ZE4V records is of course not affected by any of these ambiguities (Section 2). The data files are based on the QCD event generators PYTHIA 5.7, HERWIG 5.9, ARIADNE 4.08, and JETSET 6.3, which were run at centre-of-mass energies about $\sqrt{s}$ =14, 22, 35, 38 and 44 GeV with the detector configuration dates listed in Appendix B. The parameter settings are described in detail in [1]. It must be reemphasised that these Monte Carlo samples underwent only the standard analysis procedure described in Section 2.2, without further refining steps. Different from that, the references cited above indicate that the final analysis of the real data mentioned in Section 3.3.1 does include various refinement steps (like 3-dimensional helix fit to Jet Chamber hits with common vertex constrain). Furthermore it is likely [32] that the 9/87 reconstruction considers also the data provided by Vertex Chamber and the Z-Chamber.

$\sqrt{s}$ range [GeV]	data taking periods	run periods	$\mathcal{L}$ [pb^{-1}]	$\langle\sqrt{s}\rangle$ **[GeV]**	**multihadrons** **9/87**	**5/88**
14.0	Jul-Aug 1981	7968-8629	1.46	**14.0**	**1734**	**1792**
22.0	Jun-Jul 1981	7592-7962	2.41	**22.0**	**1390**	**1408**
33.8 - 36.0	Feb 1981 - Aug 1982	6193-12518	61.7	**34.6**	**14372**	**14347**
35.0	Feb-Nov 1986	24214-30397	92.3	**35.0**	**20688**	**20925**
38.3	Oct.-Nov. 1985	23352-24187	8.28	**38.3**	**1587**	**1605**
43.4-46.6	Jun 1984 - Oct. 1985	16803-23351	28.8	**43.8**	**3940**	**4397**

Table 1: JADE multihadronic event statistics from selected data taking periods. $\mathcal{L}$ denotes the corresponding integrated luminosities, $\langle\sqrt{s}\rangle$ die weighted means of the respective centre-of-mass energies.

3.3.3 Available BOS Versions of MC Data

The complete BOS files output by `MCJADE`, with PYTHIA as underlying event generator, and also random samples of the `SUPERV` result banks on which the newly generated ZE4V records are based were kept for later purposes and cross checks (Appendix B). All of these files are suitable to be directly processed by the Supervisor or the interactive JADE graphics.

The currently available `MCJADE` BOS banks might for example be useful for future tests of the already mentioned refined event reconstruction steps which are still inactive but highly desirable to be included in the Supervisor.

3.4 Calibration Files

For the same reasons mentioned in Section 3.3, the files with the JADE calibration constants were converted to ASCII text files. For the reanalysis, these files were reconverted to a binary format suitable to be processed by the present JADE Supervisor version. Some obvious formatting errors made when rescuing the files were found in a certain sets of constants. They were repaired by hand as far as they could be detected by visible inspection.

The JADE calibration scheme is described in detail in [52]. Further information about the calibration constants can be found in [53–55]. The currently relevant calibration data with about 17 different groups of constants reside in three files (see Section 4 for the structuring of the resources) with the following generic names:

`bupdat0`, `bupdat1`: These are the complete sets of calibration constants which also include a list of so-called "spinning" Lead Glass blocks, i.e. Lead Glass blocks which sent permanent electronic noise and are thus useless for analysis. This list of noisy blocks changes from run to run and is needed whenever the Lead Glass data is calibrated. It was e.g. relevant for the REDUC1 [25] step and was used in the past only in special cases of a recalibration of Lead Glass pulse heights. `bupdat0` contains the constants up to run 10000, `bupdat1` contains the constants from run 10000 on.

`aupdat1`: This is a considerably compressed version of the files mentioned above, without the memory and time consuming list of spinning block constants. Usually, `aupdat1` is the *standard calibration file*.

The calibration data are not organised in BOS banks. Whenever needed, the calibration files are read sequentially by a standard subroutine called e.g. by the JADE Supervisor (Section 5.2). Basically, a record of the calibration file has a header with administrative information (like the name of the constants group, the time from which on the constants are valid etc.), followed by the calibration constants, see [52] for explicit information. Note that the binary converted calibration files are platform dependent just as the binary BOS banks. A conversion tool is supplied (Section 6) to translate the ASCII files into the correct binary format valid on the current computer platform used.

The calibration constants are not needed for MC data, but nevertheless the present Supervisor program asks for the calibration files also in this case. A complete test of the reliability of the calibration files could not be performed until now since the raw FPACK formatted data are not readable as yet.

4 JADE Program Libraries

4.1 Location

The JADE software and the calibration files reside in a `tar` archive file located in

`http://home.cern.ch/~movilla/jadesoft.gtar.gz`,
`http://www.mppmu.mpg.de/~pedro/jadesoft.gtar.gz`.

After downloading, type the command '`gtar xzfv jadesoft.gtar.gz`' in order to extract the source code and the installation tools.

4.2 Organisation

The top directory `jadesoft/` of the software package is structured into subdirectories containing the following resources:

`src0/` — The original untouched JADE source code retrieved from the former DESY IBM.
`src/` — The modified JADE source code needed for the installation.
`main/` — The modified source code of the main programs.
`lib/` — The location of the compiled libraries.
`bin/` — The location of the executables.
`job/` — The steering jobs for the main programs.
`util/` — Some utility programs.
`cal/` — The JADE calibration files.
`info/` — Information about JADE luminosities and run periods.

- The directory `src/` contains the software resources which are relevant for the installation of the programs: the BOS system (version 1979), main parts of the JADE libraries, the emulation software for some obsolete DESYLIB functions, for the PLOT-10 Terminal Control System and for various IBM/370 FORTRAN functions:

`jmc/` — "Tracking simulation" software.
`jadegs/` — General source code, includes also the "smearing simulation" code.
`patrecsr/` — Inner Detector pattern recognition software.
`source/` — Lead Glass analysis software.
`vertex/` — Vertex finding software.
`wertex/` — Vertex finding software (newer version).
`jademus/` — Muon Chamber analysis software.
`tagg/` — Tagging system software.
`toflib/` — TOF system software.
`grafix/` — Graphics event display code.
`zlib/` — `ZE4V` packing software.
`jadesr/` — Obsolete analysis routines not needed so far.
`boslib/` — BOS data management system.
`interface/` — Emulation software.

- The directory `src0/` with the untouched software contains basically the same directories, except `interface/`, `vertex/` and `wertex/`. The latter two libraries were originally embedded within `jadegs` but were reorganised here for practical reasons. Furthermore, `src0` contains the following resources:

  `jade56/` — JADE simulation version including the "Tokyo Shower Program".
  `tp9lib/` — TP9 event analysis software.

- For practical reasons, the main programs and the corresponding user steering routines (Section 5) are located in `main/`:

  `mcjade/` `mcmain.f mcjade.f mcuser.f ...` — The tracking simulation.
  `superv/` `jdmain.f user.f ...` — The JADE Supervisor.
  `jadez/` `gphmain.F xuser.F` — The interactive JADE graphics.
  `ze4v/` `ze4vjb.f` — The ZE4V packing program.

- The calibration files (Section 3.4) are located in the directory `cal/`:

  `aupdat1-ori` — Compressed calibration file for all runs (ASCII format).
  `bupdat0-ori` — Extended calibration file for runs 1-10000 (ASCII format).
  `bupdat1-ori` — Extended calibration file for runs > 10000 (ASCII format).
  `aupdat1` — Corrected version of `aupdat1-ori`.
  `bupdat0` — Corrected version of `bupdat0-ori`.
  `bupdat1` — Corrected version of `bupdat1-ori`.
  `aupdat1.b` — Binary version of `aupdat1` (standard calibration file).
  `bupdat0.b` — Binary version of `bupdat0`.
  `bupdat1.b` — Binary version of `bupdat1`.

  Only the binary versions are relevant for the event processing.

- The directory `util/` contains various utility routines (see Section 6):

  `mcgen/` — Modified OPAL MC package with CPROD interface
  `ccal/` — Conversion tool to generate binary formatted calibration files.
  `zread/` — Program to read ZE4V data.
  `Scripts/` — Some installation tools.
  `Pbin/, Rbin/` — Further programs needed for installation.

4.3 Installation

For the installation of the libraries and the programs, execute `gmake` within the root directory `jadesoft/`. Up to now, *a complete installation is only supported on RS/6000 AIX platforms* using the `xlf` compiler[8] and the `cpp` compiler preprocessor. The compilation procedure is specified by the following files:

`GNUmakefile`	—	The make file.
`GNUmake-objects`	—	The list of object files relevant for the program libraries.
`GNUmake-rules`	—	The rules to compile the FORTRAN files.
`Incdep.ksh`	—	A tool to extract `#include` dependencies.

To perform complete installation of the libraries and the main programs, simply type '`gmake all`'.[9] But also partial installations or removals are possible.

- '`gmake all`' performs complete installation
- '`gmake <lib>`' compiles the library `<lib>` (`<lib>` = `jadegs`, `patrecsr`, ...)
- '`gmake lib`' compiles all libraries
- '`gmake <main>`' creates the executables (`<main>` = `mcjade`, `superv`, `jadez`, `ze4v`)
- '`gmake clean`' removes all `.a`-, `.o`-files and also the `src/<lib>/TMP` directories
- '`gmake mclean`' removes all temporary make files
- '`gmake aclean`' removes all archive files

The generated libraries and executables are located in `lib/` and `bin/`, respectively[10]. Various environment variables must be set before installation. See the comments in `GNUmakefile` for further explanations.

5 Handling of the Main Programs

The directory `job/` contains the various shell scripts suited for the main programs:

`mcjade/mc.ksh`	—	steers the tracking simulation `mcjade`
`superv/sv.ksh`	—	steers the JADE Supervisor `superv`
`ze4v/ze.ksh`	—	steers the ZE4V packing program `ze4v`
`jadesim/jadesim`	—	steers the whole simulation and analysis procedure

The `jadesim` script is the recommended tool for running the JADE simulation. All input/output is steered via old fashioned FFREAD cards. The main task of the shell scripts is to fix the FFREAD card entries relevant for the job and to administrate various input and output files. In the following, the action of the programs are briefly described. Most of the switches in the scripts are self-explanatory.

[8] Available RS/6000 AIX platforms at the MPI are: iwsatlas2, the iwsh1 cluster, the comps cluster.

[9] Note that up to one minute may pass before the first visible action.

[10] The `xlf` compiler prints various `xlf` warning messages regarding some risky FORTRAN constructs in the code. They seem to be harmless so far, but see Section 7.

5.1 mcjade

The tracking simulation `mcjade` processes the 4-vector configuration files provided externally by HEP event generators. The events contained in these files must be in the so-called CPROD-format [56–58] (as e.g. provided by the programs described in Section 6). The program passes such CPROD records event by event to the tracking simulation and outputs the generated response of the various detector subsystems (e.g. Jet Chamber hits, Lead Glass entries) as binary BOS banks (Section 3.1.1, Appendix A). The detector configuration and various tracking options can be fixed by the calling script `mc.ksh`, see below.

Usage:

- Input files:
  - The binary CPROD 4-vector data file (suffix `.cprod`).
  - The random number generator status file `mcjade.stat` of a previous `mcjade` run. If this file does not exist, the default random number seed is used to initialise the random number generator.
- Output files:
  - A binary BOS file with the tracking result banks (suffix `.bos`).
  - The random number generator status file `mcjade.stat`.
  - Optional: A control histogram file (suffix `.hist`) and a text file (suffix `.bnk`) with the BOS banks contents in a readable format.
- FFREAD card steering variables: (`<...>` are placeholders for generic file names.)

MCSTART	ON, OFF	initialisation of the random number generator
detector configuration date:		
YEAR	1979...1986	year
MONTH	1...12	month
DAY	1...31	day
tracking options:		
SMEAR	ON, OFF	smear photon and electron energies
CONVERT	ON, OFF	photon conversion in outer tank and coil
ABSORB	ON, OFF	absorption losses
SHOW3D	ON, OFF	3 dim shower profile fit
VERTEX	ON, OFF	ON: perform Vertex Chamber tracking OFF: consider old beam pipe geometry and beam pipe counters (hardware configuration before May 1984)
number of events to be tracked:		
FIRST	1,2,...	first event to read from CPROD file
LAST	0,1,...	last event to read from CPROD file (0: process until end of file)
MCNPRI	1,...,100,...	max. number of events with 4-vector dump
I/O files:		
MCVECT	<genout>.cprod	CPROD input file
MCBOS	<mcjade>.bos	output file with the tracking result BOS banks
MCBANK	<mcjade>.bnk	output tracking result banks (ASCII format, only for tests)
BANKS	ON, OFF	print out option for this file
MCHIST	<mcjade>.hist	control histogram file
HISTO	ON, OFF	print out option for this file

Description

The program flow is sketched in Figure 4 (it is not a flow chart but nevertheless helpful for a first orientation). The program starts with the routine `MCMAIN` where the BOS system, the HBOOK histogramming package, and the FFREAD I/O steering package are initialised. Here, also the random number generator is prepared for the current run. The initial seed from which to start the random number sequence can by demand be provided externally by a random number status file (`mcjade.stat`).

After reading the FFREAD card entries, the master program `MCJADE` is called. The first action of `MCJADE` is to initialise the status of the JADE detector, i.e. the geometrical constants, the characteristic physical and electronic values, the default tracking options, etc. The corresponding default parameter settings are passed to the tracking program by linking the common blocks of the block data located in the file `jadegs/jadebd.F`, which is of central administrative importance for the whole JADE software. At the beginning, the tracking options and various information about the active software packages used for tracking are printed out.

Each event loop starts with doing some event initialisation and passing the 4-vector configuration of the currently read event from the external file to the common block `/CPROD/`. Each particle is then passed through the detector material, with the corresponding tracking status stored and updated in the BOS bank `VECT/0`. Particles originating from secondary interactions with the detector material are stored in the bank `VECT/1`. The four vectors and further particle properties of the first events are dumped. See Figure 4 for an overview of further tracking steps and of the most important tracking subroutines involved. For detailed information check the references cited in Section 2.1. At the end of an event loop, the tracking results are arranged in BOS banks (Appendix A) and written out. The most important are `JETC` containing Jet Chamber data, and `ALGN` containing the simulated Lead Glass response. At the end of the program, the file `mcjade.stat` is updated with the current status of the random number generator.

If need be, the routines `MCMAIN(mcmain.f)` and `MCUSER(mcuser.f)` are the appropriate places for easy modifications of further steering and output options which are not foreseen to be changed in `mc.ksh`. Some important switches are contained in the common blocks:

`/TODAY/`	—	contains the detector configuration date
`/CFLAG/`	—	contains general tracking option flags
`/CVFLAG/`	—	contains input data validation flags

`/TODAY/` and `/CFLAG/` are set in `MCMAIN` (`mcmain.f`) via the FFREAD card entries shown on Page 17. `/CVFLAG/` steers the validation routine `MCVALI(jmc/mcvali.f)` [59] which works off a list of input data consistency tests (like particle type code checks or consistency checks for the particle input masses, momenta and energies). Additional steering common blocks introduced for the software reanimation are also set in `MCMAIN`:

`/MYIO/`	—	I/O steering of histogramming/additional printouts
`/CPFORM/`	—	endian format flag

`/MYIO/` allow for communication with a user specific subroutine `MCUSER(mcuser.f)` introduced here for test purposes. The user routine is executed once at the end of each particle loop and is called by `WRTMCB(jmc/wrtmcb.f)`, which is responsible for writing out the BOS events.

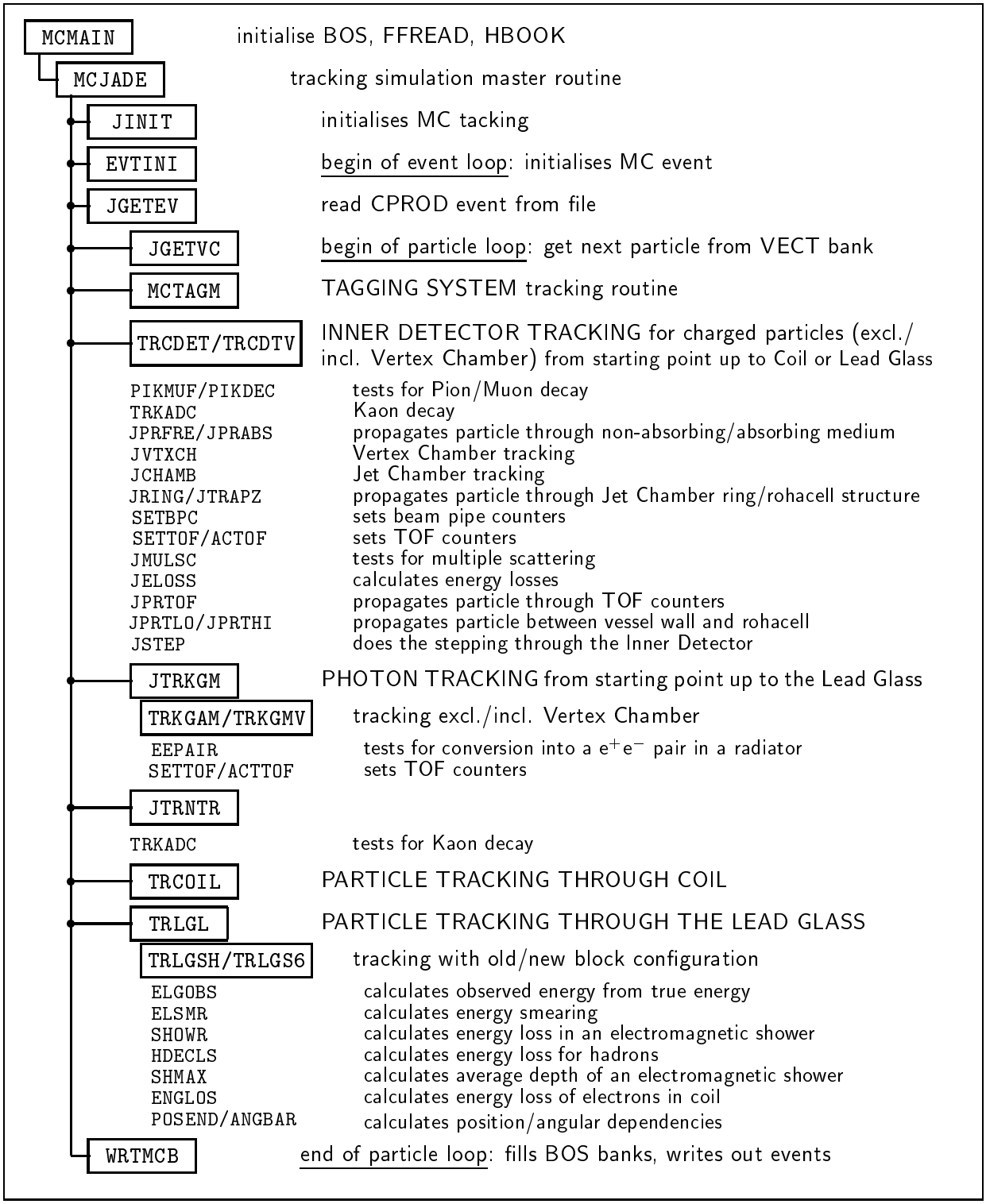


Figure 4: Schematic overview of important MCJADE tracking subroutines.

/CPFORM/ contains a flag which tells the 4-vector reading procedures JGETEV(jmc/jgetev.f) and BRVECT (jmc/brvect.f) the endian format of the current computer platform. These subroutines were modified in order to allow for the correct processing of CPROD files generated on *both big endian and little endian machines*. At the beginning, MCMAIN uses the function ENDIAN(mcmain.f) to match the endian schemes of the read CPROD file and of the platform where mcjade runs.

Remarks

It was found that the tracking program runs robustly. No remaining serious bugs were detected after generating some millions JADE events. Some attention must be paid to the following.

- The newly simulated JADE detector contains tiny but artificial "dead zones". This is due to the higher computing precision of the RS/6000 platform presently used for particle tracking compared to the original IBM/370, as a consequence of the different floating point representations (the basis of the mantissa bits is 16 on the IBM/370 and 2 on the RS/6000, the latter obeying the IEEE standard.). Caused by rounding errors in the calculation of the particle trajectories and of the geometrical acceptances of the detector subsystems, the program sometimes fails to assign a particle to one of two adjacent detector parts if the currently calculated particle position is coincidentally located nearby the boundaries between. Resulting infinite loops (approximately each 10^5 th event) were found in the Inner Detector algorithms JCHAMB(jmc/jchamb.f) and JPRTHI(jmc/jprthi.f). They were removed so far, but further potential infinite loops in other parts of the tracking program cannot be excluded for future MC runs.
- An often produced warning message originates from the validation subroutine MCVALI. They mostly refer to energy-momentum mismatch of very low energetic particles which can be regarded as harmless. In case of multihadronic events, it rarely happens that an input four vector configuration is rejected (as is at times the case in events distorted by hard photon bremsstrahlung). It was found that is desirable to disable the vertex checks.
- The random number seeds are administrated by a new subroutine called MCRAND(mcjade/mcrand.f). It ensures that the current status of the random number generator is ported correctly between subsequent MCJADE runs (via the status file mcjade.stat). For random number generation, a modified version of the CERNLIB subroutine RANMAR (interface/ranmar.f) is used, since it was found that the original version (V-113 from 1996) does not handle the random numbers counters correctly. Furthermore it was modified in order to permit a direct and fast initialisation of the administrative common block of RANMAR (this is in particular important when processing huge CPROD files).

Anyone who intends to experiment with the code should know that, in general, a) the source code of a routine is contained in a file named like the procedure but in lower case (with the suffix .f, .for, .F), and that b) several program versions might reside in a library. It was common practice to mark an obsolete program version by adding a '0' and a new version by adding a '9' to the file base name. (For example, two versions of the procedure ELGOBS exist which reside in the files source/elgobs.f and source/elgobs0.f, and the routine MCTAGM is contained in jmc/mctagm.F and jmc/mctagm9.F.)

5.2 superv

The Supervisor `superv` handles the output banks of `mcjade` and is also capable of processing real JADE data banks. In case of unsmeared Monte Carlo tracking banks (as delivered by `mcjade`), firstly a smearing simulation is performed, which is then followed by the event reconstruction. The analysis results (Inner Detector tracks, Lead Glass calorimeter clusters, etc.) are written out as binary BOS banks (Section 3.1.1, Appendix A). Some analysis options can be set in the calling script `sv.ksh`, see below.

Usage:

- Input files:
  - The `mcjade` output BOS banks (suffix `.bos`).
  - The random number generator status file `superv.stat` of a previous `superv` run (needed for the smearing simulation). If this file does not exist, the default random number seed is used to initialise the random number generator.
- Output files:
  - A binary BOS file with the result banks (suffix `.bos`).
  - The random number generator status file `superv.stat`.
  - Optional: A control histogram file (suffix `.hist`) and a text file (suffix `.bnk`) with the BOS banks contents in a readable format.
- FFREAD card steering variables: (<...> are placeholders for generic file names or paths.)

SVSTART	ON, OFF	initialisation of the random number generator
event analysis options:		
AUPDAT1	<PATH>/aupdat1.b	standard calibration file
BUPDAT0	<PATH>/bupdat0.b	1. extended calibration file
BUPDAT1	<PATH>/bupdat1.b	2. extended calibration file
		(set *either* AUPDAT1 *or* BUPDAT0 and BUPDAT1)
ZSRFTV	ON, OFF	refitting of tracks / common z vertex fits
		(still not reactivated)
VERTEX	ON, OFF	additional vertex analysis routines
		(still not reactivated)
I/O files:		
MCBOS	<mcjade>.bos	input file to process (BOS format)
FIRST	1,2,...	first event to read from this file
LAST	0,1,...	last event to read from this file
		(0: process until end of file)
SVBOS	<superv>.bos	output file with superv result banks (BOS format)
SVBANK	<superv>.bank	output file with superv result banks
		(ASCII format, only for tests)
BANKS	ON, OFF	print out option for this file

... further variables relevant for tests only ...

Description

The program flow is sketched in the Figures 5-6. At the beginning, the BOS system and various analysis packages are initialised. Further initialisations of HBOOK, FFREAD, the random num-

On call, if INDEX=	Description
0	Initial call, before first event read.
1	Called at the beginning of each new run.
2	Called immediately after event is read.
3	Lead Glass energies have been computed.
4	Fast Z vertex has been found, Jet Chamber Calibration was performed.
5	Inner Detector pattern recognition has been run.
6	Energy clusters in the Lead Glass have been found.
7	Inner Detector tracks and clusters have been associated.
8	Unused.
9	Muon analysis has been done (not supported).
*10	If last event is read, set INDEX = 13 on return.
*20	Called immediately after event is read but before Inner Detector smearing is done.
100	Called before end of job.

On return, if INDEX=	Description
1	The current event will be dropped, and next one will be read.
11	The event will be written out, and a new one will be read.
12	The job will be terminated normally.
*13	The event will be written out, and the job will be terminated normally.
else	Go to the level given by INDEX.

Table 2: The meaning of the Supervisor steering index. New level indices are marked with *.

ber generator, and user specific demands are performed by the steering routine `USER(user.f)` called repeatedly at various stages of the event analysis. The standard JADE block data residing in `jadegs/jadebd.F` are linked in the same way as in `MCJADE`.

The event loop starts with reading the event from the BOS file. If data of Monte Carlo type is detected, the program extracts the geometrical constants previously used in the tracking process (these are passed through special calibration and status records at the beginning of each MC file) and overwrites the default block data settings. Then the program follows with calling a set of subroutines performing a supplementary simulation of the Inner Detector resolution and efficiencies. Similar to the tracking simulation, the current random number seeds are passed between two consecutive Supervisor runs using a status file (`superv.stat`). Usually, Monte Carlo events must be marked at generation stage with run numbers < 100 (as is by default the case in `MCJADE`). This information and further important markers relevant for event reconstruction are supplied by the general information bank `HEAD` [28].

The program continues with a printout of detector status information, the smearing constants (if appropriate), the trigger conditions for the current run, the MC tracking options chosen previously, and information about various analysis options or software packages used for the event reconstruction. The event analysis chain following is outlined in Table 2. At various points in the program, control is passed to the `USER` routine which can make decisions about

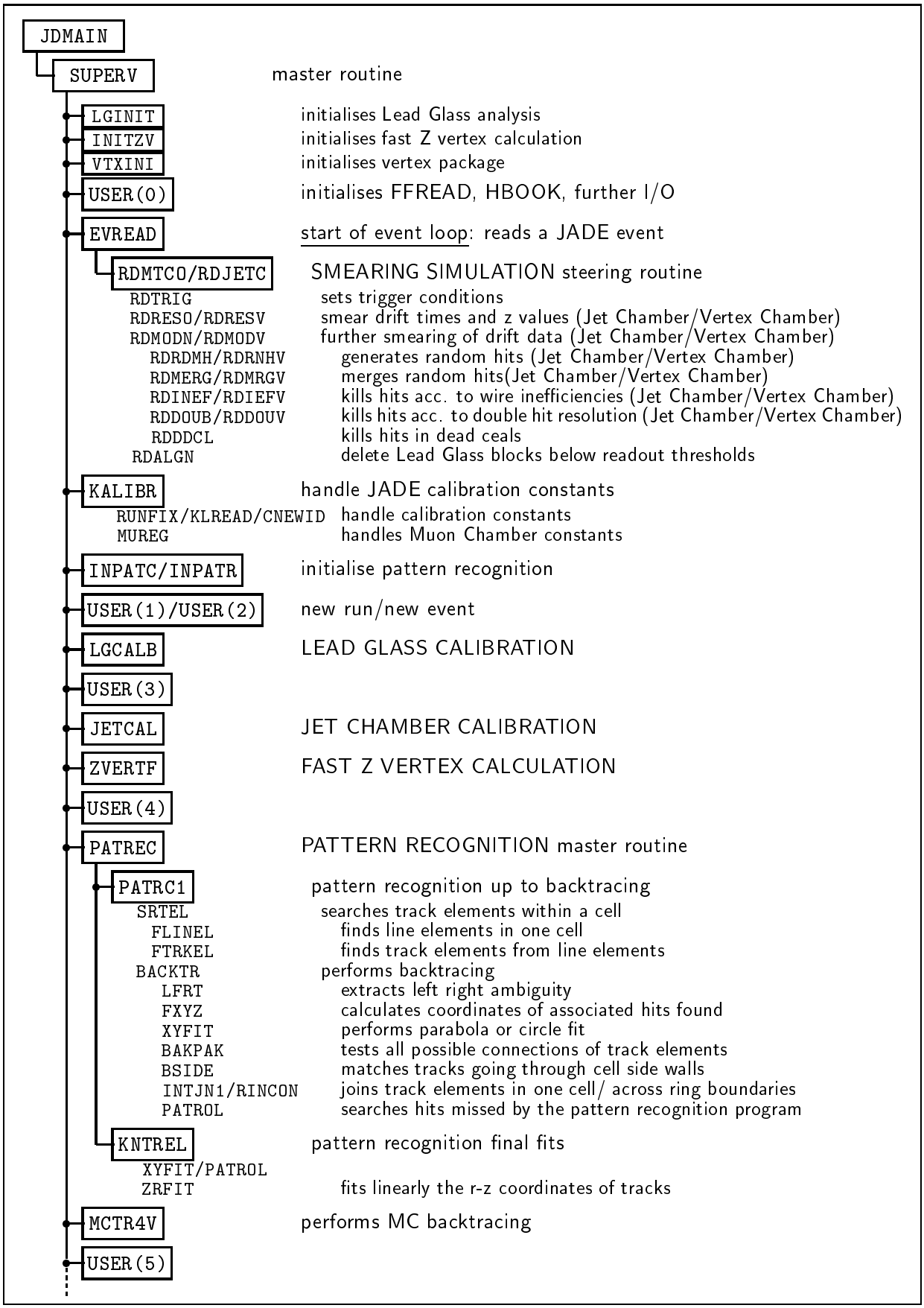


Figure 5: Schematic overview of SUPERV subroutines (part 1).

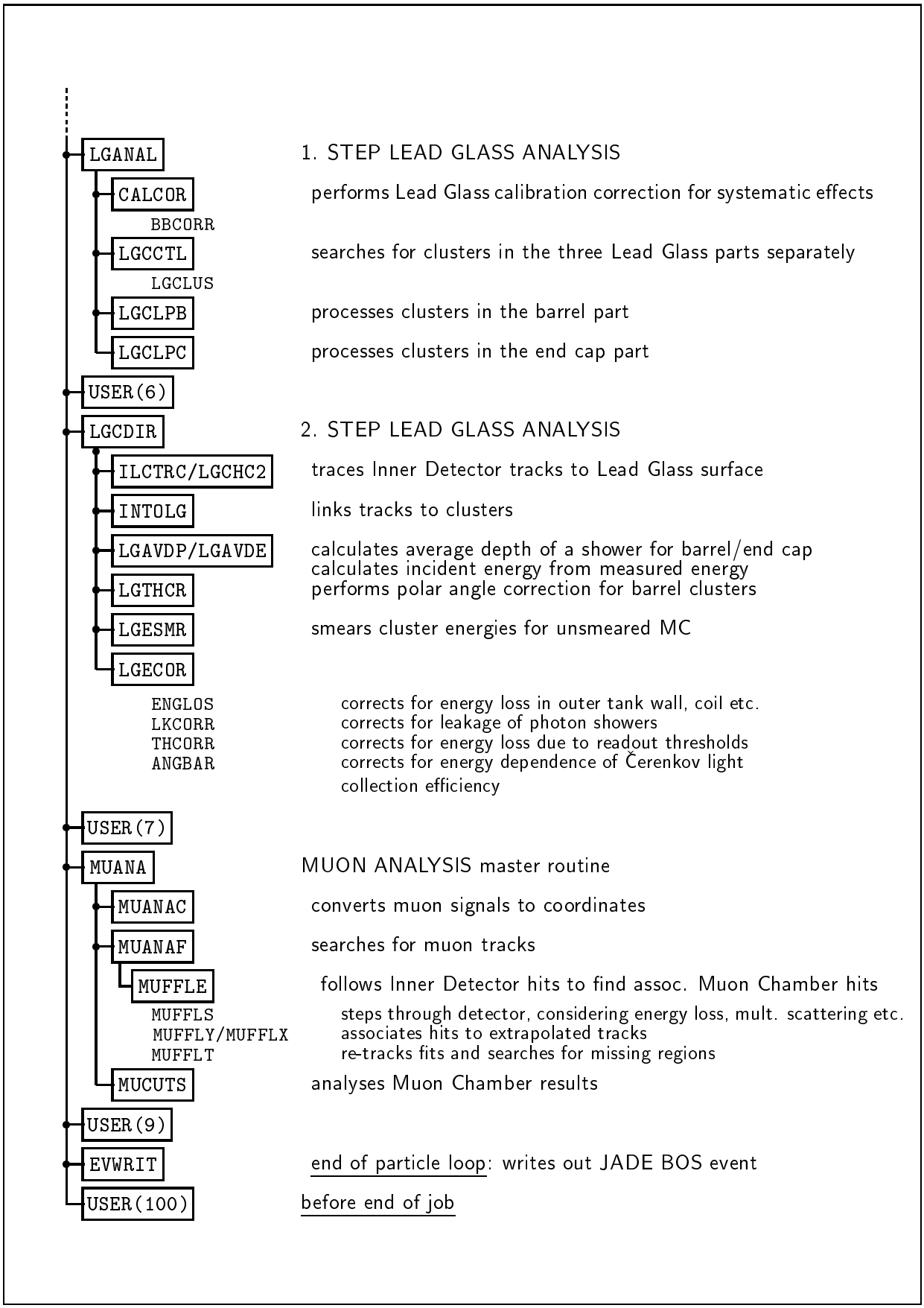


Figure 6: Overview of SUPERV subroutines (part 2).

what to do next: At each call, an INDEX value is passed to USER to indicate at which point or level the Supervisor has just finished. Thus it is possible to perform individual analysis steps based on the event processing made so far. On return, the INDEX value usually is incremented before passing the control back to the Supervisor, in order to continue with the analysis. By setting INDEX appropriately, it is also possible to skip or repeat a level, or to skip the current event. The meaning of the INDEX values is summarised in Table 2.

If a read event is the first event of a new run, the calibration constants are read from either the aupdat or the bupdat files (Section 3.4) and fixed for the current run: The raw data bank ALGL is subject of the Lead Glass calibration which automatically is performed if the bank ALGN is not present in the current event record (Section 3.1.1). The calibration routine for the Jet Chamber acts on the raw data bank JETC, with the converted data stored in the same bank JETC but with a different bank number. For Monte Carlo data, the Jet Chamber and Lead Glass calibration is not needed.

Then, a fast Z vertex calculation is performed which is merely based on Jet Chamber hit coordinates located in the JETC bank and not on fully reconstructed tracks. The results are written in the bank ZVTX. Thereafter, the pattern recognition algorithms are called if no PATR bank was found in the event record at this stage of the analysis (except for the special bank PATR/12 tentatively filled by the preceding tracking program). The tracks found and the corresponding fit results are stored in the banks PATR/10 and JHTL/10 (Section 3.1.1).

The Lead Glass analysis is subdivided into two stages. In the first step, clusters are searched for, and cluster energies are computed, thus creating the result bank LGCL. In the second step, the clusters are checked for association with the Inner Detector tracks, and various energy corrections are performed. The bank LGCL is partly overwritten with the new results.

Finally, the Muon Chamber analysis is performed with the results stored in MUR1 and MUR2. As already mentioned, the Muon analysis is unfortunately not relevant in case of Monte Carlo events generated by the present tracking programs.

For further detailed information about the analysis check Section 2.2 and references therein. Note that in order to repeat a Supervisor level, one has not only to set the USER routine INDEX value appropriately but also to delete the corresponding result banks mentioned above.

Remarks

No serious bugs were found in the program as yet. Various error messages are printed out during a Supervisor session which (to my best knowledge) can be regarded as harmless.

- The most frequently produced error message is related to subroutine PATROL(patrec/patrol.F) which searches and records hits missed by the pattern recognition programs. If sufficient new hits are found, a refit is made, and PATROL is recalled again. In approximately every 50th event, PATROL returns the error code 8, which means that too many refits are done. In this case, the routine stops and returns those hits already found. This observation may be related to the higher computing precision of the current platform.
- The routine ILCTRC(source/ilctrc.f) which is responsible for the extrapolation of Inner Detector tracks to the Lead Glass surface finds sometimes a track originating from outside the Inner Detector.

- The general copy routine `MVCL(interface/bitbyte.f)` often prints a warning message when it operates on empty BOS banks.
- Note that negative event numbers may appear in a Supervisor error messages since the `HEAD` bank considers only a 16 bit integer word to store event numbers.

Some attention should be paid to the following:

- It is desirable and straight forward to implement the track fitting package `ZSRFTV` [48] (`zsrftv.F` and further files located in `jadegs/`) and the vertex analysis package `VERTEX` [60] (see `wertex/` and `vertex/`) in the Supervisor. An appropriate place for doing this is the `USER` routine when called with `INDEX=5`, i.e. after the standard pattern recognition programs have finished. Corresponding switches for enabling/disabling these additional analysis steps are already foreseen in the steering shell script `sv.ksh`.
- The random number seeds are handled by the same subroutine `MCRAND(mcjade/mcrand.f)` used by the tracking simulation. New `USER` level indices `20` (on call) and `13` (on return) were introduced in order ensure a correct bookkeeping of the random number generator status.
- Further utility routines/functions were introduced intended to be used within `USER` subroutine.
  - The file `superv/ana.F` offers a set of subroutines for `INDEX` dependent histogramming of detector observables at various Supervisor levels.
  - The file `superv/showb.f` contains routines for dumping BOS banks in a readable form. This utility is valuable for various checks.

5.3 ze4v

The main task of the program is to compress the JADE BOS events into the handy ZE4V records. Besides the I/O controls, there are some few options and switches relevant for the various additional analysis steps mentioned in Section 2.3. These can be set by the calling shell script `ze.ksh`, see below.

Usage:

- Input files:
  - The `superv` output BOS banks (suffix `.bos`).
  - The random number generator status file `ze4v.stat` of a previous `ze4v` run (needed e.g. for the dE/dx simulation). If this file does not exist, the default random number seed is used to initialise the random number generator.
- Output files:
  - A binary BOS file with the `ze4v` banks (suffix `.bos`).
  - The random number generator status file `ze4v.stat`.

FFREAD card steering variables:

ZSTART	ON, OFF	initialisation of the random number generator
I/O files:		
SVBOS	<superv>.bos	input file with the JADE events in BOS format
EVTREAD	0,1...	max. number of events to read from this file (0: process until end of file)
EVTSCP	0,1...	initial events to skip
ZE4V	<ze4v>.ze4v	output ZE4V data file (BOS format)
ZE4VOUT	0,1...	max. number of events to write out into this file (0: write out until end of reading the input file)
ZE4VPRT	1,...30,...	max. number of readable ZE4V bank prints
ZBOS	<ze4v>.bos	output file with the full JADE BOS events
EVTOUT	1,...50,...	max. number of full BOS events to write into this file (0: write out until end of reading input file)
ZBANK	<ze4v>.bnk	output file with JADE BOS events (ASCII format, only for tests)
BANKS	ON, OFF	print out option for this file

... further variables relevant only for tests...
... further analysis options...

Description

A sketch of the program flow is shown in Figure 7. The main program `ZE4VJB` starts with the initialisation of BOS, FFREAD and I/O steering. Various analysis options are printed out. After allocating the BOS files and calibration files, the program checks whether the ZE4V records have to be created from TP banks or from `PATR/LGCL` banks. Within the present software structuring only the latter option is relevant.

The master routine `ZE4VPK(zlib/ze4vpk.for)` extracts the information from the BOS banks and fills successively the header part, the tracks part and cluster part of a ZE4V record. A switch

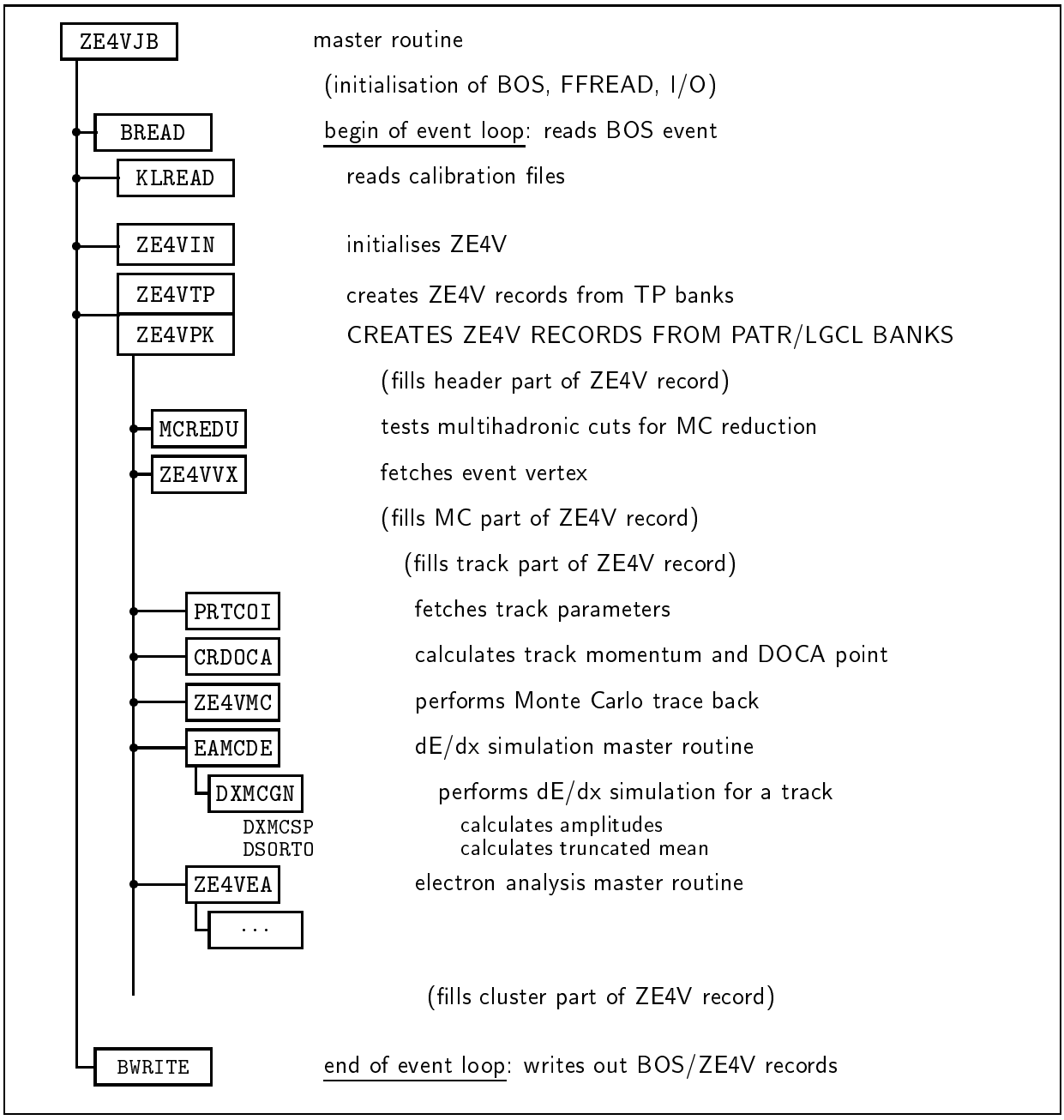


Figure 7: Schematic overview of subroutines called by the present ZE4V version.

variable `IMODE` allows for selective processing of tracks and clusters. By default, all tracks and clusters fulfilling the requirements stated in Section 2.3 are used. Track momenta and DOCA points are calculated using the `PATR` and `HEAD` information.

The events read are checked for multihadronic preselection criteria and marked with a corresponding cut flag (`MCREDU(zlib/zmcredu.f)`). The event vertex is recalculated, thus superseding the fast Z vertex calculation of the preceding Supervisor run. The run dependent radial coordinates are taken from the calibration files, whilst the Z vertex is calculated using fully reconstructed tracks.

In case of Monte Carlo events, the program performs a particle traceback to the initial hadrons and partons based on the information stored in the bank `TR4V`, which has been generated previously by the JADE Supervisor. The traceback results are foreseen to be stored in the header section of the ZE4V bank. For simulated data, also a supplementary simulation of the specific energy losses of tracks is done.

The present program version allows for calling for a software package to analyse electron candidates, thus recalling some of the Supervisor subroutines. But note that neither the analysis results stored in the final BOS banks nor the particle sections of the ZE4V records are overwritten by these additional actions.

The program ends with a printout of various statistics. In addition of writing out pure ZE4V records, the user may output the full JADE events in BOS format. The random number seeds are administrated in the same way like in `MCJADE` and `SUPERV` using a status file (`ze4v.stat`).

Remarks

- It is straight forward to prevent the program from ignoring the previous Z vertex calculation results, as e.g. supplied by the recommended `VERTEX` fitting package (which is desirable to be implemented in the Supervisor).
- It might be desirable to produce other data formats than ZE4V records, e.g. HBOOK NTuple files[11]. To do this, one has to replace/complement the BOS record writing routine `BWRITE` by an appropriate conversion routine.
- The newly generated MC files (Section B) do not contain any event tree information, but the write out option for the Monte Carlo traceback might be easily activated by resetting the variable `NPRONM` (maximum number of partons) in `ZE4VPK` from `0` to a sufficient high value.
- ZE4V events are written out as normal BOS records, each event consisting only on one BOS bank with the name `ZE4V`. However, for the subsequent analyses it is possible and desirable to process ZE4V files without using the BOS framework. An appropriate FORTRAN template for reading ZE4V data is presented in Section 6.

[11] It is aimed [61] to bring the JADE data in the NTuple format currently used by the OPAL collaboration, in order to uniformly access both JADE and OPAL data for the purpose of common QCD studies.

5.4 jadesim

jadesim is a tool for steering the four individual process levels as a whole, i.e. 4-vector generation, tracking simulation, event analysis, and ZE4V formatting. The shell script allows to start and to end the event generation at an arbitrary process level. The entire process flow with the dependencies of the programs on the generated data banks is visualised in Figure 8. The most important steering variables are set in the head of the script and are summarised in Table 3.

Steering variables:

- The variable $ecm fixes both the centre-of-mass energy $\sqrt{s}$ relevant for event generation and the detector configuration date used for the tracking and the smearing simulation. The most common combinations of centre-of-mass energies and corresponding detector configuration considered by $ecm are summarised in Table 4. It is straightforward to extend the script also for more detector configurations.

- The variable $gen specifies the event generator. Various event generation parameters are fixed in FFREAD card templates of which some general entries are edited by jadesim. Up to now, the event generators listed in Table 5 are available for the production of the CPROD 4-vector files.

- The variables $file_gen, $file_mc and $file_sv define the first process level to consider in a jadesim job. Here and in the following, the suffixes '_gen', '_mc', '_sv' and '_ze' refer to the generator step, the mcjade step, the superv step, and the ze4v step, respectively. The variables are arrays whose contents are taken as the locations of the input CPROD or BOS files needed for the next process level. An array may also be empty. Only specified files corresponding to the highest preprocessed level are relevant. For example,

  ```
  set file_gen = ( <genout>.cprod )
  set file_mc  = (  <mcjade>.bos  )
  set file_sv  = (                )
  ```

  causes jadesim to start the Supervisor using <mcjade.bos> as input file. The variable $file_gen is ignored in this case. It is allowed to specify more than one input file, e.g.

  ```
  set file_mc = ( <mcjade>.bos.1 <mcjade>.bos.2  )
  ```

  or

  ```
  set file_mc = ( <mcjade>.bos.*  )
  ```

  which means that all mcjade tracking files with the basename <mcjade>.bos have to be processed successively in a jadesim job. If neither of the arrays are set, the procedure starts with the event generation specified by the variable $gen and the corresponding FFREAD card templates.

- The variable $end defines the last level to process. The values allowed are:

  ```
  $end = 1 — event generator
  $end = 2 — mcjade
  $end = 3 — superv
  $end = 4 — ze4v
  ```

variable	values allowed	meaning
$ecm	14, 22, 35a, 35b, 38, 44	identifier for the c.m.s. energy and detector configuration
$nev_tot		total number of events to generate
$nev_seq		max. number of events to generate per sequence
$start_seq	1,2,...	initial sequence number
$gen	jt63, py57, hw59, hw58d, ar48p, cj623	identifier of the event generator used for 4-vector generation
$file_gen	<genout>.cprod	names of the input files needed for the lowest process level (<...> denote the generic file names with the path).
$file_mc	<mcjade>.bos	
$file_sv	<superv>.bos	
$end	1, 2, 3, 4	number for the highest level to process
$save_gen		saving options for output CPROD and BOS files
$save_mc	no, yes	
$save_sv		
$save_ze		
$sdir		working directory with the output files
$aupdat1		standard calibration file
$bupdat0		1. extended calibration file
$bupdat1		2. extended calibration file

Table 3: The most important steering variables in jadesim.

$ecm	$\sqrt{s}$[GeV]	tracking smearing date
14	14.0	17.07.1981
22	22.0	17.06.1981
35a	34.6	17.05.1982
35b	35.0	17.05.1986
38	38.3	01.10.1985
44	43.8	17.05.1985

Table 4: Common combinations of c.m.s. energies and configuration dates as fixed by the variable $ecm.

$gen	generator	files needed
py57	PYTHIA 5.7	GEN.CARD
hw58d	HERWIG 5.8D	GEN.CARD
hw59	HERWIG 5.9	GEN.CARD
ar48p	ARIADNE 4.08	GEN.CARD
cj623	COJETS 6.23	GEN.CARD cj623inp.dat cj623tab.dat
jt63	JETSET 6.3	GEN_JT63.CARD

Table 5: Event generators available for 4-vector production in CPROD format.

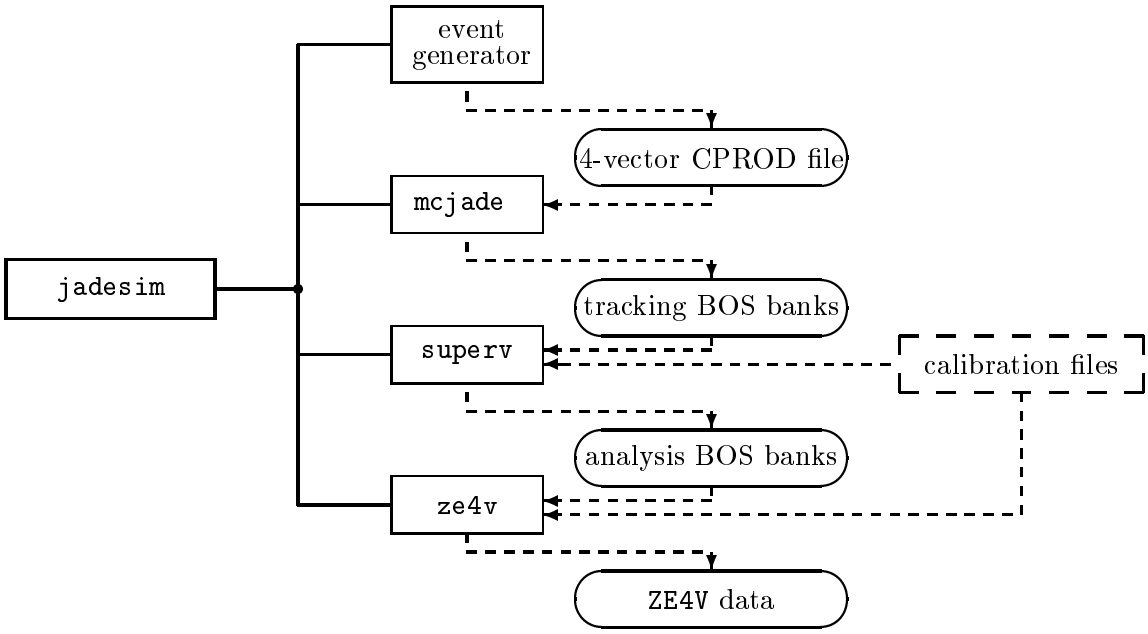


Figure 8: Schematic illustration of the process levels steered by `jadesim`.

- The variable `$nev_seq` allows to split the whole generation process with a total number of `$nev_tot` events into sequences generating subsamples with a maximum number `$nev_seq` of events. This is useful to avoid huge single data files (in particular the Supervisor result BOS banks).

- The variable `$start_seq` is the initial sequence number which basically acts as a book keeping parameter needed for labeling the output files of a sequence. Setting `start_seq=1` forces the simulation to use the default random number seeds. Otherwise, the last entries of the status files

  ```
  mcrnset.dat mcjade.stat superv.stat ze4v.stat
  ```

  (if existing) are used for the initialisation of the random number generator of the corresponding subprocesses.

- Finally, the variables `$save_gen`, `$save_mc`, `$save_sv` and `$save_ze` are memory freeing options for the respective process levels. If the value of a variable is "`yes`", the corresponding generated output files are kept, otherwise they are removed immediately at the end of a processing sequence. E.g.,

  ```
  set file_gen = no
  set file_mc  = yes
  set file_sv  = no
  set file_ze  = yes
  ```

  keeps all tracking simulation files (e.g. in order to run the Supervisor again on the same tracking files but with different smearing conditions), and also the ZE4V data needed for further analyses.

Output files:

A `jadesim` run creates the subdirectories `gen/`, `mc/`, `sv/` and `ze/` containing the output files of the corresponding active process levels. The file basenames have the generic form

```
gen/<generator type>_<c.m.s energy>
mc/mc_<generator type>_<c.m.s energy>_<configuration date>
sv/sv_<generator type>_<c.m.s energy>_<configuration date>
ze/ze_<generator type>_<c.m.s energy>_<configuration date>
```

with the `<...>` brackets denoting identifiers for the various parameter settings of the jobs. Various suffixes are appended to the file names which have the following meaning:

`.cprod` — 4-vector CPROD file
`.bos` — binary BOS files (Section 3.1.1)
`.ze4v` — ZE4V files (Section 3.1.2)
`.log` — `mcjade/superv/ze4v` program log files
`.bnk` — BOS banks in readable format (memory consuming, suitable for tests only)
`.hist` — `mcjade/superv` control HBOOK histogram files (only for tests)

In addition, the running `jadesim` sequence numbers are appended to the file names.

It is foreseen to run `jadesim` not only interactively but also in batch mode[12].

[12] Unfortunately, the IBM Job Management System "LoadLeveler" used for doing this on the AIX cluster of the MPI is not supported any longer.

5.5 jadez

`jadez` starts the interactive JADE graphics session. The recommended working directory for doing this is `job/jadez/` where some useful files are located:

`higz_windows.dat`	—	`HIGZ` window display attributes.
`F11OLS.GRAPHICS.PROFILE.MACROS`	—	predefined macros for the `JADEZ` command interpreter.
`F11LH0.AUPDAT1`	—	soft link to the standard JADE calibration file (Section 3.4).

Usage

1. Before starting the session it might be desirable to adjust the HIGZ windows attributes given in `higz_windows.dat`.
2. Type '`jadez`' to start the program.
3. The program then requests the data file with the full JADE events in BOS format to be viewed: answer by entering e.g. the path of a `mcjade` or `superv` result bank file.
4. If the allocation was successful, the calibration files are requested. Simply press ENTER for the default calibration file (`aupdat1.b`) linked by `F11LH0.AUPDAT1`, or type in the file names of the extended calibration files (`bupdat0.b` and `bupdat1.b`).

   Thereafter, the first BOS record of the specified file with JADE events is read and graphically displayed in a HIGZ window. In addition, various Supervisor information is printed out as plain text on the terminal where `jadez` has been started. The graphics programs shows the default view of the JADE detector as described in Section 2.4.
5. The command line interpreter prompts with "`--->`" and waits for further input, i.e. a command or a macro name. You can also type in a 1-line sequence of commands separated by semi-colon. Most of the available commands are listed and briefly explained in Appendix C. Further descriptions can be found in `job/jadez/HELP` and in [24].
6. Type 'STOP' to end the graphics session and to output the Supervisor statistics.

The following illustrates some commands useful to steer the graphics session:

If the events already have passed the track pattern recognition and the Lead Glass analysis, than it is sufficient to know that the command 'N' fetches the next event, or that the command 'FIND' fetches an event by run and event number.

In case that raw data or MC data are read, the JADE Supervisor analysis actions can be performed. Similar to the Supervisor (Section 5.2), control is passed to a `USER` analysis steering routine. At each analysis level, stop flags can be set with the 'LEVELS' command. This allows to cause a halt of the analysis at a level of interest and to view the results so far. By default, the stop flags are set at level 2 (that is, before any analysis is attempted) and 6 (after pattern recognition and Lead Glass cluster analysis are performed) are set. The command 'C' takes you from one stop level to the next stop level (see Table 2 for details). Note that for unsmeared

Monte Carlo data the smearing simulation is appended immediately after the event has been read.

For a graphical representation of measured quantities in the Inner Detector and the Lead Glass, you can type 'RA', 'RB' or 'RC' for a r-ϕ view and ZXA, ZXB or ZXC for a z-x view (the final letters 'A', 'B', 'C' refer to different sections and subsystems to be displayed). The standard view automatically displayed after reading an event or after proceeding to the next active Supervisor level is by default given by the first two entries of the macro file `F11OLS.GRAPHICS.PROFILE.MACROS` and can be changed using the command 'CSTV'. At each analysis level, the detector hardware may be displayed on top of the current event view by typing the 'DET' command. Projection views of the event can be superimposed with the 'PRO' command.

The commands 'RES' and the 'VRES' recall the main event reconstruction results corresponding to the current Supervisor level. Various other commands exist for performing further analysis steps or displaying detailed analysis details.

The command 'TRUE' superimposes the true Monte Carlo four vector input onto the current view. The 'ZOOM' command allows to scale a section of the current view marked with the mouse.

To save the current view of the event for a later print out, the command 'H' can be used to generate a PostScript file. The generic file name is `JADE.<nnn>.eps`, where `<nnn>` is a counter starting from `001` and incremented each time the command 'H' is entered.

Description

The main program `GPHMAIN(jadez/gphmain.F)` initialises the PLOT-10 graphics emulation and the default graphics setup, allocates the BOS data sets and the calibration files. The program calls the standard JADE Supervisor `SUPERV(jadegs/superv.F)` with the actions already described in Section 5.2.

User control is passed to a `USER` routine (`jadez/xuser.F`) adapted for the additional demands of the JADE graphics. After processing an active Supervisor stop level, `USER` calls the master routine `SCANNR(grafix/scannr.F)` which is the master routine for the interactive graphics control. It steers the event and detector hardware display, the graphical representation of the analysis results, and the handling of the keyboard and mouse control. `SCANNR` also handles additional analysis steps requested by user commands.

Those interactive actions affecting only the graphical representation of the event are passed through a separate display routine `DISPLY(graphix/disply.F)`. Other commands causing additional analysis actions are handled by `SCANNR` itself. Main routines called by `SCANNR` and `DISPLY` are

`KOMMAN (grafix/komman.F):` the command interpreter,

`EVDISP (grafix/evdisp.F):` the master routine for displaying a event,

`JADISP (grafix/jadisp.F):` the master routine for drawing the JADE detector hardware,

`RSDISP (grafix/rsdisp.F):` the master routine for drawing the analysis results.

See the files cited for further detailed documentation.

Remarks

- Almost all graphics commands work properly and do the requested actions. However, a few experts commands listed in [24] (e.g. 'EDIT') do not work error-free as yet since they were not properly adapted here (for time reasons).
- `aupdat1.b` is the standard calibration file for the graphics session. The extended calibration files `bupdat0.b` and `bupdat1.b` are only needed in case of the recalibration of the Lead Glass.
- After the generation of a PostScript printout of a JADE event using the 'H' command, the picture shown on the HIGZ window must be rebuild at least once for the next printout to get the complete graphics.
- Note that the detector hardware display colours on the HIGZ screen (white lines, dark background) appear intentionally reversed in the PostScript representation.
- Various calls to the new subroutine `SETCOL(interface/setcol.F)` were implemented in the program code in order to introduce colours in the graphics display. The display attributes (colours and line widths) of detector parts or analysis results can easily be changed in the block data subroutine `CPLTCOL(interface/setcol.F)`.

6 Utility Programs

The directory `util/` contains the following tools: an MC package to generate QCD events suited for further processing by the JADE detector simulation, a simple FORTRAN template to read and process ZE4V data, and a program to convert the JADE calibration files into binary format.

6.1 QCD Event Generator Package and CPROD Interface

The event generator code and various driver routines were taken from the QCD group of the OPAL collaboration, namely the MC package version 105 [62]. Basically, the following modifications were performed which are based on this program version: 1) The code was slightly adapted for additional use on AIX platforms, 2) the former JETSET 6.3 generator was implemented, and 3) a CPROD interface needed for the tracking simulation `MCJADE` was build in.

Resources

The software is located in `util/mcgen/` and is structured as follows:

`src/`: contains event generator and general analysis source code.

PYTHIA 5.7/JETSET 7.4 [63,64]	—	`pythia5722.car jt74opal.car pymaxi.car`
JETSET 6.3 [65,66]	—	`jt63jade.car`
ARIADNE 4.08 [67]	—	`ariadne408.car`
HERWIG 5.9, HERWIG 5.8D [68]	—	`herwig58.car herwig58d2.car`
COJETS 6.23 [69]	—	`cj623d.car`
general analysis packages	—	`px114.car ckern105.car`

`mc/`: contains the modified OPAL MC package `mc105j_lib.car` (includes various QCD analyses routines)

`pythia/`: contains the PYTHIA job script `py57.ksh`.

`jetset/`: contains the JETSET job scripts `jt63.ksh` and `jt74.ksh`.

`ariadne/`: contains the ARIADNE job script `ar408p.ksh`.

`herwig/`: contains the HERWIG job scripts `hw58d.ksh` and `hw59.ksh`.

`cojets/`: contains the COJETS job script `cj623.ksh`.

`lepton/`: contains a simple lepton pair generator and a corresponding job script.

`Plib/`, `Pbin/`: contain the installed libraries and executables (Linux platform)

`Rlib/`, `Rbin/`: contain the installed libraries and executables (AIX platform)

Installation

The software is handled by the PATCHY source code management system[13] provided by the CERN library. The compilation is supported on both Linux and AIX environment, for which the following installation tools are available:

- `mcgen/src/Inst.ksh` installs the MC libraries.
- `mcgen/mc/mc105j_lib.ksh` compiles various general MC handling subroutines.
- The executable for an event generator job can be generated by the corresponding job shell scripts (`py57.ksh`, `jt74.ksh`, ...) with the argument "`comp`" passed to the script.
- `mcgen/Install.ksh` is a simple tool to perform overall installation.

Usage

To start an event generator, run one of the corresponding job scripts mentioned above but without passing any arguments. All of these scripts are structured similarly into a compilation part and a program execution part that includes a build in FFREAD I/O handling. The event generation can be steered by various shell script variables and FFREAD card entries:

`$MCSTART`	`ON`, `OFF`	switch for random number generator initialisation
`$MC4VEC`	`ON`, `OFF`	switch for 4-vector file generation
`$NAME`		basename of the output files
`MCEVFILE`	`$NAME.<suff>`	name of the 4-vector output file `<suff>=cprod`: generate CPROD file otherwise: generate readable text files (for tests)
`MCNEVT`		number of events to generate
`MCECMS`		center of mass energy
`MCISR`	`ON`, `OFF`	switch for initial photon radiation
... MC modelling parameters ...		
... MC analyses options ...		

Examine the scripts for further explanations. The random number seeds of a job are stored in the files `mcrnset.dat` and `mcrncnt.dat`. Set `MCSTART=OFF` to force the program to use these files to initialise the event number generator for the next run.

By default, the following files are generated:

`$NAME.cprod` — 4-vector CPROD file
`$NAME.hist` — MC analysis histogram file

The output CPROD file is binary and thus platform dependent, but note that the tracking program `MCJADE` was modified in order to accept files with both big endian and little endian schemes.

[13] The antiquated PATCHY structure of the original software was kept unchanged as far as possible. It would be desirable to manage the programs with a modern code management system like e.g. CVS.

6.2 Example Program for Processing ZE4V Data

The directory `util/zread/` contains a simple self-explanatory FORTRAN template `zread.f` to read both ASCII formatted and binary ZE4V files. In case of binary ZE4V, the program is capable of processing both big endian and little endian schemes. The script `zread.ksh` can be used for compilation.

6.3 Conversion Tool for the JADE Calibration Files

The directory `util/ccal/` provides a FORTRAN program `conv.f` for converting the ASCII formatted calibration files `aupdat1`, `bupdat0` and `bupdat1` (located in `jadesoft/cal/`) into the binary files needed for the JADE Supervisor. Since the reading of the binary calibration files is dependent on the endian scheme as yet, one has to perform the conversion on the same platform where the JADE Supervisor is intended to run. The program `rconv.f` can be used to reconvert the binary file for test purposes and to check the entries of specified constants records. The script `ccal.ksh` performs the compilation.

7 The Reactivation of the JADE Software

This section summarises the various technical details concerning the adaption of the JADE software on current computer platforms. The typical problems described in the following may be helpful for further code changes or code reactivation intended.

7.1 Original Source Code

The JADE code consists of a mixture of different FORTRAN language standards, precompiler languages, and IBM/370 assembler code. Since the beginning of the software development (with the oldest code fragments dated in 1974) FORTRAN IV was the primary programming language, but later on, more and more programs were written in the better readable SHELTRAN and FORTRAN 77 languages. Thus the code generally became more and more transparent.

The major part of the JADE library source code as well as the whole BOS library is written in FORTRAN IV. A considerable part of the Inner Detector pattern recognition software was developed in SHELTRAN which allows for some kind of "structured" FORTRAN programming. A similar precompiler language called MORTRAN was used to program a few graphics and vertex analysis routines. Various analysis routines developed in the last stage of the JADE experiment, in particular the ZE4V package and the TP9 program, are based on standard and extended FORTRAN 77. Various frequently called time consuming procedures stemming from former DESY and CERN libraries are written in IBM assembler. These are e.g. fast copy routines for moving data banks, and various procedures for binary operations on data words.

The FORTRAN code is full of old-fashioned and nowadays deprecated features, like the copious use of `ENTRY` and alternate `RETURN` statements, arithmetic `IF` branches, `ASSIGN` and assigned `GO TO` statements. Apart from a few exceptions, implicit variable declaration rules for variables are thoroughly applied. Half word (16 bit) integers are implicitly assigned to variable names beginning with `H`.

Generally, a compact programming style was applied throughout major parts of the software in order to save processing and memory time, thus making it difficult to understand the processing flow.

7.2 Tasks

Parts of the code were modified and tested using various compilers on different platforms (DEC alpha OSF1, SGI mips IRIX, HP-UX system). Finally the code was optimised for the XLF compiler[14] supplied by IBM for use with the AIX operating systems and RS/6000 architectures. It supports the full FORTRAN 77 language standard but also various IBM and industry extensions to the FORTRAN language.

In order to revive the programs described in Section 5, the following objectives were considered:

- The translation of the SHELTRAN and MORTRAN code into standard FORTRAN 77 language.
- The emulation of parts of the DESY library [70].
- The emulation of IBM/370 FORTRAN intrinsic functions.
- The emulation of parts of the Tektronix PLOT-10 terminal control system [71] and of various routines of the terminal interface package IPS [72].
- The removal of various platform dependencies.
- Some code modifications necessary to remove inconsistencies or to suppress residual compiling errors or warning messages.

7.3 Handling of the Precompiler Code

The translation of the SHELTRAN code to FORTRAN 77 was performed using the precompiler `sheltran` from the Gronigen University [73][15]. The procedures affected are mainly pattern recognition programs in `src0/patrecsr/` and further track fitting routines in `src0/jadegs/`. In various cases it was necessary to modify some SHELTRAN constructs which are forbidden by the precompiler version used here (e.g. `IF (...) XFOR` statements).

The MORTRAN code found was translated to FORTRAN by hand, since only a few subroutines are affected. These are vertex finding routines modified for use with the Vertex Chamber hardware (`src0/vertex.s/vertex.s.seq`) which are only called within framework of the graphics package.

7.4 Software Interfaces

Interfaces were programmed in order to emulate the functionality of obsolete software packages as far as they really are needed by the JADE software. The emulation software is located in the directory `src/interface/`.

[14] See `http://www.ibm.com/software/awdtools/fortran/xlfortran/library/` for further information.

[15] Unfortunately, the whole precompilation procedure (performed at the very beginning of the software resurrection) is not automated in the present make file for the JADE library as yet.

7.4.1 Former DESY Library and IBM/370 FORTRAN Intrinsics

An emulation of various former DESYLIB procedures and IBM/370 specific FORTRAN functions [70] reside in

```
biby.f dlib.f
```

The files contain mostly mathematical functions, bit and byte handling procedures and copy routines. They are partially available as "fast" IBM assembler code (e.g. `src0/jmc/mvcl.assembler`, `src0/jadegs/mvb.assembler`). Some of the affected procedures are simply dummy routines now, other are interfaces to appropriate CERN library routines [74]. The bit and byte manipulation routines are of particular importance since the JADE software makes good use of it in order perform data access and manipulation. Generally, the corresponding functions were ported to FORTRAN and mapped directly onto the CERNLIB package BITBYT (M421).

Remarks

- The functionality of the DESYLIB random number generators `RN` and `FNORM` is now reproduced by the CERNLIB subroutines `RANMAR` and `RNORML` from the MATHLIB package (V113). Note that a modified version of `RANMAR` (`interface/ranmar.F`) is used, see Section 5.1.

- The intrinsic type declarations of various IBM/370 specific functions are not valid for the XLF compiler used here. Hence, to prevent e.g. the DESYLIB routines `TBIT` (logical) and `SHIFTL` (integer) to be treated implicitly as real functions, an explicit type declaration in the calling routine (`LOGICAL TBIT`) or a renaming of the function name (`SHFTL` → `ISHFTL`) (such that the proper implicit declaration rules apply) is needed.

- The JADE software copiously uses the former assembler routine `MVCL` to transfer a given number of consecutive bytes between data arrays. The present emulation is an interface to the CERNLIB subroutine `CBYT`. The interface is capable of accessing both half word and full word arrays at arbitrary byte addresses.

  Accessing half word arrays via `MVCL` might lead to a misalignment of data words (i.e. the memory addresses are not divisible by 4; this causes either a bus error or an unaligned access interrupt). A modified version `MVCL2` was introduced for these few cases. Avoiding unaligned data access was a general issue also for various other emulation routines.

- Some routines exist in two versions which only differ with respect to their address ranging capability (`MVCL`/`MVC`, `SETSL`/`SETS`). This distinction is obsolete in the present emulation, hence the small range versions were mapped onto the corresponding long range versions.

- For the emulation one has to consider that the arguments of various routines may be data types of different word lengths, i.e. either halfwords (16 bit integers) or full words (32 bit integers) or even a mixture of both. In order to prevent the affected routines from returning results which depend on the endian convention of the current computer platform, the arguments passed to an emulation routine must be rearranged appropriately before passing them to the CERNLIB function.

  For example: The former generic function `LOR` performs a logical OR operation of two bit patterns which are provided by two arguments. For the present emulation, a full word

(`LOR(IX,IY)`, `IX`, `IY` being 32 bit integers) and a half word version (`HLOR(HX,HY)`, `HX`, `HY` being 16 bit integers) of this routine was introduced. The interface directly maps the full integer version `LOR` directly onto the CERNLIB function `MBYTOR`. The half word version `HLOR` also calls `MBYTOR` but with reversed half word arrangement, if needed, depending on the endian convention of the current platform. The result of the operation is in turn a half integer word (note that `MBYTOR` is a full integer function and always requires full integer arguments). As a third case, the generic `LOR` may be called with arguments of mixed sizes. It was found that in this case calling `HLOR` with previously converting the full integer to a half integer argument lead to well-defined results (since in all observed cases the full integer argument is < 65536).

Alternative and possibly more elegant ways of handling these data manipulations might exist, but there is no way to circumvent an explicit consideration of the byte ordering at least at one point of the emulation routine. The emulation checks the endian format of the current platform using the routine `MACHINE (interface/biby.f)`. Note that the BITBYT package itself is platform independent. See Section 7.4.3 for further remarks.

7.4.2 PLOT-10 Terminal Control System

The JADE interactive graphics is mainly based on the PLOT-10 software package [71] to generate graphics on Tektronix storage-tube display terminals, and on some routines provided by the interface software for IPS[16] terminals [72]. The philosophy of PLOT-10 is, as the name suggest, to generate graphics in the same manner as drawing objects with pencil on a sheet or using a plotter. The user may work in an arbitrary user defined "virtual coordinate" system or in the "screen coordinate" system. Subroutine calls are available to transform the contents of the user window into a screen window. Various subroutines exist to generate lines and points, for drawing segments in polar coordinates and for adding structured text to the graphics. In the present emulation, the PLOT-10 subroutines were mapped onto appropriate HIGZ[17] routines [75]. The interface is contained in the files

`plot10.F ips.f setcol.F cplot10.for cpltcol.for`.

Some features of the emulation are summarised in the following:

- The status of the PLOT-10 graphics, in particular the current coordinates of the "display terminal beam", is administrated by the common block `/PLOT10/(cplot10.for)`. All PLOT-10 emulation subroutines communicate with each other via this common block. The "beam" position is affected by most graphics actions (including text output) and is permanently updated in `/PLOT10/`.
- Further graphics features (background and line colours, line widths) which were newly introduced in the present emulation are set by the subroutine `SETCOL(setcol.F)`. Calls to this subroutine are now implemented in various `JADEZ` display routines in order to provide coloured graphics. The additional graphics attributes are passed through the subroutines via the common block `/PLTCOL/ (cpltcol.for)`.

[16] Interactive Plotting System
[17] High Level Interface to Graphics and ZEBRA

- The former terminal initialisation routine `INITT` has now the job to initialise ZEBRA, the HIGZ package and the PAW storage (`IGSTRT(plot10.F)`). The default graphics attributes and colours are set, and the PostScript file and metafile control needed for a later printout are activated (`PSOPEN(plot10.F)`). The subroutine `FINITT` ends the graphics session, i.e. it terminates HIGZ (`IGFIN(plot10.F)`).

- The former definitions of the virtual window (`DWINDO`) and the screen window (`TWINDO`) are more or less interfaced to the HIGZ subroutines `ISWN` (performs normalisation transformation window definition) and `ISVP` (performs normalisation transformation viewport definition). The virtual coordinates are in floating point representation, the screen coordinates are in 4096×4096 integer raster units.

- Subroutines exist for drawing solid and dashed lines, points, and for doing (invisible) move operations. Each operation can be performed in virtual or screen coordinate space. Moreover, each graphic action can be specified using absolute or relative coordinates with respect to the current "display terminal beam" position.

  Actions in absolute screen coordinates are supplied by the subroutines

  `DRWABS` *(line)*, `PNTABS` *(point)*, `DSHABS` *(dash)*, and `MOVABS` *(move)*.

  Each graphic action can be performed in four different ways, e.g.

  `DRWABS/DRWREL` *(draws line in absolute/relative screen coordinates)*

  `DRAWA/DRAWR` *(draws line in absolute/relative virtual coordinates)*

  All drawing actions are emulated by a central subroutine `DRAW(plot10.F)`, which is basically an interface to the HIGZ drawing routines `IPL` (draws lines) and `IPM` (draws markers).

  PLOT-10 provides the subroutines `RSCALE` and `RROTAT` to scale or to rotate the coordinate system, respectively, prior to calling the drawing routines with relative arguments. Within the emulation, the corresponding transformation parameters are fixed by these routines and then passed to the subroutine `DRAW` via a central PLOT-10 steering common block `/PLOT10/`.

- The various subroutines to output single alphanumeric characters (`ANCHO`) or character strings (`ANSTR`, `EOUTST`) to the graphics screen are basically mapped onto the HIGZ subroutine `ITX` and appendant graphics attribute routines. This also applies for the former software character generator routines (`SYSSYM`, `USRSYM`) which allow for adding text of arbitrary size and orientation in the graphics.

  Additional terminal I/O routines (`TRMIN`, `TRMOUT`) exist which originally were intended to transmit character strings between the terminal and the graphics application avoiding FORTRAN I/O. In the emulation, these commands are simply rebuilt using FORTRAN `READ` and `WRITE` statements, i.e. these commands generate I/O streams on the X terminal from where `jadez` was started, and not on the HIGZ screen. This solution was found to be generally more appropriate for the purposes of the interactive JADE graphics.

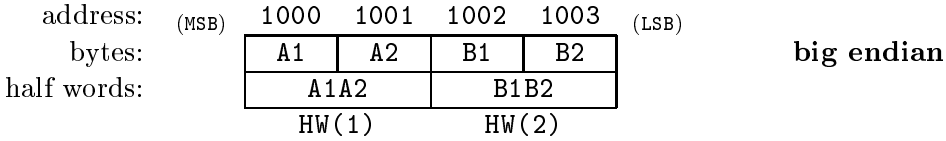
- PLOT-10 allowed for fetching screen or virtual coordinates, respectively, via a graphic cursor using the routines `SCURSR` and `VCURSR`. In the emulation, the joystick actions used formerly to flash a hair cross on the screen are replaced by appropriate mouse actions, with the cursor position provided by the HIGZ subroutine `IGLOC`.

- Some subroutines formerly used for hardcopy printout (`HDCOPY`, `HDCEXT`, `HDCDST`) have now the function to output the current HIGZ display as a PostScript file. This is done by calling `PSCLOS(plot10.F)` which deactivates the metafile and closes the currently open PostScript file. A subsequent call of `PSOPEN` makes the next graphics display again available for PostScript printout.
- The IPS subroutines `GETPDD` and `FREEDD` were formerly used to allocate and deallocate data sets. The functionality of these subroutines was emulated using FORTRAN I/O statements including an appropriate handling of error codes and administration of file names.

7.4.3 Remarks to Bit/Byte Handling Problems

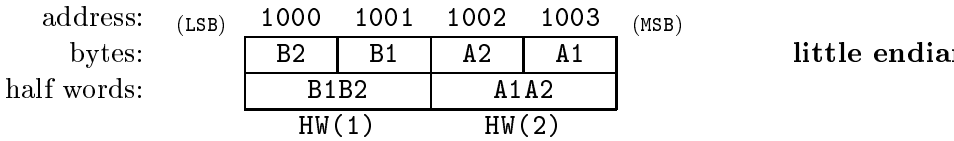
Because the first software revival steps were tried on different platforms, it was attempted to make at least the program interfaces described above insensitive to the byte endian schemes. Unfortunately, the code itself contains highly platform dependent features which only can be eliminated by code changes. This section intends to point out the typical bit and byte handling problems with the JADE software.

Integer values are typically stored in one of three sizes: one-byte, two-byte, or four-byte. The ordering of the bytes (i.e the order in which a sequence of bytes are stored in computer memory) for the integer varies depending on the operating environment on which the integers were produced. Big endian is an order in which the "big end" (most significant value MSB in the sequence) is stored first (at the lowest storage address). Little endian is an order in which the "little end" (least significant value LSB in the sequence) is stored first. For example, consider the four bytes representing the full (32 bit) integer `IW = A1A2B1B2` (hexadecimal) and a half (16 bit) integer array `HW(2)` of dimension 2 related to the full integer via the FORTRAN statement `EQUIVALENCE (IW,HW(1))`. On a big endian computer, the integer is stored as



big endian

(if `A1` is stored at storage address `1000`, for example, `A2` will be at address `1001`, and so on), with `HW(2)` being the least significant half word. On a little endian system, it is stored as



little endian

(`B2` at address `1000`, `B1` at `1001`, and so on). Here, `HW(1)` is the least significant half word. Note that within both the big endian and little endian byte orders, the bits within each byte are big endian. The following platforms are considered big endian: IBM/370 computers (used formerly for JADE DAQ and offline analyses), RS/6000 AIX, HP-UX, SGI IRIX, SUN Solaris, Macintosh MacOS. The following platforms are considered little endian: VAX/VMS, DEC Alpha UNIX, Intel Linux.

Within the JADE software, data arrays like those residing in the BOS common block `/BCS/` are commonly addressed in units of both full integer and half integer units, in the same way as described above. As a consequence, manipulations of the data may explicitly depend on the byte storage order:

- Consider the integer arrays `I`, `J` and the half integer array `HJ` related to each other via

```
INTEGER I(1),J(1)
INTEGER*2 HI(2)
EQUIVALENCE (HI(1),I(1))
```

  In the following, the symbols $\square$ and $\square\square$ represent single half integers and two consecutive half integers forming a full integer, respectively, with the two half integers arranged from left to right according to increasing storage addresses, i.e. $\boxed{\texttt{HI(1)}\,|\,\texttt{HI(2)}}$. Now consider a logical OR operation between the two numbers `4` and `8` stored in these data arrays which have to be passed to the IBM/370 intrinsic `LOR` function. If the operands are full integers, a straightforward emulation would act on the storage schematically like as follows:

```
I=4
J=8
LOR(I,J) = LOR(0|4,0|8) = 0|12 = 12   (big endian case)
LOR(I,J) = LOR(4|0,8|0) = 12|0 = 12   (little endian case)
```

  i.e. it would provide identical results on big endian and little endian machines since the access to the data is not affected by the internal representation of the integer values. On the other hand, an IBM/370 programmer thinking always in big endian convention probably tempts to address the same operation in the following way:

```
HI(2)=4
J=8
LOR(HI(2),J) = LOR(4,0|8) = 0|12 = 12   (big endian case)
```

  whereby he knows that the half integer argument of `LOR` is extended to a full word by filling zeros from the 'left' (i.e. from the MSB side of the word). On a little endian machine, the FORTRAN emulation of `LOR` has first to perform the correct half word assignment according to the programmers intention (avoiding word misalignments of the data arrays passed to `LOR`), in order to prevent the routine from returning wrong results like:

```
LOR(HI(2),J) = LOR(4,8|0) = 8|4 = 524292   (little endian case)
```

  The JADE source code is full of pitfalls like these. As already mentioned in Section 7.4.1, introducing an integer and a half integer version was found to provide an unambiguous and robust solution of the problem.

- At numerous places in the program, bit masks stored in integer and half integer arrays are set and tested for decision making. A customary way to handle single bits in the JADE software looks like as follows (taken from subroutines `XYFIT(patrec/xyfit.F)` and `INPATR(patrec/inpatr.F)`). Firstly, bit patterns are usually set by integer value assignment. For example, the statement `IXYF = 3` sets the two least significant bits of the integer `IXYF` to `1` and the other bits to `0`. To test a bit pattern in a later stage of the program, the IBM/370 intrinsic function `TBIT` is often used to find out whether a given bit

is set (returning the value `TRUE`) or not (returning the value `FALSE`). In the JADE software, the LSB is tested via the statement

```
IF( TBIT(IXYF,31) ) ...
```

hence relying on the IBM/370 bit numbering scheme where bit number 31 denotes the LSB and bit number 0 denotes the MSB. In contrast, on the DEC alpha, LSB is bit number 1 and MSB is bit number 32. This has to be considered within the emulation interface to CERNLIB function `JBIT` that works with an universal bit numbering scheme. Using the whole integer words and the logical AND in the form

```
IF( LAND(IXYF,1).EQ.1 ) ...
```

instead of explicitly numbering the bit would have been an elegant way to circumvent any of these platform dependencies from the outset.

- The following example demonstrates a type of platform dependence which can not be compensated by programming an appropriate emulation interface and is generally hard to detect: At the end of the event loop in `MCJADE`, the JADE data are organised in BOS banks and then written out in binary format. The Lead Glass data bank `ALGN` (output by the subroutine `STALGN(jmc/stalgn.f)`) starts with two half words containing a descriptor with various smearing flags which indicate certain `MCJADE` actions on the Lead Glass arrays. In the default case, the flags are set by integer value assignment

  $$\texttt{ILGL(1)=2} = \begin{cases} \boxed{0}\boxed{2} & \text{(big endian case)} \\ \boxed{2}\boxed{0} & \text{(little endian case)} \end{cases}$$

  The tracking history of the Lead Glass data is then passed to the JADE Supervisor via this integer word. Unfortunately, the cluster analysis subroutine `LGCDIR(source/lgcdir9.F)` checks the smearing flags by explicitly inquiring the *most significant* half word of the corresponding integer word and puts the value into the second element of a half integer array called `HELP`. Finally, an integer `IHELP` related to `HELP` via `EQUIVALENCE` is used to check the smearing flags:

  $$\texttt{LAND(IHELP,2)} = \begin{cases} \texttt{2} & \text{(big endian case)} \\ \texttt{0} & \text{(little endian case)} \end{cases}$$

  To conclude, the communication between the affected parts of `MCJADE` and `SUPERV` depend on the endian convention in a way which does not strike at a first glance. Its removal requires some code modification: either the flag word must be build up in `STALGN` in terms of half integer words, or the access of the information must be performed in `LGCDIR` by inquiring full integer words.

Due to the unstructured build-up of the source code it is not possible to detect all code inherent platform dependencies of the latter type with a maintainable effort. This gives reason to attempt running the JADE software always on big endian platforms.

Note that the existence of different endian schemes is not *per se* a problem. Moving unformatted data files (like the JADE BOS banks) between big endian and little endian computers requires only that the byte orders of the read data banks must be inverted before data processing. The actual dependencies on the internal representation of the numbers are only caused by the fact that bit and byte manipulation are not performed consistently in units of a fixed word length, say, 32 bit words.

7.5 Further Code Changes

Further code modifications were performed in order to suppress various XLF compiling warnings or errors and to prevent the programs from (logical) run time errors. Among other things, the code contains various IBM FORTRAN extensions which are not properly handled by the XLF compiler. See the following examples:

- The code uses the `DEFINE FILE` and the direct access `READ` statements to allocate and read data files containing records of fixed length `<rl>` (e.g. `RDDA(bos/rdda.F)`):

  ```
  DEFINE FILE <u>(<m>,<rl>,U|E,<irec>)
  READ(<u>'<irec>) ...
  ```

  where `<u>` represents the logical unit number, `<m>` the maximum number of records, and `<irec>` the running record number incremented after reading a record. `U|E` refer to unformatted and formatted records, respectively. This FORTRAN extension had to be replaced by standard FORTRAN `OPEN/READ`.

- Data records of previously unknown word lengths are usually accessed by calling a reading subroutine in the following way (see e.g. `EVREA1` called by `KLREAD(jadegs/klread.F)` or `BFRD` called by `BREAD(bos/bread.for)`):

  ```
  SUBROUTINE EVREA1(NUNIT,...,NWORD,...)
  DIMENSION IDATA(NWORD)
  READ(NUNIT) NWORD,IDATA
  ...
  ```

  The number of words `NWORD` of the currently read record ought to be provided by the first word of the record itself and is intended to be stored into the array `IDATA`. This feature is not supported by the XLF compiler since `NWORD` is not defined prior to the `READ` statement (e.g. `NWORD=0`). In the modified program version, the number `NWORD` is first supplied by tentatively reading the first word of the record. After backspacing the record, the reading subroutine is then called again with the well defined value for `NWORD`.

- `DATA` statements were found in the executable part of procedures. This features is not supported by the XLF compiler.

- In some cases, the access to common block variables contradicts the declared common block sizes or the array dimensions. This may be due to a mismatch between different code development versions. E.g., `KLREAD` writes more words in common `/CVCCAL/` (`patrecsr/mvccal.for`) via `MVCL` than foreseen by the array declarations of this common block. As a serious consequence, the data residing in the storage succeeding `/CVCCAL/` are partially overwritten. Another example: The calibration flag array `LBMC` residing in common block `/CMCCAL/` is of dimension 15 in the JADE block data `jadebd.F` and of dimension 16 in `KLREAD`. But actually, the present code version demands that it had to be extended to dimension 17.

- It was found that the argument list of some `CALL` statements do not match the argument list of the subroutine called (e.g. `ZSFIT(jadegs/zsfit.F)`, `VTXCRV(grafix/vtx.F)`). This mismatch was corrected.

- Different from the original program version, all `BLOCK DATA` subroutines are now named.
- Alternate return labels in a `CALL` statement are preceded by an ampersand '&', whereas the XLF compiler requires an asterisk '*'.
- There exist some risky and ambiguous `DO` loop constructs which the XLF compiler might interpret differently from the programmers intention.

  1. The following type of construction was found e.g. in `JRECAL (jadegs/jrecal.F)`, `TAGGTP (tagg/taggtp.F)`:

```
        GO TO 100
        ...
        DO 100 I=1,10
        ...
    100 CONTINUE
```

     From the context of the cited programs it is apparent that, in case of the unconditional `GO TO` branch, the `CONTINUE` statement ought to be a simple "no operation". Instead, the XLF compiler handles the `CONTINUE` as the end of the `DO` loop, with the loop counter variable `I` set to a value defined by the program flow prior to the branch (`I=0` if no value assignment was performed previously). This "misinterpretation" was corrected here by introducing a labelled `CONTINUE` assigned to the `GO TO` statement but behind the end of `DO` loop.

  2. The second curio found are jumps from outside into `DO` loops. This feature is forbidden in standard FORTRAN but nevertheless is (by demand) accepted by the XLF compiler which prints only a compiling warning message. Such constructions look like those sketched in example a) below (found in `PATROL (patrol.F)`, `TRUTH (grafix/truth.F)`):

a)

```
      DO 100 I=1,10
      ...
      GO TO 900
  910 CONTINUE
      ...
  100 CONTINUE
      ...
      RETURN
  900 CONTINUE
      ...
      GO TO 910
```

b)

```
      DO 100 I=1,10
      ...
      GO TO 900
      ...
  100 CONTINUE
      ...
      RETURN
  900 CONTINUE
      ...
      GO TO 100
```

     In the normal context, the program exits the `DO` loop prematurely by a conditional or unconditional branch to another part of the program. After further processing, the program jumps back into the `DO` loop and continues the loop using the "frozen" former value of the loop control variable (the XLF compiler allows even a reassignment of the loop variable). This construct effectively simulates a kind of a fast subroutine call from within in a `DO` loop but without time consuming data stacking.

     Since the XLF compiler seems to act in agreement with the programmers intention, no code modification was performed, but it is unlikely that this feature is accepted

by other compilers. This has to be considered when attempting to adapt the code on other platforms.

Last but not least, example b) (taken from subroutine `PATROL`) shows a similar construct. Here, it is not clear if the re-jump to label `900` was intended to force the program to continue with the `DO` loop or to proceed with the subsequent code. With regard to the comments of example 1, the latter case was assumed here, and the code was modified accordingly.

8 Remaining Tasks

The resurrection of main parts of the JADE software was quite successful so far, thus giving reason to be optimistic with regard to still existing problems to solve. This section recapitulates in brief various remaining JADE software revival tasks and further desirable code improvements, of which some of them are already mentioned in the present note.

The following code modifications should be straightforward:

- The implementation of the still unused but available refined event analysis routines (e.g. three dimensional track refitting with vertex constrain) into the JADE Supervisor (Section 5.2).
- Programming an interface to output fully analysed JADE events into (OPAL like) HBOOK NTuple files, instead into ZE4V records (Section 5.3).

Further partially more complex tasks are:

- Make the JADE raw data (REDUC1/REDUC2) (Section 3.2) usable for analysis:
  - Either convert FPACK records back into binary BOS banks (either big endian or little endian, with the floating point numbers transformed to IEEE standard),
  - or program an appropriate FPACK interface for the Supervisor.
- Test the converted JADE calibration files (Section 3.4) and the calibration procedure using the raw data and already preprocessed data.
- Transfer all the raw data currently residing on EXABYTE cartridges deposited at DESY to a more professional data storage management system (e.g. CASTOR at CERN).
- Reactivate the Tokyo Shower Simulation program for the Lead Glass Calorimeter (Section 2.1).
- Retrieve and reactivate the Muon System simulation code[18].
- Adapt the JADE software to PC Linux or Macintosh platforms. Power PCs with Linux are "bi-endian", i.e. they support both big endian and little endian addressing modes. Once a mode is selected, all subsequent memory loads and stores are determined by the memory-addressing model of that mode[19].

[18] It is possible that this part of the software is completely lost [41].

[19] Note that IBM has now a beta version of XL FORTRAN compiler for Power PC Linux

A JADE BOS Banks

In the following, the most important JADE BOS banks processed by the detector simulation, the JADE Supervisor, and the ZE4V packing software are summarised. See the listed references to the corresponding JADE Notes and JADE Computer Notes for further detailed information about the structuring of these banks. The format of most data banks is described in JADE Note 32 and Supplements 1-6. The general handling of BOS data is explained in [27].

BOS bank name	Ref.	explanation
`HEAD`	[28]	General event information / BOS pointer table
Lead Glass:		
`ALGL`	[18]	Raw Lead Glass data (*real data only*)
`ALGN`	[18]	Calibrated Lead Glass data
Inner Detector:		
`JETC`	[28, 76]	Raw + calibrated Jet Chamber data
`VTXC`	[77]	Raw + calibrated Vertex Chamber data
Muon System:		
`MUEV`	[78, 79]	Raw Muon System data*)
Trigger:		
`TRIG`	[33–35, 80–83]	Trigger information
Tagging System:		
`ATAG`	[28, 82, 84]	Raw information from Forward Detector
`ATBP`	[82]	Beam Pipe counter raw data
`ATOF`	[82]	Time-of-Flight counter raw data
`LATC`	[28, 82–84]	Information about which Latches were set by the detector parts
`TAGC`	[28, 84]	Tagging Drift Chamber data
Monte Carlo Information:		
`VECT`	[7, 56–58]	Monte Carlo four vector data
`PALL`	[56, 58]	Monte Carlo trace back information
Event Analysis Results:		
`TAGG`	[28, 84]	Tagging system results
`ZVTX`	[85]	Fast Z vertex
`LGCL`	[18]	Lead Glass cluster analysis results
`PATR`	[29]	Inner Detector pattern recognition results
`JHTL`	[30]	Hit Label information for `PATR`
`MUR1`, `MUR2`	[86]	Muon analysis result banks*)
Top Level Analysis Results:		
`ZE4V`	[22]	ZE4V packing result bank
`TPEV`	[23, 46]	Global event data from TP9 reconstruction*)
`TPVX`	[23, 46]	Vertex data from TP9 reconstruction*)
`TPTR`	[23, 46]	Particle data from TP9 reconstruction*)

*) The Muon System simulation and the TP9 analysis programs are not reactivated as yet.

B JADE Data and Monte Carlo Files

This section gives an overview of the available ZE4V, BOS and CPROD files. Most of the data reside in compressed `gtar` archive files stored on CASTOR[20] tapes at CERN.

The general format of the ZE4V data is described in [22]. The format descriptors for the ASCII versions can be found in the corresponding file headers. ZE4V file names are marked with the suffix "`.ze4v.<n>`", where `<n>` is the running file number. The program `zread` (Section 6) is a FORTRAN template suitable for reading and processing both ASCII and binary formatted ZE4V files.

The BOS files partially consist of the BOS banks listed in Appendix A. The MC tracking banks were newly generated using the program `mcjade` (Section 5.1). They are suited for further processing with the JADE Supervisor `superv` (Section 5.2) and the program `ze4v` (Section 5.3). BOS files have the suffix "`.bos.<n>`".

B.1 ZE4V Data

All multihadronic JADE data are available as ZE4V records in ASCII format, i.e. plain text files. Two different versions called 9/87 and 5/88 exist, see Section 3.3 for details. The JADE events are sorted by increasing run and event numbers. The present samples are subdivided according to the period of data taking. The corresponding archive files with the respective number of events (#) and storage occupancies are listed in Table 6.

B.2 ZE4V MC

Preprocessed detector Monte Carlo files based on the JETSET 6.3 generator are available as ASCII formatted ZE4V records. The simulations are based on the detector configurations of 1982, 1985, and 1986, at centre-of-mass energies $\sqrt{s} = 35$ and 44 GeV, respectively. The historic labels "ps2" and "sh" in the file names refer to the JETSET parton shower simulation with and without coherent branching, respectively [87]. There are some uncertainties about the knowledge of the parameter settings of the underlying event generator. They are presumably given in [50,51]. See Table 6 for a list of the archive files.

The newly generated detector simulation data reside in binary formatted ZE4V files. They are based on the event generators PYTHIA 5.7, HERWIG 5.9, ARIADNE 4.08, and JETSET 6.3, which were run at centre-of-mass energies about $\sqrt{s}$ =14, 22, 35, 38 and 44 GeV. The parameter settings are described in [1]. The corresponding archives files are listed in Table 7.

B.3 BOS MC and CPROD files

The complete `mcjade` tracking banks with PYTHIA as underlying physics generator were also kept. They have been used to produce the MC ZE4V records mentioned above and are contained in the files listed in Table 7. In addition, the CPROD 4 vector files from all event generators suited to be processed by `mcjade` are available for the most relevant c.m.s. energies, see Table 8.

[20] CERN Advanced STORage manager

B.4 Raw JADE Data (FPACK format)

All raw JADE data (after REDUC1/REDUC2 cuts) are available in FPACK format. At present, they reside on EXABYTE cartridges deposited at the DESY laboratory in Hamburg, Germany [41]. A test sample suitable for FPACK conversion tests is located in

`/castor/cern.ch/user/m/movilla/jade/dat/reduc/fpck4.f11kuh.jade.events`.

PREPROCESSED JADE DATA
location: `/castor/cern.ch/user/m/movilla/jade/dat/ze4v`

version	period	# total	# per file (max.)	# of files	archive name	storage occupancy
9/87	1979-85	79872	3000	27	`tr7985.v987.gtar.gz`	102 MB
9/87	1986	29433	3000	10	`tr86.v987.gtar.gz`	50 MB
5/88	1979-85	81454	5000	17	`tr7985.v588.gtar.gz`	108 MB
5/88	1986	29433	5000	6	`tr86.v588.gtar.gz`	50 MB

PREPROCESSED JADE MC
location: `/castor/cern.ch/user/m/movilla/jade/mc-old/`

tracking year	$\sqrt{s}$ [GeV]	# total	# per file (max.)	# of files	archive name	storage occupancy
1982	35	31914	4000	8	`ld63ps2.e35.tr82.gtar.gz`	54 MB
1985	44	23933	4000	6	`ld63sh.e44.tr85.gtar.gz`	43 MB
1986	35	39868	4000	10	`ld63ps2.e35.tr86.gtar.gz`	70 MB
1986	35	39880	4000	10	`ld63sh.e35.tr86.gtar.gz`	69 MB

further historic MC files with unclear processing history are located in
`/castor/cern.ch/user/m/movilla/jade/mc-old/special/`

Table 6: List of usable preprocessed ZE4V data.

NEW JADE MC location: /castor/cern.ch/user/m/movilla/jade/mc/...					
tracking date	$\sqrt{s}$ [GeV]	archive name	# total	# per file (max.)	storage occupancy
PYTHIA 5.7		.../pythia/			
Jul/17/81	14.0	py57_14.0_81-07-17.gtar.gz	100000	5000	685MB
Jun/17/81	22.0	py57_22.0_81-06-17.gtar.gz	100000	5000	812MB
May/17/82	34.6	py57_34.6_82-05-17.gtar.gz	300000	5000	2.8GB
May/05/86	35.0	py57_35.0_86-05-17.gtar.gz	499999	5000	4.8GB
Oct/01/85	38.3	py57_38.3_85-10-01.gtar.gz	100000	5000	1.0GB
May/17/85	43.8	py57_43.8_85-05-17.gtar.gz	199999	5000	2.1GB
HERWIG 5.9		.../herwig/			
Jul/17/81	14.0	hw59_14.0_81-07-17.gtar.gz	99877	5000	79MB
Jun/17/81	22.0	hw59_22.0_81-06-17.gtar.gz	99923	5000	92MB
May/17/82	34.6	hw59_34.6_82-05-17.gtar.gz	299820	5000	315MB
May/05/86	35.0	hw59_35.0_86-05-17.gtar.gz	499708	5000	553MB
Oct/01/85	38.3	hw59_38.3_85-10-01.gtar.gz	99947	5000	107MB
May/17/85	43.8	hw59_43.8_85-05-17.gtar.gz	199886	5000	221MB
ARIADNE 4.08		.../ariadne/			
Jul/17/81	14.0	ar48p_14.0_81-07-17.gtar.gz	100000	5000	82MB
Jun/17/81	22.0	ar48p_22.0_81-06-17.gtar.gz	100000	5000	97MB
May/17/82	34.6	ar48p_34.6_82-05-17.gtar.gz	299998	5000	334MB
May/05/86	35.0	ar48p_35.0_86-05-17.gtar.gz	499998	5000	586MB
Oct/01/85	38.3	ar48p_38.3_85-10-01.gtar.gz	100000	5000	113MB
May/17/85	43.8	ar48p_43.8_85-05-17.gtar.gz	199998	5000	235MB
JETSET 6.3		.../jetset/			
Jul/17/81	14.0	jt63_14.0_81-07-17.gtar.gz	100000	5000	85MB
Jun/17/81	22.0	jt63_22.0_81-06-17.gtar.gz	100000	5000	101MB
May/17/82	34.6	jt63_34.6_82-05-17.gtar.gz	299996	5000	357MB
May/05/86	35.0	jt63_35.0_86-05-17.gtar.gz	399998	5000	503MB
Oct/01/85	38.3	jt63_38.3_85-10-01.gtar.gz	100000	5000	122MB
May/17/85	43.8	jt63_43.8_85-05-17.gtar.gz	199999	5000	255MB

Table 7: List of newly generated BOS and ZE4V data.

CPROD Files (location: /castor/cern.ch/user/m/movilla/jade/cprod/...)		
$\sqrt{s}$ [GeV]	archive name	storage occ.
PYTHIA 5.7	.../pythia/	
14.0	py57.14.0.gtar.gz	95MB
22.0	py57.22.0.gtar.gz	110MB
34.6	py57.34.6.gtar.gz	371MB
35.0	py57.35.0.gtar.gz	673MB
38.3	py57.38.3.gtar.gz	128MB
43.8	py57.43.8.gtar.gz	268MB
HERWIG 5.9	.../herwig/	
14.0	hw59.14.0.gtar.gz	88MB
22.0	hw59.22.0.gtar.gz	103MB
35.0	hw59.35.0.gtar.gz	601MB
38.3	hw59.38.3.gtar.gz	124MB
43.8	hw59.43.8.gtar.gz	259MB
ARIADNE 4.08	.../ariadne/	
14.0	ar48p.14.0.gtar.gz	84MB
22.0	ar48p.22.0.gtar.gz	98MB
35.0	ar48p.35.0.gtar.gz	570MB
38.3	ar48p.38.3.gtar.gz	118MB
43.8	ar48p.43.8.gtar.gz	245MB
JETSET 6.3	.../jetset/	
14.0	jt63.14.0.gtar.gz	83MB
22.0	jt63.22.0.gtar.gz	100MB
34.6	jt63.34.6.gtar.gz	360MB
35.0	jt63.35.0.gtar.gz	482MB
38.3	jt63.38.3.gtar.gz	125MB
43.8	jt63.43.8.gtar.gz	265MB

Table 8: List of newly generated CPROD files.

C JADE Interactive Graphics Commands

The following is a list of most commands available within the JADE event display program `jadez` (Section 5.5). Further information can be found in [24].

Views:

RA displays the central detectors and TOF counters in the r-ϕ view

RB displays the central detectors, TOF counters and barrel Lead Glass (LG) in the r-ϕ view

RC displays the central detectors, TOF counters, barrel LG and Muon Chambers in the r-ϕ view

ZXA displays the central detectors and TOF counters in the z-x view

ZXB displays the central detectors, TOF counters and barrel Lead Glass (LG) in the z-x view

ZXC displays the central detectors, TOF counters, barrel LG and Muon Chambers in the z-x view

ZYA, ZYB, ZYC work like ZXA, ZXB and ZXC, respectively, but display the subdetectors in the z-y view

RU displays the rolled-out view of the LG and the forward detector

VC displays the Vertex Chamber in the r-ϕ view

CYL displays a perspective view of the Jet Chamber and the barrel LG

DET displays the detector hardware on top of the current view

PRO displays projections of the two orthogonal views

EC displays end cap LG hits in r-ϕ views

TRLG displays the T1 trigger conditions

TRG2 displays the T2 trigger conditions

CSTV changes the standard view and the automatic display of the event and the detector

Analysis results:

RES displays Inner Detector (ID) and Lead Glass analysis results

TR displays ID hits with various options

TRUE displays the original 4-vectors in Monte Carlo events

VX displays vertex finding results with several options

DEDX displays energy loss analysis results with several options

TOF displays time-of-flight analysis results with several options

AX displays the event jet axes with several options

CLUS displays the Lead Glass energy clusters in the RU view (see below)

VFIT performs a vertex fit of chosen tracks

MASS computes the effective mass of a group of particles

Control:

N goes to next event

FIND fetches an event by run number and event number

MORE switches to a new input data file

LEVELS changes the JADE Supervisor stop flags (see Section 5.2)

C continues to the next JADE Supervisor level with the stop flag set

OPT flips any of the drawing flags

STAT displays status of the drawing flags

ZOOM changes scale and/or origin of the current view (clipping boundaries are set with mouse clicks)

RESET returns to the standard scale and origin

STOP stops the program

Extras:

H generates a PostScript copy of the display

PICK returns the coordinates of the cross-hairs on the screen moved by the mouse

DRAW invokes a drawing routine for circles, lines, points

COM adds a comment to the picture

MACRO creates a macro command sequence

EDITMAC edits a user defined macro

DELMAC deletes a user defined macro

RENAMAC renames a user defined macro

References

[1] P.A. Movilla Fernández. *Studien zur Quantenchromodynamik und Messung der starken Kopplungskonstanten α_S bei $\sqrt{s}$ = 14-44 GeV mit dem JADE-Detektor.* PhD thesis, RWTH Aachen, 2003. PITHA 03/01, MPI-PhE 2003/01.

[2] P.A. Movilla Fernández. Determinations of α_S at $\sqrt{s} = 14$ GeV to 44 GeV Using Resummed Calculations. 2002. MPI-PHE-2002-08, contributed to XXXVIIth Rencontres de Moriond on QCD and Hadronic Interactions, Les Arcs, France, 16-23 Mar 2002.

[3] P.A. Movilla Fernández. α_S and Power Corrections from JADE. 2002. MPI-PHE-2002-12. Proceedings of 31st International Conference on High Energy Physics (ICHEP 2002), Amsterdam, The Netherlands, 24-31 Jul 2002.

[4] Mona Blumenstengel and Stefan Kluth. Fragmentation Functions Using e^+e^- Data from PETRA and LEP. 2002.

[5] JADE Progress Report. JADE Note 22, 1978.

[6] B. Naroska. e^+e^- Physics with the JADE Detector at PETRA. *Phys. Rept.*, 148:67, 1987.

[7] E. Elsen. Detector Monte Carlo. JADE Computer Note 54, 1982.

[8] E. Elsen. *Multihadronerzeugung in der e^+e^--Vernichtung bei PETRA-Energien und Vergleich mit Aussagen der Quantenchromodynamik.* PhD thesis, Universität Hamburg, 1981.

[9] K. Meier. *Untersuchung der Photonproduktion bei Elektron-Positron Annihilationen am Speicherring PETRA.* PhD thesis, Universität Hamburg, 1987.

[10] K. Meier. Monte Carlo Simulation of Electromagnetic Showers in the Lead Glass. JADE Computer Note 70, 1984.

[11] N. Magnussen. New Shower Functions for SF5/SF6 and New MC and Data Routines for Meier LG Shower Fitting. JADE Note 136, 1986.

[12] A.J. Finch. Tagging System Monte Carlo. JADE Computer Note 86, 1986.

[13] C.K. Bowdery. *Production of Inclusive Dimuon Events in Electron Positron Annihilation at PETRA Energies.* PhD thesis, Victoria University of Manchester, 1982. HEP-T-105.

[14] S. Yamada. Čerenkov Light Detection Efficiency for the JADE Lead Glass Counter. JADE Note 20, 1977.

[15] J.E. Olsson. The Subroutine NPECR6. JADE Computer Note 20 Supplement 1, 1987.

[16] C. Bowdery and J.E. Olsson. The JADE SUPERVISOR Program. JADE Computer Note 73, 1984.

[17] J.E. Olsson, P. Steffen, M.C. Goddard, G.F. Pearce, and T. Nozaki. Pattern Recognition Programs for the JADE Jet Chambers. *Nucl. Instrum. Meth.*, 176:403–407, 1980.

[18] S. Yamada. Analysis Program for Lead Glass Counters. JADE Computer Note 14D, 1984.

[19] D.D. Pitzl. Reconstruction of Cluster Energies in the Barrel Lead Glas. JADE Computer Note 101, 1988.

[20] D.D. Pitzl. *Quellen von Photonen in hadronischen Ereignissen der Elektron-Positron Vernichtung.* PhD thesis, Universität Hamburg, 1989.

[21] J. Hagemann, J.E. Olsson, and R. Ramcke. Inner Detector Smearing and Trigger Simulation. JADE Computer Note 66, 1983.

[22] G. Eckerlin and M. Zimmer. A New Compact Data-Format Description of Bank - ZE4V. JADE Computer Note 99, 1988.

[23] C.K. Bowdery and J.J. Pryce. The New TP Program Version 9. JADE Computer Note 102, 1988.

[24] C. Bowdery and J.E. Olsson. JADEZ - The JADE Graphics Program. JADE Computer Note 85/D, 1986.

[25] M. Goddard, J.E. Olsson, and P. Steffen. Cuts of First Data Reduction. JADE Computer Note 38, 1980.

[26] J.E. Olsson. A General Second Reduction Program. JADE Computer Note 43, 1980.

[27] V. Blobel. B O S: Bank Organization System. Dynamic Storage Organization with FORTRAN. Hamburg DESY - Internal Report F14-77-01, 1977.

[28] W. Bartel. IBM Data Banks. JADE Computer Note 23, 1979.

[29] P. Steffen. Track Bank from Pattern Recognition Program. JADE Computer Note 12, 1979.

[30] P. Steffen. Hit Lable Bank Created by PATREC. JADE Computer Note 21, 1979.

[31] E. Elsen and K.H. Hellenbrand. Re-Analysis of Multihadronic Events. JADE Computer Note 83, 1985.

[32] W. Bartel. Minutes of the JADE Software Meeting on Dec. 14, 1987 at DESY. JADE Computer Note 98, 1988.

[33] E. Gadermann and H. Krehbiel. Trigger Scheme, Realization of T1. JADE Note 30, 1978.

[34] M. Helm, H. Krehbiel, and H. Riege. Trigger T2 (Jet Chamber Fast Track Recognition). JADE Note 31, 1979.

[35] J. Allison and H. Prosper. Muon Trigger. JADE Note 28, 1977.

[36] H.E. Mills. Online Event Filtering in the JADE Data Acquisition System. *Nucl. Instrum. Meth.*, A247:525, 1986.

[37] P. Dittman. Online Event Analysis in the NORD50. JADE Note 78, 1981.

[38] H.E. Mills. Online Event Analysis in the NORD50. JADE Note 78 Supplement 1 and 2, 1985/1986.

[39] H.E. Mills. Introduction to Using the JADE NORD10S/50 Computers. JADE Note 92, 1983.

[40] D. Cords, P. Dittmann, R. Eichler, and H.E. Mills. The Data Acquisition System for the JADE Detector. *Nucl. Instrum. Meth.*, A245:137, 1986.

[41] J.E. Olsson. Private communication.

[42] *F-Package for Input/Output (Version 1.00/00)*, 1994.

[43] J.E. Olsson, G. Eckerlin, and E. Elsen, 1996. Private communication.

[44] P. Bock, J. von Krogh, et al., 1997. Private communication.

[45] S. Yamada, C. Bowdery, and E. Elsen. The JADE TP Program. JADE Computer Note 79, 1984.

[46] C Bowdery and S. Yamada. Format of the Generation 8 TP banks. JADE Computer Note 80, 1984.

[47] J. Spitzer. How to Use New ID Calibration. JADE Computer Note 94, 1987.

[48] E. Elsen and J. Spitzer. A General s-z Fit Routine. JADE Computer Note 95, 1987.

[49] J. Spitzer. Improved Resolution with Z-Chamber Hits. JADE Computer Note 95 Supplement 1, 1987.

[50] M. Zimmer. *Die Fragmentation von b-Quarks.* PhD thesis, Ruprecht-Karls-Universität, Heidelberg, 1989.

[51] G. Eckerlin. *Vergleich der starken Wechselwirkung von b-Quarks und leichten Quarks in e^+e^--Reaktionen bei 35 GeV.* PhD thesis, Ruprecht-Karls-Universität, Heidelberg, 1990.

[52] P. Steffen. The JADE Calibration Scheme. JADE Computer Note 68, 1983.

[53] P. Steffen. New Convention for Calibration Data. JADE Computer Note 58, 1982.

[54] E. Elsen and Olsson. J. Calibration for the 1986 REDUC1 and Standard Status of Rutherford Tapes. JADE Computer Note 89, 1986.

[55] E. Elsen and J. Olsson. REDUC1 and REDUC2 for 1986 Data. JADE Computer Note 92, 1987.

[56] Monte Carlo Formats. JADE Computer Note 10, 1978.

[57] E. Elsen. Monte Carlo Tracking. JADE Computer Note 26, 1979.

[58] C. Bowdery. Monte Carlo Traceback. JADE Computer Note 69, 1983.

[59] C. Bowdery. Monte Carlo Data Validation. JADE Computer Note 72, 1984.

[60] P. Dittmannn. How to Use the Vertex Fit Program. JADE Computer Note 32, 1980.

[61] Ch. Pahl. PhD thesis, Ludwigs-Maximilians-Universit"at, (in Vorbereitung).

[62] S. Kluth. Private communication.

[63] T. Sjöstrand. High-Energy Physics Event Generation with PYTHIA 5.7 and JETSET 7.4. *Comput. Phys. Commun.*, 82:74–90, 1994.

[64] T. Sjöstrand. *PYTHIA 5.7 and JETSET 7.4 Physics and Manual.* CERN-TH-7112-93.

[65] T. Sjöstrand. The Lund Monte Carlo for Jet Fragmentation and e^+e^- Physics: JETSET Version 6.2. *Comput. Phys. Commun.*, 39:347, 1986.

[66] T. Sjöstrand and M. Bengtsson. The Lund Monte Carlo for Jet Fragmentation and e^+e^- Physics: JETSET Version 6.3: an Update. *Comput. Phys. Commun.*, 43:367, 1987.

[67] Leif Lönnblad. ARIADNE Version 4: A Program for Simulation of QCD Cascades Implementing the Color Dipole Model. *Comput. Phys. Commun.*, 71:15–31, 1992.

[68] G. Marchesini et al. HERWIG: A Monte Carlo Event Generator for Simulating Hadron Emission Reactions with Interfering Gluons. Version 5.1 - April 1991. *Comput. Phys. Commun.*, 67:465–508, 1992.

[69] R. Odorico. COJETS 6.23: A Monte Carlo Simulation Program for $\bar{p}$p, pp Collisions and e^+e^- Annihilation. *Comput. Phys. Commun.*, 72:238–248, 1992.

[70] DESY R-INFO 79/C, 1977; DESY R-INFO 77/C, 1979.

[71] Tektronix Inc. *PLOT-10 Terminal Control System - User's Manual.*, 1976.

[72] P.K. Schilling. *IPS User's Guide.*, 1982. Hamburg Desy - Internal Report R2-81-1.

[73] K.G. Begeman. *SHELTRAN.* Kapteyn Laboratorium Groningen, 1991. Manual.

[74] CERNLIB - Short Writeups. CERN Program Library.

[75] HIGZ - High Level Interface to Graphics and ZEBRA. CERN Program Library (Q120).

[76] R.D. Heuer, T. Nosaki, J. Olsson, and P. Steffen. Conventions of Jet Chamber Formats for Pattern Recognition and Related Programs. JADE Computer Note 5, 1978.

[77] J. Hagemann, C. Kleinwort, and R. Ramcke. Vertex Chamber Software. JADE Computer Note 100, 1988.

[78] R. Barlow. The JADE Muon Monte Carlo. JADE Computer Note 67, 1983.

[79] R. Barlow. *Production of Inclusive Dimuon Events in Electron Positron Annihilation at PETRA Energies.* PhD thesis, Victoria University of Manchester, 1982.

[80] M. Helm and B Naroska. IBM Trigger Banks. JADE Computer Note 23a, 1979.

[81] B Naroska. Trigger Words. JADE Computer Note 23b, 1979.

[82] D. Cords. Logical Record Format for Online Data. JADE Note 32 and Supplements 1-6, 1979.

[83] B. Naroska. Tentative Allocation of New Trigger Words and Trigger Bits. JADE Note 82, 1981.

[84] G. Hughes and H. Wriedt. Data Format of the Tagging Banks. JADE Computer Note 16, 1979.

[85] B. Naroska. Bank Created by the Z Vertex Reconstruction. JADE Computer Note 17, 1979.

[86] J. Allison, C. Bowdery, I. Duerdoth, J. Hassard, H. Mccann, and H. Prosper. Muon Software Information. JADE Computer Note 22, 1979.

[87] G. Eckerlin. Private communication.