Draft 1.0

THE KLOE COLLABORATION

THE KLOE DATA ACQUISITION SYSTEM

ADDENDUM TO THE KLOE TECHNICAL PROPOSAL

ABSTRACT

We present in the following the design of the KLOE Data Acquisition System. Included are the philosophy and extensive discussions on all parts of the system.

THE KLOE COLLABORATION

A. Aloisio,[e] A. Andryakov,[b] A. Antonelli,[b] M. Antonelli,[b] F. Anulli,[h]
C. Avanzini,[h] D. Babusci,[b] C. Bacci,[j] R. Baldini-Ferroli,[b] G. Barbiellini,[m]
M. Barone,[h] K. Barth,[c] V. Baturin,[e] H. Beker,[h] G. Bencivenni,[b] S. Bertolucci,[b]
C. Bini,[h] C. Bloise,[b] V. Bocci,[i] V. Bolognesi,[g] F. Bossi,[b] P. Branchini,[k] L. Bucci,[h]
A. Calcaterra,[b] R. Caloi,[h] P. Campana,[b] G. Capon,[b] M. Carboni,[b] G. Cataldi,[d]
S. Cavaliere,[e] F. Ceradini,[j] L. Cerrito,[i] M. Cerù,[h] F. Cervelli,[g] F. Cevenini,[e]
G. Chiefari,[e] G. Ciapetti,[h] M. Cordelli,[b] P. Creti,[d] F. Donno,[b] R. De Sangro,[b]
P. De Simone,[b] G. De Zorzi,[h] D. Della Volpe,[e] A. Denig,[c] G. Di Cosimo,[h]
A. Di Domenico,[h] A. Doria,[e] E. Drago,[e] V. Elia,[d] O. Erriquez,[a] A. Farilla,[a] G. Felici,[b]
A. Ferrari,[g] M. L. Ferrer,[b] G. Finocchiaro,[b] D. Fiore,[e] P. Franzini,[h,f] A. Gaddi,[b]
C. Gatto,[e] P. Gauzzi,[h] E. Gero,[b] S. Giovanella,[h] E. Gorini,[d] F. Grancagnolo,[d]
W. Grandegger,[b] E. Graziani,[k] U. v. Hagel,[c] R. Haydar,[b] M. Imhof,[c] M. Incagli,[g]
C. Joram,[c] L. Keeble,[b] W. Kim,[l] W. Kluge,[c] F. Lacava,[h] G. Lanfranchi,[h] P. Laurelli,[b]
J. Lee-Franzini,[b,l] G. Margutti,[h] A. Martini,[b] A. Martinis,[m] M. M. Massai,[g]
R. Messi,[i] L. Merola,[e] S. Miscetti,[b] S. Moccia,[b] F. Murtas,[b] M. Napolitano,[e]
A. Nisati,[h] E. Pace,[b] G. F. Palamà,[d] M. Panareo,[d] L. Paoluzi,[i] A. Parri,[h]
E. Pasqualucci,[i] M. Passaseo,[h] A. Passeri,[k] V. Patera,[b] F. Pelucchi,[b] E. Petrolo,[h]
M. C. Petrucci,[h] M. Piccolo,[b] M. Pollack,[l] L. Pontecorvo,[h] M. Primavera,[d] F. Ruggieri,[a]
P. Santantonio,[b] R. D. Schamberger,[l] A. Sciubba,[h] F. Scuri,[m] A. Smilzo,[e] S. Spagnolo,[d]
E. Spiriti,[k] C. Stanescu,[k] L. Tortora,[k] P. M. Tuts,[f] E. Valente,[h] P. Valente,[b]
G. Venanzoni,[g] S. Veneziano,[h] S. Weseler,[c] R. Wieser,[c] S. Wölfle,[b] A. Zallo.[b]

[a] Dipartimento di Fisica dell'Università e Sezione INFN, Bari
[b] Laboratori Nazionali di Frascati dell'INFN, Frascati
[c] Institut für Experimentelle Kernphysik, Universität Karlsruhe
[d] Dipartimento di Fisica dell'Università e Sezione INFN, Lecce
[e] Dipartimento di Scienze Fisiche dell'Università e Sezione INFN, Napoli
[f] Physics Department, Columbia University, New York
[g] Dipartimento di Fisica dell'Università e Sezione INFN, Pisa
[h] Dipartimento di Fisica dell'Università e Sezione INFN, Roma I
[i] Dipartimento di Fisica dell'Università e Sezione INFN, Roma II
[j] Dipartimento di Fisica dell'Università di Roma III e Sezione INFN, Roma I
[k] Istituto Superiore di Sanità and Sezione INFN, ISS, Roma.
[l] Physics Department, State University of New York at Stony Brook.
[m] Dipartimento di Fisica dell'Università e Sezione INFN, Trieste/Udine

TABLE OF CONTENTS

1. INTRODUCTION

In this addendum, we describe the KLOE data acquisition system, DAQ: the front end electronics; the two level read out of the FEE output; the data transmission to the on line processing farm; the processing farm; mass storage and off line computing requirements. The front end electronics, FEE, is included as short descriptions of the basic components to help in better understanding the DAQ's proper functions. Emphasis is placed on the high speed two level data concentration, distribution to the farm of "single board computers", SBCs, on data flow control, run control and monitoring, and on data handling and mass storage. This document is an amplification of the preliminary description of the system which has been presented in the KLOE proposals.[1–3]

The requirements of KLOE represent a challenge in this field, since we have to deal with data rates, $\mathcal{O}(10^{11}$ events/year), considerably higher than those of typical collider experiments such as LEP or TEV-I, albeit a factor ten smaller than those foreseen at the LHC. In contrast to hadron collider experiment, the large amount of data collected with KLOE are all of physics interest, or necessary for detector calibration. Moreover, since the major aim of KLOE is to perform CP violation studies at sensitivities of $\mathcal{O}(10^{-4})$, very stringent requirements are imposed on its DAQ system, to insure that no biases are introduced at such levels. Error rates, especially if dependent on event configurations, which would be quite acceptable in other situations, must be minimized and procedures for insuring data integrity must be incorporated in the system design *ab initio*.

2. SYSTEM OVERVIEW

2.1 Data Rates and other Boundary Conditions

The expected data rate from the KLOE detector, once the DAΦNE collider reaches its maximum design luminosity, $\mathcal{L}$=1×10^{33} cm^{-2} s^{-1}, has been estimated as 10^4 events per second. Half of this rate is due to ϕ decays, the remainder is from Bhabha scattering, scaled down from the rate into the detector, ~50 kHz. Bhabha scattering events are fundamental in maintaining the calorimeter calibration scale for time and energy measurements, and to determine the tracking chamber parameters. It is also estimated that the average event size is 5 kbytes, corresponding to a total bandwidth requirement of 50 Mbytes/s.

Given the above rate, the average time for the read-out of an event from the FEE is 100 μs. Care must be taken to insure that the readout will not introduce dead times and especially to avoid the possible build-up of instantaneous backlogs. The occurrence of these backlogs might result in losses of parts of an event and would require a complicated system flushing procedure upon their occurrence. A two level data concentration scheme is the proposed solution.

While it is very hard to realize a completely deadtimeless DAQ system, what is truly necessary is to maintain the dead times at a reasonably low level, insuring that the dead time is not a function of the event configuration, a situation which could lead to the introduction of biases in the data.

2.1.1 Standards

The KLOE FEE and the data concentration scheme will use VME hardware, typically 9U cards, 400 mm deep. Wherever high volume digital data is transferred, the P1 and P2 connectors will provide standard VME functions. The P3 connector will be typically employed for analog data in and/or out. In addition, at the last level of the FEE electronics, a custom bus, using the free P2 pins, will be used for fast data transfer.

2.1.2 Dead Time

The KLOE FEE will perform analog signal conditioning and digitization at fixed dead time, chosen to be of $\mathcal{O}$(2 μs). This is equivalent to a 2% loss in luminosity, quite negligible on all accounts. Each individual detector channel will carry out analog processing and digitizing. After receipt of a trigger signal, generated by an appropriate trigger system, the entire KLOE FEE system has 2 μs in which to perform its functions after which it is ready to receive another trigger. In other words, in normal operation, no busy signals are generated by the FEE and the subsequent DAQ system.

2.1.3 Buffers

Because of the above specification, every FEE channel contains an event buffer of appropriate length, allowing asynchronous data read-out, as described later, without data losses.

2.1.4 FEE Errors

Busy signals might need to be generated by the FEE, in order to signal error conditions. These include buffer full errors; malfunction errors; data overflows; and so on. Reset and buffer flush-out must be provided at all DAQ levels.

2.2 Fast Data Read-out

The data from the FEE channels must be transferred to the on-line SBC farm at rates of 50 Mbytes/s, with a data concentration scheme that communicates with some 10,000 channels of calorimeter FEE electronics, distributed over 330 cards in 20 crates and 13,000 channels of tracking FEE electronics, distributed over 135 cards in 10 crates. A two level concentration solution is proposed. The first level, L1, is performed at the crate level, via a custom bus in the backplane, to a hardwired read-out controller, ROCK, K for KLOE, located in the crate itself. At this first level, sub-events are processed, where a sub-event is a piece of an event which is produced by a subset of the front end electronics which is connected in a chain. Group of crates or chains, of suitable length, communicate via a custom cable bus, Cbus, to the second level, L2, through a ROCK manager, ROCKM. The ROCKMs reside in their crates together with VME processors which prepare data from a string of sub-events for transmission to a given farm processor. A commercial bus, VIC, for vertical interconnect, connects all the crates in a chain allowing the VME processor to program, check and debug the FEE electronics.

2.3 Data Flow Control and Event Building

Data coming from the VME processors, for a given group of trigger numbers, must be gathered by a single board computer, SBC, where the integrity of the event is tested and the final event formatting is implemented. Specific hardware and software choices must maintain the required throughput of 50 Mbytes/s. A commercial FDDI switch with bridge functionality is used to provide parallel paths between the VME processors and farm in a scalable way. While the number of switch ports dedicated to the VME crates will be fixed initially, the number of ports dedicated to the farm can be increased as the DAΦNE luminosity increases.

2.3.1 Communication Links and Communication Protocols

The different components of the DAQ system are interconnected via Ethernet for operations requiring low bandwidth (e.g. controls, downloading, monitoring) and via FDDI for data transmission. A standard high level communication protocol is proposed, with reliable message passing, in order to be independent of the processors, the operating systems, and communication channels used by the different parts of the DAQ system.

2.3.2 Data Flow Controller

The address of the SBC that receives a string of sub-events, corresponding to a group of consecutive trigger numbers, is assigned by a VME control processor, the data flow controller or DFC, that is connected via another VIC channel to the ROCKM crates. Event integrity at the farm level, requires adequate synchronism between all the VME processors. A DAQ reset and buffer flush-out will be generated if a misalignment is discovered in the farm.

2.3.3 Event Building

The event building function is implemented in a distributed way: sub-events are collected in the ROCKMs, grouped in strings to be transmitted by the VME processors to a single SBC. The switch passes these strings to a single output port. Finally, the processor formats the single events, verifying their integrity.

2.4 Processor Farm

The farm is based on SBCs organized in crates. Each crate has a dedicated output SBC which collects data from the crate and routes it to a particular storage device, I/O SBC. The total CPU power foreseen at present in the farm is about 160 Specint'92 obtainable using a farm of about 100 processors. This farm performs event collection, formatting, integrity checking, statistics analyses, and some on-line reconstruction in an average time of 0.01 sec/event/SBC. The anticipated need for online reconstruction is about 16,000 Specint'92. The CPU power request for off-line re-processing is about three times larger. While no final decision has been made at present about the specific processors to be employed, we will describe in section 6.1 a project underway to realize an SBC employing the Digital Equipmen Corporation, DEC, Alpha-vax chip. Other commercial solutions are possible.

2.5 Software

The on-line software represents a challenge because beyond the first level of data concentrators, the software must maintain the event synchronization. Particular care must be devoted to insure that no biases are introduced, thus eliminating the dependence of the error rate on event configuration at the required level of confidence. The general architecture will be based on quasi-traditional components implemented on a non-homogeneous hardware configuration. Real-time operating systems run in every board where diskless operation is needed, both to reduce cost and to increase performance. The monitoring processes are run on general purpose VMS and/or UNIX systems.

A rather complex run controller, which is responsible for the activation and control of all the relevant tasks, is necessary, including the start/stop of a run and operator console interfaces which will allow overview and control of the operation of the system. A logger for error bookkeeping is also needed.

A tape/disk controller, needed for the output data stream control and monitoring, will be duplicated in every I/O SBC in the farm. A buffer manager will insure the correct handling of the events among processors and data receiving tasks such as the tape/disk controllers and the monitoring programs.

2.6 Mass Storage

The amount of data collected in a couple of years of running is of the order of 1,000 TB. The corresponding number of actual cassettes, assuming 30–165 Gbytes cassette will be 6,000–30,000, including the reprocessed data. The total storage requirement is even greater due to the Monte Carlo data that will be required and could only be managed with special robotics and a good computing organization. While the on-line system requires only loaders, both on-line and off-line analyses require software compatible robotics.

2.6.1 Hardware

The more promising devices are the AMPEX 19 mm tapes and the Quantum Digital Linear Tapes (DLTs), both supported by robotics systems, and offering different capacities, error rates, throughput, hardware compression availability. Decisions will be taken according to the cost/performance ratio, from measurements by the KLOE collaboration and in consultation with other laboratories.

2.6.2 Hierarchical File System

The use of robotics requires complex software, such as a hierarchical file system managing the storage history. This will facilitate the off-line reprocessing of the data. The availability of this kind of systems from industry or third parties will influence the final decision, for the storage device.

2.6.3 Database

A database will be needed, which will contain all relevant information regarding the runs and the events, such as calibration constants, trigger condition, detector configuration, collider status and performance, fill numbers, run numbers, etc. Part of this information will also be supplied in the processed data in dedicated banks such as the Run Header or the Event Header, thus simplifying the trace back of the original conditions and raw data tape searches. No special "event directory" will be needed for the KLOE raw data.

2.7 Slow Controls

By slow controls we mean the monitoring of many status and control variables. The system will probably be implemented in VME, monitoring and controlling the various parts of the apparatus. General services will be provided to control systems including: low voltages from VME crates and other power supplies, high voltages used for the electromagnetic calorimeter phototubes and for the chamber field wires, thresholds settings (e.g. of trigger modules), gas flow and composition, and finally monitoring the magnet settings.

A slow control station will run the software needed to produce monitoring statistics and carry out checks based on alarms and repetitive read-outs of the relevant values in the VME cards. The control station will also support menu and window displays of this monitoring information for both operators and other experts. For this we could use a commercial available standard such as Motif.

2.8 Conclusion

Fig. 2.1 shows a simplified block diagram of the system architecture outlined above, which will be described in more detail in the remainder of this report. The same figure indicates the information flow.

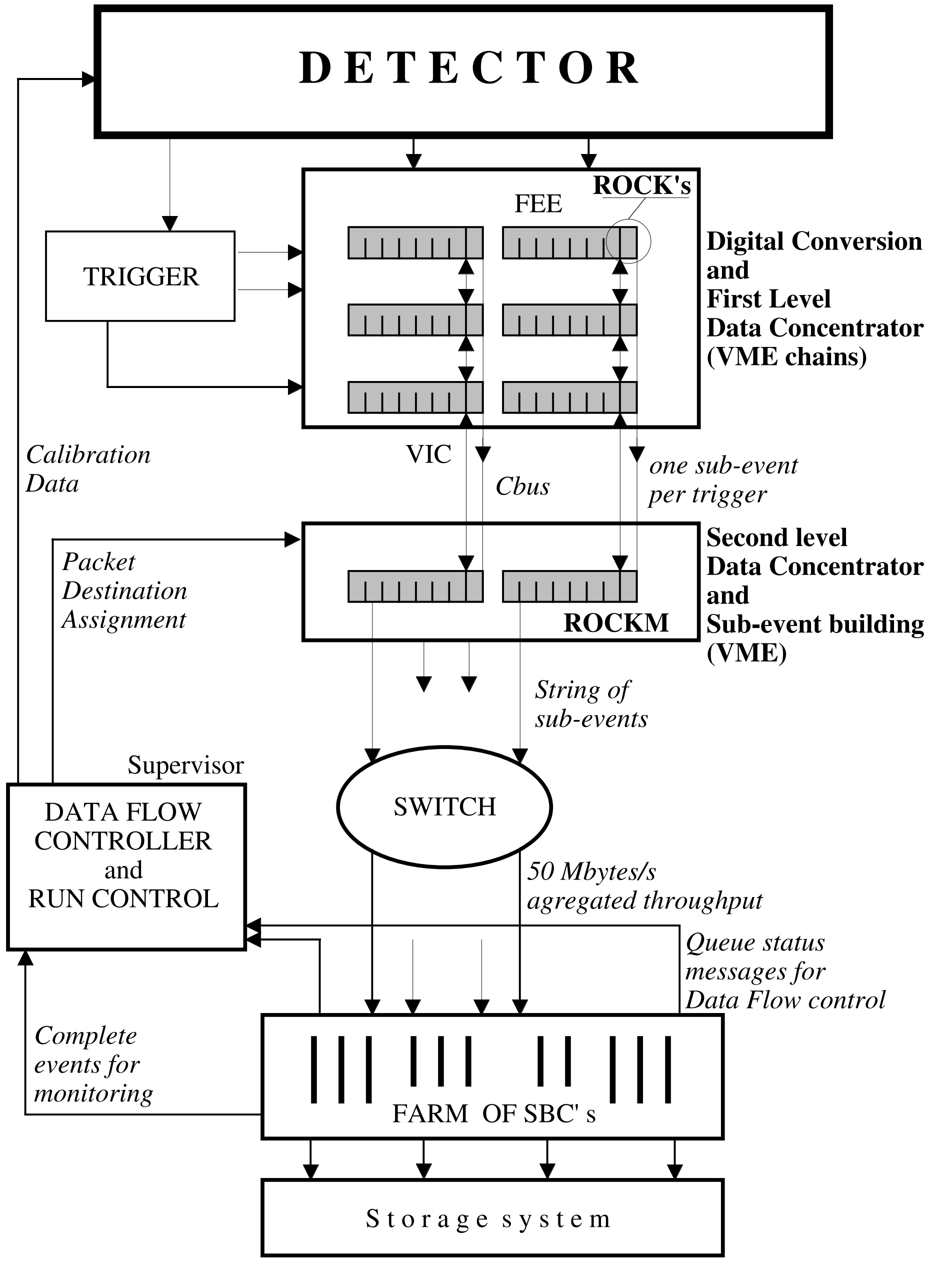

Fig. 2.1. System Architecture and Information Flow.

3. FRONT–END ELECTRONICS

3.1 Overview

The KLOE detector contains 5,000 photomultipliers, PMs, for the readout of the calorimeter, and ∼13,000 wires for the tracking chamber. The PM signals are amplified in the PMs' bases and sent to analog boards of 30 channels each. The outputs of this board are connected to ADCs, TDCs and to the trigger processor, fig. 3.1.

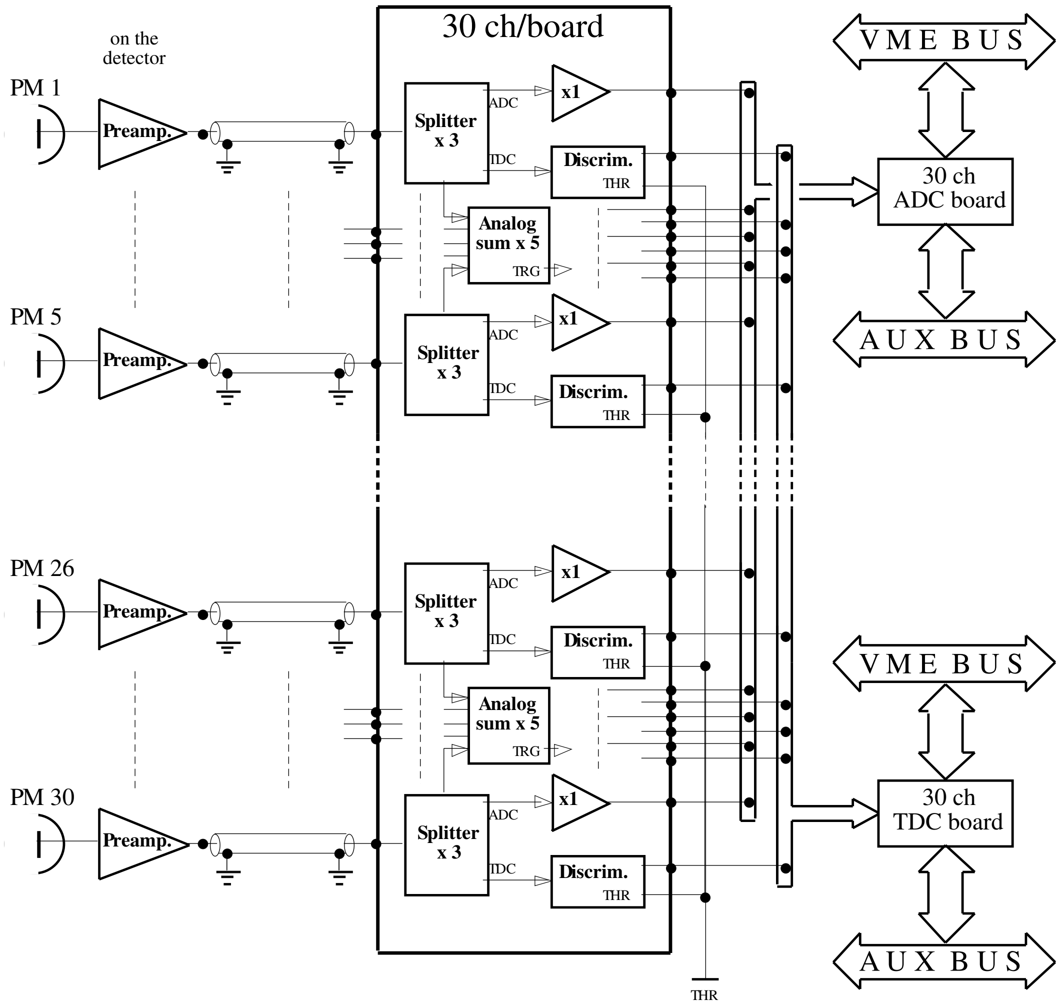

Fig. 3.1. Calorimeter Readout Scheme.

The chamber signals are preamplified at the wire feed-through and sent to boards of 48 channels each. The output of two of these boards are connected to a 96 channel TDC board for drift time measurements, fig. 3.2.

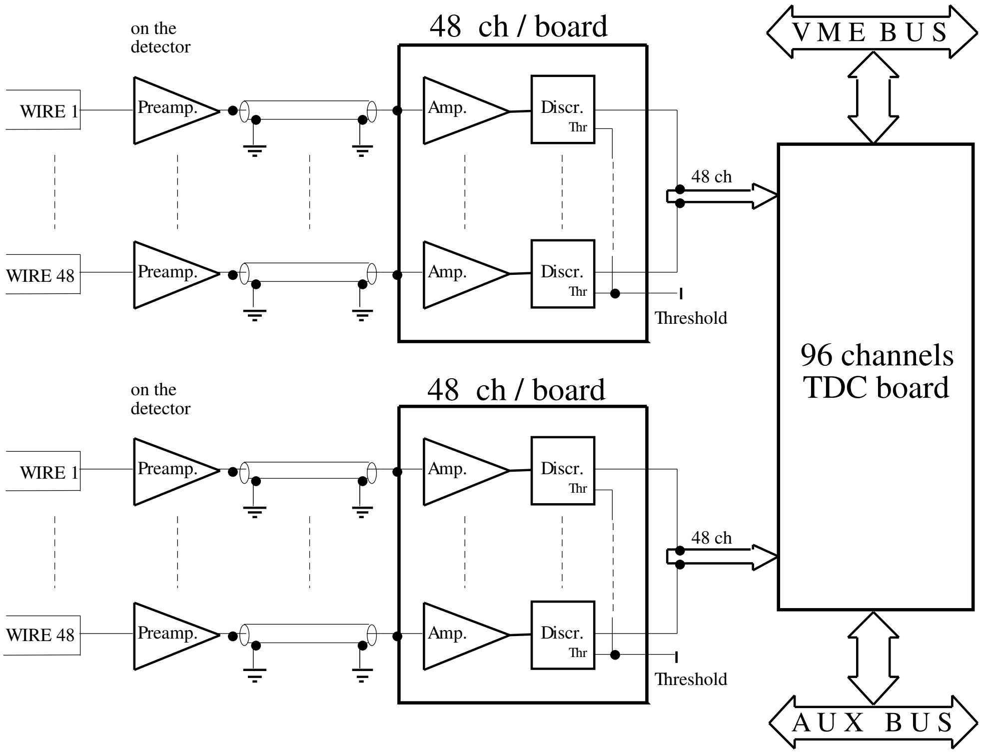

Fig. 3.2. Chamber Readout Scheme.

3.2 Calorimeter Front End Electronics

3.2.1 Preamps, Discriminators, etc.

The calorimeter front end electronics, fig. 3.3, consists of the following functional elements:

1. High voltage divider for the photomultiplier tubes, PMs, which are operated with grounded cathode and positive voltage on the anode, to avoid noise related to charging of surfaces in contact with the tube envelopes.
2. A.C. coupled preamp, for optimal coupling to the PM and driving the cable carrying the signals to the following stage.
3. A three way splitter of the analog signal to provide inputs to:
   (a) Fast, low slew, constant fraction discriminators, for time measurements with the KLOE TDCs;
   (b) Drivers, for amplitude measurements with the KLOE ADCs;
   (c) First level adders for the preparation of the signals for the KLOE trigger.

Voltage dividers and preamplifiers are built as small hybrid assemblies, mounted on the PM socket and located therefore inside the KLOE detector. Signals from the preamps are carried outside the KLOE iron yoke to the rest of the electronics via miniature coaxial cables. The preamplifier is a three transistor transimpedance device, with a conversion gain of 500 V/A. The amplifier output is connected to the cable with a series resistor equal to the impedance Z of the cable and a capacitor. The cable is also terminated at the receiving end. The back termination at the transmitting end insures negligible reflections from imperfect cable termination. It also provides a large effective time constant for the required AC coupling without the use of large capacitors.

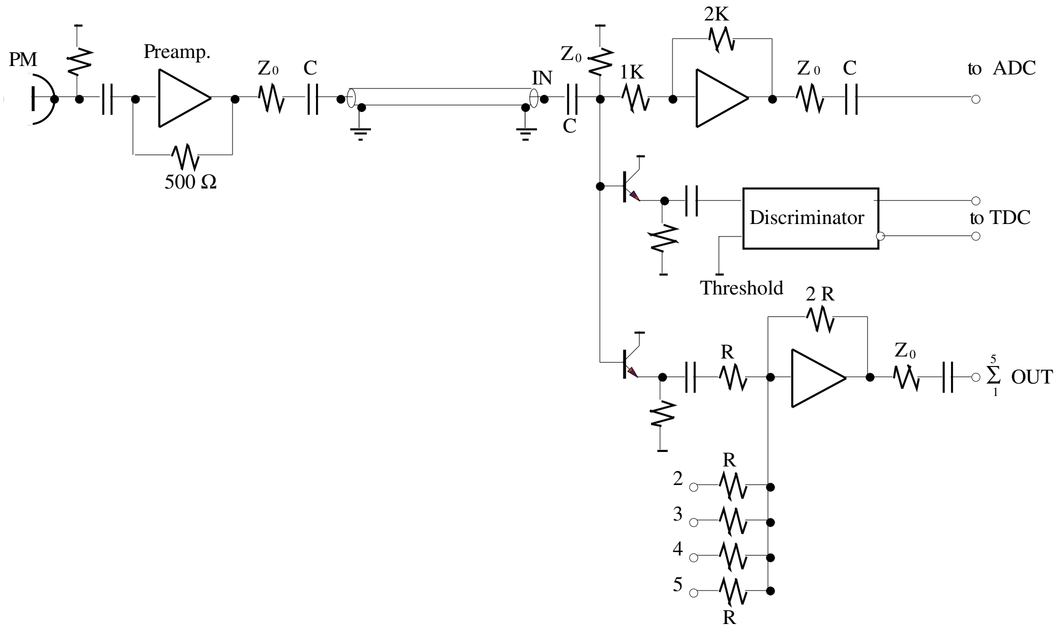

Fig. 3.3. Calorimeter Single Channel Front End Electronics.

The above elements a), b) and c) reside in 9U VME cards. Each card carries 30 channels of this electronics (dictated by the calorimeter structure which consists of 5 sections in depth) plus an as yet undefined number of adders. The signal splitting function is provided by two emitter followers and an amplifier identical to the PM preamp, operated as a unity gain voltage amplifier. Emitter followers are adequate for driving the fast discriminators and the adders for which linearities of 1-2% are quite acceptable, while a precision amplifier insures much better linearity for amplitude measurement and provides a good drive for the cable carrying the signal to the ADC boards.

The fast, low slew, constant fraction discriminator uses the standard technique of subtracting a delayed, inverted replica of the PM signal from the input signal, producing a bipolar signal. A fast, high gain comparator detects the zero crossing time, thus providing partial compensation of the variation in response time of the comparator with signal amplitude. Prototypes give time walks of 200 ps for input signals of 20 to 1,500 mV. The output of the discriminator is complementary ECL, and is carried to the TDC boards with twisted pairs.

The adders, are fast, linear transimpedance amplifiers using the same design as the PM preamps. At present, 6 five input adders on a card provide a signal proportional to the energy deposition in a basic calorimeter trigger segment. The proposed solution for the high voltage system is based on commercial components from CAEN.

3.2.2 Calorimeter ADCs

The KLOE calorimeter signal processing poses some special problems because the signal to be measured can appear over an interval Δt (0,-200) ns with respect to the trigger signal. We use a modified double sampling technique, following the integration of the PM signals. The integrated signal baseline is continuously sampled at fixed frequency on two capacitors C_a, and C_b. It is also sampled again some time after the trigger on a capacitor C_c, fig. 3.4. The difference of this last sample with the correct baseline sample, obtained in analog, is the integral of the PM signal and is digitized by a 12 bit, 1μs commercial ADC. The timing diagram for these operations is shown in fig. 3.5.

In order to perform these operation correctly and minimize the error due to the different arrival times of the signal and trigger, it is necessary for the integrator to be a large bandwidth device, with a reasonably long restoring time, a compromise between pileup and error.

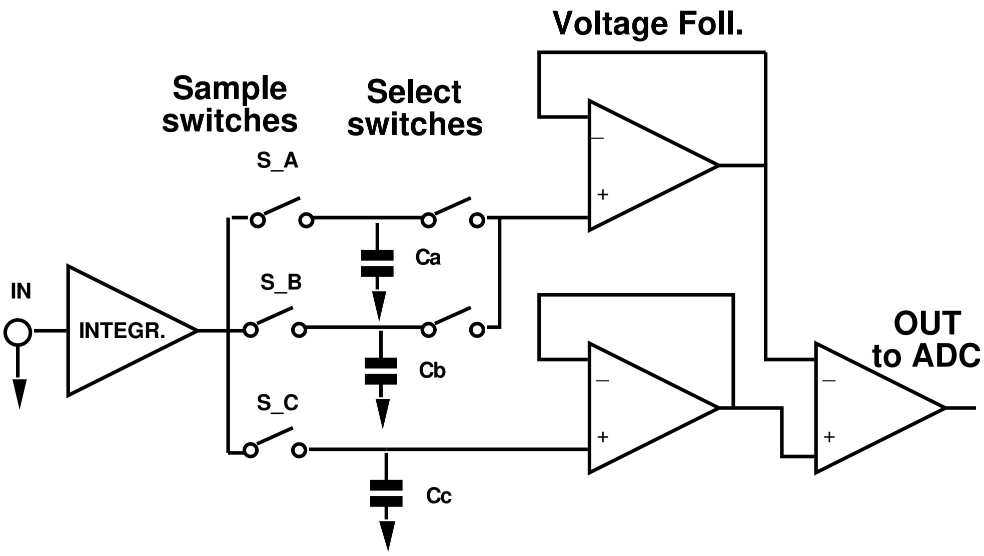

Fig. 3.4. ADC block scheme.

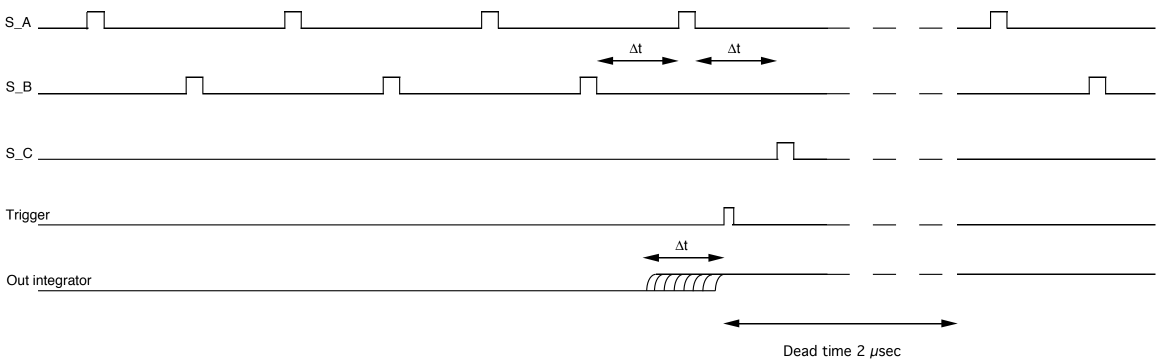

Fig. 3.5. Timing of signal sampling for the ADC.

This function is well performed by a standard charge sensitive amplifier with a j-FET at the input. The requirement of a long decay time, 100 to 1000 μs, results in the need to drive the integrator with a high impedance source. Therefore a voltage to current stage is used between the input signal and the integrator.

The 30 channel board will be 9U high and 400 mm deep, with a standard VME interface and a fast custom readout bus that uses the free row of the P2 connector. The main functions implemented on the board, for data processing and readout, are:

1. begin offset subtraction;
2. empty channel suppression in order to reduce the amount of data read out;

3. multi-event buffering to allow asynchronous readout from the front-end.

3.2.3 TDC for the Electromagnetic Calorimeter

The measurement of the time-of-arrival at the electromagnetic calorimeter is performed in a common start mode, the start being provided by the main trigger signal, synchronized with the bunch crossing time. Each TDC channel is built around a monolithic time-to-amplitude converter, see fig. 3.6, developed in bipolar technology, followed by a 1 μs conversion time 12 bit monolithic ADC chip. Each calorimeter signal is delayed by 100-300 ns with a monostable circuit and used as a stop. 30 TDC channels reside on a VME card. Full scale of a channel is 100 (200) ns with a resolution of 25 (50) psec and an integral non linearity of $\sim$0.1%.

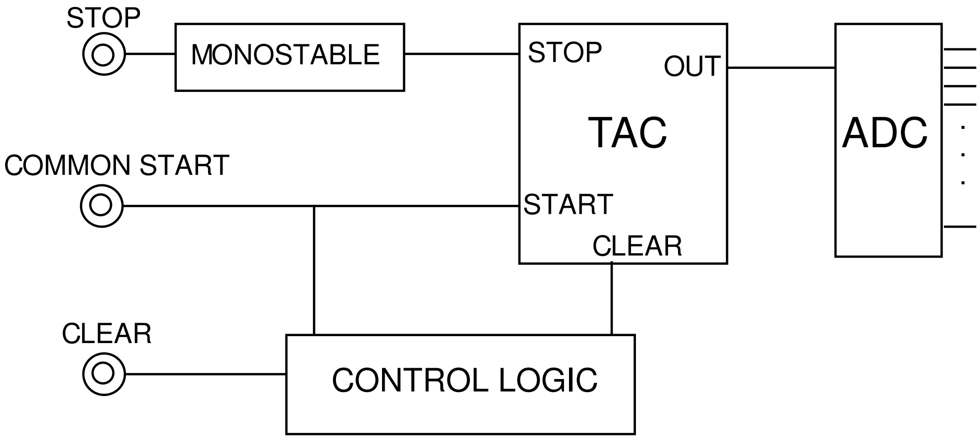

Fig. 3.6. Principle of operation of the TDC.

Functions implemented on the board for data processing and readout are:

1. Empty channel detection/suppression, via table look-up;
2. Hardware offset subtraction, via table look-up;
3. Multi event buffering to allow asynchronous fast readout of each card.

3.3 Front End Electronics for the Drift Chamber

3.3.1 Preamplifiers

It is common practice to amplify the sense wire signals at the place where they first become available, the wire feed-throughs. An investigation of different preamplifiers[4] has shown that the VTX chip[5] is the best one thanks to its extremely low power dissipation, package mass and price/channel. Specifications for this preamplifier are listed in table 1.

Table 1. Preamplifier specifications

Gain	1.0 mV/fC (50 Ω load)
Dynamic range	400 mV (3% linearity at max output)
Crosstalk	<0.5% between any two channels
Noise	860 electrons + 47 e/pF, 100 MHz bandwidth
Output risetime	5 ns
Output falltime	16 ns
Input impedance	130 Ω
Output impedance	43 Ω
Power dissipation	10m W/channel, 1 mA output pulldown current
Channels/chip	6

Since the VTX is provided as a bare die with six channels it is mounted on a card with the ancillary components shown in fig. 3.7.

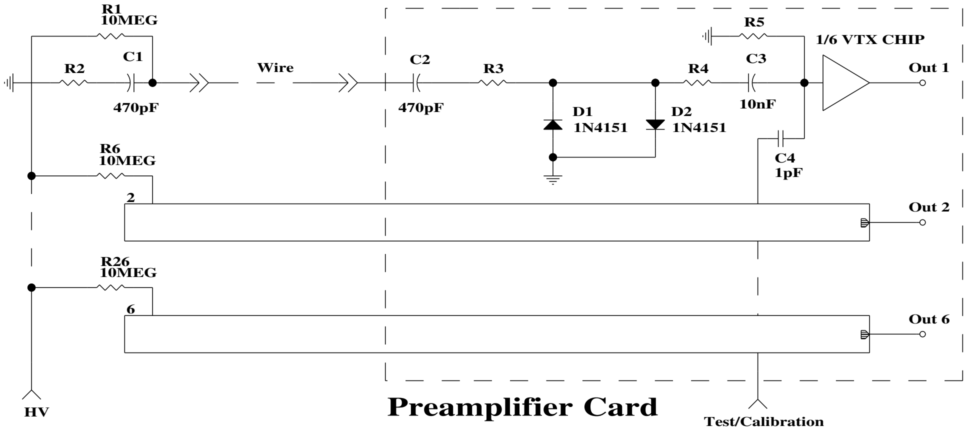

Fig. 3.7. Preamplifier card scheme.

Sense wires are coupled through a 470 pF 3 kV capacitor. Each preamplifier is protected against high voltage sparking by a diode pair and two resistors chosen to provide a series termination for the sense wire when added to the input impedance of the preamplifier. The far end of each sense wire can also be terminated, although this might not be necessary (R2, C1 in fig. 3.7). A calibration input is provided for calibration and test purposes.

3.3.2 The Amplifier-discriminator Card

Signals from the wire preamplifiers will be brought out of the KLOE detector by means of miniature 50 ohm coaxial cables. Because the expected signals from single electron clusters have an amplitude of about 10 mV, they need to be amplified before discrimination. An analog signal is also provided for pulse height measurements.

Fig. 3.8 shows the block diagram of a single channel of the amplifier-discriminator circuit, details are given in fig. 3.9.

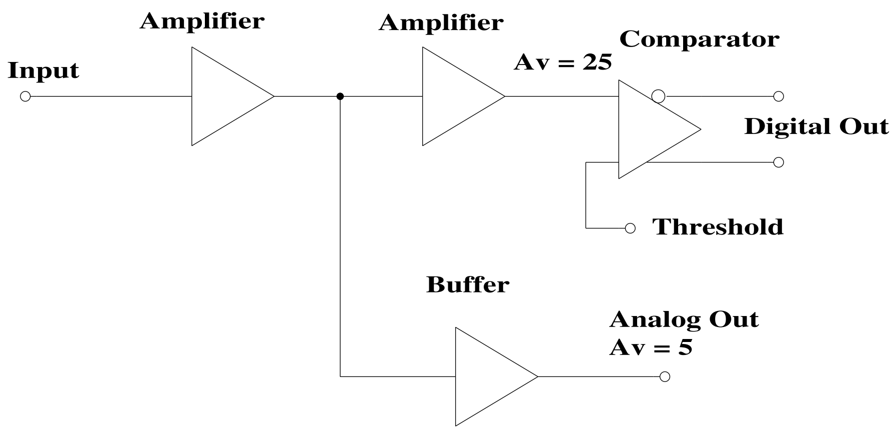

Fig. 3.8. Block diagram of one amplifier and discriminator channel.

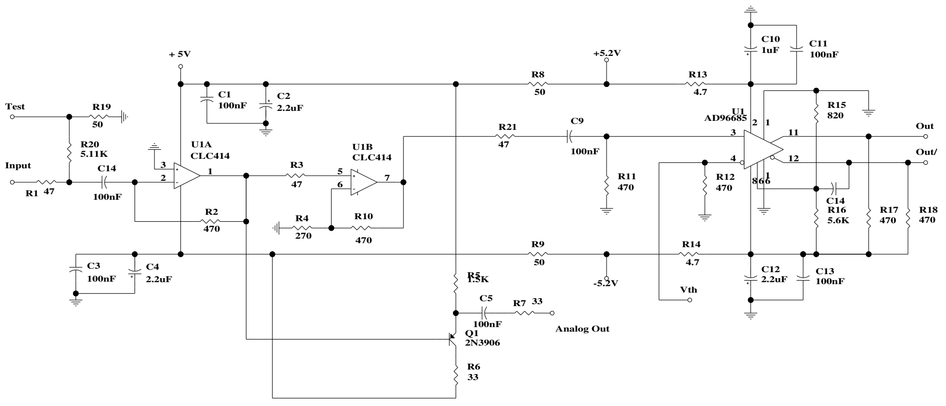

Fig. 3.9. Amplifier and discriminator circuit diagram.

The first stage is a $\times 10$ amplifier with low input impedance for transmission line termination. This stage drives both the analog output buffer and the comparator driving stage. The total voltage gain at the input of comparator is ~ 25. The comparator produces a differential ECL pulse whose minimum width is fixed to 100 ns. The threshold is adjustable through a 10 to 1 voltage divider. An input is also furnished for calibration/test purposes. Specifications for the amplifier-discriminator circuit are given in table 2.

Table 2. Amplifier and discriminator circuit specifications

Analog output voltage gain	5
Voltage gain (comparator input)	~ 25
Output risetime	~ 5 ns
Dynamic range (analog output)	~ 1.5V
Analog output	unipolar, 50 Ω
Digital out	diff. ECL, ~ 110 Ω
Calibration Input	one per card

Forty-eight amplifier-discriminator channels are packaged on a single 9U VME card together with circuits for threshold setting/sensing and the logic for addressing. Thresholds can be set for groups of 8 channels.

3.3.3 The Calibration-communication Card

An additional card in each crate of amplifier-discriminators is used for communication with the slow-control devices for threshold sensing/setting and calibration/test signal generation. Communication between the slow-control processor and the crates is via an RS232 serial link. The calibration circuit is used to supply calibration signals to the preamplifiers. A DAC controls the charge injected into the preamplifiers. Fixed amplitude signals for testing the amplifier-discriminators are also generated by this card.

3.3.4 High Voltage System

The proposed solution for the sense wire power supplies is shown in fig. 3.10. The requirements for the HV system are:

1. programmable voltage setting (0 - 2.5 kV, 1V resolution, master);
2. programmable hardware trip (50 nA - 1 μA, distributor);
3. programmable ramp down (master);
4. programmable ramp up (master);
5. voltage sensing (1 V resolution, distributor);
6. single channel on/off capability (distributor).

Because of the circuit configuration these characteristics are shared between the master and the distribution cards as shown in fig. 3.10.

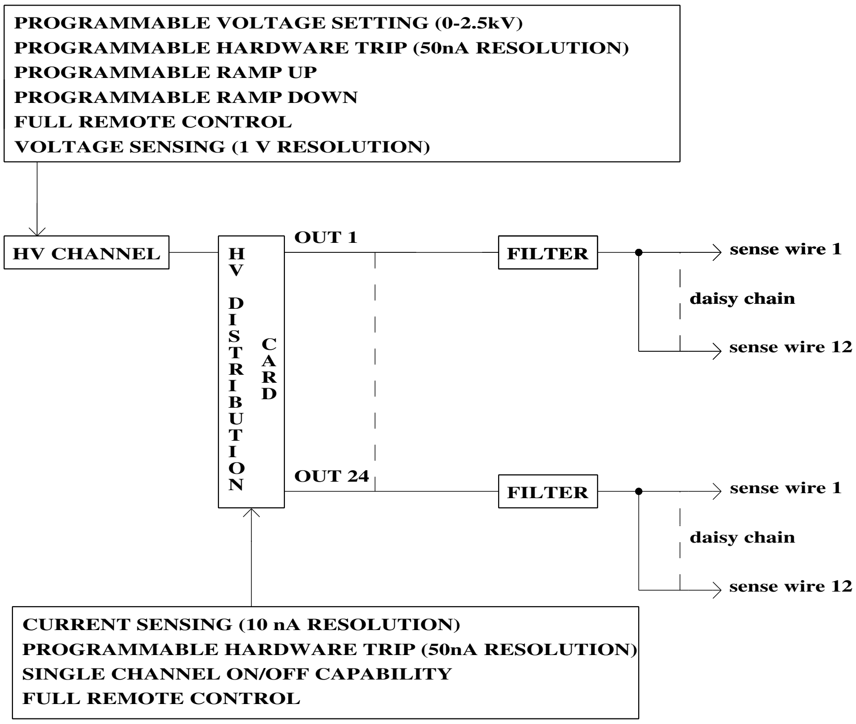

Fig. 3.10. High voltage system block diagram.

Each HV channel supplies a group of sense wires connected by a daisy chain (we expect to use a modularity of 6/12 wires per HV channel).

3.3.5 Chamber TDCs

The TDC chips for the KLOE chamber digitization will reside in groups of three (96 channels) on 9U VME boards. Most of the functionality, digitization and event buffering, is integrated into the VLSI chips. The modules themselves provide a mechanical interface to the KLOE first level crate standard and an electronic interface to the KLOE data aquisition with added functionality for testing.

The Chamber TDC modules (CTM) will receive the 96 differential input signals on a high density connector and receive the precision trigger timing and other control signals from the backplane. These signals constitute the input to the TDC chips. The digitized data stored in event buffers inside the chips can then be read directly via the VME and AUXbus interfaces or will be stored in an intermediate event buffer. Such an intermediate event buffer does not add much to the overall system costs and provides a good safety margin against buffer overflows and, most importantly for test and monitoring, additional random access functionality to VME. The usual readout path will go via the AUXbus interface and the ROCK, the VME interface is necessary for initialization, testing and monitoring. The present digital design will provide functions for extensive in-system-tests and prompt performance surveillance. Special care will be taken to insure the highest system reliability.

Fig. 3.11 shows the basic design of the digital board. The logic functions have been grouped in blocks and the major data paths are shown. There is an implicit decomposition into four decoupled functions:

1. the input section handles the high quality analog signals and the time-to-digital conversion;
2. the intermediate readout and logic section has a synchronous design for high speed and low switched currents to control;
3. the asynchronous bus interfaces to AUXbus and VMEbus;
4. the control and testing functions.

Most of the logic and data switching will be performed by a few Programmable Logic Devices, PLDs, to avoid large capacitive loads and to provide speed and flexibility for ongoing optimization of the protocols.

High density connectors at the front side of the module will receive the 96 signals from twisted pair cables. The differential ECL signals will be terminated and transformed to TTL levels to interface to the nearby TDC chips. The TDC chips will be mounted in SMD sockets. Power and ground supplies will be HF decoupled from the rest of the module. Adequate HF shielding will be provided for this part of the PCB. A TDC Controller will supervise access to the TDCs internal registers and event buffers. If there are valid data after a trigger, a block move to the intermediate memory will be requested and controlled.

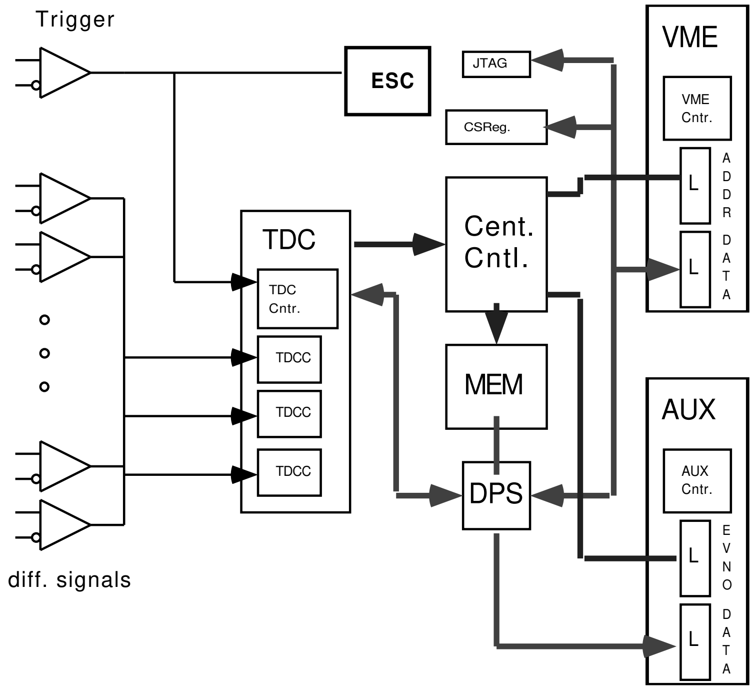

Fig. 3.11. TDC Board Block Diagram.

This intermediate memory may be configured as a 32 to 128 event buffer with FIFO type access, as well as random access from VME. The central controller organizes all data transfers in a pipelined way, arbitrating requests from the different interfaces and keeping track of the buffer structure of the intermediate memory.

The AUXbus interface consists of a PLD to receive the input signals and provide the protocol and set the latches for the data output. It will send an asynchronous request for the next data word to the Central Controller while processing the current word. The VMEbus interface has equivalent functions to the AUXbus interface (except sparse data scan) and in addition it provides access to the TDC chips and the control and testing sections. Instead of using a large data bus structure, all data transfers will be multiplexed by a high pin count PLD. This allows for high speed pipelined transfers and low EMI.

We plan to use boundary-scan test capabilities (JTAG) on the CTM. Proper interfaces and a microcontroller will be included, to allow for functional tests after production and power-on-self-tests at the board and system level. In KLOE one has a guaranteed dead time after each trigger. This time will be used to verify the proper synchronization at the system level from the central trigger fan-out and event counter to the TDC chips and local event counters. Busy and error messages are propagated back to a central supervisor. The degree of sophistication of this event synchronization will be finalized after the first prototype tests.

The measurement of the signal drift-time coming from the chamber is performed by a custom integrated circuit, fig. 3.12, which is under development by the INFN Sezione di Roma1. The circuit is a multichannel common start/stop TDC, with 32 channels per chip. In KLOE the drift-time measurement is done by recording, for each signal, its time of arrival with respect to a stop signal, which is common to all channels and generated by the trigger.

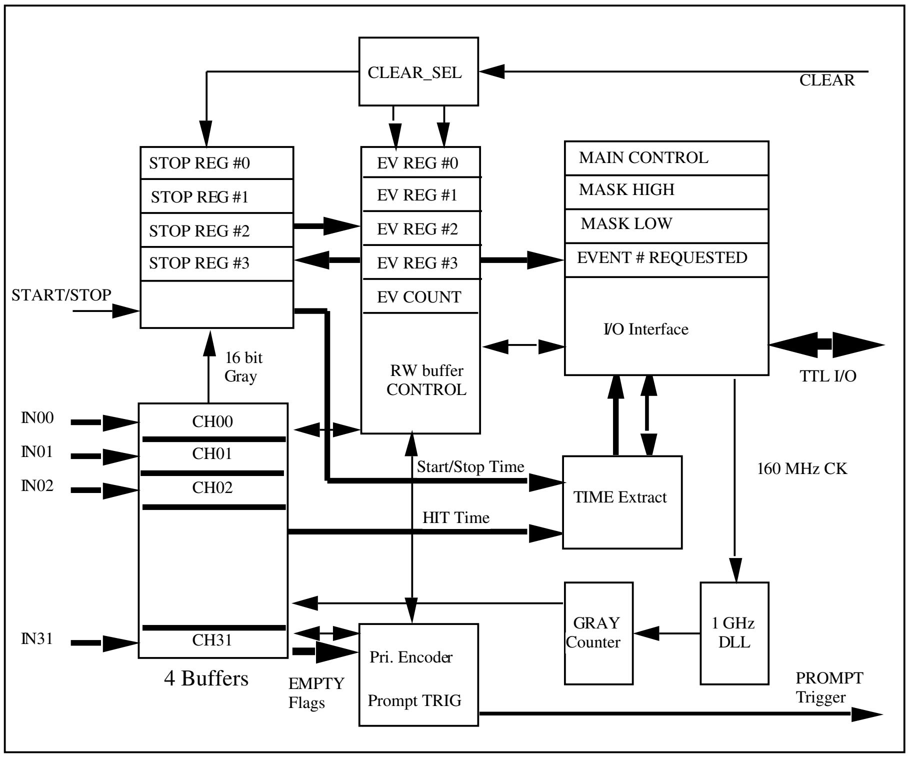

Fig. 3.12. TDC IC block diagram.

The TDC integrated circuit will be developed as a full-custom device in 0.5 micron CMOS technology. Its working frequency is a function of an external reference clock that must be supplied to the circuit and it can be as high as 1 GHz (1 nsec LSB).

The circuit is capable of detecting rising/falling edges, with a double edge resolution of 10 nsec or better. A programmable number of hits, from 1 to 16, can be stored for each channel. The hits are recorded as 16 bit words and every registered hit is stored in the chip for a programmable time interval; if during this time the chip does not receive a stop command the hit is removed, otherwise, the hit is kept for the read-out. The chip has also a multi event buffering capability, 4 events deep, which is used only if at least one hit is present in one of the 32 channels, in a time window associated with a stop signal. Dedicated prompt trigger output lines inform the read-out of the presence in the chip of data to be read. The read-out of the data passes through the I/O port at a maximum speed of 50 MHz.

The empty channels are automatically skipped during the read-out phase and for each registered hit the chip gives its absolute time value with respect to the stop signal and its channel number. Additional chip functions include an event counter; masking of the individual channels; and a self-test facility. Three of these chips will be mounted on a 9U VME board, for a total of 96 channels.

4. LEVEL 1 READOUT

Very high bandwidth data acquisition systems require high performance parallel buses which gather data from the front-end electronics and pass them to event builder engines and on-line processors. Software protocols and CPU driven operations may slow down the whole system even if state-of-the-art devices and communication channels are used. For the readout of KLOE we have chosen an architecture which relies on custom buses and hardware controllers, in order to achieve high bandwidth without software penalties and CPU overheads.

4.1 System Architecture

The proposed architecture is shown in fig. 4.1.

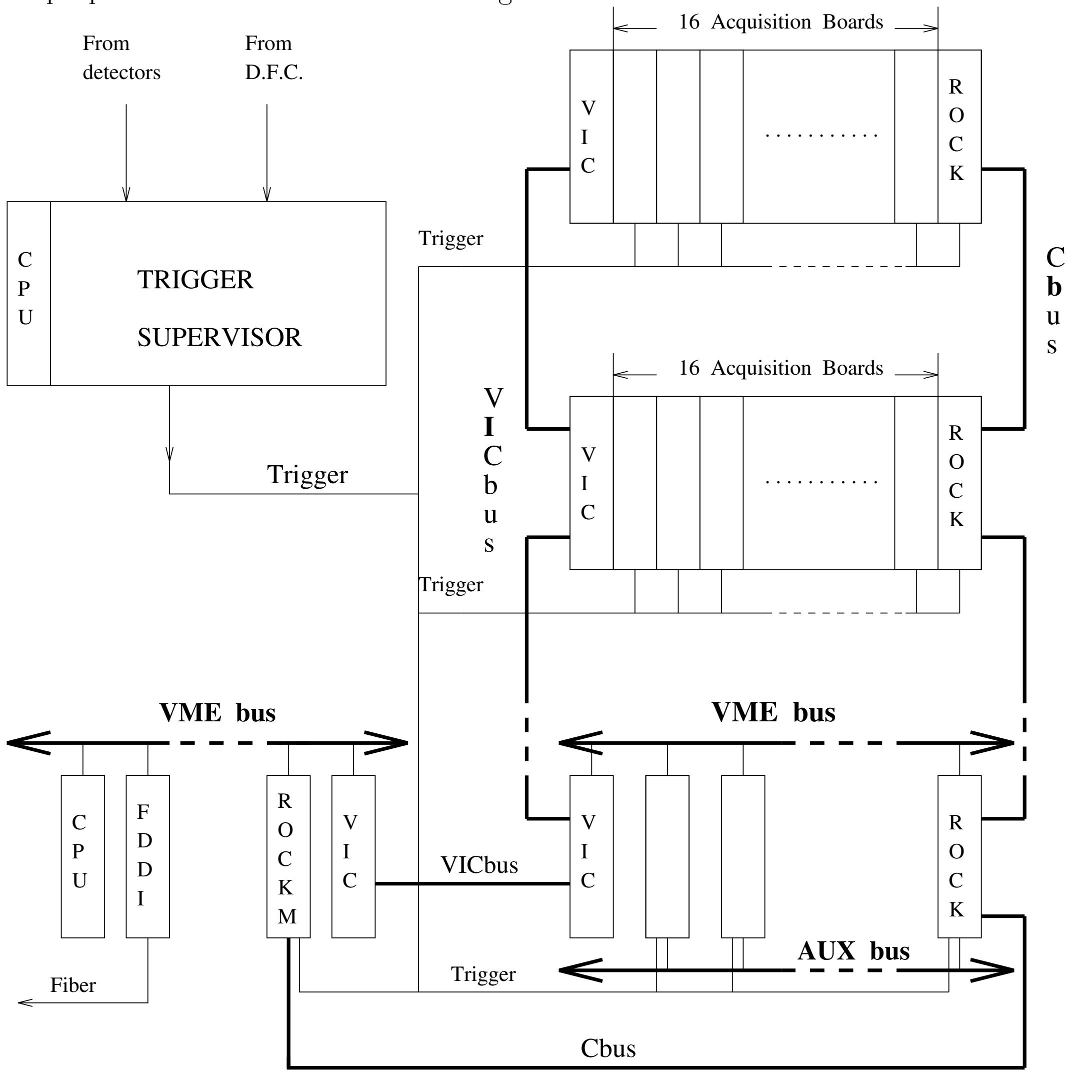

Fig. 4.1. System Architecture.

Each chain contains a number of VME crates, which hosts a VME master, acquisition modules and a Readout Controller for KLOE (ROCK). Through the VME backplane the VME master can access both the front-end modules and the ROCK. This controller is the only master on a custom auxiliary bus (AUXbus) which connects all the front-end modules. The structure and the protocol of the AUXbus are optimized to meet the requirements of high speed data transfer. In the scheme adopted the VME master initializes the whole system, runs diagnostic tests and, if necessary, samples events from time to time for performance monitoring of the system itself.

All these tasks can be performed by a VIC module connected through a vertical bus to a remote CPU. Alternatively a CPU board can be hosted directly in each front-end crate, sharing a common LAN environment. Although data acquisition operations are fully controlled by ROCKs, CPU power is still required to perform system level operations. Tuning of CPU boards allocation into the system is hence a key point to achieve all the benefits of the ROCKs dual-port architecture.

All the ROCKs of a chain are connected through a single master cable-bus, called Chain-bus (Cbus), which is driven by a ROCK Manager (ROCKM). This last module is housed in a VME crate together with a high performance CPU and a high speed I/O board for transferring data to the event builder.

4.2 Protocols Overview

The read-out system is driven by the trigger number which is just a local count of the received trigger signal. The trigger signal will be distributed to the front-end boards, the ROCK and the ROCK Manager. The front-end data acquisition boards will then associate their data with a count of the trigger signal. A mechanism for trigger number synchronization check is also implemented in the ROCK by means of a special cycle on the AUXbus.

Each acquisition board is designed to respond to a trigger number request by sending all the data associated to it onto the AUXbus. Readout data are automatically associated to the channel number inside the module. The ROCK drives the readout cycle on the AUXbus in two steps: broadcast and transfer. These operations make use of additional lines or buses defined in the following.

During the broadcast cycle, fig. 4.2, the ROCK puts on the TRbus lines a trigger number forwarded asynchronously to the whole system by the trigger supervisor of the experiment. The boards that contain data associated with that trigger number assert a line on the BRDbus. The slowest board then validates this bus with the BK line. As a result, the ROCK records the BRDbus pattern by means of a single transaction synchronous with the BK assertion.

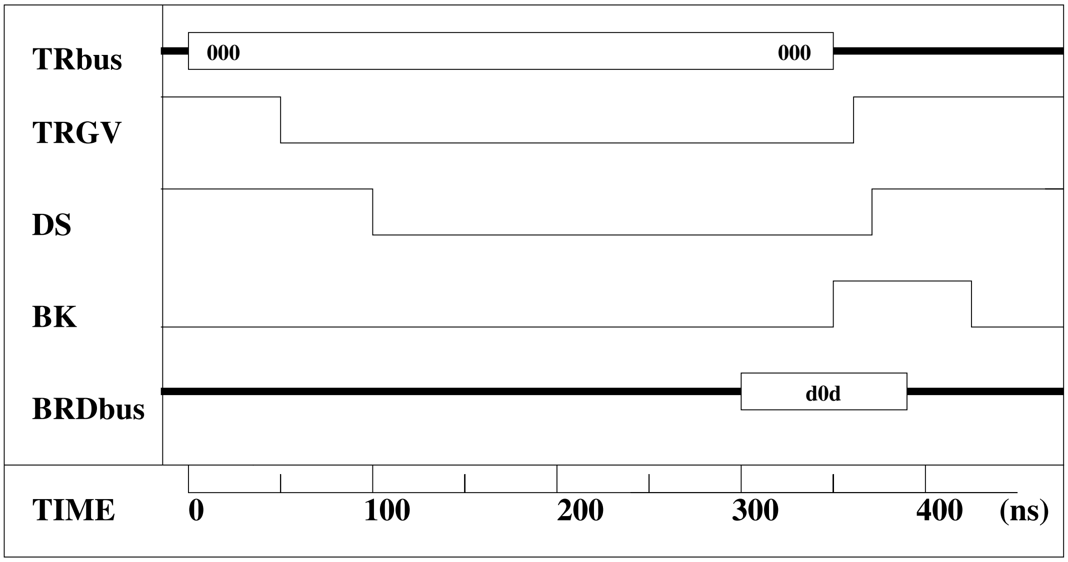

Fig. 4.2. Broadcast cycle.

The broadcast cycle then proceeds with the identification of the modules to be read and the preparation of the table containing their addresses (sparse data scan). In the following transfer cycle, fig. 4.3, the ROCK addresses successively the modules to be read. Data transfer comes in blocks for each module using an asynchronous VME-like protocol. An EOB line is used to flag in hardware the end of a block data transfer. This allows block transfer operations of arbitrary length, overcoming the VME limitations.

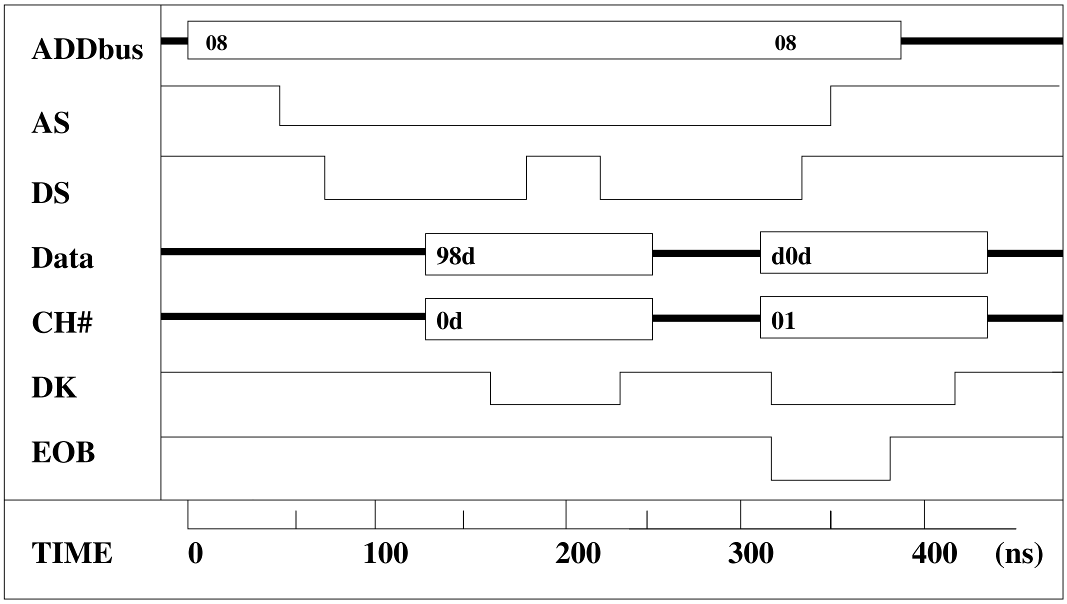

Fig. 4.3. Transfer cycle.

Inside the ROCKs, data are organized in frames beginning with a trigger number. Data words collected through the AUXbus from the acquisition modules follow. The frame is then terminated with a control word (End-Of-Frame). The frames are queued into a dual port FIFO buffer, accessible from the intercrate link. The ROCK Manager controls a chain of ROCKs through the Cbus in a token-ring fashion. Its readout cycle starts by presenting and validating a trigger number on the Cbus lines, then a token is placed onto the bus. The ROCK nearest to the master catches the token and then start the synchronous transmission of the data associated with the given trigger number. The end of the data is flagged with the EOF control word. The token is then passed to the next ROCK in the daisy-chain. This procedure is repeated until all the ROCKs in the chain have sent either the data related to the processed trigger number or an empty frame, if no data are present. Traffic on the AUXbuses and Cbus are fully decoupled by means of FIFO buffers to handle momentary increases of trigger rate. The data frame inside the ROCK Manager is similar to that inside the ROCK but contains, in addition, the crate address.

ROCK and ROCKM boards handle, in hardware, the most common error conditions under the supervision of the data flow controller. Failures of the trigger number synchronization verification, busy status, or hang-ups on the AUXbus as well as on the Cbus are reported on dedicated connectors available on the boards' front panel. The busy signal generated by the ROCK is the logical OR of the busy flags of the acquisition boards in a crate. This signal is asserted and deasserted asynchronously, depending on the buffer status of the front-end modules. Busy conditions only arise from malfunctions and are not present under normal operating conditions, as stated in the overview, section 2.1.2. The trigger signal will be disabled under such condition. Unrecoverable errors, such as a trigger number mismatch or a protocol violation, put the controllers in an idle state which inhibits the acquisition boards through a dedicated line both on the AUXbus and the Cbus. Error tracing can be performed using the VMEbus under the supervision of the data flow controller, which also manages the operation restart sequences.

4.3 Simulations

The architecture and protocols presented above have been modeled and simulated at the behavioural level using Verilog-HDL.[6] The timing of the transactions on both the AUXbus and the Cbus is simulated by assuming the use of off-the-shelf TTL components operating at conservative speeds. At present the DAQ layout proposed for KLOE[7] foresees 5 crates/chain for the electromagnetic calorimeter (with 16 ADC or TDC boards/create and 30 channels/board) and 2 crates/chain for the drift chamber (with 16 TDC boards/crate and 96 channels/board). Slightly different and more demanding values were used in the simulation: 8 crates/chain and 32 channels/board were simulated for the calorimeter and 4 crates/cain for the drift chamber.

The simulation of the activity on the AUXbus was done for an increasing number of words randomly distributed in the crate. The time the ROCK needs to complete a readout cycle increases linearly with the amount of data to be transferred, as shown in fig. 4.4.

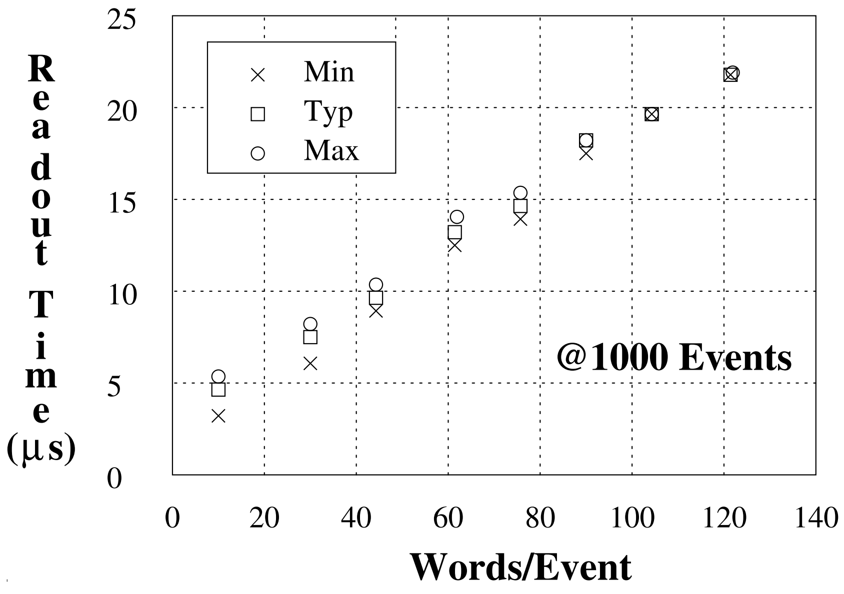

Fig. 4.4. AUXbus simulated performance.

The analysis of readout timing for the AUX bus and Cbus entire chain of 8 crates has been carried out from 60 to 480 words/chain randomly distributed in the system. 1,000 events have been simulated for each occupancy value. The typical acquisition time for a 10 kHz trigger rate, assuming 80 words/event in a EM Calorimeter chain, is about 35 μs as shown in fig. 4.5.

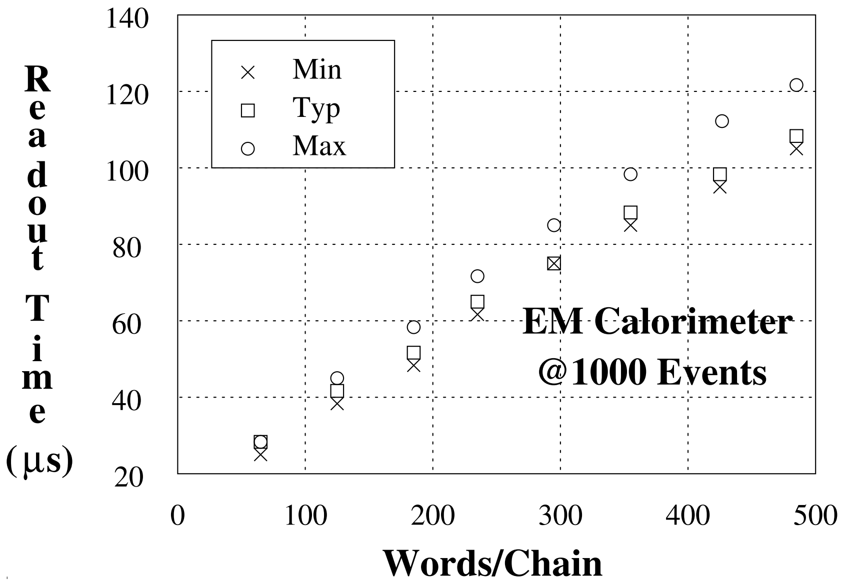

Fig. 4.5. Cbus simulated performances for EM calorimeter.

The simulations were also done for the central tracking chamber with chains of 4 crates. Each of these crates houses 16 TDC boards with 96 channels each. The typical acquisition time for a 10 kHz trigger rate, assuming 120 words/event in a tracking chamber chain, is about 30 μs as shown in fig. 4.6. The assumption that data is randomly distributed throughout the system is a worst-case condition. Real data are clustered, resulting in shorter read-out times. The simulation results were presented at CHEP94.[8]

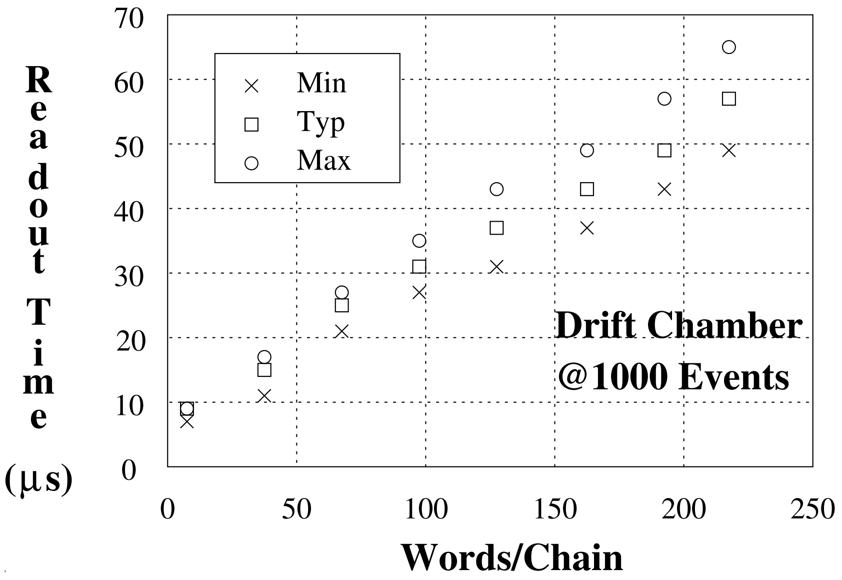

Fig. 4.6. Cbus simulated performances for tracking chamber.

4.4 Conclusions

The introduction of auxiliary buses in a DAQ scheme relaxes the requirements on the CPU and makes it possible to use VME as an additional channel for monitoring and diagnostics.

Data transmission is driven by hardware processors using reliable protocols developed specifically for KLOE. A hardware implementation of the architecture described is under development in Naples. The AUXbus controller is based on a XC3164 XILINX gate array[9] while the Cbus interface fits in a MACH210 AMD device.[10] The AUXbus (64 lines) shares the VME/J2 connector. The Cbus uses a 50 twisted-pair cable. Prototypes of both the VME/AUXbus custom backplanes and Cbus cable segments have been already designed and successfully tested. In the KLOE experiment, at the 10 kHz expected trigger rate, the DAQ system will be able to handle a 15 Mbyte/s data bandwidth per chain, assuming 32-bit words and a maximum occupancy of 400 words/chain.

5. DATA FLOW CONTROL AND EVENT BUILDING

This chapter covers the parts of the KLOE data acquisition system that concentrate sub-events collected by the ROCK Managers, ROCKMs, and distributes them to a farm of processors, where complete events are built and analysed. In contrast to systems described in previous chapters, this system is built using commercial products for both hardware and communication software. This choice offers greater portability and consequently the ability to implement heterogeneous configurations. The configuration proposed in the following takes into account the results of the measurements we made with commercial products available today.

5.1 The Level 2 VME subsystems

This DAQ component implements the second level of data concentration in the VME crates. Data collected by the ROCKs in a front-end chain are stored in local FIFOs and read by the ROCKM one event at a time using the Cbus, as was described in chapter 4. So a complete sub-event is temporarily stored in the ROCKM's FIFO. Strings of sub-events are periodically read by the CPU in the crate, through the VME bus, and buffered in its own memory.

The system will contain about 10 such crates, each of them controlling 1 or sometimes 2 front-end chains. An average throughput of 5 Mbytes/s must be supported in each crate while transporting, through an FDDI switch, the sub-event data from the CPUs to the farm. As an example, fig. 5.1 shows the L2-VME configuration collecting data from the calorimeter.

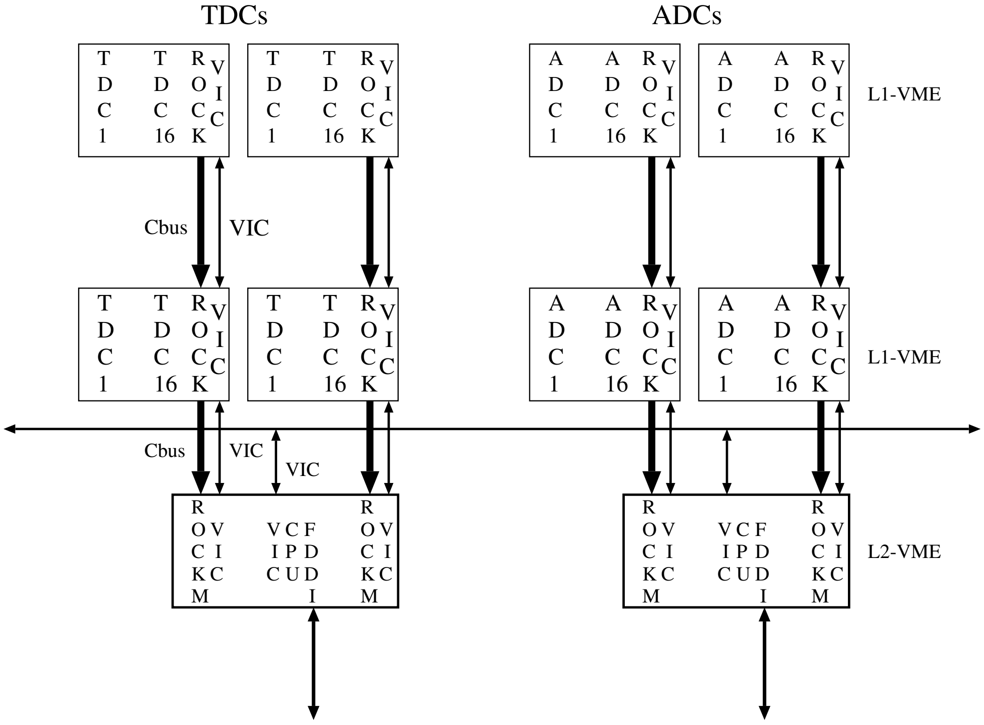

Fig. 5.1. Calorimeter read-out.

5.1.1 *Hardware configuration*

Each level 2 system will be housed in 1 standard 6U VME crate containing the following VME boards:

- A high performance CPU (in terms of network processing and VME access capabilities);

- 1 FDDI network interface offering a nominal throughput of 100 MBits/s;

- 1-2 ROCKMs;

- 2-3 VME interconnecting boards (VIC, from CES).

One VIC interface is connected to a L1-VME chain allowing the CPU to program, check and debug the FEE. A correct balancing of the load is obtained by chaining the FEE crates as outlined in the following.

The calorimeter produces about 1 kbyte/event requiring two L2 crates. The 10,000 electronics channels (5,000 ADCs and 5,000 TDCs) require about 20 FEE crates. Each L2 crate will read 2 Cbus chains containing 5 FEE crates each. The tracking chamber produces in average about 3 or less kbytes per event. This requires 6 L2 crates, each connected to 2 FEE crates on 1 Cbus chain each. The trigger produces less than 1 kbyte/event requiring a small number of VME crates and 2 communication channels managed by 2 L2-VME crates. This topology leaves enough contingency both in interconnectivity and data throughput capacity for unexpected increase of the bandwidth requirements.

5.1.2 Functionality

The first task of the L2-VME subsystem is to provide programmable intelligence to the FEE crates, which are only equipped with a hardwired sparse read-out scanner and a VIC board. 1 CPU provides enough CPU power for a whole chain of FEE crates. The tasks include, for example, programming the zero suppression functions (thresholds and pedestals) and standalone calibration. Even part of the slow control might be managed by this CPU. The second task, during data acquisition, is to read data from the ROCKM, repack them into a *string of sub-events* and ship them across the fiber or copper links, and the switch, to the correct CPU in the farm. The association between event numbers and destination farm is provided by the independent Data-Flow-Control (DFC) systems that communicate across a dedicated VIC bus with the L2 crates.

The protocol used by the ROCKM is extremely simple. The ROCKM appears as a FIFO to the processor and a dedicated Block Mover can handle most of the data transport. To communicate with the rest of the DAQ components (e.g. farm, run control system) we plan to employ a standard high level, reliable protocol such as TCP/IP running over FDDI channels. This task is demanding both in terms of VME bus bandwidth and processing power for the protocol handling. This protocol can run over other physical media like HIPPI, ATM, and thus makes the system independent of different network choices.

5.1.3 Requirements and results of measurements

Starting at an integral throughput of 50 Mbytes/s for the whole system, each L2-VME crate has to handle a throughput of 5 Mbytes/s. Because repacking of the data is inevitable this leads to a required VME throughput of 15 Mbytes/s, which is possible with today's commercial processors. The network output of 5 Mbyte/s has almost been achieved with commercial products (software and hardware). We have measured bandwidth of 6 Mbytes/s using TCP/IP and 7 Mbytes/s using UDP. The latter however leaves data consistency checks and error recovery to the calling application. We hope to improve the TCP/IP results to the measured UDP performances using new interfaces implementing DMA access to the VME bus.

Unfortunately the processors delivering the best VME transfer rate perform poorly on the network and vice versa. We are in contact with industry to find a product that satisfies both requirements. Because of the increased usage of PCI and other proprietary buses on VME processors, "on board" network interfaces (FDDI, HIPPI or ATM) will be available on mezzanine cards by the middle of 1995, further decreasing the VME transfer requirements.

5.2 Data Flow Control

The DFC component of the data acquisition system is responsible for the unique assignment of a set of sub-events (*sub-event string*) coming from different detector chains, and a given SBC in the farm system, where all the *sub-event strings* coming from different parts of the detector will be packed to produce complete events at the last step of the event-building process.

The DFC will be executed on a CPU with VME interface, connected through VICs to the L2 crates, and to the switch through an FDDI channel (see fig. 5.2 where the complete DAQ architecture is presented).

At any L2 crate, a *sub-event string* is defined to contain about 100 sub-events, corresponding to an average acquisition time of about 10 ms. It also corresponds to an average number of 50 kbytes buffered in the CPU memory, to be sent in a unique transmission operation through the switch at a maximum average required throughput of 5 MBytes/sec. Long packets must be used for good TCP/IP protocol performances.

The FDDI switch we are using at present, the DEC GIGAswitch, works as a bridge; it buffers the packets coming in independent input memories and passes or retransmits them to the destination port when the path is free. The final destination has to be known before the packet leaves the L2 CPU.

5.2.1 DFC Protocol

The assignment of events to an SBC is obtained cyclically from a static local table that is copied by the DFC into the L2 CPU's memory at appropriate times. This table contains the list of all the SBCs, labelled with a flag indicating its temporary availability. The SBCs order in the list takes into account that each switch port is connected to a crate of 10 SBCs. Sending for a long time to the same output port is then prevented. The internet standard protocol SNMP (Simple Network Management Protocol) will be used to build the table automatically and for general monitoring.

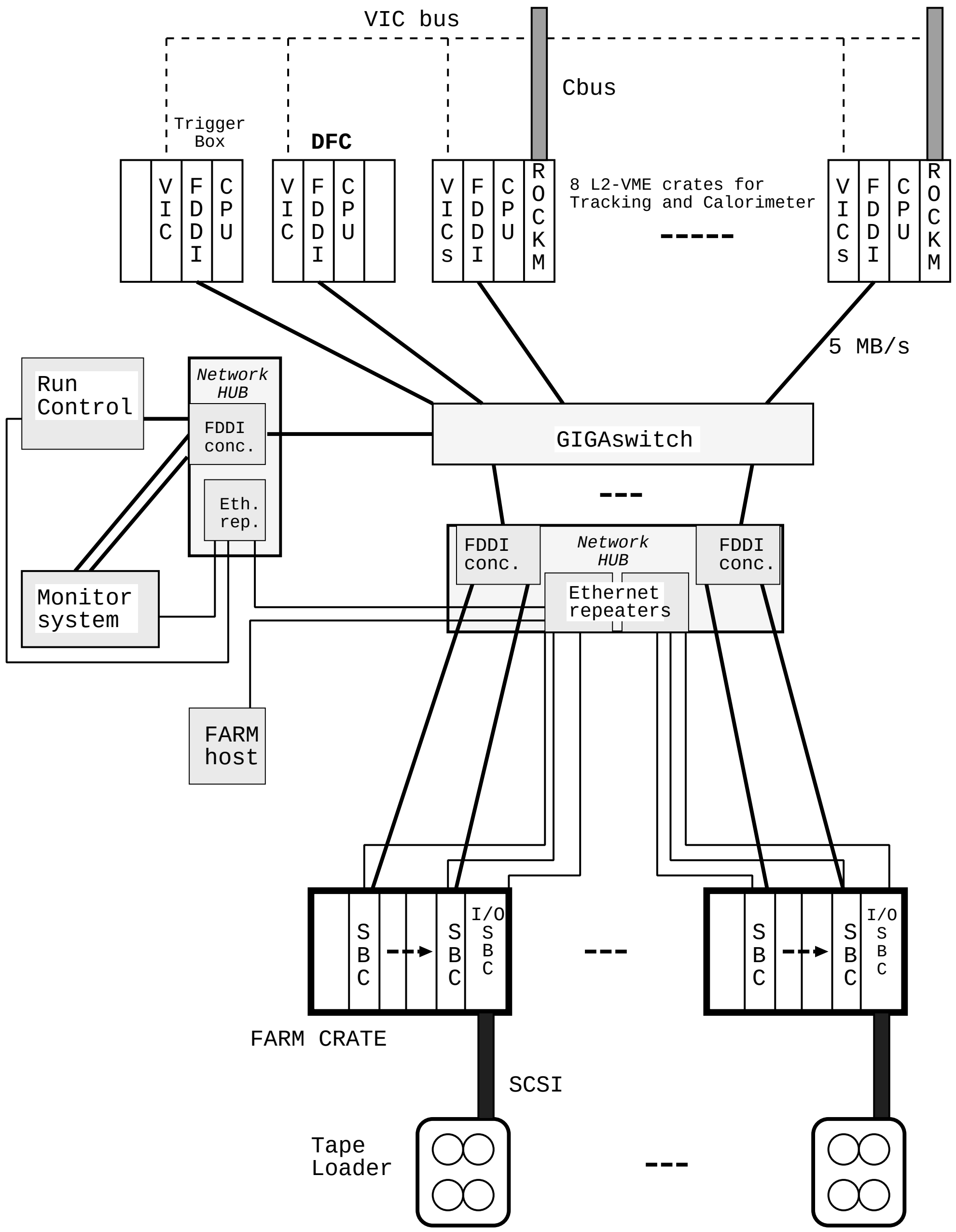

Fig. 5.2. DAQ system architecture.

The table is updated by the DFC in one of the following conditions:

- an SBC crashes. The DFC, being informed by the TCP/IP protocol mechanisms, labels the SBC as unavailable. This condition remains until the SBC reopens the connections with the DFC.

- an SBC sends a message to the DFC indicating the *'input queue full condition'*. This signal is generated when the occupancy of the queue exceeds the 70 % of the total availability, due to one of the following conditions: the previous *sub-event string* did require a very long CPU time or an input queue full condition was produced. The SBC will be labeled as unavailable until the SBC sends the message indicating *'availability'*, corresponding to an occupancy of queue less than 30%.

After any change, the new table is copied by the DFC to a secondary table in every L2 CPU, in order to synchronize the replacement of the old table with the end of globally defined *sub-event string* transmission.

The DFC does not detect misalignment in the assignment between *sub-event string* and SBC. Misalignment are detected at the farm level, from where an error message is sent to RunControl which will initiate a *'flush and restart'* operation.

5.2.2 Time schedule for completion of tests

We plan to complete the study of protocol implementation and performance by June '95.

5.3 Event Building

Event building in the KLOE data acquisition system is implemented at three different levels. At the L2 crate, sub-events from a detector segment are individually assembled by the ROCK Manager. The CPU copies their buffers in onboard memory. Many sub-events are buffered into *sub-event strings* before transmission via FDDI using the TCP/IP protocol. *Sub-event strings* corresponding to the same set of trigger numbers are sent through the switch to a single SBC. This is accomplished by assigning the same destination address to all related packets. *Sub-event strings* received by a given SBC, are subdivided into single sub-events to be reassembled to complete events subsequently.

5.3.1 Implementation

As already mentioned, we propose to implement the first setup with FDDI channels, to physically interconnect the different components requiring high throughput, an FDDI switch (GIGAswitch), to obtain the maximum possible throughput in a scalable way, a high performance CPUs both on VME and farm, well matched with the network obtainable throughput's. The TCP/IP high level protocol will be used for message sending. All the processors of the DAQ system will be interconnected by Ethernet in order to exchange message and data that require a lower throughput.

There are however some limitations in the use of high speed communication channels in a crate. These are partially due to the fact that the communication channels are implemented on separate boards. Communication with the CPU uses the VME bus which limits to 40 Mbytes/sec the maximum throughput using D32 VME. D64 VME, which could offer better performances, is not at present available. A new generation of "on board" communication channels will be available by middle '95 together with TCP/IP protocol implementation freeing the CPU for other work.

The proposed software configuration can still be used if new very high speed channels, like HIPPI, ATM, FCS, could replace FDDI allowing a reduction in the number of parallel channels. That should be possible already next year. The characteristics of the GIGAswitch and of the already existing DEC ATM switch are given in table 5.1. In the same table the TCP/IP throughput obtainable in a DEC workstation using both protocols is also shown.

Table 5.1. Comparison between ATM and FDDI switches from DEC

	GIGAswitch	ATM switch
1 Chan. throughput	100 Mbps	155 Mb/s, 140 Mb/s data or 622 Mb/s, 560 Mb/s data
Ports	≤22, 2 ports/board or ≤34 with 4 prts/brd, ATM or FDDI	≤52 with 4 ports/board
Slots	11	13
Total throughput	≤3.6 Gbps	≤10.4 Gbps
Switch	by packet	by cell 48/53 bits
Queues	input and output	input
Queues services	FIFO	first free path
Port reservation	none	Credits mechanism, eliminating retransmission
TCP/IP	~90% @ 11Mbytes/sec	~85% @ 16Mbytes/sec

5.4 Test of components

Tests on the GIGAswitch installed at LNF last year and on Alpha-CPUs (AXP 3000/500 and AXP 4000/612) have been given.[12−14] The conclusions were that the combination of the Alpha CPU and the FDDI interface allows the use of TCP/IP protocol with a throughput greater than 90% of the nominal channel capability, if large packets are transmitted and the default window size is increased, see fig. 5.3.[15] Furthermore, while the performance is not increased for packets larger than 10 kbytes, the CPU usage is slightly reduced when the packet size is increased to the maximum TCP/IP buffer size, That is a value between 56 and 64 kbytes, depending on the system. We thus propose to transfer string of sub-events containing 50 kbytes, on average.

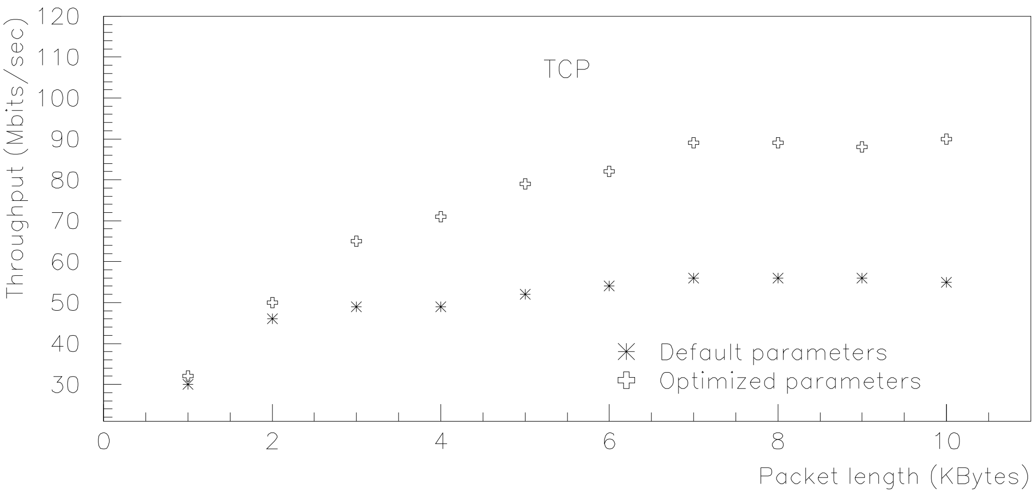

Fig. 5.3. Performances obtained with an Alpha-CPU using TCP/IP protocol.

The GIGAswitch does not appear to limit the communication between two Alpha machines working both in single and full-duplex mode. This was checked connecting two Alpha stations back to back as a two nodes network. In this case, DEC machines configure themselves in a full-duplex FDDI network where the token is no more required and the two fibers are used to transfer concurrent traffic in the two directions. Inserting the GIGAswitch in the path between the two stations, also connecting them to different boards in the switch, the throughput does not change, even when concurrent traffic between other ports is created. Multiple connections from a single node to different nodes offer a throughput that is always compatible with the sum of the single connections.

Searches for the best software algorithms to be used in implementing the communication protocol between the event-building components, which require many simultaneous connections during data taking have been performed.[16] Several hundreds lines of code has been written in ANSI C language to study these algoritms, and to check the portability to the different platforms we have actually under study (OSF/1, VMS, HP-UX, HP-RT, Lynx). This portability is extremely important because it will allow good comparison of performances and the migration to different processors at almost any time of the development work.

5.5 Event Building simulations

At the beginning of the KLOE DAQ studies, the event flow through the switch was simulated using a program written in Verilog-XL and C, running on a SUN station,[12] simulating a 22-ports GIGAswitch, maintaining 11 parallel connections at a time, at the nominal maximum throughput. The time required to switch configuration was 10 μs according to DEC specifications.

A first simulation was done to study the behaviour of the system with a total data flow of $\sim$120 Mbytes/sec, much greater than the foreseen one. In this simulation, the option of reading two or three hits per chamber channel was included.

Triggers were generated according to an exponential distribution with an average time between events of 0.1 ms. Sub-events were generated and collected in packets (*sub-event strings*). The length of the sub-events follows a gaussian distribution and the data are randomly distributed through the crates. A DFC station assigns a farm destination to every trigger or to trigger groups, sending this information through a dedicated port in the switch after receiving the trigger information. This approach has been further changed. The DFC runs now asynchronously with respect to the trigger signal and it is only stimulated by the "start of run" and SBC's queue status messages. Packets are sent to the input queues of the switch; then through the output queues to the destination farm. Every port in the switch has 4 Mbytes local memory for program and buffers, large enough with respect to our requirements.

The relevant parameters in the simulation are the number of input ports dedicated to the chamber's and calorimeter's data (*ndetch, ndetcal*) (the number of output ports dedicated to the farms will be then $nfarm \geq ndetch + ndetcal + 1(DFC)$), and the number of sub-events contained in each packet ($nevbuf$).

The results of this simulation are shown in fig. 5.4, where the maximum number of packets in the switch's input queues is given as a function of time for different values of $nevbuf$. For an event length of 1.5 ± 0.25 kbytes from the calorimeter and of 7.5 ± 1.25 kbytes from the chamber, we need $ndetch > 6$; infact, as shown in fig. 5.4a, if $ndetch = 6$ the system is unstable for each value of $nevbuf$. On the contrary, if $ndetch > 6$, the system is always stable, provided $nevbuf \geq 2$, fig. 5.4 b and fig. 5.4 c.

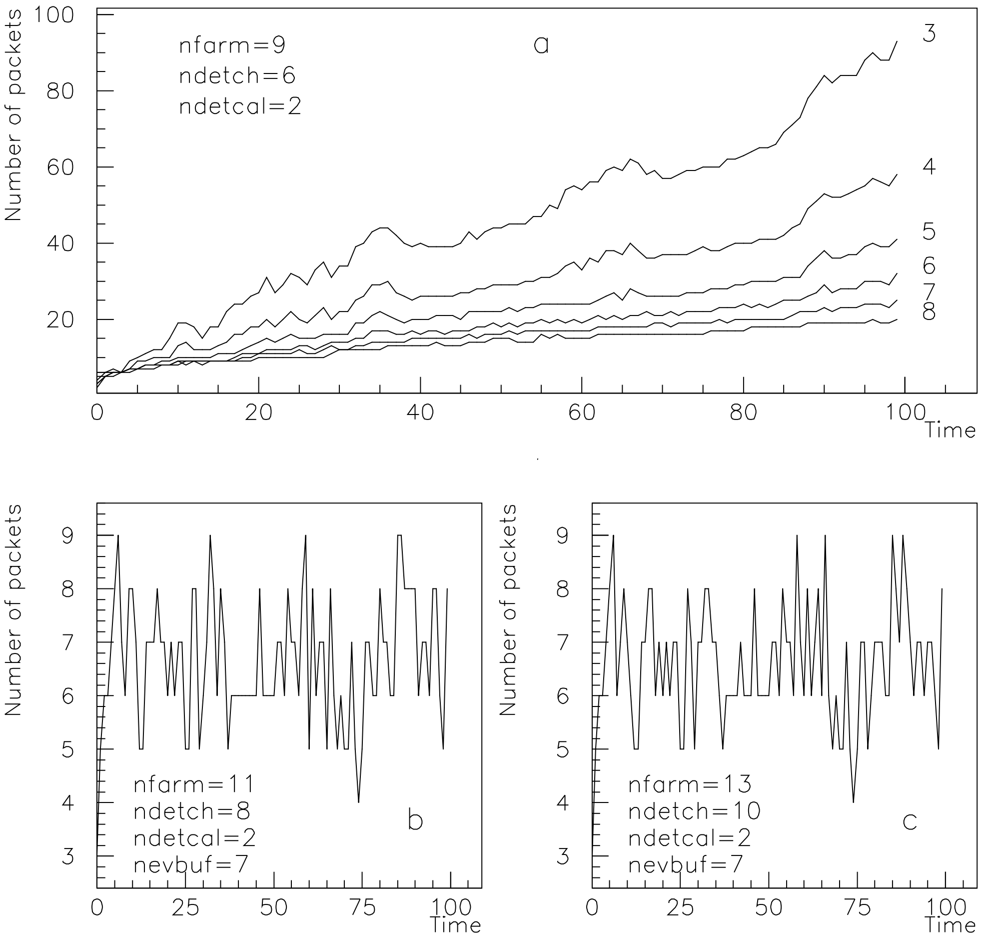

Fig. 5.4. Maximum length of the input queues.

The ratio between the transmission time through the switch and the total time is $\sim 73\%$; this is to be compared with the expected value for the same number of separated physical channels ($\sim 72\%$), showing that the switch is not a bottleneck, loosing only less than 2% of the theoretical throughput.

A more realistic simulation has been done using Monte Carlo generated events, in particular an interface between GEANFI[17−20] and the drift chamber structure was written to produce the right number and distribution of the hit cells. The amount of data is lower in this case, and the system is always stable, fig. 5.5.

In this case, however, the usage of the switch is not optimized because we had small packets and the transmission time became much less than the inter-trigger time. When an event is in the input queues, all the input ports try to transmit data to the same output port and the parallelism is lost. This generates an increase in the latency time for sub-events waiting for transmission, see fig. 5.6.

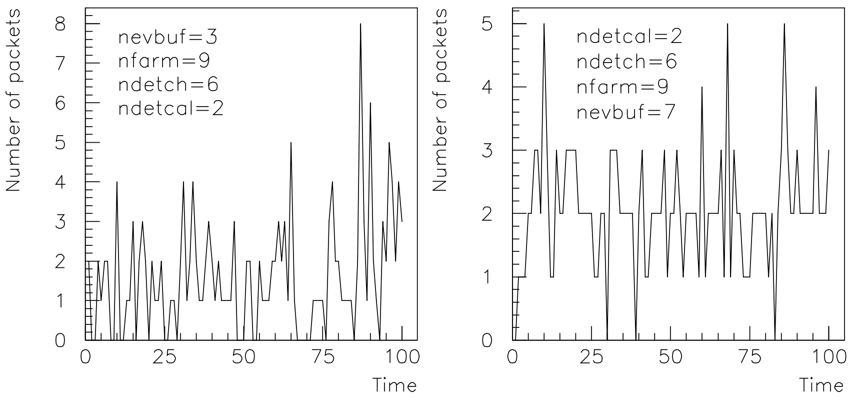

Fig. 5.5. Maximum Length of the input queues with low data flux.

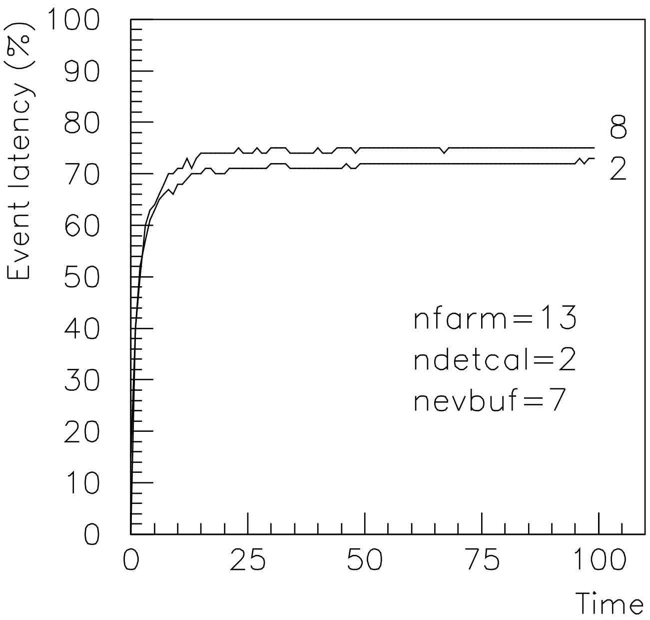

Fig. 5.6. Event latency.

This situation could be avoided using random-generated transmission sequences to every input queue. A number of packets less or equal than the number of the farms is collected and transmitted in a different order according to different sequences to each input port. In this way, both the order of the events and the parallelism of the switch are preserved and the latency time decreases (fig. 5.7). This mechanism is actually implemented in the ATM switch from DEC where the input queues are managed according to the first free path. In any case, this is not really a problem for GIGAswitch event-building applications because the switch's parallelism is always used when the traffic increases.

These simulations will be repeated with the first DAQ system prototype.

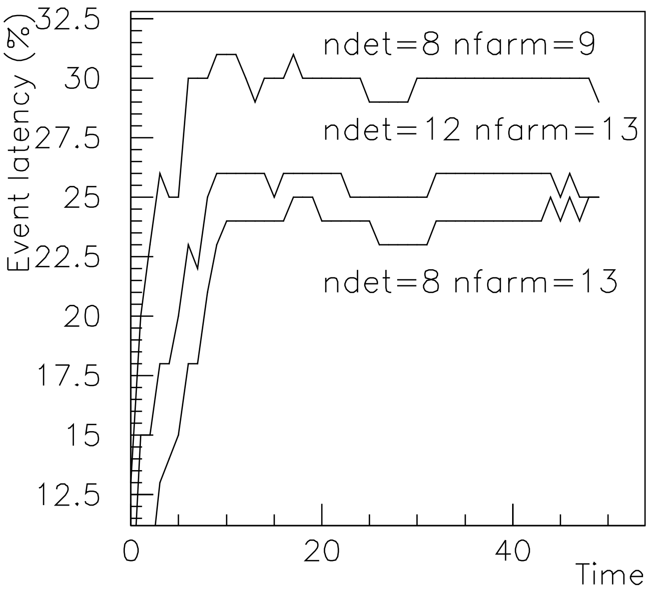

Fig. 5.7. Event latency after randomization.

5.6 Hardware configuration of the final DFC and Event Building systems

No assumptions are made here about the final hardware implementation, waiting for the announced new interface prototypes. An evaluation will be given taking into account the actual prices for FDDI interfaces and switches.

6. ON-LINE FARM

The functions of building and checking events, and book-keeping and preparing data for writing on tapes are performed by a powerful on-line farm composed by crates, each containing single board computers, SBCs, and one or more boards dedicated to output functions (I/O SBC), as presented in fig. 5.2.

Sub-event strings related to the same set of trigger numbers is received by a SBC via FDDI. The CPU builds and checks events and sends them to the I/O SBC. This latter eventually partially reorders the sequence of events and sends them to the storage system.

SBCs send information to the DFC in order to update its local availability table; send statistic, bookkeeping information, and complete events to the run controller, and to the monitor and display systems; generate error messages (e.g. misalignment, incomplete events, noisy channels); receive from the run controller commands addressed to the farm (e.g. start, stop, flush, requests for event sampling). The aggregate CPU power needed for all these tasks is evaluated to be 16,000 Specint'92.

During the initial shakedown period, the farm will be used to check the event integrity, to produce calibration data, and to perform partial event reconstruction. Full event reconstruction could become feasible on-line at a later stage. To achieve the necessary processing power, the on-line farm could consist of a set of crates containing SBCs, running a diskless, downloadable operating system. Commercial boards which can operate on VME, Futurebus+, or other proprietary buses and with the required CPU power are starting to appear in the market. Real time operating systems, such as OSF/1 RT, or diskless ones, such as VxWorks and Lynx, offer good time response.

6.1 SBC+: an Alpha 21064 based SBC

In this section will be described, an SBC, called SBC+, that is under development as a joint project between INFN and DEC, outlined in fig. 6.1. While this work is not specific to the KLOE detector, it is of course of great interest to us. The SBC+ performance will be compared in 1995 to other commercial solutions.

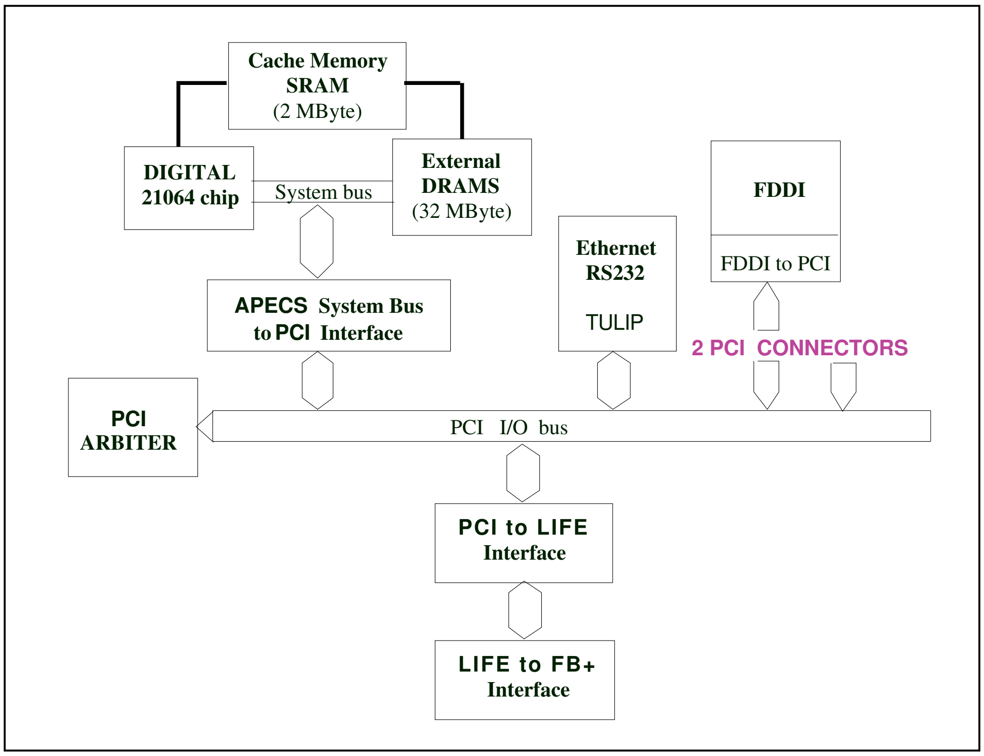

Fig. 6.1. SBC+ architecture.

SBC+ is designed around the DEC Alpha 21064 chip, which features up to 200 MHz clock, has a 64 bit architecture with dual-pipelining. SBC+ will have 2 Mbytes of secondary caches, at least 32 Mbytes of memory, and interfaces to Futurebus+, FDDI, Ethernet, RS232 and SCSI. The present version of the 21064 chip is pin-compatible the EV 4.5 version which will work at higher clock (up to 300 MHz) and will have a larger internal cache. PCI bus with two connectors will be used as input/output bus by SBC+. That will give to the board some flexibility in the communication channel: in fact in this way the PCI FDDI interface could easily be substituted by another PCI interface implementing an other communication protocol (e.g. ATM, SCI).

The LIFE chip by Newbridge Microsystems, that implements the Futurebus+ protocol, offers a nominal throughput of 60 Mbytes/s. The interface between PCI and the LIFE internal bus is under development and will be implemented in ASIC but any other suitable solution could be used in the future.

The design of SBC+ has been done in Verilog-XL using both already existing models (e.g. the 21064 chip, the PCI behavioural model, the structural model of the LIFE chip) and ad-hoc written models for memory and the PCI/LIFE interface.

The SBC+ will run one of the DEC real-time operating systems (VxWorks for AXP or OSF/1 with real-time extension), adapted to the board architecture. The final choice will take into account performances and ease of connection to the other DAQ components. As shown in table 6.1, the VxWorks operating system for AXP has proved to be more efficient in terms of context switching, preemption, use of semaphores and message passing.

Table 6.1. Rhealstone benchmarks for Alpha AXP model 3000/500.

	VxWorks for AXP	OSF/1 RT
Time, average	μs	μs
Context switch	6.5	18.1
Preemption	6.8	33.1
Message passing	17.2	92.8
Semaphore shuffle	36.7	152.2

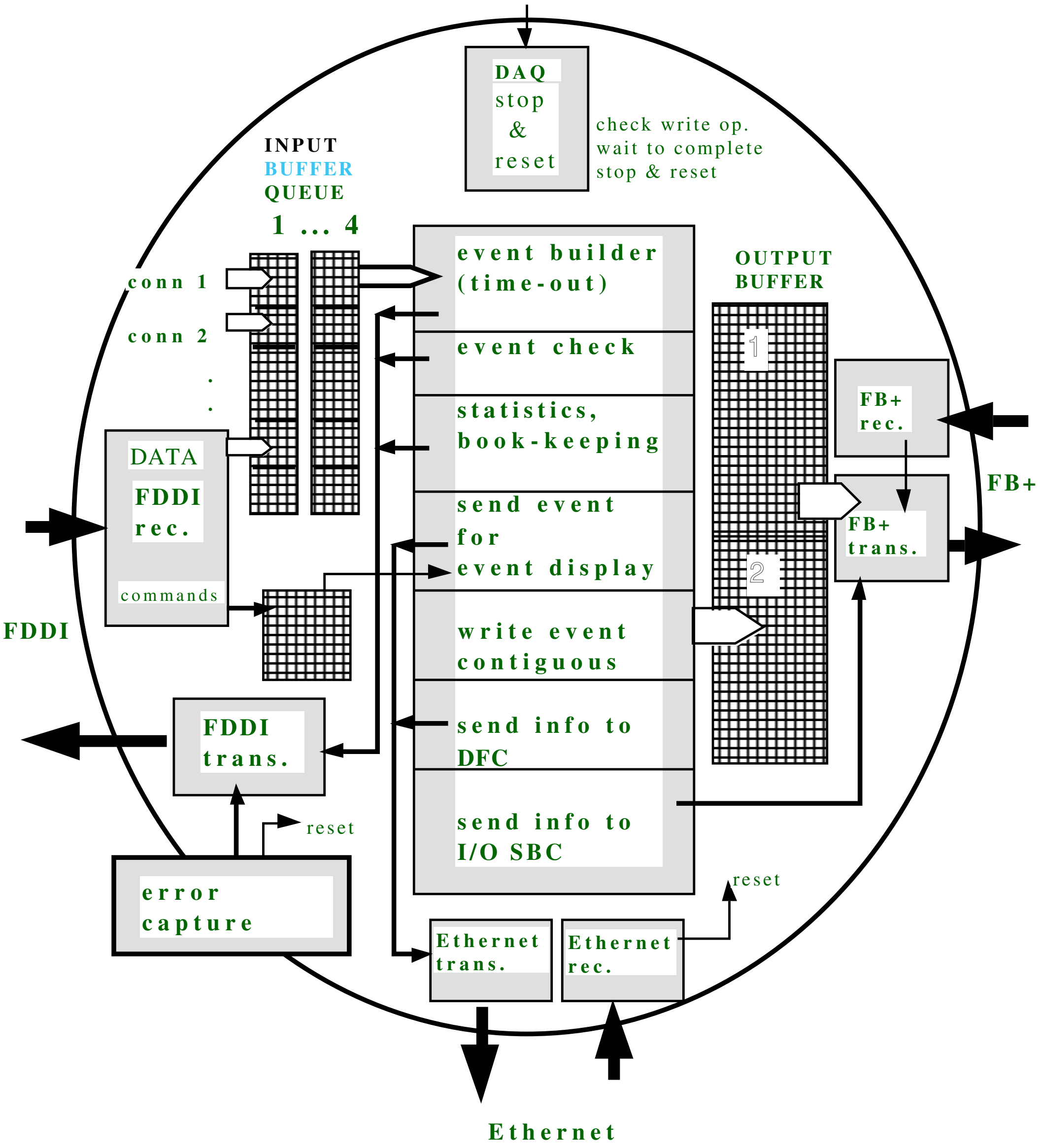

Fig. 6.3. Architecture of the on-line farm application software.

The thread concepts of the POSIX standard for real time programming will be used in writing the application program that will run on each SBC+, offering a safe and efficient environment and the possibility of developing hardware independent programs. In our application program, fig. 6.3, there will be a thread for each data or message received/sent via the input/output peripherals (FDDI, Ethernet, or Futurebus+), and a pipeline of threads for event building, checking, and reconstruction. There are 4 groups of 10 buffers, one buffer for each connection open ove FDDI, where *sub-event strings* are queued. Once the transmission to a group of 10 buffers is completed, the thread frees a semaphore allowing the event builder to access the *sub-event strings.* A timeout clock is armed to detect eventual timeout conditions in receiving. The event builder, once acquired the control of the semaphore, builds the events and copies them contiguously in the two output buffers. When one of them is full, the output thread is activated, and events are sent to the I/O SBC through Futurebus+. In parallel, statistic, event checking, and partial event reconstruction are performed, and the information sent to run controller and monitor with event samples.

7. RUN CONTROLLER, MONITORING AND SLOW CONTROL

7.1 Run Controller

The Run Controller will be implemented by a finite state machine together with the front-end processors (L2-VME) and the SBC used for the event building. The relevant state transitions will be:

1. Configuration: will initialize the processors and make sure they are ready for the start of a new run;
2. Go/Active: will be the normal transition which will set all the processors waiting for events;
3. Flush: will be the consequence of a DAQ unrecoverable error and will force all the processors to free all the buffers discarding all the data in the processing queue.

A Motif Interface will present simple menus to the operator console, together with message windows and graphics concerning the status and performances of the system.

The errors will be collected and logged in a central storage system. DAQ errors have to generate special actions regarding the Run and Triggering control of the detector, and must be traceable. The error logger is one of the software components which will make use of a general message system facility which allows communication between processes. This item will require major effort in the very early stage of the architecture definition and implementation. Since the message system has to work on many different processors, the choice is oriented toward a standard underlying transport protocol, TCP/IP, integrated by a well defined finite state machine software. In a 10 kHz rate DAQ, the timing of the messages is quite relevant: in 1 ms there will be 10 events accumulating in the buffer queues. Delivery time is critical and message losses and consequent retransmissions have to be considered as failures.

7.2 Monitoring

The natural monitoring place could be the SBC's CPU, and certainly an important part of the event checking will be done in the farm, but it is also clear that the time-critical application inside the farm, should not be compromised by this activity. The available CPU power will not be sufficient to do all the monitoring and it will be also necessary to monitor events at a later stage, when they are sent to a dedicated facility like a multi-processors computer, which can dedicate his CPU power to this important mission. Here all the histogramming and the summary of the monitored quantities can be done without interfering with the data flow. A common X-window/Motif graphics and presentation platform will be sufficient to allow a good uniformity in the user interface and a fairly good level of portability of all the common software. The software needed is essentially composed by a histogramming package like HBOOK, a presentation manager like PAW and a buffer manager library which will allow to access the events in a simplified way avoiding access conflicts and overwriting. A simplified block diagram is shown in fig. 7.1.

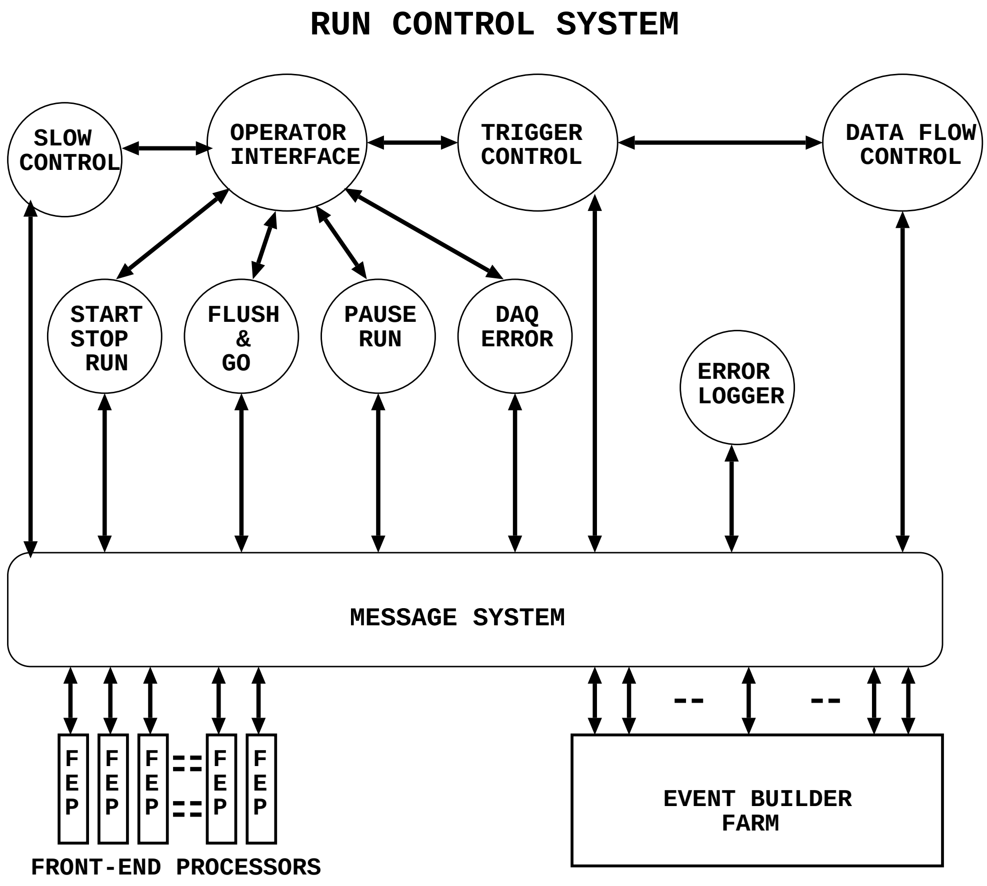

Fig. 7.1. Run Control Block Diagram.

7.3 Slow Control

The Slow Control is a special hardware/software system which will monitor and/or control the following entities:

1. Low voltages from VME crates and other power supplies;
2. High voltages used for the electromagnetic calorimeter photomultipliers and the chamber field wires;
3. Thresholds settings systems for trigger and frontend electronics;
4. Detector status variables such as atmospheric pressure and temperature.

The system will also be connected to the gas control system which will be a dedicated system necessary to control and monitor the gas flow and composition. Also the Magnet will be monitored, although the controls of this item are the DAΦNE operator's responsibility. The system will be implemented in VME for monitoring the Electronics (ADC's) and directly control some of the low voltages (like the VME power supplies) and serial communications with the various parts of the apparatus that have an internal monitoring/control system (a CPU), such as the gas system or HV power supplies.

The VME crate will then be read-out via a dedicated Ethernet connection (using optical fibre) with the slow control workstation, which will circulate the informations and, eventually, alarms. Asynchronous events, for example alarms, will also exist and they will generate messages on the slow control station monitor.

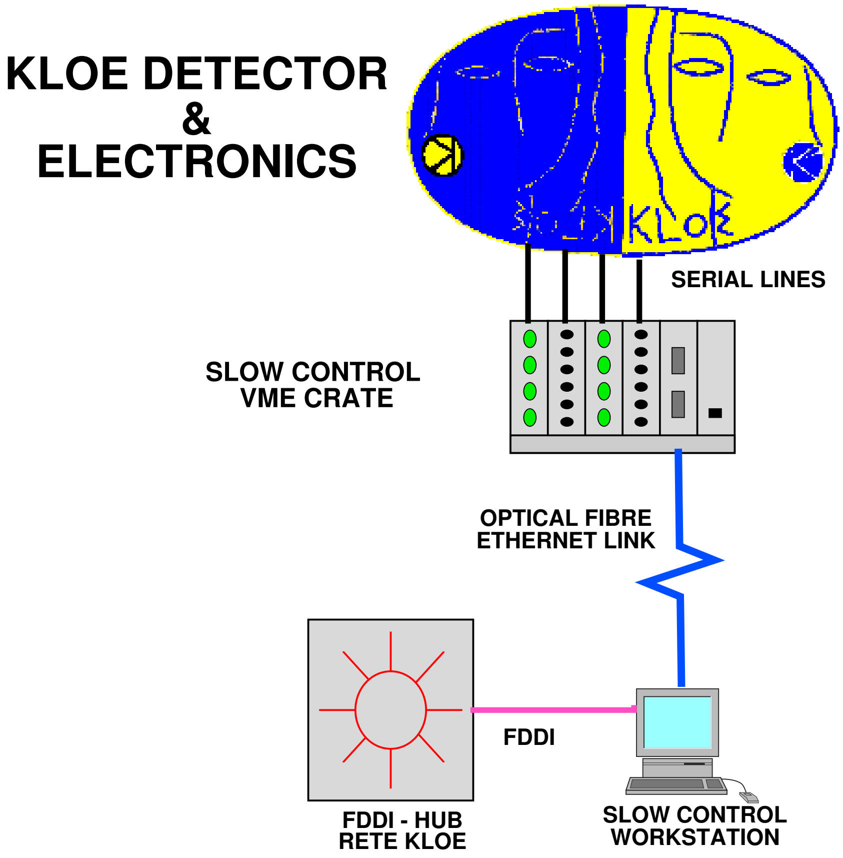

Fig. 7.2. Slow Control System.

8. MASS STORAGE

A scalable architecture is the choice for storage. A possibility is to attach a single tape unit to each farm crate containing the SBCs. The special I/O SBC collects events built by the SBCs on the crate and sends them to tape units, possibly reordering the events. According to the KLOE requirements, each tape unit has to sustain a minimum throughput of 5 Mbytes/s. In order to facilitate the reading of the tapes off-line, an automated library (where robotics and software management are implemented) where tapes can be put is important, as well as a file system able to manage the huge number of tapes that KLOE requires. Two systems are under study: the Quantum Digital Linear Tapes (DLT) and the Ampex D2 tapes (DST).

At present, the DLT tapes have a capacity of 10 Gbytes and the throughput of the TZ87 drive is rated at 1.25 Mbytes/s. The drive interface is SCSI-II. It uses a hardware compression system that could double both tape capacity and throughput. The bit error rate is lower than $1/10^{22}$. A single tape costs 70,000 lire and a TZ87 drive 10-12 Mlire (29 Mlire if completed with a 7 stack loader). An automated library exists. The first unit contains 264 tapes, 3 drives, a pass-through mechanism, a multi unit control and an input/output device for a total cost of 200 Mlire. Additional 4 units can be added for a cost of 180 Mlire each.

At the end of 1995, the tape capacity should be improved to 30 Gbytes (60 Gbytes with compression) and the throughput of the drive will achieve 3 Mbytes/s (6 Mbytes/s with compression). The library can easily be updated substituting the drives. The cost should be about the same. We assume in the following that our data could be partially compressed and that these new tapes have a capacity of 40 Gbytes.

The Ampex DST is a D2 helican scan device. Today the maximum tape capacity is 165 Gbytes. Ampex rates the maximum throughput of a FWD-SCSI interface at 15 Mbytes/s (18 Mbytes/burst). Measurements at DESY[21] reached a maximum of 7-8 Mbytes/s transfer rates from host to tape. The bit error rate is lower than $1/10^{12}$. The cost of a single tape is \$650 and of a drive ~\$150,000, which becomes \$200,000 adding a 10 tapes robot. Trends indicate doubling of the capacity, up to 340 Gbytes, each year, without price change. These tapes could be accessed via FWD-SCSI, which allows a maximum theoretical throughput of 20 Mbytes/s, or via an ECL parallel interface at 30 Mbytes/s. For these drives also there is an automated library.

At DAΦNE's full luminosity, KLOE needs 10 tapes units (plus an eventual spare device for backup) on which events are written in parallel while running. It is possible to write data produced in a day of running on 9.5 DST tapes or 81 DLT tapes. The tape price per day is thus 10.5 Mlire for DSTs and 5.7 Mlire for DLTs. At full luminosity, 10 drives completed by loaders are required for a cost of 3.2 Glire for DSTs or 290 Mlire for DLTs.

Additional tapes are also necessary for the data summary tapes and Monte Carlo data. A realistic estimate is about 900 Tbytes on tape after the first two years of running, equivalent to 22,500 DLTs or 2,650 DSTs.

It is evident at present that the Ampex solution is much more expensive than the DLT one, but in any case, the throughput of the new DLTs has to be checked in order to be sure that it will satisfy the KLOE requirements. Other architectures, with farm crates connected, eventually, via a SCSI port, to a server to manage the access to two Ampex drives at 30 Mbytes/s, will be studied in order to compare prices and performances.

As for event library management, the KLOE collaboration feels that raw data can be written on tape without creating at the same time an event catalog. On the other hand, it is impossible to rewrite the raw data on new tape when reprocessing the event, otherwise the number of tapes will increase further. That means that when it is necessary to access a raw event, it will be necessary to look for it reading directly the tapes of the run pertaining to that event, which does not create an unbearable burden.

9. OFF–LINE ANALYSIS DEVELOPMENT

9.1 Introduction

It is very important for the experiment that a common environment is established across all areas of software development. In fact, activities that traditionally are considered purely off-line on-line software. It is for this reason that the organization of the software environment and the utilization of common development and distribution tools at all stages of the analysis is crucial for the experiment.

The amount of data that the KLOE experiment will produce forces the choice of a "not heavy" memory manager to structure all physics information coming from the different subdetectors as well as the use of a common framework for all stages of the analysis. The YBOS[22] package is an extension of the BOS memory manager developed at the DESY and improved, with many features added, by the members of the CDF Computing Group at FNAL. It has a very simple structure and allows for the organization of events into subsections (banks) which can be easily accessed.

KLOE programs will run under the ANALYSIS_CONTROL[23] package based on YBOS: it is a general control structure which has performed well for online calibration and monitoring, as well as for standard DST analysis during the 1994 test run of module zero of the KLOE electromagnetic calorimeter. Both YBOS and ANALYSIS_CONTROL are well supported at FNAL. Further developments are in progress closely tracking developments in computing, providing object oriented interfaces and "sophisticated" graphics tools for debugging purposes.

The HEPDB[24] database package developed at CERN seems to comply with all requirements desired by KLOE. It is easy to use and has already been implemented in the ANALYSIS_CONTROL environment by KLOE. It is absolutely necessary to structure correctly the architecture of the database tree in order to avoid explosions in size.

KLOE has also established guidelines for writing or modifying software,[25] and has a standard metric reference .[26]

9.2 The development environment

Some time has been spent in designing a common development environment on all platforms supported by the experiment. Because a large amount of physics software has been developed under the OpenVMS operating system in the early days and because of the large presence of this OS throughout the collaboration, it has been decided to support this platform until the entire software development is moved under UNIX. In fact, a large number of vendors competitively offer high performance microprocessor based workstations running similar and largely interoperable UNIX environments at reasonably low cost. UNIX is becoming a standard in the HEP environment and a lot of tools are available free and maintained. Most of the today's available technology for computing and storage is available only under UNIX. Finally, the KLOE online farm most likely will get the most advantages by running UNIX-flavored real time systems. These are the strong arguments for KLOE to address its choice toward this platform. HP/HP-UX and DEC/OSF1 are the two UNIX platforms currently supported by KLOE.

9.2.1 *The co-existence of UNIX and OpenVMS*

The co-existence of OpenVMS and UNIX based machines creates non trivial problems of interoperability. Few tools have been developed at KLOE to ensure data sharing (performed via NFS at the moment), restricted access to data and software, and same view under each platform. The use of AFS (Andrew File System) or DFS (Distributed File System) is forseen in the future. Via the use of UPS (Unix Product Support), developed at Fermilab, and KVS (KLOE VMS Setup), a home-made product, the same command interface is given to users on all platforms to operate inside the KLOE environment. Defined rules have been given for data and software organization and development. In particular, standard guidelines have been enforced for writing code.

9.2.2 *Tools for code management*

The importance of the existence of tools for automatic version management, rebuilding, and validation of software libraries is recognized. At KLOE we developed a few tools[27] that make these tasks easy to developers, hiding the complexity of the system and allowing novices and experts to interact with it in a manner that maximizes productivity. The use of a common facility (REBUILD)[28] avoids duplication of efforts in mantaining a library. The addition of new code or the modification of an old routine is done in a standard automatic way, the same for all libraries or packages used by the experiment. This allows for interchangeability of tasks between maintainers, who need only worry about the core of the software. A still open point is the use of a code version manager at KLOE. A few existing tools are under evaluation: among the others, CVS (Concurrent Version System)[29] , RCS (Revision Control System)[30] , CMZ (Code Management with ZEBRA)[31] .

9.2.3 *Distribution tools*

Distribution tools are also very important to integrate on-site computing development with that of off-site collaborators. Some work is being done in this direction. Designing and developing distribution tools will be greatly simplified with the introduction of AFS or DFS. The idea is to distribute the source code allowing the remote server to run REBUILD and produce object libraries and executables. The use of WWW (World Wide Webber) and UNIX tools such as WAIS (Wide Area Information Service) and PERL (Practical Extraction and Report Language) are essential for our documentation management and distribution.

9.2.4 *Programming high-level languages*

Given the wide-spread expertise that exists within the HEP user community and the large amount of code already existing, the high level languages chosen by the collaboration for writing off-line code are FORTRAN 77 and C. No unique commitment has been made, however. New tools based on object-oriented techniques offer high flexibility in today's software. New software development can take advantage of new techniques while ensuring integrability with the existing FORTRAN-based code.

9.3 The data format and the memory management system

Because of the main high-level programming language chosen (FORTRAN), the complexity of the rewriting task, and the lack of man power, we decided to adopt an already existing memory manager for the KLOE experiment, and to modify it to comply with our specific requirements. After analyzing the two most commonly used memory managers, YBOS and ZEBRA ,[32] we decided to adopt YBOS as detailed below. We next describe the format of our data and the organization of the event information.

9.3.1 The YBOS memory manager

We evaluated the two most commonly used packages: YBOS and ZEBRA. They both offers the same features: YBOS supports sequential and direct access files and offers a method for transporting data banks on machines of different architectures and internal datatype representation. A mechanism of garbage collection grants the recovery of memory space by overwriting deleted banks; ZEBRA is a more sophisticated memory manager which offers all features included in the YBOS package, plus a very handy tree structure for the event banks. Using ZEBRA "reference" and "structural" links it is very easy to correlate, for instance, the track vertex of a given event and study its characteristic. Because of the number of information stored into a ZEBRA bank, the CERN package is more complete but slightly heavier in its structure than YBOS. The evaluation of the two packages consisted of measuring CPU and ELAPSED times used to store a typical KLOE event in YBOS or ZEBRA banks and to write it on disk. The results of the tests showed that YBOS is faster than ZEBRA and the difference is significant when we deal with a huge number of events. Also the size of the ZEBRA output file is bigger than the YBOS one. The details of this study can be found in the KLOE note no. 65. The computing time spent in the storage and retrieval of the data and the size of the output file are very crucial in the KLOE experiment.

9.3.2 The event data structure

The event information is stored into YBOS banks. A hierarchical scheme has been established to represent different stages of the reconstruction process.[20] The idea is to have a coherent way of storing events which is the same at each stage of the software, from Monte-Carlo and simulation analysis to "real" data. At each level of the analysis, we find a logical progression of YBOS banks from the most elementary level of the data structure to the sequentially more highly refined levels. The first and most elementary level in the topological hierarchy is the Raw Data Bank, a YBOS Bank which contains detector information as it comes out from the data acquisition system. Immediately above this is the element level. An Element Bank is a construct of the raw data which retains most of the granularity of the detector information which it represents, but uses the calibration database to correct the digitized information into real numbers corresponding to physics quantities. Elements are organized by detector reflecting that calibration is invariably done on a detector by detector basis. Above this level is the segment level. Segments are the combination of the information from one or more detector. As an example, segments may be helices in a tracking chamber or energy clusters in a calorimeter. Objects are the highest level in this hierarchy. Objects are the items which are used by physicists for analysis and should have transparent interpretations in terms of the physics of the event. However, they are conceptually different from segments in that they are associated with a physics interpretation of the event and have physics dependent assumptions regarding their kinematic attributes. At each stage of this chain a mechanism to go back to the parent bank is available.

9.4 The offline analysis tools

Here we give an overview of the Analysis Software Tools used at KLOE. The ANALYSIS_CONTROL driver allows users to put together in few minutes a fairly complicated analysis program and to drive the executable at run-time to do different kind of analysis at once. The HEPDB database developed at CERN is used at KLOE to store the experiment constants.[33] It has been used successfully up to now.[34] It also offers a distribution mechanism to client machines. The GEANFI[17−20] program is the KLOE interface to the Monte-Carlo and simulation program GEANT developed at CERN.

9.4.1 An analysis driver

An advantage in using the YBOS package is that at KLOE we can use the ANALYSIS_CONTROL package developed at FNAL. This is a general control driver which allows one to painlessly combine several independent subprograms into a single executable image. Algorithms developed by users can easily be incorporated in the production analysis package, or be borrowed by other users without changes. ANALYSIS_CONTROL offers the ability to control several modules at run-time, to manipulate the histograms associated with a given analysis module, to enter a "parameter override" menu for any modules that supply such menu (changing cuts), to stop event analysis in the middle of an analysis path on the basis of information provided by an event selection or filter module, the possibility to specify multiple output streams and the source of the input data (disk or tape, the online event, the network - DECNET or TCP/IP -, a Monte Carlo generator), and provides a mechanism for handling modules that can be run with more than one set of default parameters.

Since all existent Monte Carlo studies at KLOE use the GEANT package, an interface to ANALYSIS_CONTROL has been developed to grant a complete integration of such production with the rest of the analysis.

9.4.2 The KLOE database

The HEPDB database system has been developed at CERN by L3, OPAL and the Application Software Group of the Computer and Network Division. It complies the requirements to store detector constants, such as geometry and calibration data, to have a fast access to the information at program time execution and to limit the rate of transaction between the storage media where the database resides and the computer memory available to the user. HEPDB is based upon the ZEBRA RZ package and increases the functionalities of RZ providing data compression and optimization of performances through partitioned directories. An interface to ANALYSIS_CONTROL has been developed so that the ZEBRA structure is completely hidden to users. HEPDB uses random access files which can reside on direct access devices or in memory. The data is accessed or stored using a UNIX-like pathname, plus one or more identifiers known as keys. The fundamental concept for the operation of HEPDB is the database server, i.e. a job running in detached mode that regularly distributes updates to the machines requesting them. So, only one machine has a master copy of the database, while the slaves can have only the part of it that is requested by their own users, receiving the updates relative to that part from the master. HEPDB provides an interactive interface to control the integrity of the data, a FORTRAN library and portability among different platforms. It is absolutely necessary to the HEPDB behaviour in presence of a large quantity of data and access requests.

9.4.3 The Monte-Carlo and simulation program

For generation and simulation of KLOE events and analysis, the GEANT program from CERN has been adopted. An interface to it has been created (the GEANFI program) to introduce all KLOE parameters and geometry. The event generator and the particle tracker have been described.[19,35] The simulation of events different from Φ decays, or from single particles to study detector performances, at present is done by dedicated external generators, as it is in the case of Bhabha's and in the case of machine background. The event kinematics simulated by the dedicated codes is then read out by GEANFI, to track each particle history inside the KLOE apparatus. GEANFI includes the description of all relevant phenomena concurring to the detector responses. It allows to easily change any detector descriptor concerning both, the KLOE detector design, including beam pipe and quadrupoles in the KLOE region, and the parameters driving drift chamber and calorimeter responses. Moreover, different output levels can be chosen, to get a right-sized information for all different needs envisaged in the experiment development stage. GEANFI has been written using the tools of the CMZ code manager package, that allows to easily handle any updating in the GEANT library, to keep tracks of any change in the code itself, and to export the source to any platform provided for the KLOE software. GEANT is interfaced with a graphic-interactive package working within both, GKS and MOTIV environments. Such an utility has been extensively used to debug the Monte Carlo and to visually scan peculiar event topologies. For this purposes, GEANFI can run in reading-mode, to retrieve one or a group of previously generated events. GEANFI is capable of writing YBOS output banks and producing simulated raw data events, structured in the same way as they will be produced by the real DAQ processes. GEANFI has been interfaced also to the Analysis Driver and to the HEPDB database. This whole package provides users with a most powerful simulation tool.

10. COSTS AND SCHEDULES

The milestones for the first DAQ prototype, that is expected to operate in a reduced hardware configuration by February '96 is given below. The extension to the complete system and the final test are to be completed by the end of '96.

10.1 Time schedule

The DAQ milestones are indicated in tab. 10.1.

Table 10.1. DAQ milestones

Schedule	Hardware Project	Software Project
Done		FDDI and message passing protocol tests, Data flow control and event-building studies, CPUs and Storage devices performance evaluation
Done		Software prototype for PSI test
	DAQ Technical Addendum and DAQ Freezing	
Done	ROCK prototype	Start VME software for L1 monitoring
Mar. 95		First Data flow control protocol implementation
Jun. 95	ROCKM prototype	Start VME sub-event building software Run Control, Event building and Monitor
June 95		First prototype of the INFN-DEC Joint Project farm First decisions about the on-line farm
Sep. 95	Tests on new VME CPUs and PCI-FDDI interfaces	
Oct. 95		Decisions on data storage hardware
Feb. 96		First DAQ prototype running
Year 96		Hardware completion, final software tests Integration on-line and off-line systems

10.2 Costs

The cost of the DAQ system presented in tab. 10.3 covers the data path from the CPUs in the L2 crates up to the storage systems implemented using commercial devices. The cost of the readout controllers, and ROCKs have been given in chapter 4 and is considered to be part of the frontend electronic that is evaluated in a separated document. The ROCKM is included in this table as a part of the L2 crate configuration. Because a strong correlation between DAQ and OFF-LINE analysis choices exist, the cost of the off-line farm is also estimated in a separated table, together with the requirements in network connections, robotics and disks for staging. Due to the big improvement in computer technology announced for the first half of '95, we may still change some choices before the final complete realization in '96. Moreover we evaluate the DAQ and OFF-LINE costs using today quasi-available devices and prices, convinced that the cost can only decrease in the future.

10.2.1 ROCK and ROCKM cost estimate

A cost estimate in single units is given in Table 10.2.

Table 10.2. Estimated costs

hardware	cost (MLire)
1 ROCK	4
1 ROCKM	6
Cabling (for all)	40

10.2.2 DAQ Cost Estimate, Level 2 Crates to Storage Devices

Table 10.3. DAQ cost

Item	Units	Cost/unit	Total, MLit
L2 crate: trigger(1), DFC(1), calor.(2), chamber(6), vtxch(2)			
6U VME crate	12	13	156
CPU on VME	12	18	216
FDDI interface	12	17	204
VIC	24	6	144
ROCKM (6 spare)	18	6	108
DFC and RUN control WS with VME connection	1	50	50
GIGAswitch, bridge and switch FDDI/ATM with 2 FDDI ports	1	52	52
FDDI/ATM boards for GIGAswitch with 4 ports/board (2 spares)	8	35	280
HUB900: LAN concentrator: Ethernet, FDDI, ATM (1 spare)	3	7	21
Ethernet modules for HUB900: 32 ports/module (1 spare)	5	10	50
FDDI modules UTP for HUB900: 6 ports/module	24	9	216
Ethernet bridge for HUB900: to isolate Ethernet LANs	3	6	18
FARM: crate with 11 boards per crate: 10 for event processing(1400 specint92) 1 for I/O (1 crate spare) (FDDI/ATM interf. on-board)	11	190	2090
Storage device with loader (2 spares)	12	40	480
Workstations for monitor and downloading	4	30	120
Total			4205

The cost of the system is given above in table 10.3 . Some components are already available. In particular the GIGAswitch with 6 FDDI ports was bought by the Frascati National Laboratory to allow us the initial tests on FDDI.

10.2.3 Off-line computing cost

The off-line computing cost, presented in tab. 10.4 covers a second farm offering an equivalent CPU power; a disk capacity where one half of run can be stored and 6 tape drivers managed by a single Automatic Tape Library (Robot), together with a powerful server for exporting disks to the complete farm system. A set of workstations and/or Xterm for monitoring is also included.

Table 10.4. OFF-LINE cost

Item	Units	Cost/unit	Total MLit
HUB900: LAN concentrator: Ethernet, FDDI, ATM	2	7	14
Ethernet modules for HUB900: 32 ports/module (1 spare)	5	10	35
FDDI modules UTP for HUB900: 6 ports/module	22	9	198
Ethernet bridge for HUB900: to isolate Ethernet LANs	3	6	18
FARM: crate with 14 boards per crate (2000 specint92) (FDDI/ATM interf. on-board)	12	270	3240
Disks for staging (400 GB) Input/Output and MonteCarlo	2	400	800
Storage device: 1 Automatic Tape library (Robot) with 6 drivers (500 tapes on-line)	1	500	500
Server to manipulate robotics and disks	1	300	300
WS and/or Xterm for monitor and downloading	4	30	120
Total			5225

REFERENCES

1. *A GENERAL PURPOSE DETECTOR for DAΦNE*, The KLOE Collaboration, LNF-92/019 (1992).
2. *THE KLOE DETECTOR, Technical Proposal*, The KLOE Collaboration, LNF-93/002 (1993).
3. *THE KLOE CENTRAL DRIFT CHAMBER, Addendum to the Technical Proposal*, The KLOE Collaboration, LNF-94/028 (1994).
4. M. Carletti, G. Felici, P. Locchi, "Chamber front-end status report", KLOE Note 66.
5. R. J. Yarema *et al.*, "A High Performance Multi-Channel Preamplifier ASIC", IEEE Transaction on Nucl. Sci., 39 (1992).
6. Cadence Design System Inc., VERILOG-XL Reference Manual, Version 1.6, 1991.
7. A. Aloisio *et al.*, "Fast readout system for KLOE", KLOE Note 88.
8. A.Aloisio *et al.*, "Custom solution for a data readout architecture: a system level simulation", Proceedings of CHEP94.
9. XILINX, The Programmable Gate Array Data Book, 1993.
10. Advanced Micro Devices, MACH 1 and 2 Family Data Book, 1993.
11. C. Dalton *et al.*, "Afterburner, A network-independent card provides architectural support for high-performance protocols", Hewlett Packard Laboratories, IEEE Network 36-43, July 1993.
12. E. Pasqualucci, "Progetto di misura della violazione di CP a DAΦNE: separazione del canale $K_L \rightarrow \pi^+\pi^-$ dai canali semileptonici nell'esperimento KLOE", Ph.D. Thesis, Università di Roma 'Tor Vergata' (1994).
13. M.L. Ferrer *et al.*, "Builder protocol simulation, phase 1", KLOE Note 37.
14. "Performances of the UDP and TCP protocols using the FDDI and ETHERNET channels on several RISC machines", M. Carboni *et al.*, KLOE Note 61.
15. "High Performance TCP/IP for OSF/1 Alpha AXP Workstations (White Paper)", Digital Equipment Corporation Networks Engineering TCP/IP Program Office, Littleton, MA, March 1993.
16. W. Grandegger *et al.*, "TCP performance test with I/O-multiplexing for the KLOE event builder system", KLOE Note 120.
17. A.Antonelli, C.Bloise, A.Calcaterra, "An introduction to GEANFI", KLOE Note 3.
18. A.Antonelli, C.Bloise, A.Calcaterra, "GEANFI program - version 1.2", KLOE Note 23.
19. A.Antonelli, C.Bloise, N.Cavallo, "GEANFI handbook vers. 1.3.3", KLOE Note 52.
20. A.Andryakov, *et al.*, "KLOE data structure", KLOE Note 137.
21. private conversation with M.Ernst-Eschgarth.
22. D.Quarrie, B.Troemel, "YBOS programmers Reference Manual", CDF Note 156, version 4.00, 18 Jan 1994.
23. M.Shapiro, E.Sexton-Kennedy, D.Quarrie, "A beginner's guide to A_C and B_J", CDF Note 384, version 7.21, April 1994.
24. "HepDB, database management package, reference manual", Application Software Group, Computers and Network Division, CERN program library long writeup Q180.
25. F.Donno, F.Pelucchi, "Guidelines for writing or modifying KLOE software", version 2, KLOE Note 73.
26. A.Andryakov, *et al.*, "KLOE metric (reference systems, numbering/naming conventions, units)", KLOE Note 115.
27. F.Donno, "Software structure, management and distribution at KLOE", draft 2-95 version 2.0 KLOE Memo 4.
28. F.Donno, "REBUILD for libraries maintenance. User's guide and reference manual", CDF Note 1611, version 3.60.
29. P.Cederquist, "CVS: version management with CVS,release 0.8", Oct 1993.
30. W.F.Tichy, "Design, implementation and evaluation of a Revision Control System", Proceedings of the VI International Conference on Software Engineering, IEE, Tokio, Sep 1982.

31. "CMZ: a source code management system: CMZ User's guide and reference manual", CERN.
32. "ZEBRA, overview of the Zebra system", CN, ECP and PPE Division, CERN program library long writeup Q100/Q101.
33. F.Pelucchi, "Database user and reference manual", KLOE Note 99.
34. F.Pelucchi, "KLOE database server for test beam", KLOE Note 123.
35. KLOE Detector, "The Technical Proposal" LNF-93/002 p.83 (1993).