

**K^0 Production and
Data Acquisition
with ALEPH**

Andy Belk

A thesis submitted for the degree of

Doctor of Philosophy

of the University of London

and for the Diploma of Membership of Imperial College

October 1991

Imperial College, London

For my friends.

K^0 Production and Data Acquisition with ALEPH

Andy Belk
Imperial College, London

Abstract

The ALEPH Detector and Data Acquisition system are described. The development of the software for the system is discussed in detail, with emphasis on the use of software engineering techniques. A major part of the software involved the development and exploitation of databases to describe the geometry, topology and status of the system. Those describing the readout structure and the partitioning of the system into independent sub-systems are presented. Software packages written to aid the provision for simultaneous parallel data paths and multiple users of the apparatus are detailed.

A study of K_S^0 production in hadronic events from e^+e^- annihilation has been made for a centre of mass range $91.0 - 91.5$ GeV. A summary of the selection procedure and the processing of hadronic events measured in the ALEPH detector follows. The algorithm for V^0 and subsequently K^0 finding is explained together with a summary of its performance. The selection criteria for K_S^0 are described in detail. Results for mean multiplicity, the inclusive cross-section and K^0 to charged track ratio are presented. Some interpretation of these results concludes the work.

The “Without Whom” page

Eternal thanks:- Lucas, Tim, Jerry and Mark — who know why.

Aux CERN Pompiers et l'Hopital de la Tour pour une raison similaire.

Joh, for a lifelong friendship.

Julie, Kerstin, Sarah and Mille, for inspiration.

Gabi for the chocolate, but mostly for being wonderful.

Les mecs et filles du Chalet, pour toutes les fêtes.

And last, but not by any means least, to Rachel.

Thanks are gratefully offered to all the many people I have met over the n years of my Ph.D., from both within and outside of physics, who have provided sound advice and ideas. As usual, my memory is not good enough to remember all of their names, but I trust if they ever read this epic, they will know who they are. Names most easily recalled:- John Harvey, John Shade, Wolfgang von Rüden and Traudl Hansl-Kozanecki.

Gratitude to my supervisor, Peter Dornan, for his guidance and encouragement. To the Imperial College High Energy Physics group as a whole, with whom it has been a pleasure to work and (on occasion), to play. And to the people at CERN who make it tick ...

I would like to thank Lucas, Marco, Mike and Peter for reading through the thesis and spotting all (well I hope all !) of my spelling and grammatical mitsakes.

This work comes courtesy of the Great British Taxpayer, via the long-winded route of the Science and Engineering Research Council, the CERN ALEPH group, for a year's financial support in Geneva, and myself, for sufficient financial acumen to survive on a grant for 4 years.

And ... a big thankyou to my parents for a good head start in life.

This thesis is dedicated to my friends, for without them, Mankind has no use for Physics.

Contents

1	The Experimental Environment	1
1.1	The Detector Organisation	1
1.2	Description of the ALEPH detector	2
1.2.1	The Minivertex Detector (VDET)	2
1.2.2	The Inner Tracking Chamber (ITC)	3
1.2.3	The Time Projection Chamber (TPC)	3
1.2.4	Solenoid	4
1.2.5	Calorimetry	4
1.2.6	The Electromagnetic Calorimeter (ECAL)	4
1.2.7	The Hadron Calorimeter (HCAL)	4
1.2.8	The Muon Chambers (MUON)	5
1.2.9	The Luminosity Calorimeter (LCAL)	5
1.2.10	The Small Angle Tracker (SATR)	5
1.3	The Trigger	5
1.4	Introduction to the ALEPH Data Acquisition System	6
1.5	The DAQ System Hardware Architecture	7
1.5.1	The Fastbus Sub-system	7
1.5.2	The Readout Modules	9
1.5.3	The Trigger Sub-system	10
1.5.4	The Online Computers	10

1.6	ALEPH Software	11
1.6.1	Software Engineering and Data Modelling Techniques	12
2	The Online System Software	15
2.1	Overview of the DAQ System	15
2.1.1	Requirements	15
2.1.2	The Partition Concept	16
2.2	Requirement Solutions	19
2.2.1	Readout Protocol	19
2.2.2	Trigger Protocol	21
2.3	Software Design	24
2.3.1	Data and Control Flow	24
2.3.2	The Finite State Machine Model	25
2.4	DAQ System Architecture	28
2.5	DAQ Software Architecture	29
2.5.1	Management, Control and Sequencing	30
2.5.2	The FSM Library	31
2.5.3	Other Utilities and Packages	32
3	The DAQ Databases	37
3.1	What the Databases Describe	37
3.2	The Detector Description	38
3.3	The Readout Database	40
3.3.1	Access to the Information	41
3.4	The Trigger Layout Description	42
3.4.1	The FIO Box	42
3.4.2	Route-finding and FIO Programming	44

3.5	Implementation Architecture	46
3.5.1	The Database to the Programmer	47
3.5.2	Database Tools and Code Generation	47
3.5.3	Access to Data: Data-Hiding	48
3.5.4	Implementation of Data Sharing and Permanent Storage	49
3.5.5	The Cluster-wide Database	50
3.5.6	Implementation of Access Library	51
3.5.7	Programming Methods Used	52
3.6	User Interface	52
4	Management Software	55
4.1	Requirements	55
4.2	Partitions	56
4.3	The ALEPH Resource Manager (ARM)	56
4.3.1	Resource Names and a Hierarchy	57
4.3.2	Partitioned Resources	59
4.3.3	Design in Brief	59
4.3.4	Implementation	61
4.4	Partition Management	63
4.5	The ALEPH Partition Manager	64
4.5.1	Requirements	65
4.5.2	Design	65
4.5.3	Other features	67
4.5.4	Implementation	67
4.5.5	Opening a Partition	69
4.5.6	The Partition Manager User Interface	69

5	DAQ: Conclusions and Summary	71
5.1	Mode of Operation	71
5.2	Taking ALEPH Data	71
5.2.1	Example of a Run Sequence	71
5.3	Conclusions	72
5.4	How to do it better next time	74
5.4.1	Language Issues	74
6	Analysis Overview	77
6.1	Overview of the Analysis Procedure	77
6.2	Online Data Reduction	78
6.2.1	Zero Suppression	78
6.2.2	Multi-Stage Trigger	78
6.3	The K^0 Particle	79
6.3.1	Properties of the K^0	80
6.3.2	Production Mechanism	80
6.3.3	Hadronisation Models	81
6.3.4	Inclusive Hadron Production	81
6.4	Coordinate System and Track Parameters	82
7	Event and V^0 Reconstruction	85
7.1	The ALEPH Reconstruction Program	85
7.1.1	Track Finding	86
7.1.2	Track Fitting	88
7.1.3	Primary Vertex finding and fitting	88
7.2	Event Preselection	89
7.2.1	Hadronic Event Selection	89

7.3	V^0 Reconstruction	90
7.3.1	V^0 finding	90
7.3.2	The YV0 Package	92
7.3.3	Selection at the V^0 level	92
7.3.4	V^0 Finding Efficiency	95
7.3.5	Cross-Checks on V^0 Finding	96
8	K^0 finding	99
8.1	Selecting K_S^0 s and Enhancing the Signal	99
8.1.1	Physics Background Processes	99
8.1.2	Other Background	101
8.2	Motivation for cuts	101
8.3	K^0 Reconstruction	103
8.3.1	Cuts	103
8.3.2	K_S^0 / Λ Ambiguities	110
8.3.3	Track-sharing Ambiguities	112
8.3.4	Fitting	112
8.4	Checks for Biases	113
8.4.1	Topological Vertex Finding	114
8.4.2	Sideband Subtraction	115
8.4.3	Modification of Track Momentum Resolution	115
8.5	Efficiency Corrections	115
8.5.1	Correcting the Inclusive Momentum Distribution	115
8.5.2	Correction for K_S^0 s produced in Material Interaction	117
8.5.3	Corrections in Angular Distribution	117
8.5.4	Hadronic Event Selection Corrections	117
8.5.5	Correction for energy loss in the Detector Material	118

8.6	Systematic Errors	120
8.6.1	Systematic Errors due to Data Acquisition	120
8.6.2	Systematic Errors due to Simulation Deficiencies	120
8.6.3	Systematic Errors due to the Fitting Method	122
8.6.4	Systematic Error Summary	122
9	Results, Discussion and Conclusions	123
9.1	Overall Efficiency and Signal Purity	123
9.2	Lifetime	123
9.3	Extraction of Overall K_S^0 multiplicity from Lifetime distribution	124
9.4	Inclusive Scaled Energy Spectrum	125
9.5	K^0 Multiplicity and K/h Ratio	130
9.6	Inclusive Cross-Section	133
9.7	Further Work	133
9.8	Conclusions	134
A	Systematic Error Checks	135
A.1	Momentum Resolution — The Effects on the x distribution	135
A.2	Checking the Effects of Cut Variations	136
B	Tables of Results	138
B.1	K^0 Finding Efficiency	138
B.2	Measured x Bin Contents	138
B.3	Correction for Initial State Radiation	141
B.4	K^0 x Spectrum after all Corrections	141
B.5	K_S^0 to Charged track ratio in x	141
C	Glossary	144

List of Figures

1.1	The ALEPH Detector	3
1.2	The ALEPH Fastbus System	8
1.3	Part of the Trigger Signal Distribution Tree	11
2.1	Data Flow within an Event Builder	20
2.2	DAQ Context Diagram	24
2.3	Overall DAQ Data Flow	25
2.4	DAQ Control Flow	26
2.5	The ALEPH DAQ Run Protocol State Transition Diagram	27
2.6	Transition Dependence	27
2.7	The DAQ Sub-Systems	28
2.8	The DAQ Software Layers	29
2.9	The FSM Package Logic	31
2.10	The Run Task Architecture (1990)	32
3.1	System Database Context	38
3.2	Detector Description ER Diagram	39
3.3	The ALEPH Readout ERD and its relationship with the Detector Description	40
3.4	The ALEPH Trigger System Description as an ER Diagram	43
3.5	The FIO Module (schematic)	44
3.6	General Database Organisation	48

3.7	Cluster-wide database updating	50
3.8	Operations on a Linked List	53
3.9	The Readout Database User Interface	54
4.1	How Resources are Grouped Together	58
4.2	Defining a Partition using Resource Names	62
4.3	The Concept of Transfer Vectors	63
4.4	Partition Description ER Diagram	66
4.5	The User Interface to the Partition Database	70
5.1	An Example Session: Starting a Run	72
6.1	The ALEPH Coordinate System	82
6.2	ALEPH Track Parameters	83
7.1	Coordinate-linking Strategies	87
7.2	A V^0 Decay in the ITC	91
7.3	V^0 decay topology	93
7.4	The definition of Point of Closest Approach	94
7.5	$m_{\pi^+\pi^-}$ for simulated data with perfect tracks	97
8.1	The K^0 background in $m_{\pi^+\pi^-}$ spectrum	100
8.2	The Track Fit χ^2 distributions for Monte Carlo and real data	102
8.3	The effect of the V^0 d^0 cut on Monte Carlo and real data	104
8.4	The effect of the cut on K^0 decay length	106
8.5	The $\cos \theta^*$ distributions for Monte Carlo data	107
8.6	The effect of the $ \cos \theta^* < 0.75$ cut on the $m_{\pi^+\pi^-}$ distribution	108
8.7	Measured $\cos \theta^*$ distribution for K_S^0 s	109
8.8	Corrected $\cos \theta^*$ distribution for K_S^0 s passing all cuts	109

8.9	The effect of the opposite sign d^0 of tracks cut	110
8.10	The effect of a $ \cos \theta^* < 0.75$ cut on the $m_{\pi^+\pi^-}$ vs. $m_{p\pi}$ distribution	111
8.11	$m_{\pi^+\pi^-}$ vs. $m_{\pi^+\pi^-}$ for K_S^0 candidates sharing a track	112
8.12	$m_{\pi^+\pi^-}$ invariant mass spectrum and result of fit	113
8.13	$m_{\pi^+\pi^1}$ opposite and same sign combinations	114
8.14	The $K_S^0 \rightarrow \pi^+\pi^-$ finding efficiency in x calculated from simulated data .	116
8.15	The correction curve for the effects of Initial State Radiation	116
8.16	Ratio of polar angle distributions for real and simulated data	118
8.17	The shift in the K_S^0 mass peak at $x = 0.012$	119
8.18	Difference between the fitted and the standard K^0 mass	119
9.1	The $K_S^0 \rightarrow \pi^+\pi^-$ finding efficiency as a function of lifetime	124
9.2	The fitted K^0 lifetime for simulated and real data	125
9.3	The K^0 x distribution	126
9.4	The K^0 x distribution — low x	127
9.5	The K^0 x distribution including $1/\beta$ factor at different CM energies	128
9.6	The K^0 x distribution including $1/\beta$ factor — low x values	128
9.7	The K^0 x distribution including $1/\beta$ factor — mid x values	129
9.8	The K^0 x distribution including $1/\beta$ factor — high x values	129
9.9	The mean K^0 multiplicity as a function of centre-of-mass energy	131
9.10	The K_S^0 /charged track ratio as a function of x	132
9.11	The mean K_S^0 multiplicity as a function of the Lund s/u parameter	132

List of Tables

1.1	Abbreviations for ALEPH sub-detectors	1
1.2	A 32-bit Fastbus Geographic Address	9
3.1	Data Representation Terminology: Equivalents	48
6.1	Predicted origin of K^0 s -quark	81
7.1	The Properties of Standard V^0 Particles	90
7.2	Summary of cuts applied to V^0 candidates	93
7.3	The Sources of V^0 Finding Efficiency Losses	97
9.1	References for other Centre-of-Mass Energy Experimental Results	127
A.1	The mean x resolution for each x bin	135
A.2	Variation in K_S^0 multiplicity with cuts	136
B.1	K_S^0 efficiency from simulated data	139
B.2	Measured K_S^0 signal — real data	139
B.3	Measured K_S^0 signal — Monte Carlo data	140
B.4	Correction factors for initial state radiation	141
B.5	x Spectrum for process $K_S^0 \rightarrow \pi^+\pi^-$ — real data	142
B.6	x Spectrum for process $K_S^0 \rightarrow \pi^+\pi^-$ — Monte Carlo	142
B.7	Extrapolated lowest three bins of x Spectrum for $K_S^0 \rightarrow \pi^+\pi^-$	142
B.8	$N_{K_S^0}/N_{ch}$ in x — real data	143

B.9 $N_{K_S^0}/N_{ch}$ in x — Monte Carlo	143
---	-----

Preface

High Energy Physics is a science of the vast and the minute. On the one hand we have the complexity and extreme physical size of the experiments and accelerators, and on the other, the simplicity and tiny scale of the fundamental particles and interactions. There is the enormous amount of data taken by an experiment, to be stored and analysed, and the intricacies of each software and hardware element. Together with this, there are the numbers of people involved in the projects, presenting very non-pure science problems of management and organisation. The ALEPH collaboration is itself made up from 30 institutions world-wide, involving around 450 scientists plus numerous technicians, engineers, secretaries, and administrators. These organisational problems represent a serious challenge for current and future experiments.

This thesis is based on work with ALEPH, a large, multi-purpose particle detector at the LEP (Large Electron-Positron) accelerator at CERN. The first part describes the development of the Data Acquisition System (DAQ) for the experiment, and how some of the problems mentioned above are combatted. The second part concentrates on a study of K^0 production in ALEPH.

The LEP Collider

The LEP Collider is a 27 km circumference machine designed to produce e^+e^- collisions at a centre-of-mass energy above 80 GeV. For LEP I, the maximum energy attainable is 100 GeV, a range which includes the Z^0 boson peak, allowing detailed studies of Z^0 reactions. The initial design luminosity is $1 \times 10^{31} \text{ cm}^{-2} \text{ s}^{-1}$, which, for the reaction $Z^0 \rightarrow q\bar{q}$ gives at maximum approximately 1 event per second. This is an important number for the detector as it represents a minimum performance requirement to avoid loss of events (dead-time). The total luminosity received by ALEPH for the 1989 and 1990 runs was 1020.64 (in 1989) plus 7449.535 (in 1990) nb^{-1} . This corresponds to a total of 184,998 hadronic events. For this analysis, a selection of events from the 1990 data sample, amounting to 120,551 events, was used.

The Detector

ALEPH is a general-purpose particle detector built to measure the properties of events created by e^+e^- collisions in LEP. Typical events are complex, having many particles distributed over 4π in the form of jets. Although the event rate is quite high at the Z^0 peak, at higher energies (LEP II) it may decrease by two or three orders of magnitude. ALEPH is designed to trigger on all interesting events, even with higher luminosity, in particular those producing a Z^0 . The aim is also to accumulate as much data per event

as necessary, over the maximum solid angle, in order not to lose information about the event.

ALEPH is designed to have excellent charged particle spatial and momentum resolution and good lepton and photon identification. These requirements dictated the form of the detector. Practical limitations such as physical size, cost, manpower and available technology necessitate the resulting apparatus being a compromise. The detector is described briefly in Chapter 1 and covered in depth in [1, 2].

Outline

A modern High Energy Physics experiment can be separated into four distinct concerns:

1. The Detector
2. Data Acquisition
3. Data Processing
4. Physics Analysis

The detector is a conglomerate of the detection media, support structures, signal amplification modules and data transmission lines, plus the infrastructure necessary for it to function. This hardware is then connected to the Data Acquisition, which collects all the information from an event from all the detector electronics and puts it into a coherent form. The Data Acquisition system itself is a hybrid of hardware and software, where part of the software is closely coupled to the processors embedded within it. The Data Processing stage turns the raw experimental data into values more appropriate for measurements. Physics analysis is then done on this data. This involves the input of physics theory and the extraction of results via computer, and is exclusively done in software. This thesis presents some of the work from the software areas of both Data Acquisition and Analysis.

Chapter 1 forms the introduction to the detector and the work from the DAQ area presented later on. It describes the experimental environment: the hardware of the detector and the computer system. Chapter 2 outlines the development of the DAQ system from requirements, through the design stage, to the final architecture. Chapters 3 and 4 are a detailed examination of part of the author's contribution to the DAQ system software. Chapter 5 is a final summary of the work and some conclusions and speculation on the system and relevant issues.

Chapter 6 explains the overall analysis chain of Chapters 7 through 9 and provides some

theoretical background on the K^0 particle. Chapter 7 examines the procedure for reconstructing tracks and events from the raw data. The algorithm for finding V^0 particles is discussed. Chapter 8 explains the techniques used to enhance the K^0 signal, and how the results are extracted from the data. Chapter 9 presents the results of the preceding chapter and gives some interpretation. A comparison with physics models and a discussion of further studies form the conclusion.

The work described in chapters 3 through 5 — the design and implementation of the Online and Readout databases, the associated libraries, the route-finding logic and the Partition Manager — is the author's own, with the exception of § 4.3, where the Resource Manager program was initially implemented by John Shade, of Digital. Chapters 8 and 9 describe the analysis performed by the author on the ALEPH data, and, in addition, the V^0 finding algorithm described in § 7.3 required some technical modifications to enable its use in this context.

Notation

All words or phrases in bold font are defined in the Glossary (page 144 onwards), in addition to any explanation in the text.

The use of a particle symbol is generally taken to include the corresponding anti-particle, except where stated otherwise. For example, " K^0 " includes " K^0 " and " $\overline{K}^0$ ".

References to χ^2 quantities always refer to the normalised χ^2 per degree of freedom.

Quantities surrounded by angular brackets, such as $\langle N_{K^0} \rangle$, imply average values, generally per event.

Typesetting Information

This document was typeset using $\text{\LaTeX}$ on a Vax. The pictures contained herein are all in PostScript format, and were produced in one of the following ways:

- From HBOOK4 histograms using PAW;
- From HBOOK4 histograms using MN_FIT;
- Using the Sight program on a VaxStation, and converting the resulting file to PostScript;
- Using a Macintosh, generating a PostScript file and prepending the Apple Macintosh PostScript Library to it.

The L^AT_EX style used was based on *report* with some modifications by the Author, Dr. Lucas Taylor and Simon Patton. The Bibliography information was processed using the BibTex program.

“The most potentially creative yet soul-destroying thing in the entire universe is a blank sheet of paper.”

Anyd, October 1989.

Chapter 1

The Experimental Environment

The ALEPH detector is introduced and its structure, both physical and logical, described. The Trigger system is outlined. The DAQ system, comprising both the computer system and the readout infrastructure, is covered in detail.

1.1 The Detector Organisation

ALEPH is constructed as a collection of independent sub-detectors. Their names are listed in table 1.1, and their functionality is described in § 1.2. Each of these sub-detectors has its own sensitive medium, hardware enclosures, necessary services, such as gas and cooling, signal wiring and all the associated electronics. A schematic of the apparatus is shown in figure 1.1.

Component	Abbreviation
Vertex Detector	VDET
Inner Tracking Chamber	ITC
Time Projection Chamber	TPC
Electromagnetic Calorimeter	ECAL
Hadronic Calorimeter	HCAL
Small Angle Tracker	SATR
Luminosity Calorimeter	LCAL
Bhabha Calorimeter	BCAL
Trigger (Level 1)	XTRG
Trigger (Level 2)	YLV2

Table 1.1: *The ALEPH Sub-detectors (in 1990) and their standard abbreviations*

This categorisation is split up further: the sub-detectors themselves are composed of **sub-components**, which correspond to mechanically independent sub-structures. These are subdivided into **slots**, which correspond to a single mechanical module. This hierarchial structure is reflected by the description of the detector in a relational database [3]. The three levels of detector are necessary, but there is no restriction on the horizontal divisions: a sub-detector such as the ITC consists of one slot and one sub-component (made from the slot). The TPC has 36 slots and 2 sub-components.

The properties of the detector relevant to the analysis presented in chapters 7 – 9 are discussed below. Some of the details are unnecessary for the DAQ discussion but are included now for completeness. Descriptions of the ALEPH detector, covering both the design, construction and calibration in greater depth than appropriate for this study, are available in [1, 2].

1.2 Description of the ALEPH detector

The reconstruction of an event requires knowledge of the particles produced in the interaction. The information about each particle of interest may be broadly separated into:

- Identification: of the intrinsic particle properties such as mass and charge;
- Kinematics: the momentum and direction in space;
- History: the point of creation, and of any subsequent decay.

No single detection technique is capable of providing all the necessary information, hence the apparatus is a combination of sub-detectors employing different detection methods.

1.2.1 The Minivertex Detector (VDET)

The VDET (labelled *a* in the illustration, fig. 1.1) came into operation for the 1990 period of data-taking. It was ¹ made up of two concentric layers of silicon wafers, at 90 mm and 120 mm radius, with 12 modules on the inner layer and 15 on the outer layer. Each layer provides 2048 points in *Z* and 15×512 or 12×512 in ϕ (with a small overlap), covering polar angles from 48° to 132° .

It was not used in the analysis described later as a number of problems were experienced in both the readout and the offline analysis.

¹A new Minivertex Detector has been installed for 1991.

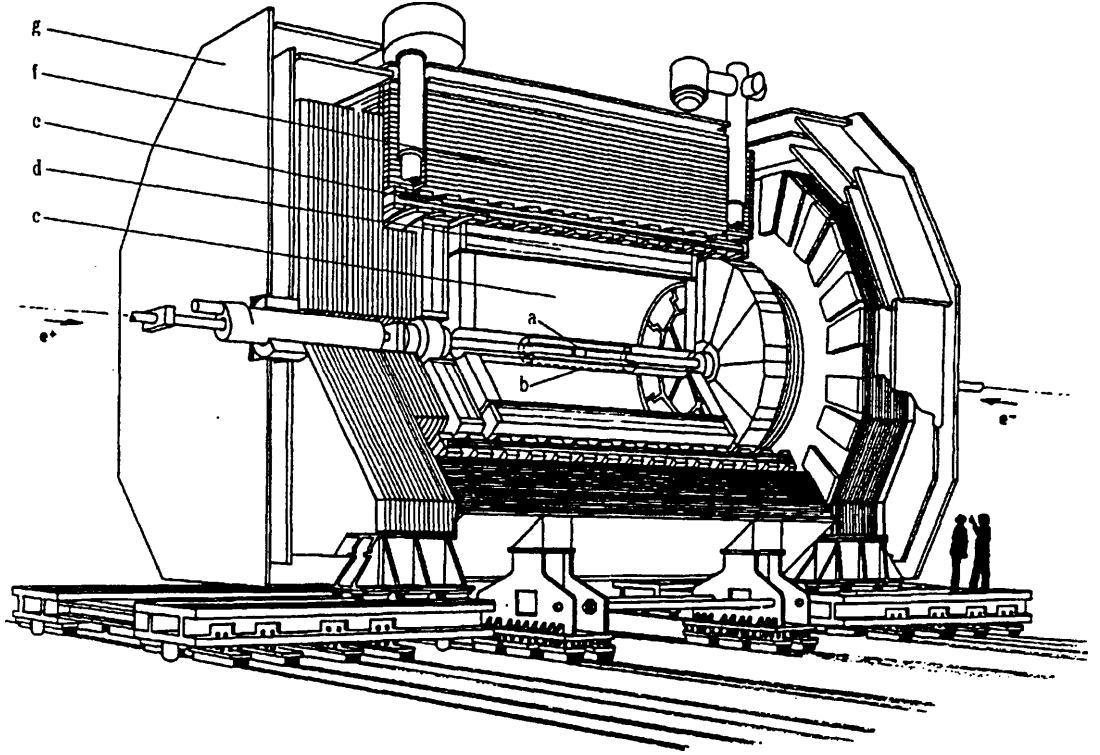


Figure 1.1: *The ALEPH Detector (1990)*

1.2.2 The Inner Tracking Chamber (ITC)

The ITC (labelled *b* in the illustration, fig. 1.1) is a cylindrical multiwire drift chamber with eight layers of sense wires parallel to the beam axis from 160 mm to 260 mm radius. Tracks with polar angles from 14° to 166° traverse all eight layers. The four layers closest to the beam contain 96 wires, with a maximum drift distance of 5.0 mm to 6.5 mm, whilst the outer four have 144 wires, with maximum drift distances from 4.7 mm to 6.0 mm.

The ITC serves a dual purpose in ALEPH: it provides up to 8 coordinates (r, ϕ, z) for charged particle tracks, and gives track information for the Level 1 trigger (see later). In the $r - \phi$ plane, positions are determined by wire number and drift time, achieving a resolution of about $100 \mu\text{m}$. The z coordinate is determined by the time difference between the pulses arriving at each end of the sense wires; the precision is around 50 mm.

1.2.3 The Time Projection Chamber (TPC)

The large cylindrical TPC (detector *c*, in fig. 1.1) is the main tracking chamber, giving up to 21 high resolution spatial coordinates for tracks with polar angles from 47° to 133° . The achieved spatial resolution of $\sigma_{r\phi} = 175 \mu\text{m}$ and σ_z between 0.74 and 1.5 mm, dependent on the position in the chamber, gives a momentum resolution of $\Delta p/p^2 = 0.0012 (\text{GeV}/c)^{-1}$ [4]. Combined with the ITC, the resolution becomes $\Delta p/p^2 = 0.0008 (\text{GeV}/c)^{-1}$ [4]. This is very close to the design value of $0.0007 (\text{GeV}/c)^{-1}$

indicating that the calibration and alignment procedures are well understood. Tracks with at least four coordinates are reconstructed down to 15° .

The sensitive volume, from an inner radius of 1.8 m to an outer radius of 3.8 m over a length of 4.4 m, contains an $ArCH_4$ (91%:9%) gas mixture maintained at atmospheric pressure. A high uniform electric field ($\pm z$ direction) is maintained between a central membrane (nominally $z = 0$, held at around -25 kV) and the instrumented end-plates. The two ends of the TPC (the TPC sub-components) are each constructed from eighteen sectors. Each sector has anode wires running perpendicular to the radial direction, at 4 mm spacing. Beneath these wire planes are rows of cathode pads measuring 30 mm along the radial direction and 6.7 mm perpendicular to it. Charged particles passing through the TPC generate ionisation electrons, which drift along the electric field towards the end-plates. z coordinates are derived from the arrival time of an ionisation pulse. The $r - \phi$ coordinates are the result of interpolating the signals induced on the cathode pads.

1.2.4 Solenoid

A uniform 1.5 T magnetic field is provided by a superconducting solenoidal coil (positioned at e , in the diagram fig. 1.1). This is necessary to introduce curvature in the motion of charged particles, in order to measure their momentum.

1.2.5 Calorimetry

The calorimeters provide useful cross-checks for particle identification and play a major role in the trigger. They were not used in the analysis study presented later.

1.2.6 The Electromagnetic Calorimeter (ECAL)

The ECAL (d , fig. 1.1) is divided into three sub-components, the barrel and two endcaps. These contain 24 and 2×6 slots respectively. The construction is a lead — wire chamber sandwich. The cathode readout is subdivided into a total of 73,728 projective towers. Each tower of about $1^\circ \times 1^\circ$ solid angle is read out in three stacks of 10, 23 and 12 layers (respectively 4, 9 and 9 radiation lengths). The signals from the 45 wire planes of each of the 36 modules are also read out.

1.2.7 The Hadron Calorimeter (HCAL)

The HCAL (f , fig. 1.1) has 23 layers of streamer tubes interleaved in the iron of the magnet return yoke, read out by a total of 4608 projective towers. Signals from each of

the tubes are also read out. The modules are rotated in azimuth by $\sim 2^\circ$ with respect to the electromagnetic calorimeter so that the inactive zones of the two calorimeters do not overlap. The two endcaps and the barrel of the hadron calorimeter cover polar angles down to 6° .

1.2.8 The Muon Chambers (MUON)

The Muon chambers (*g*, fig. 1.1) provide additional information for particles passing through the HCAL. These are usually muons, though some pions fail to be stopped in the HCAL, due either to punch-through (where they interact but within the multiple scattering cone expected for a muon) or sail-through (where they pass through without interacting).

1.2.9 The Luminosity Calorimeter (LCAL)

Similar in its construction and readout to ECAL, LCAL extends from 45 to 155 mrad, providing energy and position measurement of the showers produced by gammas or electrons. The LCAL is the main detector for low angle Bhabha events providing the beam luminosity measurement for ALEPH. The two ends are positioned between the ECAL end-cap inner edges and the beam-pipe.

1.2.10 The Small Angle Tracker (SATR)

The SATR has 9 planes of drift tubes, covering an angular range from 40 to 90 mrad. Its major role is to complement the function of the LCAL by giving a precise measurement of track angles of electrons in very low angle Bhabha events. It is mounted on the LCAL faces nearest to the interaction point.

1.3 The Trigger

ALEPH has three levels of trigger, which aim to record all visible Z^0 events with the minimum of background.

The sub-detectors used by the trigger system are: ECAL, HCAL, ITC, TPC, LCAL. The signals from these detectors can be combined into segments of solid angle. A physics trigger is obtained by establishing trigger conditions in each segment independently and then requiring particular combinations of segments or by global triggers such as total measured energy. A full discussion of the trigger design, philosophy and thresholds can

be found in [1, Part IX]. A basic aim was to trigger on single particles or energy clusters, and not, unless forced by large background, to require correlations between more than one segment. The trigger system is flexible enough to allow either case.

Hadronic events were triggered by two independent first level triggers:

1. an ECAL-based trigger, requiring a total energy on the wires of 6 *GeV* deposited in the ECAL barrel or 3 *GeV* in either of the ECAL endcaps or 1 *GeV* in both in coincidence;
2. an ITC-HCAL coincidence, for penetrating charged particles, requiring 6 ITC wire planes and 4 to 8 planes of HCAL tubes in the same azimuthal region.

Luminosity events were triggered in two ways:

1. a coincidence of > 15 *GeV* deposited in LCAL on one side with > 10 *GeV* on the other side, without requiring azimuthal correlation;
2. a single arm requirement of 32 *GeV* deposited in either side of LCAL.

In addition, prescaled single arm triggers with 10 and 15 *GeV* thresholds were recorded to provide an estimate of the beam related background.

All types of events were processed simultaneously through the same trigger system (with the same dead-time). In order to ensure that the events were counted during the same live-time, the list of enabled triggers and the status of each of the relevant subdetectors was recorded with each event; events were accepted only when both ECAL and LCAL were running and when both ECAL and LCAL triggers were enabled.

1.4 Introduction to the ALEPH Data Acquisition System

The primary role of the Data Acquisition (DAQ), or Online, system during normal data-taking can be summarised as follows:

- The collection, processing and storage of the data from the detector;
- The selection of interesting events over background;
- The control of the detector: ensuring it is in the correct operating state.

The algorithms employed in Online Software are on the whole simple, as the major purpose of the DAQ is the movement and organisation of data and control of the detector. This distinguishes the underlying philosophy of the system from Offline software, which is primarily concerned with the application of relatively complex physics and statistical algorithms to analyze the data.

The ALEPH DAQ system consists of a number of sub-systems: the front-end electronics, the readout controllers, the event builders, the trigger, the main computer system, workstations, slow control devices and the networks connecting them all. Additionally, there is the software controlling each part and the overall DAQ. These are described in the following sections. In addition, to ensure minimum wastage of beam time, the DAQ system must give rapid feedback on the integrity of the data as it is collected. This includes the detection of errors during the readout process, the monitoring of detector control parameters, checking that they lie within allowable limits, and the examination of an unbiased sample of data with enough statistics to check the performance of all hardware channels.

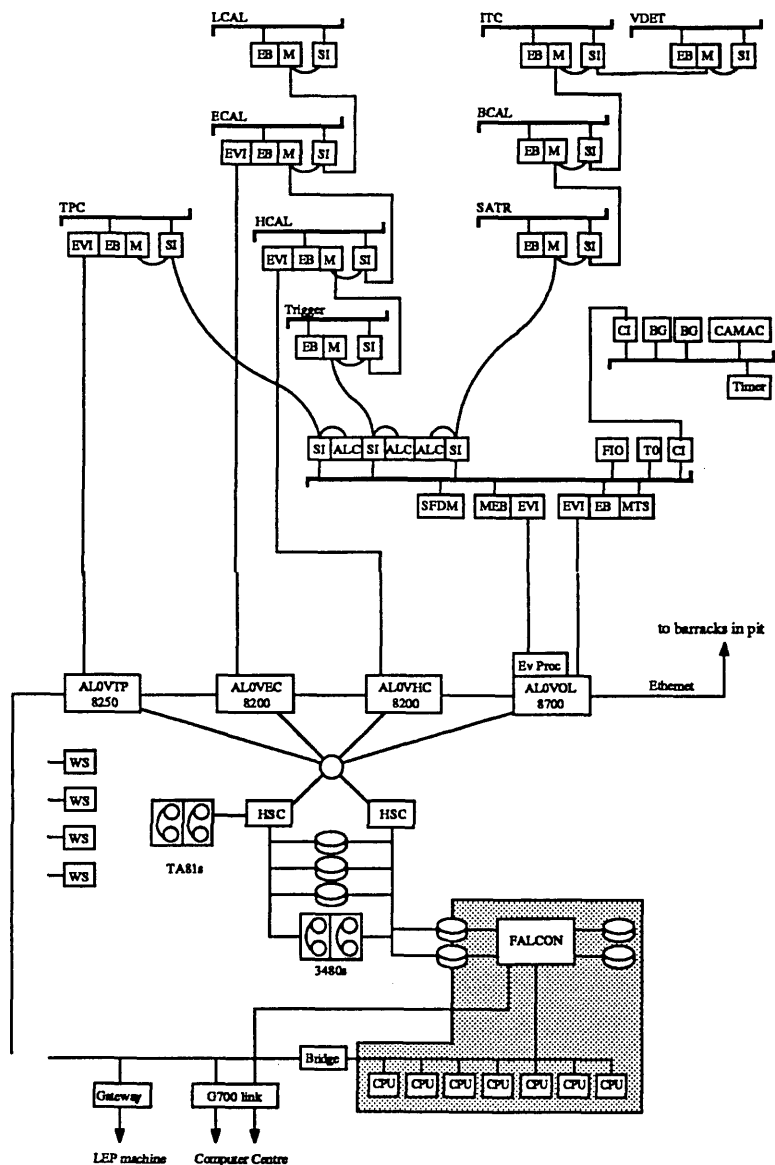
1.5 The DAQ System Hardware Architecture

The architecture of the DAQ is dictated to a great extent by the detector structure described in § 1.1: it follows a similar hierarchial structure. The hardware of the system was based on the Fastbus [5] standard with a purpose-built interface to a VaxCluster. The purpose of Fastbus is to provide, in a flexible manner, a high-bandwidth path for data transfer from the lowest levels of the front-end electronics, through the Event Builders to the Vax interface. The overall system is depicted later in fig. 2.3.

1.5.1 The Fastbus Sub-system

The Fastbus sub-system consists of approximately 150 Fastbus crates (about 1200 modules) connected in a tree, with Segment Interconnects (SIs) joining the branches (see fig.1.2). The tree structure is split up according to sub-detector, following exactly the division of the physical hardware.

The actual connectivity of the system is programmable. The SIs provide the facility to pass only certain categories of signals to branches connected either side. This can be used to isolate parts of the tree at particular times, such as for testing. As each SI must be programmed to provide a different set of routes, some form of database is required to store the information, which can also maintain the state of the system at any time, avoiding the need to interrogate the modules themselves. The ALEPH Fastbus database [6, 7] was developed for this purpose.

Figure 1.2: *The ALEPH Fastbus System*

bit 31 — bit 24	bit 23 — bit 8	bit 7 — bit 0
group	Must be zero	slot

Table 1.2: *A 32-bit Fastbus Geographic Address*

Modules are defined by means of an address. This consist of two parts: a Group address and a Geographic address (essentially the slot number of the module in a crate). Each address is unique in the system, and thus is used as the principal identifier of modules in the database. The way they are constructed is depicted in table 1.2.

Fastbus provides several means of inter-module communication. Any module initiating a data transfer is termed a **Master**; one responding to this initiation a **Slave**. Only one module can be Bus Master at one time. Thus a simple read/write between modules involves at least two Master/Slave transitions, and, as these must be arbitrated each time, this method is somewhat inefficient. However, there are also service requests, which allow one module to interrupt another which can then read data from it. This means the master can await many requests, and then read out all the modules in one go, reducing arbitration overhead. Finally, there are interrupt messages, which combine an interrupt of another module with the delivery of an entire message.

1.5.2 The Readout Modules

Connected to Fastbus are the readout modules which collect the data generated by the detector, process and transfer it to the next stage in the pipeline, until it arrives in the main computer. The different stages are detailed below.

- **Front-end Electronics:** This is the detector-specific hardware that performs all of the analogue processing on signals coming from the detector. Frequently, the electronics are positioned in close spatial proximity to the detector — usually to amplify any signal near to its generation to avoid noise pickup.
- **Digitisers:** Used to generate the digital representation of signals so they can be transferred through Fastbus and simple to process by computer. They are in most cases dedicated (but standard) Fastbus units. Some, such as those for the TPC, have been developed specially at CERN.
- **Readout Controllers (ROC)s:** These modules are the lowest masters in the system. They read out the digitisers, perform zero-suppression, and also pedestal subtraction.

- **Event Builders:** Event Builders (EBs), described in detail in [8, 9], are Fastbus masters that collect and order the sub-events from whichever readout modules lie below them in the tree. The resulting data is then passed up to the next DAQ level, another Event Builder or a Vax. Tasks such as re-formatting and data monitoring are also done by software running concurrently on the EBs.

The main programmable processing elements of the readout, the EBs and ROCs, are referred to collectively as Readout Processors (ROPs).

1.5.3 The Trigger Sub-system

A separate system for distributing the trigger signals was required because of the small time available for trigger decisions. Passing the information through Fastbus would have been impractical due to excessive jitter in the time needed to pass the signal to every module. Thus the signals are passed using dedicated cables and a tree of programmable routing boxes (known as “Fan-in, Fan-Out” or FIO units — see § 3.4.1) from the trigger decision logic to the front-end electronics and Readout Controllers. The Trigger and timing signals are generated by a **Trigger Supervisor**, which receives the trigger decisions from the Level 1 and 2 hardware logic plus a clock synchronised to the LEP bunch-crossing. Part of this system is shown in fig. 1.3.

1.5.4 The Online Computers

The Online computer system comprises four large Vax computers and about 25 VaxStations connected in the form of a cluster. This allows all the computers (or nodes) to share disks, which simplifies the organisation of the system greatly and it provides redundancy: a node can fail and the system will continue to take data.

The VMS operating system that runs on the VaxCluster is used extensively by the Online software; the features used are covered in the relevant sections. The microprocessors in the DAQ system are of the Motorola 68020 variety, and run the OS9 operating system. The two provide multi-task capability by time-slicing between processes. Any process can be considered independent of any another, so that DAQ functions that are autonomous can be implemented as separate processes. The two operating systems also supply facilities for file access, interprocess communication and node-to-node communication.

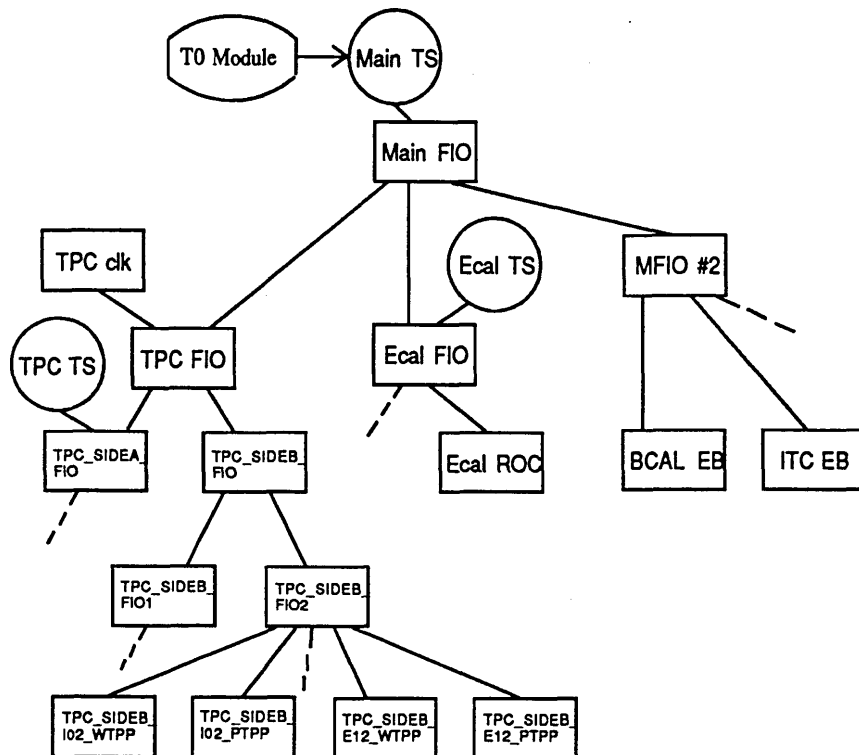


Figure 1.3: *Part of the Trigger Signal Distribution Tree*

1.6 ALEPH Software

Given the size of the experiment, the number of data channels, the event rate and the multiplicity of the interactions themselves, writing good software becomes a difficult task. The problems that arise in large software projects are essentially due to the increased complexity. This results from:

- Too many communication paths
Without a well-defined hierarchy, the number of communication paths grows rapidly with the number of people in the project.
- Too much information
The volume of facts and details about the project become impossible to follow manually.
- Too many side-effects in the program
Inter-relationships between different parts of the program logic and data grow and become increasingly difficult to understand.
- Too large for one person to understand
Beyond a certain size, no one person can grasp the details of the whole project.

- Too many changes
Too high a rate of changes or updates cannot be followed by documentation or programmers' knowledge.
- Lack of expertise
In HEP, most software is currently written by Physicists, not Software Engineers, who often have little experience of writing well-engineered or "good" software.
- Inhomogeneity of design and implementation
The distributed development of software over a number of universities and labs with no strict design and coding standards can lead to very inconsistent approaches to problems and interfaces.

1.6.1 Software Engineering and Data Modelling Techniques

Following experience from industry [10], the ALEPH collaboration decided to adopt some of the ideas and methods of Software Engineering for both the Offline and Online software development [11]. The principles of Software Engineering (SE) aim to produce software that does what was intended and be delivered in the timescale required, with accurate documentation. The techniques of SE attempt to rationalise and impose a certain discipline on the **Software Life-Cycle**, the process of software development from initial requirements, design specifications, prototyping, coding, testing, debugging, documentation and maintenance. ALEPH chose the **SASD** technique, along with the Entity-Relationship Data Model (ERD Model) which seemed an appropriate form with which to model Physics data.

With the ERD technique, data is grouped according to some criterion. These groups are termed entities. Thus a "Track" entity might contain such things as measured momentum, measured charge, and start position. These are termed the "attributes" of the entity. On a diagram, the entity is drawn as a box, with its attributes listed inside e.g. figures 3.2 and 3.3. Supposing the Track entity is related to another, called "Hit", by "used-in-track-fit", this is shown as an arrow from Hit to Track. Since more than one hit is used on a track, this is denoted by a double-headed arrow. One advantage of these techniques is that they provide a simple visual representation of the system, and thus are also useful documentation. Additionally, having to think through the system description in a rigorous, systematic way can give some insight into potential problems or solutions.

The ADAMO system [12] provides the facilities necessary for translating the data model into code and a **Data Dictionary**. The data model is described in a language known as DDL, or Data Definition Language — very simply it is a textual description of a group of Entity-Relationship Diagrams. The Data Dictionary is a repository for all data, both the values thereof and their description. Tools provided by the ADAMO system

allow the compilation of DDL source files into a Data Dictionary, where checks on data consistency and redundancy can be made. Additionally, documentation can be generated automatically from the data descriptions. The Offline reconstruction program (**JULIA**) has been developed using this method [13], initially with very stringent guidelines [14] and the entire physical detector construction is specified in a database described by ADAMO (see [15] for other applications). One result of this is the availability, on any ALEPH computer, of a description of each data structure used by the Offline analysis programs.

The Online Software also made large use of these techniques, particularly for the design of a number of databases used by the system (see Chapter 3). An overall picture of the methods used is presented in [16].

Chapter 2

The Online System Software

2.1 Overview of the DAQ System

The design of the DAQ system was preceded by a study of the aims of the system and a review of the available and foreseeable technology. This led to the definition of a set of requirements [17] and then a detailed simulation of the system [18] provided a list of desirable design features. These features included asynchronous readout of modules, which makes the readout time a function of the average for each module, as opposed to the maximum of all modules in the case of synchronous readout. The study also demonstrated that the tree structure (from fig. 1.2) would give optimal performance given the planned detector organisation, providing data buffering and synchronisation were performed at the nodes. The functional specifications for ALEPH were published in [19]. Since these are fundamental to the design of the system architecture, they will be examined in some detail.

2.1.1 Requirements

The development of a DAQ system involves a number of distinct, but overlapping, phases:

1. Setting-up
Installing all the hardware, preparing software, etc.
2. Testing
Checking all the hardware is functioning. Ensuring software fulfills its intended function.
3. Debugging
Fixing initial hardware errors, connecting cables to the right channels, correcting software bugs, etc.

4. Calibration

The calculation (and storage) of pedestals, zero offsets, etc.

5. Data-taking

Running the detector with cosmic-ray or beam-beam data being read out and recorded.

6. Maintenance

Repairing hardware faults, “improving” software, etc.

The intention of the ALEPH DAQ system was to be as flexible, reliable and at the same time efficient, as possible. This meant that phases 1, 2 and 3 should proceed as quickly as possible, whilst 4, 5 and 6 could be performed simultaneously for sub-sections of the DAQ, at any appropriate time. For example, the TPC would be able to perform a cosmic run whilst ECAL calibrated pedestals and HCAL fixed dead channels. The phases listed above bear a close similarity to those of the Software Lifecycle (see page 12).

To achieve this, it must be possible to split the DAQ system up into smaller parts in such a way that they can individually support all the above-mentioned activities. This requires, in particular, a mechanism for distributing different trigger signals independently to these parts. Some can be Physics triggers, others for calibration; the rates may not be the same, nor the trigger conditions. Provision for this at the hardware level is done in two ways. Firstly, dedicated modules called Trigger Supervisors (TS) are used to generate the trigger signals. They can operate stand-alone or from an input source that satisfies the interface protocol [20]. Secondly, the paths of the trigger signals from the TS to wherever needed are programmable. This was achieved by the use of FIO units as mentioned in § 1.5.3.

In Fastbus, and on the Vax, it is necessary to keep the data streams from these parts of the DAQ separate.

2.1.2 The Partition Concept

The term **partition** is used to represent the concept stated above: a sub-set of the DAQ system capable of performing all the activities involved in the DAQ development phases. Thus a partition in its widest sense is the environment providing these features. This can be any subset of the DAQ, including the detector hardware, electronics, readout modules and processes running on the Vax. They form a data pipeline which, for data-taking, terminates on a cassette or disk file.

If one views the DAQ system as an upside-down tree structure, it is clear that data is flowing from the “leaves”, through the branches, into the trunk of the tree. For the

concept of a partition to be worthwhile, it is necessary that separate data streams can flow up through the tree, independent of one another.

The DAQ system requirements were given in [19] in terms of partitions. Since the initial study to determine the requisite features of the DAQ system, practical experience from beam tests of equipment and from the development and testing of the system itself led to modifications to these requirements. Some of the changes were relatively dramatic and demonstrated the necessity to build into the design the ability to modify characteristics of the system ¹.

The final list can be summarised as follows:

1. Partitioning

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. The minimal partition

The minimal data-taking partition can be one Readout Controller. This would normally also include a Trigger Supervisor and one or more Event Builders. The data is read out into a control computer. The control computer must therefore be capable of reading the readout controllers directly. Empty partitions are allowed for testing.

3. Trigger source

All the Readout Controllers within one partition are configured to respond to the same trigger source or none — this choice could be made at run startup time. One or more sub-partitions responding to the same trigger are normally 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. A facility must be available to provide Trigger spying which allows trigger signals to be sent to connected ROCs, but for the response to be ignored. Different partitions can, in this case, use the same trigger, though clearly the data will not be synchronised. For the case of no Trigger Supervisor, the trigger signals must be generated externally (in hardware) or simulated in software.

4. The ALEPH Partition

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

5. Multiple Parallel Data Paths

All partitions are controlled centrally though the data stream from one partition can

¹From F.Brooks, *The Mythical Man-Month*: .

"The only constancy is change itself ... The first step is to accept the fact of change as a way of life rather than as an untoward and annoying exception."

end in any computer. Many partitions must be able to run simultaneously from one or more computers with Fastbus interfaces, with data being stored on independent channels. This provides partial redundancy to computer system failure.

6. Partition Descriptions

A full record of all partition descriptions must be maintained, along with their current status.

7. Partition Scope

Many processes can run in the context of a single partition. However, a process (or task) running in one partition may not interfere with any other partition.

8. Partition Configuration

Changes in the configuration of partitions are restricted to inactive partitions. A Fastbus interface is required in order to perform the re-configuration. Re-configuration of the partitions must be made by software only, without any recabling being necessary. Re-configuration operations cannot affect the configuration of another partition, but are permitted to cause a pause in data-taking (related to the properties of an FIO module — see § 3.4.2) as the sole exclusion to requirement 7.

9. Dead-Time

Dead-time should be minimised. For this, it should be possible to:

Change run without stopping the trigger; Change partition without stopping the DAQ completely; Turn Readout Processors on and off without changing partition; Use asynchronous SR handling for readout, as this avoids having to wait for the slowest processor; Use a multi-level Trigger for maximum flexibility; Detect and act upon errors as automatically as reasonable.

Any dead-time should be due to the trigger and not the control computer.

10. User Interface

The system should be as transparent to the operator as possible, so that any learning curve is as gentle as possible.

There are also a number of implicit yet unstated requirements. These include such things as “Must acquire ALEPH data and record on cassette”, “Must be reliable”. They must also be satisfied.

These requirements place design restrictions on both hardware and software. The hardware design had to be fixed early on, traditionally having the longer lead-in time. Thus the software design was provided with a hardware environment within which it should satisfy the aforementioned criteria. Naturally, account was taken of what was deemed possible to do in software.

2.2 Requirement Solutions

All of these requirements were satisfied by the final system. In particular, the following proved very important to the DAQ system function as a whole:

- **Partitions and Multiple data paths**
From the Event Builders upward, each readout module was capable of handling multiple data streams. The Vax-Fastbus interface, the EVI [21, 22], also provided this functionality. The data that arrived in the Vax was kept in a buffer that allowed any number of separate partitions to read from and write to it [23].
- **Partition Re-configuring**
This was dealt with in hardware by the provision of the FIO modules. Software was supplied to perform the programming (see § 3.4.2).
- **Partition Descriptions and Scope**
Two software packages provided this functionality. They are described in §§ 4.3, 4.4.
- **User Interface**
A number of methods were used, but in particular, a standardised menu package was used for most user interaction (see § 2.5.3).

Another primary issue was the establishment of protocols between the different levels in the Readout hierarchy and in the Trigger system. By enforcing a complex sequence to proceed via a number of well-defined steps, the aim is to simplify the testing and troubleshooting phases. The two main system protocols are explained in the next sections.

2.2.1 Readout Protocol

The readout consists of several levels, each higher one containing a decreased number of modules and handling all the data from below. The Readout Protocol is implemented using service requests (SRs). When Readout Controllers (ROCs) have valid data, they issue a service request. This is picked up by the Service Request Handler (or SRH), a program running on a Fastbus master nominated for the task. The SRH contains all the information telling it where service requests should arrive from, and which masters should then perform the readout. A Fastbus message is sent from the SRH to the master of the readout device, the next module upstream in the data pipeline, supplying the Fastbus address of the slave and other details. This master will read the sub-event descriptor from the slave module, giving the address and length of data, and perform a block read. Once it has collected all its data, the master will then issue an SR itself, and the procedure continues until all the data for an event is held within the main computer. This dedicated

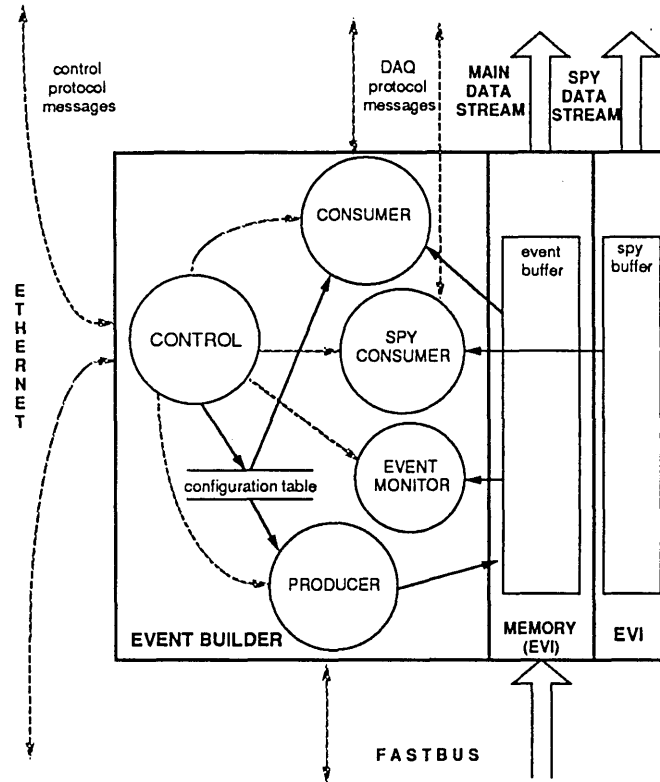


Figure 2.1: *Data Flow within an Event Builder*

SRH solution was dictated by a need to provide a SRH at each Fastbus tree layer, to avoid having to re-configure the Fastbus routing when partitions change (each partition has a Fastbus Broadcast Class allocated to it for the Level 2 Trigger broadcast).

Each Event Builder expects all the sources below it in the tree to issue a request to be read out, even if the data is all zero. It checks the trigger number present in each event to detect any mismatches. In this manner, one can ensure that all the data from an event is read out and if not issue a warning. This provides a very powerful facility for tracing errors.

The readout of modules upstream is performed by a **producer** task, and the data transmitted by a **consumer** task (fig. 2.1). The data is read into a buffer, controlled by the Buffer Manager (BM) [24]. This program provides memory allocation and deallocation facilities, separate pipelines for different data streams and access to restricted spy memory. Any reformatting or monitoring task can request to read data before it is passed on to the next stage by the consumer. The BM is an element of the DAQ service layer, outlined in § 2.5. This method of systematically filling a buffer (**event building**), signalling completion with an SR and being read out is used for all the levels in the readout. This provides a mechanism for skipping a level, if necessary, by passing the job of responding to SRs up to the next.

2.2.2 Trigger Protocol

The trigger system operates at a much higher rate than the readout. Thus the trigger protocol is implemented in hardware. It is conceptually a little more complicated than the readout, as it involves a number of sub-detectors plus the Trigger logic. The full protocol is defined in [20]; however, the fundamentals are as follows. The timing signals are all controlled by the Trigger Supervisor (TS) [25], which is a programmable device capable of down-scaling any given input signal by varying factors, detecting any protocol errors and timeouts and also capable of generating the Fastbus broadcasts containing information required by each readout module in an event (trigger number, trigger mask, etc.). It also contains counters which can be used for checking correct function and dead-time calculations. The TS is connected to the ROCs via a tree of Fan-In Fan-Out (or FIO) units. These are essentially programmable switches, and are described in § 3.4.1.

The Trigger system for data (i.e. normal data-taking) is based on a three-level system. The Physics criteria for each level were explained briefly in § 1.2. Essentially the first level makes an initial quick ($5\mu s$) but simple trigger decision to reduce the data rate to one which is acceptable for the gating of the TPC signals. This rate is limited to avoid the build-up of space-charge in the drift regions of the TPC. The second level decision is made using signals from the TPC trigger pads to supplement the track information. The TPC drift time is approximately $50\mu s$, thus the Level 2 decision is not made until after this delay. This level then further reduces the rate to one which the third level (based on a software processor) can use all the available event data to make a selection decision. Only after a Level 2 Yes are all the subdetector data read out, minimising dead time. In 1990, Level 1 ran at a rate up to 20 Hz (design maximum approximately 500 Hz) whilst full events (Level 3) were written out at around 0.3 Hz on average.

The standard Trigger sequence has three possibilities which correspond to:

1. Event rejected by Level 1 — here readout has not started;
2. Event accepted by Level 1, rejected by Level 2 — the readout modules are reset;
3. Event accepted by Level 1 and by Level 2 — the event data is read out.

Level 3 decisions are not part of the trigger protocol. Any of the large number of possible error states — timeouts or protocol violations — optionally result in the trigger signals being frozen, allowing analysis of the situation.

Listed below are the signals generated as input for the Trigger Supervisor module. BXIN and EBX come from the $T0$ module [26], originating either from the LEP pickups, from the HCAL cosmic signal or from an internally generated clock. L1STR, LVL1, L2STR, LVL2, TMT and Trigger Mask come from the trigger electronics. REFIN is generated (when necessary) by the calorimeters.

- **BXIN Bunch crossing input:** This is derived from the LEP bunch pickup signal that comes every $22\ \mu\text{s}$ by the use of a phase-locked-loop to ensure its stability. It arrives $2\ \mu\text{s}$ before the true bunch crossing and thus is actually delayed from the previous crossing. In cosmic mode it arrives just after a cosmic ray is detected.
- **EBX Early BX:** Like BXIN, this is a delayed signal from the previous bunch crossing, but is $5\ \mu\text{s}$ in advance of the true crossing.
- **L1STR Level 1 Strobe:** Signal for when Level 1 Trigger decision is available.
- **LVL1 Level 1 decision:** The result of the Level 1 trigger logic. It is sampled when L1STR goes high.
- **L2STR Level 2 Strobe:** Signal for when Level 2 Trigger decision is available.
- **LVL2 Level 2 decision:** The result of the Level 2 trigger logic. It is sampled when L2STR goes high.
- **TMT Random (eMpTy) event indicator:** Generated by the Trigger Logic. The Random Trigger.
- **Trigger Mask:** A bitmask describing the trigger conditions that were satisfied by the event. Generated by the Trigger logic. Transmitted to the ROCs by a Fastbus broadcast from the TS. This can be used to selectively turn off full readout of non-relevant ROCs (e.g. for Bhabha events), but has not yet been implemented.
- **REFIN Refresh Input:** This is a special signal used to prevent events during refresh cycles of the calorimeter electronics. It is a general purpose inhibit.

The TS produces a number of outputs (see below) and receives input from the connected ROCs. It is responsible for ensuring that the sequencing of the readout is correct, and hence has internal logic checks for timeouts and protocol violations.

- **BX:** This is the Trigger Supervisor's copy of BXIN, forwarded to the ROCs. It is used to calculate the t_0 for drift distances etc.
- **EGBX:** This is used by some sub-detectors to "prime" the readout hardware (like switch on gating voltages in the TPC). It is the EBX signal gated with the READY line input (i.e. the readout controllers are ready for an event).
- **GBX:** This is the gated version of BX, started when BX arrives and EGBX is high.
- **L1NO:** If the Level 1 trigger logic rejects the event, this signal is raised. This allows detectors to clear their readout quickly to avoid loss of beam time. Causes the TPC gating to be closed.

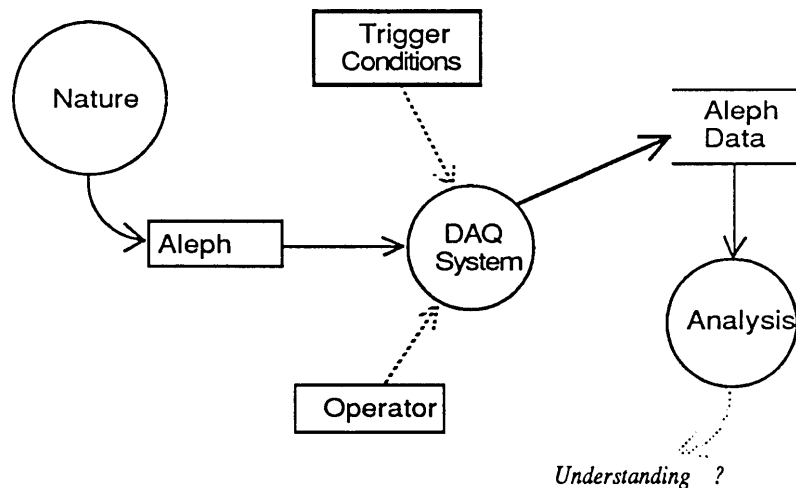
- L1YES: Signals that the event is accepted by Level 1 trigger logic. Causes the TPC gating to remain open. Initiates front-end readout for some sub-detectors.
- L2NO: Level 2 version of L1NO. Stops readout and resets readout processors.
- L2YES: Level 2 version of L1YES. Allows readout to continue if started, otherwise starts it.
- L2Broadcast: Fastbus broadcast sent to all readout processors giving information to be incorporated in event data: Trigger Number, Trigger Mask, etc.
- GREF: This is a copy of REFIN active only in cosmic mode.
- READY: This is the ready state of the TS. The TS is not ready whenever BUSY, L2YES, REFIN or PAUSEREQ are active.

The ROC replies with some signals of its own. These are as follows:

- BUSY: Each ROC holds this high until it has finished all readout and processing. There is a maximum allowed time for this to catch ROCs that fall into infinite (or excessive) processing loops, or some other failure [20]. The signals from all ROCs are ORed in the FIO unit.
- GBXACK: This is simply the acknowledgement that GBX was received and that the ROC is capable of reading out data. At the ROC level it is the same as BUSY. As this must be returned within $2\ \mu s$ it is implemented in hardware. The signals from all ROCs are ANDed in the FIO unit.
- PAUSEREQ; This is a signal that can be used by a ROC to request an extended readout time (the allowed BUSY timeout). The signals from all ROCs are ORed in the FIO unit.

The trigger sequence for an event passing Level 1 and 2 would go as follows:

1. TS receives EBX from $T0$ module. Generates EGBX.
2. TS receives BXIN from $T0$ module. Generates GBX.
3. TS receives GBXACK from ROCs.
4. TS receives L1STR. Samples LVL1. Generates L1YES.
5. TS receives L2STR. Samples LVL2. Generates L2YES.
6. TS generates L2Broadcast.
7. TS waits (for a maximum of about 6 seconds) until BUSY falls.
8. Repeat.

Figure 2.2: *DAQ Context Diagram*

2.3 Software Design

Much as the hardware design had involved the use of models to describe and help simulate the detector readout, software models were used in the design of the software architecture. As mentioned earlier, the ADAMO system [12] was used for data modelling, based on entity relationship diagrams [27]. In the SASD methodology, data flow diagrams [28] provide a way of describing how procedures accept, modify and pass on data. With the DAQ system, we have a number of distinct operational phases, which are appropriate to model using state transition diagrams [29] (see § 2.3.2).

The Context Diagram (fig.2.2) represents the overall view of the system. This picture is then sub-divided into increasingly more detailed views of sub-systems, sub-sub-systems, ... The reason for using a model is to foresee where the logical boundaries of the sub-systems lie and how the interfaces between them must be defined. If the software can be written in a manner reflecting these boundaries, a much more modular system will result.

2.3.1 Data and Control Flow

Different techniques are used to represent different parts of the system. A data model provides a way of specifying the content and format of the data at any point in the system, and a schematic representation of the path(s) for the flow of data. This is a powerful way of checking the consistency of the data throughout the system. The data content is specified using an Entity Relationship diagram, or ERD [27]. With these, one can produce a tabular description of the data. A number of examples of the form of

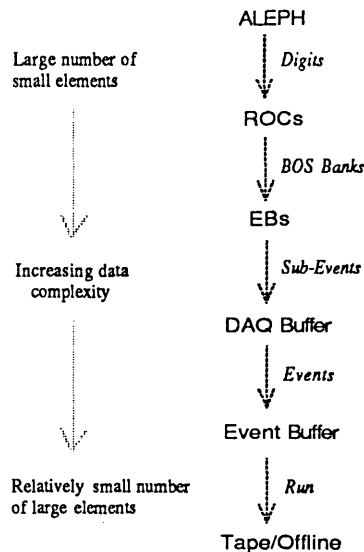


Figure 2.3: *Overall DAQ Data Flow*

data description occur in the following chapters. The simple data flow diagram, shown in fig.2.3, illustrates how the information produced by the experiment undergoes a number of transformations before it reaches the offline stage. Clearly, the data flow here is a sequential one.

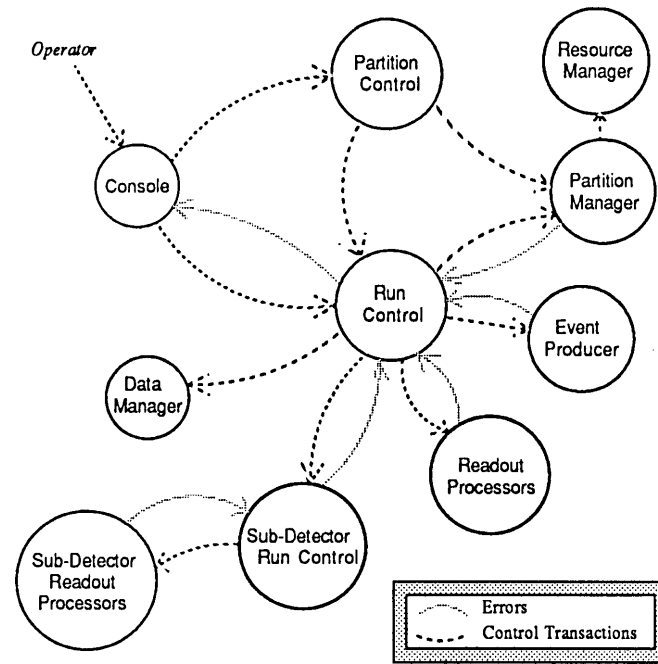
The addition of control flows to this type of diagram permits the modelling of the transactions occurring in a multi-tasking system, one in which many jobs can be performed at the same time ². In a multi-tasking system, the designer cannot rely on the simplicity of sequential $A \rightarrow B \rightarrow C$ style processing; however, they provide performance gains and flexibility, which outweigh the extra complexity.

A transaction in this sense is a control or data flow that can be generated at any time. No longer do we have a purely sequential data flow.

2.3.2 The Finite State Machine Model

The control of complex systems can be simplified by modelling the system as a finite state machine i.e. by defining the states in which the system can be and the transitions between the states that are permitted. This means that at any given time it is possible to determine precisely which part of the control logic of the system is active at that time. The machine is depicted by a State Transition Diagram or STD. The ALEPH DAQ was modelled in this way and runs using the software protocol illustrated by the states and

²In many computer systems *simultaneous* is not exactly accurate - the system will time-slice between the tasks - but to the user it is effectively the case.

Figure 2.4: *DAQ Control Flow*

transitions in figure 2.5.

Within large systems there are usually dependencies between the states in which the various components of the system can find themselves. For example, if the run is “active” then the TPC high voltage must be “on”. This is modelled using a **Petri Net**, a simple example of which is shown in figure 2.6. With many levels in a system, a cascade of dependencies results. The logic needed to implement state transitions and invoke dependent transitions between the different components can thus be very complicated. For ALEPH, a library was provided (§ 2.5.2) to facilitate this.

How a Finite State Machine Works

A system described by a state transition diagram, is in an undefined state to begin with. Each sub-system (dependency) must therefore be brought into a known state. This is what is termed initialisation. Thereafter, each change of state is induced by a condition. If the condition is valid (i.e. it is expected for the current state), a state transition is attempted. The FSM model allows dependent tasks to be included in the system at a lower level. This implies that for a high level transition to be successful, all the dependent lower-level tasks must also complete their transitions successfully. The hierarchy thus defined allows any errors to be propagated up through the system — the **Error** condition is valid for any state — and causes a transition to the appropriate error state at the highest level. By examining the success or failure of each dependent task, the source of the error can be determined. Ideally, each possible source of error could be mapped to a transition, but

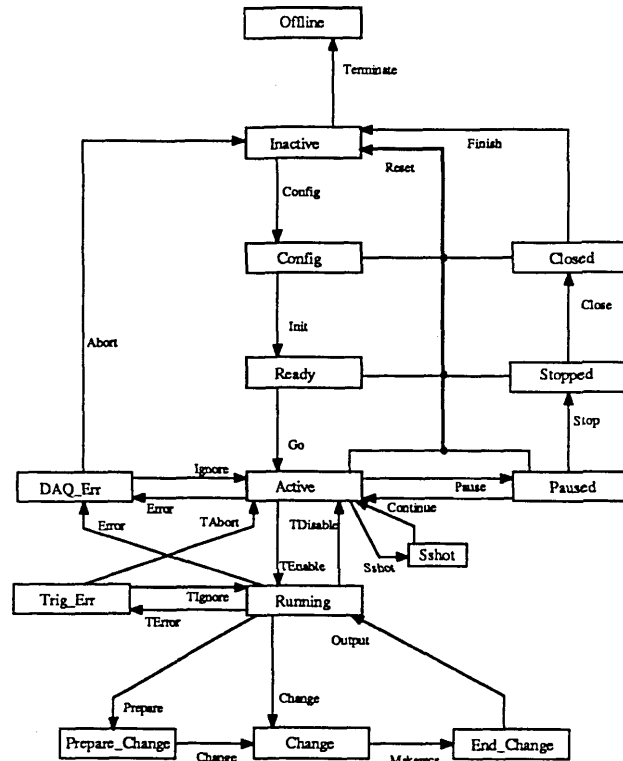


Figure 2.5: *The ALEPH DAQ Run Protocol State Transition Diagram*

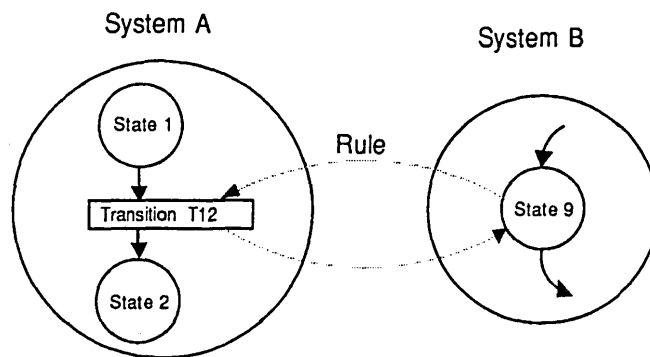
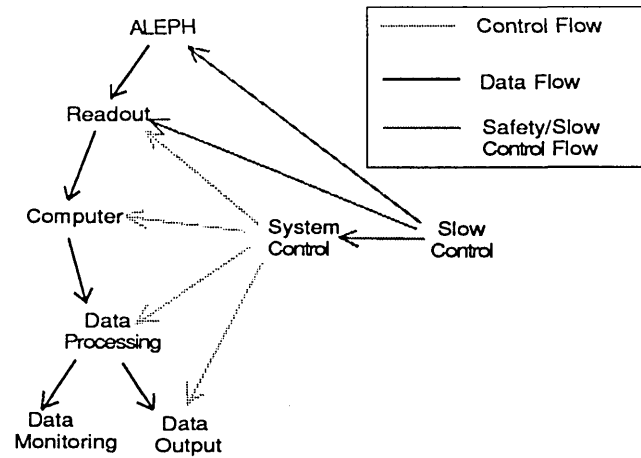


Figure 2.6: *The Principle of the Petri Net — Transition Dependence: the transition T12 in system A is dependent on system B being in state 9*

Figure 2.7: *The DAQ Sub-Systems*

practically this would result in an unreasonable number of states. In practice, the possible sources of transition failure are minimised to sensible values, with a requirement that the system can perform the necessary transitions from offline to data-taking in a short length of time.

2.4 DAQ System Architecture

To first order, the DAQ system can be broken down as follows:

- Experiment Hardware
- Readout System (transfer of data from experiment to computer — Fastbus)
- Computer System
- System Control/Management
- Data Processing
- Data Monitoring
- Data Output
- Control and Monitoring of the Experiment Hardware (Slow Control)

The relationship between these is illustrated in figure 2.7. They can be split very simply into two groupings: the DAQ system infrastructure, comprising the detector, the Fastbus system and the Computer system, which have been covered briefly in Chapter 1, and the software of the system. The DAQ software is the subject of the following sections.

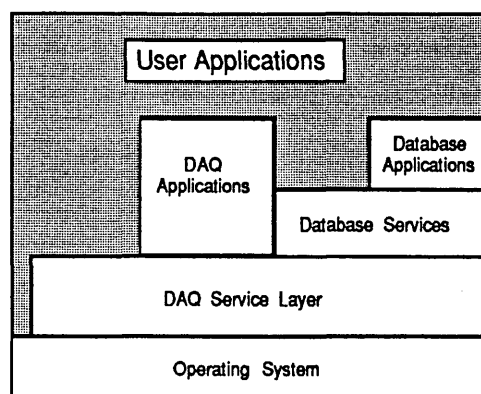


Figure 2.8: *The DAQ Software Layers*

2.5 DAQ Software Architecture

The aims of the DAQ software can be split up into a number of largely independent functions. These are carried out asynchronously and should communicate only via well-defined protocols. They are described as *tasks*.

The design of the DAQ software, in general, follows a layered approach (fig. 2.8). At the lowest level, since the operating systems are fixed, the system functions provided by these can be exploited without the need to ensure portability. Above the operating system, a layer of DAQ services was provided such that for most DAQ-related operations, the operating system was hidden (this provides some software compatibility between the two DAQ operating environments — OS9 and VMS — which therefore requires less debugging, and produces consistent code, which is easier to maintain). Above this, we have the application layer, which consists of complete programs satisfying one part of the system functionality. A large part of the analysis and design effort was to specify precisely the DAQ service layer, i.e. the interface, scope and functionality of modules. Naturally, any constraints imposed by the DAQ hardware must be taken into account by the DAQ software design.

Analysis of the phases involved in setting up the detector for data-taking, the checks that must be made, calibration requirements, etc. showed that the system could be sequenced as a finite state machine. The description (as of 1st October 1990) of the system states is as shown in figure 2.5.

The major part of the software was provided as a service layer for use by users or more specific DAQ applications. The following sections describe some of the major software components and their roles.

2.5.1 Management, Control and Sequencing

Managing a complex system such as the ALEPH DAQ system requires the storage of a great deal of information. Future detector configurations requiring the minimum of software modifications must be foreseen. For this reason a number of database structures were designed for implementing the DAQ service layer. Specifically, these were:

- The Fastbus Database
- The Readout Database
- The Resource Database
- The Partition Database
- The Slow Control Database
- The Run Control Database
- The Scheduler Database

In addition specialised databases were used by the various sub-detector groups for their particular applications (e.g. by the Trigger group for keeping trigger type masks and options). In order to ensure a coherent system, with no duplication of information, the content of the databases and the way in which they interrelate was modelled.

The Fastbus database stored information on each Fastbus module used by the DAQ. The ability to locate modules by name, serial number, Fastbus address, crate position, etc. was provided by the Fastbus Database Library — an access layer for the database [6]. It also provided the data needed to program the segment interconnect modules with the routing information, and a user interface to modify the contents of the database.

The Readout, Resource and Partition Databases are dealt with in detail in the following chapters.

The Slow Control Database stored information on required levels of current, voltage, gas pressure and allowed tolerances, etc. for the detector “life-support systems”, and alarm destinations, etc. for communication of status data. Facilities to modify the information were again provided, plus a server for communication with the microprocessors performing the monitoring and a display for viewing the Slow Control status.

The Run Control Database contains the information specifying which task was required for which detector activity (these being Setup, Initialisation, Calibration, Data-taking, etc.). Additionally, the definition of the system states and the conditions for moving between

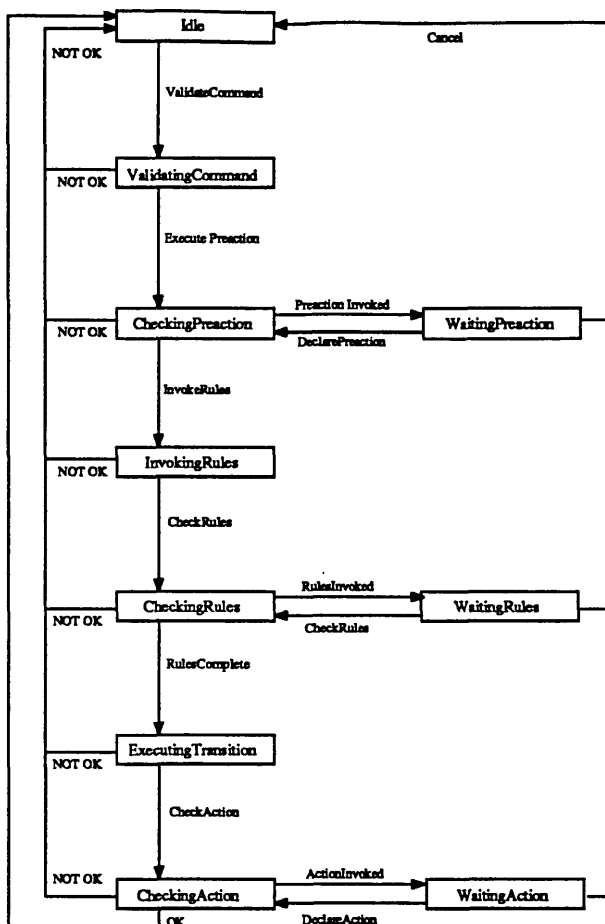


Figure 2.9: *The FSM Package Logic*

them was built into the Run Control. The sequencing of the experiment was performed automatically following the rules defined by the FSM logic, summarised earlier.

The Scheduler database contains data specific to each task — where the program resides, its required inputs, outputs and such things as privileges, priorities and memory requirements.

2.5.2 The FSM Library

The approach followed in the design of the package was to provide a set of standard routines for keeping track of the allowed states and transitions and for implementing the finite state machine logic when changes in state are required. The library itself was implemented as a finite state machine described by the diagram in fig. 2.9.

One of the results of using the FSM protocol, is that system control is performed from one central point, and can be used at all levels in the readout. The FSM library (part of the DAQ service layer) was implemented on both the Vaxes and the micros, allowing the task architecture to be extended to all levels, as shown in figure 2.10. Since the definitions of the transitions, conditions and dependent tasks were held in a database, it was trivial

be broken and remade at any time, from any terminal, by any user. This functionality is particularly useful for periods of work where the user is away from the control room.

The Partition Control Process

Partitions play a major role in the ALEPH DAQ system. They form the environment within which data-taking activities occur. The major part of the software to handle partitions is the Partition Manager. This library and database are described in detail in Chapter 4. The Partition Control Process (**PCP**) is the stand-alone program that initiates the creation of any tasks needed for a partition, including, for example, calibration and run controller tasks.

Ashell

The ALEPH shell program simply provides the initial connection between the Server program (which knows nothing of partitions, etc.) and the Partition Control Process needed to start an activity. It interrogates the Partition Database for a list of partitions, and, when requested, launches the PCP for the requested partition.

The Scheduler

The DAQ software functions are split into tasks. Each task is defined in the Scheduler database, and can be referred to simply by name. Any user needing to start a task makes a request to the Scheduler via a function call to the Scheduler library. The Scheduler starts the task running on the dedicated DAQ **batch** queue. Using batch queues enables the Scheduler and users to employ all the facilities of the VMS Queue Manager, including convenient cluster-wide job control, automatic output logging and queue information display.

The Run Controller

The Run Controller is the control task within a partition. It controls all the tasks needed to perform any kind of data-taking. This would usually be LEP-generated events, but could be calibration, testing, etc. The Run Controller forms the top task of the tree of dependent tasks used for the readout. It performs all the sequencing according to the rules defined by the protocol specified on the run control database. Any errors are reported to it and it therefore provides facilities to handle or report errors.

Data Readout

The arrangement of readout tasks forms a tree structure, based on the task dependencies described by the Finite State Machine model. At each level below the Vax — the Run Controller — each microprocessor runs a manager task that itself controls the sub-tasks within the module. Which sub-tasks are required for the current activity is described by a Readout Configuration Table (RCT) [31], which is constructed from information held in the Partition, Readout and Fastbus databases (see page 66). Standard Control, Producer and Consumer tasks, plus the software needed to access Fastbus from an Event Builder are part of the DAQ service layer, referred to as the Readout Library.

On both Vaxes and Event Builders, the data is buffered whilst whole events or sub-events are accumulated. The software to manage this data is known as the Buffer Manager and is also provided in the DAQ service layer [24, 23].

Messages

At a very early stage in the DAQ development it was clear that a message passing system was necessary. A package providing communication facilities between the Vaxes and the microprocessor systems in the readout is available. It allows simple messages to be exchanged between any task on one machine with any task on another. This is used mainly by the menu package for transmitting the necessary layout information to remote tasks, but is also used for the downloading of readout configurations (§ 2.5.3) and state transition requests/replies.

Monitoring

Monitoring software is provided for the application programmer, in the form of a standard interface to the data through a code template and access routines. This allows access to the data stored in the buffer on the Vax. The user then supplies the code necessary to perform the job required, which can include offline code if suitable.

Histogramming

All histograms are defined in a central database and can be filled from the micros, or the Vax, by calling standard routines. The data is passed as a message to a server on the main computer which stores the information in a histogram buffer. Most are overwritten periodically, in order to economise on space.

Error handling

A set of functions providing access to VDUs displaying informational, warning, error and alarm messages is provided to enable the central coordination of error flagging and logging.

Slow Control

The Slow Control infrastructure is based on Utnet. This is a slower, but cheaper network than Ethernet. The interface to the Vax is via gateway connections. A large number of devices can be supported and periodically polled to check their correct functionality. If some measured value exceeds the programmed tolerance, the device interrupts the Vax server, sending an informational message, and this would be forwarded to the task (i.e. process) responsible for that part of the apparatus. If necessary, the task can take appropriate action, perhaps even halting data-taking. All the devices, responsible tasks and nominal values and tolerances are stored on a database, hence a reliable database system is critical. The system is described in [32].

The safety system is independent of the Slow Control and provides special warnings and alarms for gas, water, helium level and temperature.

LEP Communication

Information concerning the LEP machine is very useful to monitor online performance. For this, data on beam luminosity, machine status, etc. is kept locally in a database and updated via a network connection to LEP. This is described in [33]. The database can be accessed by any user.

Chapter 3

The DAQ Databases

3.1 What the Databases Describe

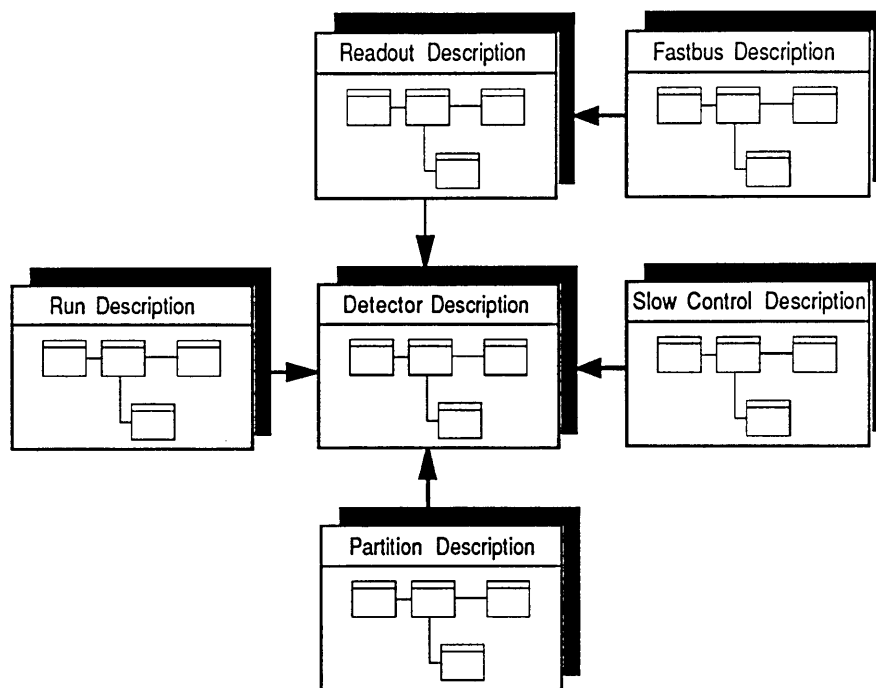
The idea of using a database to describe a system is to avoid writing programs that make assumptions about that system, which would result in an inflexible solution to the problem. With a program that retrieves all necessary information from a database, any changes in the system can be made purely by database updates or modifications. The attractions of this are threefold:

1. Re-configuration of the system is easy. If, for example, one detector must be left out of the readout, this can be accomplished by changing one database entry.
2. Security. Modelling the system by a database, which can be checked for correctness, gives confidence that the system will not do anything unexpected. This is like using a Monte Carlo to make predictions about the detector performance — it is important to model the physical processes accurately, such that the results of the simulation can be used to find problems or differences in the data.
3. With a central database the problems of data duplication, and the accompanying consistency problems when changes are made, are avoided.

This approach is termed **data-driven**.

The aim is thus to describe both the detector and the readout system in a database. This then makes the task of coping with changes in the DAQ structure, or even additions, relatively straightforward. There is an in-built assumption that it should be possible to model any experiment and thus the scheme is, in principle, portable to other experiments.

At the time of design of the online databases, the detector hardware was already described in a database — the ALEPH ADBS offline database — which contained information on

Figure 3.1: *System Database Context*

geometry and material composition. It also held data concerning detector channels, such as dead wires, noisy circuits, calibration offsets, and run-dependent constants, including nominal beam position, run luminosity and number of triggers. This database was used by the simulation program (**GALEPH**), the reconstruction program (**JULIA**) and the analysis program (**ALPHA**).

The organisation of the data, however, is specific to each sub-detector. The Detector Description was therefore designed as an abstract picture of the detector, providing access to information about any part of the detector in a uniform way, regardless of sub-detector type or position in hierarchy. Navigating (following relationships from entity A to entity B) from the general description to the Offline one was accommodated.

3.2 The Detector Description

Shown in fig. 3.1 is a pictorial overview of the major databases used by the DAQ system. It illustrates how it is possible to relate information from one part to data in another by navigating via the Detector Description. Relationships between detector components and Slow Control or Readout devices had to be maintained, and the database contained this information. For example, to determine the Fastbus processors assigned to read out TPC sector 14, one finds the entry for `TPC_SIDEA_E14`, follows the pointer to the ROC table in the Readout database, extracts the Fastbus names from that table and follows the link into the Fastbus database.

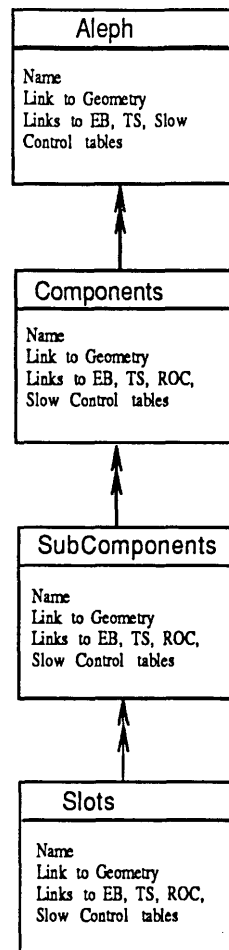


Figure 3.2: *Detector Description ER Diagram*

The data held in the database are shown very simply by an ER diagram (figs. 3.2 and 3.3). The lines joining the entities represent relationships between the data within the entities. A single-headed arrow represents a one-to-one relationship. Several single-headed arrows from one entity show a one-to-many relationship. A double-headed arrow is a many-to-one relationship. This is the way one describes, for example, which Fastbus modules (many) belong in which crate (one) and which coordinates in the TPC are used by a track. A relationship can also be from X to one of A , B or C (e.g. an Event Builder belongs to either ALEPH, a sub-detector or a sub-component). These are described by double lines into a circle with arrows leaving (e.g. figure 3.3). A bar at one end or other of an arrow indicates that the relationship may be null, otherwise there must always be a valid entity at each end. For a more in-depth explanation, the reader is referred to [12]. Connections between data items that are not described by such means are possible — by means of programmed algorithms — but then cannot be checked for consistency using the data model. How the diagrams metamorphose into code is explained in § 3.5.2.

The nomenclature for the hierarchy of the detector was borrowed from the ADBS database. It has since developed into an abstract picture of the detector, which is more a reflection of the readout structure than the hardware construction. It is no longer necessary for

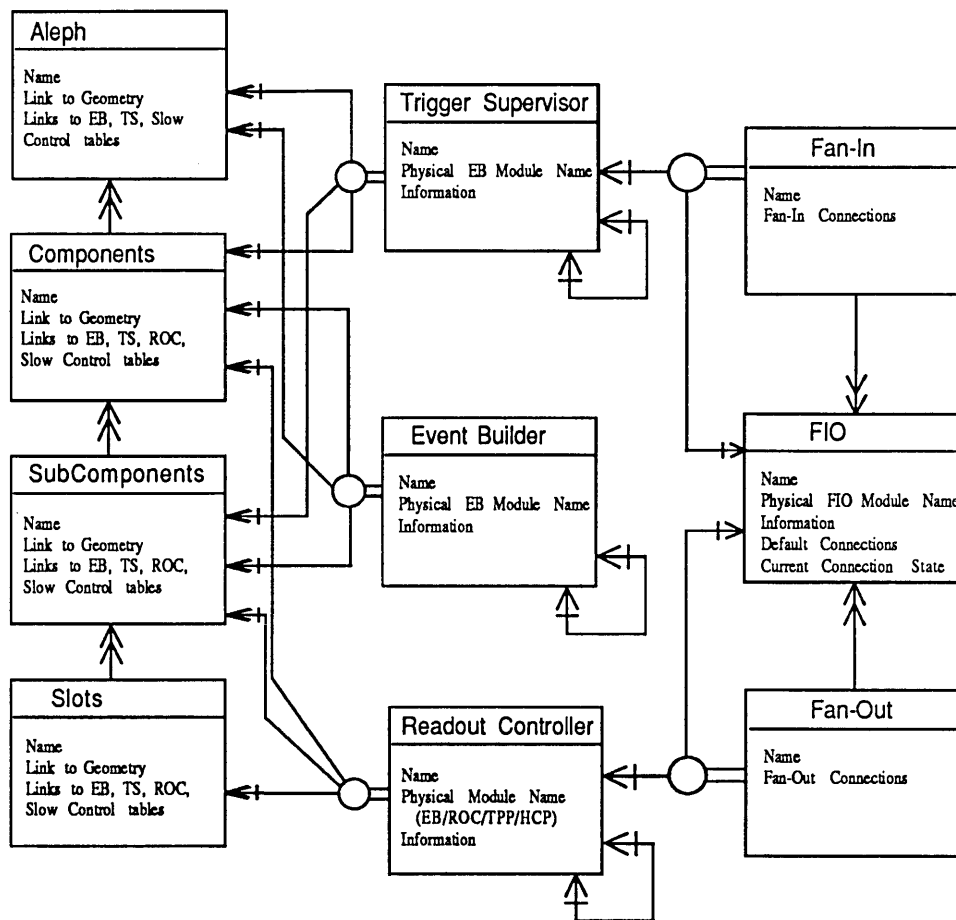


Figure 3.3: *The ALEPH Readout ERD and its relationship with the Detector Description*

a **component** to be a sub-detector, but instead it may comprise a number of smaller sub-detectors, grouped together to optimise readout performance.

3.3 The Readout Database

The idea of the Readout database is to provide the link between the partition (detector) and the physical Fastbus modules used for data acquisition. To do this, the Readout database must describe the hierarchy of the readout system, and the different functionality of the modules involved, independent of their hardware implementation. The overall picture of this is shown in fig. 3.1.

Figure 3.3 illustrates the hierarchial nature of the detector description, and how the readout sub-system is organised in parallel levels.

Ideally, one would define a readout function for every module in the Fastbus system, but this would entail an undesirable increase in the complexity and size of the database, which would affect the update performance (§ 3.5.5). A natural cut-off point is the ROC

level, below which neither DAQ specifications, nor DAQ protocol requirements [19] were provided. This has resulted in very different readout implementations at this level for each sub-detector, due to the varying physical construction and performance constraints of the individual sub-detectors. Fine details such as ADC or TDC readout, refresh cycles, calibration channels, etc. are dealt with below the ROC level.

The readout section specifies the logical function of each object in the readout chain; here it is important to note that this may be performed by *any* physical module. For example, if a Fastbus module is changed, the procedure will run as before, but the readout database is updated to point to a different physical device. The result is the new device performing the task of the one it replaced. This is all transparent to the Run Operator. It is also possible to skip one or more levels in the readout tree, by operator choice. This provides a facility for avoiding faulty modules if a replacement is not available.

Lastly, the readout database maintains the list of resources belonging to the readout, and their resource names. These are the labels used to control access to resources in the DAQ system. They are formed as a combination of the name of the detector and their logical function, e.g. "TPC_EB". The names, and their construction, are left until the relevant chapter, § 4.3.1.

3.3.1 Access to the Information

The information held in the database is needed by various parts of the DAQ system:

- Run Control
- Main Event Producer (on the Vax)
- Producers/Consumers on Event Builders
- Monitoring Programs
- Diagnostic Programs
- The Database Editor

Normally, access is via a subroutine library. This provides a mechanism for data-hiding, making the high-level user unaware of the format of the underlying data structure. This provides the possibility of upgrading the database without having to change higher level code, since the subroutine calls remain identical. Access to any additional features can be supported simply by creating a new subroutine.

The access routines can be split into five types, shown below.

- Initialisation
- Low-level Management Routines
- General Access Routines
- Editing Functions
- Complex Functions

Initialisation and low-level routines are specific to the implementation method as explained in detail in § 3.5. The general access routines, which include simple functions such as “find EB”, “read ROC”, “delete TS”, “find next ROC”, etc., provide the basic access level to the data. The higher complexity functions like those for editing and specialist tasks performed frequently by the DAQ system, are built on top of these basic routines. They can be considered as just a set of macro instructions, but, since they are compiled into the access package, they are very efficient.

3.4 The Trigger Layout Description

While Fastbus forms the data connection between readout devices, and is described by the Fastbus database, the Trigger paths which provide all the necessary signals for the ROCs are formed by separate physical connections. These paths start at Trigger Supervisors (TSs) and terminate at ROCs. Since the functions concerned are described in the Readout database, it is logical to group the Trigger route information with them, as a set of relationships between these modules and the Fan-In Fan-Out (FIO) modules. The Trigger Supervisor is in fact implemented as a plug-in board on an Event Builder (EB) and it is very useful to know which Event Builder. This was not apparent at the time of the initial design, but was incorporated by adding a relationship to connect a TS to an Event Builder.

The overall trigger data description is shown below (fig.3.4).

3.4.1 The FIO Box

The FIO module is a hardware device designed to route a group of bidirectional ECL level signals. A group of signals may combine several inputs and several outputs. The module can route the output signals from the Local or Main port (fig. 3.5) to one or more of eight destination ports (A – H). Input signals from the same set of eight ports are combined using either a logical AND or logical OR operation, and routed back to the same Local or Main port. An example output signal would be GBX (Gated Bunch crossing) whilst

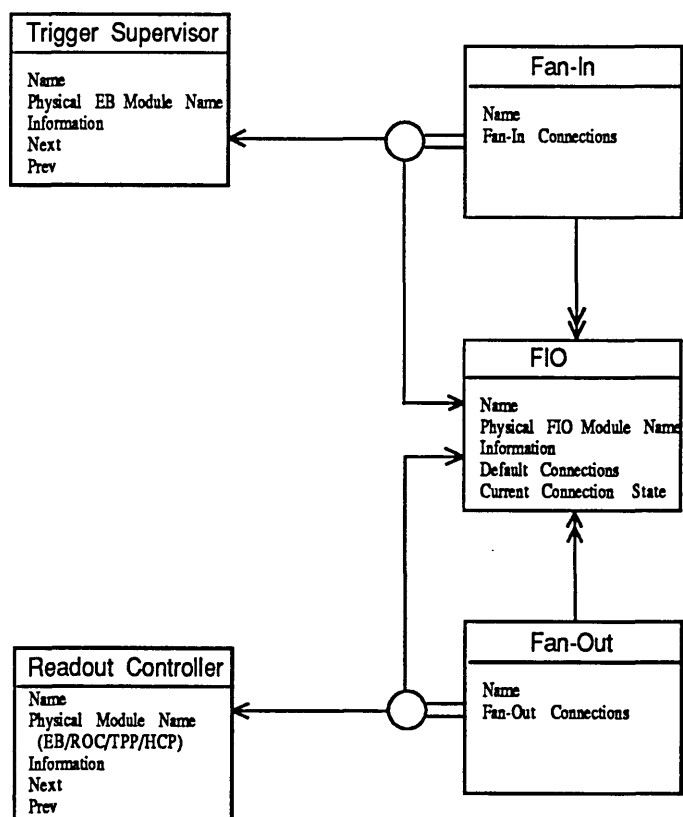


Figure 3.4: *The ALEPH Trigger System Description as an ER Diagram*

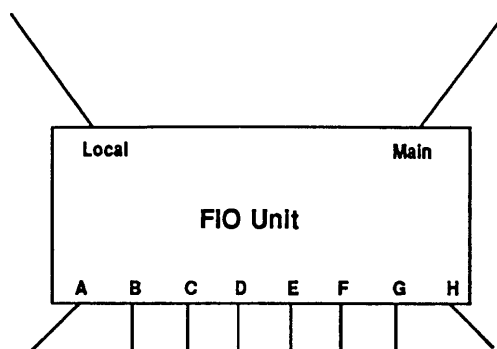


Figure 3.5: *The FIO Module (schematic)*

one input signal is the ROC BUSY line (all ORed together). Many combinations are permitted, and the route-finding software has to model this, as well as overcome some hardware limitations. The two “top” ports, Local and Main, are electrically identical but by convention the Main port is used for signals originating from the main ALEPH trigger system. The Local port is used for sub-detector generated or simulated triggers.

It is not possible to read the state of these connections back from the module due to the design [34], and so it is essential to store the programmed state of each module in the database. The state of the connections can then be made available to the diagnostic tools available to examine the status of the FIOs whenever a trigger protocol or hardware error occurs. Knowledge of the connections can be used to calculate the source of the problem.

3.4.2 Route-finding and FIO Programming

Once a partition is defined, and the hardware modules that will be read out determined, it is necessary to connect all the lowest level modules (ROCs) to the trigger signals provided by the Trigger Supervisor of that partition. Reference to the picture of the Trigger system, fig. 1.3, is useful to follow the description of the procedure.

The basic task is to find an unused connection from each ROC to the Trigger Supervisor, via the FIO units. Each FIO unit used must be re-programmed in order to make any new electrical connection, whilst maintaining existing connections. If no unused connection can be found, then data-taking cannot start. The FIOs are connected in the form of a tree, with the Trigger Supervisors as the trunk and ROCs as the leaves.

There are a number of restrictions built into the Trigger system or FIO which make the route-finding algorithm somewhat simpler:

1. Any sub-detector trigger supervisor is connected to the Local port of an FIO.

2. The maximum depth from a local TS to a ROC or FIO via a Local port is two. Any deeper connections must use the Main ports. This limits the required "look-ahead" in the algorithm to two.
3. They are not reversible devices: the trigger signals come into the Local or Main ports, and are distributed to the ports labelled A-H (see fig.3.5).

The algorithm proceeds as follows:

For each ROC, the link to the next FIO is found. A check is then made on the Local port of this FIO to see if the TS is connected there. If not, it tries to go up one more level in the tree, and check that FIO. If either its Local or Main port have the TS connected, the requisite FIO connections are stored, and the process repeated for the next ROC. Otherwise, the link from the current FIO Main port is followed. If this is another FIO, the search process is repeated. If this is a TS, then, provided it is the correct one, the search has succeeded, the FIO connections are recorded, and the search is started for the next ROC in the list. If the TS is not the correct one, or the FIO port is not connected to anything, an error is signalled and the search stops.

Once the set of connections for each FIO is obtained, it is necessary to determine which other ports are in use by other partitions (these will be re-programmed but with the same parameters) and which ports remain unused. In these situations various options exist for the default state of the port. They are recorded for each FIO in the database:

1. Leave port in current state.
2. Disconnect port i.e. do not pass output signals; ignore input signals.
3. Spy on the current partition being connected.
4. Spy on the signals coming from the Local port.
5. Spy on the signals coming from the Main port.
6. Spy on the signals coming from the port which is not being connected (known as "other").

Spying is a state whereby the signals from the Trigger Supervisor are passed on to the modules further down the trigger tree, but the replies (if any) are ignored. This is extremely useful for testing and debugging hardware without having to fully implement the protocol or hold up data-taking. The latter option is provided for the TPC. Its readout controllers (TPPs) require a special clock signal that is used to gate the GBX signal into the TPP to reduce timing jitter. This is routed through the FIOs. The TPPs always have to be connected either to this or to a dummy clock. The dummy clock was provided at

the Local input ports of the TPC main FIOs, whilst the true clock signals came into the Main ports.

Other FIO Programming Constraints

One unexpected hardware restriction, due to the design of the FIOs, means that under certain circumstances an FIO freezes¹ if it receives a spurious signal. Unfortunately, whilst being programmed, the FIOs tend to generate noise on their output lines, which had the effect of freezing any FIOs connected further up the tree. One solution to this was to program the FIOs in a particular sequence (as they can be unfrozen after programming them); however, it turned out to be far simpler and reliable to program them all and then unfreeze them *en masse* afterwards.

It is not possible to program the FIO units without stopping them passing signals. This means that any partition that makes use of an FIO already carrying trigger signals from another would interfere with its data-taking (i.e. stop triggers arriving at the ROCs). To date, due to the arrangement of the FIOs and TSs, this has not proved to be a limitation. A mechanism is provided (using the Resource Manager, below) to enable the other partition to be informed. It can then halt its own trigger until the re-programming was complete.

The last complication is related to the computer system as a whole: since a Vax cluster was used, and any node could, in principle, re-program the FIOs to start a run, it is necessary to keep track of which FIO routes are used in a manner such that the information is available cluster-wide. This is achieved by booking each FIO port in a route by name, using the Resource Manager (described in Chapter 4). Any failure to book a port means that it is already in use by some partition on one of the nodes in the cluster. If the connection is essential, this means that the new partition cannot start; if not, the information can be used to determine how to re-program the FIO to preserve the existing connection.

3.5 Implementation Architecture

The DAQ is an online, real-time system. Since heavy use is made of the databases, overall performance is highly dependent on speed of database access. The implementation method chosen for the database system was therefore largely optimised for efficient access. There is a trade-off between complete flexibility, as provided by commercial database packages such as Oracle and the speed provided by a completely hard-wired approach. As outlined

¹The FIO is designed to halt activity and maintain its inputs and outputs at the current level on receipt of a HOLD signal. This is very useful for problem-finding.

before the virtues of a flexible solution are many, but at the same time, the overhead for simple read or write access to a relational database system was too large. A principle use of the database would be repeated queries for information, often fairly complex, but nevertheless identical in search requirements. The time overhead in this case, compared to the built-in access function approach, is very large.

Oracle does, however, have some features that proved indispensable. It provides facilities for the manipulation of the database structure, such as adding new tables, attributes or relationships, whilst preserving existing data. Any fixed-size tabular database (which was our chosen implementation) cannot be re-structured (say by adding another attribute to a table) easily. The solution was to maintain Oracle compatibility, via a set of tools, such that its features could be used when desired, but the run-time overhead was avoided. Any of the Online databases can be copied to Oracle; modified, backed-up or simply queried, and then copied back.

3.5.1 The Database to the Programmer

The programmer sees the tables as Vax FORTRAN structures (table 3.1). Although not standard FORTRAN 77, the benefits of using higher-level data structures were felt more important than pure standardisation. These included more readable code, easy data abstraction and no memory manager overhead — such as that from a system like BOS or ZEBRA ², used in the Offline programs. Compatibility was maintained by using the same data format for the structures as for the BOS banks, which meant data could be transferred simply by means of a memory copy.

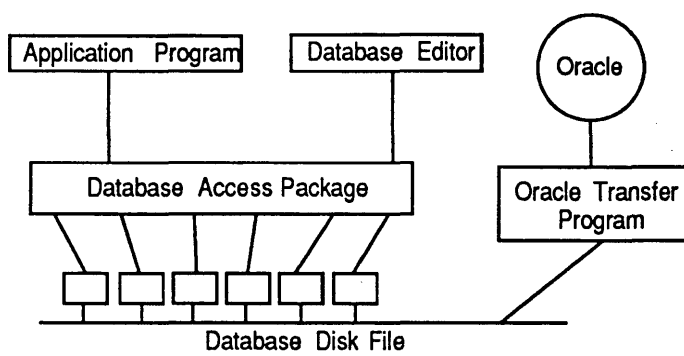
3.5.2 Database Tools and Code Generation

Starting from the Entity-Relationship diagrams, the database descriptions are translated manually into DDL, the ADAMO ³ data definition language. The collection of DDL is then compiled, using the ADAMO CSP (Conceptual Schema Processor) tool, into a data dictionary. During this process, the consistency of the data definition is checked against the knowledge stored in the data dictionary (known as the ADD). The ADD file is in a special format not trivially accessible from a standard FORTRAN program. Facilities exist within the ADAMO system to allow the programmer to check any data in a table against the constraints defined within the ADD. While very useful, this feature can be emulated by checking data on input, though this is clearly less elegant than a more automatic approach. Again, the resultant implementation was the result of a choice between flexibility and efficiency.

²Memory management systems for FORTRAN 77. See Glossary for further information.

³For all information on the ADAMO toolset, the reader is referred to [12].

ADAMO	ORACLE	Vax FORTRAN	FORTRAN 77 + BOS
Entity Set	Table	Array of Records	Bank
Attribute or Relationship	Column	Field in a Structure	Word in bank
Instance of an entity	Row	One Record	First (column-size) words

Table 3.1: *Data Representation Terminology: Equivalents*Figure 3.6: *General Database Organisation*

From the ADD file, FORTRAN include files are generated, using a special ADAMO tool VLB, or Vax Language Binder ⁴. These include files define the Vax FORTRAN structures used to represent the data (see Table 3.1 for the translation between different data representations). In addition to these standard ones, tools to list the database contents and monitor usage were available. The necessary programs to generate Oracle tables from the FORTRAN structure specifications in the include files and exchange data between the database files and Oracle were also written in-house [35].

3.5.3 Access to Data: Data-Hiding

A layer of interface functions, as outlined in § 3.3.1, was provided for each database. Restricting access to the database via these routines had the following advantages:

- The data is hidden from the user. Changes in format, size, notation, ordering, etc. are all transparent, thus user code remains independent of such changes.
- Any potential problems related to data integrity or consistency can be controlled by including extra checks in these routines.

⁴This is an adaptation of the original ADAMO tool, FLB, which is intended for FORTRAN 77 programs

- Database updating protocols can be hidden inside the routines.
- Accesses to data can be transparently logged or time-stamped, etc.

There are, of course, disadvantages: for example, not every operation that can be performed, has an associated service routine. In these cases, the database **primitives** must be used, which can be somewhat inefficient. This overhead is minimised by coding higher-level routines for any frequently required series of requests and providing them as service routines in addition. This is standard database practice (see [36] for some discussion of the matter). The architecture of this scheme is illustrated in fig.3.6.

As a simple extension to this scheme, all the databases in the DAQ system had dedicated editors, with read, modify, delete and query capabilities. They also used a consistent menu-driven operator interface to reduce the effort required to learn how to use them.

3.5.4 Implementation of Data Sharing and Permanent Storage

Data held within a FORTRAN structure is memory resident. On a Vax computer, it is a simple matter to enable this data to be “seen” by more than one process: i.e. to make the data **global**. This was done by declaring the memory used for the database as a **system-wide global section**. This is a facility provided by VMS which allows the virtual memory of any process to be mapped on to the same pages as those of another process. The mechanism is explained in the Vax/VMS manual [37]. Any update is immediately seen by all processes accessing this memory. For DAQ purposes, it was also desirable to allow any process (providing certain criteria were satisfied) to update the data. This means that the system is not reliant on one particular computer or task functioning correctly. Once this is the case, some inter-process control of access to the information is required, to avoid conflicting modifications being made. This is discussed in the section below. The special option files needed by the VMS linker to enforce the format of the structures and make them contiguous in memory (necessary for global sections) were generated automatically by the Oracle-FORTRAN exchange tools [35].

To provide permanent storage for the data, it must be copied to a disk file. This can also be saved for backup or archive purposes. VMS provides the necessary system calls to enable the global section to be attached to a disk file in such a way that the system can be requested to update the file from memory (or vice versa) automatically ⁵. The last problem to surmount was due to the nature of the DAQ system computers: rather than one large computer, we have several smaller Vaxes in a cluster. For the same reasons that

⁵The mechanism is in fact identical to that used for virtual memory. Virtual memory pages are assigned to blocks on the disk file. Whenever the page is swapped out of memory (this can be done either by the system or by a system call from the user) it is written to the disk file.

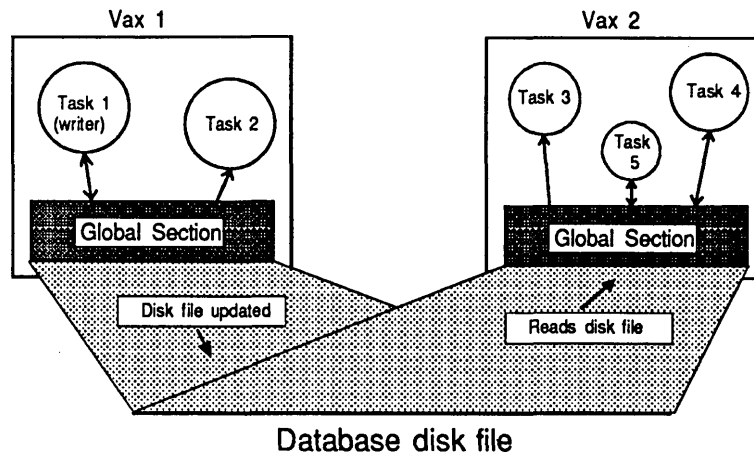


Figure 3.7: *Cluster-wide database updating*

restricting database updates to one process was felt undesirable, limiting access to one machine was also to be avoided. The solution to this was the cluster-wide database.

3.5.5 The Cluster-wide Database

In order to maintain data consistency between machines, a method for communicating changes between nodes was needed. The Vax operating system has a set of system level routines known as the Distributed Lock Manager, or DLM, (see [38] for a full discussion of the mechanism used) which provide low-level cluster-wide communication for purposes such as this. For example, file access from VMS is managed using this tool.

The important features of the DLM were exploited predominantly by the Resource Manager, described in Chapter 4. Here, a brief introduction is given in order to describe the update protocol. Changes made to a database were done in an interlocked fashion, using a common package to perform the access control. An illustration of the process is shown in fig.3.7. A subset of access levels, termed **lock modes** in [39], were used for the database update protocol. These represent the level of access to data requested by a user, and were as follows:

- Read only access

Here the user is interested only in a copy of the database at any one time. Updates are not propagated to a process using this mode. Read only mode is compatible with all those specified below.

- Write access

This mode is requested when a process wishes to make a change in the database.

Provided the request is granted, the process can perform any modifications. It then requests a return to a lower access level (usually Read), which initiates the update of any process copies which have Read access. This mode can only be held by one process at a time. It is incompatible with Protected Read.

- Read access

This is the standard access mode. Any changes in the database are propagated to the user's copy via an interrupt (which forces a re-mapping of the disk file). Read mode is compatible with all the other modes.

- Protected read access

This mode gives special protection if it is vital to guarantee that no changes can occur in the database for however long the mode is held by a process. It is compatible with all other read modes, but not with Write mode.

- No access

Used to indicate that the process does not need database information at present, but will access it later. Useful for monitoring usage, and improves access speed.

This implementation produces no overhead for read access, which was the primary criterion, as no check need be made to see if data has changed. When an update is made, any process accessing the database receives an interrupt which may take up to one second to complete (this is a function of file size, hence the desire to minimise the size of a database). The modifying process must wait for this to finish before continuing.

Additional Run-time Checks

Since the Readout database is linked very closely to the running of the DAQ system, there were extra checks on modifications to the database during a run. For example, the Readout database is used to locate all the Fastbus modules to be read out. This should not be changed while a run is in progress. To achieve this, the Resource Manager was used by the Readout Database access routines to book and release any readout object being updated. This is explained in the section on Partition Management (§ 4.5.4).

3.5.6 Implementation of Access Library

The benefits (from a software reliability and maintenance point of view) of an access library layer on top of the database have been explained. In addition, it is then possible to make use of a feature of VMS to provide the following:

- A pre-linked library that has all internal references resolved and can be shared (one file) between any number of different programs. This results in a much reduced

program link time, and a much smaller size executable file, as all the routines are already contained in the shareable library.

- The ability to introduce new versions of the software (for bug-fixes, addition of new features or testing) without the need for re-linking. A simple command allowed a user to switch back and forth between versions. This was done using **Transfer Vectors** — see § 4.3.4.
- One place only where software updates need to be made (as every program uses the same copy of any routine).

The only restriction is that the size of the database is fixed when the shareable library is made. Since the databases are cluster-wide, and *must* be the same size, this is in fact a useful side-effect. Due to the above list of advantages, **Shareable Image Libraries** were used for all the packages described in Chapters 3 and 4 of this thesis.

3.5.7 Programming Methods Used

Many of the databases consisted of entities which possessed $n \rightarrow 1$ relationships or at least could be seen as groups. A simple way to implement this was the linked list. The name (or some other identifier) of the head of the list is recorded in one table, and then the table in the list has two attributes, known as **Next** and **Previous**, whereby the list can be scanned upwards and downwards. The advantage occurs when an entry must be inserted or deleted. For an insertion, the new table can be placed either as the first entry in the list, as the last, or, if order is critical, only two pointers (one **Next**, one **Previous**) need to be updated in the existing list. Similarly, when an entry is deleted, only the **Next** link of the previous entry, and the **Previous** link of the next entry, need to be changed (to point to each other). Provided the number of entries remains small, the search time for such a system is not too much larger than for a binary tree-type arrangement. The processes are illustrated in fig.3.8.

3.6 User Interface

The standard ALEPH menu package, UPI, was used to build a menu-driven user interface to the database, allowing the user to add new modules, modify existing entries and delete old entries. Fig. 3.9 illustrates how an FIO entry is displayed. The interface was arranged in a manner that permitted only a particular subset of possible operations to the database, in order to simplify the interface (i.e. reduce the number of menus on the screen and the displayed information to a minimum level). For example, to describe the Trigger connections, ROC to FIO connections must be entered at the ROC level, FIO to FIO/TS

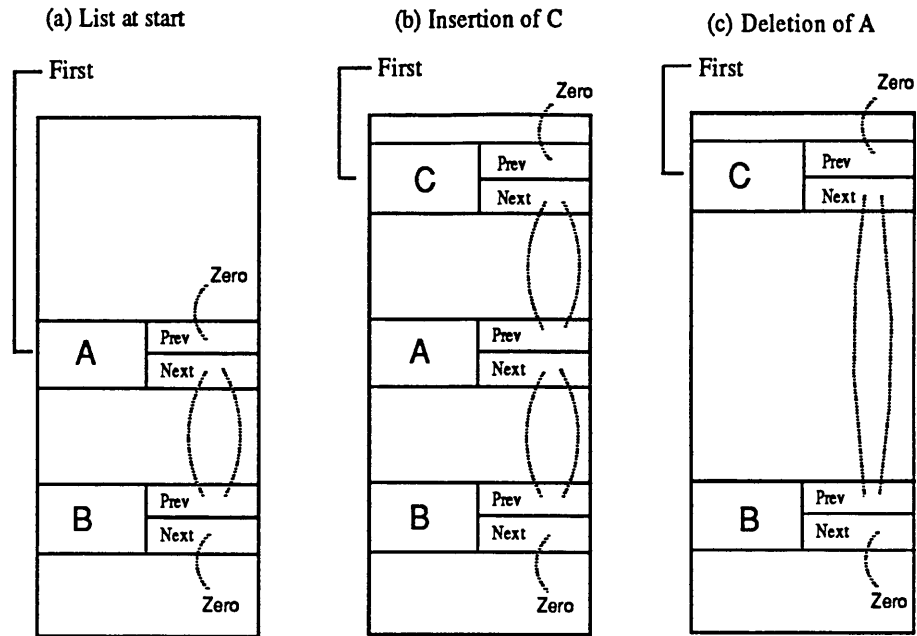


Figure 3.8: *Operations on a Linked List*

connections go only from FIO input ports to an FIO or TS output. This avoids making the user try to update too much information in one go, which would increase the likelihood of making a mistake.

```

----- Readout Database -----
Choose Detector level
COMPONENT: _____
SUBCOMP: _____
SLOT: _____
-----
Choose Resource TYPE: FIO
-----
-< Resource Name >-
-----
+-----+
Add
Edit TPC_SIDEA_FIO1
Renam
Group
Delet
Quit
-----
Editing Resource: _____
Resource: TPC_SIDEA_FIO1
Last changed: 25-APR-1990 19:04:16.95
-< Special Information on this Resource e.g. "Fact" filename >-
TPC local distribution FIO side A
Direction: F000 Logic: 4000A000 Ports: 1F1E
Default Connection mask: 1E Spare flag: 0
Fastbus Device Name: FIO_80_16 and Type: FIO
FB primary address CRATE Port 80000016
===== Trigger: Fanin connection to TS/FIO: (blank => none) =====
Main: FIO/TS Name: _____ FIO OUTPUT Port 0
Local: FIO/TS Name: TPC_FIO FIO OUTPUT Port 1
+-----+ Fan Out status of FIO +-----+
Fout Port: 1 ROC/FIO: TPC_SIDEA_FIO0 Port 1
Fout Port: 2 ROC/FIO: TPC_SIDEA_FIO1 Port 1
Fout Port: 3 ROC/FIO: TPC_SIDEA_FIO2 Port 1
Fout Port: 4 ROC/FIO: TPC_SIDEA_FIO3 Port 1
Fout Port: 5 ROC/FIO: TPC_SIDEA_FIO4 Port 1
Fout Port: 6 ROC/FIO: Port 0
Fout Port: 7 ROC/FIO: Port 0
Fout Port: 8 ROC/FIO: Port 0
+-----+
[ACCEPT] [CANCEL] [RESET]
Linked to:
-----
One match. FIO selected.

```

Figure 3.9: The Readout Database User Interface

Chapter 4

Management Software

Any multi-user system, of which the ALEPH DAQ is an example, possesses a number of inherent conflict problems, for which some method of organising access to any part of the system is required. What follows is covered in more technical detail, but with less explanation, in [40] and [41].

4.1 Requirements

To introduce some terminology: a **resource** is a logical entity that is available to one (or more) users. This may be a hardware device, an area of memory or a value (such as a run number) etc. One example would be the readout controller for one HCAL endcap.

It will usually be desirable to restrict access to a resource in some way — to prevent different users from overwriting the same data, using the same run number or resetting a readout device whilst it is in use.

This can be achieved in a number of different ways. Hardware level access control can be used, but this is inflexible, dedicated to specific hardware devices (and therefore not generally applicable) and, frequently, too costly and complex. This mode of protection is used, typically, in computer disk hardware. An alternative (since the DAQ system is driven from the central computer) is to implement control in all the driver (base-level) access software. However, this may involve unnecessary overhead, as each time the resource is accessed, a check is made. This has the advantage of being secure — anyone using a resource must go via the driver software, and is appropriate in situations such as Slow Control (e.g. of high voltages), where no unauthorised access can be tolerated. This principle was used for controlling database access, as described in the previous chapter.

The method that ALEPH adopted for general use was the least fail-safe option, but provided most flexibility. Each resource is assigned a unique name (maintained in a

database) and whenever a user wishes to access the resource, a call to **book** the resource, using this name, must be made. Once “ownership” is established, the user is free to perform whatever operations are necessary, and must release the resource on completion of the task. Any other user attempting to book this resource in the intervening period will be denied access. No physical protection is involved — the system relies on each user abiding by the protocol. An analogy is a set of traffic lights controlling vehicle movement, where the resource under contention is the area of the road junction. Everything works provided *all* the drivers obey the lights ... The essential advantage of this method is that it can be modified easily, is independent of the type of resource and involves little real-time overhead.

ALEPH also desired the capability for running independent data streams in parallel through the DAQ system. This is known as **partitioning**.

4.2 Partitions

A partition, as an independent subset of the DAQ system (described in § 2.1.2), is somewhat complex. To simplify the idea for the user, a partition is described by just a collection of parts of the detector. This defines the source of the data which will be used offline and is therefore most appropriate for data-taking. It is also simple to visualize how the system as a whole can divide up into sub-detector partitions, or sub-sets thereof, using the description of the detector as a guide. The software then uses this definition to build the necessary environment as follows:

From the partition description one can determine the active elements of the readout, which gives a list of physical devices to be read out. A separate description for each node in the readout tree, specifying the sources and destination of event data, is made and sent to the node (the Run Configuration Table or RCT). The paths for trigger signals are defined by the FIO connections to the terminal nodes, or ROCs, and are programmed. These actions, all derived from the initial partition name, are described later on in § 4.4. Additionally, if any part of the detector requires a particular program for operation, this can be started automatically. An overview of this procedure, the system task architecture and the software and hardware protocols have been presented in [7].

4.3 The ALEPH Resource Manager (ARM)

Following the choice of a general method for managing the resources in the DAQ system, a set of requirements was decided upon. After discussions amongst the Online group, a consensus emerged:

- Performance:

It is important that the booking of resources does not incur too large an overhead.

- Cluster-wide booking:

With many computers and users distributed over the system, it is necessary for resource management to cover them all.

- Hierarchical resources:

This is the ability to group resources such that some collection can be combined and booked at once. This results in less overhead both for the system and for the user. Since the detector is naturally hierarchical, and many of the resources in contention will be related to it, this was deemed vital. Access to the group resource should imply the same level of access to all resources in the group.

- A Naming Convention:

Some convention for the names of booked objects must be enforced ¹.

- Generic Resources:

Often it is possible to allow a limited number of duplicates, or copies, of a resource. These can be accessed by a particular name, and the ARM should allow booking of copies up to a pre-defined maximum. For example, the available Fastbus Broadcast classes for the Level 2 Trigger: *L2BRCL_1* – *L2BRCL_15*. These resources are termed **Limited Generic resources**. Additionally, there are some objects, referred to as **Sequential Generic resources**, which provide an unlimited supply of ever-increasing values, such as run numbers. Provision for these was required.

- Partitioning:

Since partitions were to play a major role in the DAQ running, and it was also foreseen that resources within a partition should be controlled differently to those outside, some features to cope with different partitions were necessary. Some discussion of this follows in § 4.3.2.

- DAQ system-wide:

This was not high on the list of priorities, but it was felt nonetheless that the ability to book resources from Event Builders would be very useful. To date, this has not been implemented, mainly due to lack of time.

4.3.1 Resource Names and a Hierarchy

To reflect the hierarchical nature of the detector and the readout, and to use reasonably mnemonic names, the following scheme was adopted:

¹“The beginning of Wisdom is to call things by their right names.”

— *Old Chinese Proverb*

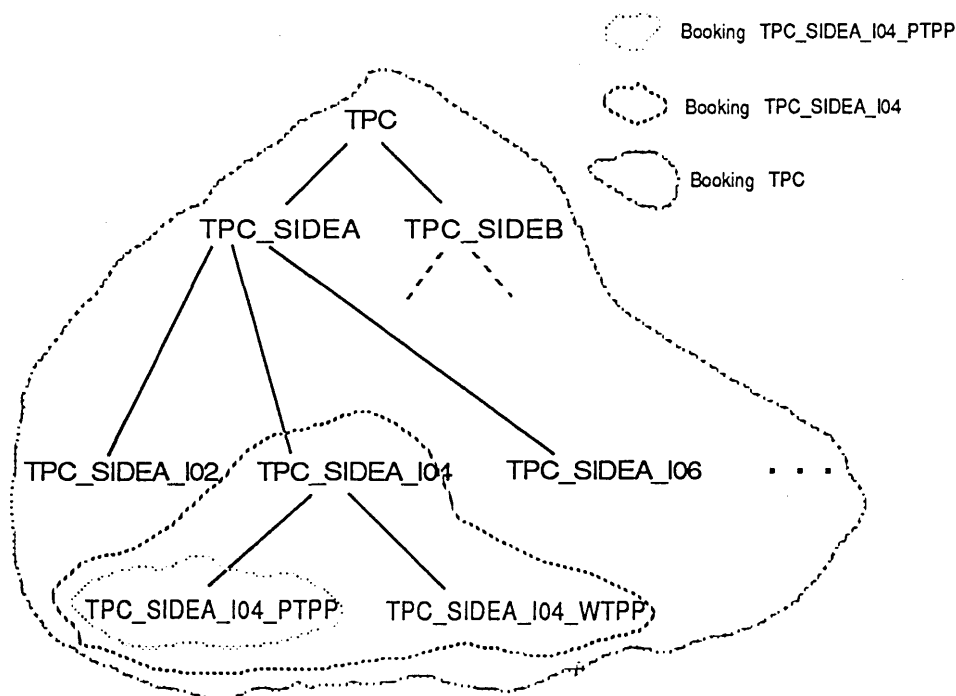


Figure 4.1: *How Resources are Grouped Together*

- The detector names are extended versions of those used offline (for technical reasons the offline ones are restricted to six characters).
- Levels in the hierarchy are reflected by an underscore in the name.
- The function, or generic name, of the resource, is appended to the detector name.

This gives names such as “TPC_EB”, for the TPC Event Builder, “HCAL_ENDB_HCP”, for the HCAL Control Processor on End B and “TPC_SIDEA_I05.PTPP” for the Pad TPP on one of the TPC Inner sectors. An exhaustive list is available in [42]. This allows the group booking facility of the Resource Manager (RM) to be implemented by requiring that by booking “TPC”, all TPC resources below this level are also booked. In common wildcard terminology, this is like booking TPC_* (see fig. 4.1 for example). For illustration purposes, one can imagine making an airline booking, where the seat is given as “VIRGIN_FLIGHT701.ROW5.SEATA”. If one were to charter the whole flight, one could book “VIRGIN_FLIGHT701”, or just one seat row as “VIRGIN_FLIGHT701.ROW5”. The actual names used, as stated in § 3.3, are stored in the Readout database ².

²“The beginning of foolishness is to call things by their wrong names”
[John Shade, October 1988]

— *Vietnamese Corollary*

4.3.2 Partitioned Resources

Partitions are intended to be completely independent. For this reason, there is a simple rule that must be enforced by the Resource Manager: any resource contained within a partition *cannot* be booked by any user that is not part of the partition. As a result of this, we have to add three more requirements to the earlier list:

1. Access to any resources contained within a partition must be refused to any process external to that partition, but granted to those within a partition, providing the usual booking rules are satisfied.
2. It must be possible for processes to join partitions by request.
3. It must be possible for processes to leave partitions by request.

The other requirements were already handled by the implementation.

4.3.3 Design in Brief

The design of the RM centred around the use of the VMS Distributed Lock Manager [38], which provided the basic inter-cluster communication facilities, and the concept of named locks. A lock has a number of properties:

- A name.
- A numeric identifier.
- A lock mode, specifying the level of ownership of the lock.
- A lock value block, which is an area of storage available to any owner of the lock.

For any given name, many locks can be held by different processes, provided they are all compatible. They are primarily intended for file access control, hence the nomenclature. The numeric identifier distinguishes locks of the same name. The relationship between lock modes is explained in [38, 39] but can be summarised as follows:

The modes possible are: Null, Concurrent Read, Concurrent Write, Protected Read, Protected Write and Exclusive. As implied by the names, they provide increasing restrictions on access by other requesters. Ownership of a lock in Exclusive mode implies no access to other requesters. With Protected Read, neither Exclusive or the two write modes are compatible. And so on. The combinations provide sufficient flexibility to implement the resource booking levels. Any number of requests can be queued on a lock, but only those

compatible with the current owner's mode are granted. When a lock is released, the next compatible request in the queue is granted.

The purpose of the ARM is to provide a simple interface to the DLM whilst adding the extra features used by ALEPH.

The Naming Convention is honoured by creating a lock for each node in the resource name (nodes being separated by underscores). Booking "TPC_SIDEA_EB" results in the creation of three locks, "TPC", "TPC_SIDEA" and "TPC_SIDEA_EB". The requester must successfully lock all these for a booking request to succeed. The possible reasons for a request failing are listed in the sub-section below. No comprehension of the name is built in, as that would require knowledge of the external system and reduce the applicability of the ARM. Security is provided by the using only names from the database.

For the Resource Manager, partitions are described by defining the boundaries of the partition, using the naming convention to specify the nodes of the tree. The procedure is illustrated in fig. 4.2

A partition is labelled by a number, known as the **Partition ID**, that is then used for every reference to that partition. This number must be uniquely assigned to a partition. Resources that do not belong to a partition explicitly can be booked by anyone, regardless of their partition state.

Finally, a request can include a timeout value, specifying the maximum time the requester is prepared to wait for a resource if it is currently booked by another process.

Booking: Success or Failure

A booking request will either succeed or fail due to one of the following (in order of checking):

- Partition ID mismatch. The requested resource is contained within a different partition to the requester. This only occurs for resources declared to be within a partition.
- A node higher up in the tree is already booked at an incompatible mode.
- A node lower down in the tree is already booked at an incompatible mode.
- The resource itself is already booked at an incompatible mode.

A facility to determine who currently owns the resource, following a failure, is provided.

4.3.4 Implementation

The implementation of hierarchical levels was achieved by constructing two trees of locks: one, the Primary tree, comprising those nodes actually booked, and the other, the Shadow tree, containing marker locks that indicate the level of booking for all nodes below. This was necessary as, in order to check that booking a node at one level is legal, one must ensure the new request is compatible with any resource already booked lower down the tree. A search through all the currently booked nodes could be done, but this has drawbacks. Timing problems might arise unless the whole booking system is halted during the search, which is undesirable. The procedure would also take a long time to complete if there were many resources. Instead, the RM need only check for compatibility down to the node being booked, as all successfully booked resources leave a lock on the Shadow tree with the same mode as the requested resource.

Two aspects to the question of partitions are:

1. How to define them and
2. How to distinguish resources within a partition from those outside.

Problem (1) is handled by the user providing a list of resources that correspond to the top nodes of any sub-tree in the partition. Since the RM knows only about resource names, this is the only way, in principle, to achieve this. It is a general applicable solution, but for complex partitions, the definition is also complicated. Simplifying this procedure is one of the functions of the Partition Manager (see § 4.4). Problem (2) is solved by creating a special tree, with a **Partition Lock** as the top node. All resources in the partition are renamed (internally) such that they lie below the Partition Lock. To achieve this, use is made of the Lock Value block mentioned above. The partition identifier (ID) is stored there, such that when the RM discovers a non-zero partition ID in the current node's lock value block, it translates the name into a partitioned name. One additional advantage of this is that the whole partition can be booked simply by booking the Partition Lock. The way this is performed is illustrated in fig. 4.2.

The Resource Manager was written in ADA. Some of the reasons for this included ADA's facilities for the handling of data-structures, code readability, type-checking and the interface to the operating system. In particular, contrary to Fortran, as recursion is possible for ADA procedures and functions, solutions to the design specifications were very elegant. For example, to book TPC_SIDEA_I02, requires the creation of a lock for each node on the shadow tree. So to do this the function attempts to book the node up one level first, i.e. TPC_SIDEA, by calling itself. This, in turn, attempts to book TPC, which in turn tries to book "". Here the procedure is at the top level, so it stops recursing and returns. The caller then performs the book node operation, and then returns back to the

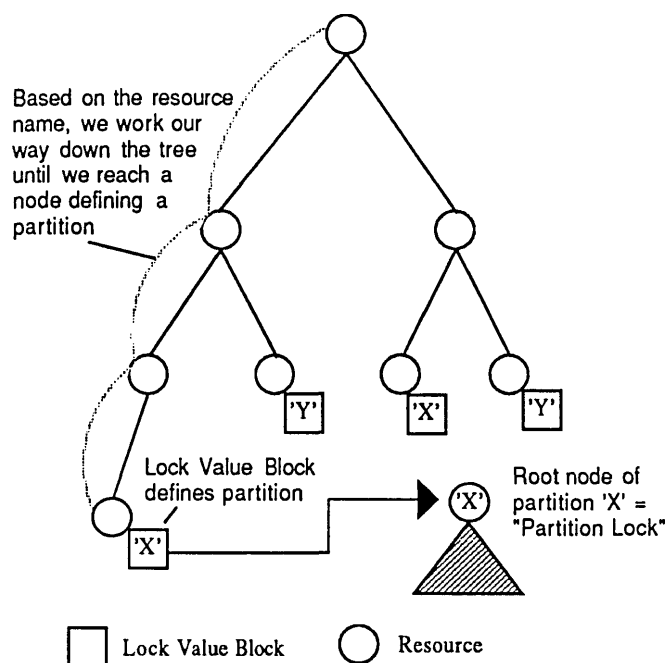


Figure 4.2: *Defining a Partition using Resource Names*

previous level. And so on. In the case of a booking failure, releasing nodes is performed in a similar, but reversed, manner.

Internally, the RM maintains a linked-list of nodes booked, both Shadow and Primary, to aid the process of releasing locks and checking for duplicate requests (which are allowed by the Lock Manager). As ADA allows dynamic memory allocation, this can also be implemented very simply without the need, as in Fortran, for allocating space at compile time.

As with the database packages described earlier (§ 3.2,3.3), the RM was released as a Shareable Image library for reasons of utility. In order to allow version updates without users having to re-link **Transfer Vectors** were used to keep the actual memory addresses of routines independent of the code calling them. This is especially important in a complex system such as the DAQ, where there are many inter-relating packages. Users are not always aware which packages they actually use, as many are referenced indirectly by another higher-level package. The transfer vector concept is illustrated in figure 4.3.

Generic Resources

Contrary to standard resources, for generics, some external knowledge, of the resource in question, is required. To maintain the generality of the package, this is provided by the user when a generic resource is created (which must be done prior to any booking). Thus, for a limited generic resource, the number of possible copies is given. For sequential

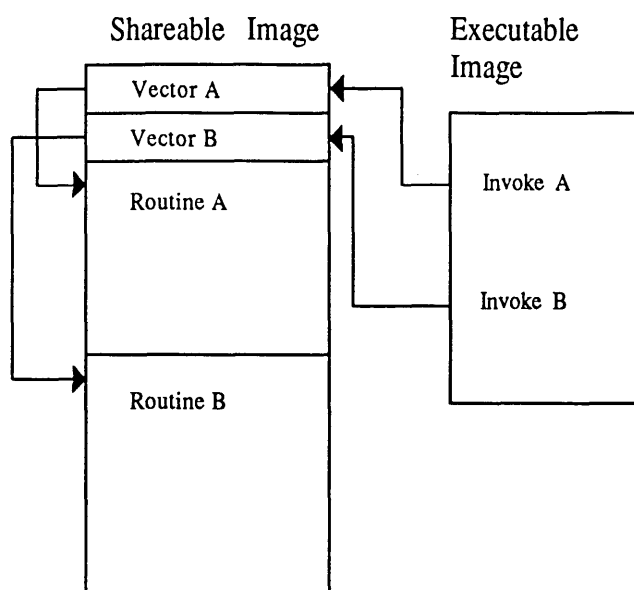


Figure 4.3: *The Concept of Transfer Vectors*

generics, the start value is given. These constraints are stored in a small file, along with the resource name and current value. The file is accessed via a **logical name** or file alias, which allows individual users or groups to maintain their own private Generic Resource database. This is extremely useful for testing purposes.

4.4 Partition Management

The function of the Partition Manager (PM) is to enable the operator of the DAQ system to view a partition as one self-contained object. It must thus hide all the differences between the actual makeup of individual partitions and allow access to and control of the partition purely by a single identifier. As discussed in § 4.3, the PM also performs all the booking necessary to define a partition, thus sealing off a sub-set of the DAQ system from DAQ tasks outside.

Returning to the flight analogy, the PM is similar to the charter firm which can book all flights to New York, independent of airline. They take care of all the details such as differing number of seats or rows, size of aeroplane, flight numbers and the operator (the Ski Holiday Company) only has to worry about the number of passengers for a particular destination.

4.5 The ALEPH Partition Manager

Whilst a Partition is defined in terms of parts of the detector, it is a collective term that includes both the hardware and software entities needed to perform data-taking. The hardware side comprises primarily the following:

- Some non-overlapping sub-divisions of the ALEPH detector (may be none)
- Their associated Readout Controllers (ROCs)
- The Event Builders needed to read out the ROCs
- Zero or one Trigger Supervisor
- A number of FIO modules
- The Trigger Cables between TS, FIOs and ROCs

The software side consists of the programs running on the hardware items mentioned, plus some additions:

- A uniquely assigned Partition Identifier (ID) (to distinguish logical data streams in the Event Buffer)
- A uniquely assigned Fastbus Broadcast Class ³ (to which only ROCs and EBs in that Partition respond)
- The Trigger Routes between TS and ROCs (i.e. the signal connections programmed in the FIOs)
- An optional Level 3 trigger
- One Partition Control Process (PCP)
- The set of cooperating Vax processes that perform the required DAQ functions for the partition (data-taking, calibration, tests ...)

The level of partition “involvement” varies. Event Builders, as stated earlier, are multi-user devices, and the *hardware* can thus be shared by partitions. However, the task performing the event-building, running in the micro, is exclusively assigned to one partition.

³Fastbus Broadcasts can be restricted to certain segments or modules by using a Class other than zero.

4.5.1 Requirements

The Partition Manager has a number of roles to perform which arise from the requirements of Chapter 2. These are the following:

- The maintenance of the Partition Database, containing a description of each partition plus any additional information pertaining to a partition. This would include the Partition ID, a number uniquely assigned to a partition.
- The provision of access methods to allow any user to interrogate the information contained in the database.
- Algorithms for generating all the necessary configuration information for the readout of a partition, enforcing the **Partition Rules**.
- Algorithms for generating all the necessary configuration information for the trigger system of a partition, also requiring the Partition Rules to be met.
- Utilities for monitoring partition usage, providing a status display, etc. and a user-friendly interface to the database for use by the operator.

The Partition Rules are limitations on the possible configuration of the Readout and Trigger system. They are imposed either by the topologies of each system (in which case the rules are simply an application of the information stored in the databases), by the availability of certain modules or connections (specifically a Trigger Supervisor and the FIO-FIO connections) or by rules imposed by the program to satisfy the requirements for a functioning partition (§ 2.1.2).

4.5.2 Design

The first step in the design of the PM was to separate it into the sub-packages listed in the previous section. Following analysis of the requirements, a design of the database structure (see fig. 4.4) was done. Once the information present in the database is well-specified, it is easy to determine what access methods are required. The philosophy of the Partition Database design was identical to that for the databases described in Chapter 3. Encapsulation of the data was considered obligatory. Similarly, the layered approach, by which there are a set of primitive operations possible, which directly access the data, and then macro functions superimposed that make use of these primitives, was also used.

The actual information content of the database was decided after a number of iterations of the design – prototype – test loop. Firstly, to describe a partition required a list of those detector elements defined as part of the partition. Whilst an efficient way of doing this

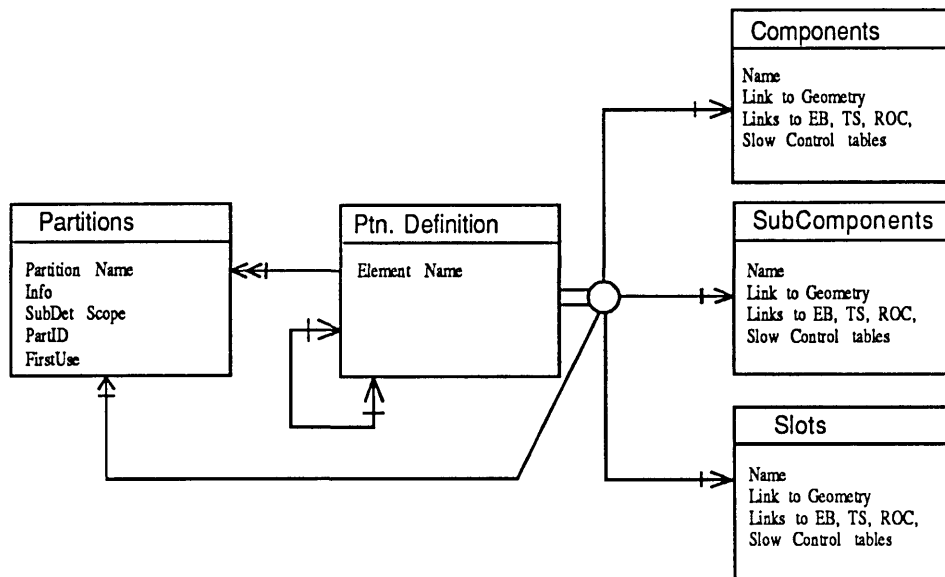


Figure 4.4: *Partition Description ER Diagram*

was via relationships, in practice this meant inextricably linking the Partition Database and Detector Description such that if the latter was modified, the Partition Database would require updating as all the pointers would change. Thus the link was made using the names of detector elements. Whilst a little slower, this method was very versatile — it allowed a partition to be defined in terms of other partitions, since the database entry is a name which can be interpreted as either that of a partition or part of the detector. This ability to nest definitions meant that sub-detector groups could operate and maintain their sub-detector partitions (**sub-partitions**) independently — for example changing the definition slightly if the sub-detector has hardware problems. This leaves the **main partitions**, such as ALEPH, unaffected. Name clashes are possible but this is dealt with simply by assuming the name is a partition first, and then a sub-detector element the second time around.

For full readout configuration, each node in the readout tree, either an EB or a ROC, needs to be provided with a number of items, stored in what is termed the Run Configuration Table (details in [31]). Those obtained from the Partition Manager are as follows:

1. Partition name and ID
2. Whether the node in question is a Trigger Supervisor
3. Fastbus Masters: addresses and type
4. Fastbus Slaves: addresses and type
5. Whether or not a **Spy Channel** is available

Additional information is required, but this is obtained from the Run Control database.

The procedure goes as follows: Given the module name, the PM uses the readout database and the detector description (refer to § 3.3,3.2) to determine which ROPs are at the next higher level in the hierarchy (masters in the previous list). Reference to the Fastbus database indicates whether spying is available or not. Then the Readout database is used again to find all the ROPs in the next lower level that belong to the partition (called slaves in the list). The sequence continues until the last node in the tree is reached. Reference is made again to the Fastbus Database to obtain information on the type of ROC, as the Readout Library (§ 2.5.3) must adjust to the small hardware differences between the ROCs.

The algorithm for Trigger Route finding was described in § 3.3. For this, a Trigger Supervisor is required. The TS for the partition is located in the following way: From the lowest detector element in the partition, a check is made at each level for a TS module. If one is available, its name is noted and booked as a resource in the partition. If not, the search proceeds up a level, until the main TS is reached. If this is busy, the search reports failure.

With knowledge of the Trigger Supervisor and the list of lowest level ROPs (ROCs), the partition trigger requirements are defined and the trigger routes can be located and programmed.

4.5.3 Other features

Since all the partitions are booked using the Resource Manager (see § 4.3), scanning over all booked partitions or detector elements allows the current status of the detector and partitions to be viewed. This feature is used to provide various monitoring utilities.

4.5.4 Implementation

As with the Readout and Detector Description databases, the database section of the PM was implemented as a shareable image. The method of encapsulating the data by access functions was also applied because of the advantages stated earlier (see § 3.5.3). The extra access routines plus all the more complex algorithms and booking routines were also placed in a shareable image library like the Resource Manager, using Transfer Vectors for all the routines (see § 3.5.6 for discussion of this topic).

The information held in the Partition Database consists of the following:

- The Partition Name

- The Sub-detector Context (this is the name of the sub-detector if this is a sub-partition). Sub-partitions for the moment are arbitrarily confined to one sub-detector. This restriction is imposed by the user-interface and is not present at the database level, and can thus be removed if felt unnecessary. One could imagine, for example, defining partitions by geometrical boundaries.
- The list of elements making up the partition: either sub-partition names or names of detector elements.
- General comment text describing use of partition/changes/etc.

The Partition Rules were implemented in a variety of ways. As mentioned, many of the requirements were controlled simply by the restrictions in topology imposed by the Detector and Readout descriptions of the database. For example it is not *possible* to program an FIO to send signals to a non-existent ROC (unless of course the database is incorrect, but this can be checked by a scan of the Fastbus system). This illustrates the advantage of correctly modelling one's system in a database, and using the information therein. Control of Trigger routing was done by booking the ports of each FIO unit whenever used — see § 3.4.2. The PM also provided built-in rules for whenever default cases arose — for example, in the case of no dedicated sub-detector Trigger Supervisor (this was the case for the smaller sub-detectors), the PM would find the nearest one available. By nearest, this refers to the minimum number of layers of the Readout Tree traversed. Usually this would be the Main Trigger Supervisor. If one Event Builder is flagged as not in use, the PM automatically uses the next upstream. This means, in principle, that the system can be read out directly from the Vax (this has not been attempted, as it would be very inefficient !). Lastly, a careful choice of resource name ensures that partition, trigger route and resource booking function correctly: For example, if "TPC" is booked, then all of the "TPC" sub-detector is in use and it is not necessary to book "TPC_FIO" (the TPC Fan-In Fan-Out unit) as this is included already. However, if "TPC_SIDEA" is booked, then "TPC_FIO" is not in the partition. The port used to transport the trigger signals for this partition IS booked though, by the route-finding algorithm. This means that the FIO is available to another partition for use (and therefore re-programming), but with restrictions. Similarly, the TPC Event Builder, "TPC_EB", would need to be booked if any part of the TPC below it is used in the readout. In this case it is a multi-user device and can handle more than one partition simultaneously (as can the software running on it — SRH, BM, etc.), so exclusive booking is unnecessary. If the whole TPC is used, access to the EB is undesirable (and would presumably be useless for readout purposes), and is prevented, as the resource name indicates it is booked as part of "TPC".

When partitions are opened, all the sub-partitions and detector elements are booked as part of the partition. Since the PM database routines (and the Readout Database

routines) check for booked resources before they permit modification, this prevents any changes to open sub-partitions or readout modules currently in use, whilst the partition is open. This is a useful security feature as it avoids transient changes in the databases without the Run being stopped (which may occur during DAQ system problems), which would be difficult to track Offline. The PM also checks that a sub-partition is not booked or used in the definition of another partition before it allows it to be deleted from the database.

Dealing with partitions defined in terms of other partitions (sub-partitions) was handled by a recursive algorithm ⁴. Every name in a partition element is checked to see if it is a partition. If so, and that partition is not already “unravelled” (the process of turning a partition into a list of detector elements), then the function to do this is called. If it is already unravelled, then it must be a detector name as well. If this is not the case, an error has occurred — it is signalled, and the procedure terminates with an error condition. If another sub-partition is found, the function makes note of where it has got to, and calls itself with the new partition name.

4.5.5 Opening a Partition

A partition is opened by the Partition Control Process. Child processes are started to perform data-taking tasks, and they inherit the partition ID. These child processes may then use the ID to signal to the Resource Manager (ARM) that they are partition members. They can then freely book resources within the partition according to the standard rules, which still apply inside a partition.

4.5.6 The Partition Manager User Interface

The Partition Database Editor, which is the user interface to the database, was implemented using the standard ALEPH menu package, UPI [30]. An example menu is shown in fig. 4.5. This interface also made checks on the names used in the partition definitions — they must either be detector element names or partitions. As mentioned earlier, there is also an extra restriction that a sub-detector partition can only be made up from that sub-detector’s detector elements.

⁴Recursion in FORTRAN ...!

```
-----Partition Editor-----
Subdet Context ALEPH____
List - Select by:
Partition Name ALEPH____
-----
Show Definition
Add Partition _____
-----
Modify Partition
-----
Delete SELECTED Partition
+++++ Actions +++++
Open a Partition (activate)
Close an active Partition
-----
[QUIT]
-----
```

Figure 4.5: A Screen Image showing part of the Partition Database User Interface

Chapter 5

DAQ: Conclusions and Summary

5.1 Mode of Operation

The multi-partition principle has proved very valuable. Whilst setting-up the individual sub-detectors before data-taking commenced, each sub-detector group had their own environment within which to work. Initial all-ALEPH runs were performed by starting a full ALEPH partition, and then any sub-detector with problems was removed temporarily, while the remainder continued. Sub-detectors could then re-join at an appropriate break in data-taking. During development periods, many partitions run concurrently, with the majority being dummy (empty) for test purposes. There are also extra sub-detectors available for tests (they are present in the detector description but have either minimal or no associated readout hardware).

5.2 Taking ALEPH Data

Data-taking activity is controlled centrally from the operator's console. Whilst a number of sub-detector experts are commonly present, all sequencing is controlled from this one console, by a single person.

5.2.1 Example of a Run Sequence

The Server application is started. This allows the operator to start the necessary tasks via the Scheduler. An Ashell application is started, from where the operator can request a partition be started. Once running, the Partition Control Process opens a partition, and provides a list of applications, or activities, that can be run within the partition. The Run Controller is one, and starting this enables the operator to begin a data-taking run. From the Run Controller menu, a run is started. All the extra tasks required for a

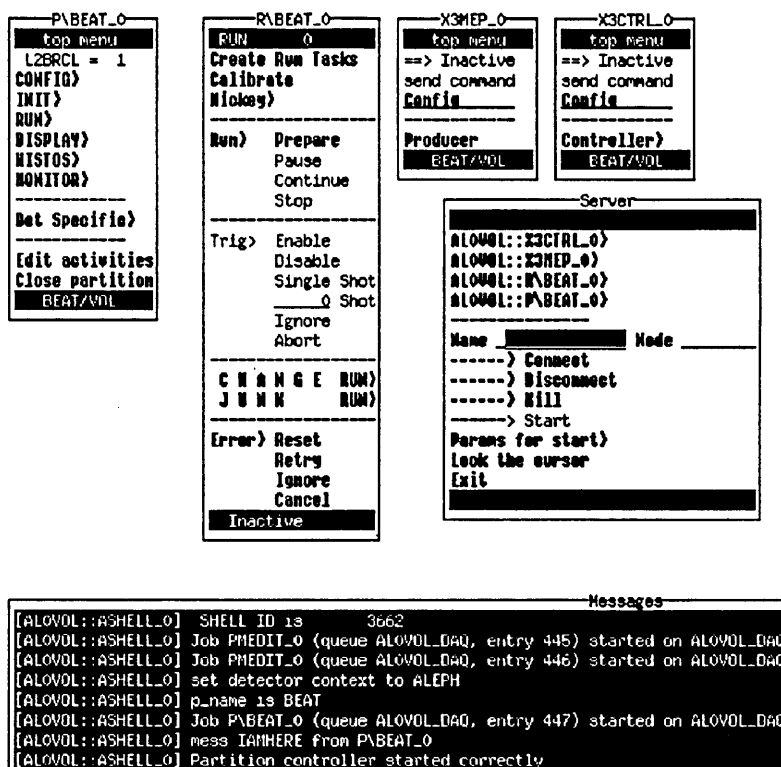


Figure 5.1: An Example Session: Starting a Run

run are initiated by the Scheduler on request from the Run Controller, including those on the micros. Data-taking continues, once the trigger is enabled, until either a run stop is requested or a problem occurs. The run may then be terminated, the partition closed, edited, and re-opened for a calibration, followed by more data-taking. Figure 5.1 shows an example session.

5.3 Conclusions

The Data Acquisition system has been in constant use since ALEPH began taking data and has proved very reliable as a whole. The overall DAQ efficiency for the 1990 running period (calculated as luminosity where data was taken divided by the available luminosity provided by LEP) was about 87%. Of this overall figure, 3% was due to dead-time (as a consequence of the Level 2 trigger), and about 10% due to hardware faults or problems in the sub-detector readout. Losses due to the Vax-based software were close to zero.

One very useful feature was the widespread uniformity in the software as a consequence of the number of packages designed to perform most of the common tasks in a general and uniform way — the DAQ Service Layer.

The flexibility of the system has proved very useful — during program development,

the setting-up period and data-taking. New sub-detectors have been successfully added, split and re-arranged, requiring no design changes and only minor code changes. Extra modules were added, requiring updates in the readout description and trigger connectivity. In general, for a system to be adaptable it must have a description from which all tasks are driven. Complex databases are a necessary part of such a system, but a distributed database facility with adequate performance and the required functionality is not available commercially. The exploitation of the facilities of the VMS operating system have allowed these problems to be overcome in-house.

During the system evolution, certain Software Engineering techniques and principles were used more than others. The task architecture and data flow descriptions were written early on and the initial design has remained essentially frozen. The reasons for this are twofold. In some cases no design changes have been required. In the other cases, the facilities of the available tools meant that changes to the design and the propagation of these to the code had to be done manually. This would have involved much effort and could have introduced errors. The data description itself, however, written in DDL, has evolved. This was simpler as the chain from DDL to FORTRAN data structures is automated. A widespread understanding of good coding techniques prevailed throughout the group (such as modularity, information-hiding, well-defined interfaces, etc). This, it was felt, contributed greatly to the robustness and consistency of the final system. The maintenance of documentation had similar problems to those of the design documents — code was updated, improved and appended to. The documentation, either in printed form, and therefore fixed, or in computer help files, remained static. There is an obvious need for a facility to extract documentation from the source code itself. It would seem necessary, however, for this to be useful, to either enforce standards for the commenting of code, or make the documentation a necessary part of the code (perhaps via interface specifications *à la* ADA ?).

One important aspect of Software Engineering that is missing in this technical description is that of organisation of the system as a whole [43], including such things as version releases, backwards compatibility and testing of software. Rather late on in the development of the DAQ, a disciplined method for performing these (somewhat tedious, but necessary) tasks was introduced [44]. This resulted in a much stabler system for the sub-detector groups to work with, allowing them to concentrate on their own problems, relatively isolated from changes in the underlying system.

The system is currently (i.e. at time of writing) undergoing an upgrade involving some changes in the software described. Some of these changes are outlined below in § 5.4. Most have arisen from experience of the running system and a desire to improve it, rather than dissatisfaction. This illustrates well that the most critical aspect of any DAQ system is not that it does everything but that it is simple to change or add functionality. With the emphasis that was placed on well-structured software, it is hoped that this job is made

easier.

5.4 How to do it better next time

Despite the success of the system, it is always possible to improve things. One obvious enhancement would be the extension of the Readout Database to describe the entire Fastbus system, including the front-end electronics. At the time of design, this was felt too sub-detector specific, and would probably have taken too long to implement.

A more profound change is being tested at the time of writing. Following experience, it seems more natural to describe the readout hierarchy purely within the Readout Database itself. i.e. to add a relationship “Read-Out-By” to the EB and ROC tables. This then obviates the need to navigate to the Detector Description, up one level, and then back to the Readout Database. One feature of the old system is thereby lost — if there were two EBs at the same level, disabling one meant that the other would be used automatically. The idea is also, however, to add a facility to change the readout hierarchy “on-the-fly”. This will allow any reasonable readout configuration to be programmed by the operator.

Some ideas exploited in the system are applicable to any complex data acquisition system; for example: the sequencing of events, the dependency of one task on the successful operation of others, the description of the readout in a generic manner and the partitioning concept. It seems desirable to incorporate these in a generic DAQ system that can be applied to other experiments, reducing the effort required to get a DAQ system up and running. Indeed, in much the same way that the detector hardware can be modelled and simulated using packages such as GEANT [45] or GISMO [46], the DAQ system should be described and operated by generic software, with as few as possible additions to cope with the specifics of the detector. The benefits in terms of maintainability and cost are clear. A corollary to this is, for similar reasons that hardware from older experiments is not re-used, that software technology is always advancing, and in HEP, the requirement for the latest technology is often stronger than cost.

The question of computer languages for use in HEP is always a hotly debated issue, and to date, FORTRAN has always been preferred. A few comments on this issue follow.

5.4.1 Language Issues

In Online systems, which are essentially software systems for a dedicated piece of hardware, portability of code is less of an issue. Of primary concern is performance, the second, support for the language on the system, and then other matters such as functionality, readability and data structures.

It is obvious that databases and data structures will have major relevance to future and current online systems. Here, standard FORTRAN is lacking. There are a very limited number of data types, and no way of defining user-specified types, and whilst packages have been written to overcome this limitation, they suffer from a lack of type-checking, run-time overhead, unhelpful diagnostics and somewhat obscure manipulation methods. Data structures are provided by the Vax FORTRAN compiler, but they are non-standard. Secondly, FORTRAN data is either local to a routine or completely global (in a COMMON block). Ideally, one would provide data that has a scope somewhere between, such that the data is directly visible only to the access routines and external routines *cannot* affect the data, save via these access methods.

As mentioned, C was the only supported language for the micros, and the influence of the language grew as a result. One of the changes in progress now involves the creation and access of a Histogram database, but using C structures and C routines in preference to FORTRAN. The format of the C structures is backwards compatible with the old FORTRAN structures and COMMON blocks [47]. Lately, C has also gained in popularity due to the arrival of X-Windows ¹. This product provides very useful features, including some functionality that replaces part of that provided by the DAQ services (the Server). It is written in C and when calling the library functions, the code is far simpler and readable in C than in FORTRAN.

C, however, uses somewhat terse syntax, and there are more sophisticated languages preferred by computer professionals. Any evolution in HEP will be a gradual one, in order to preserve the existing investment in software and expertise.

¹X-Windows is a windowing, graphics and communications protocol developed at MIT.

Chapter 6

Analysis Overview

The analysis sequence employed in this study is outlined and is followed by a brief description of online data reduction and the experiment trigger. The known properties of the K^0 particle are then summarised and the coordinate system and track parameters used in ALEPH explained.

6.1 Overview of the Analysis Procedure

The LEP collider is the world's highest energy e^+e^- storage ring. Measurements of many physical parameters and particle properties remain to be made at these energies. This analysis presents the first ALEPH measurement of the cross-section for

$$Z^0 \rightarrow q\bar{q} \rightarrow K_S^0 + X \quad (6.1)$$

at centre-of-mass energies near the Z^0 pole.

The analysis proceeds in a number of stages: from the DAQ system, the data is processed by the reconstruction program, JULIA, and then transferred to the CERN computer centre. The data is then in a format known as the POT, or Production Output Tape. By eliminating rarely used data from each event the data size is reduced producing a DST, or Data Summary Tape, format. This is then available for use by analysis groups. For this study, the data were passed through a further processing stage where loose selection criteria (cuts) were applied to the sample. The remaining data were then studied in a final analysis step, where the K_S^0 candidates were required to pass a number of tighter selection requirements. The choice of cuts and the algorithms employed are discussed in § 7.3.3 and chapter 8.

6.2 Online Data Reduction

The raw data produced by the experiment is in the form of a list of digitisings and channel numbers specifying where the passage of a particle was detected. These can be translated into coordinates by knowledge of the channel to detector position relationship, obtained from the offline database. Each such coordinate is referred to as a hit. The data stored on tape consists of all these hits plus calibration constants and book-keeping information. Additionally, information from the three stage trigger is also recorded. The amount of data per event is very large — approximately 100 Kbytes for hadronic events — thus it is necessary to reduce the space used to as little as possible. This data reduction is done in a number of stages, each optimised for the data size and rate at that level. The process can continue right down to the level of the individual physicist in order to minimise the resources occupied by data sets and improve throughput. A discussion of the philosophy behind this can be found in [48].

6.2.1 Zero Suppression

Some data reduction is performed by the DAQ system by suppressing readout channels that contain no information. The relevant information is stored as [channel number, value], hence empty channels can be skipped. This represents a major saving as, for the majority of events, most channels are zero.

6.2.2 Multi-Stage Trigger

The multi-level trigger is the second stage of data reduction. The philosophy is to measure all interesting events — to do this, only very minimal physics requirements are imposed at the Level 1 and 2 trigger stages. This led to the choice of trigger conditions explained in § 1.3. At the third level well-recognised background events are rejected.

The Level 1 Trigger

As discussed in chapter 1, the Level 1 Trigger for hadronic events was based on a minimum energy threshold detected by the ECAL or charged particle penetration through the HCAL in combination with track data from the ITC. The trigger rate from Level 1 for the 1990 data-taking period was between 2 and 20 *Hz* depending on beam conditions and luminosity.

The Level 2 Trigger

The Level 2 Trigger was used for the 1990 data-taking period. It provided an extra level of rejection based on tracking information from the TPC as a replacement for the ITC requirement, giving θ information in addition to ϕ . The Level 2 decision was available within $67\mu s$ to miss only two bunch crossings in the case of a rejected event. After the Level 2 requirement, the 1990 trigger rate was reduced to around 0.9 Hz when running at peak luminosity.

The Level 3 Trigger

The Level 3 part of the trigger was only used to reject background events passed by the single neutral electromagnetic trigger, an additional Level 1 trigger used for $\nu\bar{\nu}\gamma$ events. It never rejected the total energy or charged track triggers.

The overall trigger efficiency for $Z^0 \rightarrow \text{hadrons}$ was 100% as a result of the redundancy provided by Level 1 [49].

6.3 The K^0 Particle

The K^0 and $\bar{K}^0$ mesons are charge conjugates of one another and possess opposite strangeness (+1 and -1 respectively), the quark composition of the K^0 being $|d\bar{s}\rangle$. When discussing the strong and electromagnetic interactions, since these interactions conserve strangeness, the $|K^0\rangle$ and $|\bar{K}^0\rangle$ are appropriate. However, the K^0 decays weakly, and the weak interaction does not conserve strangeness. Instead, one can construct linear combinations of the states $|K^0\rangle$ and $|\bar{K}^0\rangle$, namely $|K_S^0\rangle$ and $|K_L^0\rangle$ which are associated with the particles observed from weak decays. The states $|K_S^0\rangle$ and $|K_L^0\rangle$ have no definite strangeness.

The short-lived K_S^0 decays in only two significant modes [50],

$$K_S^0 \rightarrow \pi^+ + \pi^- \quad B.R. = 68.61\% \quad (6.2)$$

$$K_S^0 \rightarrow \pi^0 + \pi^0 \quad B.R. = 31.39\% \quad (6.3)$$

each with CP eigenvalue +1. The K_L^0 decays in many modes including [50]:

$$K_L^0 \rightarrow \pi^+ \pi^- \pi^0 \quad (6.4)$$

which is predominantly an eigenstate of CP with eigenvalue -1. If CP invariance were valid, it would follow that K_S^0 and K_L^0 were eigenstates of CP with eigenvalues +1 and -1 respectively. With a conventional choice of phase, the linear combinations

$$|K_1^0\rangle = \frac{1}{\sqrt{2}}(|K^0\rangle + |\bar{K}^0\rangle), \quad CP = +1 \quad (6.5)$$

$$|K_2^0\rangle = \frac{1}{\sqrt{2}}(|K^0\rangle - |\bar{K}^0\rangle), \quad CP = -1 \quad (6.6)$$

are eigenstates of CP . Thus CP invariance would imply $|K_S^0\rangle = |K_1^0\rangle$ and $|K_L^0\rangle = |K_2^0\rangle$.

However, experiments in 1964 [51] and more recently [52], have observed the decay

$$K_L^0 \rightarrow \pi^+\pi^- \quad (6.7)$$

in which the final state has $CP = +1$. This means that $|K_S^0\rangle$ cannot be identified with $|K_2^0\rangle$ nor $|K_L^0\rangle$ with $|K_1^0\rangle$. Instead, the weak eigenstates must be written

$$|K_S^0\rangle = \frac{1}{\sqrt{1+|\epsilon|^2}}(|K_1^0\rangle + \epsilon|K_2^0\rangle) \quad (6.8)$$

$$|K_L^0\rangle = \frac{1}{\sqrt{1+|\epsilon|^2}}(|K_2^0\rangle + \epsilon|K_1^0\rangle) \quad (6.9)$$

where ϵ is a small complex number.

However, as the $K_L^0 \rightarrow \pi^+\pi^-$ decay has a branching ratio of only 0.2% [50], and with a lifetime almost three orders of magnitude higher than the K_S^0 , for the purposes of this study we can use the approximation

$$K_S^0 \approx K_1^0 \quad (6.10)$$

$$K_L^0 \approx K_2^0. \quad (6.11)$$

6.3.1 Properties of the K^0

The K_S^0 has a mass of $497.671 \pm 0.031 \text{ MeV}$ [50]; its lifetime of $8.92 \times 10^{-11} \text{ sec}$ corresponding to a $c\tau$ of 2.675 cm . The main decay modes of the K_S^0 (equations 6.2 and 6.3) have branching ratios of 68.61% (charged) and 31.39% (neutral). The other modes (e.g. $K_S^0 \rightarrow \pi^+\pi^-\gamma$) have branching ratios at or below the 10^{-3} level. In this analysis the K_S^0 is detected solely through its decay to two charged pions. Conclusions are drawn for the K^0 by assuming the K^0 is made up from equal proportions of K_S^0 and K_L^0 .

6.3.2 Production Mechanism

The K^0 observed in hadronic events can originate from several different sources dependent on the origin of the s (or $\bar{s}$) quark:

- Directly from the decay of the Z^0 into two quarks, in this case: $Z^0 \rightarrow s\bar{s}$. Here the primary quark ends up in the K^0 .
- From the hadronisation process, i.e. an $s\bar{s}$ is pulled from the vacuum.
- From decays of particles containing heavier quarks such as b or c , e.g. $D^- \rightarrow K^0\pi^-\pi^0$.

Source	Fraction
Primary s -quark	25 %
Fragmentation s -quark	40 %
Decay of b, c mesons/baryons	35 %

Table 6.1: *Predicted origin of K^0 s -quark in Z^0 hadronic decays*

The ALEPH Monte Carlo (DYMU2+HVFL) [53], based on the Lund model [54, 55], predicts that the origin of the s -quark is as listed in table 6.1. The simulation also shows that a negligible source of K_S^0 is from regeneration from K_L^0 in the detector material. Additionally there are a small number of K_S^0 produced by material interaction (of K^+ and K^-). The sum of these two processes is predicted to be approximately 0.4 % of the total K_S^0 multiplicity.

6.3.3 Hadronisation Models

Existing theoretical models for the hadronisation of quarks and gluons combine predictions of perturbative QCD with phenomenological models to describe the non-perturbative regime. The most successful model for ALEPH [56] has been the Lund model [55] which describes the fragmentation of quarks and gluons in terms of parton showers [55]. Successive branchings of the type $q \rightarrow qg$, $g \rightarrow gg$, $g \rightarrow q\bar{q}$ are repeated until the invariant mass of the resulting two-parton system falls below a specified cut-off parameter. The model possesses a number of parameters which can be “tuned” to describe the real physical processes better. Ideally, these parameters would be fixed theoretically, but to date most are chosen to fit data results. Clearly one measure of the predictive power of the model is how little the parameters need to be changed as a function of energy, since one set of parameters must describe many diverse physical processes.

6.3.4 Inclusive Hadron Production

The global properties of hadronic events have been measured at LEP energies [56, 57] using the ALEPH detector. The results for mean charged track multiplicity, the spectrum of charged tracks in $x = 2E/\sqrt{s}$ and the Z^0 hadronic cross-section are used in this analysis where appropriate. Many of the comparisons made with the Monte Carlo are also relevant: for the K_S^0 study, charged track reconstruction is critical. This has been shown to be well simulated by studies of charged particle multiplicity and angular distribution [56], tracking reconstruction quality and efficiency [4, 58, 59, 60] and hadronic event selection using charged tracks [49].

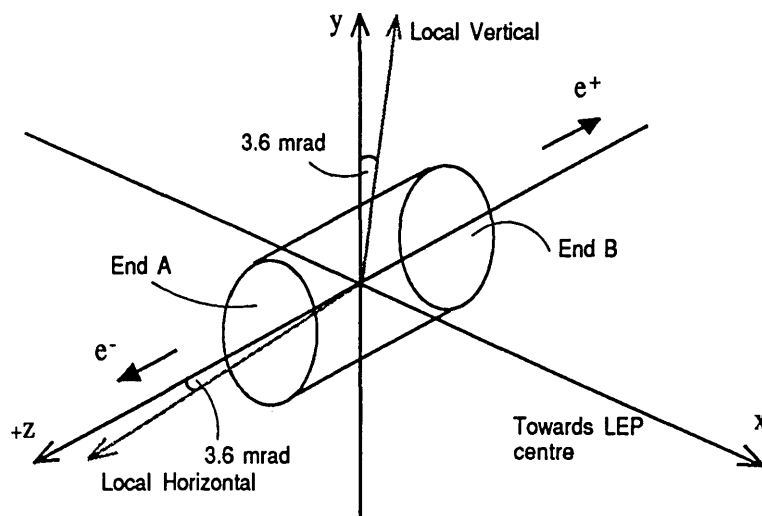


Figure 6.1: *The ALEPH Coordinate System*

6.4 Coordinate System and Track Parameters

The detector coordinate system is shown in figure 6.1. The origin is defined as the nominal centre of the detector. The x axis is in the horizontal direction pointing towards the centre of LEP, the y axis is vertically upwards and the z axis is along the direction of the electron beam. The electrons in LEP travel towards End A, the positrons in the negative z direction towards End B. If one stands in the centre of LEP, side A is to the left of ALEPH, side B to the right. In a cylindrical coordinate system the $r - \phi$ plane is perpendicular to the beam.

Charged particles travel in a helical trajectory in the solenoidal magnetic field. Some energy loss is incurred due to ionisation, but this is small — of the order of 5 MeV for a 1 GeV track [61]. The track parameters used to describe particle trajectories are shown in figure 6.2. In the $r - \phi$ plane R is the radius of the trajectory. The track passes through (x_0, y_0) at its point of closest approach to the origin. S is the distance along the trajectory in the $r - \phi$ plane; $S = 0$ at (x_0, y_0) . The angle Ψ is defined by $S = R\Psi$. The z dependence of the trajectory is given by $z = z_0 + S \tan \lambda$ where λ is the dip angle of the track and z_0 is the z coordinate at the point (x_0, y_0) .

Other useful parameters are:

- ω Signed inverse radius of curvature in the $r - \phi$ plane. ω is positive if the track bends anticlockwise in the $r - \phi$ plane. This sign corresponds to the opposite of the particle charge, for a magnetic field direction in $+z$.

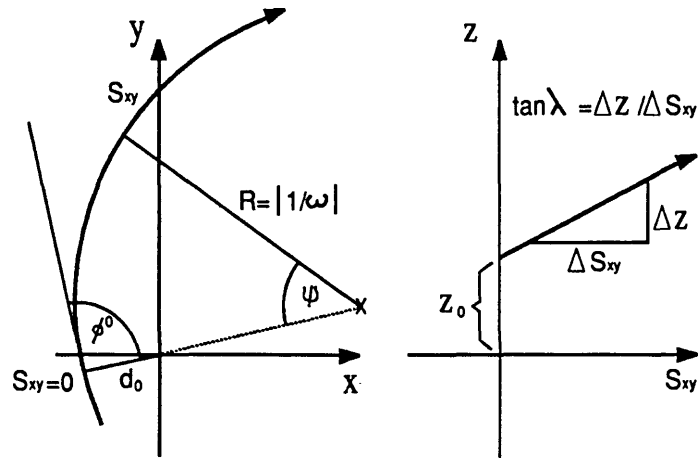


Figure 6.2: *ALEPH Track Parameters*

d_0 Distance of closest approach of track to z -axis, signed according to angular momentum of track about origin — positive angular momentum about z -axis gives $d_0 \geq 0$ otherwise $d_0 < 0$.

ϕ_0 Azimuthal angle of tangent to the track at (x_0, y_0) .

Chapter 7

Event and V^0 Reconstruction

In this chapter event reconstruction, the selection of events for study and V^0 finding and reconstruction are discussed.

7.1 The ALEPH Reconstruction Program

The tasks performed by the ALEPH reconstruction program, **JULIA**, are listed below and those applicable to this analysis are explained in more detail in the following sections. Data from the DAQ were processed by **JULIA** on a run-by-run basis. One run would typically consist around 1500 triggers, of which about 11 % were hadronic Z^0 decays and 10 % were Bhabha events.

- **Coordinate finding**

The information provided by the DAQ is usually in some coded format as output by the TDCs/ADCs/etc. Conversion from this into detector hits is the first stage of **JULIA**. These hits are then combined into coordinates [1, V.54].

- **Pattern Recognition**

This consists mainly of track finding using the coordinates from the tracking chambers, and cluster finding in the calorimeters.

- **Fitting**

A helix parametrisation is applied to those tracks found in the previous stage. Calorimeter clusters are combined into calorimeter objects which can be ECAL, HCAL or both and optionally associated to a charged track.

- **Particle Identification**

This uses a combination of information from the TPC and calorimeters: dE/dx , ECAL shower profile estimators and HCAL hit patterns.

- Primary Vertex finding

The event-by-event primary vertex is found by performing a global vertex fit to all tracks passing certain initial cuts.

- General Vertex finding

JULIA performs some general vertex finding and fitting for V^0 candidates. The algorithm used is outlined in detail in § 7.3.

- Monitoring

Monitoring of many quantities is performed to provide the offline analysis with summary histograms, cross-checks and useful average quantities. An example is the mean of the fitted primary vertex, which measures the beam envelope central position.

The output of JULIA is referred to as the Production Output Tape, or **POT**.

7.1.1 Track Finding

The investigation of V^0 s relies heavily on the performance of tracking chambers used to detect charged particles traversing the detector volume, usually originating from close to the interaction point. The tracking detectors, ITC and TPC, provide spatial, momentum and charge measurement, plus some particle identification. They were described in § 1.2.

Charged particles originate from near the interaction point and travel in a helical trajectory in the solenoidal magnetic field (see § 6.4 and [1, V.56] for the nomenclature). Initially only TPC coordinates are used for track finding. Track chains are built from neighbouring coordinates, starting with those coordinates at largest radii. The four steps in this procedure are as follows:

- Three points consistent with the hypothesis of a helix are located — the **triplet**. In order to construct a triplet, starting with a candidate coordinate, three different methods are tried in order (see figure 7.1): the first searches for a second and third coordinate in the direction towards or away from the origin (nominal primary vertex), the second finds the next closest hits, whilst the last loops over all neighbouring coordinates. Once a triplet is found the coordinates are not re-used for any other triplet. This is to reduce computation time. Initial helix parameters are computed with these three points.
- The part-helix is then extended in either direction, stepping by an appropriate interval and using a numerical approximation of the helix equation to calculate the updated parameters. Coordinates that lie in the spatial region closest to the

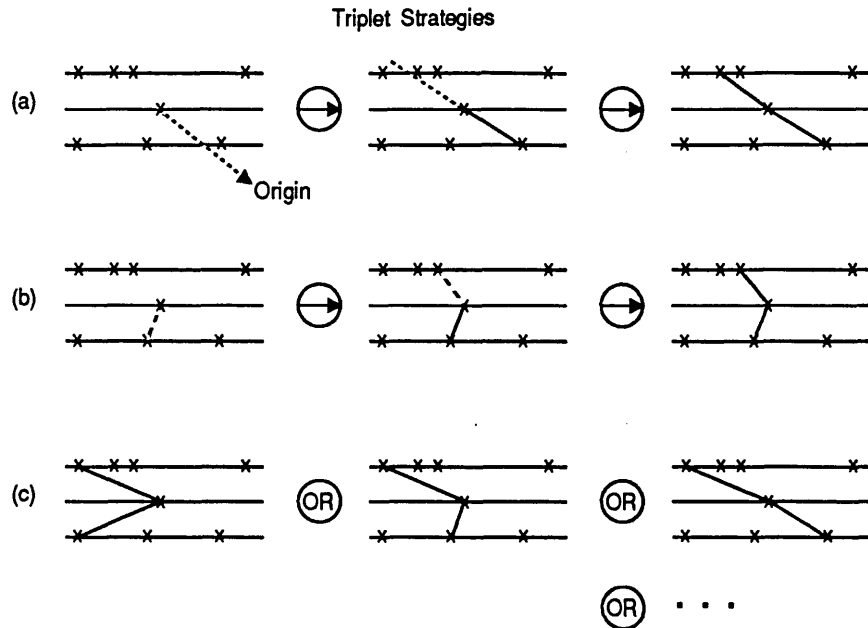


Figure 7.1: *The three strategies used to find an initial track segment, or triplet. The first is efficient for high momentum tracks originating near the primary vertex; the second is efficient for spirals. The last is intended for messy situations.*

continued helix are added to the track segment. The helix parameters are then re-calculated, until no more coordinates are located.

- A fit to the helix parameters is performed (see § 7.1.2). The result is then extrapolated to larger radius. Coordinates closest to the helix path (within some “road”) are included in the track fit, provided they are reasonably consistent with the existing fit .
- The helix parameters are re-fitted once more and the result used to extrapolate to smaller radius.

Chains that are consistent with the same helix are combined into single tracks as there may be breaks due to dead zones. A final re-fit is then performed. The procedure is described in detail in [62, TPC Tracking Software].

Tracks in the ITC are located by extrapolation of TPC tracks. Any points in the ITC within a road defined by the fitted helix are added to the track provided they are consistent i.e. lie on the helix within the ITC measurement error [62, ITC-TPC Tracking Status].

At the last stage, any ITC coordinates remaining are used to find tracks within the ITC alone. Since the ITC has a slightly larger angular acceptance than the TPC, it would be productive to extrapolate these tracks back into the TPC to attach any unused TPC

coordinates, but this was not done.

7.1.2 Track Fitting

There are two track fit methods used in the tracking software. The first is a simple circle fit in the $x - y$ plane plus a straight-line fit in the $s - z$ plane (fig.6.2 [62, The Helix Fit, § 4.6.1],[63, 64]. This can be followed optionally by a simultaneous Newtonian iteration in all helix parameters, but was not used to date, as the projections are sufficiently accurate. The second is an algorithm based on the Kalman filter [62, Track Fitting, § 4.5 and Global Track Fit, § 8]. It can deal with kinks in tracks (due e.g. to multiple scattering at the ITC-TPC boundary) and also energy loss along the helix. A feature only recently used has been the ability of the algorithm to optionally reject bad coordinates if the fit improves by some noticeable extent.

The chain finding and linking use only the simple fit, as it is much faster. For the final track fit, however, the latter method is used as it is rather more rigorous in its treatment of errors and uses a more acceptable approximation for multiple scattering. The result of the fit is the optimal set of helix parameters plus a covariance matrix giving the estimated errors on all six parameters and their correlations.

The full performance of the tracking in ALEPH (and the TPC in particular) is discussed at length in [4].

7.1.3 Primary Vertex finding and fitting

JULIA calculates an event by event primary vertex position by combining as many “good” tracks as possible into one global vertex fit [65]. Only tracks with momenta larger than some minimum value and originating near the origin are used. All combinations of tracks that satisfy a minimum multiplicity requirement are tried in the vertex fit. The combination with a minimum confidence level of 99.9 % for that number of tracks and lowest χ^2 is kept. Remaining lower momentum tracks are then added provided they are compatible with the existing vertex. The method can take into account the constraint of the beam envelope which improves the resolution in the y direction by more than one order of magnitude.

JULIA also keeps a running average of the found primary vertex for the duration of a run. This gives a good determination of the beam centroid, provided that the beam position does not fluctuate excessively during the run. For the 1989 and 1990 data-taking periods, this was not a problem.

7.2 Event Preselection

As part of the reduction process, the data was passed through a classification stage where some selection on event type was performed. These types, such as “hadronic selection by charged tracks”, “dilepton candidates” and “hadronic selection by calorimeter information”, each correspond to a classification number (e.g. Class 16, described below). Each event could be assigned one or more numbers. During analysis, events satisfying combinations of types could be preselected without the need for further processing.

The quality of the data was split into three categories, Perfect, Maybe and Duck (not useful for analysis). For a run to qualify as Perfect or Maybe, the TPC must be functioning. For this analysis therefore, both Perfect and Maybe events are used.

7.2.1 Hadronic Event Selection

The category used for this analysis was known as Class 16, selected as a hadronic event by the following criteria:

- At least five good tracks, with a good track defined as having:
 - ≥ 4 TPC coordinates;
 - $d^0 < 2 \text{ cm}$;
 - $z^0 < 10 \text{ cm}$;
 - $\cos \theta < 0.95$ i.e. the track must cross at least 6 TPC pad rows.
- Total charged energy of all good tracks, assuming pion masses, $> 10 \%$ of beam energy.

The resulting efficiency (estimated from Monte Carlo) is $97.4 \pm 0.3\%$. A full discussion of the selection method, results and errors can be found in [60].

The major source of background in this selection comes from inelastic e^+e^- scattering known as two-photon events [60]. They contribute about $0.4 \pm 0.1 \%$ to the selected hadronic events. $\tau^+\tau^-$ -events contribute $0.26 \pm 0.03 \%$ [60]. These must be subtracted from the final number of events used in any $e^+e^- \rightarrow \text{hadrons}$ analysis. For this study, 120551 hadronic events were used, corresponding to $122952 \pm 369 Z^0 \rightarrow q\bar{q}$ events.

V^0 Particle	Mass (MeV)	Lifetime ($\times 10^{-10}$ s)	$c\tau$ (cm)
K_S^0 [49]	497.67	0.8922 ± 0.0020	2.675
$\Lambda^0 / \bar{\Lambda}^0$ [49]	1115.63	2.632 ± 0.020	7.89
γ	0.00	—	—

Table 7.1: *The Properties of Standard V^0 Particles*

7.3 V^0 Reconstruction

V^0 s are neutral particles which decay in flight into two charged particles. In particular, for a V^0 decaying in the ITC or TPC sensitive region, two tracks are seen originating at a space-point with no parent track. — see fig. 7.2, which is an example from ALEPH data. In addition to the above criteria, the decay vertex of a V^0 candidate is usually required to be well separated from the primary vertex. This can be used to distinguish it from the many fake combinations produced by tracks originating from the primary vertex.

7.3.1 V^0 finding

The general principle behind a V^0 finder is to find all pairs of tracks (one positively charged, one negatively) whose combination yields a parent particle satisfying the above definition of a V^0 .

The known particles for which this is the case are listed in table 7.1. Photons fall into this category as they can convert in material into an e^+e^- pair. They originate mainly from π^0 and η^0 decays and, to a lesser extent, as radiative products from the event. The relevant decays of the V^0 s are as follows:

$$K_S^0 \rightarrow \pi^+ \pi^- \quad (7.1)$$

$$\Lambda^0 \rightarrow p \pi^- \quad (7.2)$$

$$\bar{\Lambda}^0 \rightarrow \bar{p} \pi^+ \quad (7.3)$$

$$\gamma \rightarrow e^+ e^- \quad (7.4)$$

The general algorithm followed to find V^0 s therefore involves:

- Some quality cuts which are applied to all charged tracks;
- For good tracks all positive and negative track combinations are considered;
- The two tracks are required to cross in space within some margin;

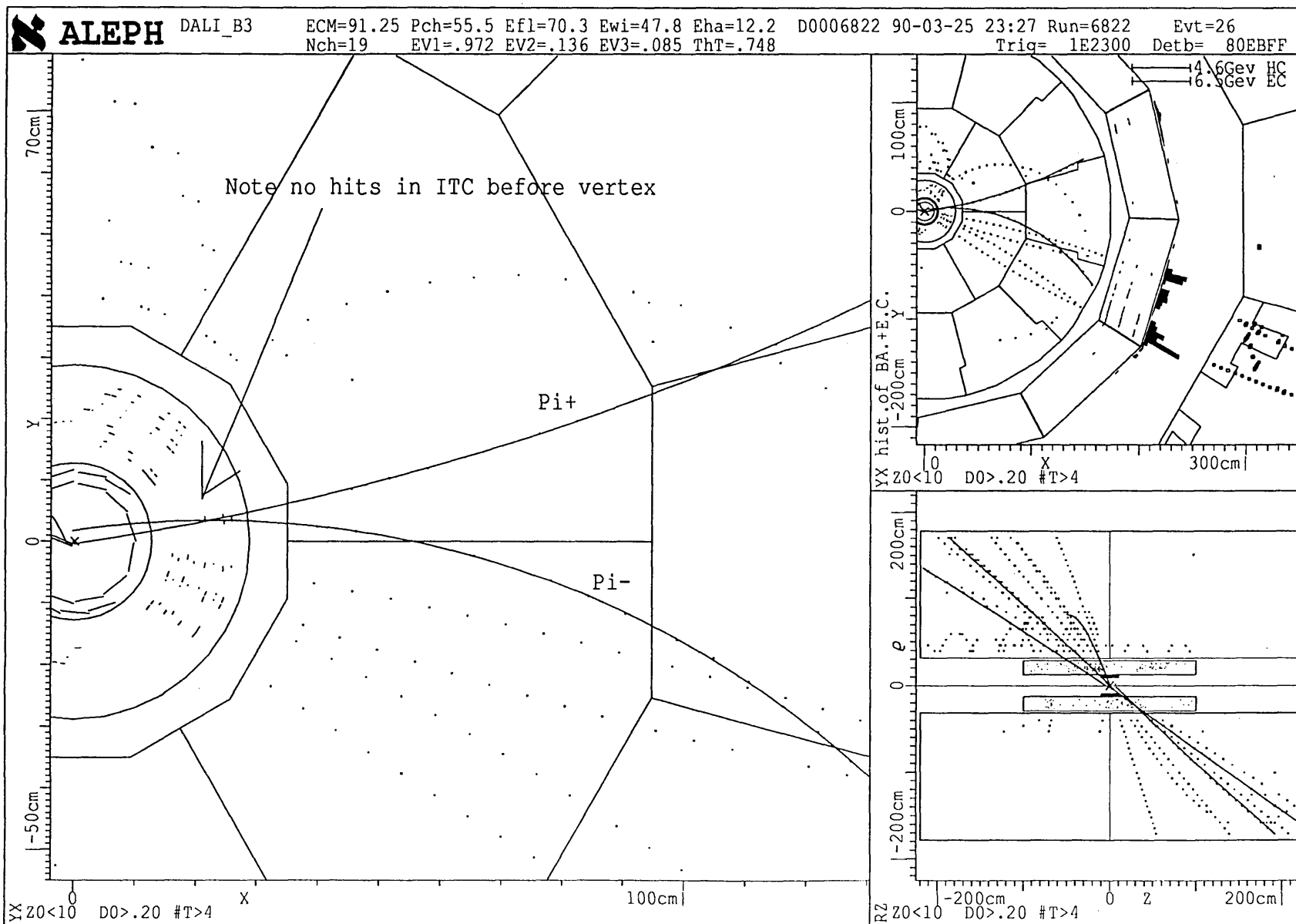


Figure 7.2: A V^0 Decay in the ITC

- The resulting vertex position must be on the correct side of the interaction point;
- Additional cuts which are performed to enhance the signal to noise and select particular V^0 species.

The specific method and cuts employed are now explained in some detail.

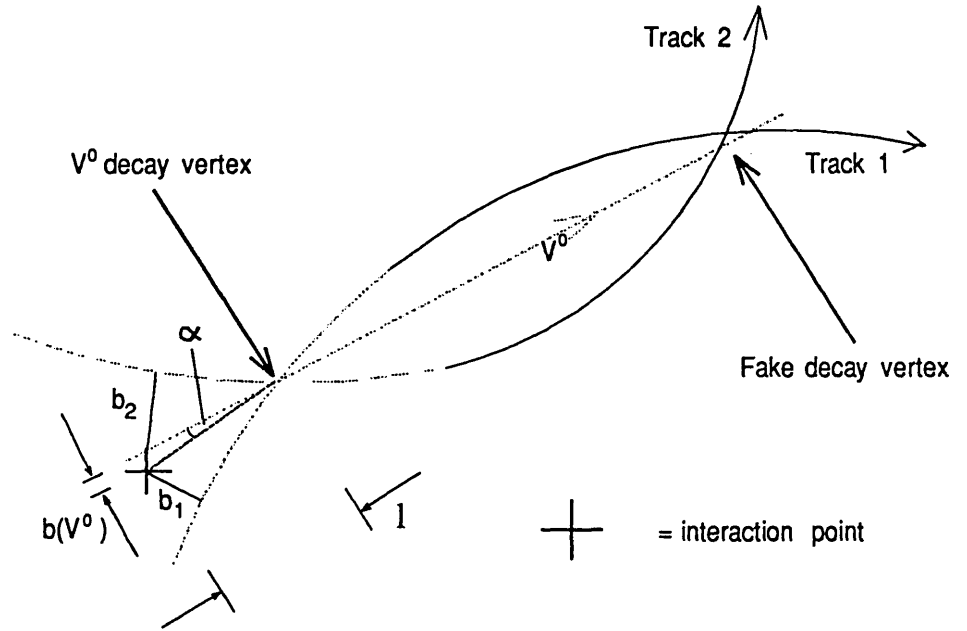
7.3.2 The YV0 Package

V^0 finding for ALEPH is performed by a software package known as YV0 (see [66, pp.59–70]). The algorithm is a development of that used in the UA1 experiment [67]. It is contained in the reconstruction program, JULIA, and the resulting V^0 candidates recorded on the POT. The standard cuts were, however, inappropriate to this study for three reasons. Firstly, they were designed to leave about 2 V^0 candidates per event (due to storage constraints). This reduces the efficiency unnecessarily. Secondly, a cut on the mass pull (defined as $\chi^2_{m_{K^0}} = (m_{\pi^+\pi^-} - m_{K^0})^2 / (\sigma_{m_{\pi^+\pi^-}})^2$ for a K^0 candidate) produced a peak in the background distribution centred on the K^0 mass. This made the background very difficult to estimate purely from data. Lastly, some of the cuts were on χ^2 quantities, the distributions of which showed marked differences between Monte Carlo and real data samples (such as figure 8.2). This was mainly due to an underestimation of the errors from the track fitting for real data. The agreement has been greatly improved by better detector alignment procedures [4, 68] but the latest results were not available at the time of this analysis. Whilst very powerful, cutting on a particular value would result in the method rejecting more Monte Carlo candidates than data, giving an incorrect efficiency. These problems are further discussed in § 8.2.

For these reasons YV0 was run as a stand-alone program on both Monte Carlo and real data with these cuts removed and others loosened to improve the overall efficiency. One side-effect is that the size of the resulting data structure increases by a factor of four, but this is compensated for in part by not keeping all the available event information. A reduced data set, the **V0DST**, was made containing only tracking, V^0 and (if present) Monte Carlo particle history information. This gave a factor five speedup in running the final analysis stage, as a large proportion of real time is spent reading and unpacking compressed data. The size of the real data was also reduced to around 5 % of the DST. A full description of the selection cuts used for this analysis appears below; a summary is found in table 7.2.

7.3.3 Selection at the V^0 level

The procedure starts by considering all charged tracks and applying the following set of cuts (referred to later as the V^0 cuts).

Figure 7.3: V^0 decay topology

The V^0 Cuts — detailed in § 7.3.3 and figs. 6.2,7.3	Value
No. of TPC coordinates on daughter tracks	≥ 4
Polar angle of daughter tracks	$\cos \theta < 0.95$
Momentum of daughter tracks	$> 100 \text{ MeV}$
Minimum separation of daughter tracks	$ DM < 2 \text{ cm}$
Vertex on correct side of origin	Daughter track $\Psi > 0.0$
Decay vertex within TPC	$R < 180 \text{ cm}, Z < 220 \text{ cm}$
Coordinates on daughter tracks	None before decay vertex
Pointing angle	$\cos \alpha > 0.92$

Table 7.2: Summary of cuts applied to V^0 candidates

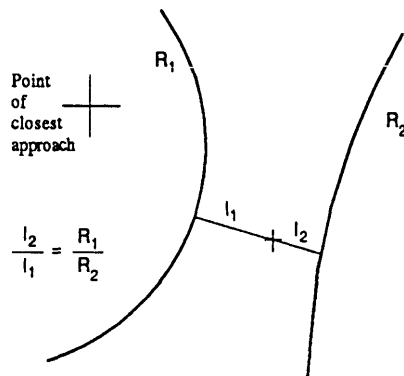


Figure 7.4: *The definition of Point of Closest Approach*

- ≥ 4 TPC coordinates. This cut ensures that the momentum measurement is reasonable. ITC-only tracks suffer from bad two-track resolution in jets, due to the single hit electronics, often giving four or less ITC coordinates per track. This does not give very good precision on the momentum or position.
- Requiring $\cos \theta < 0.95$ means the track must pass at least six TPC pad rows. This means that at least four coordinates should be available from one side of the TPC (a spiralling track can produce many coordinates but would not pass this cut). This is, in effect, a cut on the transverse momentum with respect to the beam, or p_T , of the track, of about 70 MeV.
- A cut at $p > 100$ MeV removes low momentum tracks which tend to have worse measured track parameters due to multiple scattering and/or energy loss in the detector material.

These are a subset of those cuts used to define “good tracks” for event selection (§ 7.2.1). They ensure each track is reasonably well measured.

For each positive and negative pair, the crossing point of the two tracks in the $x - y$ plane is located. If they do not intersect, then the point of closest approach is used. This point is defined as lying on a line connecting the two tracks such that the line has minimum length. The position of the point along the line is determined by weighting according to the radius of curvature of the track (see fig. 7.4). The maximum separation allowed (DM

in table 7.2) is 2 cm in (x, y, z) . As can be seen from figure 7.3 this can result in two solutions. The ambiguity is resolved by rejecting the vertex with the largest $|\delta z|$ between the daughter tracks. The phase angle Ψ of each track helix is defined as the angle of rotation along the helix about its central axis, zero being the point of closest approach to the primary vertex — where d^0 is measured. These angles must be positive at the approximate vertex, ensuring a positive lifetime. The approximate decay position must lie within the outer TPC boundaries (R and Z in table 7.2).

A vertex fit is then performed. The original point of closest approach is used as a starting value and the vertex position is found by iteration. The procedure modifies the daughter track parameters such that they originate from the same space point. The optimal position is found by minimising the required variation weighted by the errors on the respective track parameters. A measure of the goodness-of-fit in the form of a χ^2 is calculated and could be used as a discriminating parameter. If the fit does not converge, which occurs less than 0.1 % of the time, the V^0 candidate is rejected.

At this stage there are still many fake V^0 s due to random combinations of tracks coming from fragmentation. To reduce this background some cuts are made using the found vertex position. A candidate is rejected if:

- One or both decay products have coordinates before the found crossing point. This can reject a few very low momentum K_S^0 s where one pion actually travels backwards in the lab frame, but in most cases the track would not be reconstructed anyway. A K_S^0 decaying within a jet can result in hits due to other particles being wrongly assigned to one or other daughter track. The fraction of candidates rejected by this cut was 0.5 %.
- The angle between the V^0 line-of-flight and its reconstructed momentum is less than 18° (α in fig. 7.3 and table 7.2). This is a cut against K_S^0 s produced in material interaction. For decay vertices close to the interaction point, this can reject real K_S^0 s if the position of the decay or primary vertex is badly determined. The latter problem is overcome by requiring a minimum length-of-flight of the V^0 (made in the following analysis stage — § 8.3.1).

7.3.4 V^0 Finding Efficiency

Losses after the V^0 finding stage are attributable to several sources:

- No track produced
Very high momentum Lambdas and Kaons, and those with lifetimes much larger than $\tau_{K_S^0}$ can pass through all the tracking volume before decaying. They will

therefore be invisible to the search for charged track pairs. Since the lifetime of both are well known, this effect is predicted by the simulation;

- Geometrical effects

One or both of the daughter tracks did not pass through the sensitive region of the detector. This is simulated correctly to a few parts per mil [58];

- Track quality cuts

Tracks were produced but failed the minimal track quality requirements. The cuts are chosen such that the tracks lie in a region which is well understood [56]. The error due to this effect was estimated by varying the cuts on track quantities and is included in the overall K^0 cut systematic error (Appendix A.2);

- Tracking losses

One or both of the daughter tracks was missed by the track finding. If the overall track finding efficiency is η_T , then the resultant efficiency for V^0 s will be $\approx \eta_T^2$. It is not exact since there is some correlation between the two tracks as they have a common parent. A value for η_T has been calculated [4, 69, 58] for tracks satisfying varying quality cuts, and is found to be approximately 98.6 % for tracks with momentum larger than 1 GeV . Since this study exploits tracks of lower momentum, however, a check for agreement between the simulation and real data is needed. This was done by studying the relative numbers of unused coordinates in simulated and real data, and by scanning a large number of both types of events for visible differences [60, 59]. Only 3 missing track candidates from a sample of 5500 were due to effects judged not reproduced by the Monte Carlo;

- V^0 cut losses

Any cut to remove combinatoric background on a statistical basis will also throw away some V^0 s. The contribution to the systematic error must be estimated from the simulation: this was studied by varying the cuts over a range of values.

The estimated individual efficiencies for these losses, for the $K_S^0 \rightarrow \pi^+\pi^-$ process, are shown in table 7.3. The track quality cut error is included later on, so for these sources an error of 0.7 % is quoted.

7.3.5 Cross-Checks on V^0 Finding

To check the functionality of the V^0 finding package, some Monte Carlo data were used to generate tracks with zero error i.e. they corresponded exactly to those generated by the simulation. This data was then passed through the algorithm to check that the procedure

¹The systematic error attributable to track quality cuts is included in that involving all variations due to cuts at the K_S^0 finding stage — see § 8.6 and Appendix A.2

Effect	Calculated Efficiency and Error
No track produced	$96.0 \pm < 0.1 \%$
Geometrical	$93.9 \pm < 0.1 \%$
Track quality cuts	$86.1 \pm 0.0 \%^1$
Track finding inefficiency	$95.1 \pm 0.1 \%$
V^0 cuts	$76.2 \pm 0.6 \%$
Overall	$56.2 \pm 0.7 \%$

Table 7.3: *The Sources of V^0 Finding Efficiency Losses*

works, that it introduces no bias on the central mass of any peaks and gives a mass resolution corresponding approximately to the known K_S^0 width. The output is shown in figure 7.5. The resolution of the histograms is not sufficient to distinguish the K_S^0 width, but they demonstrate that there is no in-built bias on the K_S^0 mass resolution. Moreover, they show that the background is not enhanced around the mass peak and that the K_S^0 mass (the peak centre) is not shifted.

As another systematic cross-check, a second V^0 finding package known as **YTOP** [65] was used to calculate secondary vertices. This program differed from the YV0 package in that it merely performs a geometric fit at the vertex and attempts no momentum re-fitting. It

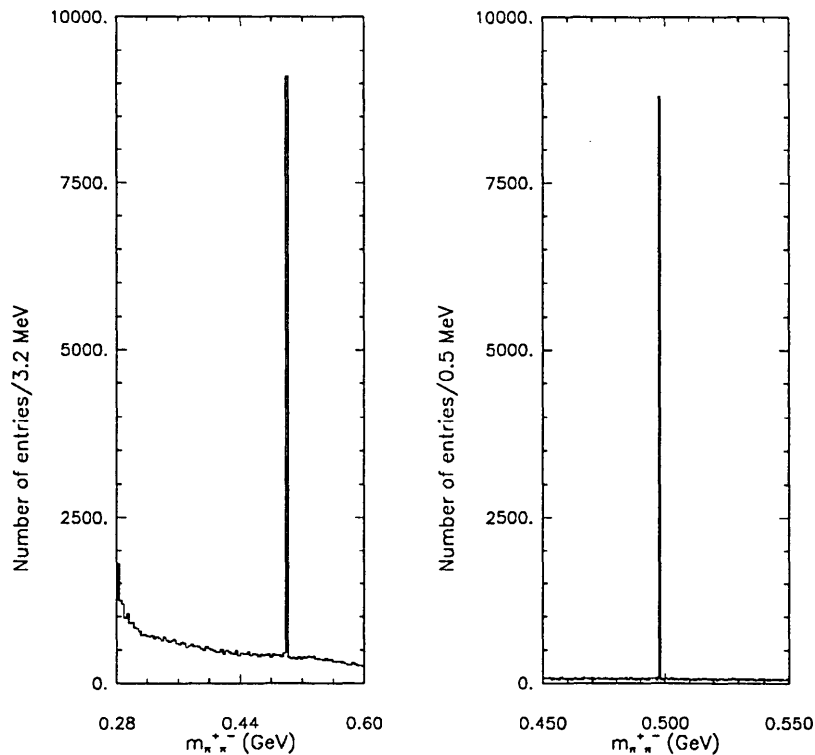


Figure 7.5: $m_{\pi^+\pi^-}$ for simulated data with perfect tracks. Left: The significant part of the $m_{\pi^+\pi^-}$ spectrum. Right: In detail, with a bin width of 0.5 MeV.

also imposes no requirement on track combinations, and can thus fit a secondary vertex to pairs of both oppositely signed tracks and same sign tracks. The same sign (V^{++} and V^{--}) combinations give an indication of the shape of the combinatorial background spectrum under the mass peak of the true V^0 s. This is discussed in § 8.4.

Chapter 8

K^0 finding

8.1 Selecting K_S^0 s and Enhancing the Signal

By exploiting the known properties of the K_S^0 (§ 6.3), the signal can be enhanced. In the following sections the choice of appropriate requirements for K_S^0 candidates is discussed. It is important to consider the possible sources of background to the $K_S^0 \rightarrow \pi^+\pi^-$ decay which are summarised in the next section. The studies made of the effects of these cuts both on the simulated data and real event samples are described.

8.1.1 Physics Background Processes

Standard V^0 particles other than K_S^0 give similar signals: Λ^0 s, $\bar{\Lambda}^0$ s and γ s. Most of the γ s are not a problem as they populate the low-mass region of the $m_{\pi^+\pi^-}$ plot (see figure 8.1) and give only a small, almost linear background in the region under the K^0 mass peak. The daughter tracks from these γ s have badly measured opening angles and hence an incorrectly measured invariant mass. Assuming each track is a $\pi^\pm$ as required for a K_S^0 leads to the distribution shown.

The Λ^0 s produce a wide bump in the $m_{\pi^+\pi^-}$ spectrum — this produces a non-linearity in the background that, save for using the Monte Carlo, is difficult to predict exactly and therefore take account of. A cut is therefore required to remove as much Λ^0 contamination as possible — see § 8.3.1.

The remaining physics background comes from short-lived particles which decay into two oppositely-charged daughters, such as $\phi \rightarrow K^+ K^-$. These can be removed by requiring the decay vertex to be well separated from the interaction point (§ 8.3.1).

The $m_{\pi^+\pi^-}$ distributions for various particle types

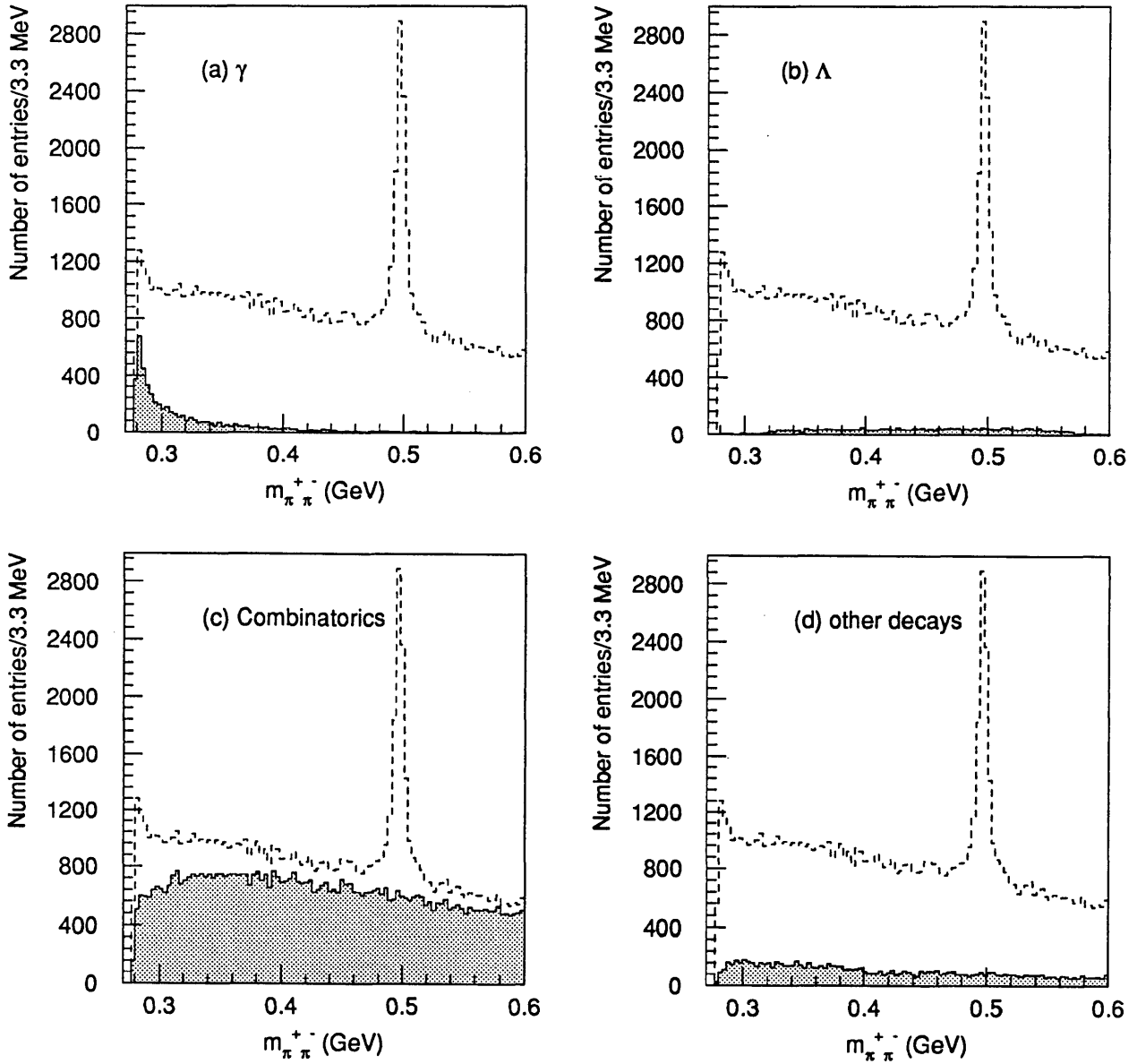


Figure 8.1: The K^0 background in $m_{\pi^+\pi^-}$ spectrum (simulated data) — after V^0 cuts. The visible peak in the total sample (dashed line) are the K_S^0 s.

8.1.2 Other Background

Fake K_S^0 s, where the candidate originates from random positive and negative track combinations, form the largest background in the V^0 sample. Most combinations will give an invariant mass outside of the K^0 mass range, and can thus be removed by a simple mass requirement. Some remain and they must be removed by either topological or kinematic cuts. The detector resolution enforces some limits below which combinations are indistinguishable from true K_S^0 decays. Thus the level of this background is in essence a measure of the tracking performance of the detector, and is reduced by improved spatial and momentum precision. Some finite background level is unavoidable (“combinatorics”, figure 8.1). For measurement of a cross-section, this background can be removed in the fit, and is not a problem. For the lifetime, the sideband-subtraction technique (§ 8.4.2) again corrects for the background. The main reason for reducing combinatorial background is to improve the signal to noise ratio which increases the statistical accuracy of the measured signal. It also aids searches for parent particles decaying into K_S^0 s.

Real K_S^0 s produced by interactions in material (as discussed in § 6.3.2) are also to be viewed as background as they do not originate from the processes under study. Cuts are used to reduce the contamination from this source (see § 8.3.1) and the Monte Carlo is used to estimate the (small) correction.

8.2 Motivation for cuts

The aim of this analysis is to measure the $e^+e^- \rightarrow Z^0 \rightarrow K_S^0 + X$ cross-section. With the large statistical sample of hadronic events, the systematic part of the error becomes the limiting factor in the measurement accuracy. One source of systematic error was the simulation.

The reproduction of standard parameters such as charge multiplicity and momentum spectra have been shown to be in agreement [55]. However, this study was undertaken during the early running of ALEPH, and a relatively small amount of Monte Carlo data was available. There was no full simulation of the vertex detector, which would have some effect on track d^0 resolution due to kinks. In particular, the effect of multiple scattering in the TPC gas is taken into account in the track fit only approximately. This was known to introduce a bias in the χ^2 -of-fit distributions for lower momentum tracks [62, Track Fitting, § 4.5]. It was felt that using cuts based on errors or quantitative “goodness-of-fit” (χ^2) would be unreliable and introduce a high systematic uncertainty.

The distributions of track fit quality (χ^2 -of-fit, fig. 8.2) for the simulation and real data, and the fractional difference, demonstrate that a cut involving this variable would remove more real data tracks than in the simulated data. This would lead to a bias in the efficiency

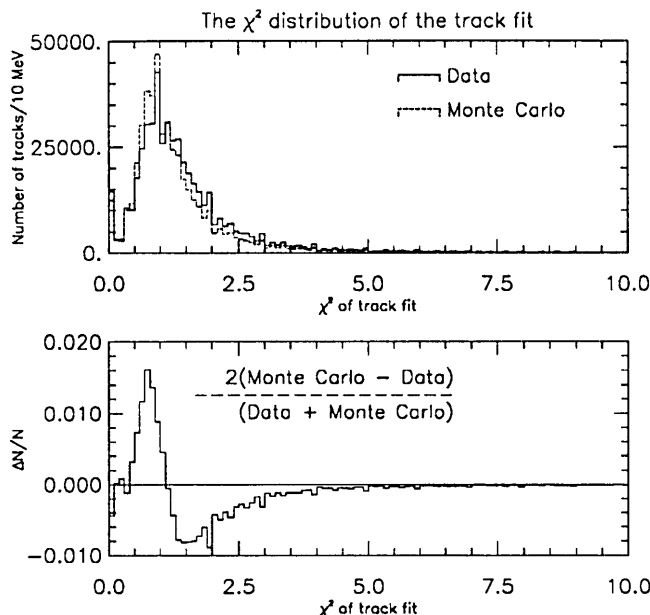


Figure 8.2: The Track Fit χ^2 distributions for Monte Carlo and real data

for either the background, signal or both. The effect is small, but non-negligible. For example, a cut at $\chi^2 < 5.0$ would remove 1 % more data tracks than Monte Carlo. A cut at $\chi^2 < 2.0$ would reject 6 % more. The effect is magnified for the χ^2 of the V^0 vertex fit which involves two tracks and a further calculation. Rather than introduce a correction procedure, which would itself involve some considerable systematic unknowns, these cuts were avoided.

Moreover, χ^2 cuts are time-dependent. As the detector became better understood, alignment improved, calibration procedures were refined and the simulation accuracy increased. χ^2 distributions are sensitive to these improvements. Changes in the real data distributions will not be reflected in the simulation. This source of problem is reduced during the lifetime of an experiment, as the apparatus and initial processing stages become stable. The Monte Carlo and data distributions shown in fig. 8.2 are much more similar than those from the earliest data-taking periods. It is thus felt that for any study reliant on the simulation, as is an efficiency calculation, selections based on such quantities should only be made if very loose, where only small absolute differences will be apparent, or after good agreement between simulation and data is independently confirmed.

For determination of the cross-section, it is vital to know the uncertainties in efficiencies if they cannot be reduced to a negligible effect, and for this, the Monte Carlo simulation must match the data extremely well. Each cut has been studied to check that the final measurement is invariant when the cut is loosened or tightened.

Lastly, if any single cut is very strong, then even a small systematic difference between the simulation and the real data for that quantity is likely to have a large effect on the efficiency. It is therefore the aim to avoid very powerful cuts unless the simulation is known to be very good. Although a method of choosing cut values by computer optimisation of the signal to noise ratio has been used previously [70], it was felt this would ignore the problems described in this section.

8.3 K^0 Reconstruction

The reconstruction of K_S^0 s consists of selecting those V^0 candidates that satisfy the cuts described below. They are chosen with a view to avoid the systematic problems explained in the last section. This was considered more important than choosing cuts which would just reject as much background as possible. Ambiguities between candidates particles are resolved and then an appropriate shape fitted to the resulting signal.

8.3.1 Cuts

In this sub-section the cuts used to enhance the K^0 signal (the K^0 cuts) are described along with the physical basis for each and the aim of the cut. The effect of each cut is shown using an invariant mass ($m_{\pi^+\pi^-}$) plot, with a discussion of any losses and checks against possible biases. This distribution also forms the basis of the cross-section measurement. For each cut four histograms are shown: two for real data, two for the Monte Carlo data. The open histogram is the $m_{\pi^+\pi^-}$ invariant mass spectrum output by the V^0 finder. The left shaded histogram shows the spectrum for K_S^0 candidates kept by the cut. The right shaded histogram shows the spectrum of those rejected.

Constraining the V^0 origin to the Interaction Point

Most Λ and K_S^0 are produced in fragmentation or from the decay of a short-lived hadron. This means that they come from very close to the interaction point. A cut on the d^0 (the $x - y$ distance of closest approach to the beam axis) and the z^0 of the V^0 is made at 2 cm and 5 cm respectively. The momentum vector of the V^0 is the combination of the two daughter tracks. This removes many fake V^0 s from random combinations and also eliminates a large proportion of those V^0 s originating from material interactions. The effect of the d^0 cut on real and Monte Carlo data is illustrated in figure 8.3. The major loss of K_S^0 s with this cut is for those produced from longer-lived particles or cascades of decays, such as $B \rightarrow D^* + X$ ($c\tau = 0.035$ cm), $D^* \rightarrow D^+ + Y$ ($c\tau$ small), $D^+ \rightarrow K^0 + Z$ ($c\tau = 0.032$ cm). A large boost factor will result in a production vertex for the K_S^0 away from the interaction point. However, this loss is tiny, as is shown by figure 8.3.

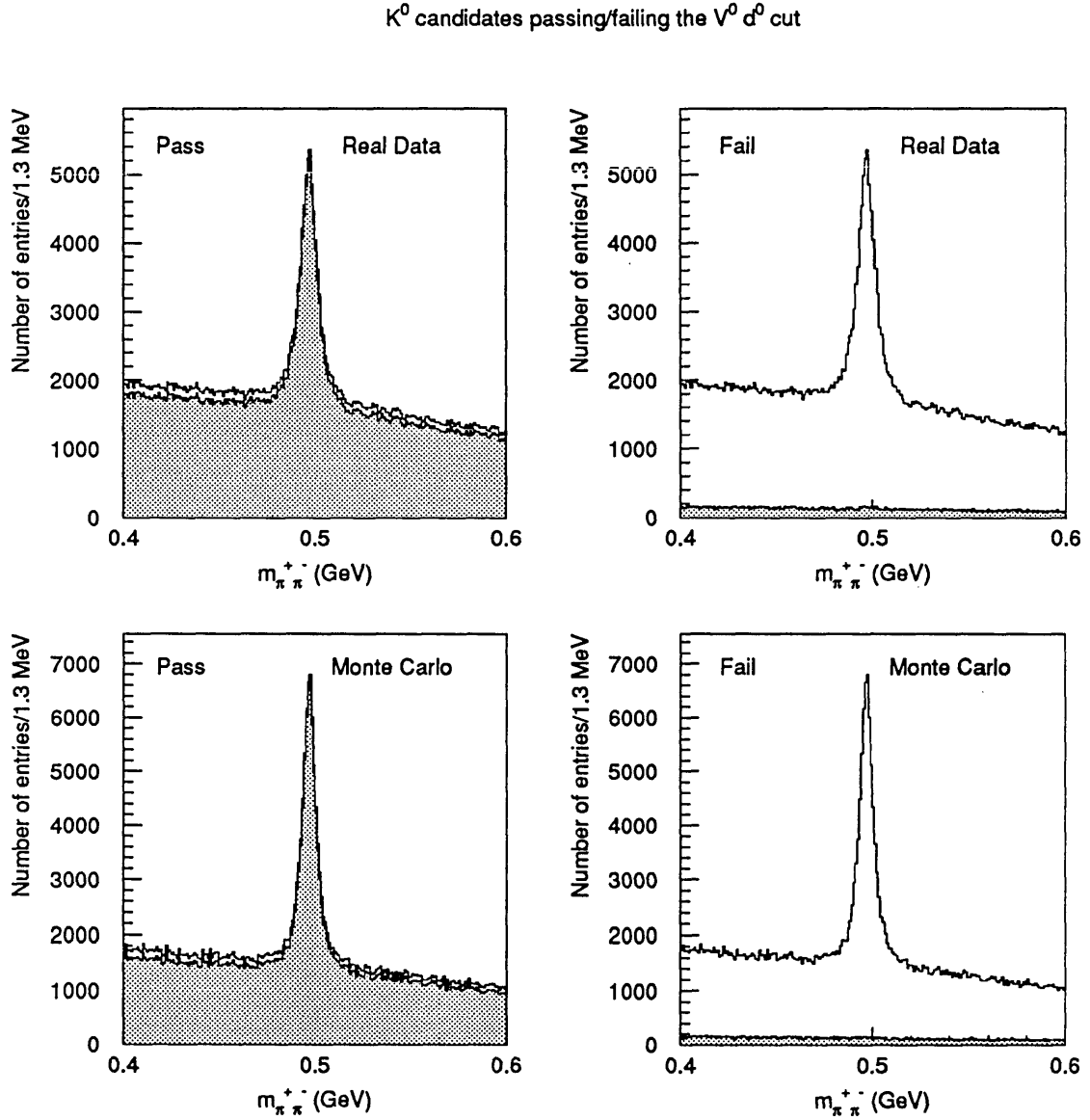


Figure 8.3: *The effect of the $V^0 d^0$ cut on Monte Carlo and real data: The shaded region indicates those candidates passing (failing) this cut. The open region represents the candidates passing the V^0 finding package cuts.*

If these decays are not well simulated, say the lifetimes are wrong, this will introduce a bias. So the d^0 cut is set at a value that is large relative to the known cts of these decays. A secondary problem is the error on the found interaction point — if this is large, the d^0 is badly defined. Again, the choice of 2 cm is set so as to be much larger than the error on the d^0 .

Secondary Vertex remote from Interaction Point: Decay length

The largest background in the sample is due to random combinations of charged tracks (π^+ , π^- , K^+ , K^- , p etc.) originating from fragmentation or decays with very short lifetimes. These can be largely removed by a cut on the distance of the V^0 decay vertex from the interaction point.

For this cut to introduce little bias, one requires that the distance is accurately measured, which involves knowledge of the interaction point plus the error on the decay vertex position. The K^0 lifetime in the simulation should also be correct, otherwise an explicit correction for efficiency in lifetime is necessary. Fortunately, the K^0 lifetime is well known, as is the decay process, and the lifetime measured in the data is well reproduced in the simulation (fig. 9.2). The choice of 2 cm is made in order to minimise the number of true K_S^0 s which are lost, whilst being sufficiently larger than the error on the distance. The effect of this cut is shown in figure 8.4.

Removing Λ and γ contamination — $\cos \theta^*$

Since gamma conversions have a very small opening angle it should be possible to eliminate most gamma background by a cut on a minimum opening angle. However, high p_T K_S^0 s also tend to have small opening angles thus a cut like this would severely reduce the efficiency for higher momentum kaons, where the sample is already limited by statistics. By considering the angular distribution of the K_S^0 decay in its rest frame, a cut that removes most Lambda and gamma background, without biasing the momentum spectrum, is possible. In the V^0 rest frame, θ^* is defined to be the angle between the positive decay track and the V^0 line of flight. Since the K_S^0 has zero spin, it decays isotropically in its rest frame. A uniform distribution in solid angle $d\Omega^* = \sin \theta^* d\phi^* d\theta^*$ corresponds to a flat differential distribution in $\cos \theta^*$. This is shown in figure 8.5, last plot, where the open histogram is the distribution for all V^0 candidates (no other cuts) and the shaded histograms are for the relevant particle types. For gamma conversions, the incorrect Lorentz boost of the electron and positron under the K_S^0 hypothesis results in peaks at $\cos \theta^* \simeq \pm 1$ (figure. 8.5, first plot).

Most importantly, the cut rejects most Λ contamination. The Q-value of the Λ decay is very small and since the proton is much heavier than the pion, Λ decays are asymmetric in the laboratory frame with the proton emitted forwards in the Λ direction. The Lorentz

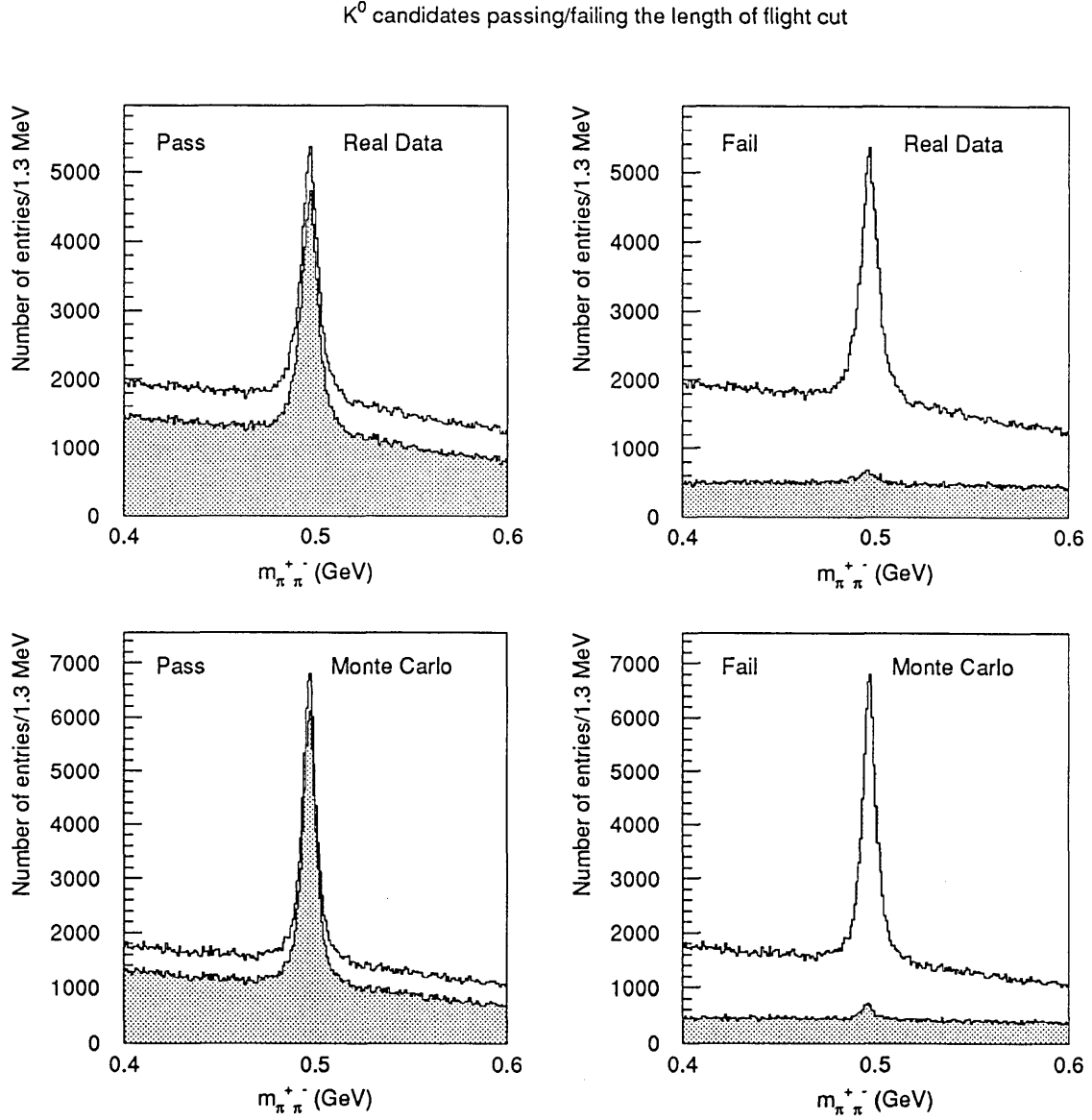


Figure 8.4: *The effect of the cut on K^0 decay length (real and Monte Carlo data): The shaded region indicates those candidates passing (failing) this cut. The open region represents the candidates passing the V^0 finding package cuts.*

The K^0 candidate $\cos\theta^*$ distributions for different particle types

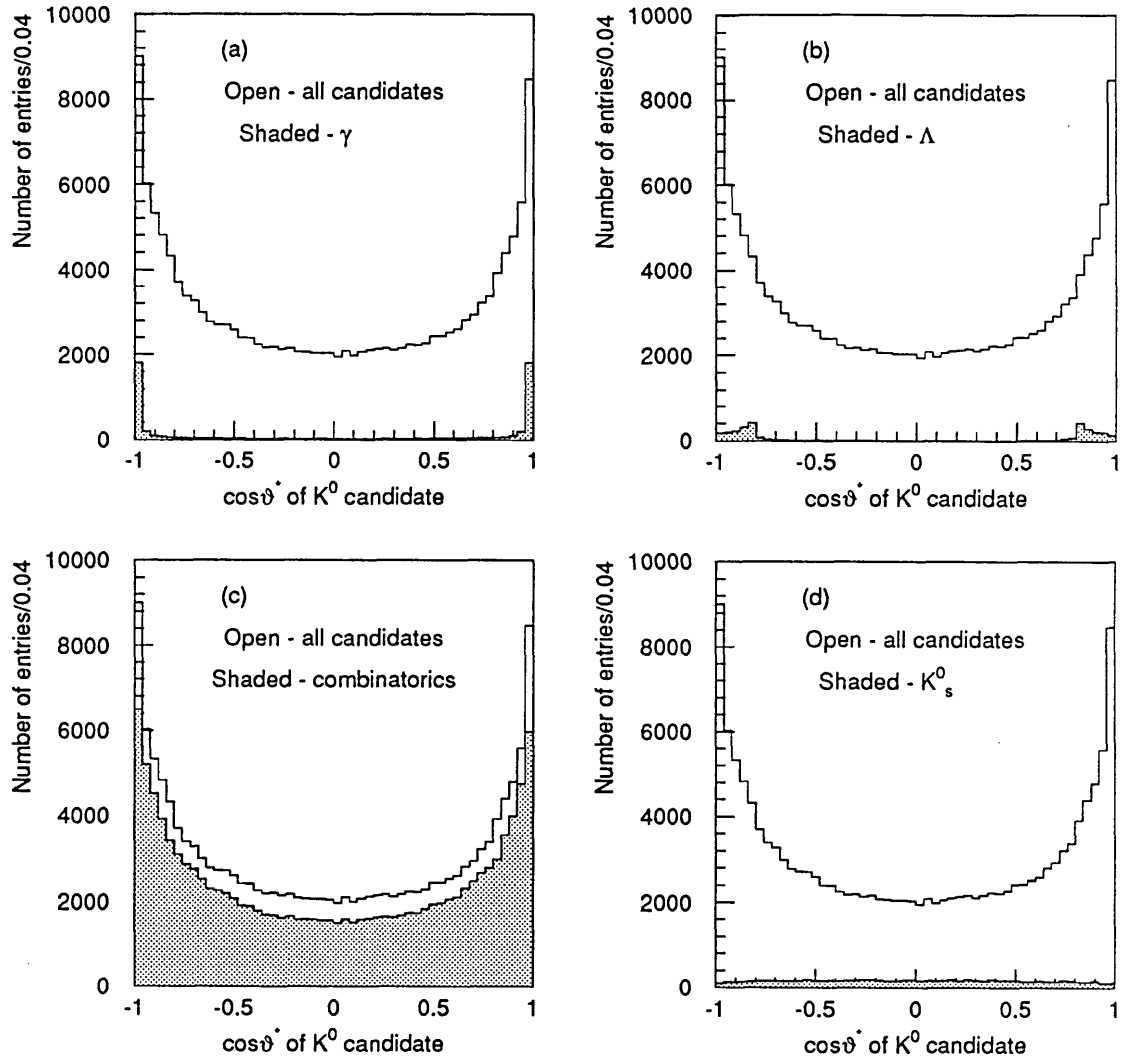


Figure 8.5: The $\cos\theta^*$ distributions for Monte Carlo data

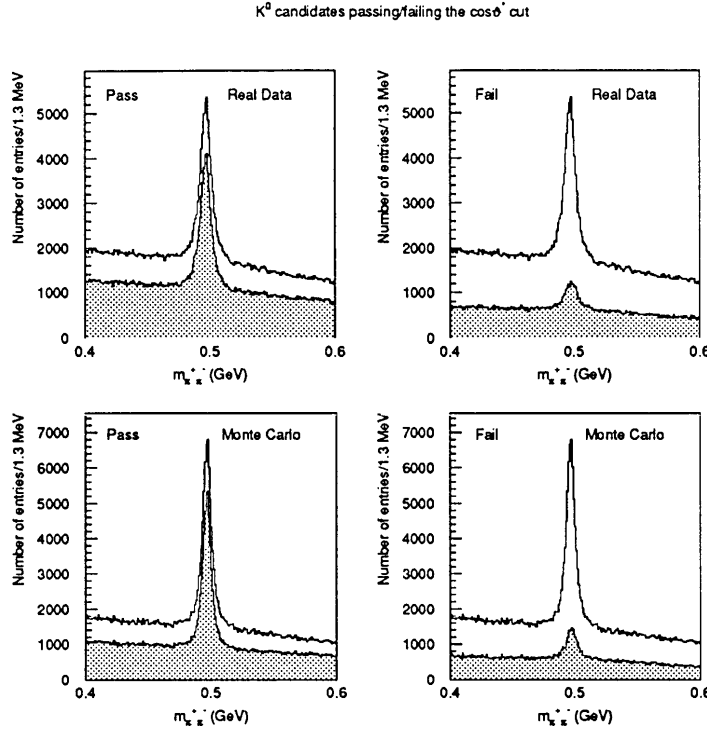


Figure 8.6: *The effect of the $|\cos\theta^*| < 0.75$ cut on the $m_{\pi^+\pi^-}$ distribution (real and Monte Carlo data)*

boost to the (assumed) K_S^0 rest frame, therefore results in the proton and pion tracks lying at small angles to the K_S^0 candidate line of flight. The $\cos\theta^*$ distribution thus gives peaks between $\pm 0.75 - 1.0$ (see figure 8.5, top right plot). This distribution is symmetric as the angle measured is with respect to the positive particle direction, which, for a Λ , is the proton 50% of the time only. The effectiveness of the cut for removing Λ s is seen in fig. 8.10.

The resulting distribution after all cuts, and corrections for background contamination, should be flat in $\cos\theta^*$ (figure 8.7). Losses at larger values of $|\cos\theta^*|$ are visible. These result from lower momentum K_S^0 decays where one of the daughters travels in nearly the opposite direction to the original K_S^0 . The daughter's momentum in the lab frame will be lower than that from a decay where the daughters travel perpendicular to the original K_S^0 direction. As the track finding efficiency is lower for low momentum particles and there is a minimum momentum requirement in the selection cuts, this is expected. The effect is reproduced by the Monte Carlo, and the efficiency corrected distribution is flat (fig. 8.8).

Opposite sign d^0

As any true K_S^0 with a finite lifetime will decay at a point away from the interaction point, the decay product tracks will not be seen to originate from the interaction point – they have $|d^0| > 0$. However, requiring these d^0 s to be larger than some cut rejects some from

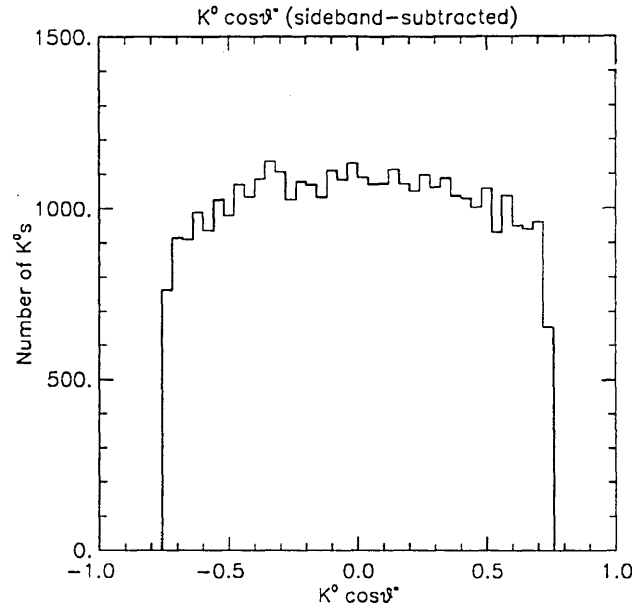


Figure 8.7: *The measured $\cos \theta^*$ distribution for K_S^0 s passing all cuts and corrected for background by sideband-subtraction (real data).*

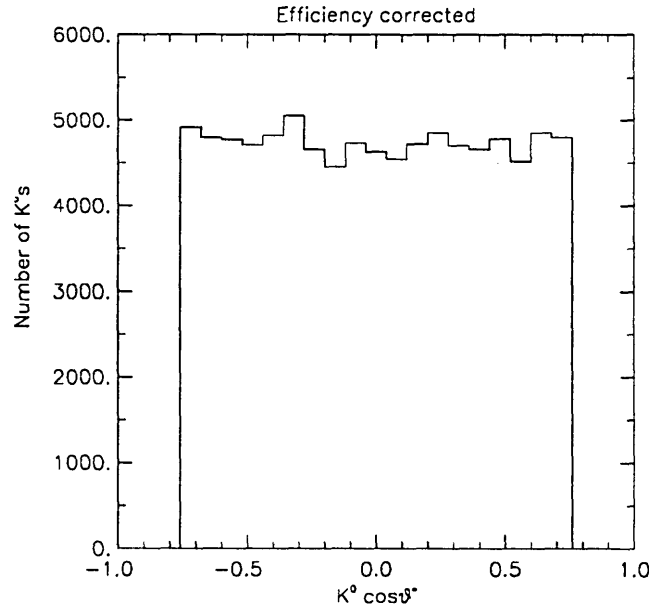
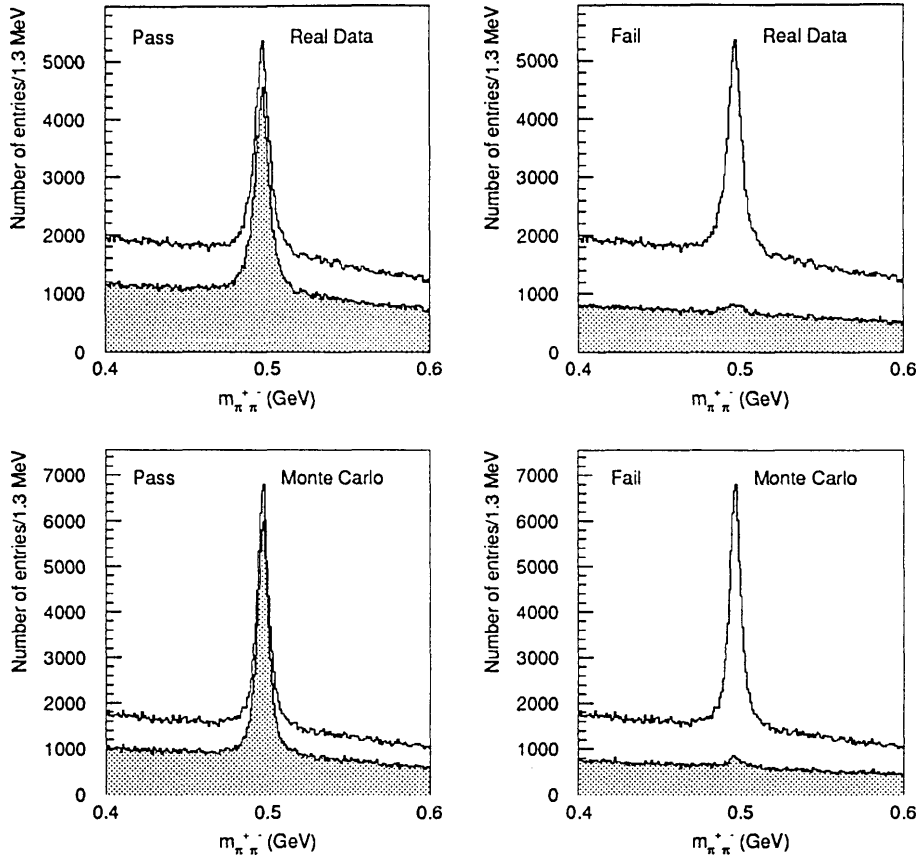


Figure 8.8: *The $\cos \theta^*$ distribution for K_S^0 s passing all cuts, corrected for background by sideband-subtraction and corrected for efficiency (real data).*

K^0 candidates passing/failing the opposite sign d^0 requirementFigure 8.9: The effect of the opposite sign d^0 of tracks cut

high momentum K_S^0 decays. By considering both daughter tracks and placing a minimal requirement on them, background candidates can be effectively removed with little loss in efficiency. The d^0 s of the daughter tracks are therefore required to be of opposite sign. This arises from the ALEPH track parameter sign convention (see figures 6.2 and 7.3). The cut can reject very short lifetime K_S^0 s with daughters having a mis-measured (displaced) d^0 , but these are lost also due to the decay length cut. For the same reason, more Λ decays are rejected, as the proton tends to point back to the interaction point. Figure 8.9 demonstrates that few K_S^0 are lost due to this cut, but the combinatoric background is markedly reduced.

8.3.2 K_S^0 / Λ Ambiguities

All the Λ s are effectively removed by the $\cos \theta^*$ cut. This can be seen by a plot of $m_{\pi^+\pi^-}$ vs. $m_{p\pi}$ before and after the $\cos \theta^*$ cut (fig. 8.10).

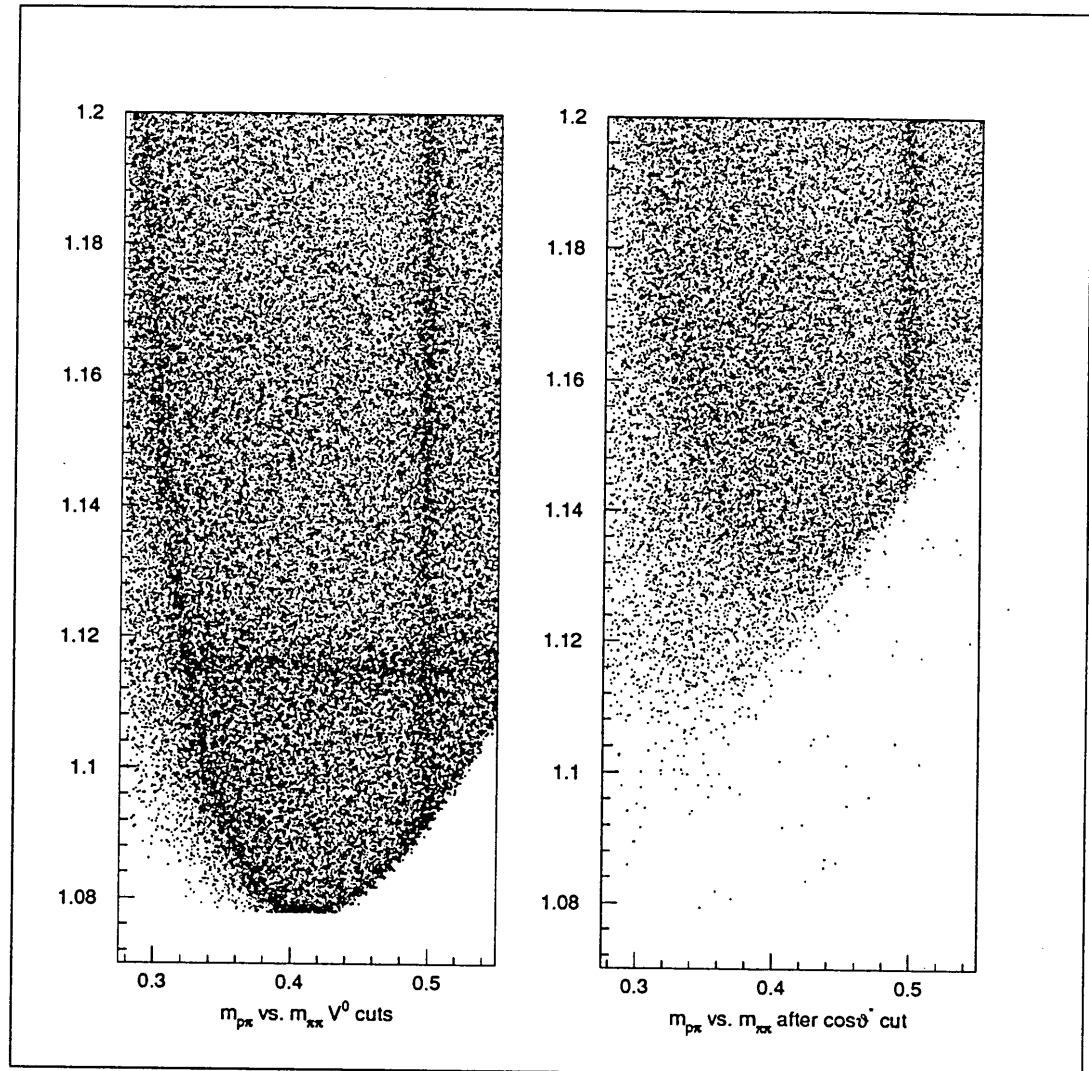


Figure 8.10: *The effect of a $|\cos\theta^*| < 0.75$ cut on the $m_{\pi^+\pi^-}$ vs. $m_{p\pi}$ distribution: only the low Λ mass part is shown (real data)*

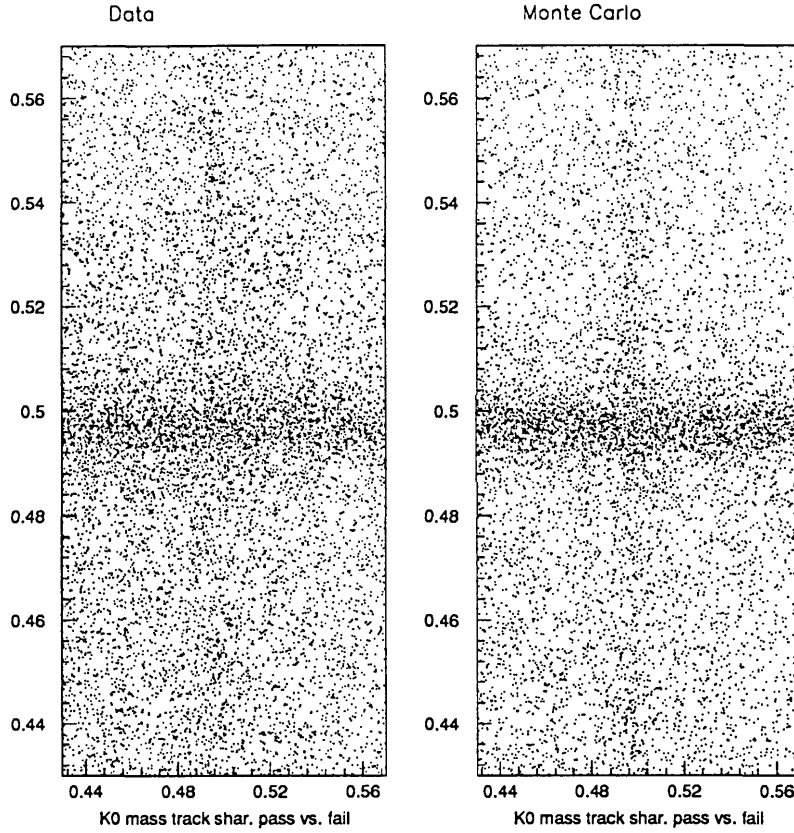


Figure 8.11: $m_{\pi^+\pi^-}$ vs. $m_{\pi^+\pi^-}$ for K_S^0 candidates sharing a track — real data and Monte Carlo compared

8.3.3 Track-sharing Ambiguities

Some K_S^0 candidates are found to share a daughter track with one or more other candidates.

These ambiguities are resolved by choosing the “best” candidate, using a cut based on the quality of the V^0 . The candidate with the smallest distance of closest approach between the daughter track trajectories was kept and the others rejected. The result of this requirement is illustrated in fig. 8.11.

8.3.4 Fitting

To extract the number of K_S^0 s from a sample still containing some fraction of background, a form of subtraction is necessary. The method used was to fit two functions, signal plus background, to the invariant mass spectrum, and use the resulting signal area as a measurement.

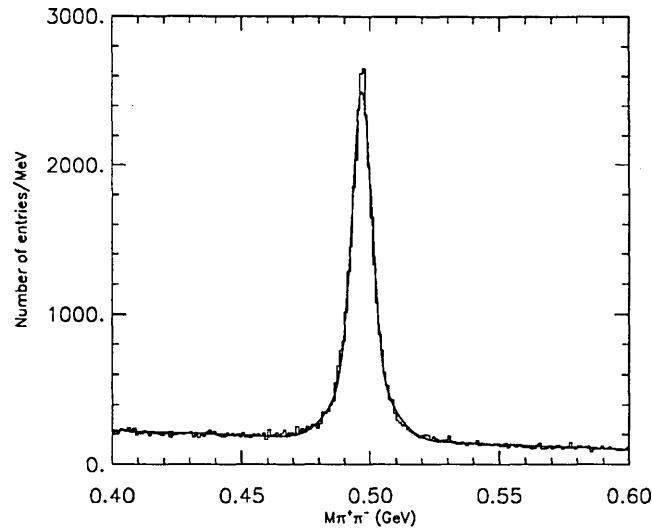


Figure 8.12: $m_{\pi^+\pi^-}$ invariant mass spectrum (GeV) and result of fit (real data)

The background was found to be well represented by a straight line, within the mass range of the fit. The signal was more complex: whilst at one momentum, the mass resolution is approximately gaussian, over even a small range of momentum this was found to be inadequate. Additionally, due to the usual presence of tails in the detector resolution function, a fitted gaussian would not include the whole signal. A double gaussian was therefore used. This is the sum of two gaussians, both constrained to the same central fitted value, with the relative areas allowed to vary in the fit. The width of one was constrained to be a fixed multiple of the other (2.3) to prevent the tails being artificially extended into the background distribution. An example fit to the complete sample is visible in figure 8.12. For very low statistics (i.e. the region $x > 0.5$ ¹) a single gaussian was adequate.

The results of fits were compared with those using a simple mass sideband-subtraction technique (§ 8.4.2) and with same-sign background subtraction (§ 8.4.1).

8.4 Checks for Biases

In order to gain confidence in the results and ensure the selection algorithm was performing as expected, a number of cross-checks were performed. These are explained in the following paragraphs.

¹Scaled energy of K^0 candidate, $x = 2E_{K^0}/\sqrt{s}$

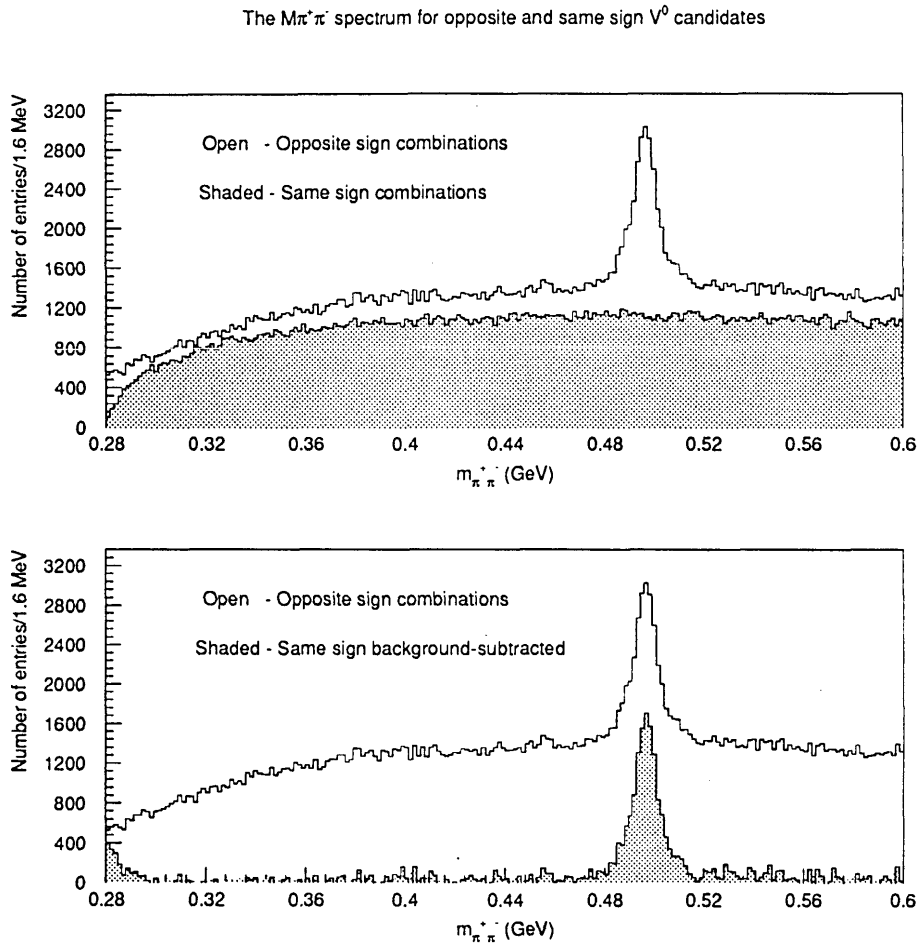


Figure 8.13: $m_{\pi^+\pi^-}$ opposite and same sign combinations using YTOP package (real data). The lower histogram shows the result of subtracting the same-sign distribution (with a weight factor of 1.2) from the opposite sign distribution

8.4.1 Topological Vertex Finding

The YTOP package, as mentioned earlier (§ 7.3.5), can make secondary vertex fits for both opposite and same sign track pairs. Comparing the two spectra (figure 8.13) gives an independent check of the background subtraction method. Provided the K_S^0 background is mainly combinatorial in nature, the background function described by the fit should be well-represented by the same sign background. The relative heights of the two histograms are different in part due to the available combinations for opposite and same sign pairs. For example, for 5 positively charged tracks and 5 negative, there are 25 V^0 combinations, 10 V^{++} and 10 V^{--} . This is one reason why these particular K^0 cuts were chosen — to attempt to reduce the known background to purely combinatorics. This should mean, in the absence of any narrow resonances very close to the K^0 mass, that the background is very smooth and looks like the same sign histogram.

The same sign mass spectrum also usefully demonstrates that the cuts do not impose any bias in the K^0 mass region that would affect the cross-section calculation.

8.4.2 Sideband Subtraction

The sideband-subtraction technique was used to correct for background in the lifetime and $\cos \theta^*$ distributions. As above, assuming the background remaining after cuts is essentially combinatorial, then the mass sidebands should give a very good representation of the shape of any other distribution due to this background. This shape can be subtracted from that produced under the K^0 mass peak giving, before efficiency corrections, a direct measure of the distribution. This technique is equivalent to fitting a linear background function over the limited mass range used for the sidebands ($m_{K^0} \pm 0.04 - 0.06 \text{ GeV}$).

8.4.3 Modification of Track Momentum Resolution

It can be seen from the invariant mass spectra, that the width of the K_S^0 mass peak is slightly wider in the data than in the simulation. This is due to some underestimation of tracking errors in the real data. To check that this discrepancy in the simulation does not affect the calculation of K_S^0 finding efficiencies, a sample of Monte Carlo data was processed through the V^0 finder after the track coordinates had been smeared with an additional small Gaussian resolution function. This resulted in a K_S^0 mass peak wider than that in the real data. The efficiency was calculated once more with this data, and found to change by only 0.8 % (table A.2), which is within the quoted systematic error.

8.5 Efficiency Corrections

8.5.1 Correcting the Inclusive Momentum Distribution

The fit procedure described in § 8.3.4 was applied to mass spectra for K_S^0 s separated into bins in x . This was repeated for Monte Carlo distributions, both with and without the detector simulation. An efficiency was then obtained for each x bin (fig. 8.14). The choice of bin width was mainly determined by available statistics and a desire to maximise the range of information, both to low and high x values. The efficiency can be seen to peak at around $x = 0.1 - 0.15$. At lower x values, it falls due to tracking losses. At high x , a larger fraction of K_S^0 s do not decay deep enough in the TPC to leave two good tracks, hence they are not seen. The mean efficiency was 34% when weighted with the expected momentum distribution.

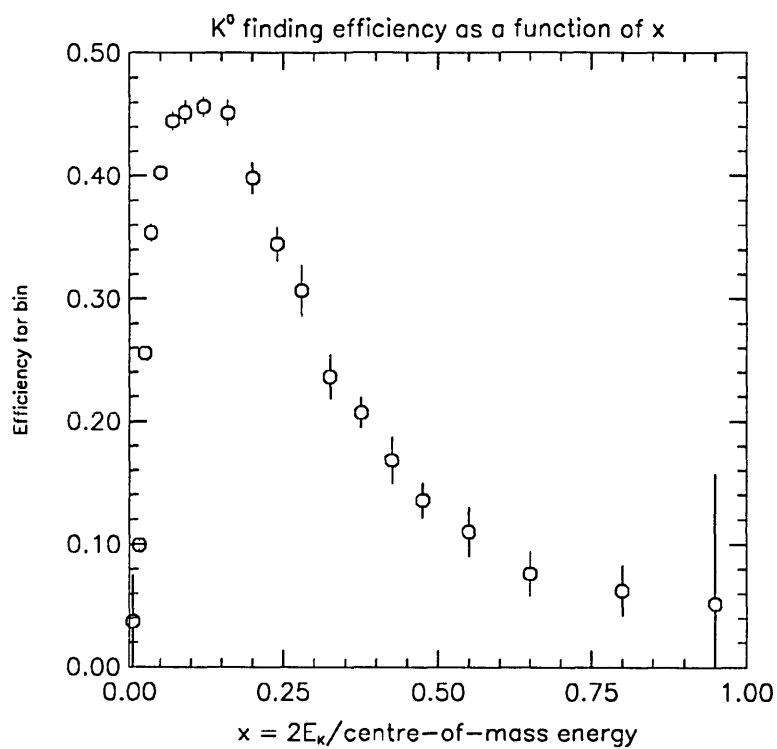


Figure 8.14: The $K_S^0 \rightarrow \pi^+\pi^-$ finding efficiency in x calculated from simulated data

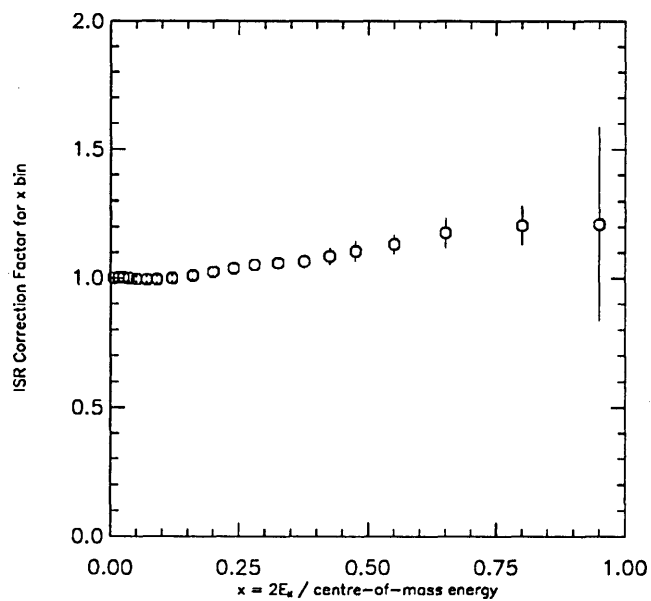


Figure 8.15: The correction curve for the effects of Initial State Radiation — from simulated data

The effect of initial state radiation on the process under study is to reduce the effective centre-of-mass energy of the e^+e^- system. This effect must be removed in order to give a measurement at the true centre-of-mass energy that can be used in comparisons at other energies. In order to obtain an accurate correction, the energy range of the events used in the analysis was restricted to 91.0 – 91.5 GeV. The correction factors for each x bin were obtained using the Monte Carlo. These were produced using only the event generator — i.e. no detector simulation — with the initial state radiation option switched off for one run. The resulting change in the mean K^0 multiplicity is an increase of 1 %. The correction factors are shown in figure 8.15, and are calculated as

$$y_j = \frac{N_j}{I_j} \quad (8.1)$$

where N_j is the number of K_S^0 s generated with no I.S.R. in the j^{th} x bin and I_j is the number of K_S^0 s generated with I.S.R. in the same x bin.

8.5.2 Correction for K_S^0 s produced in Material Interaction

The Monte Carlo predicts that 0.34 % of the total number of K_S^0 s produced originate from interactions in material (K^+ , K^- , Λ). 0.21 % of the total are estimated to come from recombination of K_L^0 into K_S^0 . After the selection cuts this impurity was found to be reduced to less than 0.1 % of the sample. This was smaller than the overall statistical error so no correction was done.

8.5.3 Corrections in Angular Distribution

The detector, whilst basically symmetric in ϕ , is not necessarily so in the polar angle θ . Any difference between the simulation and the data would require a correction. The two distributions are, however, compatible within statistical errors (fig. 8.16), and so no correction was needed.

8.5.4 Hadronic Event Selection Corrections

The background to the K^0 signal from non-hadronic events passing the standard hadronic selection cuts (§ 7.2.1) was estimated by running the analysis procedure on Monte Carlo $\tau^+\tau^-$ events that satisfy those cuts. The contamination of the hadronic event sample due to this source is estimated [60] at 0.26 %. For the events passed through the K^0 finding, only 0.003 K_S^0 s were found per τ event.

The two-photon background was estimated in the same way. Fewer than 0.06 K_S^0 s were reconstructed per event. Since the two-photon cross-section is much lower than the hadronic

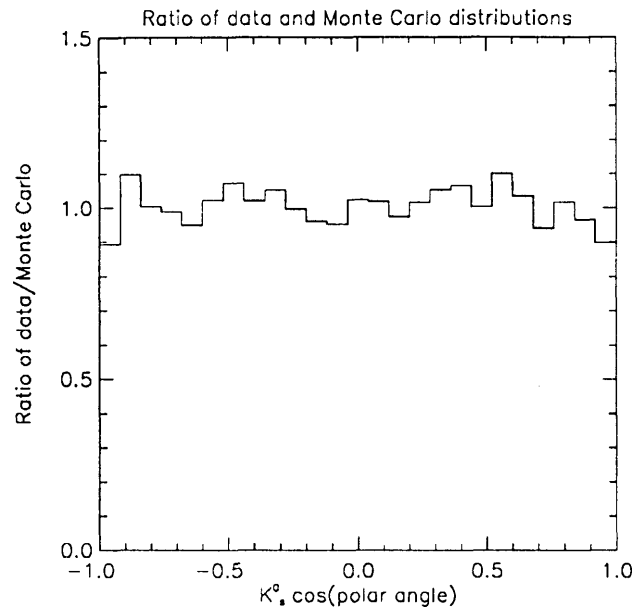


Figure 8.16: *Ratio of polar angle distributions for real and simulated data*

one, this background was negligible.

As a result the correction for these background source is made by subtracting the appropriate number of background events from the hadronic data sample. This results in a very small increase of 0.4 % in the K^0 cross-section (§ 9.4).

8.5.5 Correction for energy loss in the Detector Material

As charged particles traverse the detector region, they lose energy due to ionisation in the gas. The track fit is then biased towards a slightly smaller p_T [62, Three Dimensional Track Fitting, § 7.5]. Whilst this is simulated, there is some discrepancy between the result of the simulation and that expected by simple calculation [61, 71]. This can also be measured in the data. The effect is visible in the K_S^0 invariant mass spectrum as a shift in the central value of the peak (fig. 8.17). For high energy K_S^0 s, the change is negligible; for low energy K_S^0 s, it is clearly present. The result of this discrepancy between Monte Carlo and Data is to introduce an error in the calculation of the efficiency for the lower x bins. However, since the shift corresponds to an energy of up to about 4 MeV in the worst case, and is followed to within 1.5 MeV by the simulation (see fig. 8.18), i.e. less than 2 % of the lowest x bin width, the error is negligible. The correction was thus not needed ².

²The problem has since been attributed to the GEANT program, used by the ALEPH simulation. The relativistic rise in the dE/dx curve was neglected [71]

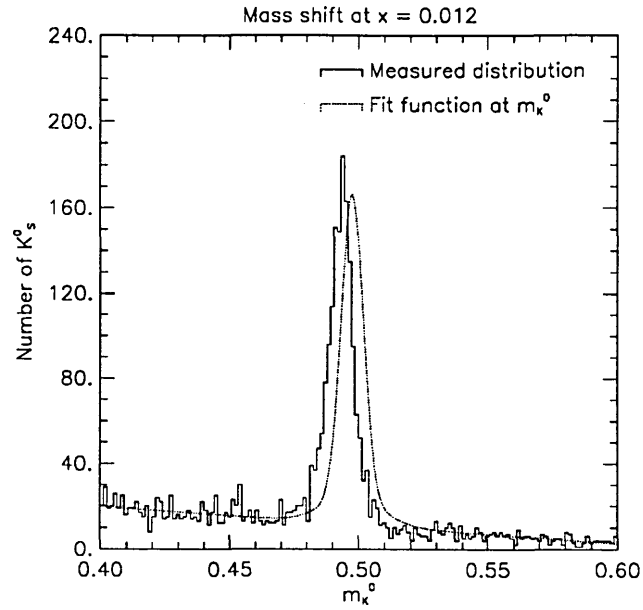


Figure 8.17: The shift in the measured K_S^0 mass peak at $x = 0.012$, or $p_{K_S^0} = 230$ MeV

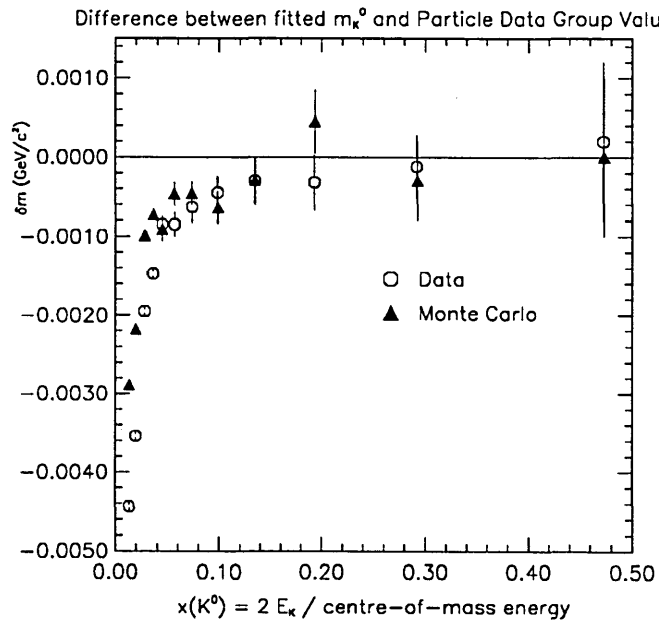


Figure 8.18: The difference between the fitted and the standard P.D.G. K^0 mass as a function of x (real and Monte Carlo data)

8.6 Systematic Errors

These can arise from three major sources:

1. Systematic problems in the data acquisition:

For example if high multiplicity events cause failures more often then a systematic bias is introduced.

2. Differences between the simulation and the real data:

For the real data, alignment and calibration must be performed, which for the Monte Carlo is automatically correct. For the Monte Carlo, the detector material and resolution may not be simulated correctly. Both these effects give rise to an error in the efficiency. The efficiency can be also be inaccurate if the production of K_S^0 s is not well simulated — for example effects can arise because of two track resolution limitations or track losses in jets due to the high hit density. If not corrected for explicitly, differences in the production of K_S^0 s can introduce a bias due to cuts on the quantities in question.

3. Fitting and signal extraction errors.

8.6.1 Systematic Errors due to Data Acquisition

There were a small number of events where some readout error occurred and, for example, either some data was lost or (rarely) misplaced by one event. The uncertainty was quantified by studying the trigger patterns of the failing events, and the events immediately before and after in the run. A check was also made by examining the events where only one of the two hadronic event triggers had been activated. The end result was a systematic error of 0.04% on the total number of hadronic events [60].

8.6.2 Systematic Errors due to Simulation Deficiencies

The tracking detector alignment and calibration has been checked in detail [4, 68] and found to give results very close to those predicted by the Monte Carlo.

The accuracy of the detector simulation has been checked in a number of studies [55, 57, 59, 69], in particular for charged track measurement by studying track residuals and balancing momentum in back-to-back di-muon events. The error from this source is included in that described in the V^0 finding section, § 7.3.4.

A bias in the x spectrum can result from a large difference in the momentum resolutions of the real data and simulation. This is for the following reason: K_S^0 s in momentum (or

x) bin m will be smeared by the resolution for that bin. As a result some will fall into the next highest bin, $m + 1$. A similar effect for this next bin will result in some K_S^0 s being found in the lower bin. If the distribution is flat, the numbers only differ if the resolution is a strong function of momentum. If the x distribution is falling quickly, then more K_S^0 s will move from bin m to bin $m + 1$ than the reverse. This will result in an incorrect efficiency for bin $m + 1$ if the Monte Carlo does not closely match the Data. The detector resolution is, however, sufficiently high as to make this effect negligible (Appendix A.1).

The existing comparisons between Monte Carlo and Data [56, 57] used slightly different event selection cuts, including a sphericity axis requirement. The K^0 finding procedure was therefore repeated using these cuts. The overall change in K^0 multiplicity was only 0.4 % (table A.2, “QCD Group event selection cuts”). This is within the statistical error on the multiplicity, which is increased because the overall efficiency with these cuts is considerably reduced (to approximately 14 %). Since these comparisons demonstrated good agreement between Data and Monte Carlo in the area of jet variables, no explicit investigation into the effect of varying the event generator parameters was felt necessary. The effect of changing the s/u parameter in the ALEPH Monte Carlo, which determines the relative rates that s quark pairs are generated during fragmentation compared to u quarks, is, however, covered in § 9.5.

It is visible from figs. 8.3 – 8.9 that the background level in the Real Data is slightly higher than in the simulation. This could indicate a difference in overall K^0 finding efficiency between data and Monte Carlo. The application of these “QCD Group event selection cuts” reduced the data background to that of the Monte Carlo. Since the final number of K_S^0 s did not change significantly (previous paragraph), this was not a problem for the measurement of K_S^0 s. The source of the difference in background level seemed to be due to several factors, the most important of which are:

- The material of the detector, which affects the rate of photon pair production and thus gamma conversions into e^+e^- was at the time not fully included in the simulation [71]. These pair conversions are subtracted for the charge multiplicity comparisons in [56] and thus would not be visible. Whilst the number of extra tracks from this effect is quite small (about 0.5 tracks per event), the resulting contribution to the number of possible $+/-$ track combinations is large, because these tracks do not originate from the origin;
- The momentum resolution is slightly too good in the Monte Carlo. Since the momentum spectrum for tracks from conversions is rapidly falling this means more tracks that would fail the minimum momentum cut fall into the next higher bin. This again results in a slightly increased level of background.

The systematic error originating from the choice of cuts has been estimated by varying the cut limits and measuring the resultant change in multiplicity. Table A.2 in appendix A.2

shows the variation for different cuts. The final result was also re-calculated for 1989 data, when the material in the detector was reduced (less silicon in the vertex detector). This result is shown in the same table. A check was also made for the lowest, highest and middle x bins. There was no significant extra bias for these bins. The contribution to the systematic error from this source is 1.9 %.

8.6.3 Systematic Errors due to the Fitting Method

Fitting errors were checked by varying the background by one σ and looking at the resultant change on the number of K_S^0 s. This was found to be negligible compared to the statistical error of the fit. Also by comparing the overall total of reconstructed K_S^0 s to the sum of those in each x bin it was demonstrated that no systematic bias arose in the fit due to lower statistics.

8.6.4 Systematic Error Summary

The error in the measured hadronic cross-section (§ 7.2.1) must be included in a final absolute cross-section. Where the results are relative to the number of hadronic events it is only necessary to note that the measured values are assumed independent of the hadronic event selection efficiency. To check this, the simulated events that failed the hadronic selection (4094 of 156030) were studied to determine any systematic difference in $\langle N_{K_S^0} \rangle$. The events fell into two categories:

1. Many tracks in the event were confined to the low angle region, and thus were lost down the beam pipe;
2. A relatively large fraction of the available energy was taken by initial state radiation.

In the former case, no significant difference in $\langle N_{K_S^0} \rangle$ was found. For the latter case, the correction for initial state radiation (§ 8.5.1) is sufficient. The contribution to the overall systematic error from this source is 0.04 %.

A systematic error of 2 % is assigned due to the above-mentioned effects.

Overall we have 0.7 % (track losses, V^0 finding, V^0 cuts) + 2.0 % (selection cuts, alignment, simulation of detector resolution, background levels, fitting errors). These are assumed independent and combined in quadrature to give 2.1 % systematic error on the total number of K_S^0 s.

Chapter 9

Results, Discussion and Conclusions

9.1 Overall Efficiency and Signal Purity

The mean $K_S^0 \rightarrow \pi^+\pi^-$ finding efficiency was 34 %; the resulting purities, taken in a 20 *MeV* window about the fitted centre of the signal, are:

- real data: after V^0 cuts, 41 %. After K^0 cuts, 82 %;
- Monte Carlo: after V^0 cuts, 47 %. After K^0 cuts, 86 %.

The difference is attributable to two factors: (1) The background level in the data, as mentioned in § 8.6, is slightly higher. (2) The width of the K^0 peak is a little wider in the data, hence for the mass window used, more signal is lost in the data than the simulation.

9.2 Lifetime

The K_S^0 lifetime provides a check on the validity of the selection procedure. The K_S^0 lifetime τ is related to the decay length, l , by

$$l = \beta\gamma c\tau \tag{9.1}$$

where $\gamma = \frac{p}{m\beta}$ is the time-dilation factor. For convenience we use lifetime to refer to the quantity $c\tau$ hereon. The set of K_S^0 candidates passing the cuts still contains some background. To eliminate the systematic bias from this, only those K_S^0 candidates within a narrow (0.02 *GeV*) mass window are used, whilst those in the sidebands either side ($m_{K^0} \pm 0.04 - 0.06$ *GeV*) are used to give a prediction of the size and shape of the lifetime distribution of the background. Subtracting one from the other removes the background contribution.

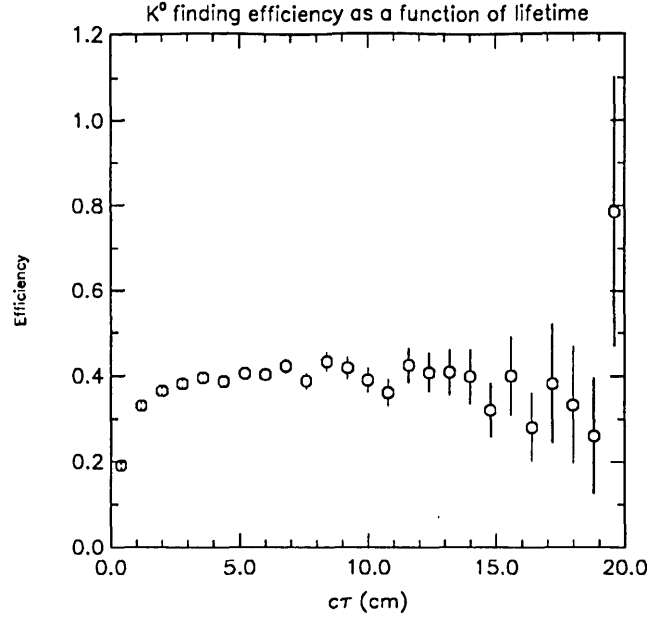


Figure 9.1: The $K_S^0 \rightarrow \pi^+\pi^-$ finding efficiency as a function of lifetime

The resulting distribution is then corrected bin-by-bin for efficiency. As the lifetime is not a function of momentum, the variation in efficiency with momentum is averaged over all the bins. The resulting correction is thus essentially flat (figure 9.1), except for very short and very long lifetimes where the minimum decay length cut and detector acceptance effects reduce the efficiency.

Figure 9.2 shows the final curves and fit results for real and simulated data. An exponential fit to the simulated data in the interval $2.5 \text{ cm} \leq c\tau \leq 20 \text{ cm}$ gave a result for $c\tau_{K_S^0}$ of $2.64 \pm 0.05 \text{ cm}$. This is in agreement with the value determined for real data of $2.63 \pm 0.05 \text{ cm}$. They are both consistent with the world average lifetime measurement of $2.675 \pm 0.006 \text{ cm}$ [50], demonstrating the validity of this analysis.

9.3 Extraction of Overall K_S^0 multiplicity from Lifetime distribution

The lifetime distribution can also be used to extract the average K^0 multiplicity. As the lifetime distribution is exponential:

$$N(l) = N_0 \exp\left(-\frac{l}{\beta\gamma c\tau_{K_S^0}}\right) \quad (9.2)$$

where $N(l)$ is the number of K_S^0 s remaining undecayed after a distance l . N_0 is the original number of K_S^0 present at $l = 0$. Using $y = l/\beta\gamma$ to remove the time dilation factor, the number decaying between y_2 and y_1 , $y_2 < y_1$ is:

$$N(y_2) - N(y_1) = N_0[\exp(-y_2/c\tau_{K_S^0}) - \exp(-y_1/c\tau_{K_S^0})] \quad (9.3)$$

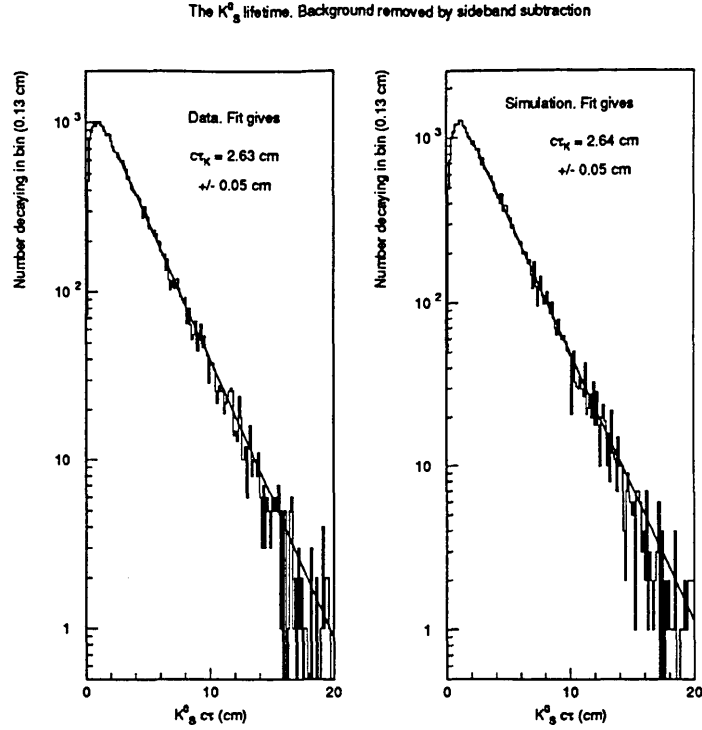


Figure 9.2: The fitted K^0 lifetime for simulated and real data

$$\Rightarrow N_0 = \frac{N(y_2) - N(y_1)}{[\exp(-y_2/c\tau_{K_S^0}) - \exp(-y_1/c\tau_{K_S^0})]} \quad (9.4)$$

This applies to the distributions after correction for efficiency.

Using the bins between $y = 3.0 \text{ cm}$ and $y = 20.0 \text{ cm}$, where the efficiency is approximately flat, and correcting for the loss of signal as the K_S^0 mass window is restricted to $\pm 20 \text{ MeV}$, gives a mean $K_S^0 \rightarrow \pi^+\pi^-$ multiplicity of $0.723 \pm 0.02 \pm 0.05$. The systematic error is due to the uncertainty in the efficiency, estimated by varying the level of background subtracted within some range. This range was chosen such that the resulting lifetime fit was within 1σ of the P.D.G. value.

9.4 Inclusive Scaled Energy Spectrum

The inclusive K_S^0 cross-section

$$\frac{1}{\sigma_H} \frac{d\sigma_{K^0}}{dx}$$

has been measured for the scaled energy region $0.011 < x < 1.0$, corresponding to the kinematic limits. The result is compared with the Monte Carlo prediction in figures 9.3 and 9.4¹. Fair qualitative agreement is seen over most of the x range, though the data is systematically lower in the small x region and higher for the larger x region ($x > 0.2$).

¹The data are listed in tables B.5 and B.6

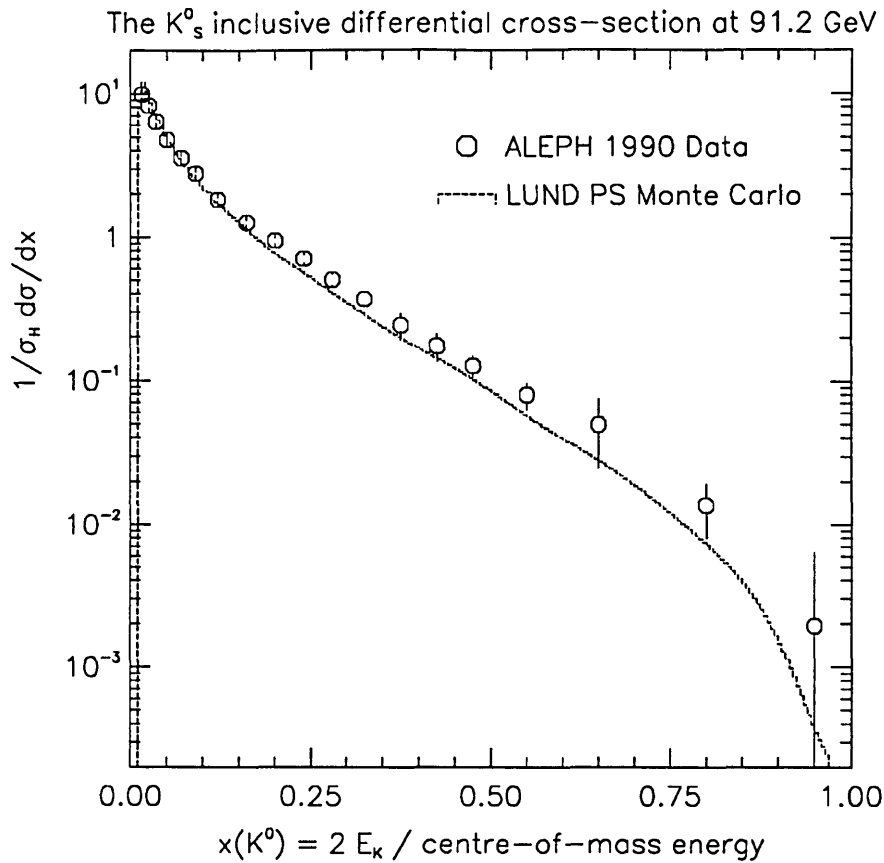


Figure 9.3: *The K^0 x distribution (errors are statistical only)*

The difference in the mid to high- x range can be accounted for by the larger overall K_S^0 multiplicity seen in real data (§ 9.5), i.e. by adjusting the Monte Carlo model slightly. In the x region below 0.035, however, the simulation and data differ by a margin significantly larger than their relevant statistical error. This can be interpreted in two ways: firstly, the Monte Carlo spectrum is artificially soft, and thus there is an indication that the model may not be accurate for this x region. Given that overall the indication seems to be that the model produces too few K_S^0 s, this explanation seems unlikely. Furthermore, at low x , the shape of the distribution is determined largely by phase space constraints, which one expects to be well modelled. Alternatively, there could be a systematically lower efficiency in real data for low momentum tracks. This signifies a deficiency in the detector simulation. Further study is required to determine the exact nature of the problem. For the purposes of this analysis, an estimate of the systematic error in the overall K_S^0 multiplicity from this effect is made by extrapolating the lowest three bins using the Monte Carlo prediction. Two values for $\langle N_{K_S^0} \rangle$ are therefore quoted in the next section. For the rest of this chapter, the original measured distribution is used.

The inclusive spectrum can be compared with results from other experiments at different centre-of-mass CM energies (table 9.1).

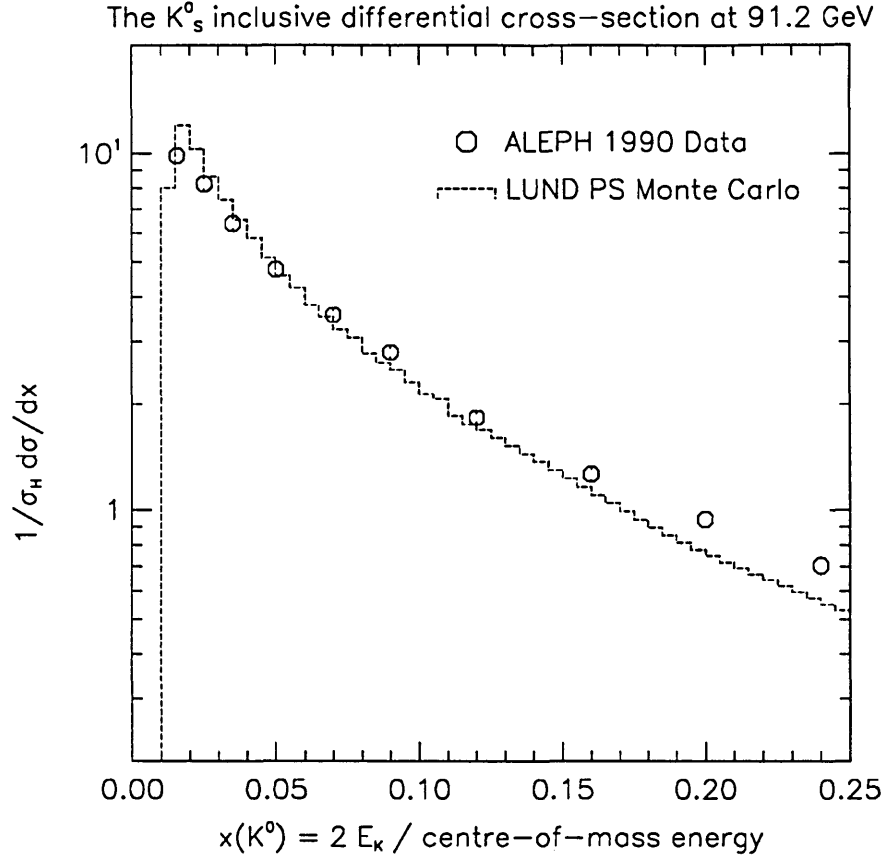


Figure 9.4: *The K^0 x distribution at low x values (errors are statistical only)*

Experiment Name	Accelerator	CM Energy Range	Reference
ARGUS	DORIS	9.5 GeV	[72]
CLEO	CESR	9.5 GeV	[73]
TPC	PEP	29 GeV	[74]
HRS	PEP	29 GeV	[75]
MARK II	PEP	29 GeV	[76]
PLUTO	DORIS, PETRA	9.1 – 31.6 GeV	[77]
JADE	PETRA	12 – 35 GeV	[78]
TASSO	PETRA	12 – 34 GeV	[79]
CELLO	PETRA	35 GeV	[80]
OPAL	LEP	91 GeV	[81]

Table 9.1: *References for other Centre-of-Mass Energy Experimental Results*

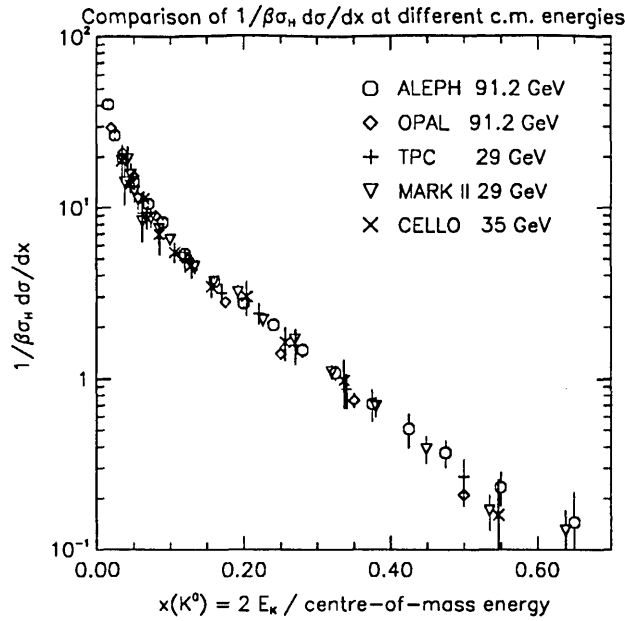


Figure 9.5: The K^0 x distribution including $1/\beta$ factor at different CM energies

For this purpose the Invariant Cross-Section (for all K^0 s)

$$\frac{1}{\beta\sigma_H} \frac{d\sigma_{e^+e^- \rightarrow Z^0 \rightarrow K^0}}{dx}$$

is plotted in figures 9.5–9.8. These show that the distribution does not alter very significantly with energy.

The only other result at LEP energies, that of OPAL, includes all energies, not just at the Z^0 peak, and lacks correction for initial state radiation. These two effects and statistical error account for the observed differences.

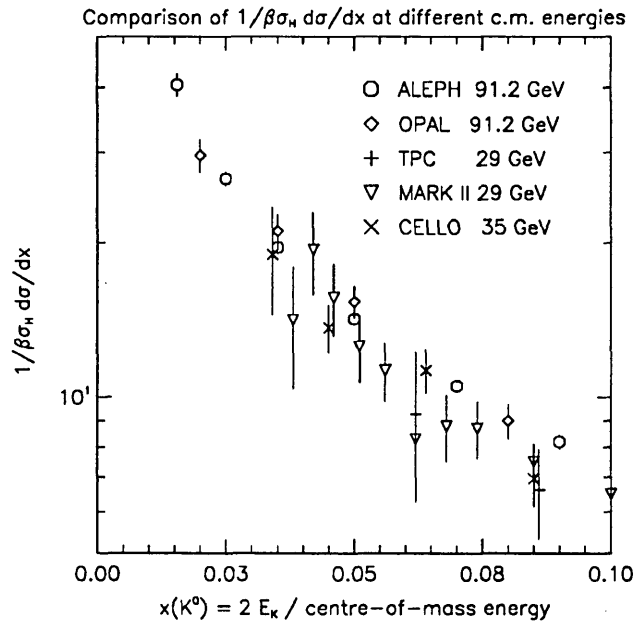


Figure 9.6: The K^0 x distribution including $1/\beta$ factor at different CM energies — low x values

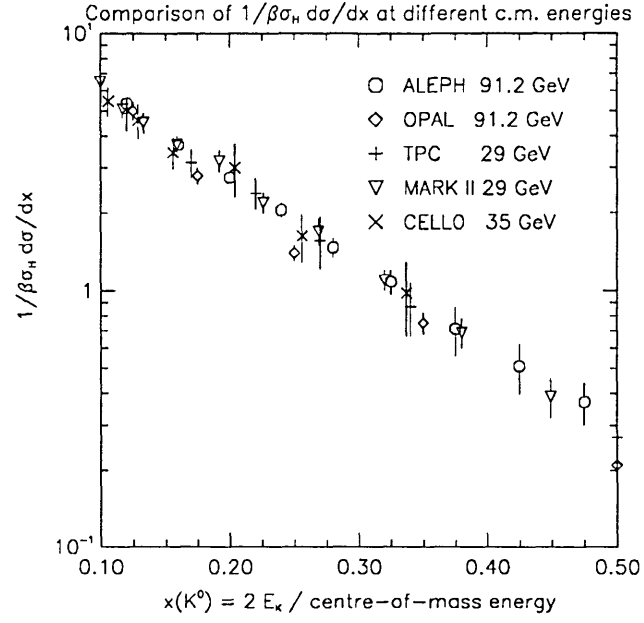


Figure 9.7: The K^0 x distribution including $1/\beta$ factor for various experiments — mid x values

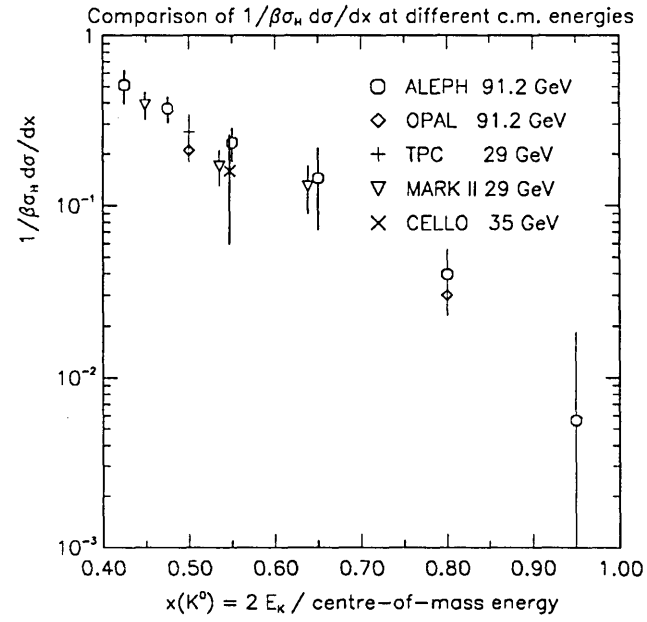


Figure 9.8: The K^0 x distribution including $1/\beta$ factor for various experiments — high x values

9.5 K^0 Multiplicity and K/h Ratio

By integrating the x spectrum one obtains the mean K_S^0 multiplicity per event. The systematic error is separated into that discussed in Chapters 7 and 8, and that due to extrapolation to low x using the Monte Carlo. This gives:

- $\langle N_{K_S^0} \rangle = 0.727 \pm 0.0085$ (stat.) ± 0.015 (syst.) ± 0.013 (extrapolation), or
- $\langle N_{K_S^0} \rangle = 0.734 \pm 0.0091$ (stat.) ± 0.015 (syst.) ± 0.013 (extrapolation) after correction for initial state radiation.

Alternatively, if the Monte Carlo is used to extrapolate the lowest three x bins, the result is:

- $\langle N_{K_S^0} \rangle = 0.751 \pm 0.0084$ (stat.) $\pm_{0.028}^{0.015}$ (syst.) $\pm_{0.013}^{0.0}$ (extrapolation), or
- $\langle N_{K_S^0} \rangle = 0.759 \pm 0.0089$ (stat.) $\pm_{0.028}^{0.015}$ (syst.) $\pm_{0.013}^{0.0}$ (extrapolation) after correction for initial state radiation.

These values can be compared with the predictions of the Monte Carlo of:

- $\langle N_{K_S^0} \rangle = 0.712 \pm 0.0018$ (stat.), or
- $\langle N_{K_S^0} \rangle = 0.718 \pm 0.0027$ (stat.) after correction for initial state radiation.

The plot in figure 9.9 shows how the ALEPH result compares to existing measurements at other energies. The K^0 multiplicity is shown assuming a branching ratio $K_S^0 \rightarrow \pi^+\pi^-$ of 68.6 % and a factor of two for K^0 and $\overline{K}^0$, which includes K_L^0 . It scales closely with the overall event particle multiplicity — the $\log s$ curve is superimposed to illustrate this. The curve is scaled to match the ALEPH measurement. It is seen to be higher than the points at lower energies. This is an indication that the K^0 x spectrum is dictated partly by K_S^0 s originating from primary quarks in $s\bar{s}$ events. Since at the Z^0 pole, there are approximately twice as many such events as at lower centre-of-mass energies, there is a net rise in the mean multiplicity. This effect is correctly included in the Monte Carlo model. These additional K_S^0 s would also be expected to contribute more in the high x region, thus the agreement seen in fig. 9.8 is perhaps surprising. However, for $x > 0.5$ the statistical errors are large and data points are lacking, thus no strong conclusion can be drawn.

The OPAL value of 2.1 ± 0.14 K^0 s per event, is compatible with the ALEPH result of $2.12 \pm 0.025 \pm 0.08$ (initial state radiation correction not included).

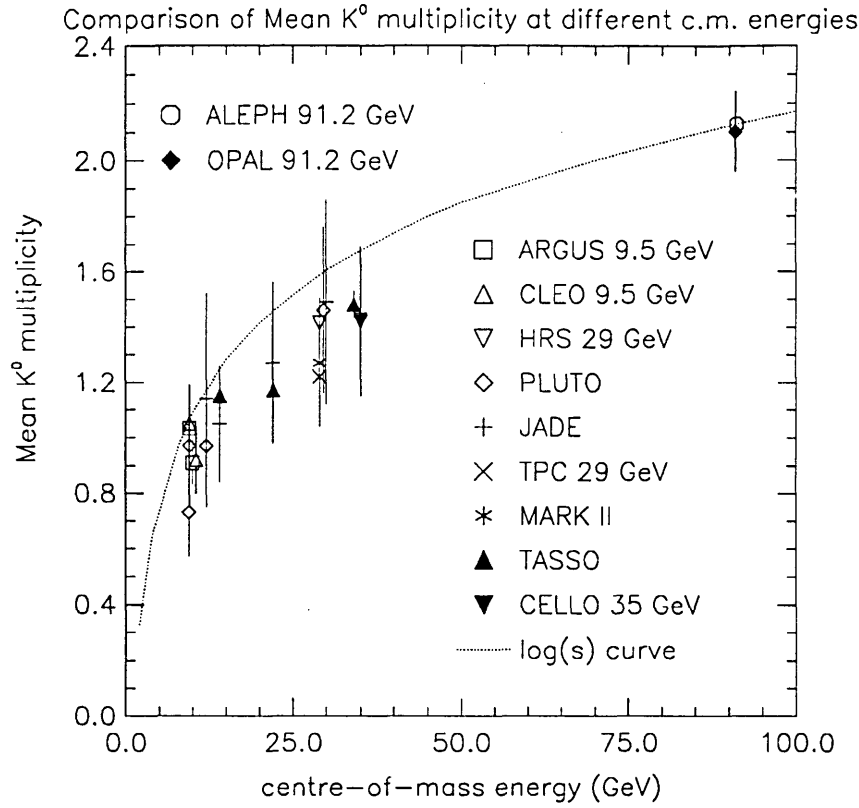


Figure 9.9: The mean K^0 multiplicity as a function of centre-of-mass energy

To investigate this scaling effect the ratio of K_S^0 multiplicity to the number of charged tracks can be calculated (K/h). Figure 9.10 shows this value as a function of x . Agreement with the Monte Carlo is reasonable at low x but the data is systematically higher at medium to large x . The value $N_{K_S^0}/N_{ch}$ approaches 0.28 at high x , corresponding to a limiting K^0 to charged track ratio of 0.56. The ratio is suppressed at low x due to the heavier mass of the strange quark.

The value of the K_S^0 multiplicity can be used to give limits for the Lund Monte Carlo simulation s/u parameter [55]. This number is the relative probability of extracting an $s\bar{s}$ quark pair from the vacuum during hadronisation compared to that of a $u\bar{u}$ pair. The graph in fig. 9.11 illustrates how the K_S^0 production rate varies with s/u . It can be seen that the error of $\pm 4\%$ on the mean K_S^0 multiplicity restricts the s/u value to a range of 0.305 – 0.354 (including systematic error), with a central value of 0.326. The value for s/u used in the simulation was 0.30. This is compatible with the result within the systematic error. There is an indication, however, that the value is too small, and should be set nearer to 0.325 (see fig. 9.11). This tendency has also been evident in previous experimental results [74, 79, 82].

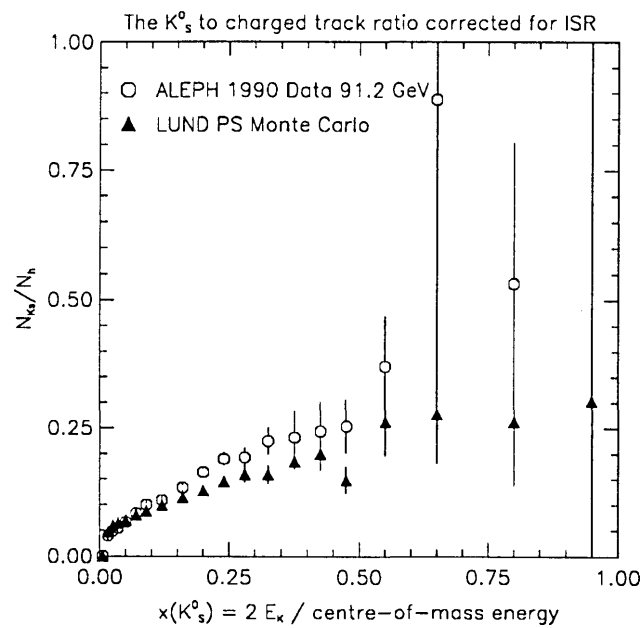


Figure 9.10: The K_S^0 /charged track ratio as a function of x , all K_S^0 but no K_L^0 .

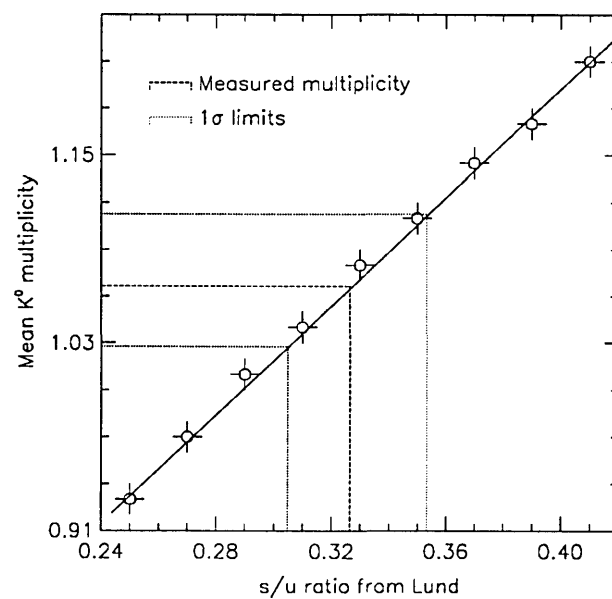


Figure 9.11: The mean K_S^0 multiplicity as a function of the Lund s/u parameter

9.6 Inclusive Cross-Section

By using the measured ALEPH luminosity recorded for the data sample used in this analysis, the peak cross-section for the process $e^+e^- \rightarrow Z^0 \rightarrow X + K_S^0 \rightarrow \pi^+\pi^-$ can be calculated. However, the data has been taken over an energy range $91.0 < \sqrt{s} < 91.5 \text{ GeV}$ and not solely at the Z^0 peak. The mean value obtained therefore must be corrected according to the shape of the peak and the different luminosity at each energy point to give a measurement at the peak. For this, it is assumed that the K_S^0 cross-section scales with the hadronic cross-section, which at the peak is $41.68 \pm 0.66 \text{ nb}$ [83] after correction for initial state radiation. The ratio of the observed peak value to mean value in the energy range $91.0 - 91.5 \text{ GeV}$ is 1.015. This gives

$$\sigma(e^+e^- \rightarrow Z^0 \rightarrow X + K_S^0 \rightarrow \pi^+\pi^-)_{\text{PEAK}} = 22.39 \pm 0.28 \pm 0.82 \text{ nb}$$

If the standard branching ratio for K_S^0 is assumed, and a factor two for K_S^0 and K_L^0 , then the total K^0 inclusive cross-section at the Z^0 pole,

$$\sigma(e^+e^- \rightarrow Z^0 \rightarrow X + K^0)$$

is $65.27 \pm 0.82 \pm 2.4 \text{ nb}$.

9.7 Further Work

With more time many interesting aspects of the K_S^0 system could have been studied. More work is needed, in combination with studies of the $\Lambda \rightarrow p\pi$ system, to provide information on correlations between strange particles. This may give insight into the fragmentation process.

With very accurate vertex and track information it is conceivably possible to reconstruct heavy flavour decays and thus measure the relative rates of the processes $decay \rightarrow K_S^0$ and $hadronisation \rightarrow K_S^0$.

By using the selected K_S^0 sample, and combining the reconstructed particles with pion tracks, a measurement of the $K^{*\pm}$ inclusive cross-section is possible. From this one can deduce the Vector to Pseudo-Scalar ($\frac{P}{P+V}$) production rates, and provide further constraints on the Monte Carlo model.

Since the magnitude of the hadronic cross-section around the Z^0 pole changes by a large amount, the variation of the K_S^0 cross-section with centre-of-mass energy in this region should be checked to scale accordingly. Away from the Z^0 pole, the enhancement in $s\bar{s}$ events should be removed and the mean K^0 multiplicity as plotted in fig. 9.9 is expected to lie much closer to the extrapolated lower energy measurements.

9.8 Conclusions

This analysis has provided the most accurate measurement of the K_S^0 inclusive cross-section at centre-of-mass energies near the Z^0 pole. As a result, the first constraint on the Lund model, with respect to strangeness production at this energy, has been produced.

Appendix A

Systematic Error Checks

A.1 Momentum Resolution — The Effects on the x distribution

The following table shows the resolution in x for each bin calculated using a momentum resolution of $\frac{\Delta p}{p^2} = 0.0008 \text{ (GeV/c)}^{-1}$ [4]. It can be seen that the resolution only becomes significant (i.e. more than 10 % of the bin width) for $x \geq 0.375$.

For the $x = 0.375$ bin, there are approximately 400 entries. For the next bin, 200 entries. Within σ_x of the boundary there are 24 entries on the left side and 17 on the other. This gives a worst case imbalance of 4 entries, i.e. 4/200 or 2 %. Compared to the statistical error of the bin, this is negligible.

x bin centre	bin width	resolution (σ_x)	x bin centre	bin width	resolution (σ_x)
0.0155	0.009	0.000004	0.280	0.04	0.003
0.025	0.01	0.00002	0.325	0.05	0.004
0.035	0.01	0.00004	0.375	0.05	0.005
0.050	0.02	0.00009	0.425	0.05	0.007
0.070	0.02	0.0002	0.475	0.05	0.008
0.090	0.02	0.0003	0.55	0.1	0.010
0.120	0.04	0.0005	0.65	0.1	0.015
0.160	0.04	0.0009	0.80	0.2	0.024
0.200	0.04	0.0014	0.95	0.1	0.033
0.240	0.04	0.002			

Table A.1: *The mean x resolution for each x bin*

Cut set imposed	Resulting $\langle N_{K_S^0 \rightarrow \pi^+\pi^-} \rangle$ Multiplicity
Standard — 1989 Data and Monte Carlo	0.719
Standard — 1990 Data and Monte Carlo	0.720
Lifetime ($c\tau$) > 2 cm	0.714
K_S^0 Decay Length > 5 cm	0.724
K_S^0 Decay Length > 0 cm	0.715
K_S^0 $d^0 < 5$ cm	0.722
K_S^0 $d^0 < 0.5$ cm	0.719
K_S^0 $\cos \theta^* < 0.5$	0.722
No K_S^0 $\cos \theta^*$ cut	0.734
Daughter track $d^0 > 1$ cm	0.709
No opposite sign d^0 cut	0.722
Vertex fit $\chi^2 < 8.0$	0.727
Daughter track $p > 0.3$ GeV	0.732
No daughter track p cut	0.730
Daughter track $\cos \theta < 0.8$	0.725
Daughter track $\cos \theta < 1.0$	0.729
> 4 coordinates on each daughter track	0.711
V^0 pointing angle cut $\cos \alpha > 0.75$	0.712
QCD Group event selection cuts	0.723
Momentum resolution in simulation widened	0.726

Table A.2: The change in measured K_S^0 multiplicity for different cut values

A.2 Checking the Effects of Cut Variations

The “standard” set of cuts are as follows:

- The “ V^0 ” cuts from § 7.3.3, table 7.2;
- K_S^0 $d^0 < 2$ cm;
- K_S^0 Decay Length > 2 cm;
- K_S^0 $|\cos \theta^*| < 0.75$;
- Daughter track d^0 s have opposite signs.

The largest excursion from the standard cuts result is 0.014 or 1.9 %. This is included in the systematic error quoted in § 8.6.

N.B. These values were not explicitly corrected for efficiency in x and therefore differ slightly from the final quoted result. The Lifetime cut and Vertex fit χ^2 cut were not used in the final analysis but are included in this table for completeness. The slight increase in cross-section with no $\cos \theta^*$ cut is due to the Lambda contamination and is reflected by the Monte Carlo.

Additionally, using the Monte Carlo to extrapolate the lowest three x bins results in an increase of $0.025 K_S^0$ s per event. This is treated as an asymmetric systematic error if the extrapolation is used — see table B.7.

Appendix B

Tables of Results

Included here are tables for all the relevant results presented graphically in the latter four chapters.

B.1 K^0 Finding Efficiency

B.2 Measured x Bin Contents

x bin	Bin centre	$\langle x \rangle$	Efficiency	Error
0.011 – 0.02	0.0155	0.016	0.0995	0.0029
0.02 – 0.03	0.025	0.025	0.2555	0.0048
0.03 – 0.04	0.035	0.035	0.3538	0.0067
0.04 – 0.06	0.050	0.050	0.4024	0.0061
0.06 – 0.08	0.070	0.070	0.4449	0.0077
0.08 – 0.10	0.090	0.090	0.4520	0.0091
0.10 – 0.14	0.120	0.120	0.4566	0.0078
0.14 – 0.18	0.160	0.160	0.4515	0.0103
0.18 – 0.22	0.200	0.199	0.3982	0.0124
0.22 – 0.26	0.240	0.239	0.3448	0.0139
0.26 – 0.30	0.280	0.279	0.3067	0.0208
0.30 – 0.35	0.325	0.323	0.2362	0.0182
0.35 – 0.40	0.375	0.373	0.2072	0.0121
0.40 – 0.45	0.425	0.424	0.1684	0.0192
0.45 – 0.50	0.475	0.473	0.1358	0.0141
0.50 – 0.60	0.550	0.544	0.110	0.020
0.60 – 0.70	0.650	0.645	0.076	0.018
0.70 – 0.90	0.800	0.761	0.063	0.020
0.90 – 1.00	0.950	0.923	0.052	0.11

Table B.1: *The K_S^0 finding efficiency computed from simulated data*

x bin centre	Measured Contents	Error	x bin centre	Measured Contents	Error
0.0155	1061	39	0.28	709	37
0.025	2531	61	0.325	500	34
0.035	2725	64	0.375	287	59
0.05	4651	74	0.425	164	31
0.07	3850	70	0.475	94	14
0.09	3059	63	0.55	94	12
0.12	4035	68	0.65	39	17
0.16	2721	62	0.80	17	4
0.20	1767	51	0.95	1	1
0.24	1131	45			

Table B.2: *The measured K_S^0 signal in x for real data — raw bin contents*

x bin centre	Measured Contents	Error	x bin centre	Measured Contents	Error
0.0155	1514	44	0.28	771	51
0.025	3651	66	0.325	497	38
0.035	3777	64	0.375	320	17
0.05	5983	81	0.425	182	20
0.07	4667	71	0.475	75	10
0.09	3432	61	0.55	104	17
0.12	4824	73	0.65	30	7
0.16	3105	63	0.80	12	4
0.20	1858	53	0.95	0.5	0.5
0.24	1182	45			

Table B.3: *The measured K_S^0 signal in x for Monte Carlo data — raw bin contents*

B.3 Correction for Initial State Radiation

B.4 K^0 x Spectrum after all Corrections

The extrapolation of the lowest three data x bins using the Monte Carlo gives the values shown in table B.7.

B.5 K_S^0 to Charged track ratio in x

x bin centre	Correction Factor	Error	x bin centre	Correction Factor	Error
0.0155	1.002	0.0077	0.280	1.051	0.019
0.025	1.002	0.0079	0.325	1.058	0.022
0.035	1.000	0.0093	0.375	1.064	0.026
0.050	0.996	0.0077	0.425	1.084	0.032
0.070	0.994	0.0092	0.475	1.10	0.039
0.090	0.995	0.011	0.550	1.13	0.037
0.120	0.999	0.0092	0.650	1.18	0.057
0.160	1.009	0.011	0.800	1.21	0.076
0.200	1.024	0.014	0.950	1.21	0.38
0.240	1.037	0.017			

Table B.4: Monte Carlo computed correction factors for initial state radiation

x bin centre	$\frac{1}{\sigma_H} \frac{d\sigma_{K_S^0}}{dx}$	Error	x bin centre	$\frac{1}{\sigma_H} \frac{d\sigma_{K_S^0}}{dx}$	Error
0.0155	9.92	0.47	0.28	0.507	0.044
0.025	8.28	0.26	0.325	0.375	0.039
0.035	6.43	0.20	0.375	0.249	0.052
0.05	4.81	0.11	0.425	0.176	0.039
0.07	3.59	0.095	0.475	0.128	0.023
0.09	2.81	0.086	0.55	0.081	0.017
0.12	1.84	0.047	0.65	0.050	0.025
0.16	1.27	0.043	0.80	0.014	0.0055
0.20	0.949	0.042	0.95	0.0019	0.0044
0.24	0.710	0.041			

Table B.5: x Spectrum for process $K_S^0 \rightarrow \pi^+\pi^-$ — real data

x bin centre	$\frac{1}{\sigma_H} \frac{d\sigma_{K_S^0}}{dx}$	Error	x bin centre	$\frac{1}{\sigma_H} \frac{d\sigma_{K_S^0}}{dx}$	Error
0.0155	11.15	0.11	0.280	0.435	0.011
0.025	9.43	0.10	0.325	0.302	0.008
0.035	7.02	0.086	0.375	0.216	0.007
0.050	4.87	0.050	0.425	0.154	0.006
0.070	3.43	0.042	0.475	0.106	0.005
0.090	2.49	0.036	0.550	0.0639	0.0028
0.120	1.74	0.021	0.650	0.0305	0.0020
0.160	1.13	0.017	0.800	0.0079	0.0007
0.200	0.787	0.014	0.950	0.00077	0.00031
0.240	0.585	0.013			

Table B.6: x Spectrum for process $K_S^0 \rightarrow \pi^+\pi^-$ — Monte Carlo

x bin centre	$\frac{1}{\sigma_H} \frac{d\sigma_{K_S^0}}{dx}$	Error
0.0155	11.00	0.11
0.025	9.29	0.10
0.035	6.93	0.08

Table B.7: Extrapolated lowest three bins of x Spectrum for process $K_S^0 \rightarrow \pi^+\pi^-$ — real data

x bin centre	$N_{K_S^0}/N_{ch}$	Error	x bin centre	$N_{K_S^0}/N_{ch}$	Error
0.0155	0.040	0.002	0.28	0.193	0.019
0.025	0.049	0.002	0.325	0.22	0.03
0.035	0.055	0.002	0.375	0.23	0.05
0.05	0.066	0.002	0.425	0.24	0.06
0.07	0.083	0.002	0.475	0.25	0.05
0.09	0.100	0.003	0.55	0.37	0.1
0.12	0.109	0.003	0.65	0.89	0.6
0.16	0.133	0.005	0.80	0.5	0.3
0.20	0.164	0.008	0.95	0.0	20000.0
0.24	0.190	0.013			

Table B.8: K_S^0 to charged track ratio, $N_{K_S^0}/N_{ch}$ in x — real data

x bin centre	$N_{K_S^0}/N_{ch}$	Error	x bin centre	$N_{K_S^0}/N_{ch}$	Error
0.0155	0.047	0.002	0.28	0.159	0.016
0.025	0.059	0.002	0.325	0.159	0.018
0.035	0.064	0.002	0.375	0.184	0.016
0.05	0.069	0.002	0.425	0.20	0.03
0.07	0.080	0.002	0.475	0.15	0.03
0.09	0.088	0.003	0.55	0.26	0.07
0.12	0.099	0.003	0.65	0.28	0.10
0.16	0.114	0.004	0.80	0.26	0.13
0.20	0.127	0.006	0.95	0.3	0.9
0.24	0.146	0.009			

Table B.9: K_S^0 to charged track ratio, $N_{K_S^0}/N_{ch}$ in x — Monte Carlo

Appendix C

Glossary

AEB An Aleph Event Builder - a general purpose Fastbus master. See also MEB, LEB, OS9, [8].

ALONL Online VaxCluster system. 5 Large Vaxes plus about 25 VaxStations. Runs VMS and DAQ software. See also VaxCluster.

ALPHA The ALEPH Analysis Program — a skeleton and library for use by Physicists to simplify the job of accessing ALEPH data.

ALPROD (ALeph PRODUCTION) Program that is run over all the data coming from the reconstruction facility FALCON running JULIA to generate DSTs and EDIRs.

batch A mode of running a computer program where no terminal connection is necessary. Normally output is logged to a file. Sometimes known as a background process.

Beam centroid Oval region in (x, y) where electron and positron bunches pass through one another. Projection of beam spot in $x - y$ plane.

Beam spot Region in (x, y, z) where the electron and positron bunches pass through one another. The **Event spot** should lie within this region. Determined by a number of methods: see BOM, JULIA, Beam Centroid.

BM Buffer Manager. A piece of software managing the allocation of storage, with multiple data-stream capability plus asynchronous operation. ALEPH has two versions: one for the Vax [23] and one for the Event Builders [24].

BOM Beam Orbit Monitor. Used by LEP for feedback for the beam position control. Used in ALEPH to provide online information about the beam position in the centre of the detector.

BOS A memory management system written by Volker Blobel at DESY. It provides standard FORTRAN with rudimentary facilities for dynamic data structures.

- Book** The action of reserving an element of the Data Acquisition System for the use of a restricted set of tasks. This set can be those in a partition or a single task. See also RM.
- C** Only supported programming language on the micros (under OS9). Also used extensively on the Vax. See also OS9, VAX.
- CM** Centre-of-mass. An abbreviation for the available energy in the reference frame where colliding particles have no net momentum. In LEP, this usually corresponds very closely to the lab frame.
- compatible** A resource that is already booked can be booked by another process if the two requests are compatible, i.e. no access conflicts will result.
- Context Diagram** Topmost level data flow diagram that summarises the data flows of the entire system. See DFD.
- DAQ** Abbreviation for Data AcQuisition System.
- Data Dictionary** A repository holding a description of all the data items and structures for a project.
- Data Logger** Descriptive name for the computer plus software that writes raw data to tape and / or disk. See also VAX, FALCON.
- DFD** Data Flow Diagram. Diagram used to show how data and procedures inter-relate. Part of the SASD methodology for software design.
- DST** Data Summary Tape. The data-set one stage after the POT. Some data reduction done by removing uninteresting events and dropping irrelevant data for particular event types (such as luminosity events). See also MINI.
- EB** An Aleph Event Builder - a general purpose Fastbus master. See also MEB, LEB, OS9, [8].
- EDIR** Event DIRectory. List of all events in a run with classification data for each defining the event and trigger types. Produced by ALPROD.
- ERD** Entity Relationship Diagram. See ERD Model.
- ERD Model** Entity Relationship Data Model. A data model based on a technique where data is described in tabular form with attributes within a table and relationships (connections) to other tables.
- EVI** Event builder Vax Interface - add-on board for an EB which converts it into a Vax-Fastbus Interface. See also MEB, MIF, EB, [21].

Event spot The point in space where the electron and positron from the beams originally annihilate.

FALCON The Local Area VaxCluster dedicated to running the reconstruction program Julia. Shares disks with Online cluster for efficient data transfer. See also ALONL, VaxCluster.

FIO Fan-In fan-Out module. A programmable electronic switch used to route ECL level trigger signals from a Trigger Supervisor to the ROCs. See also TS, ROCs.

FORTRAN Common programming language - comes in a number of flavours. See also FORTRAN 77, Vax FORTRAN, C.

FORTRAN 77 Standardised programming language that provides portability between computers of differing types. See also FORTRAN, C.

Freezing A facility provided by the FIO units whereby they can be commanded to halt in their current state maintaining the existing signals on their output ports. Used for debugging trigger timing problems. See also FIO.

GALEPH The ALEPH event simulation program; a contraction of GEANT and ALEPH. See also GEANT.

GEANT The CERN general library used to simulate the passage of particles through matter.

global A term for a data item accessible from any part of a program. Functional opposite of local.

global section Method of defining an area of memory in a Vax that can be shared between processes.

Hardware Anything mechanical e.g. Detector chambers, wires, the computer's insides, a video screen. Compare with Software.

HCP HCAL Processor. Readout Controller for HCAL. See also ROC, EB, TPP.

Hits Coordinates generated by the detectors. Used for track finding and fitting.

JULIA The ALEPH reconstruction program. Processes raw data and generates the Production Output Tape with track, calorimeter object and vertex information. See also POT.

LEB Local Aleph Event Builder - performs sub-detector event-building function. See also EB, MEB, OS9.

Lock A Lock in Vax/VMS jargon is a way of coordinating access to something in the computer system that must be shared, such as a file, where if more than one person tried to update it at once, some changes would be lost. One says that the first person must lock the file, then release the lock, when the second person (who may be in a queue, waiting), can then lock it, and make whatever changes are desired. See also [38].

Lock Mode The access level requested or granted for a Lock. See also Lock, [38].

Lock Value Block Area of storage assigned to a Lock, available for general use. In the case of the ALEPH Resource Manager, the Lock Value Block is used to store the resource's partition ID and internal name. See also Lock, RM.

Logical Name The Vax/VMS equivalent of an alias for a file or other entity.

Main Partition A Partition constructed exclusively from other (sub-)partitions.

Master Active module in a Fastbus operation. Initiates operation. Compare with Slave.

MEB The Main Event Builder - at the top of the readout tree - has the Level 3 trigger connection. See also EB, MIF, LEB, OS9, Level 3.

MIF The Main InterFace Event Builder - the secondary EB at the top of the readout tree - used for special tests or in the event of failure of MEB. See also EB, MEB, LEB, OS9.

MINI Mini-DST. The compressed data-set. The data size is reduced by keeping a reduced amount of information and integerizing the remaining data such that the ALEPH Compression/Decompression package can work optimally. See also DST, POT.

OS9 Name of microprocessor operating system used by Aleph Event Builders and Slow Control Gateways. See also EB, OS9, VMS, C.

Partition A subset of the DAQ system capable of performing DAQ activities autonomously.

Partition ID Unique numerical identifier for a partition.

Partition Lock A Lock or resource that is the parent lock of all resources in a partition. By booking the partition lock an entire partition is booked. See also Lock, PM, RM.

partitioning The act of dividing up the DAQ system into autonomous self-contained DAQ sub-systems.

Partition Rules External constraints imposed on the ALEPH DAQ system to enable the simultaneous running of multiple partitions.

PCP Partition Control Process. Process on Vax that initiates all partition activity. All processes in a partition are child processes of this. See also Partition.

Petri Net A diagrammatic description of the how the transitions of one part of a system depend on the states of other parts of the system.

PM The Partition Manager. The package responsible for the organisation of partitions.

POT Production Output Tape. The data-set containing the results of the reconstruction program JULIA. Used as input for analysis programs and the DST production program (ALPROD). See also ALPROD, JULIA.

resource An element or module in the DAQ system that may at some time be required by more than one task for conflicting purposes. Also, Limited Generic resources are resources that may be used in duplicate form up to a maximum number of copies. Sequential Generic resources are an unlimited supply of ever-increasing numbers.

RM The Resource Manager. The package responsible for the management of cluster-wide resources in the DAQ system.

ROC Readout Controller. Lowest level in the Readout database. These modules perform the raw data readout and in some cases, processing. They pass their data up to readout processors higher up the DAQ pipeline, such as Event Builders. See also EB, ROP, HCP, TPP.

ROP Readout Processor. The generic name for an Event Builder or Readout Controller.

SASD Structured Analysis Structured Design. A methodology defining how to approach the analysis of problems and the design of the solution. In this context, particular to computer software.

Shareable (Image) Library This is a library that has all its internal references to other functions or routines resolved (i.e. pre-linked). It also has NO global data in it. This means it can be shared by any users. This results in a shorter link time and smaller executable programs (since the routines reside in the library image). Only one copy of the library is needed and whenever it is updated, all users automatically get a new version, making maintenance much easier.

Slave Passive module in a Fastbus operation. Does not initiate transfer. Compare with Master.

Software A computer program, or collection thereof. Compare with Hardware.

Software Life-cycle The stages in the life of a piece of software: from requirements specification, through analysis, design, prototyping, implementation, testing, debugging, documentation and maintenance.

Spy Channel A hardware add-on to an Event Builder allowing identical copies of data in memory; one is read out by the normal DAQ, the other by a Spy process running on a separate Vax. Very useful for monitoring.

SR Service Request. Fastbus-level interrupt from a Slave to a Master. See also Fastbus, Slave, Master.

SRH Service Request Handler. A Task running on one Event Builder responsible for receiving (and acknowledging) SRs from Slaves (usually ROCs) lower down in the hierarchy. See also EB.

Sub-detector One part of the whole detector that is dedicated to a specific detection method.

Sub-partition A partition made up only from elements contained within one sub-detector.

TPP TPC Processor. The readout controller module for the TPC. See also ROC, HCP, EB.

Transfer Vectors A method by which libraries of functions can be provided such that changes in their code do not require users to re-make their programs.

Triplet A group of three coordinates or hits joined to form a track segment.

TS Trigger Supervisor. Control unit for the trigger protocol. Issues timing signals and checks for correct responses.

UPI User Interface Package. A set of routines providing menu and dialogue facilities both on the Vax and under OS9.

V0DST The reduced data-set used for the V^0 analysis containing only tracking, V^0 and, if present, Monte Carlo truth information. Made from the DST by the V^0 finding program.

VAX Trade name of main computers performing the control and data management functions of the DAQ system. 32-bit word minicomputers. Also available as personal workstations known as VaxStations. In the online environment, run the VMS operating system. See also Host, VaxStation, VMS.

Vax FORTRAN Type of FORTRAN implemented on the Vax computers, offering more complex data structures and facilities than FORTRAN 77. A large proportion of online software is written in this language. See also FORTRAN, FORTRAN 77, VAX.

VMS Operating System running on Vax computers. Multi-tasking, multi-user system. Very sophisticated communication and file management facilities.

ZEBRA A memory management system written at CERN. It provides standard FORTRAN with rudimentary facilities for dynamic data structures. Has more built-in features than its earlier counterpart BOS.

References

- [1] W. Blum (editor). The ALEPH handbook. Technical Report ALEPH 89-77 Note 89-03, ALEPH, 1989.
- [2] ALEPH collab., D.Decamp et al., Nucl. Instr. and Meth. **A294** pages 121–178. 1990.
- [3] A.Putzer. The ALEPH Data Base System. Technical Report ALEPH 86-113, ALEPH, 1986.
- [4] W.Wiedenmann. Tracking with the ALEPH Time Projection Chamber. Technical Report ALEPH 90-156 TPCGen 90-8 and CERN, ALEPH, 1990.
- [5] IEEE. *IEEE Standard FASTBUS Modular High-Speed Data Acquisition and Control System*. ANSI/IEEE, 1986. Std. 960-1986.
- [6] D.R.Botterill. Fastbus database and initialisation. Technical Report ALEPH 89-206 DATAQ 89-39, ALEPH, 1989.
- [7] A.Belk et al., IEEE Trans. on Nucl. Sci. **36** pages 1534–1539. October 1989.
- [8] K.Einsweiler et al., IEEE Transactions on Nuclear Science **35** pages 316–320. February 1988.
- [9] A.Marchioro. Event Builder User's guide. Technical Report ALEPH 88-59 DATAQ 88-11, ALEPH, 1988.
- [10] G.Kellner. Software engineering. In *1988 CERN School of Computing*, number CERN 89-06, 1989.
- [11] G.Kellner. Development of software for ALEPH using structured techniques. Technical Report ALEPH 87-35 COMPRO 87-9, ALEPH, 1987.
- [12] R.Brazioli, S.M.Fisher, P.Palazzi, and W.R.Zhao. ADAMO notes v3.0. ALEPH Note, 1988.
- [13] J.Knobloch. ALEPH offline SASD experience. In W.Wojcik D.Perret-Gallix, editor, *International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics, Lyon*. Editions du Centre national de la Recherche Scientifique, 15, quai Anatole France, 75700 Paris, 1990.

- [14] G.Kellner, J.Bunn, S.M.Fisher, P.Palazzi, and W.R.Zhao. Practical guidelines for the use of SASD. Technical Report ALEPH NOTE 142, ALEPH, 1986.
- [15] P.Palazzi. Use of ADAMO data management system within ALEPH. Technical Report ALEPH 87-27 NOTE 87-6, ALEPH, 1987.
- [16] T.Charity, J.Harvey, and R.McClatchey, *Comp. Phys. Comms.* **45** pages 433–441. 1987.
- [17] Dataflow Group. Requirements for the ALEPH DAQ. Technical Report DATACQ 84-1, ALEPH, 1984.
- [18] S.R.Amendolia and J.F.Renardy, *IEEE transactions on Nuclear Science* **36** pages 602–605. 1989.
- [19] Dataflow Group. ALEPH DAQ: Hardware functional specifications. Technical Report DATACQ 85-21, ALEPH, 1985.
- [20] R.Bernard. Trigger supervisor specifications. Technical Report ALEPH 89-13 DATACQ 89-4, ALEPH, 1989.
- [21] V.Mertens. EVI: A high-speed interface between fastbus and VAX-BI. In *Adriatico Conference on the Impact of Digital Microelectronics and Microprocessors on Particle Physics, Trieste*, March 1988.
- [22] B.Jost et al., *IEEE Trans. on Nucl. Sci.* **36** pages 1452–1455. October 1989.
- [23] D.Quarrie. Data acquisition system programmers reference manual. Technical Report Note No. 166 (V3.4), CDF, CDF Computing Group, Fermilab MS318, Batavia IL 60510, USA, September 1989.
- [24] P.Mato. OS-9 buffer manager user's guide. Technical Report ALEPH 88-207 DATACQ 88-40, ALEPH, 1988.
- [25] J.F.Renardy, *IEEE Trans. on Nucl. Sci.* **36** pages 1464–1468. October 1989.
- [26] C.N.Booth. ALEPH t_0 module. Technical Report ALEPH 89-112 DATACQ 89-18, ALEPH, 1989.
- [27] P.P.Chen, *ACM Transactions on Database Systems* **1-1** pages 9–36. 1976.
- [28] P.T. Ward and S.J.Mellor. *Real-Time Modeling Issues, Modeling Transformations*, volume 1, chapter 3,5 and 6. Yourdon Press, 15/17 Ridgemount St., London WD1E 7BH, 1985.
- [29] P.T. Ward and S.J.Mellor. *Specifying Control Transformations*, volume 1, chapter 7. Yourdon Press, 15/17 Ridgemount St., London WD1E 7BH, 1985.

- [30] C.Arnault, J.Bourotte, J.Harvey, and A.Lacourt. UPI: the ALEPH user interface package. Technical Report ALEPH 87-41 COMPRO 87-11, ALEPH, 1987.
- [31] J.Martin. Run configuration tables. Technical Report ALEPH 89-139 DATACQ 89-23, ALEPH, 1989.
- [32] T.Charity, J-M.Maugain, W. von Rüden, and S.Wheeler. The ALEPH slow control system. In *Computing for High Luminosity and High Intensity Facilities, Santa Fe, New Mexico*, number 209, pages 386–394. American Inst. of Phys., 1990.
- [33] J.F.Renardy. The ALEPH-LEP communication system. Technical Report ALEPH 90-162 DATACQ 90-12, ALEPH, 1990.
- [34] G.McPherson. The ALEPH FIO module. Technical Report ALEPH DATACQ 84-21, ALEPH, 1984.
- [35] A.Belk, D.R.Botterill, and J.Harvey. The aleph online database. Technical Report ALEPH 88-205 DATACQ 88-38, ALEPH, 1988.
- [36] C.J.Date. *An Architecture for a Database System*. System Programming. Addison-Wesley, Reading, Massachusetts, third edition, 1975.
- [37] Digital, DEC, 77 Reed Road, Hudson, Massachusetts 01749, USA. *Introduction to VMS System Services*. Sections, § 11.7.
- [38] W.E.Snaman Jr. and D.W.Thiel, Digital Technical Journal 5 pages 29–44. 1987, From: DEC, 77 Reed Road, Hudson, Massachusetts 01749, USA.
- [39] Digital, DEC, 77 Reed Road, Hudson, Massachusetts 01749, USA. *Introduction to VMS System Services*. Lock Management Services, § 12.
- [40] A.Belk and J.Shade. The Aleph Resource Manager. Technical Report ALEPH 89-140 DATACQ 89-24, ALEPH, 1989.
- [41] A.Belk. The ALEPH Partition Manager. Technical Report ALEPH 89-138 DATACQ 89-22, ALEPH, 1989.
- [42] A.Belk and J.Harvey. Resource naming convention. Technical Report ALEPH 88-4 DATACQ 88-1, ALEPH, 1988.
- [43] P.Kunz, Comp. Phys. Comms. 57 pages 191–197. 1989.
- [44] A.Belk and O.Callot. Files, logical names, releases and other things. Technical Report ALEPH 89-48 DATACQ 89-9, ALEPH, 1989.
- [45] R.Brun et al. GEANT 3. Technical Report CERN DD/EE/84-1, CERN, 1987.

- [46] W.Atwood et al. GISMO. Technical Report Talk given at CERN on 22nd April 1991, CERN, 1991.
- [47] A.Belk. The C Language Binder – User Guide. Technical Report ALEPH 91-4 DATACQ 91-1, ALEPH, 1991.
- [48] M.Delfino. A Hierarchical Organization for the ALEPH Analysis Software. Technical Report ALEPH 87-88 SOFTWR 87-13, ALEPH, 1987.
- [49] A.Blondel and E.Blucher. Hadronic event selection with charged tracks. Technical Report ALEPH 90-171 PHYSIC 90-99, ALEPH, 1990.
- [50] PARTICLE DATA GROUP collab., M.Aguilar-Benitez et al., Phys. Lett. **B239** . April 1990.
- [51] J.H.Christenson et al., Phys. Rev. Lett. **13** pages 138–140. 1964.
- [52] D.Fournier. NA31 results on CP violation in K decays, and a test of CPT. Technical Report CERN EP/90-01, CERN, 1990.
- [53] F.Ranjard. The Kingal User Guide. Technical Report ALEPH 87-53 SOFTWR 87-7, ALEPH, 1987.
- [54] T.Sjöstrand. The Lund Monte Carlo for Jet Fragmentation and e^+e^- Physics - JETSET version 6.2. Technical Report LU-TP-85-10, Lund University, 1985.
- [55] T.Sjöstrand and M.Bengtsson, Comp.Phys.Comm. **43** page 367. 1987.
- [56] ALEPH collab., D.Decamp et al., Phys. Lett. **B234** . January 1990.
- [57] D.Decamp et al., Phys. Lett. **B 255** pages 623–633. 1991.
- [58] H.Meinhard. Efficiency of track finding and track reconstruction. Technical Report ALEPH 90-170 NOTE 90-6, ALEPH, 1990.
- [59] R.St.Denis. Study of tracks at the edge of acceptance. Technical Report ALEPH 91-33 PHYSIC 91-32, ALEPH, 1991.
- [60] R.Johnson. Scanning events for track losses. Technical Report ALEPH 91-34 PHYSIC 91-33, ALEPH, 1991.
- [61] S.Wasserbaech. Energy loss in matter by charged particles. Technical Report Minutes of the Tracking Meeting held 18th January 1991, ALEPH, 1991.
- [62] J.Knobloch and P.Norton. Status of Reconstruction Algorithms for ALEPH Version 3 (Draft). Technical Report ALEPH 91-xxx NOTE 91-xxx (to be published), ALEPH, February 1991.

- [63] L.Garrido. Track fitting with multiple scattering. Technical Report ALEPH 88-075 NOTE 88-011, ALEPH, 1988.
- [64] N.Chernov and G.Osokov, Comp. Phys. Comms. **33** pages 329–333. 1984.
- [65] M.Bosman and G.Lutz. Topological vertex software for ALEPH. Technical Report ALEPH 88-46 NOTE 88-6, ALEPH, 1990.
- [66] J.Knobloch and P.Norton. Status of reconstruction algorithms for ALEPH. Technical Report ALEPH 88-46 NOTE 88-6, ALEPH, 1988.
- [67] K.Karimaki. Formulae for the UA1 V^0 finding algorithm. Technical Report UA1-TN 84-31, UA1, 1984.
- [68] W.Wiedenmann. Performance of the ALEPH Time Projection Chamber. Technical Report CERN-PPE 91-24, CERN, 1991.
- [69] I.R.Tomalin. Track finding efficiency in the central detector. Technical Report ALEPH 89-135 SOFTWR 89-18, ALEPH, 1989.
- [70] I.R.Tomalin. *Strange Baryon Production in e^+e^- Annihilation*. PhD thesis, Oxford University, 1988.
- [71] S.Wasserbaech. Checks of energy loss and bremsstrahlung in GALEPH. Technical Report Minutes of the Tracking Meeting held 12th April 1991, ALEPH, 1991.
- [72] ARGUS Collaboration, Z. Phys. C **44** pages 547–555. 1989.
- [73] S.Behrends et al., Phys. Rev. D **31** page 2161. 1985.
- [74] H.Aihara et al., Phys. Rev. Lett. **53-25** pages 2378–2381. 1984.
- [75] S.Abachi et al. Quark hadronization probed by K^0 mesons. Technical Report ANLHEP-PR-89-73, also IUHEE-89-4 and UM-HE-89-25, HRS, 1989.
- [76] H.Schellman et al., Phys. Rev. D **31** pages 3013–3016. 1985.
- [77] PLUTO Collaboration, Phys. Lett. **104B 1** pages 79–83. 1981.
- [78] JADE Collaboration, Z. Phys. C **20** pages 187–206. 1983.
- [79] TASSO Collaboration, Z. Phys. C **27** pages 27–37. 1985.
- [80] CELLO Collaboration. Inclusive strange particle production in e^+e^- annihilation. Technical Report DESY 89-140, DESY, 1989.
- [81] OPAL Collaboration. A study of K_S^0 production in Z^0 decays. Technical Report CERN-PPE/91-86 89-140, CERN, 1991.

- [82] M.Derrick et al., Phys. Rev. D **35** pages 2639–2649. 1987.
- [83] ALEPH Collaboration. Measurement of electroweak parameters from Z decays into fermion pairs. Technical Report CERN-PPE 90-104, CERN, 1990.

“There is no national science, just as there is no national multiplication table”.

Anton Chekov