CERN/LEPC/84-8
LEPC / M 46
January 23, 1984

MEMORANDUM

To: LEP Experiments Committee

From: ALEPH Collaboration

Subject: Data Acquisition and Data Analysis

---

1. INTRODUCTION

This memorandum addresses the questions on Data Acquisition and Data Analysis which were put forward by the LEP Experiments Committee (LEPC/M 43). It summarizes our present understanding of the situation, and gives an account of our ongoing work and plans for the future.

The next section deals with the Data Acquisition complex. Section 3 covers the Off-line Analysis. Section 4 states our position with respect to Networks. Section 5 outlines our strategy for searching for rare events in a total sample of 10^7 physics events. Finally, the last section summarizes the resources we request from CERN.

We would like to underline that the ALEPH experiment in general, and its Data Acquisition and Data Analysis in particular, are very much dependent on good support from CERN. Out of the more detailed list given in the last section, we wish to stress the importance of CERN support in three areas which are of particular concern to us:

- FASTBUS development.
- A large VAX computer for program development.
- Increase of the central computing capacity.

2. DATA ACQUISITION

2.1 Trigger

The trigger aims at reducing the rate of background events to a level acceptable for tape writing (1-2 Hz) while accepting all good events. In general, the detectors start their digitization upon arrival of a bunch-crossing signal. The readout is then inhibited in case the trigger condition fails. A high level of flexibility has to be built into the triggers since the nature of backgrounds and their rates are not well known. The main source of background events, which should be rejected by the trigger, is expected to be beam-wall interactions of off-momentum particles.

The trigger system is organized in two levels:

1. Level 1 trigger: the trigger will be fast. The total delay from beam crossing to the input of the TPC gating circuit will be ≈1.5 μs. The rate must not exceed a few hundred Hz. The trigger serves to open the TPC gate and to initialize the Level 2 trigger. It will not introduce any dead-time, since beam crossing occurs only every 23 μs.

2. Level 2 trigger: This is based mainly on the TPC and decides whether to continue processing the data or to clear the data acquisition system in case of failure. The trigger should reduce the rate to less than ten Hz. This information is available a few microseconds after the end of the TPC drift-time, say 50 μs after beam crossing. The dead-time introduced in this stage will be of the order of a few percent.

As will be discussed in section 2.3, a post read-out selection is planned, performed by an Event Processor acting on complete events. The Event Processor will reduce the data rate to 1-2 Hz if it is still too high after Level 2.

A Trigger Supervisor will be the central logic to coordinate the various trigger levels and to distribute the appropriate signals to the TPC and the data-acquisition system.

2.1.1 The Level 1 Trigger

The main idea of this trigger is to be sensitive to tracks and/or energy deposits. Based on the experience of PETRA and PEP experiments, we propose a trigger which is satisfied by any one of the following conditions:

1. at least two minimum-ionizing tracks (minions)

2. at least one minion and one energy cluster (low threshold)

3. total electromagnetic or hadronic energy (high threshold)

In addition to these three main triggers there will be a number of special triggers: triggers on single photons, two-photon events with an electron tag; very low angle e^+e^- or $\mu^+\mu^-$ events. The topological acceptance of two tracks and the energy threshold of clusters can be varied to find the best compromise between the acceptance of good events and the background rate. The trigger system is redundant, i.e. most real events will fulfill more than one trigger condition.

The detector elements used in the Level 1 trigger are:

1. the inner chamber (IC)

2. the hadron calorimeter,

3. the electromagnetic calorimeter,

4. the muon chambers,

5. the luminosity monitor.

Tracks will be defined by a track segment found in the IC and a signal above minion threshold in a segment of the hadron calorimeter. Track segments in the IC alone may not be sufficient, since this chamber is very close to the beam pipe and is therefore exposed to heavy background. Geometrical correlation between the track segments in the IC and the calorimeter information will not be required.

1. The inner chamber has 8 layers of sense wires and 8 cathode layers. Two trigger processors are used to search for track segments in the $r\phi$ plane and in the rz plane. In both planes the track segments are requested to emanate from the intersection point.

2. The hadron calorimeter uses layers of streamer tubes, with pad readout arranged in 5160 towers. Many towers will be added to form one trigger segment. A possible choice of one such segment is 30° times 30° solid angle coverage. The barrel would then be divided into 12 ϕ-bins and 4 θ-bins. For the end-caps, 6 ϕ-bins and 2 θ-bins are adequate. This leads to a total of 72 trigger segments. To ensure a uniform energy response for clusters, four adjacent segments are summed. We obtain 72 overlapping segments, four times the size of the original segments.

3. The electromagnetic calorimeter consists of a total of 72 000 towers. The segmentation for trigger purposes is the same as in the hadron calorimeter. A minion traversing the calorimeter gives a signal corresponding to $\approx$250 MeV of electromagnetic energy. The noise in a trigger segment is expected to be prohibitive for triggering on minions. The electromagnetic calorimeter will therefore only be used to search for energy clusters. A low energy threshold per trigger segment equivalent to $\approx$1.5 GeV should be feasible, as should be a total energy threshold of 10 GeV in the whole calorimeter.

4. The muon chambers represent a tool for helping the calorimeter in defining penetrating particles such as muons. In addition to the muon signal in the calorimeter, a correlated hit in the muon chambers could be required.

5. The luminosity monitor: Trigger signals are derived from the calorimeters. Back-to-back correlation of two energy clusters form the trigger of Bhabha events.

In order to build track-track or track-cluster triggers described above, we feed two sets of signals into 72x72 twofold coincidence matrices: i) the minion signals and ii) the energy cluster signals. The matrices are programmable; they can select any possible topology of two calorimeter signals. In response to background conditions, these correlations can be less or more restrictive. The output of these correlation matrices forms the input to the Level 1 trigger logic, where additional coincidence requirements, such as the number of tracks found

in the IC, are added. The Level 1 trigger logic then communicates with the Trigger Supervisor.

Another trigger condition is entirely based on energy requirements: the energy in one sector of the detector (e.g the sum of 6 or 12 segments) exceeds a certain threshold (perhaps 10 GeV), or two such sectors are in coincidence, but a lower energy level.

2.1.2 The Level 2 Trigger

The basic requirement of the Level 2 trigger is that the tracks in the TPC point to the bunch-crossing region. This can be done progressively within the 40 μs TPC drift-time, with the final answer available a few microseconds after that. The TPC trigger works on the rz projection using dedicated cathode pads on the TPC end-plates. Tracks with $p_t \geq 1$ GeV/c appear as essentially straight in the rz projection: the TPC trigger recognizes, in real time, straight tracks from the crossing point with a tolerance of $\Delta z \approx 10$ cm. Essentially any event with tracks coming from the crossing point will be accepted.

2.2 Readout of components

ALEPH is divided, for the purposes of readout, into sub-detectors. Each of these subdetectors is equipped with its own readout system and on-line VAX computer located in the experimental zone. The main data acquisition system (DAQ) transfers data from the sub-detector buffer memories to the Data Logger which forms part of the central on-line computing system.

Interaction between the readout system, the trigger logic, and the LEP machine signals is handled by the Trigger Supervisor. On receipt of LEP timing signals, this device activates the front-end electronics and, following a Level 1 trigger, inhibits further triggers until either the event is rejected by a Level 2 trigger or readout is completed into the buffer memories and the front-end system is reactivated. De-randomizing buffers, three to four events deep, will be situated before the Event Processor (see below).

FASTBUS is used exclusively for the data transfer and fast control; CAMAC is relegated to slow control and is interfaced to the DAQ via FASTBUS. Most sub-detector readout systems will be built in FASTBUS. All, however, will connect to their computers and the main DAQ via FASTBUS.

As an example, the TPC readout system is designed now according to FASTBUS specifications. It handles the analogue information derived from about 50 000 channels connected to anode wires and cathode pads. In each of these channels, the collected charge is digitised by FADCs (effectively 9 bits) in time slices of 100 nsec over the full drift time (40 μsec) of the TPC. This system is built into about 120 FASTBUS crates, each of which consists of several Time Projection Digitizers (TPD), a Time Projection Processor (TPP), and other modules. Each TPD has 64 channels of FADCs, zero-suppression logic, and sufficient fast access buffer memory to accomodate two events. The TPP module provides the necessary local intelligence (M68000 microprocessor, 32K RAM, and

8-16K EPROM) to validate these data and organize the flow into the main DAQ. In the event of a Level 2 trigger, the process of zero-suppression and scanning these memories is begun, taking about 2.5 msec for completion.

The estimated average size of a typical physics event (20 charged tracks and 20 photons) is 100 Kbytes, which can be subdivided as follows: TPC 80K bytes, electromagnetic calorimeter 5K bytes, hadron calorimeter 5K bytes, trigger and remaining elements 10K bytes.

The DAQ will be designed to accomodate an instantaneous data rate of 10 Mbytes/sec (i.e., two events separated by 10 msec.), but a mean rate of only 1 Mbyte/sec. The functional requirements of the DAQ as agreed within the Collaboration are appended as Annex I.

2.3 Event Processor

The design of the ALEPH data acquisition system incorporates an Event Processor (formerly referred to as Level 3 Trigger) which acts on complete events. It is a general purpose processor and will run a crude event analysis program. Its purpose is twofold: it can be used to filter the events which passed the Level 2 trigger; it can also be used for the classification of physics events, enabling them to be steered to different data logging streams.

The hardware realisation of the Event Processor is not yet clear. The required computing power could be provided by an array of 3081E emulators or other processors. The number of such processors depends on the rate of events at the output of the Level 2 trigger, and on the time needed to process one event.

2.4 Central On-line Computer System

The tasks to be performed by the central on-line computer system are:

1. **Overall control of the experiment.** This includes the display of monitoring information, control of the Event Processor, information exchange with the sub-detector computers, and initialisation of the FASTBUS readout system.

2. **Data recording.** The data from the detector, having passed through the Event Processor, have to be recorded on some mass-storage device. Our preferred solution would be to send the data via a dedicated high-speed link to the computer centre for central recording. If central recording is not possible, the data might be written on high-density magnetic tapes (6250 bpi) or hopefully on a higher density medium like Video disks, if available from industry before the data-taking.

3. **Central data base.** The central data base will contain the master copy of all the software needed by front-end microprocessors, sub-detector com-

puters and the Event Processor. It should also store calibration constants, the run book, and any information relevant to the running and maintaining of the experiment. The data base must be accessible by all computers at the experiment, and also via the network in order to allow outside institutes to keep up-to-date for the analysis.

4. **Sample analysis and playback of events.** The central on-line computer system should be able to run the full reconstruction software to allow samples of the events to be analysed directly, which is useful for the monitoring of the apparatus. High-resolution graphics will be needed to access the resulting information.

The tasks mentioned make quite diverging demands on the system they run on. Points 1) and 2) need a real-time environment, points 3) and 4) are less time-critical, but they need a rather big storage space and large amounts of CPU time.

We intend to purchase the hardware for the main on-line computers at the latest possible date, in order to obtain the highest performance for the limited amount of money available. It is clear, however, that these machines will be members of the VAX family of computers, as the software will be developed over several years in advance on VAXes.

2.5 Communication with the LEP machine

We foresee the following types of signals being exchanged between ALEPH and LEP:

1. Information from LEP on the status of the LEP machine, including:

   - The beam energy
   - The number of bunches
   - The beam currents
   - The beam lifetimes
   - The luminosity, if available
   - The vacuum pressure(s)

   The parameters should be sent periodically, perhaps every ten seconds.

2. A precise timing signal from LEP for every bunch crossing.

3. Information sent from ALEPH should include counting rates sampled over about one second and possibly also an estimate of the luminosity. This should help LEP to monitor and optimize beam conditions.

2.6 Costs and sharing of costs of the Data Acquisition

The cost estimates quoted in the Technical report were: 1.2 MSFr for main computers paid by the whole collaboration, and 0.95 MSFr for data acquisition paid by CERN, respectively. We recall that the cost given for data acquisition include only the central part of the overall system; the front-end microprocessors and the sub-detector computers are included in the cost of the different sub-detectors.

The costs for operation and maintenance of the main computers are the responsibility of CERN. Maintenance costs for the data acquisition will be charged to the general operation budget of ALEPH, i.e. computer maintenance paid by one institute will be accounted for in evaluating this institute's participation to the general operating cost.

3. OFF-LINE ANALYSIS

3.1 Software development for on-line and off-line programming

Many features of the software for on-line and off-line applications will be in common. We will therefore consider all those applications which are not directly connected with data acquisition. Even this boundary is not strict since we will use a common man-machine interface, a common graphics package, and common tools for software development and maintenance.

3.1.1 Environment for software development

Most of the institutes involved in ALEPH wish to participate in the on-line and off-line software development. There will be projects involving people from different institutes and projects migrating from an institute to another. The majority of physicists will work at their home institute, coming to CERN for limited and varying periods of time, but still willing to continue their work. This will be facilitated by a uniformity of development tools. Most of the institutes want to have available at home the totality of the code written and the updates must be carried out in the shortest possible time.

The ASSET (the "Aleph Standard Software Environment and Tools") is a standardized computer environment, i.e. what a user, logged in to a terminal, sees as available to him. It provides a standard set of tools and facilities to communicate with all other ASSET's of ALEPH. Every user working on software development or data analysis has access to at least one ASSET via a terminal.

The proposed solution can be implemented now but is flexible enough to include future developments in hardware and software. Detailed information on the choices for computer hardware and operating systems, common software tools and a strategy for network connections can be found in Annex II.

The main decisions are :

- DEC VAX-11 and VMS operating system is the primary configuration.
- IBM (compatible) computers and VM/CMS operating system will be supported, possibly with reduced functionality.
- Personal Work Stations will be considered at a later stage.

3.1.2 Major tasks of software system

In this section we give a brief overview of the major tasks that shall be accomplished by the software system.

- displays of detector status, machine status and run conditions for current situation and accumulated for a period of time,
- monitoring of detector output, logging of key parameters and deficiencies,
- sample event reconstruction with different sampling modes for individual detector modules, displays of histograms or full event displays,
- (partial) event reconstruction in the Event Processor to reduce background and to flag interesting events,
- determination of calibration constants from special calibration files or partial reconstruction in detector modules,
- off-line event filtering,
- pattern recognition of tracks and energy clusters in various parts of the detector,
- reconstruction of tracks and showers, reconstruction of vertices, particle identification in TPC,
- event displays with interactive manipulation of presentation, event processing and specific selections for event scanning and detailed studies,
- efficient batch version of reconstruction program for bulk data processing,
- Monte Carlo generation of physics processes and background,
- tracking of Monte Carlo events through detector with simulation of detector response and digitising,
- processing of Monte Carlo events through full reconstruction program and displays of reconstructed and truth information,
- analysis programs for histogramming, statistical analysis and manipulation of data.

3.1.3 Organisation of software development

Careful planning is necessary to cope with the specific problems of the ALEPH software :

- large project with many options,
- many people working decentralised on development,
- long life-time of the project,

- continual evolution of physics, LEP machine, detectors, algorithms and computer hardware,
- many people are going to use the software, therefore well organised and well documented software is required.

The development of a software system can be separated into several distinct phases :

- definition of requirements for software project,
- specifications,
- architectural and detailed design,
- implementation of software (coding),
- operational testing and installation,
- production, maintenance and evolution.

The end of each phase represents a milestone in the development of the software. Phase 1, i.e. the definition of requirements has been finished. The complete document is included as Annex III. The time-table for further phases in the development cycle can be given as follows:

late 1984	specifications
early 1985	architectural and detailed design
early 1986	implementation of code useful for detector module tests
1987	complete implementation of code
1988	operational testing and installation, processing of calibrations and cosmic ray data

3.2 Computing capacity required for off-line analysis

The following estimate represents our present understanding of the off-line analysis situation.

1. Assumptions on tape writing

Average event length	100 kbytes
Event writing rate	1 Hz
Storage medium (a)	6250 bpi tapes
Events per tape	1000
Data taking per year	3000 hours
Tapes per year	10,000

(a) The 6250 bpi magnetic tape is used as an example of a data storage medium.

2. Assumptions on off-line analysis

The off-line analysis load depends, among other things, on the ratio of good events to background events. This ratio will vary with time be-

cause of improvements in the LEP luminosity and in our understanding of the detector and trigger conditions. In the subsequent discussion we assume (in the early period of LEP operation) a ratio of good events to background of 1:5.

Events per year submitted for analysis	10^7
Time per event for fast software filter	1 sec. CP IBM 168
Reduction factor from fast software filter	5
Events per year to be reconstructed	$2 * 10^6$
Time per event for reconstruction (a)	25 sec. CP IBM 168
Multiplication factor (b)	4
Total time needed per tape (1000 events)	6.5 hours CP IBM 168
Total time needed per year	65,000 hours CP IBM 168
Fraction of time needed at CERN (c)	1/3
Total time needed at CERN per year	22,000 hours CP IBM 168 (= 4 IBM 168 equiv.)

(a) Reconstruction should be relatively straightforward in the ALEPH detector. Hence we extrapolate from PETRA/PEP detectors linearly with the amount of information per event.

(b) This multiplication factor accounts for Monte Carlo generation, Monte Carlo event analysis, calibration, DST analysis, base-load per programmer and overheads. The factor 4 is based on experience with several large experiments.

(c) At the start-up of LEP we expect most of the off-line analysis to be concentrated at CERN. The quoted fraction will be aimed for after the running-in period when things have stabilized and the computer power of IN2P3, Heidelberg, Munich, RAL and Saclay can be fully used. We expect, however, the initial load on CERN to be partly compensated by a lower LEP luminosity.

3.3 Computing capacity available for off-line analysis

These estimates are based on computers available now or in the very near future. Several laboratories will have access to the computing centres of IN2P3 and Rutherford Laboratory, and their contribution has been included into the ALEPH share. Small computers like VAX 750/780 and IBM 4341/4361 are normally not used for large scale batch processing and are omitted from the table. They will be used for program development/maintenance and for event scanning, however.

Collaborator	Computer (1984)	ALEPH share	168 factor	168 equiv.
CERN	IBM 3081/K	0.15	3.50	0.50
	SIEMENS 7880	0.15	2.70	0.40
	CDC 875	0.15	3.00	0.45
IN2P3	3081/K or CYBER	0.15	3.50	0.50
HEIDELBERG	IBM 3081/D	0.12	3.00	0.36
MPI MUNCHEN	AMDAHL 470/6	0.20	1.50	0.30
	SIEMENS 7880	0.20	2.70	0.54
RUTHERFORD	ICL ATLAS 10	0.15	6.00	0.90
	IBM 3081/D	0.07	3.00	0.21
SACLAY	IBM 3081/K	0.05	3.50	0.17

In order to compare these figures with our production and analysis computing requirements, reasonable numbers have been assumed for the ALEPH share, as well as the CP power of the computer relative to the IBM 168. Both these numbers are subject to considerable uncertainty. The total weighted figures sum up to about 4.4 IBM 168 equivalent. The average number of CP hours is about 120 per week, or 6000 per year. Multiplying these figures together one gets a total of 26,000 IBM 168 hours available, which is smaller than the expected total need of 65,000 IBM 168 hours by a factor of 3. Furthermore, in order to achieve results in an acceptable time during the initial phase of the experiment, CERN must provide a larger share of the computing power.

We therefore request that the central batch capacity at CERN be increased by a factor 4 by 1988.

Whether this increase is partly due to the use of emulators is not important to us. We would accept any solution provided the user is not unduly constrained by a particular choice and it is operated centrally.

3.4 Graphics

The complexity and the high information content of events originating in the ALEPH detector call for a graphical representation. The basic function of graphics will be to provide interactive display facilities to enable visualisation of tracks, energy deposits in the calorimeters, etc.

Graphics will be used at all stages of the program life cycle. During program development it will help debugging. The efficiency of pattern recognition

algorithms will be studied and improved using interactive event displays. At the time of first data taking event displays will be needed to verify program performance under realistic conditions and to hand-reconstruct complex events that may not be treated properly by the early version of the reconstruction program. During physics analysis graphics will be used for interactive histogramming and statistical analysis of events.

The graphics software should be able to cope with a variety of graphics terminals of different complexity ranging from low-cost graphics terminals up to graphics workstations supporting local 3D image transformations.

The rapid evolution of graphics hardware makes it difficult to decide now which system is to be adoped. It is probable that within a few years from now systems with the performance of the MERLIN VAX-Megatek combination will be available at much reduced cost. Personal work stations equipped with high resolution screens provide some of the required graphics functionality at relatively low cost. Colour terminals should be considered seriously since they offer extensive possibilities of association and representation of additional information.

4. NETWORKS

Flexible and reliable telecommunication links are essential for a distributed program development effort. They facilitate the efficient use of computing resources and improve communication between members of the collaboration.

Several proposals for realistic implementations have been presented by working group 2 in the report on "Computing at CERN in the LEP era". We fully support these proposals for standardized facilities which should be available in due time.

We intend to implement ALEPHNET as an interactive high-level communication protocol on top of existing, commercially available networks, to shield the ALEPH physicist working on his terminal from the inhomogeneity and variability of the current telecommunications environment.

The requirements of ALEPHNET arise from the large programming effort that such a wide collaboration has to undertake in order to have the software ready by 1988. File transfer and mail facilities are essential, but also remote access and real time communication (both task-to-task and user-to-user) have proven important in large distributed software projects in other fields.

For the on-line system only VAX computers will be used. For the offline computers, out of the 25 laboratories in the ALEPH Collaboration, 16 own a VAX

or have access to one, 5 have access to an IBM and to no VAX, 4 have no interactive computer facility. In view of this we have chosen to base ALEPHNET on DECNET, the standard network software of the VAX operating system, plus some gateways to connect the IBMs of some laboratories.

All VAXes at CERN, including those in the experimental area, will be connected via DECNET using local ETHERNET lines. These local area networks are then connected to the CERN backbone, which in turn provides access to outside laboratories via gateways.

Connections currently available between CERN and outside laboratories have been implemented using a variety of ad hoc solutions, and we expect the situation to continue to develop in this way for some time. We hope that standardized solutions will emerge. The four LEP collaborations should collaborate in this respect since their needs are similar. We encourage the DD Division of CERN in its efforts to provide standardized networking facilities.

5. STRATEGY TO SEARCH FOR RARE EVENTS

5.1 Introduction

We were asked to state our strategy with respect to Data Acquisition and Data Analysis, to search for ten events of the type $Z^0 \rightarrow e^+e^-H^0$ out of 10^7 Z^0 events. In subsection 5.2 below, we discuss specifically this process. Here we wish to discuss a more general strategy of searching for rare events.

The ALEPH Data Acquisition system contains an Event Processor (section 2.3) which can be used for flagging specific event topologies. On the basis of such flags it will be possible to perform an on-line split of interesting events.

It is difficult to define here the split criteria of the Event Processor. For example, one might think of retaining events with two lepton candidates, which would already cover events of the type $Z^0 \rightarrow e^+e^-H^0$.

One would aim at a reduction factor of at least ten for this on-line split. A much larger reduction factor seems unrealistic if one wants to retain a variety of event classes. Still, a reduction factor of ten would suffice to reduce the number of events to be reconstructed to less than 10^6 per year. This production would be part of the planned Data Analysis programme of ALEPH, and hence is covered by the request for computing capacity already made (section 3.2).

A completely different scenario is the following: after having logged 10^7 physics events, a new physics question turns up, which was not foreseen in the

Event Processor split criteria. In this case, all the 10^7 events (most of which have not been reconstructed) have to be looked at in as short a time as possible.

To solve the problem, we propose a dedicated off-line split of the 10^7 events. Assuming that one event can be handled in one second IBM 168 CP time, this would take 2800 hours. Assuming further that a reduction factor of 100 can be achieved, the time to reconstruct fully the remaining 10^5 events is 700 hours. Altogether this effort involves a total of 3500 hours of IBM 168.

It is conceivable that this could be absorbed into the Data Analysis programme of ALEPH. However, it almost certainly would create a major problem if it had to be done fast. Notice also that the uncertainty on the estimated computing need is large.

As a consequence it would be desirable if ALEPH had access to additional processing power to cope with occasional peak loads in the data analysis. The way in which this capacity may be provided is not obvious to us. One possibility is the installation of a Processor Bank, for example at CERN, consisting of an array of 3081E emulators or other processors. We would encourage the CERN DD division, or development groups in other major computing centres, to study this problem and to propose solutions for handling occasional peak loads in data processing.

5.2 Data reduction for $Z^0 \rightarrow e^+e^-H^0$

We have generated a set of events $Z^0 \rightarrow e^+e^-H^0$ (m_H = 50 GeV) and processed them through our Monte Carlo program GALEPH. A set of hadronic $Z^0 \rightarrow q\bar{q}(g)$ events (q = u,d,c,s,t,b with m_t = 30 GeV) using the Lund program has also been generated.

The basic aim is to identify electron showers making use of the electromagnetic calorimeter and of some crude track information from the TPC. The program was based on a cluster-finding algorithm for the e.m. calorimeter, and a simple track-finding procedure using the TPC space points as derived from the pads.

The detailed cuts have been chosen as follows:

- Two clusters in the e.m. calorimeter, each with E>5 GeV.
- If $E_1+E_2 < 50$ GeV, then $|\cos\Theta_{12}| < 0.96$, where Θ_{12} is the angle between the clusters as seen from the origin.
- The recoil mass from the two clusters is < 60 GeV.
- Each cluster is associated with at least 4 space points obtained from different TPC pad rows, located within a road connecting the origin with the cluster position. The road width was ±5 cm in $r\phi$ and ±5 cm in rz.

With these cuts, 80% of the $e^+e^-H^0$ events are retained. The most prominent background is due to hadronic events where photons from π^0 decay simulate electron clusters. The suppression factor for hadronic events is 4×10^{-3}, hence

the initial sample of 10^7 events is reduced to 4×10^4 events. This number of events can be fully reconstructed without major difficulties, and then be exploited using the power of the ALEPH apparatus in all details.

The CPU time per hadronic event used for the selection of events was 0.5 seconds IBM 168 equivalent. For real events, we would expect this to take about twice as long because of the large amount of data to be read in per event.

An example of a $Z^0 \rightarrow e^+e^-H^0$ event as seen by the ALEPH detector is shown in figure 1.

6. RESOURCES NEEDED FROM CERN

The ALEPH Collaboration hopes to get help from CERN in the following areas of work :

1. **FASTBUS development.** As stated already in our reply to the report on "Computing at CERN in the LEP era", we hope that the general items will be developed or selected by the EP electronics group and supported by the EP electronics Pool as it is for CAMAC now. It is our understanding that the computer interfaces are in the hands of the DD/EE group and that the general purpose FASTBUS software will be available from the DD/OC group.

2. **Microprocessors.** The cross-software products should be available under VAX/VMS. Development of stand-alone microprocessors serving as test-stations for FASTBUS equipment, and of interfaces of microprocessors to FASTBUS are needed. We would appreciate efforts to standardize the use of microprocessors for slow control functions (monitoring of voltages, interlocks, gas flow, etc.).

3. **General Software Packages.** There are many areas where general software packages should be used. We think in particular on source code management, relational data base, communication software, command and menu packages, and graphics software.

4. **Off-line software development.** Development of software for ALEPH needs interactive computing from the very beginning. We intend to base our development mainly on VAX computers. A large centrally supported VAX at CERN is absolutely essential for ALEPH to serve as a continually available tool for program development and test for many users. Central support will be needed to provide for consultancy, program library maintenance, installation and introduction of new tools and packages for the many VAX computers used by ALEPH and other LEP collaborations.

5. **Processing power for data analysis.** ALEPH will have no possibility to provide its own computing centre at CERN. We therefore have to rely on adequate processing power offered by the CERN central facilities.

We ask specifically for an increase of the central batch capacity by a factor of 4.

We would appreciate studies on the installation of additional processing power, e.g. in the form of processor banks composed of emulators, to cope with occasional peak-loads in the data processing.

6. **Network facilities.** We ask for the support of DECNET for file transfer and electronic mail. We would appreciate all efforts to standardize the network facilities.

7. **Education and Training.** Cern should provide adequate means for education and training of users of new software techniques and hardware products. Qualified consultancy is equally important.

8. **New storage media.** In view of the large amount of data expected at LEP all efforts should be made to look for new storage media with larger capacity and better access than offered by high-density tapes.

   We would appreciate a study of the problem of central recording of the data.

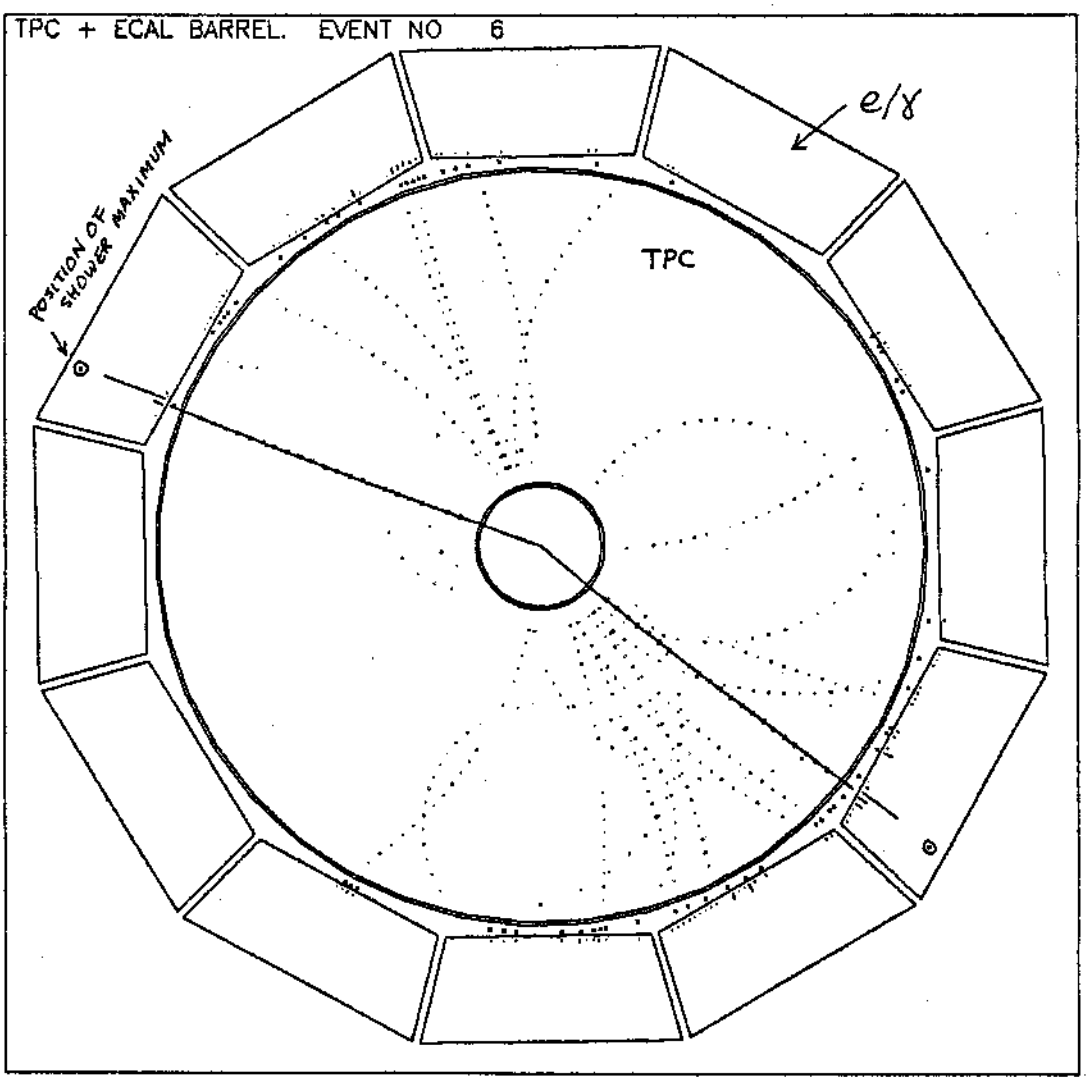

Fig. 1 An event of the type $Z^0 \to e^+e^-H^0$ with m_H = 50 GeV. The two electromagnetic showers are matched with TPC tracks.

ALEPH NOTE # 115

REQUIREMENTS FOR THE ALEPH DATA ACQUISITION DESIGN - DATA FLOW

1. INTRODUCTION

The evolution of both the hardware and software components of the ALEPH Data Acquisition System (DAQ) will take place in stages which can be readily identified. Progress through these stages needs to be adequately documented, with the completion of each stage marked by the issue of a summary paper which acts as an input reference for the next stage.

Stage 1 ends with the definition of the functional requirements of the DAQ.

Stage 2 will end with the specification of the technical solutions, hardware and software, proposed to meet these requirements.

Later stages will cover the detailed design, construction and commissioning of the various elements of the system.

This document is the result of several months of discussion within the ALEPH Data Flow Subgroup, and marks the end of Stage 1 of the Data Flow studies.

It lists the REQUIREMENTS of, and makes RECOMMENDATIONS about, the ALEPH DAQ Data Flow System.

Other essential parts of the DAQ, such as Local Area Networks, Data Bases etc, which have not been considered here, will be covered in future Stage 1 documents.

2. GENERALITIES

Sub-detectors

ALEPH is considered an integrated system of Sub-detectors such as the TPC, Electromagnetic Calorimeter, etc. Each major sub-detector will be equipped with its own VAX computer; several smaller sub-detectors may share one VAX. These sub-detector computers will be used for

MONITORING the performance of the sub-detector, obtaining data by SPYing on the data flow,

CALIBRATION of the sub-detector as an independent device,

CONTROL of the sub-detector system e.g. gas, H.T.

MAINTENANCE of the sub-detector electronics (i.e. fault finding and diagnosis)

Read-out of Sub-detectors

The read-out of each individual sub-detector, down to the buffered interface with the DAQ, is the responsibility of the constructors of that sub-detector. The DAQ rules apply up to, and including, this interface. They also apply to any computers attached to the sub-detectors, and to their connections to the system.

Each sub-detector will have associated with it a set of logical modules necessary to read the data from the data sources (front-end electronics) into the output data buffers associated with that Sub-detector. These modules will be referred to as Read-out Controllers and Event Builders.

A **Read-out Controller** initiates and controls the read-out of data from the data sources in response to signals from the Trigger Supervisor.

An **Event Builder** receives data from the data sources and assembles the correct event structure in a Buffer Memory.

DAQ Structure

The DAQ will have a classical 'tree' structure in which data from the Sub-detectors will be brought into a final single data stream.

That part of the DAQ concerned with Data Flow from the sub-detector buffers comprises :

a Trigger Supervisor
an Event Builder
a Buffer Memory
an Event Processor
the Host Computer
the Data Logger.

The **Trigger Supervisor** is the sole recipient of the LEP timing signals, Level 1 and 2 physics trigger signals, and other similar signals which affect the running of ALEPH as a whole. Depending on the kind of trigger received, it activates the relevant Sub-detector Read-out Controllers, or distributes the necessary clear signals. It also applies the system dead-time to the trigger signals.

Any Sub-detector which from time to time runs with independent triggers specific to that Sub-detector will also have to have a Trigger Supervisor. When the Sub-detector is running independently, this will perform the same function for the Sub-detector trigger as the Main Trigger Supervisor performs for the DAQ as a whole. Designers of Sub-detector electronics will be urged to use the same design of Trigger Supervisor throughout the system.

The role of the Sub-detectors in independent operating modes is returned to later, when partitioning of the DAQ is considered.

The **Event Processor** provides for more refined event rejection than is possible at Levels 1 or 2, and may also be used to preselect and flag event topologies of particular interest before passing them on to the Data Logger.

The **Host Computer** is the main on-line computer, situated in the surface control room. It is the control computer for the DAQ, and as such has a privileged position in comparison with the Sub-detector Computers.

The **Data Logger** is the device which writes the data to the storage medium (eg. tape). It will probably be a smaller computer than the Host Computer.

The Read-out Controllers and Event Builders, although logically distinct, do not need to be physically distinct modules, and could be, in whole or in part, software.

SPY modules will be used to extract data without affecting either the data or the transfer speed.

Buffer Memories are used where necessary to de-randomise data flow through the system.

All the computers in the DAQ will be connected to a Local Area Network. This network should allow remote log-in, general file transfer and task-to-task communication.

Data Taking

The Level 1 and Level 2 trigger logic delivers signals to the DAQ. The DAQ then communicates with the sub-detectors and controls the readout of events. Events are written to tape by one device only, the Data Logger. Sub-detector computers may 'Spy' on the data.

3. REQUIREMENTS

"Mandatory" requirements have been agreed by the Data Flow group; the "recommended" ones represent a consensus opinion within the group. Where relevant, the same point may appear in more than one sub-section.

Figures 1 and 2 should be taken as a visual help in understanding the concepts outlined in this paper.

3.1 Global Requirements

MANDATORY

1. The DAQ must be able to accept triggers, read out the Subdetector information, and output the data to the storage medium.

2. The DAQ must be able to take calibration data in all sub-detectors, and output that data to the storage medium.

3. The DAQ must be capable of generating and accepting 'de-bugging' data from sub-detectors.

4. Operations 1, 2 and 3 above must be able to take place simultaneously in different parts of the detector.

5. FastBus must be used for Data and Control within the DAQ.

6. The DAQ must be able to write data to several independent recording streams.

7. All sources of data must format their information according to a common data structure.

3.2 Rates and bandwidths

MANDATORY

1. The DAQ must match the following basic figures for rates:

Beam crossing frequency	45 KHz
LEVEL 1 trigger frequency	<500 Hz
LEVEL 2 trigger frequency	<10 Hz
Average event size	100 KBytes

2. The DAQ must be able to cope with a peak data rate of 10 MBytes/sec.

3.3 System partitioning and reconfiguration

MANDATORY

1. The DAQ must be capable of being partitioned into logically independent sections so that many users can work independently on different parts of ALEPH.

2. A PARTITION is any subset of the DAQ which has been configured to respond independently to triggers.

3. The minimal partition is all or part of a sub-detector system, which must include a Trigger Supervisor, Read-out Controllers and local Event Builder together with a control computer.

4. One or more sub-detectors responding to the same trigger must be combined into a single partition. The trigger may be internal or external to the partition, and must be handled by the partition's Trigger Supervisor.

5. All partitions must contain a control computer, which cannot affect any other partition.

6. The whole DAQ running with normal physics triggers must also be a partition. It would normally include all the Sub-detectors and the Main DAQ system.

7. This partition must be controlled by the Host Computer only.

8. During normal data taking the Host computer must not increase the dead time in the data flow to the Data Logger.

9. When the system is reconfigured into several partitions, each partition must be logically independent of every other partition.

10. Only the Host Computer can change the configuration of the partitions into which the system is configured. It must keep a full description of these configurations.

11. It must be possible for several partitions to time-share the main data channel to the Data Logger, so that logically each partition has its own data channel and Data logger. It must be possible to write separate tapes for each partition.

12. Reconfiguration of the partitions must be made by software only, without any recabling being necessary.

13. Reconfiguration operations must only affect those partitions being reconfigured.

Recommended

1. While the partitions provide 'vertical' subdivisions, the DAQ should also have independent 'horizontal' slices, i.e. layers must be identified which have unique input from lower levels and unique output to following levels. This is necessary for independent testing and trouble shooting within the DAQ.

2. The reconfigurations of the system should be chosen in such a way as to allow for the maximum flexibility regarding the interconnections of the Sub-detector Computers.

3.4 Event rejection

MANDATORY

1. Processors in sub-detector read-out systems, and sub-detector computers, must not reject events. All events accepted by the Level 1 and Level 2 Trigger Logic must be passed on to the Event Processor.

2. Any device which detects abnormal conditions (e.g. time-outs, buffer overflow etc) must always flag the data. It must be possible for the partition control computer to suppress the onward transmission of faulty data, but not of the fault flag.

3. Event rejection after Level 2 acceptance, for whatever reason, must be taken in one place only, under control of the Host Computer.

3.5 Triggers

MANDATORY

1. All interaction between the Level 1 and Level 2 Trigger Logic and the sub-detector read-out must be handled by the main Trigger Supervisor.

2. All sub-detector systems which need independent triggers for calibration purposes must have a dedicated Trigger Supervisor to control their read-out for that specific calibration purpose.

3. If two Subdetectors use the same calibration trigger (and both have a Trigger Supervisor), they must merge into a single partition governed by only one of the two Supervisors.
   There must be one and only one active Trigger Supervisor in each partition.

3.6 Data paths

MANDATORY

1. Fastbus must be used for data and control paths.

2. It must be possible to transfer data from any downstream level in the DAQ to any upstream one, and to read it back; the most downstream level is the tape and the most upstream ones, as far as the Data Flow design is concerned, are the buffers in the various sub-detector read-outs.

3. Any further downloading, into the front end electronics, for example, must be done by the sub-detector itself, and under the control of the sub-detector computer.

4. All the sub-detector computers must be able to reach any part of the hardware via the DAQ. As a result, all CAMAC in the system must be interfaced via FastBus.

Recommended

1. Special links for data transfer should be adopted only if the need arises due to limitations in Fastbus itself.

2. When similar devices are clustered together for parallel operation a Supervisor should be employed to control their communication with the preceding and following sections of the DAQ system, so that they appear externally to be one device.

3. Protocols based on levels rather than pulses should be implemented to improve reliability and minimize dead-lock situations.

3.7 Buffers

MANDATORY

1. The sub-detector buffers must follow a common design philosophy for all sub-detectors. They must all store the same number of events, have the same mode of operation, and the same interface to the DAQ.

2. All buffer memories must be extendable, so that either the number of events, or their size, can be increased if necessary.

Recommended

1. Abnormal data conditions, like the non-zero-suppressed full TPC read-out, are not expected to be handled by the buffers.

2. A modular design of de-randomising buffer, rather than a single large memory, is to be preferred. Each sub-detector will have its own modules.

3.8 Sub-detector computers

MANDATORY

1. The computer running the software needed for a particular sub-detector is referred to as the sub-detector computer. Each sub-detector computer must be able to obtain the complete data pertaining to that sub-detector. In SPY mode it will not necessarily be able to obtain every event.

2. If a sub-detector computer fails, another sub-detector computer must be able to perform its tasks, even if this means slowing down the performance of the tasks of these computers. This re-configuration must be done with software.

3. A sub-detector computer must not interfere with data flow in any partition other than the partition of which it is the control computer.

3.9 Event Processor

MANDATORY

1. The Event Processor must be anticipated in the DAQ design from the beginning.

2. The DAQ must function whether or not the Event Processor is physically present.

3. The Event Processor must be located in the surface counting room.

4. The Event Processor must be able to run off-line software without translation i.e. the standard language selected by the off-line group must be available.

5. If the Event Processor is not present, it must be replaced by a "dummy" hardware device which passes all the events to the next level downstream in the data flow.

6. It must be possible to simulate the function of the Event Processor by software in the Host Computer.

3.10 Trace-back facility

MANDATORY

1. The DAQ system must include trace-back facilities for 'trouble-shooting' throughout the whole DAQ.

2. Common de-bugging modules must be used throughout the system, including those parts of the sub-detector electronics which are implemented in FastBus.

Recommended

1. A control path independent of the main data channel should be implemented for general communication and fault-finding.

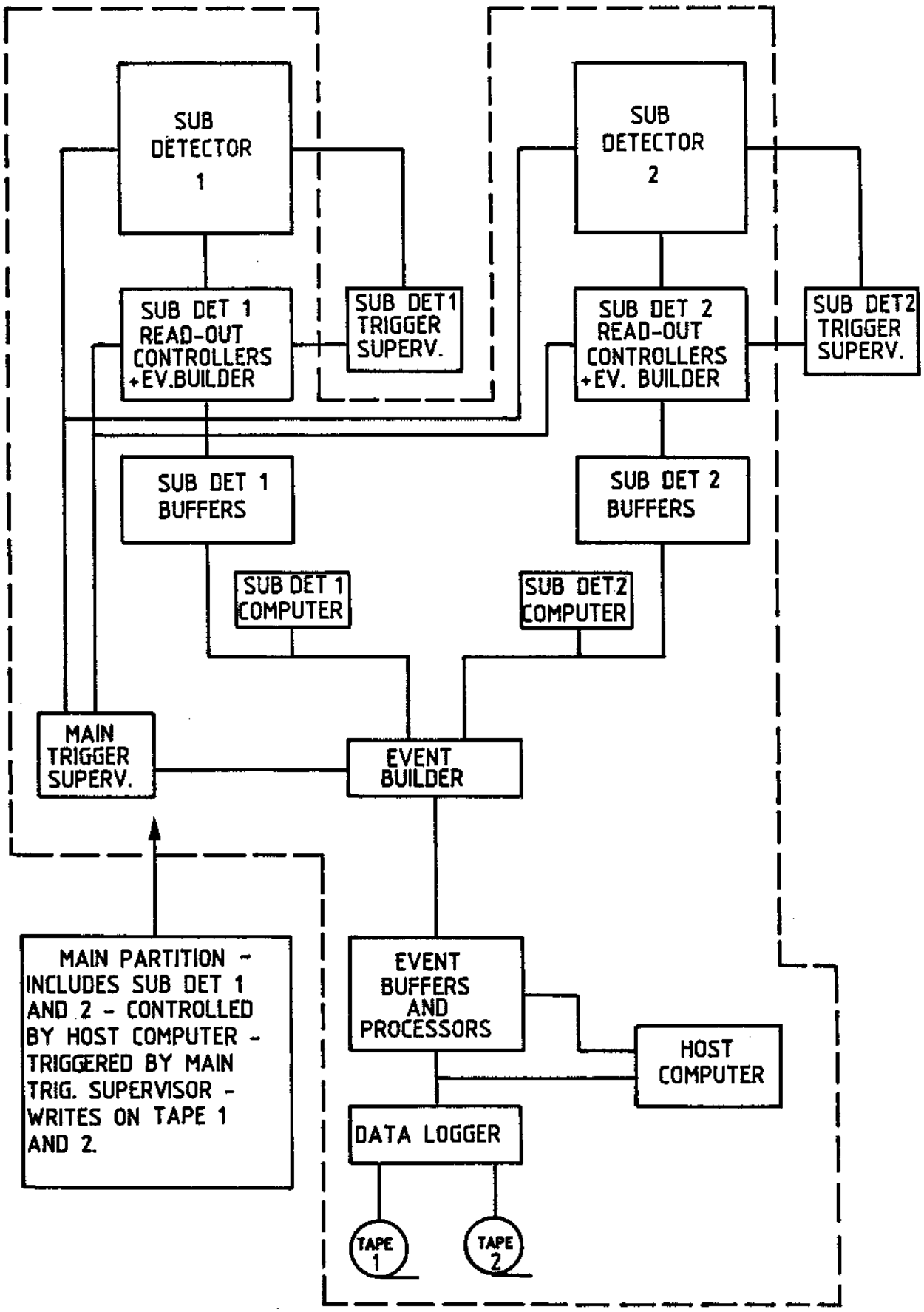

Fig. 1 LOGICAL SKETCH OF MAIN DAQ SYSTEM

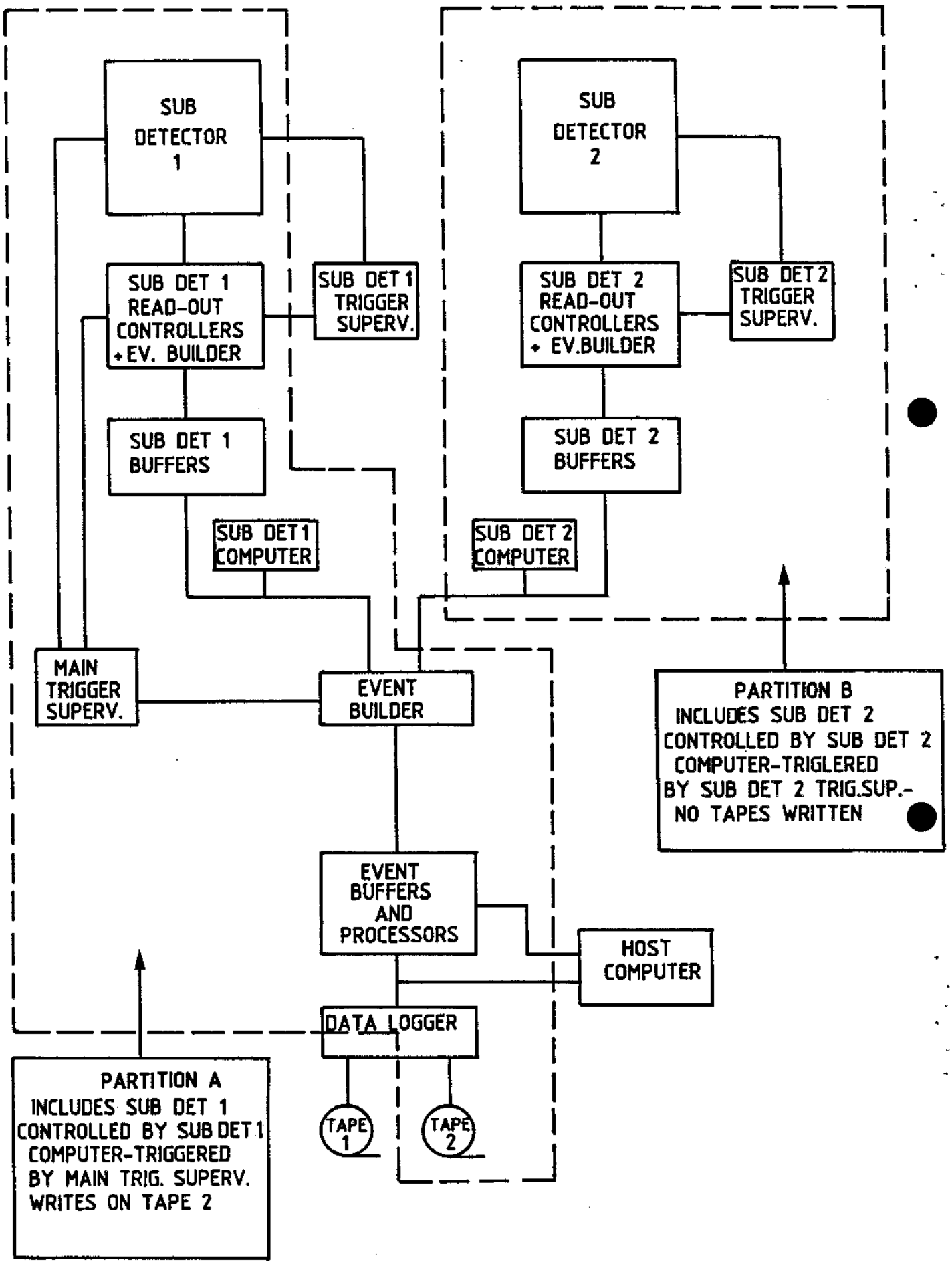

Fig. 2 LOGICAL SKETCH OF DAQ FOR A TWO PARTITIONS EXAMPLE

Aleph Note # 112

Date : December 19, 1983
Distribution : ALPNOT,SOFTWR
Keywords : SW
Authors : see footnote

ASSET

The ALEPH Standard Software Environment and Tools

Abstract :

The environment for distributed software development is described. Computer hardware and network connections are based on existing systems with particular emphasis on future developments. An overview of required software tools is given with further details to be defined later. Implications of distributed software development are discussed.

Authors : S. R. Amendolia, F. Dydak, F. James, G. Kellner, F. Liello, P. Palazzi, J. F. Renardy, A. P. White

1. INTRODUCTION

The idea to use a standard software environment for the development of software for ALEPH was first presented by P.Palazzi [1]. Other suggestions [2,3,4] were presented and a detailed discussion took place at the meeting in Pisa in May, 1983. The present note describes the solution accepted at this meeting.

The following statements outline the scenario for software development in the ALEPH collaboration :

- Most of the institutes will participate in the on-line and off-line software development effort.

- The majority of physicists will work on their project mainly at their home institute, coming to CERN for limited periods of time, but still willing to continue their work at home or at CERN.

- Efficient communication between various teams working on software development through electronic mail is absolutely essential to keep everybody informed about various discussions, decisions, open questions and problems, changes in design specifications, changes to code and documentation, etc.

- Software modules and packages will be developed by several teams at different places to be combined into a complete software system. A common set of tools and standards for program development will ease the problem of integration.

- Most of the institutes intend to have available at home the totality of the ALEPH application software, data files, calibration, detector positions, etc. Access to centrally maintained master file copies of information must be remotely available in the shortest period of time.

- The selected hardware and operating system must facilitate development and use of common packages for on-line and off-line programs.

- The proposed solution must be open-ended in order to accomodate new developments and evolution in availability of tools and hardware.

2. ASSET, THE ALEPH STANDARD SOFTWARE ENVIRONMENT AND TOOLS

The ASSET is a local computer environment, i.e. what a user, logged in to a terminal, sees as available to him in a fully standardized fashion. It consists of one of the selected computers and operating systems, a standard set of tools and facilities to communicate with all other ASSET's of ALEPH in a standardized way.

Every user, working on software development or data analysis for ALEPH, has access to at least one ASSET via a terminal.

All ASSETs with the same hardware and operating system have the same functional capabilities, they are interchangeable. Some specialized features will logically and physically be assigned to specific ASSETs, e.g. management of a specific project, masterdom of a Database, fancy graphic station, etc. Different functionality may be available for ASSETs of different configuration.

2.1 Computer hardware and operating system

1) DEC VAX-11 and VMS operating system is the primary configuration.

The main reasons for this choice are :

- The survey showed that a large fraction of the laboratories already have a VAX (their own or shared with other groups) or will be able to get one in the near future. The use of the VMS operating system assures compatibility with other potential users of the same computer.
- The VAX has been chosen as the standard on-line computer by ALEPH.
- The concept of the "family" of VAX computers ensures that the programming environment and tools look absolutely identical to the user irrespective of the particular model that is being used, and future developments are foreseen to follow exactly the same line.

2) IBM (compatible) computers and VM/CMS operating system will be supported, possibly with reduced functionality.

The main reasons for this choice are :

- Several laboratories have access to an IBM (or the compatible Amdahl-Fujitsu-Siemens) computer but may not have access to a VAX.

- IBM will certainly be one of the main computers for bulk batch production later on. Development and verification of programs on this computer should facilitate the transition phase from development to production.

- VM/CMS offers many of the features for interactive usage which are similar (not identical, unfortunately) to features available on VAX.

The reduced functionality implies that not all tools or procedures may be available. This may be due to the fact that certain tools are not transportable (but may not be required on every ASSET) or due to the fact that large IBM mainframes are not particularly suited for certain applications (e.g. interactive graphics).

2.2 Software tools

Apart from the actual hardware and basic operating system of the ASSET, there are a large number of important facilities which must be available to the ALEPH programmer to allow him to participate in the distributed development of software. These software development tools, which are not necessarily independent of each other, may be classified roughly as:

1. Tools specific to a software design methodology are discussed in a separate report on this subject. Depending on the methodology chosen, this may impose constraints on all the other tools.

2. It is recognized that the availability of UNIX tools is desirable in a software development context. A number of products have appeared on the market, that allow UNIX to run on top of vendor's operating systems (VAX/VMS, IBM/VM, and Apollo/Aegis, to name three interesting ones). If such a solution were adopted for ASSET, this would immediately give us access to a number of unified tools such as:

   - Shell, a powerful command language.
   - Full screen editors.
   - Document preparation tools (NROFF, TROFF) and associated spelling checkers, equation formatters, etc.
   - File manipulation tools.
   - and others, including those necessary for software design methodologies based on UNIX

3. Mail, file and message transfer, and remote login facilities are discussed below in section 2.3.

4. Utility program libraries are independent of ASSET to first order, and are not discussed here. Separate studies should be prepared on these important topics:

- Memory management.
- Data models and data base management.
- Graphics.
- FORTRAN extensions, such as bit handling, copy utilities, vector and array handling, mathematical functions.

5. The most important tool is that to be used for the maintenance and distribution of the source code, a task which has up to now been performed by PATCHY. It is generally recognized that the currently available PATCHY is not well adapted to modern methods of software development, and the requirement that our source code manager be portable leaves us with only two choices: To work for the development of an improved PATCHY, or to use a commercially available product. At least one such product already exists and is apparently suitable.

6. Documentation is one of the critical aspects of software development and tools in this area will be very important.

7. FORTRAN tools are those which understand the syntax of FORTRAN and can help in checking things which do not turn up in normal compiler diagnostics, load maps, etc. They can provide considerable help in writing good and reliable FORTRAN. These tools include:

   - The interactive debugger, which clearly must be closely integrated into the operating system itself, and its availability on VAX, IBM/VM, and PWS is an important consideration in the choice of ASSET machines.
   - A code verifier, which checks that code and documentation conform to the agreed set of conventions and standards.
   - Program cross-reference analyzer.
   - Code beautifier.
   - Syntax-oriented editor.

   Excluding tools specific to a design methodology, as well as those provided by an operating system like UNIX, we can consider three different approaches:

   - **Commercially available toolsets** such as SOFTOOL should probably be considered, even though they cost money.
   - **Public-domain toolsets** are of course cheaper, but we know of only one serious candidate: TOOLPACK/IST, developed at the University of Colorado and soon to be distributed by Argonne. It is in fact a whole UNIX-like programming environment, and could prove interesting if and when it exists.
   - We could also **develop our own tools** around the CERN-developed FORTRAN syntax-analyzer FLOP. The existing FLOP is mainly a code beautifier, with automatic indenting of DO-loops and resequencing of statement numbers, but it is a powerful framework for

performing other operations on FORTRAN code, such as editing and verification of standards.

2.3 Communications between ASSETs

This section describes our present view of communications between ASSETs and should be used as a guide for initial implementation. Since networking is an area of rapid development and better solutions for our needs may appear in the future, this document will be revised accordingly.

ALEPHNET is forseen as an interface to shield the ALEPH physicist working on his ASSET from the inhomogeneity and variability of the underlying data communication environment. It will be implemented as an interactive high-level communication protocol using standard network facilities, to be seen as a common feature of all ASSETs.

Facilities needed

In order to provide the services described in the introduction the network connecting the ASSETs should provide the following facilities:

1. Mail
   Mail is not a subset of file transfer. A mail protocol should include facilties for the user (reply, archive, recall..), a powerful distribution list mechanism and tools to update the user directory.

2. File transfer - File access
   File transfer is the possibility to copy files through the network. File access is the possibility to search for remote files and use them without copying, it is useful for random access (data bases) or for looking at small parts of large files.

3. Remote login
   Any user of an ASSET can, from his terminal, log into any ASSET in the network. This is used for scanning his own mail when not at home, for accessing specialized features not available on the current ASSET and to check remote files or programs for validation. This facility is also useful for a better distribution of the resources because it allows a user of an overcharged machine to use a machine temporarily underloaded thanks to the ASSET standardization.

4. Real time communications
   The two flavours: Task-to-task and Person-to-person have proven to be of the highest importance in large distributed software projects in other fields. The first one is mandatory in online applications.

5. Job submission - output retrieval

Used for running batch jobs on remote machines either for a better resource utilization or for production on non ASSET machines.

Network software

It has been chosen to base the communications between ASSETs on DECNET.

For the IBM ASSETs, a gateway policy has been defined: One solution can be to attach a gateway to each IBM. These gateways can be small PDPs connected to the network and running standard DECNET software and some software emulating an IBM terminal concentrator. This software already exists and allows any terminal connected to any computer in the network to be seen from the host IBM as a standard terminal hardwired to a terminal concentrator. The only drawback of this solution is that a terminal actually hardwired to an IBM is not able to see the network. This minor problem can be solved connecting the local terminals to the IBM through the gateway. Another solution for the laboratories who have chosen to join the IBM sponsored network called EARN, to which the CERN IBMs will also be connected, would be to implement a single gateway between the rest of the network and the EARN subnetwork possibly using the GIFT protocol. This would, however, give a reduced functionality with respect to the rest of the network because EARN has a reduced potentiality with respect to DECNET, allowing only message transfer and remote job submission.

Network description

All ALEPH machines in CERN will have access to a communication server running DECNET and acting as a gateway between the CERN-wide network and the various national networks. This communication server should also have a connection with the public X.25 network to allow for occasional connections that do not justify the expense of leased lines and as a backup resource in case of failure. It can be either a logical section of a general purpose VAX or a dedicated small computer down-loaded from any VAX in the network. We expect that the CERN wide network so defined will run on the transport facility available at CERN at the time of its implementation (e.g. the CERN backbone). The on-line computers will be connected in a local area network, integrated into the CERN-wide network, also running DECNET on an ETHERNET physical layer or some even faster media.

Great Britain and Italy already have their national research networks, the first, JANET, is based on UK protocols, the latter, INFNET, on DECNET. The UK laboratories are now studying the possibility of running DECNET on the JANET hardware. This would mean that 10 of the ALEPH laboratories could be connected using DECNET with the only expense being the purchase of the interfaces to connect the communication server to the national gateways at CERN.

French laboratories already have leased lines to CERN and a fraction of the bandwidth could be reserved for the DECNET traffic.

German laboratories are planning to join EARN running on leased lines paid for by IBM. On the other hand the German public data network DATEX-P is available and so both solutions mentioned above for connecting the IBMs are possible.

For Greece a packet switching network, EURONET, is available. For Denmark and China communication solutions are still under investigation.

Other networks, especially CERNET and SNA, are currently used by some ASSET computers, and will continue to do so, but we do not intend to integrate them in the uniform software environment of the ASSET, they will remain in the class of "special facilities".

2.4 Evolution of the ASSET concept

The choice of the VAX family running VMS, with the implementation of UNIX tools, and with the existing networking facility via DECNET, keeps our door opened for more and more appealing new solutions, like the Personal Work Stations (for which UNIX and ETHERNET seem to be de facto standards). PWS (and Personal Computers) which run IBM/VM or VAX/VMS operating systems are already available or are expected to be available within a few months.

PWS offer superb facilities for interactive operation and graphics at interesting prices and they will play an important role in the future for ALEPH software development. At present there is a wide variety of PWS with incompatible hardware, operating systems and network connections and the market is changing rapidly. This was the main reason why PWS were not considered in the initial phase. A clarification and stabilisation can be expected. Large computer manufacturers may step in in the near future. Delaying a decision on PWS by about 2-3 years could facilitate a coherent integration of PWS as ASSET.

3. ORGANISATIONAL ASPECTS OF PROGRAM DEVELOPMENT ON ASSETS

This chapter outlines a strategy for distributed program development on several ASSET and organisational procedures and support required.

1. Management of information

A set of clear procedures has to be defined to make distributed software development successful. This must include procedures for complete version control of code and documentation, information on all additional information related to the project, status of various tasks and overall planning. Access to information must be available to all people involved in the project. The following points give a very limited outline.

- different laboratories will be responsible for separate sections of programs (e.g. module or combination of modules)
- the general program file base is kept on a central ASSET, this is the Master File Base (MFB) maintained by the Program Librarian.
- material to be included in the MFB must have been tested by the authors on their ASSET with agreed procedures according to the test plan.
- the Program Librarian will run the same set of procedures. Code and documentation will be handed back to the authors in case of discrepancies with agreed standards. Similarly if material does not pass tests on other selected computers.
- accepted code and documentation will be included in the MFB, the set of tests repeated again and the MFB updated if successful. Detailed information on updates will be communicated to all ASSET.
- material from groups who do not have an ASSET has to be verified on an ASSET before it can be submitted to the Program Librarian. Then the standard procedure will be applied.
- correction of errors found in material of other authors should be communicated to the authors and the Program Librarian, updated material must be provided by the authors and the previous procedure will be followed.
- if more than one group works on a given program section this has to be communicated to the Program Librarian who will make sure that all material from all sources is included when the program section is updated. Verification of the complete material has to be done by all groups concerned.

- the exact form of code and documentation on the MFB depends on the final choice of the Source Code Management system, it will always be directly accessible by every ASSET, however.

- writing of code and documentation, formal verification with defined procedures, installation of new updates from the MFB for code documentation and data files is the responsibility of the group who produces the program segment.

- testing of new material, updates of the MFB, version control, history of updates, updates of test data base with results of tests and distribution of information is under the responsibility of the Program Librarian.

2. Management of environment

A certain amount of support is required to ensure that the proposed system is viable for many years despite the increase of information. Details have still to be defined but the following points will have to be addressed.

- support of a Program Librarian for maintenance of the Master File Base as outlined above.

- verification of consistency and integrity of information and analysis of performance of procedures and overall system.

- distribution and maintenance of tools.

- standardisation of ASSETs, i.e. naming conventions, user privileges, advice to individual ASSET system programmers for technical questions, etc.

- general network maintenance.

- general data base support.

4. IMPLICATIONS OF THIS CHOICE AND ACTIONS NEEDED

The ASSET offers opportunities for significant manpower savings in the software development phase, by increasing programmers' productivity, reducing unnecessary continual compatibility overheads (and related frustration), and hopefully leading to more reliable programs, and therefore better physics. It is clear that all this does not come for free, and financial, organizational and training investments are needed, from all participating institutes and from CERN.

a) Financial:

Laboratories of ALEPH participating in the software development must find a way to acquire an ASSET or to have convenient access to one. The investment in hardware and software will depend on the configuration available already and can not be given here in general terms. A separate note will be prepared which will provide such information.

Laboratories will have to subscribe to their national PTT packet switching telecommunication network, and eventually influence their national policy in the direction of standard international networking.

Adequate ASSET facilities will have to be made available at CERN not only for CERN-based people but also for physicists who come for short or extended visits and would like to continue work in the familiar environment.

Adequate manpower will have to be provided to support operating systems, libraries, maintenance of master file bases, networks, etc. as outlined above.

b) Organization

Close contacts between ALEPH laboratories and CERN (and possibly other LEP collaborations) will be needed to optimise procurement of hardware and software from Digital Equipment Corporation and other sources. This ensures that operating systems and software tools can be obtained within a common licensing scheme and that costs can be kept down. New versions of the operating system and packages can be installed everywhere within a short time interval in a coordinated way.

c) Training

Last, but not least, the good old days when a 2-weeks FORTRAN course, the knowledge of a few simple control cards and HBOOK calls were enough to transform a physicist into a programmer - analyst - software en-

gineer are over [8]. The complexity of our task requires extended knowledge, so-called "user-friendly" systems are not enough.

We will try to develop software and documentation in a way which should make it easy to adapt them to new requirements and make it easy to use them by people who have not participated in early development. This will require familiarity with modern techniques of software engineering. A series of tutorials - inside ALEPH or on a more general basis - will be needed to ensure that these techniques are used efficiently by people who participate in the software development.

5. GLOSSARY

- ALEPHNET: the network of ASSETs implemented with extra software relying on existing protocols
- ASSET: the software development machine with the set of tools for that type of machine.
- CERNET: current high-speed general purpose network on the CERN site
- Data Base (DB): all stored information relevant to an organisation. It is normally taken to mean that the data are also structured in some sense.
- DATEX-P: name of PTT packet switching network in Germany
- DB: see Data Base
- DECNET: complete set of network protocols and programs for DEC VAX and PDP11 computers
- EARN: European Academic and Research Network, sponsored by IBM
- ETHERNET: a local area network technique originally developed by XEROX for which many computer manufacturers have now announced support
- gateways: the contact point between different networks in which protocol conversion can take place
- GIFT protocol: General Internetwork File Transfer protocol (under development)
- INFNET: the Italian INFN network
- JANET: UK Joint Academic Network
- Master File Base (MFB): contains all version of code, documentation and any other information related to software development
- Personal Work Stations (PWS): a desk top computer dedicated to a single user, usually 32-bit with raster screen and network connections, distinguished from Personal Computer (PC) by its greater power
- SNA: the IBM Standard Network Architecture
- Software Design Methodology: a set of management procedures and techniques with computer support to assist in software development and maintenance
- Source Code Building: the process of combining and manipulating source files, to construct new source files. These will often form the input to a compiler.
- Source Code Management: process of storing versions of source code such that old versions may be retrieved.
- Target machines: are in this context the large main frame computers to be used for bulk data processing.
- UNIX: a popular, portable interactive operating system developed by Bell Labs. available on a wide range of computers. UNIX (or look-a-likes) are fast becoming a standard operating system for 16/32-bit microcomputers and PWS
- UNIX tools: many software packages developed for UNIX systems are available free or for a very small fee
- VAX: name given to a range of 32-bit computers manufactured by Digital Equipment Corporation

- VM/CMS: Virtual Machine/Conversational Monitor System, an IBM operating system running on 43xx and 30xx series machines and compatible machines of other manufacturers. Provides efficient support for large numbers of interactive users.
- VMS: the virtual memory operating system for DECs VAX computers for interactive and batch users
- X.25 protocol: the agreed standard for the first 3 layers of the ISO model for data communications

6. REFERENCES

[1] - ALEPH SW/82-3, November 18,1982
ASSET and ALEPHNET,some thoughts for the definition of ...

[2] - draft, O.Braun, May 10, 1983
A 'soft' solution for the ALEPH Software Development Environment

[3] - Version 1, J.Boucrot,S.M.Fisher,F.James,M.Rumpf, May 11, 1983
Triple ASSET

[4] - draft, S.M.Fisher, May 12, 1983
ASSET, an IBM solution

[5] - ALEPH SW/83-2, March 1,1983
Minutes of the second Plenary ALEPH software meeting

[6] - CCITT Yellow Book vol VIII.2
Data Communication Networks

[7] - ECFA/82/60 Networks For High Energy Physics
An interim report on wide area communications

[8] - Kenneth G. Wilson, Theoretical science and the future
of large scale computing, CERN Computer Seminar 25/2/1983,
see CERN Courrier, vol.23, no.5, June 1983

Aleph Note # 111

Date : 11 Nov 83
Distribution : ALPNOT,SOFTWR
Keywords : SW
Authors : Aleph SW Group
Contact : G. Kellner

SOFTWARE FOR ALEPH

PHASE 1 : DEFINITION OF REQUIREMENTS FOR SOFTWARE SYSTEM

CONTENTS

Chapter 1

USER REQUIREMENTS - GENERAL

1.1 INTRODUCTION

In the following, the requirements for all the software, on-line and off-line, will be discussed except the data acquisition software which will be treated elsewhere. The list of requirements given below is certainly not complete and will evolve with time.

There are two chapters of User Requirements following this general one. They concern on-line and off-line user requirements. Here they are separate chapters despite the fact that there are many connections between them. It is obvious that the same histogram, graphics, statistics and man-machine interface packages should be used. The dividing line between data acquisition and what is here called "on-line software" is not well defined, and it is highly desirable that this line should be even more ill defined.

1.2 REQUIREMENTS

1-1 Mandatory... Code shall work on all selected computers.

1-2 Mandatory... The applications software shall be well documented.

1-3 Mandatory... Code and documentation shall be transportable and easy to install.

1-4 Mandatory... Sufficient information shall be recorded defining the data format and program versions that data may be reprocessed under identical conditions.

1-5 Mandatory... Error messages should be comprehensive and comprehensible.

1-6 Mandatory... Comprehensive exception handling shall be provided.

1-7 Mandatory... Statistical information on processing and error occurrences shall be logged.

1-8 Mandatory... The same software system shall be available interactively or in batch mode.

1-9 Mandatory... The program organisation shall be such that users may readily tailor their own jobs.

1-10 Mandatory... Programs shall have access to correct geometry information and to details of defective parts of the detector system.

(1-11 Recommendation... Mass storage space shall be available to hold at least several hundred thousand reconstructed events and reconstructed Monte Carlo events for interactive physics analysis.)

1-12 Mandatory... Code and detector descriptions will be shared where possible between different parts of the software system.

1-13 Mandatory... An off-line event display package (containing the on-line event display), with 2D and 3D graphics on a variety of output devices including high quality hard copy printer is required.

1-14 Mandatory... The display package shall allow interactive display, reconstruction and updating of output.

1-15 Mandatory... The interactive display package shall be available for program development.

1-16 Mandatory... The interactive display package shall work for both real and Monte Carlo events.

Chapter 2

USER REQUIREMENTS - ON LINE

2.1 INTRODUCTION

The on-line software has to enable the survey of the detector to be performed and to check the quality of the data while running. This will be done by a monitoring task which will run automatically and will include event reconstruction with tasks which can be invoked by the user.

After certain periods of time, such as end of run, a summary program will collect information (run statistics, spectra, etc) and be able to output it.

In general the reconstruction algorithms will be less sophisticated (e.g. no full error analysis) and less precise then the final reconstruction, since not all constants are yet available. Yet there is no reason why some events should not be passed through what would be normally considered full off-line processing while in the "on-line" machine.

2.2 REQUIREMENTS

2-1 Mandatory... Rapid display of current detector status, machine status, luminosity and run statistics shall be available. This information shall be logged.

2-2 Mandatory... Monitoring output for each module in tables histograms and other displays shall be available. Some displays shall be always available and others shall be under user control.

2-3 Mandatory... Information on all defective parts of the detectors shall be stored and made available to programs or people requiring it. Detection of defective parts shall be automatic but with manual override.

2-4 Mandatory... On-line event reconstruction with different sampling modes for individual modules shall be available. A global event classification shall be made. Results may be output and reset without affecting the data acquisition system.

2-5 Mandatory... An event display shall be available which will permit visualisation of raw data for different thresholds and of reconstructed events. Views, modules and event selection can be chosen by the user.

2-6 Mandatory... Multiple output streams shall be provided which are distinct from those of the data acquisition system and which may be used for recording special events.

2-7 Mandatory... Remote access to the on-line system shall be available.

Chapter 3

USER REQUIREMENTS - OFF LINE

3.1 INTRODUCTION

This part of the software contains mainly calibration and reconstruction programs. The data processing may be split into several steps.

A first pass might collect large event samples sufficient to determine calibration constants. Event filtering might be necessary before full event reconstruction.

A later pass might provide a general event reconstruction and event classification for the bulk of the events in batch mode on a central computer. It may be this pass does not give the best reconstruction in every aspect (e.g. secondary vertices might not be reconstructed)

A selection of reconstructed events according to different criteria may result in split data files which can be further analysed (e.g. "QED"-,hadron-,heavy lepton-events,etc.).

3.2 REQUIREMENTS

3-1 Mandatory... Calibration programs are required which might use special calibration files on a data base or real events to produce calibration constants for use in the reconstruction.

3-2 Mandatory... Pattern recognition program for charged tracks with high precision are required to achieve a spatial resolution very close to the theoretical one and with an efficiency close to 100% within the fiducial volume and for momenta above some threshold of perhaps 200MeV. The minimum observable momentum is 50Mev and the maximum momentum with charge determination is 90GeV. The track-finding should work globally (using the inner chamber and TPC) but also work in a modular fashion.

3-3 Mandatory... Code is needed to determine the charge and momentum vector of charged particles and the corresponding error matrix.

3-4 Mandatory... Code is required to perform track finding in the muon chambers and to track through the calorimeters and link to the charged track in the TPC.

3-5 Mandatory... Code is required to perform cluster finding in the calorimeters.

3-6 Mandatory... Code is required to measure the direction and energy of neutral clusters in calorimeters.

3-7 Mandatory... Code is required to associate charged tracks with clusters found in the calorimeters.

3-8 Mandatory... Code is required to perform primary vertex identification and reconstruction. A kink in a track and conversions shall be recognised.

3-9 Mandatory... Electromagnetic shower identification in the e-gamma calorimeter and possible association with a track (e-gamma separation) shall be possible.

3-10 Mandatory... Code is required for hadronic shower identification and energy moments of the shower.

3-11 Mandatory... Code is required to identify particles from dE/dx in the TPC and to express the result in terms of probabilities; the wire assignment to a track having already been performed in the track reconstruction.

3-12 Mandatory... Event classification shall be summarized in an event code.

(3-13 Recommendation... Simple reconstruction shall be available in order to filter events. This may use the results of on-line reconstruction.)

(3-14 Recommendation... Event processing time per event for bulk processing should be kept low enough such that with the expected computing power, processing may keep up with data taking.)

3-15 Mandatory... A modular Monte Carlo program to simulate events and detector response (including digitisation) is required. The user must be able to select different physics models and to enable or disable different detector modules. The output must be as "real" data (except for the additional "true" quantities) and could be used as input to reconstruction and event display programs.

3-16 Mandatory... A fast tracking algorithm in the Monte Carlo is required using the "true" track information and fast shower simulation. Error correlations need not be taken into account.

3-17 Mandatory... Fast procedures for the shower simulation in the Monte Carlo are required (perhaps by accessing shower libraries on mass storage devices).

3-18 Mandatory... Specialised vertex code for reconstruction of secondary vertices of decays is required.

3-19 Mandatory... Kinematic fitting code for reconstructing events for some standard hypotheses (e.g. tau-decays, charm and bottom production and decay) is required.

Chapter 4

SOFTWARE MANAGEMENT

4.1 INTRODUCTION

The development of a software system can be separated into several distinct phases :

- definition of requirements for software project,
- specification,
- architectural design,
- detailed design,
- implementation of software (coding),
- operational testing and installation,
- production, maintenance and evolution.

The total period, from inception of the project to final disposal in some archive, is called the software life-cycle. The end of each phase represents a milestone in the development of the software. Each phase has to be terminated with thorough documentation of work completed, a critical review, a detailed plan for the next phase, and an updated overall plan for the remainder of the project. This overall management plan must contain realistic estimates of time and resources required across the system life cycle. However, accurate estimates cannot be expected during the initial phase of the project.

The purpose of this document is to describe the requirements for the design of the Analysis Software for the ALEPH detector. This represents the first phase of the software life-cycle.

Requirements are normally based upon the 'user' requirement, tempered by operational restrictions which gives rise to the system requirements. The problem with this case is the definition of the user. He is not distinct from the developer of the software. Some 'users' are concerned with the system itself, some with major application programs, some with private application programs, and some wish to remain ignorant of programming. The physics requirements are somewhat distinct in that all other requirements should support them.

The definition of the requirements for the analysis software for ALEPH started at the beginning of 1983. The software has to be ready at the start-up of LEP, and it has to be maintained and developed for some 5-10 years. Some 20-30 people may work concurrently during various phases of the software life-cycle.

The software system will probably never be considered as "finished" during the life time of the ALEPH detector due to continued evolution of physics aims, LEP accelerator, detector hardware, software implementation and computer hardware.

Several teams will work on the implementation of the software at different sites.

The long life-cycle of the software system implies that many people who have not participated in the design and implementation phases will have to modify existing code or develop new code to be integrated into the overall system.

4.2 REQUIREMENTS

4-1 Mandatory... The software system should be designed such that changes in general predictions of theoretical models, changes to operating conditions of the accelerator, modifications and upgrades to the detector, the computer hardware or to software may be accommodated with a minimum amount of redesign.

4-2 Mandatory... An overall management plan has to be layed down during the initial phase in order to define the organisational details for the analysis, design and implementation of the proposed software system.

4-3 Mandatory... This plan has to define needs and allocation of manpower and resources to various subprojects, it has to define time scales which allow measurement of achievements with respect to projections at the major milestones of the project.

4-4 Mandatory... This plan has to be updated during the software development to provide a current picture of what is happening so that effective measures can be taken in time before any serious problems are encountered.

4-5 Mandatory... Performance criteria must be defined for parts of the software system to include as appropriate, time to execute, response time, reliability and precision

4-6 Mandatory... A document must exist (this document) which summarises all the features that are expected from the software system for the ALEPH detector in order to achieve the physics goals that were the original motivation to build the detector.

4-7 Mandatory... This document should formally be reviewed in detail by the users and by the software designers. Its approval constitutes one of the major milestones of the software project.

4-8 Mandatory... Corresponding documents which will require the same review procedures and may contain information partly in the form of diagrams will be required at the end of specification, architectural design and detailed design.

4-9 Mandatory... A quality assurance plan must be devised which includes version control to ensure that the entire software system (code and documentation in all forms) is always consistent with the rules set out in this document.

Chapter 5

THE SOFTWARE ENVIRONMENT

5.1 INTRODUCTION

From the investigations carried out in the first part of this phase, it is clear that most of the institutes involved in ALEPH wish to participate in the on-line and off-line software development effort, contributing with a limited number of persons each. There will be projects involving people from different institutes (distributed projects) and projects migrating from an institute to another (mobility of projects). The majority of physicists will work at their home institute, coming to CERN for limited and unpredictable periods of time, but still willing to continue their work (mobility of people). Software will be developed in different places by different people, but must work within the complete software system. This will be facilitated by a uniformity of development tools. Most of the institutes want to have available at home the totality of the code written (universality of software and data) and the updates must be carried out in the shortest period of time (synchrony of software and data).

At the start up of LEP some of the computers that were used for program development may be used for data production. The major fraction of the bulk data processing (data reduction) will be done on large main frames.

People will have to understand the behaviour of the detector which will almost certainly necessitate frequent modifications to software, to data base information which later may become relatively static, to the data dictionary and to the operational procedures. Fast feedback is required.

5.2 REQUIREMENTS

5-1 Mandatory... Participation in all aspects of software production for the entire life cycle of the software shall be possible for those having access to one of the approved software development computers.

5-2 Mandatory... The preferred software development computer is the VAX family with VMS. IBM with VM/CMS will also be supported but not necessarily at the same level.

5-3 Mandatory... A set of tools will be required with certain functionalities on each type of software development machine ... but not necessarily the identical tool on machines of different type. (The Aleph Standard Software Environment and Tools - ASSET - shall be defined as the software development machine with the set of tools for that type of machine).

(5-4 Recommendation... For each type of machine the ASSET must be fully standardized.)

5-5 Mandatory... The ASSET must provide access to programs, documentation, data base and mail from the decentralised user community, and provide procedures for update of source code, with records of all such changes and communicate these to everybody concerned with minimal delays.

5-6 Mandatory... Source code management tools with common functionality will be required.

5-7 Mandatory... Source code building tools with common functionality will be required.

5-8 Mandatory... The ASSETs should include the top level computers used in the on-line development and monitoring system, to facilitate the sharing of common tools between the on-line and the off-line software environments.

5-9 Mandatory... ALEPHNET will be provided to link the ASSETs by a network.

5-10 Mandatory... ALEPHNET shall allow login to remote machines, file transfer, electronic mail and job submission.

(5-11 Recommendation... ALEPHNET shall allow transaction processing.)

5-12 Mandatory... ALEPHNET shall accept as transport media the public packet switching network, private or leased lines with standard interfaces and the local area network selected for use in the data acquisition system.

(5-13 Recommendation... A special ALEPHNET protocol will be required to make the network look the same to any ASSET user, no matter which hardware implementation is underlying.)

(5-14 Recommendation... ALEPHNET must cope with the cases of links not easily affordable via computer (China?), and must provide the user with suitable ways of diverting the communications to the best suited channel (link, normal mail, pigeons...).)

5-15 Mandatory... The application software shall have no strong dependence upon any specific hardware or operating system.

5-16 Mandatory... A first selection of target machines should be made at a very early stage to ensure that software is compatible for the selected computers and that special constraints imposed by hardware and operating systems are taken into account.

(5-17 Recommendation... ALEPHNET shall also be available on all target machines to facilitate movement of information of all types, including code and sections of the data base.)

Chapter 6

SOFTWARE DESIGN METHODOLOGIES

6.1 INTRODUCTION

Methodologies have been developed, which allow the expression of requirements and design specifications formally, to display relations with graphical presentations, and to trace from requirements to implementation and back by keeping all information in a data base. In this way all updates of the software system during the entire life-cycle will be recorded and can be verified at any stage.

6.2 REQUIREMENTS

6-1 Mandatory... Some formal means should be used to specify requirements, design and implementation details precisely. Natural language is generally not very suitable to represent all aspects of a complex software system.

6-2 Mandatory... Proven software design methodologies should be selected and used from the very start of the project.

6-3 Mandatory... The methodologies shall cover all phases of the software life-cycle.

(6-4 Recommendation... A single methodology shall be chosen)

6-5 Mandatory... The methodologies shall provide verification at all stages.

6-6 Mandatory... The methodologies shall provide documentation understandable to non-experts so that progress of the software project can be made visible.

6-7 Mandatory... The methodologies shall be suitable for use in a distributed development environment.

(6-8 Recommendation... The methodologies shall provide automated tools and graphical output.)

6-9 Mandatory... The methodologies must be reasonably easy to learn for people who are using it and others who merely wish to monitor progress.

6-10 Mandatory... The methodologies should be open-ended to cater for new developments.

(6-11 Recommendation... The methodologies shall be proven with commercial support and generally available at a reasonable cost.)

(6-12 Recommendation... The methodologies shall be a European product or at least have a good European support.)

Chapter 7

DATA REPRESENTATION

7.1 INTRODUCTION

A Data Base may be defined as "all stored information relevant to an organisation", and it is clearly desirable to have a uniform way of viewing all the information relevant to ALEPH. Such information includes: all the event information stored on tapes, operational conditions for experiment, run statistics, log book type information, geometrical dimensions of detector components, material constants, patch panels, correspondence between preamplifiers and channels, cable characteristics, cable swaps and other read-out deficiencies, geometrical alignment of detector components, calibration constants, detector performance, i.e. efficiency maps, insensitive regions, production statistics and bookkeeping information.

It is not known, at present, how much information will have to be stored on the data base. It is realistic to assume that a large volume of information will accumulate with time. A variety of storage media will be required, but though the storage medium may vary the way of viewing the data must be the same.

In the past, to organise data in the computer memory, several memory management schemes have been designed. They all suffer from the same drawbacks, namely that: code is hard to write and almost impossible to read, dumps from crashing programs are hard to interpret and that at all stages a manual has to be consulted, and there is no guarantee that the manual is accurate or up to date.

It is expected that during the life-cycle of the experiment, the data will change both in content and in form as a consequence of evolution in physics, accelerator and detector hardware. The direct binding of the data entities to the code, which has been the practice so far, does not offer the desired flexibility to follow this evolution. A change in the size or content of a data bank, for instance, often requires extensive recoding. The situation is aggravated when imbedded memory management systems are used.

Some of the difficulties in the current way of working result from failing to adopt a data model, and to set up a means of representing it with a data dictionary.

The data model approach relies on a formal specification of the data structure where each entity is explicitly defined. This

definition resides in the computer system and offers the "conceptual" view of the environment. The binding of the data to the code occurs at compilation time through high level constructs.

A mechanism is required for the formal specification of the data structure, independent of internal data representation. This is achieved through a Data Description Language (DDL). This description may be converted to the format of the data dictionary, providing access to data objects by name from programs and interactively. Documentation of the data structure requires only a means of representing the data dictionary on a piece of paper.

An interesting possibility seems to be an extended relational model (eg. Entity-Relationship or similar). The DDL defines every object in the conceptual scheme in terms of name, type and range of values it can take. It can also establish relationships between objects.

It is envisaged that a memory management system will be used to represent the data in memory. This will not be visible in the application program code. Calls will be produced by a translation mechanism which refers to the Data Dictionary.

It is clearly important to cooperate with the on-line system developers here, as elsewhere to ensure that the data model may, by being universally accepted, give the greatest benefit to all.

7.2 REQUIREMENTS

7-1 Mandatory... A data model will be adopted with distinct conceptual, external and internal schemata.

7-2 Mandatory... The data model will be applied universally within the on-line and off-line systems, and to the description of the hardware.

7-3 Mandatory... A data definition language (DDL) will be defined to describe the data at the conceptual level.

7-4 Mandatory... An interface between this DDL and a data dictionary (DD) will be defined.

7-5 Mandatory... An interface to the adopted programming language will be required to provide independence of the programs from the strategy chosen for storage, and to provide the necessary functionality to access the data.

7-6 Mandatory... This language will be consistent with the coding standards requirements.

7-7 Mandatory... A tool must be available to check code written in the language for consistency with the DD.

7-8 Mandatory... It must be possible to request execution time checking for consistency with the DD.

7-9 Mandatory... Access to all information must be provided to authorised users.

(7-10 Recommendation... Access control should be provided at the level of read, write and update for information with different levels of granularity (that is by field, record or some other subset of the total information).)

7-11 Mandatory... Representations of the data must exist for storage in memory.

7-12 Mandatory... Representations of the data must exist on direct access devices and the language adopted must provide an interface between this and data in memory.

(7-13 Recommendation... The representation of the data on a direct access device will make use of a relational data base management system, which supports ad-hoc interactive queries.)

(7-14 Recommendation... It must be possible to access information on a remote computers direct access storage.)

7-15 Mandatory... Representations of the data must exist on sequential access devices and the language adopted must provide an interface between this and data in memory

7-16 Mandatory... A machine independent representation of the data must exist and the language adopted must provide an interface between this and data in memory

7-17 Mandatory... Representations of the data must exist on graphic output devices (though for some pieces of information it may only be a textual representation) and the language adopted must provide an interface between this and data in memory.

7-18 Mandatory... Representations of the data must exist as textual output (this will ensure comprehensible dumps) and the language adopted must provide an interface between this and data in memory.

(7-19 Recommendation... As volume of data per event is potentially large, it must be possible to partition the information in order to access only the relevant part.)

7-20 Mandatory... All computers will need read/write access to a defined part of the data base.

(7-21 Recommendation... All information must have a defined home, though many copies of it may exist for operational reasons.)

(7-22 Recommendation... If it is found that certain pieces of information must be duplicated, an automatic process should be available for quickly informing other computers (which have expressed an interest in that type of data) when the information has changed in a way which they regard as significant.)

Chapter 8

GRAPHIC REPRESENTATION OF INFORMATION

8.1 INTRODUCTION

The complexity and the high information content of events originating in the ALEPH detector call for a graphical representation rather than pages of paper filled with numbers. The basic function of graphics will be to provide interactive display facilities available on different graphics terminals to enable visualisation of tracks and of all other types of information described by the data model. It should allow us to visualise all aspects of the data.

The rapid evolution of the graphics hardware make it difficult to decide now which system to adopt. It is probable that within a few years from now systems with the performance of the MERLIN VAX-Megatek combination will be available at much reduced cost. Personal work stations equipped with high resolution screens provide some of the required graphics functionality at relatively low cost. Colour terminals should be considered seriously since they offer extensive possibilities of association and representation of additional information. The evolution of the market has to be followed closely.

Colour should be fully exploited.

8.2 REQUIREMENTS

8-1 Mandatory... A transportable graphics package supporting 3D raster and vector devices shall be available.

8-2 Mandatory... A standard device independent graphics language, shall be provided by the package.

(8-3 Recommendation... The package should support metafiles.)

8-4 Mandatory... Graphics stations with different functionality shall be supported.

8-5 Mandatory... Graphics workstations shall be available to support local 3D image transformations.

8-6 Mandatory... The package should support a wide range of graphics attributes including line style, intensity, and colour along with a variety of fonts.

(8-7 Recommendation... The application software should make use of appropriate graphics attributes.)

(8-8 Recommendation... The output devices should support a wide range of graphics attributes including intensity and colour.)

(8-9 Recommendation... A package shall be available to assist in scene modelling for presentation purposes and which supports hidden line removal, depth cue, perspective, colour and annotation.)

8-10 Mandatory... A flexible 2d and 3d histogram display package shall be provided.

Chapter 9

MAN MACHINE INTERFACE

9.1 INTRODUCTION

In the past few years the interaction between human beings and computers has experienced a rapid evolution from batch processing using punched cards via running interactively prepared batch jobs to truly interactive systems.

Most of the computing work presently done in high-energy physics is still of the second type. But the trend is clear and it is easy to predict that, by the start-up of LEP, truly interactive systems will be used. Interaction will be used in the fields of event presentation and physics analysis, program development and debugging, setting up of production jobs, online control and monitor systems, ad-hoc queries to the direct access part of the data base, and mail.

The careful design of the man machine interface is vital to produce a system which is liked by its users.

Terminals which are sufficiently similar that they may be used with identical software may be so grouped into classes.

9.2 REQUIREMENTS

9-1 Mandatory... There shall be a unique way of conceptualising the interaction between the user and the complete software system both on and off line. This means that the user may think about what he wants to do in abstract terms, and only then consider the hardware through which the interaction will proceed.

9-2 Mandatory... There shall be different external interaction schemes matched to the capability of the hardware used.

(9-3 Recommendation... The interaction schemes shall be common to on-line and off-line.)

9-4 Mandatory... A limited number of classes of terminal types will be supported and the interaction scheme should exploit the facilities of each class of terminal

9-5 Mandatory... The software for each terminal class must support a common interactive scheme based on the facilities available on the full range of approved terminals, so that a user faced with an exotic terminal may use it as a teletype if he wishes.

9-6 Mandatory... Where keywords are used they shall be unique for each function.

(9-7 Recommendation... Panel input (ie entering information in a 'form' on a screen type of terminal) and graphic input shall be exploited.)

9-8 Mandatory... Hierarchically structured HELP information should be available at all stages of the dialogue.

9-9 Mandatory... Sensible defaults should be provided, including a system default and whatever the user had used last.

9-10 Mandatory... The experienced user must be able to skip past defaults, without displaying them, to avoid frustration.

(9-11 Recommendation... In order to recover from user errors it shall be possible to undo the result of the last user action. Ideally one would be able to step back through the whole session.)

9-12 Mandatory... Errors in the interaction should be handled by accessing a part of the data base containing an error code, short error message, long error message and a link to the HELP system. The various parts would be used according to the class of terminal and the level of experience which the user has declared to the system.

9-13 Mandatory... Vital parts of the system shall be protected from accidental or malicious damage.

Chapter 10

DOCUMENTATION STANDARDS

10.1 INTRODUCTION

Documentation is essential for every user of the software system, and it comes in two extreme forms. One is the basic source of information to understand the design specifications, to follow through its implementations and to provide the user with enough details so that he can implement changes. The second basic type of document is for the casual user who merely wants to use the system and has no interest in modifying it.

There is of course a user somewhere in between who wishes to write his own small piece of code to run within the system; this documentation is perhaps the hardest to prepare.

The top level of documentation will be maintained with the code and will logically include all the design documents. If a program design language (PDL) were adopted then detailed descriptions of all code will exist. It would be especially convenient if the PDL were maintained as comments in the code.

The lowest level of documentation would have to be written the hard way, but there would not be much of it, and it would not be sensitive to small changes in code.

The intermediate documentation corresponds roughly to the conventional "User Manual", but much of such manuals concern themselves with descriptions of data banks. All this information is already available in the Data Dictionary.

10.2 REQUIREMENTS

10-1 Mandatory... Documentation shall be provided at 3 levels: for the 'casual' user who merely wants to use the software to analyse and examine a few events, for the 'average' user who wishes to incorporate a piece of his own code but does not want to concern himself with too much of the design documentation, and for the 'expert' user who may wish to modify the specification and design of the system.

10-2 Mandatory... Documentation must be submitted concurrently with the code and updated whenever changes are made.

10-3 Mandatory... Documentation describing the data shall be derivable from the data dictionary.

10-4 Mandatory... Such documentation as required by the program management and methodology adopted will be maintained.

10-5 Mandatory... Documentation should be organised in such a way that any section can be accessed interactively or several sections compiled into a printable form.

10-6 Mandatory... Selection of sections shall be by level (as defined above) and subject.

10-7 Mandatory... Indexing and cross referencing shall be provided on-line to facilitate location of information.

10-8 Mandatory... The same indexing and cross referencing system shall be visible in the printed documentation.

10-9 Mandatory... The indexing and cross referencing system must be adequate to teach even the novice without too much pain.

(10-10 Recommendation... Text should be interspersed with actual code inside modules and subroutines. It should be possible to extract this information to produce the sections of documentation.)

10-11 Mandatory... All documentation shall exist in English.

(10-12 Recommendation... Tools to verify correctness of documentation should be used.)

Chapter 11

PROGRAM ARCHITECTURE AND ORGANISATION

11.1 INTRODUCTION

Software is required to perform the tasks described in the user requirements chapters. Special situations may call for procedures not forseen in the standard program suite. It is desirable to have the flexibility to address these problems with the minimum of coding effort.

Noting that the detector itself is modular, and that we must support a distributed software development system then in order that the software produced shall be comprehensible and maintainable it must be written in well defined modules. Changes to the algorithms used should of course not change the external appearance of a module.

It is convenient to define two classes of module. A user module must follow certain rules to ensure that the data structure is not damaged. The principal rule is that a user module may not modify or delete its input data structure. It may of course add to the data structure or it may make temporary additions to the data structure which it deletes before returning. It is of course necessary to have modules which may modify the data structure, in particular for rerunning and interactive work, so some system modules are required.

11.2 REQUIREMENTS

11-1 Mandatory... The software shall be divided into well defined modules, some of which will be system modules and some user modules. A module is an independent piece of code which, with the aid of algorithms, transforms a well defined input to a well defined output.

11-2 Mandatory... The modularity shall be such that a good interactive system may be maintained.

11-3 Mandatory... A user module will take only the data structure (ie that part of the totality of information which is stored in memory or random access disk-like storage as defined by the implementation of the data model) as input.

11-4 Mandatory... A user module will not modify the input data structure.

11-5 Mandatory... A user module will have no input or output other than that defined by the implementation of the data model (except for special debugging information and perhaps a very restricted set of system flags)

11-6 Mandatory... User modules shall have low coupling and high cohesion. This means that they may not have side effects nor modify hidden variables, and that each module must have a simply defined function.

11-7 Mandatory... System modules must exist to perform i/o to sequential data sets within the confines of the data model.

(11-8 Recommendation... Every subdetector shall have at least one module for each of analysis, simulation (inverse of analysis) and monitoring.)

11-9 Mandatory... A system module must exist which may interact with a user to modify the data and its structure.

11-10 Mandatory... A system module must exist which acts as a supervisor to combine other modules in any valid program structure and which will not allow an invalid structure to be created. An invalid structure would be one in which a module did not have the input data it required. This could be due to mispositioning of another module, which if executed early enough, would have produced the required information.

11-11 Mandatory... This supervisor must be able to interact with the user both before and during execution of the user modules.

11-12 Mandatory... Modules shall invoke algorithms which are well separated from the management of the module such that the replacement of one algorithm by another shall not require any changes outside the module except perhaps different control information.

11-13 Mandatory... An agreed set of utility routines will be defined.

11-14 Mandatory... This set of utility routines will only be extended after a defined procedure has been followed.

11-15 Mandatory... Batch programs should record useful information in the direct access part of the data base to simplify checking the results of a batch job, and to identify the program version and input data such that the job could be rerun.

Chapter 12

CODING STANDARDS

12.1 INTRODUCTION

Certain conventions on programming style are essential in order to make code readable, especially for people who have not participated in the design and implementation phases. The fact that different programmers have contributed to different sections of the program should not be immediately evident.

Some constraints may be introduced by the software methodology chosen and by its support tools.

12.2 REQUIREMENTS

(12-1 Recommendation... A single high level program language will be chosen.)

12-2 Mandatory... Conversion to machine code may be by means of a portable pre-compiler and a standard compiler or just a standard compiler.

12-3 Mandatory... Input to the compiler shall be a defined subset of an ANSI standard which will run on the chosen development computer(s) and analysis computer(s).

12-4 Mandatory... There will be a convention for statement labelling if allowed by the language

12-5 Mandatory... There will be conventions for layout of code if allowed by the language.

12-6 Mandatory... There will be conventions for all name constructs in the language.

12-7 Mandatory... Each section of code will have a comment section at the top with a defined layout to define its function in terms of its interaction with the data structure and the algorithm used, and a revision history. It is not necessary to repeat information stored by the Source Code Manager adopted.

12-8 Mandatory... There shall be sufficient in-line comments to understand the flow of the program.

(12-9 Recommendation... Comments and code shall be relatable to the documentation.)

(12-10 Recommendation... Structured code shall be used where appropriate.)

Chapter 13

TEST FACILITIES

13.1 INTRODUCTION

A standard set of tests must be defined as an integral part of the detailed system design. This set of tests must always be passed successfully before updated code will be integrated into the program system.

A set of tools has to be selected which can be used to verify compliance with the test plan on all computers where the software will be installed.

The data base will contain input data sets, test programs and results for each section of the code to be tested and is essential to serve as a master for comparisons at any time.

This part of the data base must be available to all ASSETs and production computers

13.2 REQUIREMENTS

13-1 Mandatory... A standard set of tests must be defined as an integral part of the detailed system design. This set of tests must always be passed successfully before updated code will be integrated into the program system.

13-2 Mandatory... All new code shall be tested to see that it conforms to the coding standards

13-3 Mandatory... The data base will include for each section of code to be tested input test data, test programs and results of the test.

13-4 Mandatory... A standard set of procedures must be provided to make the comparisons with the data base information fully automatic and provide information if and where discrepancies are found.

13-5 Mandatory... Tools will be used to check logical program paths through the code

(13-6 Recommendation... Sufficient information shall be provided by different levels of debug option to follow the actions of the program.)

13-7 Mandatory... The new code will be integrated with the rest of the module and tested on the development machine and on some specified target machines with a known data sample and test program held on the data base

13-8 Mandatory... Before releasing the code it shall be tested on one of each type of target machine with a known data sample and test program held on the data base

13-9 Mandatory... It shall be possible to add specific "troublesome" events (those which gave problems to some program version) to the test data.

GLOSSARY

ALEPHNET - the network of ASSETs. This may be some special protocols on top of some existing ones.

ASSET - the software development machine with the set of tools for that type of machine.

Data Base (DB) - all stored information relevant to an organisation. It is normally taken to mean that the data are also structured in some sense.

Data Definition Language (DDL) - defines the conceptual schema in a data model.

Data Dictionary (DD) - is the stored form of the conceptual schema of a data model.

Data Model - describes a system by means of distinct conceptual, external and internal schemata. See for example ANSI/X3/SPARC/DBMS framework report (Information Systems Vol 3, 1978)

DB - see Data Base

DD - see Data Dictionary

DDL - see Data Definition Language

HELP system - interactive documentation scheme.

Panel input - IBM term for entering information on a 'form' on a screen terminal.

PDL - see Program Design/Description Language

Program Description/Design Language (PDL) - see Program Design/Description Language

Program Design/Description Language (PDL) - is an informal language with a few keywords which may describe in any level of detail the action of a program. At its most detailed level, about 1 line of PDL corresponds to 6 lines of Fortran.

Source Code Building - the process of combining and manipulating source files, to construct new source files. These will often form the input to a compiler.

Source Code Management - process of storing versions of source code such that old versions may be retrieved.

System/User modules - are distinguished by the restrictions which apply to user modules which do not apply to system modules. A user module must follow certain rules to ensure that the data structure is not damaged. The principal rule is that a user module may not modify its input data structure. It is of course necessary to have modules which may modify the data structure, in particular for rerunning and interactive work, so some system modules are required.

Target machines - are in this context the large main frame computers to be used for bulk data processing.

Transaction processing - sequence of operations performed as a group, such that failure of one of the operations will leave the system in the initial state.

User/System modules - see System/User modules.