Rapport de stage
-
Configuration Dynamique de l'Arbre VME du Calorimètre de l'Expérience H1
-
Dynamic Set-up of the Calorimeter VME Tree of the H1 experiment

Jean-Baptiste Chabane

Mars - Juin 2000

REMERCIEMENTS

Table des matières

1 Présentation de l'Institut d'Accueil -
Presentation of the Host Institute **5**
1.1 Le Centre de Recherche DESY . . . 5
1.2 L'Accélérateur HERA et l'Expérience H1 . . . 5
1.2.1 HERA . . . 5
1.2.2 L'Expérience H1 . . . 6

2 Purposes of the Training Period -
Les Objectifs du Stage **7**
2.1 Study the DAQ System -
Étude du Système DAQ . . . 7
2.1.1 Description of the Current System . . . 7
2.1.2 Reasons for Changing the System . . . 9
2.1.3 The New Architecture . . . 9
2.1.4 Becoming an Expert . . . 11
2.2 Embarking the System Migration -
Entamer la Migration du Système . . . 11
2.2.1 VME/VIC Compiler Program . . . 11
2.2.2 Use the PPC as Server . . . 12

3 Accomplished Jobs -
Le Travail Réalisé **13**
3.1 Discovery and Study of the DAQ System -
Découverte et Etude du Système DAQ . . . 13
3.1.1 The VME Bus . . . 13
3.1.2 The VIC 8250 . . . 14
3.2 Assembling a Test Bench -
Montage d'un Banc d'Essai . . . 16
3.2.1 RIO2 Card Set Up . . . 16
3.2.2 The Arbiter Card . . . 17
3.2.3 Accessing to a Remote Crate . . . 17
3.3 Code Adaptation and Tests on the Test Bench -
Adaptation du Code et Tests sur le Banc d'Essais . . . 19
3.3.1 File "config.vme" and VME Tree Structure . . . 19
3.3.2 Initialisation and Set Up Functions . . . 21
3.3.3 Code Adaptation . . . 23
3.3.4 Results on the Test Bench . . . 23
3.4 Test on the Real System -
Test sur le Système Grandeur Nature . . . 24
3.4.1 The Vicscan Program . . . 24
3.4.2 Plugging the Card . . . 24
3.4.3 Running the Compiler Program . . . 24

3.4.4 The Following Steps . . . 25
3.5 Contributions and Results - Apports et Résultats . . . 27
3.5.1 A Faster Compiler . . . 27
3.5.2 A Client-Server Architecture for the Compiler . . . 28
3.5.3 Ready for Changing the DAQ Architecture . . . 28

INTRODUCTION

Ce stage de fin d'études fait partie intégrante de la formation ES2I (Études Supérieure d'Ingénierie Informatique) à l'ESIL (École Supérieure d'Ingénieurs de Luminy). Dans mon cas, ce stage de 4 mois est la première partie d'un projet de longue haleine qui doit s'étendre sur 2 ans, et auquel je continuerai à participer par l'intermédiaire de mon service militaire en tant que coopérant.

Situé à Hambourg dans le nord de l'Allemagne, DESY (Deutsches Elektronen-SYnchrotron) est l'un des fleurons de la recherche dans le domaine de la physique des particules. A l'origine, accueillant un seul synchrotron d'une envergure de quelques dizaines de mètres, DESY est depuis une dizaine d'années l'hôte de l'accélérateur de particules HERA d'un diamètre d'environ 2 kilomètres. HERA est un collisionneur de haute énergie électron (ou positron) proton et est utilisé par 4 expériences de physique des particules : ZEUS, HERMES, HERA-B et H1.

Le système d'acquisition de données du calorimètre de cette dernière fut le cadre de mon stage. Reposant sur l'étude des collisions entre protons et électrons, H1 a pour objectif de déterminer la structure interne du proton, d'étudier les intéractions entre particules et d'observer de nouveaux phénomènes.

L'étude des interactions entre les protons et électrons de l'expérience H1 repose entre autres sur un énorme calorimètre qui permet de "visualiser" les émissions de particules élémentaires lors des collisions entre les particules. Le calorimètre est composé de nombreux détecteurs et de tout un réseau temps réel d'acquisition des données de ces détecteurs.

Ce système d'acquisition de données, appelé DAQ pour "Data Acquisition", repose sur des batteries d'ADCs (Analog Digital Convertissor), de DSPs (Digital Signal Processor), un système de déclenchement permettant l'acquisition de données lors d'événements intéressants, le tout géré par des processeurs AMD 29K et un système d'exploitation temps réel VRTX.

Ces processeurs et ce système d'exploitation datent aujourd'hui d'une vingtaine d'années et offrent le strict minimum de services. L'objectif à atteindre au terme de ces 2 ans de projet est de faire migrer la DAQ vers un système Lynx-OS qui offre les avantages du temps réel et des systèmes UNIX, tout en s'appuyant sur des processeurs plus puissants à savoir des Power PCs. À terme, cette nouvelle structure doit permettre des gains au niveau électronique, calibration et initialisation du système DAQ plus rapides, mais aussi au niveau qualité de services : meilleure interface utilisateur, possibilité d'offrir des graphiques en ligne de meilleure qualité ainsi que des interrogations simultanées par plusieurs utilisateurs via Internet.

Le sujet de ces 4 mois de stage s'intègre entièrement dans le projet global. L'objectif à la fin de ces quatre mois est de démarrer la migration du système, à savoir : l'installation d'un Power PC muni du système d'exploitation OS9 sur la DAQ sans interférer avec le système existant, d'entreprendre les modifications nécessaires de code concernant l'initialisation de l'arbre VME de ce système d'acquisition et d'amorcer une structure client-serveur.

Pour des questions d'accessibilité à tous les membres de la collaboration H1, les parties techniques de ce rapport seront rédigées en anglais.

Chapitre 1

Présentation de l'Institut d'Accueil - Presentation of the Host Institute

1.1 Le Centre de Recherche DESY

DESY (Deutsches Elektronen Synchrotron) est un centre de recherche national à fonds publics. Son objectif est de conduire des recherches fondamentales sur les sciences physiques avec comme principales priorités:

1. La compréhension des propriétés fondamentales de la matière dont la physique des particules.
2. L'utilisation du rayonnement synchrotron dans la physique des surfaces, la science des matériaux, la chimie, la biologie des molécules, la géophysique et la médecine.
3. Le développement, la construction et la mise en route des équipements associés à l'accélérateur de particules.

Fondé le 18 décembre 1959, DESY compte aujourd'hui 1046 postes permanents et un budget de 270 millions de Marks financé à 90% par l'état allemand, le reste par la ville de Hambourg et l'état fédéral de Brandenbourg. 3400 scientifiques venus de 280 universités et instituts de 35 pays différents coopèrent tous ensemble à l'avancement des projets mis en place.

1.2 L'Accélérateur HERA et l'Expérience H1

1.2.1 HERA

HERA est l'acronyme de "Hadron-Elektron-Ring-Anlage". Unique au monde, il s'agit du seul collisionneur électron (ou positron) proton de haute énergie. Construit sur le site de DESY, ce collisionneur est utilisé par quatre expériences : ZEUS, HERA-B, HERMES et H1. Voici les deux principaux objectifs visés par ces différentes expériences :

1. La compréhension des forces fondamentales et le décodage de la structure interne des protons
2. Observer de nouvelles formes de la matière.

1.2.2 L'Expérience H1

H1 est le fruit de la collaboration de plus de 400 physiciens venant de 52 laboratoires de 12 pays différents. Cette expérience, pour étudier les collisions entre les protrons et les électrons de l'accélérateur HERA, a nécéssité la construction d'un grand détecteur installé autour du point de collision. Ce détecteur a une envergure de $12 \times 10 \times 15$ mètres et pèse 2800 tonnes. Il est principalement constitué d'un système de chambres à fils, d'un calorimètre à argon liquide, d'un aimant supraconducteur de 5,8 mètres, de chambres pour la détections des muons. Le tout comporte près de 78000 cellules de détection.

Le calorimètre. La collision entre les électrons et les protons produit des particules élémentaires (électrons, positrons, photons) et composées comme les hadrons (particules constituées de quarks). Le calorimètre H1 est constitué d'un empilement de plaques d'acier (calorimètre hadronique) et de plomb (calorimètre électromagnétique) plongés dans 70 m^3 d'argon liquide. Le calorimètre électromagnétique permet surtout la détection des photons et des électrons (ou positrons) alors que le calorimètre hadronique, comme son nom l'indique, est tout particulièrement destiné à la détection des hadrons.

Le principe de détection électronique. Un calorimètre sert à mesurer l'énergie des particules. Le principe de détection électronique du passage des particules repose sur l'ionisation de la matière. Le passage d'un électron dans la matière ionise les atomes du milieu. En appliquant un champ électrique suffisamment fort, les électrons arrachés des atomes sont accélérés et, en heurtant les autres atomes constituant la matière, arrachent d'autres électrons. De ce phénomène découle que la charge des particules ionisées est amplifiée. Les cellules de détection ne sont finalement que des capacités sous tension qui se déchargent partiellement lorsque les particules ionisées les traversent. Dans notre cas, la matière qui est ionisée par les particules, est l'argon liquide.

Les événements. En pratique, le calorimètre H1 n'enregistre pas des données en permanence mais attend les collisions entre électrons et protons qu'on appelle "événements". Malgré un croisement des faisceaux d'une fréquence de 10,4 MHz, l'expérience ne retient en moyenne qu'une cinquantaine d'événements chaque seconde. Pour enregistrer les collisions intéressantes, le calorimètre a besoin d'un système de déclenchement et d'acquisition de données. C'est pourquoi deux types de signaux sont extraits du calorimètre à argon liquide.

Les 78000 canaux ne délivrent chacun qu'un signal pour chaque croisement de faisceaux mais ces signaux suivent deux traitements en parallèle. Le premier traitement est très rapide et se fait à partir de cartes Trigger, de FADCs (Flash Analog Digital Converter) qui sont des ADC rapides et des DSPs. Le résultat obtenu après bon nombre de sommations, d'intégrations et de suppressions de zéros (afin de comprimer les données) donne lieu aux signaux de déclenchement. Ces mesures de l'énergie sont assez grossières (8 bits de précision) mais vont permettre de determiner s'il y a un événement ou pas. De plus, ces signaux donnent le comportement temporel du calorimètre, ce qui est très important dans l'analyse ultérieure des signaux. Le deuxième traitement en parallèle est plus long mais offre une mesure plus fine des énergies (12 bits de précision). Ce traitement passe, pour sa part, par des ADCs et des DSPs. Les opérations suivies sont similaires au premier traitement mais ici, l'intégration est plus grande de sorte qu'on obtienne une meilleure précision de la mesure. Le résultat de ce deuxième traitement sont les signaux calorimètre.

La décision d'enregistrement est prise par le Central Trigger qui reçoit les signaux de déclenchement et qui est configuré en fonction des études du moment. En fait, le Central Trigger détermine des coïncidences qui conditionnent directement le déclenchement de la prise de données. Si l'enregistrement est décidé, les signaux déclenchement et les signaux calorimètre sont transmis à la DAQ centrale de l'expérience qui stockent tous les résultats des évènements. L'envoi des données se fait via les chassis Events Builders (voir section 2.1.1).

Chapitre 2

Purposes of the Training Period - Les Objectifs du Stage

2.1 Study the DAQ System - Étude du Système DAQ

2.1.1 Description of the Current System

The Liquid Argon Calorimeter, the Tail Catcher, the Plug and the Spacal are all different types of calorimeters composing the subdetector group called Calorimeter. The Trigger Analog crates, the Sub-Trigger and the Central Trigger participate to the trigger data of event.

The figure 2.1 presents an overview of the full DAQ System. It is composed by the different branches constituting the DAQ and the main node where all the branches are coming from.

- The VIC crate: It contains one Motorola 68K card linked via ethernet to one VAX, and one QBUS card linked, once again, to the VAX, and all VIC cards allowing access from the local crate to 5 branches described below.
- The Calorimeter branch (1): it contains ADCs, DSPs (Calorimeter, Hadronic Detector, Tail Catcher, Plug), Sub-Trigger Controller crate and the Event Builder Calorimeter crate including one 29K card and the central DAQ access.
- The Trigger branch (2): it contains FADCs, the VME crates including the 11 Trigger DSPs and the Event Builder Trigger crate including one 29K card and the central DAQ access.
- The Calibration branch (3): it is the calibration crate and the 10 Trigger Analog crates.
- The Trigger Analog branch (4): further 15 Trigger Analog crates.
- The Spacal branch (5): it contains ADCs and Spacal Calorimeter DSPs, Sub-Trigger Controller crate, and Event Builder Calorimetre crate.

The branch numbers are defined in the config.vme file described in section 3.3.1.1 and 3.4.4.1 of this report.

The VME crate A crate contains a VME bus (section 3.1.1) on the backplane. On this bus are plugged cards in goal to communicate together.

The VIC cards The VIC cards (section 3.1.2) give the possibility to interconnect in a transparent way up to 15 VME crates via a VMV bus (interconnection between crates). This permits to obtain the same services as a VME bus including more than a 100 cards. In fact, it's slightly slower. Thus, one card on the local VME bus can communicate with another card plugged on a

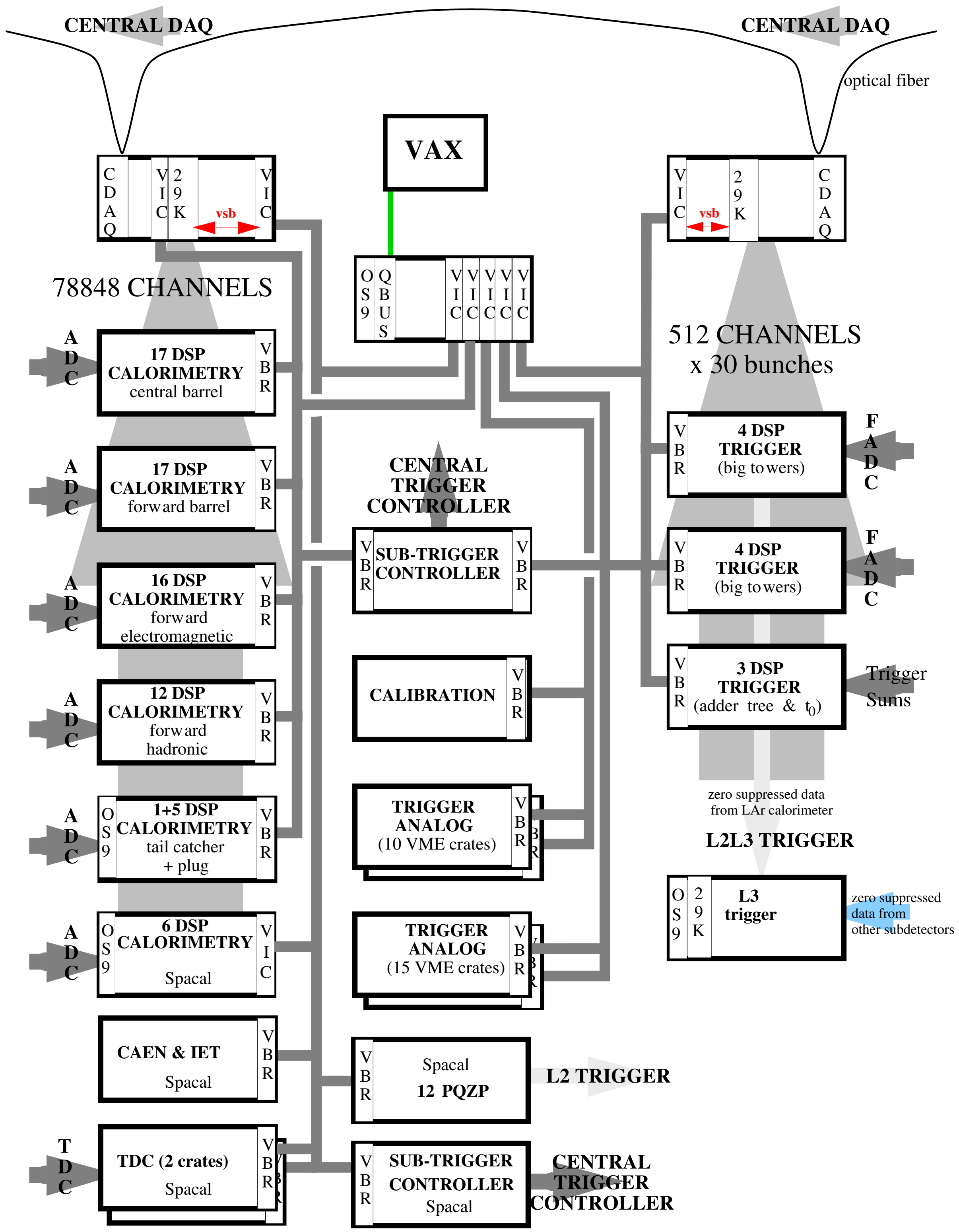

FIG. 2.1 – *Description of the DAQ configuration up to May 2000. The main node is the local crate including 5 VIC cards. This scheme is divided into 5 branches coming from each VIC card.*

remote VME bus via a VIC card and a VMV bus. This is exactly the case on the calorimeter architecture where there is a VME crate including 5 VIC cards, each of them permitting access to the other remote VME crates.

The VBR cards The VBR (Vertical Bus Receiver) cards are simple and passive VMV cards. Usually, VME crates are connected by a VMV bus with on one end a VIC card and to the others, VBR cards.

The OS-9 What is named OS-9 on the graph is in fact a Motorola 68030 card with OS-9 operating system. This card manages all the calibration, the system hardware initialization and the run control.

The VAX The VAX uses a VMS operating system and is an interface to the Motorola 68K card via the QBUS card. It offers a multiuser system, account management and automatic backup, and others services which are not available on OS-9. It will be replaced in the new configuration by a SUN featuring Solaris operating system.

The Event Builders There are 2 Event Builders which are dedicated to the event management. According to Central Trigger Controller which judges if the collisions are interesting or not, they send the data from the calorimeter to the Central DAQ. One Event Builder is dedicated to the trigger branch data, the other to the calorimeter branch data.

The Sub-Trigger Controller It is the only crate to be linked to 2 branches : The Calorimeter branch and the Spacal branch. It's also linked to the Central Trigger which manage the event trigger.

2.1.2 Reasons for Changing the System

Some hardware to change The Motorola 68K processors and the AMD 29K are no more produced. This is a problem for the maintenance of the system because engineers knowing this old systems as OS-9 become scarce.

Towards more speed The H1 experiment is running all day long. In fact, the accelerator can not product endlessly protons, and the experiment cycle (average 15 hours) includes a 1 hour period where the accelerator is stopped. After each stop, initialisation and calibration operations have to be made before running again the experiment. Currently, these operations are long, more than half an hour. This time is due to the hardware capacity and after a few tests on new hardware, we can estimate that these operations could be made in less than 10 minutes.

Towards more services The experiment needs a permanent supervision and there is always a shift crew of 3 persons, 24/24h, on the supervision hall to detect and identify any failure. If the problem is important, they call an expert who has to come on the site at any hour to repare. In fact, analyzing problem, in most of cases, could be made from home or from everywhere by internet. This is now possible by implementing new technology on the system.

2.1.3 The New Architecture

The goal of the new architecture is to offer a better working environment with more services. At the moment, the user environment is determined by the old systems : 68K and 29K cards, OS-9 and VRTX operating systems, the VAX... But everything concerning the hardware of the different detectors is still valid and all the crates containing DSPs, trigger and calibration cards will not change. The main changes concern the 2 events builders and the VIC crate. (fig. 2.2)

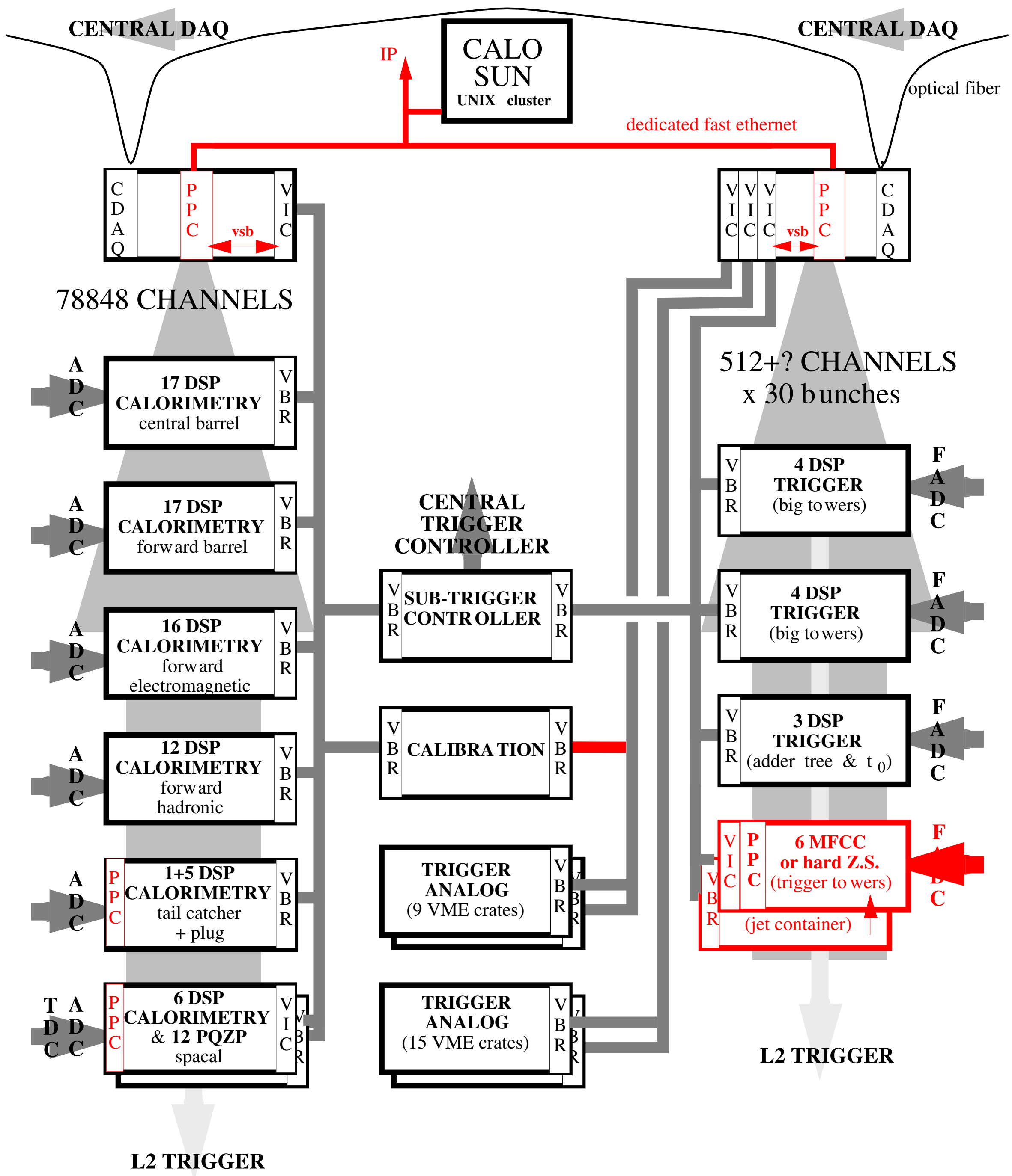

FIG. 2.2 – *The new DAQ configuration planned to be implemented from september 2000. The local crate will be removed and its functions will be realized by the 2 event builders which will include Power PC processors. 4 branches will be still valid. One will come from the calorimeter event builder VIC, and 3 others from trigger event builder VICs. A dedicated fast ethernet will link the 2 event builders and a SUN server.*

The first idea is to replace the old 29K processors in the event builders by Power PC processors. This change imply more powerfull for calculation in the event builders. The VIC crate utilities is mainly to run performing the driver tasks. The driver is the program which manages VME accesses for client processes running on OS9 or VAX. With Power PC on event builders, these can do this task and the VIC crate becomes obsolete.

The VIC crate is also the main node of the 5 branches of the VME tree. This utility will althought shift on the event builders. The Spacal branch of the 5 main branches will be merged with another one. So, only 4 branches of which 2 already exist implicitly, have to be reported on event builders. In fact, the event builders already manage their own branch and only the Trigger Analog branch and the Calibration branch have to be displaced on an event builder crate. On the graph, these branches appear on the trigger event builder.
In fact, the 68K processor of the VIC crate will be replaced by 2 Rio2 cards including Power PC processor plugged each one on an event builder crate instead of 29K processors. These new cards, performing Lynx-OS real time operating system, will have to manage the VME access to their own branch. On the current system, the driver views the calorimeter system as a single VME tree. On the new system, 2 drivers will run, managing their own VME branch access, viewing the calorimeter system as 2 VME subtrees.

The second idea is to replace the old VAX by a new SUN server. The PPCs will be directly connected to the SUN by a dedicated fast ethernet link.

2.1.4 Becoming an Expert

The first objective of this training period was to study the current system and the planned system. Working on the calorimeter is working close to the hardware and before programming whatever, to know very well the hardware principles is required. That's why the first part of this training period is to learn and understand the mechanims and the principles of the VME bus. The VME bus is essential in the system, exactly as VIC cards which are inseparable.

Becoming an expert includes also the software aspects. Actually, during this training, I had to adapt the running software on the new platform (PPC, Lynx-OS).

2.2 Embarking the System Migration - Entamer la Migration du Système

In fact, all these changes can not be made, if the experiment is running. Currently, the experiment is running most of the time. Sometimes, electron proton beams are refilled, and the experiment uses these breaks for calibration, and more time if things have to be repared. This pause allow to make tests on the system. The experiment will be shut down for a few months next september and for the moment, our task is to prepare what will have to be done from september.

2.2.1 VME/VIC Compiler Program

The Compiler program uses a source file to program the VICs and manages dynamically the VME tree. This compiler is a stand-alone test program but uses the functions which are the central part of all DAQ programs.

We have already seen that the VIC cards allow to view, via the VMV bus, a remote VME bus on the local VME bus. VIC cards have to be configured to permit the desired access to VME distant crate. The Compiler program permits to initialize, from an abstract file describing all the Calorimeter DAQ, the local VIC of the VIC crate but also to initialize the distant VIC cards on event builders crates. After the complete initialization of the system, accessing a distant card is transparent through the local VME bus.

At the moment, this program runs on 68K processor and the OS-9 platform. The first step of the project was to plug a Power PC card on the VIC crate without interfering with the whole running system of the experiment and to make the Compiler run on the PPC. We wanted to be sure that all the VIC cards are initialized in the same way that when the program is running on the OS-9 platform.

2.2.2 Use the PPC as Server

The second step was to implement the PPC as a server using IP sockets, treating requests coming from outside. Client stations will be able from a menu program to run on the PPC the VICs initialisation and all other programs. This second point is a new step towards a network implementation.

Each step includes some test phases and we can not directly work on a running system. That's why the work is prepared on a test bench that we had to install and configure.

Chapitre 3

Accomplished Jobs - Le Travail Réalisé

3.1 Discovery and Study of the DAQ System - Découverte et Etude du Système DAQ

I began to study a small but essential part of the DAQ system. The aim is to become an expert of this part and enlarge my knowledge to the close environment when necessary.

We will not make an exhaustive description of the specifications of the different hardwares. Only the interessant part for our job will be retained. The sections 3.1.1 and 3.1.2 are specially destinated to H1 experts.

3.1.1 The VME Bus

References: American National Standard for VME64, ANSI / VITA 1-1994.
: Wade D. Peterson, The VMEbus Handbook, VITA.

Each crate of the system is a VME standard crate. All cards are plugged on the VME bus of the crate backplane and all the DAQ system is closely linked to VME aspects.

3.1.1.1 VME Bus Features

1. The VME bus protocol is a master-slave protocol. That means that the functional modules called masters, transfer data to and from functional modules called slaves. A master can initiate data transfer bus cycles. A slave module detects bus cycles generated by masters and participate in the cycles if they are selected.
2. The VME bus is an asynchronous bus. That means that there are no clocks used to coordinate data transfers. Data is passed between modules using interlocked handshaking signals.
3. Data bus width : 8, 16, 24, 32 bits.
4. Address width : 16, 24, 32 bits.
5. For address and data transfers of up to 32-bits, VME bus uses non-multiplexed architecture. It means that the bus has dedicated pins for addresses and for datas.
6. The VME bus can use the error detection (BERR : bus error)
7. Data transfer rate is 0 to 40 Mbyte/sec.
8. It has flexible bus arbitration and can support up to 21 processors (multiprocessing).

3.1.1.2 Data Transfer Cycles

VME bus offers 8 types of data transfer bus cycles of which Read-Write Cycle, Read-Modify-Write Cycle, Block Transfer Cycle and Multiplexed Block Transfer Cycle.

In this project, only first one interest us. The Read-Write Cycle begins when a master broadcast an address and an address modifier code. When selected, a slave captures the address and responds to the cycle by asserting DTACK (Data Acknowledge), BERR (Bus ERRor) or RETRY. This cycle supports 16, 24 or 32-bit addressing mode.

Below follow some important signals:

- The address modifier (AM0-AM5) code is driven by master. It accompanies an address and indicates the size and the type of address transfer. It is used by slaves to determinate which address lines should be decoded.
- Bus requests (BR0-BR3) are asserted to a requester whenever its master needs the bus. Before accepting the bus, the master waits until the arbiter grants the bus by way of the bus grant daisy-chain (BG0IN-BG3IN).
- The bus busy (BBSY) is driven by masters when they are using the data transfer bus. The arbiter monitors this signal to determine when it should grant the bus to another master or interrupt handler
- The bus grant signals (BG0IN-BG3IN and BG0OUT-BG3OUT) are part of the bus grant daisy-chain and are driven by arbiters and requesters.

The daisy-chain is one the most important aspect for understand the arbitration mechanisms. In fact, all the BGxOUT of a slot is connected to the BGxIN of the following slot. It permits to the cards close to a master to know quickly when the bus is free.

3.1.1.3 Bus Arbiter

A central bus arbiter is required in all VME bus systems. This functional module prevents more than one master from using the data transfer bus at any time. The arbiter must reside in the slot 01 position.

Three main algorithms are used for arbitration : single level arbiter, priority arbiter and round-robin arbiter. On the DAQ configuration, only the single level arbiter algorithm is used.

It monitors only one bus request level and grants the bus using one daisy-chain. In fact it doesn't arbitrate at all. It uses the bus grant daisy-chain to do that job. Since each requester decides to accept or pass the daisy-chain, no central arbitration function is needed.

3.1.2 The VIC 8250

Reference :: VIC 8250, VMV to VME One Slot Interface, CES documentation

One of the main purposes of the VME bus is to permit the easy implementation of architectures allowing the management of complex data streams. Moreover, interconnecting two or more back-planes is particullary needed in large and complex control systems or data acquisition systems as high-energy physics experiments. The VIC 8250 offers transparent interconnection up to 15 VME crates on the VMV bus.

3.1.2.1 The Major Modules of the VIC

VME Slave: This module connects the VIC to other VME master units. The VME Slave resources are mapped to the VME in a 1Mbyte window in the VME standard address space. This VME field is called the "VME CSR space" and allows the VMV master internal register, the VME Slave internal registers (through the VMV), the internal buffer memory and the MMU routing RAM to be accessed. This module also transmits the VME bus cycle to the VMV bus through the

MMU routing module. The VME cycles with addresses decoded by the MMU routing module are transmitted over the VMV to the other VME crates in the system.

VSB Slave: This module is similar to the VME slave module with the same abilities. VSB mapping is done through a 16 Mbyte window, and the offset is software programmable through the VME.

Internal Resources: It contains all the local registers controlling the VIC unit. The control and status information is implemented with 4 Z-CIO devices completely software programmable from the VME side.

MMU Routing: This module acts as a dynamic decoder over the VME bus. The different address fields, of which VME STD and VME EXT are divided into 1 Mbyte windows covering the complete addressing range of these fields(VME STD A24, 16 pages of 1 Mbyte, VME EXT A32, 4080 pages of 1 Mbyte and VSB STD A32 4096 pages of 1 Mbyte). Each VME page can be marked to be taken by the VMV master module and sent over the VMV to the other VMV Slave units.

VMV Master: It drives the VMV bus, incoming VME cycles destined for VMV are controlled by this specific logic. Arbitration phases and handshaking cycles are transparent to the VME/VSB cycle.

VMV Slave: This module decodes the VMV cycles and converts them into one of the four types of cycle : DIRECT cycles to access VMV Slave internal registers, VME-VME cycles to access VME resources, VIACK cycles to generate VME IACK cycles and MEMBUF cycles to access the "internal buffer memory".

VME Master: It converts the decoded VMV VME-VME cycles and the VMV VIACK cycles into standard VME cycles. The conversion is done in real-time at every cycle.

VME Arbitration: It handles the VME arbitration as an arbitration requester and if necessary as an arbiter.

3.1.2.2 CSR Space VME Mapping

The VIC CSR space is mapped onto 1Mbyte in the VME standard addressing space. The base address of the VIC space is set by the hex rotary switch located on the card. For example, if the switch is set to "5", the VIC CSR space is decoded in the VME standard addressing space at address $50'0000 - $5F'FFFF. It is important not to enable the MMU page corresponding to the CSR space because it could cause incoherence in the access to the resources. Enabling the MMU page corresponding to the CSR space will transmit the signal to the remote crate, depending of the configuration, remote cards could respond to the signal in the same way as the local cards and generate incoherent signals on the local crate.

The VIC CRS Space is mainly composed by the following resources :

- Z-CIO registers at $00'0000
- MMU Memory Routing at $02'0000
- VMV CSR Space at $04'0000
- Internal Buffer Memory $08'0000

For our jobs, only the two first resources has to be set up. The VMV CSR Space Registers refers to the VMV configuration and the Internal Buffer Memory is mapped by the different busses through their slave port and can be used as a shared data area for multi-processing operating systems. These 2 aspects will be transparent for us and we will only describe on the following section the Z-CIO registers and the MMU Memory Routing.

3.1.2.3 VIC Internal Resources

Z-CIO Registers The Z-CIO resources are shared in 4 Z-CIO devices which contains the important control and status bits of the unit. Each one contains 3 ports A, B and C, and one control register. The device #0 contains the most important control and status bits of the VIC. The device #1 contains control and status information for the VSB Slave and the MMU logic module. The 2 other devices are dedicated to VME and VSB interrupt controller.

The port A of the device #0 mainly permits to define the VMV arbitration requester mode and to generate a VMV reset for about 100 microseconds. The port B gives information on the VIC card setup as state of some switches and the presence of a VMV terminator TER 6260 needed on the VIC cards ended the VMV bus. The port C gives the VMV Slave crate number.

On the device #1, port C is the most important. This port enables VMV MMU logic master, VME standard port MMU and VME extended port MMU. This port is the very important and problems could come from forgetting to enable the port MMU.

MMU Memory Routing The MMU memory stores page descriptors and the error status associated to each page defined in the system. 16 pages of 1 Mbyte can be stored for VME standard access (A24) and 4080 pages of 1 Mbyte for VME extended address (A32). The VME extended address page descriptors range from $2'4000 to $2'7FBC. The VME standard address page descriptors are from $2'7FC0 up to $2'7FFC.

Each page descriptor is a word of 32 bits composed as following:

bits <00..03>: VMV destination crate address

bits <04..09>: defines the VME address modifier code generated during the VME access in the destination crate.

bits <10..21>: determines high address bits <A20..A31> which map the 1 Mbyte page over the 32-bit addressing range of the VMV.

bit <22>: OK Page. When this bit is set, the corresponding VME page of 1 Mbyte is decoded and passed to the VMV. The page descriptor is validated Very important bit!!!

bit <23>: Word Swapping. When set, access in word or in byte to the resources aligned in 32-bit words is fitted in the regular VME way. This bit must be set to access the internal buffer memory of the other VIC 8250s from the VME pages.

bits <24..31>: Error and Status Report Byte.

3.2 Assembling a Test Bench - Montage d'un Banc d'Essai

The first objective was to plug a RIO2 card (with PPC integrated) and a VIC and try to access a remote VIC. It requires to study hardware and configuration aspects and finally operate.

3.2.1 RIO2 Card Set Up

Once the RIO2 card plugged, the card has to be set up. It is only possible from a console directly plugged on the card. A few monitor commands are available to configure the different parameters of the card. Only 2 commands are essential : SHOW and SET. The first command permits to view the current parameters and the second to set them up.

The boot parameters have to be set up to run the card. The boot command loads and executes the Lynx-OS kernels. But this file can be on a local disk, on the FPROM or on ethernet. The command "set boot" permits to specify the file device, name, type and other parameters proper to each kind of boot. The command "set inet" allow to specify host, boot server, gateway and name

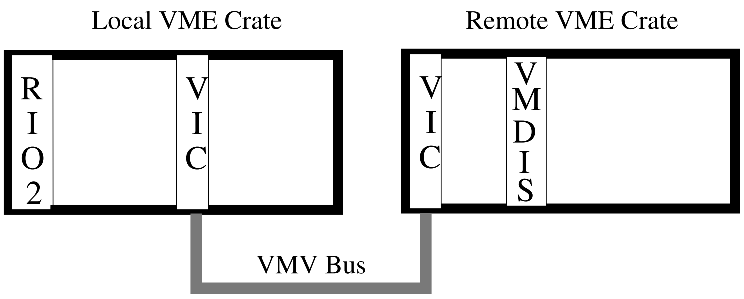

Fig. 3.1 – *View of the Test Bench. The RIO2 card in the local crate is in slot 1 and acts as an arbiter. The VIC in the remote crate is the arbiter of its crate. One test is to access from the RIO2 card to VME Display registers.*

server addresses and mounted directories. For a boot from FPROM, the kernel file lynx.os has to be compiled with the correct parameters and loaded into the FPROM.

On the test bentch, the PPC boots from a local disk. On the DAQ system, the PPC boots now from FPROM. The boot speed is althought independant of the network behaviour and the boot is also insentive to disk loss in accident case.

3.2.2 The Arbiter Card

During the crate configuration design, a card must be chosen for insure the arbiter functions. The arbiter monitors bus requests and grants control of the data transfer bus, one master at a time.

Things to know follow:

- The arbiter must be plugged on the slot 1 of the VME bus.
- The card has to be hardware configured. For example, for the RIO2 card, the switch VME_SLOT1 must be “ON” if the card has to be the arbiter and is plugged in slot 1. This switch must be “OFF” in the others cases.
- If card on crate can be VME master, take care to the daisy chain. The backplane must have some method for closing the bus grant and interrupt acknowledge daisy-chains in empty slots. Jumper pins are placed between the bus modules. Shunts must be installed on these pins whenever the card slot is empty, thereby bypassing the daisy-chains. Without these shunts, cards can be out of the daisy-chain and never become VME master. Concretely, if you observe, with the VME Display card, a Bus Request without Bus Granted, it could mean that the card trying to become VME Master is out of the Daisy-Chain.
- VIC card can be the Arbiter. For being arbiter, a VIC must be equipped with a VME arbitration module (piggy back) performing slot 1 functions.

A valuable device for working on test bench is the VME Display Card. It permits to view all the VME bus lines. Identification of failures and successes become easier with this device.

This first configuration permits me to make my first test programs with the Lynx-OS library for accessing the local VME crate.

3.2.3 Accessing to a Remote Crate

The configuration of the test bench is the following : one VME crate with a RIO2 card plugged in slot 1 as arbiter and one VIC, one other crate with one VIC in slot 1 as arbiter and one VME Display Card. The 2 VICs are linked by a VMV bus. (fig: 3.1)

The VICs at the ends of the VMV bus must be equipped with terminator modules to avoid signal reflection.

Example This example describes how to enable the PPC in the local crate to access (Read/Write) a DSP in the branch 2 VME Remote Crate through the VME Extended A32 mode. After building VME tree, the DSP device is mapped at VME Extended address $123X'XXXX in the remote Crate. The CPU wants to access this card through its VME Ext address $056X'XXXX. After initialization, the CPU can access the card (directly and transparently) at its VME address $056X'XXXX.

The First step is to initialize the VME Extended page descriptor associated with the VME Extended address. The Page descriptor associated to VME External address $056X'XXXX is the page descriptor number 1056. The page descriptor 1056 can be accessed through the VIC CSR Space MMU memory at address $2'0158 ($2'0000 + 56*4). At this address, the page descriptor must be initialized with parameters : Number of crate, Address modifier code, VME address high, OK Page, No swap,... For this example, the content of these fields gives the value $00448F91.

This value must be loaded into the MMU memory of the VIC. After this setup, every memory cycle in the VME External area from $0560'000 to $056F'FFFF in the Master Crate will be sent over the VMV to the Remote Crate with VME External addresses from $1230'0000 to $123F'FFFF.

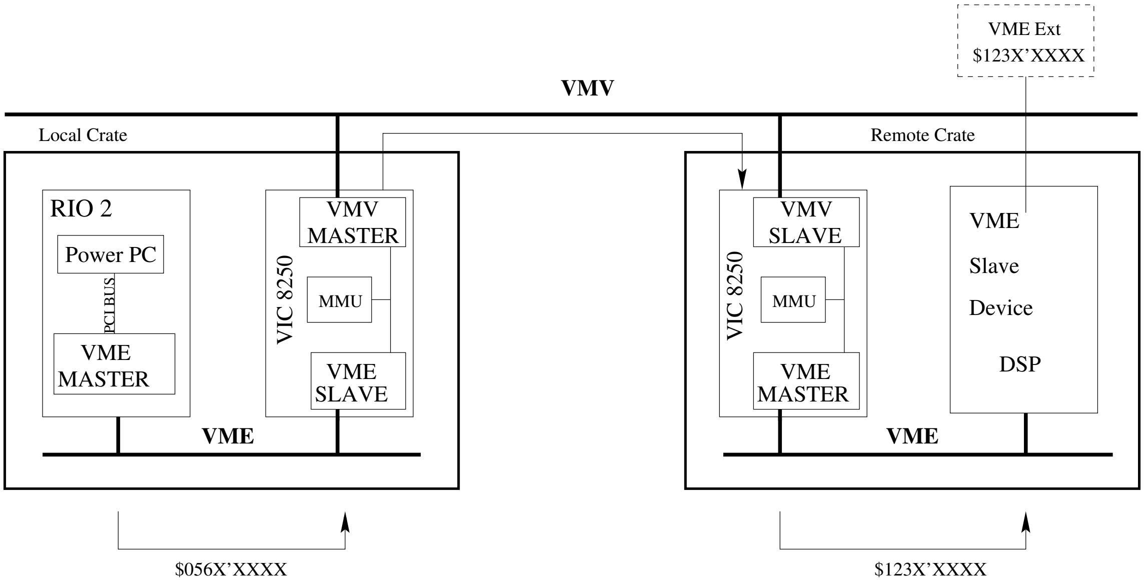

FIG. 3.2 – *A remote crate access example. The Power PC in the local crate accesses to a DSP in the remote crate through its VME Extended A32 address $056X'XXXX. Correctly configurated, the local VIC VME slave recognizes the address, the MMU Routing chip makes the conversion and the VMV master transfers the converted address through the VMV bus to the remote crate. On the remote VME bus, the access appears as an A32 access at address 123X'XXXX.*

Lynx-OS Library The mapping is a way to manage addresses on different bus. On the RIO2, chips and the PPC processor deals with a PCI bus. The different cards in a crate deal with the VME bus to communicate together. The mapping soluce the address translation between PCI and VME.

On the PPC, the mapping to access external cards is static or dynamic. For the first part of my tests, I used the dynamic mapping (Function : find_controller). We will not retain the dynamic mapping for the global system because we need contiguous addresses for all the system and the dynamic mapping offers only maximum 3 Mbyte sized windows. This size is not enough for the mapping of the global DAQ system including about 260 cards. The size of the static window is \$e00'0000 bytes After some tests, we knew that the system needs \$c30'0000 bytes of addressing space. The static window size is althought enough.

The aim of these first tests was for me to understand the principles of VME and how to access to a remote crate. I tried A24 and A32 accesses on the local and remote VME bus. Once, these principles acquired, I passed to the Compiler program adaptation from 68K to PPC.

3.3 Code Adaptation and Tests on the Test Bench - Adaptation du Code et Tests sur le Banc d'Essais

Instead of trying to adapt directly a complex structured program, my objective was to adapt a stand-alone program which uses the same functions as the program running on the DAQ Calorimeter. This program is called compiler and initializes and sets up all the VICs on the local crate and on the event builders.

Here is the structure of the main program Compiler:

```
build_vme_tree(); /* vme tree structure construction */

displayconfig();  /* local vme tree display */

/* local vic initializations and configurations */
vicinit();
vicset ();

ebc_Vme_tree();   /* ebc vme tree construction */

displayconfig(ebc_config,stdout); /* ebc vme tree display */

/* ebc vic initializations and configurations */
vicinit_ebc();
vicinit_ebcSPACAL();
vicset_ebc();
vicset_ebcSPACAL();

ebt_Vme_tree();    /* event builder vme tree construction */

displayconfig();   /* event builder vme tree display */

/* event builder vic initializations and configurations*/
vicinit_ebt();
vicset_ebt();
```

Some of these functions had been programmed ten years before. They were not at the current ANSI standard and few modifications had to be made to run on Power PC.

3.3.1 File "config.vme" and VME Tree Structure

In order to provide some modularity, the boards of the system are organized into subsystems and partitions. A subsystem contains all the boards of a given type. Subsystems are divided into partitions. These partitions are correlated to the detector partitions but not strictly corresponding. All the boards inside a partition will have the same control parameters, whereas boards from different partitions can have different ones.

The complete system is described in the text file "config.vme". In this description:

– Each type of boards is declared in a board dictionary with a logical name
– Each subsystem and VME partition is defined by a number and a logical name
– Each VME partition contains the description of the physical location of cards.

In the board dictionary, boards can be declared with up to 4 address modifiers, which will correspond to 2 kinds of VME access plus 2 kinds of VSB access.

Here is the config.vme file format :

```
* Board dictionary describing the parameters of all the cards types of the system *

Card=DSPC          * Name of the a card type       *
am1=$39/am2=$3f    * Address modifiers of the cards *
Size=$40000        * Size of the card               *
Space=A24D32       * Access 24 bits, Data 32 bits   *

Card=CALI          * Name of another card type      *
am1=$39            * Address modifiers of the cards *
size=$10000        * Size of the card               *
Space=A24D16       * Access 24 bits, Data 16 bits   *

* System description *

System=DaqCalo     * Logical name given for the whole system *

SubSystem=CALO     * Logical name of the subsystem          *
        Number=1/CardType=DSPC     * Number of the subsystem, the card type it uses *
                Partition=FB       * Logical name of the partition                  *
                Number=1           * Number of the partition                        *

                        Location=remote/branch=1/Bustype=both
                     * Card access through the branch 1 of the VMV *

                        crate=2/NumberOfCards=17/NumberOfPresentCards=1
                     * 17 DSPC cards forseen in the crate 2. Only one is present *

                Partition=CB
                Number=0
                        Location=remote/branch=1/Bustype=both
                        crate=1/NumberOfCards=17/NumberOfPresentCards=17

SubSystem=CALIB    * New subsystem declaration              *
        Number=3/CardType=CALI
                Partition=Lar
                Number=0
                        Location=remote/branch=3/Bustype=vme
                        crate=$5/NumberOfCards=1/NumberOfPresentCards=1

End                * End of the file *
```

The function "build_vme_tree" parses and analyzes the file config.vme. After compilation by the routine "build_vme_tree", the system organization is stored in a tree structure. This structure is defined in vme_tree.h file and is as on the figure 3.3. Moreover, the function "build_vme_tree gives a unique address for each card of the global system and these addresses are contiguous. This is important for the mapping and for the VIC initialisation that we will see later.

3.3.2 Initialisation and Set Up Functions

Each initialisation functions is composed by the following calls:

```
VicVmeMap();    /* Map the VIC8350 base address on VME             */

VicCsrMap();    /* Initialize the pointers to the VIC8350 resources */

VicZinit();     /* This function initialize the four Z8536         */
                /* controllers according to the VIC8250 specifica-  */
                /* tions. The devices are first reset and at the    */
                /* end data ports are enabled and interrupt are     */
                /* left disabled.                                   */

VicVreset();    /* This function sends a few hundred usec           */
                /* reset pulse to the VMV.                          */

VicMmuClr();    /* Clear MMU RAM                                    */

VicVmvArb();    /* This function allows to read and over-           */
                /* write VMV arbitration mode bits in ZCIO dev 0    */
                /* port A.                                          */
```

Whatever the functions, each setup functions is composed by the following calls:

```
VicVmeMap();     /* Map the VIC8350 base address on VME             */

VicCsrMap();     /* Initialize the pointers to the VIC8350 resources */

VicMmuWin();     /* This function allows to write read and          */
                 /* enable a page in the MMU ram                    */

VicMmuEna();     /* This function allows to access the MMU           */
                 /* RAM control bits in ZCIO dev 1 port C.          */
```

All these functions are included in the vmeusr library. They are all independent of the processor except the function "VicVmeMap" which does the mapping for a FIC8230 in VME A24D32 addressing mode. This is no more valid. Now, the vmeusr library contains the VmeAddA32, VmeAddA24 and VmeAddA16 functions which returns static mapping window address for Power PC in the different addressing mode.

The function vic_init searches the local VIC in the VME structure and initializes them. The function vic_set parses all the VME structure and opens on each local VIC, the pages corresponding to cards on its branch. On the different VICs of the DAQ system, about 550 pages are opened.

The difference between initialisation and setup routines for the local crate and the event crates is at parsing and analysis VME tree structure level. On the main program, a first VME structure is made from the config.vme file for the local crate initialisation and setup calls. Two other structures are made later by re-handling the first structure for event builders functions. We will not detail this functions and their mechanisms because they will not be used in the new system.

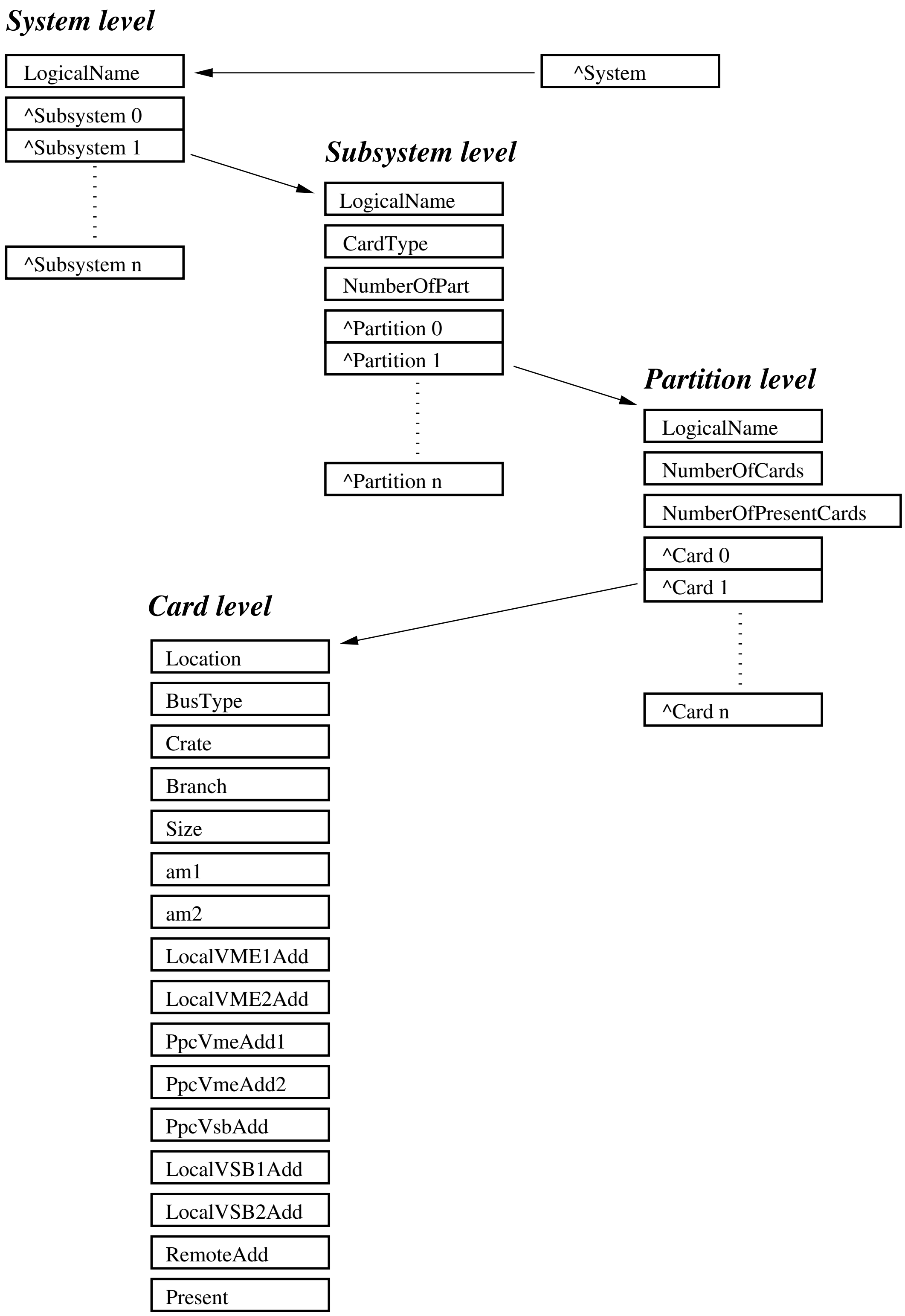

FIG. 3.3 – *The VME tree structure. Pointers are represented by "^name_of_the_pointer" and the pointed structure are designated by arrows.*

3.3.3 Code Adaptation

The adaptation of the code consisted of several tasks:

- Translate the code to ANSI C standard
- Date and time OS-9 functions have be replaced by Lynx-OS functions
- Changing the static window addresses corresponding to the PPC system.

Moreover, we observed during our tests that opening on a the VIC a page 47 for example, opens the page 46. In the Compiler code each page opening was preceded by an increment in the page number. This was a mistake in the current CES library where the VIC MMU RAM was defined as below:

```
struct vic_mmuram
{
        long vsbs[0xfff];         /* MMU VSB space */
        long vmee[0xff0];         /* MMU VME extended space */
        long vmes[0x10];          /* MMU VME standard space */

};
```

And in the documentation, the MMU RAM was defined as following:

```
struct vic_mmuram {
        long vsbs[0x1000];        /* MMU VSB SPACE */
        long vmee[0xff0];         /* MMU VME EXTENDED SPACE */
        long vmes[0x10];          /* MMU VME STANDARD SPACE */
};
```

The second version is right. In the old version file, the MMU VSB space was reduced of 1 byte to permit compilation. OS-9 compilators has a 32K limitation on C structures. That's why the structure has been reduced and programmer had to make an increment for opening a VME Extended pages and non increment for VSB pages. On the PPC, this limitation doesn't exist and we implemented the good version of the library.

For more legibility on the test bench, we implemented a new function for catching bus error.

3.3.4 Results on the Test Bench

On the test bench results were limited by our configuration including only 2 crates and 4 cards. Our objective on this test bench was to understand the mechanisms, adapt as well as possible the code, trap all the bus errors and be sure that no "segmentation fault" will appear when we will try the compiler on the real system.

Segmentation fault means that forbidden space is written. On the DAQ system, it could cause damages and imply dangerous and impredictable results.

With a config.vme file describing the test bench, the compiler was running without problem. With the config.vme describing the calorimeter, a lot of bus error were caught without segmentation fault.

3.4 Test on the Real System - Test sur le Système Grandeur Nature

3.4.1 The Vicscan Program

Testing on the calorimeter implies to be efficient because working time is reduced. We implemented the vicscan program in order to view the VIC status. This program checks all the VICs present in the local crate and show the Z-CIO registers status and all the A32 pages opened on the VIC MMU RAM.

3.4.2 Plugging the Card

The environment at the North Hall where the experiment is situated, is noisy because of the processors, the cooling and the refreshing air. A lot of cables are connected in front of the crates and plugging the PPC card needs a lot's of concentration and precaution. The following procedure is to avoid the incident risks which could block the experiment:

1 Working in accordance with the shift crew. The calorimeter branches have to be switched off for stopping recording on it.
2 Notice the current configuration of the crate to be able to come back in problem cases.
3 Switch off the crate, plug out the cards which close the visibility and verify the daisy-chain. It can be necessary to added or removed some jumpers in the backplane.
4 Plug all the cards including the new one and switch on the crate.
5 Ask to the shift crew to make a "Warm Start" (initialisation jobs) to verify the system.
6 In case of problems, come back to the old configuration (remove the card and verify jumpers) and try to diagnose the errors.

Currently, the PPC is booting from the FPROM and is connected to the network by ethernet. We can now log on it from any stations.

3.4.3 Running the Compiler Program

After running the compiler the vicscan permited us to view some surprising results:

- "VICs" at the 16 possible addresses although only 5 VICs are on the bus.
- After 2 runs without new initialisation, the vicscan results were always different : number of opened pages, not necessarily the same opened pages and some same opened pages but with different parameters.
- At the VIC addresses, we found some pages correctly opened, missing pages and some pages which didn't have to be opened.
- This facts were independent of the initialisation type (OS-9 or Lynx-OS).

Below, an extract of our file result corresponding to the configuration of the branch trigger VIC is given. Some explanations are in comments :

```
Dump of VIC at CPU address 0xDE900000 (VIC #9):
                      /* There is really a VIC at this address */

 ZCIO #0:
  C (VMV slave #)    =  0x7
  B (mbox1-3,VMV)    = 0x70                /* These values are good for the Z-CIO   */
  A (VMVarb, ...)    = 0x00
 ZCIO #1:
  C (VMV enable)     =  0xF VMV VSB VMEA24 VMEA32
 MMU: VME(A32) 00A: VMV=4 AM=39 pg=000 OK /* First opened page : VIC 2.1           */
 MMU: VME(A32) 056: VMV=1 AM=39 pg=000 OK /* FADC.0.0 FADC.0.1 FADC.0.2 FADC.0.3   */

                                          /* Here is missing a page for FADC.0.4!!!*/

 MMU: VME(A32) 058: VMV=3 AM=39 pg=000 OK /* FADC.1.0 FADC.2.0 FADC.3.0 FADC.4.0   */
 MMU: VME(A32) 05A: VMV=1 AM=39 pg=001 OK /* DSPT.0.0 DSPT.0.1 DSPT.0.2 DSPT.0.3   */
 MMU: VME(A32) 05B: VMV=2 AM=39 pg=001 OK /* DSPT.0.4 DSPT.0.5 DSPT.0.6 DSPT.0.7   */
 MMU: VME(A32) 05C: VMV=3 AM=39 pg=000 OK /* DSPT.1.0 DSPT.2.0 DSPT.3.0 DSPT.4.0   */

 MMU: VME(A32) 05D: VMV=F AM=09 pg=040 OK /* This page must not be opened !!!      */
 MMU: VME(A32) 05F: VMV=F AM=28 pg=07F OK /* This page must not be opened !!!      */

 MMU: VME(A32) 077: VMV=4 AM=09 pg=001 OK /* 29K.1.0                               */
 MMU: VME(A32) 079: VMV=4 AM=39 pg=002 OK /* FICVRTX.1.0                           */

                                          /* Here is missing a page for MEB.1.0!!! */

 /* and it still continues in the same way */
```

In fact, the 68K card, according to the config.vme file, in the local crate having $7F'FFFF address space reserved for VME slave on the bus, have in fact $100'0000 address space reserved. It means all the A24 address space. The 68K card answered to any addresses and caused interferences during accesses.

After closing the FIC A24 slave window, the configuration on the local VICs after initialisation from 68K and after initialisation from the PPC are the same. Moreover, the event builders VICs initialisation and configuration was also running without problem.

3.4.4 The Following Steps

3.4.4.1 Adapting the Compiler for Event Builders

In the new architecture, all the VIC initialisation will be made from the local crate. One PPC will be plugged in each event builder and each PPC will only initialize its local crate and its branches. For simplicity and easier maintenance, we want that there will still be an unique "config.vme" file. This is important to avoid 2 problems :

- Giving an unique address for each card with 2 independant files on 2 different processors could not be easy.
- Someone could modify one "config.vme" file and forget to change the other one.

One unique file will describe the system and the same analysis programs will run on the 2 PPCc. In this way, the given address for each card of the system will be the same for each PPC and for all the calorimeter.

To initialize only the local branch, we chose to include in the "config.vme" the IP name of each of the PPC and the branch number that each one has to initialize. For example:

```
Host=ambon        * Host name is ambon              *
        Branch=1
        Branch=2  * initialize the branch 1, 2 and 3 *
        Branch=3

Host=buru         * Host name is buru               *
        Branch=4
        Branch=5  * initialize the branch 4 and 5    *
```

This aspect is now implemented in the compiler. To recognize which host the event builder is, the compiler use the simple function "GetHostName". I had to re-handle the configuration file analysis functions for parsing this new structure ahead of the config.vme file. This data are stored in a table of "HostStruct" :

```
struct HostStruct { char HostName[StringSize];
  char HostBranch[NumberOfBranchs];};
```

Before running the initialisation of the VIC, the compiler now verifies in this table, if it has to initialize the branch.

3.4.4.2 Some Modifications in the Library VME_UTILS

In the display and initialisation VME_UTILS library, the functions fic_vme_add1, fic_vme_add2 and fic_vsb_add calculed the FIC VME and VSB addresses. These functions are now replaced by 3 new fields in the card structure : PpcVmeAdd1, PpcVmeAdd2, PpcVsbAdd (Rf: Figure 3.3). In the PPC implementation, these fields contain the result of these functions. The calculation of PPC VME address is now implemented once at the VME tree building and put in the fields PpcVmeAdd1, PpcVmeAdd2 and PpcVsbAdd.

3 macros “PpcVme1”, “PpcVme2” and “PpcVsb” are also implemented and each call to these functions will be replaced by this macro in the adaptation of the other programs running on the DAQ.

3.4.4.3 Compiler and Client-Server Architecture

The shell command for running the compiler is : compiler <option>.

Six options are available:

- local : initialize the local crate VICs
- ebt : initialize the event builder trigger crate VIC
- ebc : initialize the event builder calorimeter crate VICs
- displocal : display the complete VME tree configuration view by the local crate
- dispebt : display the complete VME tree configuration view by the event builder trigger crate
- dispebc : display the complete VME tree configuration view by the event builder calorimeter crate

A server daemon called “daqd” permits remote communication by IP sockets. It receives from clients strings and depending on these, executes by system calls the compiler commands. Currently only the command “compiler” is available for the server. When this command is received by the server, this runs successively the three initialization commands:

- compiler local
- compiler ebt
- compiler ebc

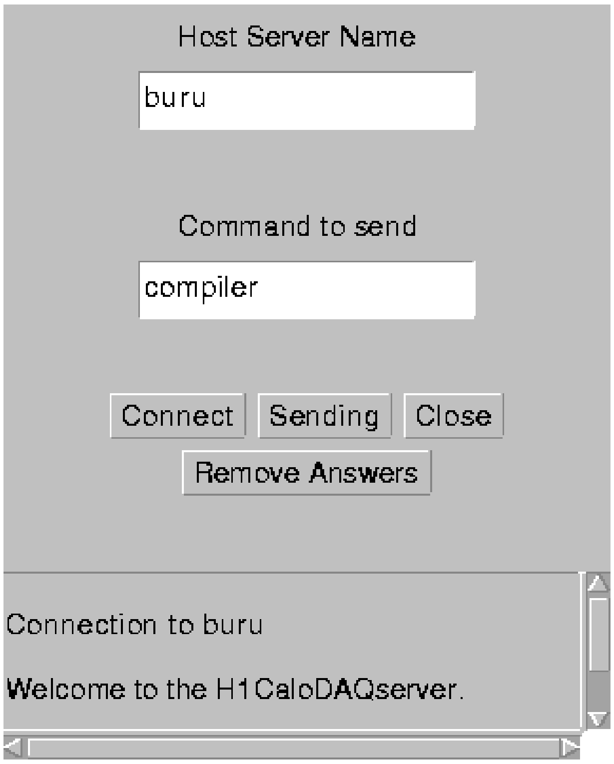

FIG. 3.4 – *The JAVA Client Interface. Connection to the server by this interface is possible by specifying the name of the host where the server is running and by clicking on the "Connect" button. The "Sending" button sends the string in the window "Command to send" to the server. "Close" button closes the current connection. The button "Remove Answers" clears the window just below.*

I've programmed the JAVA client. Fitted with a graphical interface, it permits to connect to a server by IP socket, to send commands to the server and to receive answers on a small window. It uses AWT (Abstract Windowing Toolkit) graphical objects. It is mainly composed by an action listener and the server listener. The action listener allow the user interaction by listening mouse events. When buttons are clicked, corresponding functions are called.

The server listener thread is started and stopped by connection and disconnection to the server. This is an endless loop listening on the current socket and displaying on the windows, streams from the server.

3.5 Contributions and Results - Apports et Résultats

3.5.1 A Faster Compiler

According to CES engineers, it was the first time that VICs are managed by Power PC. It was althought not evident that this confuguration could work and it necessited to be tested.

Currently, we have now a compiler program running on PPC with exactly the same effects as the program running from OS-9. It sets up all the system, close to 550 pages are opened on the different VICs. The compiler is four times faster on PPC than on the 68K. In fact, the VME bus transfer rate can not be accelerated but all concerning the processor is faster.

Functions	OS-9 / 68K	Lynx-OS / PPC
build_vme_tree();	3 400 ms	137 ms
vicinit(); vicset();	3 900 ms	2 436 ms
ebc_Vme_tree();	270 ms	5 ms

vicinit_ebc();
vicinit_ebcSPACAL();
vicset_ebc();
vicset_ebcSPACAL();
	180 ms	35 ms

ebt_Vme_tree();	260 ms	5 ms
vicinit_ebt(); vicset_ebt();	140 ms	20 ms
TOTAL	8 150 ms	2 638 ms

The access to the config.vme file and the VME tree building is 25 time faster than on the 68K. We observed that access to the file via ethernet is as fast as accessing directly to the local disk. This result is althought only due to the power of the PPC processor which has to analyse, parse a file and build a tree structure. This result was predictible because it is only processor operations.

Likewise, initialisation and configuration of VIC is one third faster. VME accesses are not faster than possible but parsing the VME tree structure is a considerable time factor.

3.5.2 A Client-Server Architecture for the Compiler

We have now a compiler running on PPC, a server program and a client applet. This is the complete test structure we wanted to develop for this training period. Tests have been successfull and it is a real progress for the futur DAQ implementation. We are now able from any station to initialize and configure the calorimeter DAQ by a graphical applet.

3.5.3 Ready for Changing the DAQ Architecture

VME management on PPC with all VME tree initialisation and the VME server architecture are implemented. A lot of work is still waiting from september with performing final hardware migrations and finalizing the adaptation of all user procedures.

CONCLUSION

L'expérience professionnelle Ce projet fut un vrai travail d'ingénieur puisqu'il ne m'a pas simplement fallu mettre en oeuvre des connaissances acquises en milieu scolaire mais plutot découvrir de nombreuses caractéristiques d'un matériel qui m'était inconnu. De plus, ce stage était très complet puisqu'il recouvrait aussi bien des aspects hardwares que logiciels. Même si le soutien d'experts m'a quelques fois été nécessaire pour dénouer les problèmes, j'ai acquis une certaine façon d'appréhender les difficultés. Il faut rajouter à cela que le travail est très pointu au niveau hardware et que l'expérience y est un atout majeur. Face à l'immensité de connaissances à découvrir dans ce milieu, on a parfois l'impression de tout découvrir mais c'est ce qui fait la richesse de ce travail.

DESY est une immense collaboration de nombreux laboratoires et les expériences tournent principalement grâce à l'implication et à la bonne volonté de chacun. L'esprit de collaboration est quelque chose qu'il faut retenir de cette expérience car elle a été et restera probablement le fil conducteur qui fait avancer la connaissance.

Les résultats De ce projet de longue haleine, ce stage de quatre mois m'a préparé aux 16 suivants qui composeront ma coopération au service militaire. J'ai désormais abordé et acquis un certain nombre de connaissances qui seront essentielles pour la suite du projet.

Pratiquement, je suis rentré de plein pied dans le vif du sujet. L'adaptation du code au Power PC ainsi que le matériel rondement testé sont de bonne augure pour la suite puisque les résultats ont été concluants. Nous avons préparé au mieux le shut down du sytème qui aura lieu en septembre, puisque nous savons d'ores et déjà que l'initialisation et la configuration des VICs par le programme Compiler à partir d'un Power PC se fait sans problème et plus rapidement que sur l'ancien système. De plus, l'architecture client-serveur est déjà mis en place avec la possibilité de lancer à distance les procédures implémentées sur Lynx-OS.

L'expérience humaine Si dépourvu de bien des repères, l'adaptation au milieu n'a pas toujours été évidente, j'ai beaucoup appris à DESY que ce soit sur le plan professionnel comme sur le plan humain.

Partir dans un pays dont je ne pratiquais pas la langue était un défi que j'ai toujours voulu relever. Si on ajoute à cela que la physique des particules était un domaine que je connaissais peu et qu'à DESY de nombreuses cultures sont représentées, vous comprendrez que cette expérience est pour moi unique et que je suis heureux qu'elle puisse continuer pendant 16 mois encore.