CERN 86-01
Data Handling Division
30 January 1986

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

VMEbus IN PHYSICS CONFERENCE

CERN, Geneva, Switzerland
7th and 8th October 1985

PROCEEDINGS

GENEVA
1986

CERN — Service d'Information scientifique — RD/699 - 2300 - janvier 1986

ABSTRACT

The first conference "VMEbus in Physics" was held at CERN on 7th and 8th October 1985. The conference surveyed the applications of the VMEbus standards in physics, with special emphasis on particle physics and accelerator control. Developments in the definition of the standards and in the formation of users groups were discussed. Manufacturer's representatives were given the opportunity to appreciate the requirements of the fast-growing VMEbus market in the physics community. These proceedings contain the unedited text of the oral and poster presentations given on that occasion.

PREFACE

That VMEbus is a standard already accepted by a large number of physicists and engineers working in experimental physics, was amply demonstrated by the number of participants to this first "VMEbus in Physics" Conference, held at CERN on 7th and 8th October 1985. The Organising Committee had the pleasure of welcoming a total of about 300 people from CERN member states and a number of other countries. Some 50 of the participants were from industry, which was an encouraging sign of the cross-fertilisation of ideas already bringing benefits to both manufacturers and users in this rapidly evolving technology.

The Conference consisted of eight invited papers covering aspects of the organisation of VMEbus in industry, of user groups, the progress of standardisation and of the major applications of VMEbus at CERN. In addition to these invited talks, 19 papers and six posters were presented and one round-table discussion was held.

In order to be able to produce these proceedings with the minimum of delay and hence with the maximum of benefit to the participants and the VMEbus user community as a whole, no editing has been done. The contents of the texts, as published, are therefore the sole responsibility of the respective authors.

The Organising Committee would like to thank the speakers and presenters of posters for their participation and all those who were involved in the running of the Conference for their contribution to its success.

The Organising Committee

CONTENTS

Page

Page

PRIAM and VMEbus at CERN

C.Eck

Geneva, 10 December 1985

1. Introduction

PRIAM is the French acronym (PRojet Interdivisionnaire d'Assistance aux Microprocesseurs) for Interdivisional Project for Microprocessor Support.

The CERN management established the PRIAM project in June 1983 following the recommendation of CERN's Steering Committee for Microprocessor Standardisation. This committee studied the problem of hardware and software support for 8 bit and 16/32 bit micros at CERN and made the following main recommendations:

- 8 bit systems should use the MC6809 processor and the G-64 bus

- 16/32 bit systems should use the M68000 processor family
and the VMEbus

This paper will concentrate on the PRIAM work in the 16/32 bit area.

2. Areas of PRIAM Activity

2.1 Information

PRIAM has to market its services, to inform prospective users about them. And not the last of its services again is information distribution. How is this done?

The PRIAM project, together with the Online Computing group at CERN, publishes four times per year the "Mini and Micro Computer Newsletter", MMCNL [1],which is distributed free of charge in more than 800 copies (550 outside CERN). It contains articles on PRIAM work, contributions by users, and a reference section with pointers to people and additional information channels.

On CERN's IBM (HELP PRIAM under Wylbur) and on the PRIAM-VAX (use apropos and/or man commands) simple documentation access schemes have been installed. These tools are rather crude but have proven to be useful for people at CERN and regular collaborators. PRIAM plans to update/improve this information channel and to discuss how to make it available to a broader user community. The antisocial behaviour of a shabby group of people, Hackers and Co, make this a difficult task.

PRIAM organises, through the CERN stores, access to printed documentation like: Compatible Products Directory, Specification Manuals.

CERN, represented by the PRIAM project, is a Regular Member of VITA, maintaining a useful information flow in both directions.

And finally PRIAM organises seminars, presentations, and last but not least events like this conference.

2.2 Training

To offer training on various items connected with its support work is an important activity of PRIAM.
Courses on MC68000, VMEbus, RMS68K, UNIX, MC68020 have been and will be offered at CERN. It is a difficult task to find the right level of a course for people with widely differing prior knowledge and to get instructors for general, non commercial, courses.
PRIAM urges VITA to establish a comprehensive offer of courses on all aspects of the VMEbus family of standards.

2.3 Consultation

Compared to the size and complexity of its task, the PRIAM project has a very limited number of staff. PRIAM support has therefore to concentrate on general tools and services and cannot get involved in direct help for specific applications. Nevertheless good advice is available to people starting on a development or wanting to use a product in the area of PRIAM responsibility. These consultations are as important for PRIAM as they are for a client or prospective client of PRIAM. The client should not get lost in exotic choices and the project has to keep aware of what is going on and how well its services fit the users needs. PRIAM's developments need the innovative ideas which come from a wide user base.

2.4 Hardware Support

Again, full support of VMEbus hardware would require several times the number of staff PRIAM can count on. The project has therefore to concentrate on the most essential elements of such a support. They can be split into two broad classes: providing information and recommendations concerning the VMEbus market on one side, and helping that CERN staff and collaborations to obtain access to this market at the most favourable conditions. The first point requires a constant survey of the VMEbus market and multiple contacts with manufacturers. With a fast growing market this becomes more difficult. VITA's Compatible Products Directory will help us in this respect.

It is necessary to evaluate and test certain products in detail. Hopefully such evaluations can in the near future concentrate on the functionality of a module, the compatibility with the VMEbus specs having been tested in one of the certification labs being set up with the help of VITA. PRIAM will also offer its services to channel information back from CERN users to VITA and the manufacturers. After internal discussions this should allow for example to present a consistent demand for development of a certain module or even the VMEbus standards.

The second point, access to the market, can be partially fulfilled by keeping the most common parts (backplanes, connectors, chips etc) in the CERN stores. Providing access to board level products is much more complicated. Buying a large stock of CPUs, memories etc. for later redistribution would be a very risky operation considering the rapid development of the market. Random purchases of small quantities by numerous groups at CERN will not very likely achieve good commercial conditions, let alone the increased effort of justifying each purchase decision to the CERN administration.
The PRIAM project decided, therefore, to set up so called blanket orders with a small number of firms, selected on the basis of a formal request for information (sent out mid November '85). Selected firms would have to open up their complete product range to buyers from CERN with favourable conditions (discounts, delivery times, service and repair agreements). In return they will get the major part of CERN's orders of VMEbus equipment. This procedure will need regular revision, based on the experience with it and on major developments in the VMEbus market.

2.5 Software Support

Every group trying to set up a support for microprocessor software development will have to answer the crucial question: what can be bought from outside and what has to be developed in-house? Valid answers to this question will vary widely according to circumstances. PRIAM's decision to base its software support mostly on in house developments is the result of the following considerations:

- One cannot buy, what is not available yet. Some of the developments of PRIAM supported software started well before PRIAM (around 1980), little good quality micro-processor software was offered by industry.
- The distribution of PRIAM software without costly licences is a must. CERN collaborates with many small institutes.
- PRIAM software must be portable to different brands of computers used by the collaborating institutes. Portability allows software to be installed on new workstations coming on the market.
- PRIAM software should be consistent and coherent; following the same conventions, having the same user interface, allow free language mixing, supported by a common symbolic debugger.
- And finally, in a highly competitive research environment quick response to user demands for bug-fixes or developments has a non-negligible value.

The status and the next developments of PRIAM software are presented in the following sections.

2.5.1 Working Environment

A programmer using PRIAM software should be able to develop his code on a single-user workstation, networked to powerful servers, link it to executable packages and downline load it into the target for execution. Execution can be under the control of a symbolic debugger running mostly on the comfortable workstation with only a minimal part of the debug monitor executing in the target.

For a hardworking developer of microprocessor code this looks too good, to be true: and it is not yet completely true.
PRIAM software is however not far from this picture already and the plans are to provide exactly the above environment around the middle of 1986. Today most of the cross software development is still done on larger timesharing computers (IBMs, VAXes, ND-500s etc.), the main workhorse being a VAX-11/785 running Berkeley BSD 4.2 UNIX. But the cross software written in portable Pascal, has been ported to some small workstations with good networking capabilities and PRIAM will soon select a few supported types.

The symbolic debugger is still target resident (including its symbol tables) but work has started to split it up and define a suitable host/target protocol.

2.5.2 Language Support

PRIAM supports a macro assembler, M68Mil[2], and Pascal, Modula-2 (mainly for accelerator control applications), Fortran-77 (mainly for physics applications), and C (mainly for communications software).

Figure 1 gives an overview of PRIAMs cross-software. All language processors follow the same programming conventions [3], producing the same object code format, assuring free language mixing and a common symbolic debugger. A slightly modified version of the object code format CUFOM [4] (CERN Universal Format of Object Modules) has been standardised by IEEE under the name MUFOM. The pusher will either produce Motorola S-format for downline loading into the target or a format suitable for a DATA I/O PROM programmer.

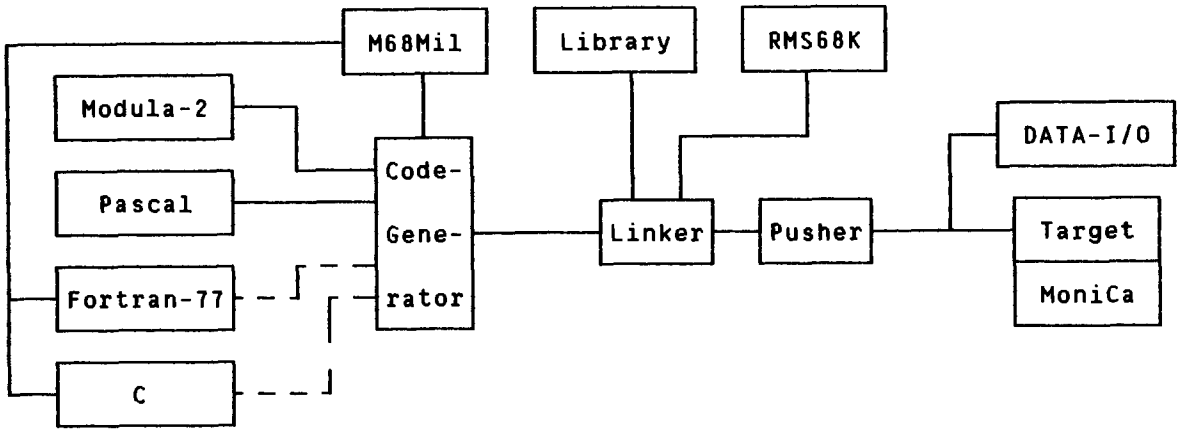


Figure 1: PRIAM Cross-Software

The elegant and practical choice of a common code generator for the compiler front-ends is at the moment only realised for Pascal and Modula-2 whilst Fortran-77 and C compilers (bought from industry) produce assembly code, which has to pass via M68Mil with a corresponding loss in symbolic debug information. A Fortran-77 front-end is in production and will be available mid 1986. Work has started on a C front-end, but lack of manpower does not allow a definite completion date to be quoted.
The code generator for the MC68020 is in its final test phase.

2.5.3 Debug Monitor

The name of PRIAM's debug monitor is MoniCa: "Monitor in Camac", where it originated in 1980. MoniCa provides interrupt driven I/O based on logical channels, named MIOS (MoniCa I/O System), run-time support in the form of integer and floating arithmetic and exception handling, and finally symbolic debugging for all languages following the PRIAM programming conventions. Version 1.0 of MoniCa [5] (in field test now) contains MC68020 support and a line by line assembler. 80% of MoniCa is written in portable Pascal, which will ease the task of splitting it into a host resident and a target resident part for remote debugging. MoniCa contains some 30 commands in an open ended structure, allowing specific user commands to be added.

2.5.4 Real Time Kernel

MoniCa is sufficient for single task systems plus interrupt driven I/O. Applications, which need multitasking, are supported by PRIAM via the RMS68K real time kernel. RMS68K is integrated into the PRIAM software. A directive library is callable from PRIAM supported high level languages. I/O is based on MIOS and MoniCa can run as exception monitor task of RMS68K providing symbolic debugging in a multi-task environment. Version 4.4 of RMS68K [6] runs also on the 68020 and multi-processor support is announced for a later version.

3. Standards and Support

As a conclusion of this short presentation a few words on these heavily inter-related terms, which will clarify the position of PRIAM.

A standard without the necessary level of support will never be accepted by users, and support can in no way achieve its goal without a prior selection of the items to be supported. Efficient support will therefore either create an "in-house standard" or, better, reinforce an existing, carefully chosen, standard.

Every choice is sometimes felt as a restriction. It is only natural that a support group will be put under pressure to change some of its decisions. And here another important term has to be considered, namely stability. Support decisions have to be stable to protect acquired know-how and capital investments. But how can one achieve stability in an area as rapidly changing as microprocessors and bus standards?

The only possible way out of this dilemma can be found by properly understanding what a standard should be. A healthy standard, a standard to which a support group should be able to stick to, should be seen as a living entity. It must be able to develop and adapt, either via compatible extensions or via new members within a family of standards.
Development needs a lot of pushing and pulling and lobbying. And the value of a chosen standard for a support group will depend heavily on the extent of the influence the community of users of this standard has on the body who is developing (maintaining) the standard.

VITA, the VMEbus International Trade Association, is the body maintaining the family of VMEbus standards. The constitution of VITA, which is not dominated by a single firm, and their attitude towards participation of the user community in the association should make the VMEbus standard ideal for a support group.

A lot will depend now on the activity of VITA user groups. Let us hope that this conference has helped to trigger activities in Europe.

* * *

References

[1] Mini and Micro Computer Newsletter, CERN, Data Handling Division
Editor: Chris Parkman (Tel: +4122833963)
Distribution: Ketty Protoulis (Tel: +4122835839)

[2] Software Support for Motorola 68000 Microprocessor at CERN
M68Mil Cross Macro Assembler
Horst von Eicken, CERN 83-12, Data Handling Division

[3] Software Support for Motorola 68000 Microprocessor at CERN
CERN Convention for Programming the MC68000 Family
B.Carpenter and R.Cailliau, Cern 84-12,Proton Synchrotron Division

[4] Cross Software Using a Universal Object Module Format, CUFOM
Jean Montuelle and Ian M. Willers
EURO IFIP 79, LONDON,26-29 September 1979

[5] Software Support for Motorola 68000 Microprocessor at CERN
MoniCa Reference Manual
Horst von Eicken,in preparation,CERN,Data Handling Division

[6] M68000-Family Real-Time Multitasking Software User's Manual
Motorola Inc., Microsystems
M68KRMS68K/D9,March 1985, Ninth Edition

INTRODUCTION TO VITA

L. Hevle

Executive Director
VMEbus International Trade Association
Suite #E, 01229 N. Scottsdale Road
Scottsdale
AZ 85253
USA

Abstract

VITA (The VMEbus International Trade Association) is an independent "not-for-profit" organisation whose object is to promote VMEbus technically as well as commercially.

VITA is organised under a Board of Directors and an Executive Director into three committees: Technical, Promotional and Users. Membership is at one of four levels: Sponsor, Senior, Regular (all of which have voting rights) and Associate (for individuals).

VITA OBJECTIVE

. . . TO ACCELERATE THE VMEbus TECHNICAL AND COMMERCIAL SUCCESS INTERNATIONALLY . . .

VITA STRATEGIES

- o NON PROFIT CORPORATION
- o FEW DIRECT EMPLOYEES
- o ACTIVE COMMITTEES
- o MEMBER SUPPORTED

VITA STRUCTURE

- o CORPORATION
  - NON PROFIT STATUS
- o EMPLOYEES
  - EXECUTIVE DIRECTOR
  - EUROPEAN FACILITATOR
  - AMERICAS FACILITATOR
- o COMMITTEES
  - TECHNICAL
  - PROMOTION
  - USER

CURRENT VITA MEMBERSHIP

- o MANUFACTURING ORIENTED
  - SPONSOR 2 @ $ 100K/YR
  - SENIOR 3 @ $ 25K/YR
  - REGULAR 32 @ $ 2.5K/YR
  - ASSOCIATE 23 @ $ 200/YR
- o USER ORIENTED
  - OVER 200

ASSOCIATE MEMBERS

Graham Moore Canada
Michel Gault France
Aaron Weis Israel
S de Vries Netherlands
Peter Magneli Sweden
Dipl.-Ing. Clarinval Switzerland
Max Hugelshofer Switzerland
Thomas O'Neill USA
James Kopchinski USA
Thomas Boures USA
Donald Sims USA
William Burton USA
David Allen USA
Geoff Stevenson USA
Thomas Leonard USA
Ken Brandt USA
Richard Wiedeman USA
Richard Main USA
Syed Mahmud USA
Jim Geers USA

REGULAR MEMBERS

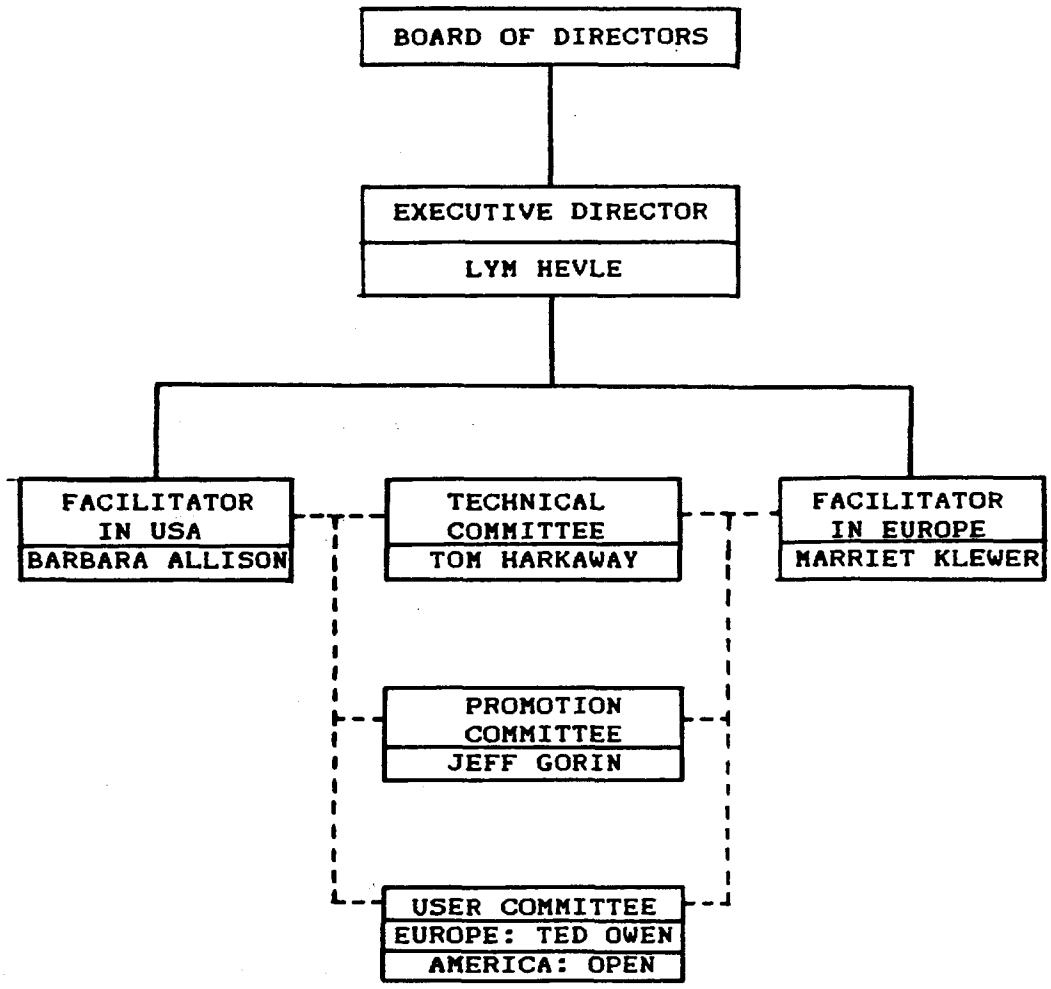
Interactive Circuits & Systems Ltd. Canada
Dy-4 Systems Inc. Canada
VMES Benelux B.V. Holland
Compcontrol B.V. Netherlands
Microproject B.V. Netherlands
Burr-Brown Ltd. Scotland
Fabrimex A.G. Switzerland
CERN, Data Handling Division Switzerland
Convergent Technologies USA
Emulex Corporation USA
HVE Engineering, Inc. USA
Tektronix, Inc. Inst. Sys. Grp. USA
Mini Computer Technology USA
Central Data Corporation USA
Interphase Corporation USA
Force Computers, Inc. USA
Ironics Incorporated USA
Mizar Inc. USA
Mupac Corp. USA
National Instruments USA
Scanbe, Division of Zero Corp. USA
Schroff Inc. USA
Sky Computers Inc. USA
DSP Systems Corp. USA
Moor A.G. Switzerland
Plessey UK
Eltec Elektronik GmbH West Germany
Pep Elektronik Systems GmbH West Germany
Stollmann GmbH West Germany

SPONSOR MEMBERS

- MOTOROLA
- PHILIPS/SIGNETICS

SENIOR MEMBERS

- BICC VERO
- HAMILTON/MOSTEK
- XYCOM



TECHNICAL COMMITTEE

- o SPECIFICATIONS
  - VMEbus REV C IEEE & IEC
  - VMXbus REV B IEC
  - VMSbus REV A IEC
- ACTION REQUESTS
  - FROM MANUFACTURER MEMBERS
  - FROM USER MEMBERS
  - FROM STANDARDS COMMITTEES
- o FUTURES

PROMOTION COMMITTEE

- o PRESS RELEASES
- o ADVERTISEMENTS
- o INTERVIEWS
- o PROSPECTUS
- o VMEbusiness (news)
- o VMEbus Systems (mag)
- o PRODUCTS DIRECTORY
- o LOGO/SEAL SLICKS
- o TRADE SHOW/SEMINARS

VMEBUS USER GROUPS

E.C.G.Owen, Daresbury Laboratory
Daresbury U.K.

There is much activity within the European research community on VMEbus systems and the need for some form of collaboration in this area at a European level is thought to be benificial. We have therefore to consider what form this activity should take to obtain maximum benefit for our users.

Two major questions arise, one, the common sharing of our technical developments and two, the best approach to the growing number of worldwide VMEbus manufacturers. In addition, it is also essential that we have some voice in the future standardisation activities which will take place in the VMEbus field.

In the area of sharing technical expertise and development, it is hoped that we can collaborate via the VMEbus working group which has been set up by ESONE (European Standards Organisation for Nuclear Electronics) under the Chairmanship of Chris Eck at CERN. If, however, we are to have strong influence on standardisation and the industrial manufacturers, it is essential that we are associated to the much larger user community of potential and current VMEbus users.

VITA (VMEbus International Trade Association) has declared its intention of forming user groups throughout the world for the purpose of education, liaison, association with the manufacturers and the promotion of the VMEbus standards, and it is to this organisation we must affiliate to make our views known.

VITA users are already operational in the USA and it is in our own interest to ensure that an active user group organisation is created in Europe.

To form user groups in Europe does, however, bring its own problems, we have our own national factors to consider, we do not have a common language and therefore many documents will need to printed in more than one language, in some cases there are regulations about finance which would make difficult a single European committee consisting of representatives from each country.

Under the proposed VITA structure it is also seen that regional and local VITA groups will be formed if the membership numbers make this valid. The VITA structure also allows for special interest groups to be represented at European level if their size and/or influence justify this, and it at this level we expect our community to be represented via the ESONE working group.

To start the formation of VMEbus user groups in Europe, I am trying to find volunteers to act as the initial national chairmen and for some countries we already have a nominee who is prepared to take on this task. It is also obvious that we will need the help and support of the VMEbus manufacturers in the formation of these national groups and this they seem willing to provide.

It will, of course, take some time to get User Groups into action, as most of the people involved will, like me, have full time jobs and therefore only be able to attend to VMEbus user matters in their own spare time and we will require the help of all interested people during the initial stages.

Some documentation is in preparation and we will contact all those who have already expressed an interest during the conference. If, however, you wish to register your interest then please refer to the contact list below.

Most of us are convinced of the need to have a strong user association and VITA seems to be the best way of achieving these aims, I would ask you all to help make VITA USERS EUROPE an active and expanding organisation.

Ted Owen,
SERC,
Daresbury Laboratory,
Warrington WA4 4AD,
United Kingdom.

Telephone 925 65000 Telex 629609

For registration on a VITA mailing list and general VITA enquiries:

Mariette Klewer,
VITA,
Begijnstraat 10,
5303 XW Veldhoven,
The Netherlands.

Telephone 31 40 544372

For information on the ESONE VMEbus Group:

Chris Eck,
DD Diviaion,
CERN,
CH 1211 Geneva 23,
Switzerland.

Telephone 22 834260 Telex 419000 CER CH

VMEBUS STANDARDS

A. Schellekens

Philips International
Department of External Standardisation
5600 MD Eindhoven
The Netherlands

Ladies and Gentlemen,

In his presentation Mr. L. Hevle this morning has introduced to you the VMEbus International Trade Association. It is an independent, non-profit, organisation, whose object it is to promote the VMEbus technically as well as commercially.

The VMEbus Specification Manual, issued by VITA, is one source of documentation of the VMEbus.
Another source of documentation of the VMEbus is the forthcoming international standard, to be issued by the International Electrotechnical Commission.
An there are other sources of VMEbus documentation as well.

The question then arises:
How long will it take till homologation of the documents of the various sources is achieved, if ever?

Before answering that question, I would like to address a subject that is closely related to that question, namely that of the status or if you like that of the acceptance of the documents.

Each source of documents for the VMEbus has its own figure of merit.

For the purpose of this presentation I would like to make a distinction between two sources of standards:

At one side an Industry Standard or a de-facto standard.At the other side an IEC standard, that is a standard issued by the International Electrotechnical Commission.

The creation of an Industry Standard is characterized by the following 3 items:

a) The draft of the standard is under the control of its orginators during its preparation.

b) A restricted group of industries or individuals is involved in the preparation.

c) When the standard is issued finally, considerable marketing efforts are required to promote the standard and to gain its acceptance by third parties.

There are many examples of successful Industry standards.
A rather recent example is the standard for the Compact Disc player, where initially two companies were involved in the preparation of the standard.

But how is the situation when the IEC is in the process of creating a standard.
It is rather the opposite of the process involved by an Industry Standard.

The creation of an IEC standard is characterized by the following 3 items:

a) The draft of the standard is under control of the IEC during the preparation. The originator has no more control over it.

b) A large number of industries, interested parties or individuals is involved in its preparation.

c) Acceptance of the standard is by voting. Voting rights have the members of the IEC.

Let me explain this point a little more in detail.

Overhead sheet 1

Members of the IEC are the National Committees of 42 countries. They collectively represent 80% of the world population that produces and consumes 95% of the electric energy.
Individuals are not member of the IEC.
Their interests are represented by the National Committee of their respective countries.

The National Committees represent the interests of the various parties in their country such as :
- manufacturers
- users
- trade associations
- government
- academic and engineering professions

Not shown on the overhead sheet , because it only applies to some countries , are the standards organisations of that particular country.

It is the prerogative of the National Committe for the IEC to organize the standardization efforts in its country.

In conclusion it can be said that consensus about the proposed IEC standard is achieved in two stages:

- first at National level between the various parties in the country concerned

- secondly at International level, between the National Committees for the IEC

The second stage is governed by the Rules of the IEC.
These Rules provide for dissolving conflicts which may arise because of national interests.
In a democratic institution like the IEC these Rules give a fair opportunity to each National Committee to express its views ; the domination by one or more National Committees is prevented.

It goes without saying that the process of achieving consensus is a slow one. This in spite of all efforts to avoid unnecessary delays. It is a sequential process and the National Committees need to be given sufficient time to study the proposed standard. On the other hand, the time required to achieve consensus about an Industry Standard by marketing efforts should not be underestimated.

In conclusion it can be said that consensus about the standard is gradually building up in case of an Industry standard.
It is already achieved when the IEC standard is issued.

That brings us back again to the question of the homologation of the documentation of the VMEbus from the various sources.

I have noted with pleasure that VITA participates actively in the standardization efforts of the IEC. As a trade organisation they are represented in some of the National Committees and as such participate in the preparation of the IEC Standard.

On the other hand, and this is a quite natural behaviour of a commercially oriented organisation, comments on the standard, either at National level or at international level in the IEC, are picked-up and entered in the VMEbus Specification Manual.

The front sheet of the VMEbus Specification Manual, Revision C, contains a notice that the document includes recommended changes from the IEC/Sub-committee 47B and the IEEE P1014 standardscommittee. It refects the intention to come to a single standard for the VMEbus.The VMEbus Specification Manual may be considered as a preview of the forthcoming IEC standard.

The IEC standard for the VMEbus, known as IEC Publication 821, is in the process of printing.

The report of the voting on the draft standard showed support for the standard by 21 countries.

Overhead sheet 2

Against approval were 2 countries.
The rules of the IEC require that the draft is approved by a majority of the votes in favour. Majority means that more than 80% of all votes submitted should be in favour. Abstentions are not counted.

The major countries active in the field of microprocessor systems were in favour of the publication of the draft as a standard.

The technical comments of the British National Committee which led to the negative vote could be met and have been introduced in the standard by the Editing Committee.

The Editing Committee has completed its task and the standard is in process of printing.
It is expected that the standard will be available in early spring 1986.
It will be known as IEC Publication 821: Microprocessor system bus II, 8-bit, 16-bit and 32-bit data.

Now that the standard is reality, the next question is: How will the standard be kept up-to-date.

It is evident that new material should be added and that shortcomings should be amended.

However this should be carefully balanced against the interest of those parties who have already implemented the bus and whose interests are effected by that action.

The IEC Rules of Procedure provide for that balance.

All new material proposed to the IEC is treated in the same way and there is no distinction between a proposal for a new standard or a proposal for an amendment of an existing standard.

Only if consensus is reached, as expressed in a majority vote in favour of the proposal, an existing standard will be amended or a supplement to an existing standard will be published.

And with this theme we are already leaving the business of to-day and entering the future.

What will be the probable direction of standardisation effords?

I would like to split this part of the presentation into two.

The first part will deal with new activities which have already been approved as new activities and which have already started.

The second part will be more prospective. I will present some areas of new activities which I think will be interesting for you.

These are mere ideas. To become new activities in the IEC, they have to be proposed by a National Committee.
And they have to be supported by other National Committees to gain approval as New Work.

But first the current activities.

Two buses are proposed as an addition to the VMEbus.
These are known as the VMXbus and the VMSbus, a Sub-system bus and a Serial bus respectively.

The proposals for these buses have been referred to so called Special Working Groups.

Special Working Groups are composed of experts in a particular field.

It is the task of a Special Working Group to sort out a particular problem and to prepare a first draft of a standard.
The Special Working Group on the VMXbus has met in the beginning of September in California.
At this moment the results of the meeting are not yet known.

The Special Working Group on the VMSbus met in the second half of September in Paris.
Agreement was reached on all open questions related to the Serial bus.
A proposal for a standard will be submitted to the National Committees for comments in the beginning of 1986.

The proposal will contain the protocol for the serial bus and further details regarding the Backplane Serial Bus.
That is the serial bus that make use of the two tracks of the backplane of the VMEbus that are reserved for that purpose.
(P1 connector b21 and b22).

The proposal will furthermore contain the details regarding the Interbackplane Serial Bus.
This Interbackplane bus is an extension of the Backplane Serial Bus and is intended to connect to other backplanes. The Interbackplane Serial Bus is buffered and uses two pairs of twisted wires as a medium. The proposal provides for a total length of 40 m for the Interbackplane Serial Bus.

Now that we have seen the current activities in the field of standardization of buses, lets enter the future.

Sub-committee 47B of the IEC, the committee responsible for standardization in field of microprocessor systems, has identified at its last meeting the following areas of work.

Overhead sheet 3

The areas which are shaded blue are those areas in which Sub-committee 47B is currently active.

Apart from buses, the activities concern:
- Terms
- Definitions
- Symbols and
- Microprocessor Programming

For the other areas, including the red areas, there is no activity at present.
National Committees are invited to submit proposals to start the work.

The red areas are a special case.
They are related to the subject of compatibility of board level products.

Compatibility means that boards of different manufacture will operate together on the same bus.

It also means that a board of one manufacturer may be exchanged for an identical board of another manufacturer.

It goes without saying that compatible products shall comply with the requirements listed in the VMEbus standard.

Compliance with the requirements has to be testable.
Results become comparable when standard test and measuring methods are employed.

There are already several test and measuring methods available as IEC standards. These methods cover devices ranging from simple diodes to fairly complex integrated circuits.

Sub-committee 47B recognized that board level products require additional, specialized test and measuring methods.

Standard test and measuring methods are a first step towards compatibility.

The second and final step towards compatibility could be the application of a Quality Assessment System for board level products.

Overhead sheet 4

A typical Quality Assessment System would assess:

- Electrical characteristics, including operational/functional characteristics.

- Environmental characteristics, including endurance (Immersion in cleaning solvents, Robustness of terminations, Shock, Vibration, Acceleration, Damp heat, Dry heat, Storage at extremes of temperature, Transient Energy).

- Mechanical characteristics. Dimensions, visual examination.

Such a system would thus not only assess the compatibility of the board level products, it would also assess their reliability.

And the reliability date from current production at the manufacturer would be available to the users.

For such a Quality Assessment System the IEC may be a good choise.

The IEC operates a Quality Assessment System known as:
IEC Quality Assessment System for electronic components.

Today, components covered by this quality assessment system range from resistors and capacitors to discrete semiconductors. Integrated circuits are about to be added to the range of components. The required documentation for these devices will become available in the near future.
This documentation will include complex integrated circuits such as microprocessors.

The question then arises: If such complex devices as microprocessors can be assessed under the IEC Quality Assessment System, would it be also applicable to board level products?

The arrangements required for microprocessors look similar to the requirements found in the VMEbus standard.
There is of course the additional problem of the characteristics of the backplane, but would this problem be unsurmountable?

On the other hand the advantages of the use of the IEC Quality Assessment System are quite impressive:

Overhead sheet 5

- Characteristics of the board level products are easily comparable.
  They are tested using standard test and measurement methods.
  They are tested against known limits.

- Essential characteristics to be tested are agreed internationally and are mandatory.

Additional characteristics may be specified.

- Verification of the function over the full operating temperature range.

- Reliability of the product is assessed by environmental and endurance testing.

- Modular approach of VMEbus standard allows for large variaty of board level products, to be assessed on the basis of the modules.
  That is the required documentation can be simple and quite straight forward.
  (Masters,Slaves,Location monitor,Bus timer,Interruptor, Interrupt handler,Requester,Arbiter,Clock driver).

The approval of a board level product is repognized by all countries participating in the IEC Quality Assessment System.

The IEC Quality Assesssment System is a truely world-wide certification system.

There are 22 National Authorized Institutions involved in the certification of components.

Overhead sheet 6

These National Authorized Institutions are authorized by the responsable national organisations such as government, trade association, standards organisations.

It shows that the claim of the IEC to operate a world-wide certification system is well founded.

For those who are interested in the IEC Quality Assessment System for electronic components there is a glossy paper brochure available.

The reason why I introduced the IEC Quality Assessment System to you as a candidate for a Quality Assessment System for board level products is that at this moment Technical Committee 47B is polling the National Committees about their interests in quality assessment for microprocessor systems.

Apart from showing interest,the National Committees are also requested to submit proposals for the documents required for the quality assessment for microprocessor systems.

Another reason is that I am very interested in your reaction. The Netherlands National Committee has also to answer the request of Sub-Committee 47B regarding quality assessment of microprocessor systems.

In fact, much of the material of this presentation is derived from the discussions we have had so far in the Netherlands National Committee.

The Netherlands National Committee has not yet taken a decision.

They realized that the problems associated with test and measuring methods and with the assessment of quality are indeed difficult and complex problems.

Answering the request of Sub-Committee in the affirmative would also contain the obligation to submit proposals for test and measuring methods in the first instance and later on the proposals for the quality assessment.

This work would put a heavy claim on the available resources of the Netherland National Committee and that for a considerable time.

This commitment can only be justified if there is a real interest in quality assessment for microprocessor systems.

And that interest should not be restricted to the Netherlands, it should, if possible, be world wide.

In search for reactions, in search for interest, I have confronted you, Ladies and Gentlemen , with the problem!

To assess or not to assess , that is the question!

Sheet 1

INTERNATIONAL ELECTROTECHNICAL COMMISSION

- RESPONSIBLE FOR INTERNATIONAL STANDARDIZATION IN THE ELECTRICAL AND ELECTRONICS FIELDS
- COMPOSED OF 42 NATIONAL COMMITTEES
- NATIONAL COMMITTEES COLLECTIVELY REPRESENT 80% OF WORLD'S POPULATION THAT PRODUCES 95% OF ELECTRIC ENERGY

NATIONAL COMMITTEES ARE EXPECTED TO BE FULLY REPRESENTATIVE OF ALL ELECTROTECHNICAL INTEREST IN THEIR RESPECTIVE COUNTRIES

INTERESTS INCLUDE
- MANUFACTURERS
- USERS
- TRADE ASSOCIATIONS
- GOVERNMENT
- ACADEMIC AND ENGINEERING PROFESSIONS

Sheet 2

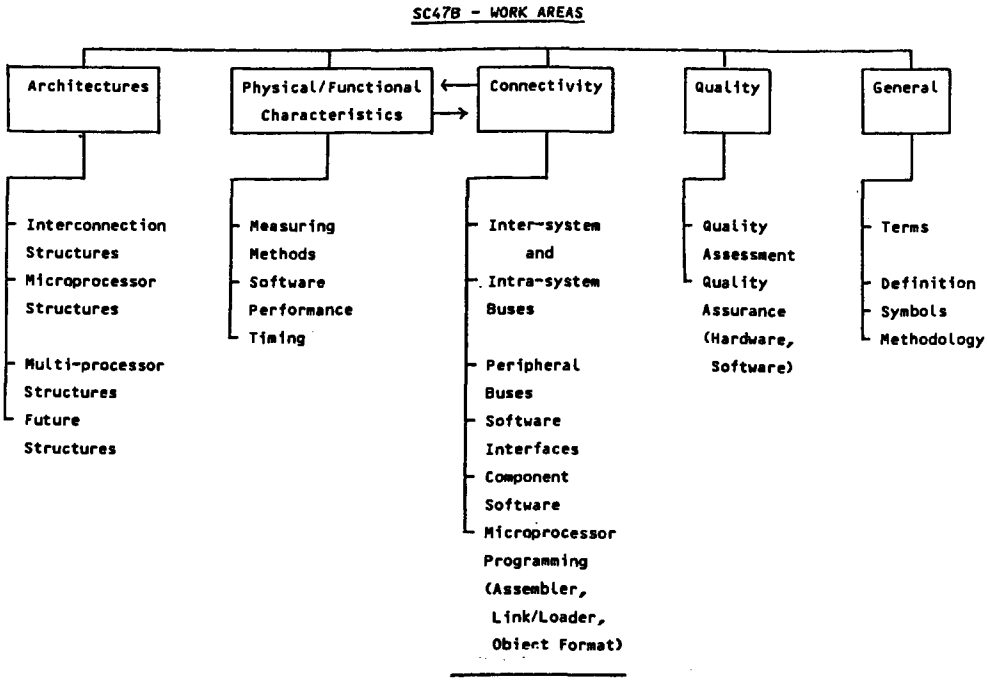
SUPPORT FOR THE IEC STANDARD ON VME BUS

FOR APPROVAL: 21 COUNTRIES

- AUSTRIA
- BELGIUM
- BULGARIA
- CHINA
- CZECHOSLOVAKIA
- EGYPT
- FRANCE
- GERMANY
- HUNGARY
- ITALY
- JAPAN
- NETHERLANDS
- NORWAY
- POLAND
- ROMANIA
- SOUTH AFRICA
- SPAIN
- SWITSERLAND
- U.S.A.
- U.S.S.R.
- YUGOSLAVIA

AGAINST APPROVAL: 2 COUNTRIES
- SWEDEN
- UNITED KINGDOM

Sheet 3



Reproduced in Switzerland
Central Office of the IEC
3, rue de Varembé - GENEVA
47B(Secretariat)36

Sheet 4

QUALITY ASSESSMENT SYSTEM

★ **ELECTRICAL CHARACTERISTICS**
OPERATIONAL / FUNCTIONAL CHARACTERISTICS

★ **ENVIRONMENTAL CHARACTERISTICS/ENDURANCE**

★ **MECHANICAL CHARACTERISTICS**

Sheet 5

IEC QUALITY ASSESSMENT SYSTEM

★ CHARACTERISTICS OF PRODUCTS ARE EASILY COMPARABLE

★ ESSENTIAL CHARACTERISTICS ARE MANDATORY ADDITIONAL CHARACTERISTICS MAY BE ADDED

★ VERIFICATION OF THE FUNCTION OVER THE FULL OPERATING TEMPERATURE RANGE

★ RELIABILITY DATA AVAILABLE TO USERS

★ APPROVAL INTERNATIONALLY RECOGNIZED

Sheet 6

IECQ

IEC QUALITY ASSESSMENT SYSTEM FOR ELECTRONIC COMPONENTS

22 NATIONAL AUTHORIZED INSTITUTIONS

- AUSTRALIA
- BELGIUM
- CHINA
- CZECHOSLOVAKIA
- DENMARK
- FRANCE
- GERMANY
- HUNGARY
- INDIA
- IRELAND
- ISRAEL
- ITALY
- JAPAN
- KOREA (REPUBLIC OF)
- NETHERLANDS
- NORWAY
- POLAND
- SWEDEN
- SWITSERLAND
- UNITED KINGDOM
- U.S.A.
- U.S.S.R.

VMSbus FACILITIES FOR MULTIPROCESSING AND FAULT TOLERANCE

Mira PAUKER, PHILIPS CTI, Fontenay-aux-Roses, France

ABSTRACT

This contribution emphasizes VMSbus as a complementary control and data path to the parallel backplane VMEbus.
Multiple paths between boards provide multiprocessors with shared resource allocation and fault tolerant functionality.
Event message latency times are minimized for tightly and loosely coupled multiprocessor systems, using VMSbus intrabackplane and interbackplane links with self arbitration and resynchronization capabilities.
Redundancy in the fault isolation path, an alternative route to reset or disable failed boards, group addressing for broadcast and broadcall operations are basic means for fault localization and recovery.
Data link compound groups such as semaphores, signature-checking semaphores, tokenpassing, multiaddress talker and listener, locking transaction listener and talker are high reliability tools provided by the VMSbus.

1 INTRODUCTION

The VMSbus provides a serial communication path within a closely-coupled computer system and/or among systems in close proximity. It represents a complementary control and data path to the VMEbus parallel data path.
VMSbus uses two signal lines, a common single sourced clock signal and a data line on which the stations place and sample data. The data is logically OR-ed among active transmitters. Every transmitter continuously monitors and tracks frame progress on the link, solving bus contention problems by a straightforward protocol.

This presentation focuses on VMSbus as a response to multiprocessor system requirements and fault tolerant system needs.

2 VMSbus OVERVIEW

2.1 Layered representation

The VMSbus can be represented with a layered organization as in Figure 1 :

- Link Layer modules
  . HEADER SENDER (HS)
  . FRAME MONITOR (FM)
  . HEADER RECEIVER (HR)
  . DATA SENDER (DS)
  . DATA RECEIVER (DR)

They are always organized in groups of functional modules. These groups can be regarded as a higher sublayer within the Link Layer.

- Physical Layer modules (free standing modules) :
  . BUS ACCESS MODULE
  . SERIAL CLOCK MODULE

- Medium :
  . Conductive paths

- Higher layer management controls transactions between **functional groups** of Link Layer modules like the following :
  . A **Controller** group, formed by a HEADER SENDER and a FRAME MONITOR,
  . A **Flag** group, formed by a HEADER RECEIVER and a latch,
  . A **Talker** group, formed by a HEADER RECEIVER and a DATA SENDER,
  . A **Listener** group, formed by a HEADER RECEIVER and a DATA RECEIVER.

The functional groups involved in a transaction are typically situated on different boards, as illustrated in Figure 2.

2.2 Layer interface signals

2.2.1. Physical Layer and Physical Medium Interface

Two signals in the Medium provide communication among Physical Layer modules :
SERCLK - provides clocking information to Bus Access Modules
SERDAT* - bidirectional.

2.2.2 Link Layer and Physical Layer Interface

Six signals represent this interface :
SCLK, RONE, RSTART - from the Physical Layer to the Link Layer
XONE, XSTART, XJAM - from the Link Layer to the Physical Layer

SCLK transmits timing information to the Link Layer. The other signals relate to SERDAT*.

2.2.3 Link Layer module Interface

Five signals describe operation between the Link Layer modules :

SELECT - HEADER SENDER output to the FRAME MONITOR, indicating that HS won the bus arbitration

FRAME IN PROGRESS - FRAME MONITOR output to the HEADER SENDER, indicating a frame is in progress

DSAE - HEADER RECEIVER output to the paired DATA SENDER, indicating DSAE bit state in the Header subframe

S SELECT, R SELECT - HEADER RECEIVER outputs to the paired DATA SENDER, or DATA RECEIVER, upon detection of respective selection code in Header subframe

2.2.4 Higher Layers/Link Layer Interface

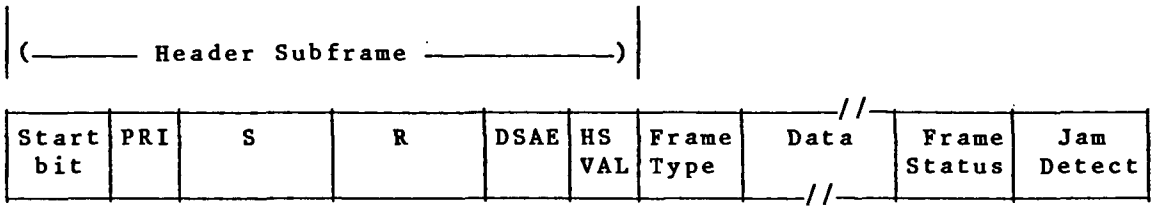
Some of the following signals, used to represent the interface of the Link Layer modules **with higher layers serial bus management**, will be detailed by the explanations concerning high reliability built-in features, presented in 6.3 to 6.6.

S STROBE, R STROBE - sourced by HR
S ENABLE, R ENABLE - sent to HR
SENT, CANCELLED - sourced by FM
RESOURCE FREE - sent to HS
Priority Port, S Code, R Code, Data Frame Port, Status Port
- sourced by FM
Code 1 Port, Code 2 Port, SEND12, SEND21
- sent to HS
LOST ARB - sourced by HS
Code Port - sent to HR
Data Port - sent to DS or DR
Data Size Port - sourced by DR
DSENT - sourced by DS
RESET* - sent to all LL modules

2.3 Frame protocol

The serial communication frame defines a transaction protocol surrounding data or controls with synchronization and error detection bits.

A typical frame representation is :



1	3	10	10	1	1	3	8-256	3	1

Length (bits)

Control frames do not contain the data subframe.
Cancelled frames contain neither data nor status subframes. The Frame Type value is forced to 111 by the entity which cancels a frame.
The Frame Type subframe value 000 characterizes control frames. The values 001 to 110 give the length of the data subframe.
The Frame Status field indicates the status of selected modules. A"1" in the most significant bit indicates an unsuccessful operation due to a data size conflict between DATA SENDERS or RECEIVERS, or incorrect selections during control or data frames. The other two bits indicate the selection of modules by the fields S or R.

Different entities provide the information in the various subframes. It is a **read-write** protocol, checking the active participation of the implied modules.

The Jam detect bit is sampled by the modules involved in the frame transmission to check frame synchronization. If frame synchronization is lost, the FRAME MONITOR drives a sequence of 512 one-bits, which will resynchronize all modules. The presence of a "1" in the Jam detect bit makes the participating modules ignore preceding frame transmission.

3 SUPPORT FOR MULTIPROCESSOR SYSTEMS

The main features of the VMSbus are oriented to support multiprocessor systems.

3.1 Messages between distributed processing elements

VMSbus data frames carry messages :
- in tightly coupled configuration sharing a common memory and a single operating system, as well as
- in loosely coupled configurations with a local memory and local O.S. for each processing element.

3.2 Addressing capabilities

VMSbus generalized addressing is widely useful for :
. broadcast addressing,
. group addressing,
. polling by broadcall operations.

3.3 Resource allocation

VMSbus provides special mechanisms like :
. semaphore implementation,
. token implementation,
which are important for reliable multiprocessing.

4 VMSbus MULTIPROCESSING ORIENTATIONS

4.1 Throughput oriented

The VMSbus autonomy allows :
. maximizing the number of independent jobs done in parallel by general purpose computers.
. balancing the workload between CPU-s and I/O-s.

4.2 Availabiliby oriented

The alternative link offered by the VMSbus is suitable :
. for real time on line applications.
. for failsafe operation (or short downtime).
. for I/O intensive applications.
. to maximize the number of interdependent tasks done in parallel.

4.3 Response oriented

Optimized VMSbus allocation and predictable delivery time are essential :
. for dedicated/embedded applications.
. in specialized applications.
. in CPU intensive applications.
. to maximize the number of cooperating processes done in parallel.

The VMSbus functionality that responds to these needs is summarized by the following.

5 VMSbus SUPPORT FOR FAULT TOLERANCE

The VMSbus frame protocols are oriented for short autonomous communications, efficient to failure reply.

5.1 In fault confinement

VMSbus can be used to limit the spread of fault effects in a system, in the following ways :
. under Operating System management, VMSbus can be used to effectuate consistency checks of range.
. before performing a function, the VMSbus can be used for requesting and confirmation of resource availability.

5.2 In fault detection

VMSbus frames are most suitable :
. to **notify** system elements of a fault occurence,
. to do consistency **checking** concurrently with the useful transfers on the parallel bus,
. to implement **watchdog** timers and time outs.

5.3 **Dynamic** redundancy can be achieved by **switching on** spare components, using VMSbus control frames.

5.4 **Retry** operation

In case of **transmission error** detected by error checking included in the VMSbus protocol, a second operation attempt can be successful (up to 15 are provided by the SCC 68173 controller).

5.5 **Diagnosis** information can be provided about **the location** and its type of a failure, using VMSbus data frames.

5.6 **Reconfiguration**

VMSbus can **isolate** failed components or boards by switching them off, with a possible degradation of performance, or **replace** them by back-up spares.

5.7 **Restart** of the system

Can be achieved in three ways : **hot** - if no information was lost ; **warm** -if some processes can be resumed ; or **cold** if no process is surviving and a complete reload is necessary.

5.8 **On line repair** is possible using procedures equivalent to reconfiguration.

5.9 **Reintegration** of an on-line repaired board can be achieved without interrupting serial bus operation, by using the jam protocol of the VMSbus.

6 VMSbus BUILT-IN HIGH RELIABILITY FEATURES

6.1 Arbitration Mechanism

An arbitration mechanism is built into the procedure for transmitting frames :
- no separate lines are used,
- no dedicated time period is used.

Fair access to the bus can be ensured by the higher-layer logic **arbitration strategy**. The priority value of a HEADER SENDER can be **increased** when the arbitration is lost or can be **decreased** when a frame is cancelled.

6.2 HEADER SENDER VALIDATION

The HSVAL bit guards **against the possibility that all HEADER SENDERS might drop out of the VMSbus arbitration.** If HSVAL is a zero bit, the frame must be ignored.
The protocol allows several HEADER SENDERS to send the same Header subframe together. For applications in which this is not appropriate (e.g., Semaphore Set frames), the higher layers must ensure unique header subframes.

6.3 ENABLE S, ENABLE R validation

When a HEADER RECEIVER matches a selection codes in the S or R field, it will cancel the frame if the corresponding higher layer input ENABLE S or ENABLE R is false. This is useful if a data receiving buffer is not yet ready to accept new data or if some other resource is not ready for the frame operation.

6.4 S, R STROBE validation

The entire frame is checked for successful transmission before the S and/or R STROBE signal validates the frame.

No STROBE will be generate if :
- the most significant bit of the frame status field is one (abnormal operation),;
- the jam bit is true (frame desynchronization).

6.5 DSENT validation

The DATA SENDER output DSENT informs higher layers that it has successfully transmitted data. This allows two strategies for Talker or Listener groups : an **On-Demand** Talker/Listener or a **Transaction** Talker/Listener. An On-Demand Talker is always ready to send the most recent data provided by its higher-layer logic. An O.D. listener is always ready to receive new data and present it to its higher layers. Such "always ready" Talkers and Listeners, require double buffering or other means to avoid mixturing of old and new data.

A Transaction Talker sends data only once when it is provided by its higher layer logic. A Transaction Listener requires that its higher layers must read out received data before it will accept more. For such groups, the ENABLE S or ENABLE R signal controls whether the group is ready or whether it cancels the frame.

6.6 SENT/CANCELLED loop signals

The FRAME MONITOR, coupled with a HEADER SENDER, informs higher layers of the serial bus management of the results of frame transmission. The SENT output indicates successful transmission. The CANCELLED output indicates that a selected group was not ready. The LOST ARB output indicates the frame lost the arbitration to a higher priority one.

6.7 JAM protocol

The Jam procedure provides security against frame-level desynchronization among VMSbus modules.

This procedure is also useful in **extended configurations** where the serial bus modules do not share a common Reset signal. The Jam sequence allows **live insertion** of modules, assuring frame resynchronization with other controllers.

7 VMSbus MODULE GROUPS DEDICATED TO HIGH LEVEL FUNCTIONS

7.1 Simple Flag

The simplest group is formed by a HEADER RECEIVER (with its ENABLE S, R inputs set true) and a latch. Among other applications, the latch output can **enable/disable** the system **bus interface** of a board, **or** provide a signal to **Reset** the logic of a **board. Fault area confinement, reconfiguration** after repair, or backup recovery can be managed using Flag groups.

7.2 Multiaddress Flag

A Flag can be set or reset by means of any of several selection codes. Such a group is described as consisting of several HEADER RECEIVERS whose S STROBE outputs and R STROBE outputs are OR-ed logically. Such a Flag can be used to **dynamically disable** Masters on particular boards or all boards of a certain type.
Multiaddress Flag HEADER RECEIVERS are configured with their ENABLE S and R inputs true.

7.3 Multiaddress Talker

In this group, a DATA SENDER can be selected by means of any of several selection codes. The S-SELECT outputs of several HEADER RECEIVERS are OR-ed to produce the DATA SENDER's S SELECT input.

This group can be useful in system **monitoring** or **diagnosis,** reading data from a unique address or from a set of DATA SENDERS. In the latter case, the result can be the largest value among them (due to Data Arbitration Enable feature), or the logical OR of their data.

7.4 Multiaddress Listener

In this group a DATA RECEIVER can be selected by any of several selection codes, the outputs of several HEADER RECEIVERS are OR-ed to produce an R SELECT input.
In multiprocessor systems, this group allows **event notification** or **message sending** to all processors or to a group of processors.

7.5 **Semaphore** group

A single-bit **mutual exclusion mechanism** is provided by a compound group formed by a controller, a Flag and some additional logic.
The Flag status represents the status of a sharable resource in the multiprocessor system.
The controller provides access to set or clear this Flag, corresponding to the allocation or deallocation of the shared resource.
This group differs from a Simple Flag group in that the set frame is cancelled if the Flag is already set.
It is recommended to implement Flags in parallel with same addresses, one for each controller that can request the resource. These Flags must be in compliance. The serial bus protocols ensure this : a Flag is set by a successful "set frame" and also when a set frame is cancelled.

The semaphore operation with VMSbus includes two selection codes : a semaphore code, assigned to each sharable resource controlled by the Serial bus and a requester code, assigned to each group that can request control of any resource. The different Controllers trying to acquire the same resource MUST use different requester codes, so that there is a unique winner of the VMSbus arbitration. The resource and requester codes are typically assigned at configuration time.

7.6 Signature-checking semaphore

For high security operations, an additional requirement is applied to this type of semaphore group. The requester code is checked by the Flag logic, for the right to share the resource. Additional HEADER RECEIVERS, one for each permitted requester code, have their R STROBE outputs OR-ed. The access rights for each requester code are determined at the configuration time.

7.7 Token Passing Group

This is an alternative way to share common resources. A set of token passing groups form a logical "ring". Each group knows the selection code of its "successor" group, to which it sends frames representing the token.
The resource can be controlled by a group as long as it has received the token and has not yet sent it to its successor.
Two mechanisms for error checking are considered :
- one RING ERROR logic included in the controller surveys the cancelled frames, on an attempt to set a Token Flag already set or to reset a Token Flag already reset. This ensures that only one token frame is passing through the logical ring ;
- a recommendation is done to maintain a timer controlling the maximum time the resource is used per group.

7.8 Locking transaction Listener/Talker

Two other compound module groups are aimed to allocate a resource to a process, instead of semaphore usage.
The Locking Transaction Listener, similar to a Multiaddressed Transaction Listener and a Flag, ensures a Listener receiving several frames of a data unit, in case of a file transmission, from one Talker, before allowing that Listener to be open to any other suitable-configured Talker. So, in case of a printer, a semaphore mechanism can be directly build in the printer serial bus interface.
The Locking Transaction Talker, equivalent to a Multiaddress Transaction Talker and a Flag, ensures the locking of a Talker to a Reading Controller, up to the end of the transmission of successive frames, until it is released by the controller.

These were some examples of VMSbus modules possible operations, in order to satisfy multiprocessor system requirements and high reliability environment.

8 FRAME TRANSMISSION OPTIMIZATION

Several built-in protocol features ensure minimal occupancy of the VMSbus medium by shortening rejected frames and then, avoiding unecessary one's.

8.1 Frame cancellation

Any addressed module, not ready to participate to the VMSbus transaction, has to cancel that frame by sending 1's in the Type field which follows the Header Subframe. In such a case, the frame is shortened to its minimal length of 30 bits :

Start bit	PRI	S	R	DS AE	HS VAL	1 1 1	JAM det
1	3	10	10	1	1	3	1

But this is not a "productive use" of the serial bus and cancelled frames has to be prevent if possible.

8.2 - Handshaking groups

Handshaking Writing Controller and Handshaking Transaction Listener are built to avoid frame cancellation and improve bus efficiency, controlling the transactions by the rate of the Listener data processing.

8.3 Variable Priority Controller

This module group associates a 3-bit counter to the priority port of the HEADER SENDER. Its initial value is set by the serial bus management, when the Header is configured.
When a frame is cancelled, the FRAME MONITOR of that controller clears the priority counter, avoiding to repeat sending. The same clearing mechanism can be used to ensure bus "fairness", after winning the bus allocation. In case the bus allocation is lost, the counter value must be incremented to increase the chances to win. This method ensures productive traffic on the bus.

9 CONCLUSION

I tried to show how VMSbus contributes to multiprocessing offering a concurrent and independent link to interconnect system boards. This link will facilitate scheduling and synchronization of the Processing Elements.
Fault tolerance capabilities introduced by this complementary bus will reinforce VMEbus usage in high reliable applications.
The Silicon support of the basic modules of this VMSbus protocol is opening its spread usage.
The newcoming SCC68173 VMSbus controller and the SCB68171 VMSbus interface chip are first circuits dedicated to this protocol. The operating systems have to incorporate efficient management of this communication link with all its wide opening features.

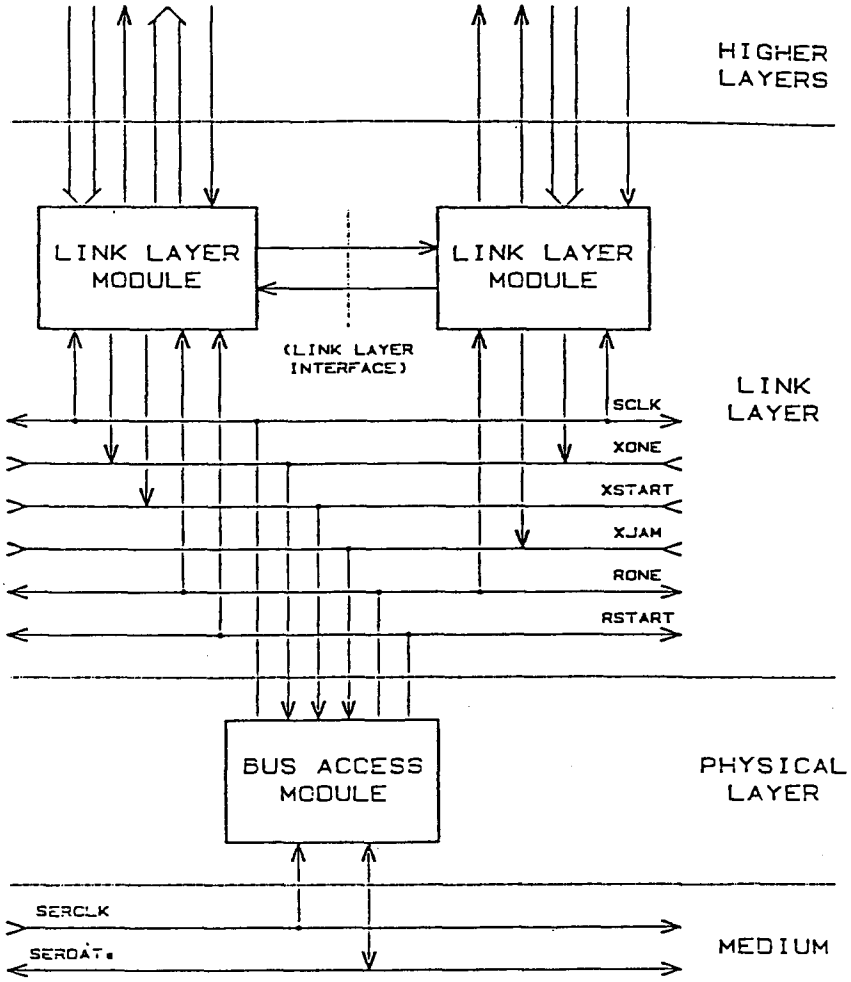


Figure 1 - VMSbus layered organization

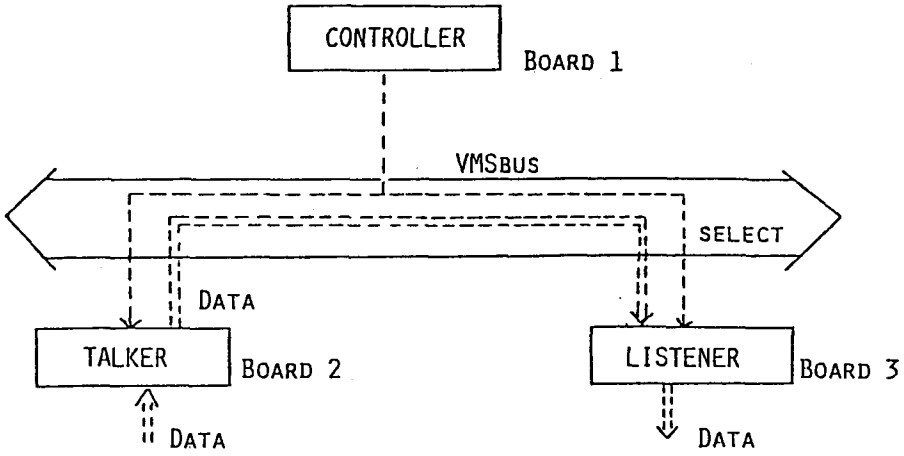


Figure 2 - Groups of modules on different boards

VMEbus INTERFACE CHIP SET

by N. Nissen

Valvo GmbH, Hamburg

INTERRUPT GENERATOR 68154

The 68154 Interrupt Generator provides an interface between an interrupting device and the system bus such as the VMEbus. Its three primary functions are

- Generate system bus interrupts.
- Control daisy chain acknowledge line.
- Provide interrupt vector if needed.

The chip is controlled by a local master via a local databus and appropriate local control signals.

It is a 40 pin bipolar device. Samples should be available in 1985.

For systems, which don't have to have local master control, a solution with programmable logic seems to be more appropriate.

INTERRUPT HANDLER 68155

The 68155 Interrupt Handler can handle seven system bus interrupts plus six local interrupts and a non maskable local interrupt. The output signals IPL 0..2 indicate the interrupt priority level.

For system bus interrupts the 68155 works with the bus control devices 68172/6815 to aquire the interrupt vector from the system.

For local interrupts an interrupt base vector presented by the upper five bits of a byte can be provided. The lower three bits are defined by the interrupt level.

It is a 40 pin bipolar device. Samples should be available in 1985.

For systems which don't require the functional complexity as of the 68155, a solution with programmable logic seems to be more appropriate.

VMEbus CONTROLLER 68 172

The 68172 VMEbus Controller is an interface device for the VMEbus. The device can be implemented for three different configurations:

- master only
- slave only
- master/slave

It can be used either with a processor type interface or with a DMA controller type interface.

An internal synchronization scheme provides for save address decoding.

The controller is designed for dual port operations (shared slave) in order to avoide a "deadlock" situation, which may occur when on a slave access the local master has already started a cycle for the VMEbus.

It is a 28 pin bipolar device. Samples and data sheets are available.

For master only applications the

68175 BUS CONTROLLER

can be used. The controller is an asynchronous design and requires an additional 35 ns delay line.

Here an error / retry sequence is included. It is a 24 pin bipolar device. Samples should be available in 1985.

VMSbus CONTROLLER 68 173

The VMSbus Controller 68173 is a register oriented peripheral device, that includes a collection of the basic functional modules described in the VMSbus specification. It interfaces to the VMSbus via the 68171 VMSbus Interface and to a standard 68000 family bus or other parallel buses.

It comes with

- Header Sender paired with frame monitor with up to fifteen automatic retries on lost arbitration.

- Alternate frame start by single SYNCN input.

- Six Header Receivers with multiplexing capability.

- Programmable single or multiple data transmission/reception with data valid control.

- Four programmable flag modules with two direct outputs for immediate control.

- Interrupt control and upper five bit (of byte) vector base. The lower three bits are for modul identification.

- Four byte data buffer for sender and receiver.

It is a 28 pin CMOS device. Samples should be available early 1986. Product Description (Valvo VDP 8503) is also available.

VMEbus INTERFACE 68171

The VMSbus Interface connects one or more VMSbus controllers to the VMSbus itself. It provides bus driving and receiving in addition to latching data in both the transmit and receive directions.

This guarantees for save operation at maximum speed, taking into account all additional delays on the backplane under worst case conditions.

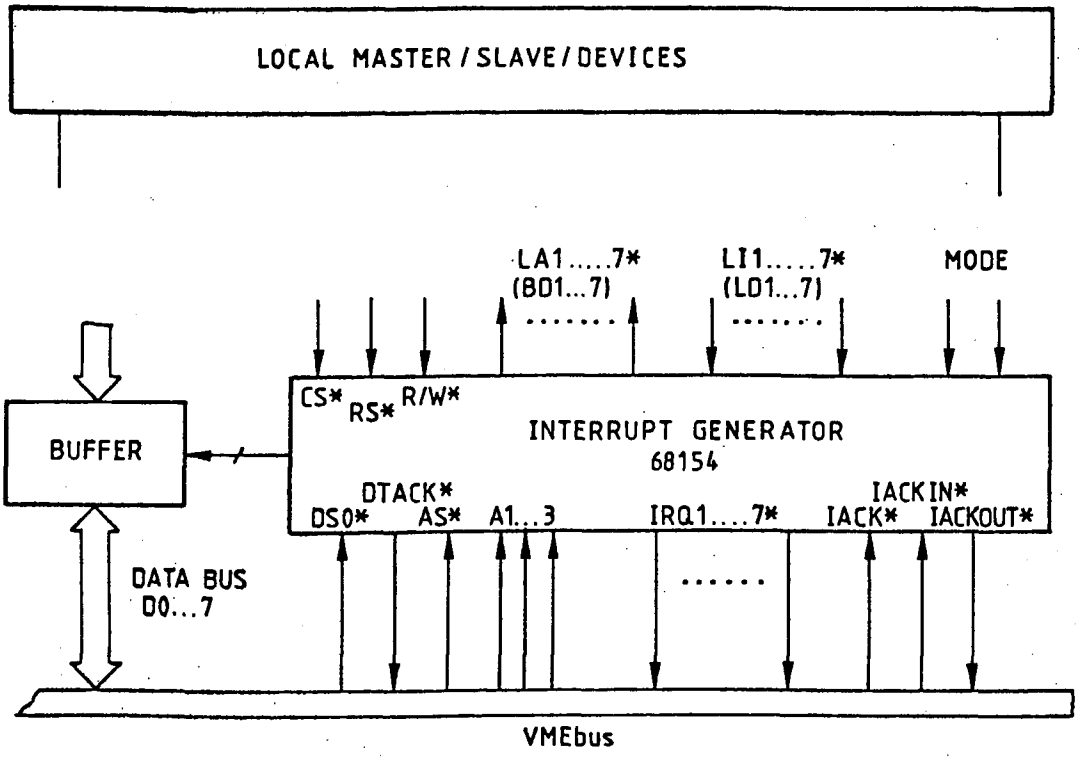
It is a 16 pin bipolar device. Samples should be available early 1986.

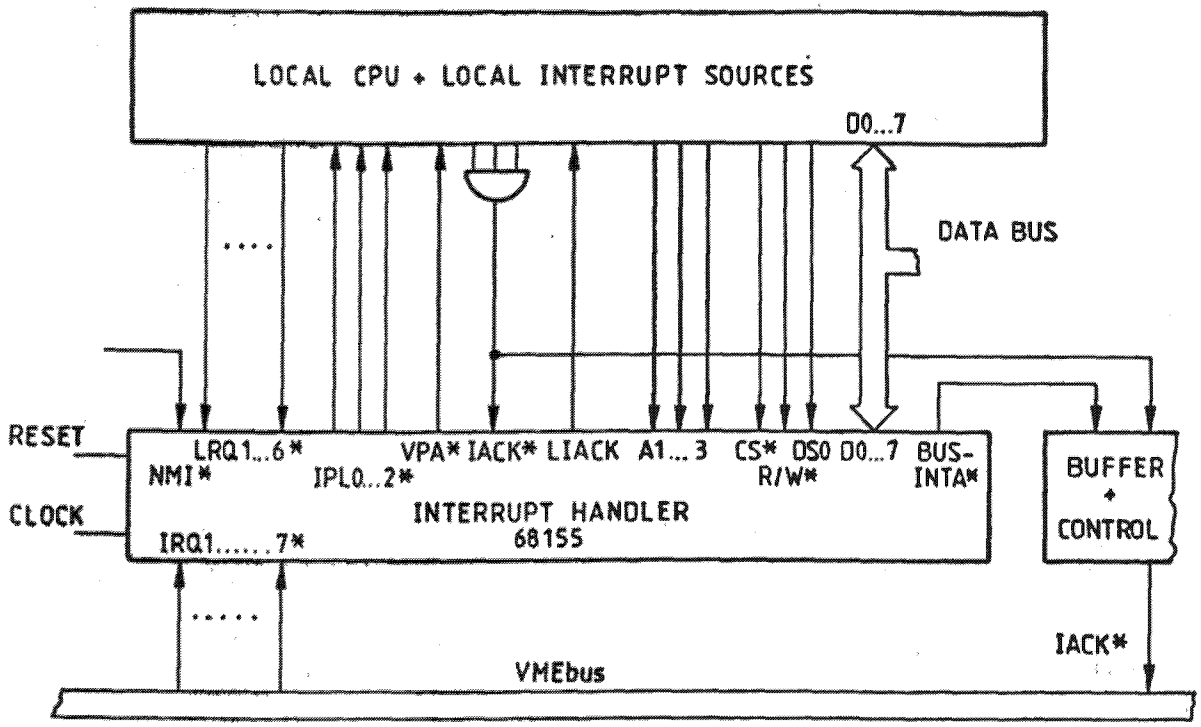
VME- und VMS-Buscontroller

68154 INTERRUPT GENERATOR
68155 INTERRUPT HANDLER

68175 VMEbus CONTROLLER (MASTER ONLY)
68172 VMEbus CONTROLLER (BUSCON M/S)

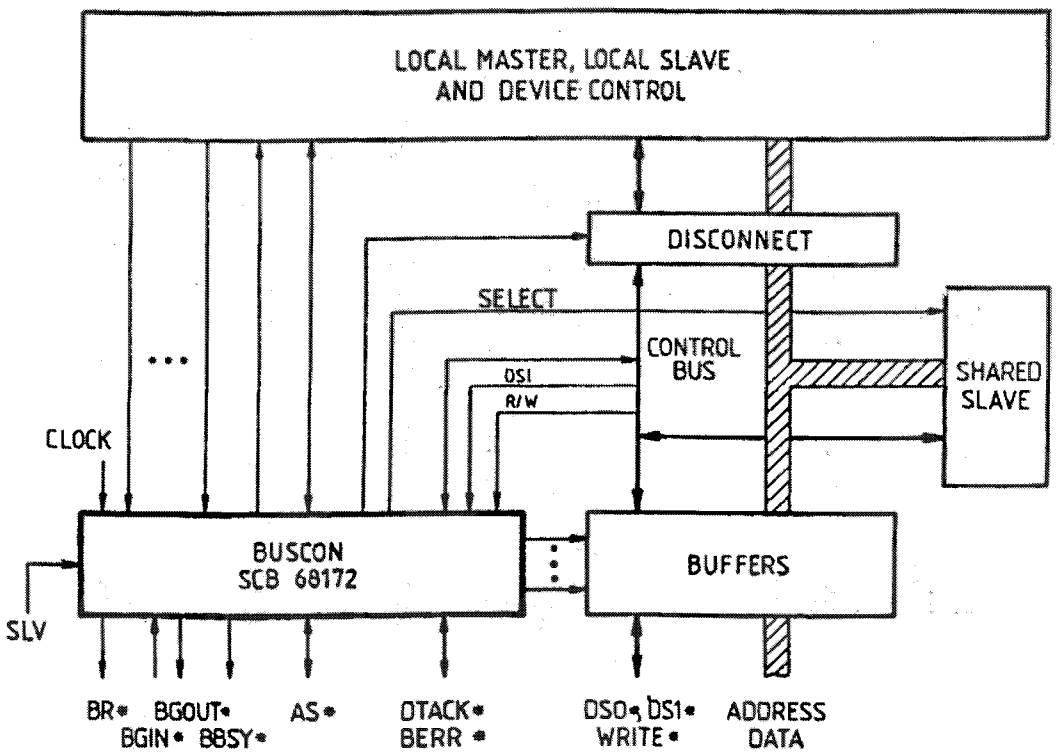
68171 VMSbus INTERFACE
68173 VMSbus CONTROLLER



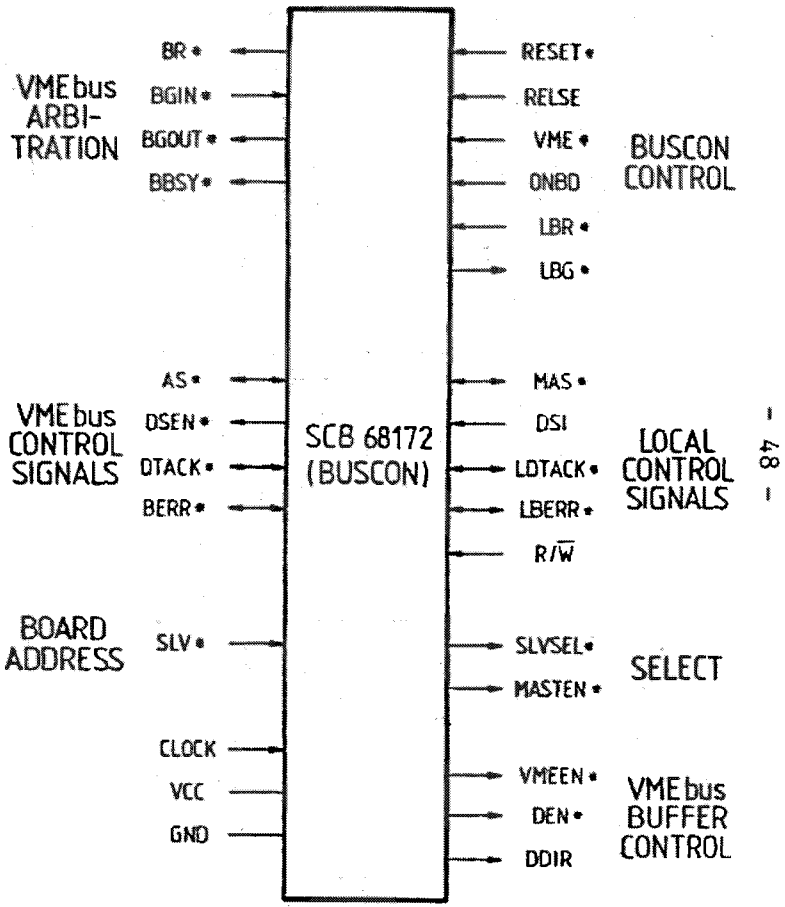


- PROVIDES VMEbus ARBITRATION CONTROL
- CONTAINS STROBE DELAY CONTROL
- INITIATES VMEbus RELEASE
- HAS VMEbus CONTROL SIGNAL BUFFERS FOR AS*, DTACK*, BERR*, (DS0*, DS1*)
- BIDIRECTIONAL FOR MASTER/SLAVE APPLICATIONS
- BIPOLAR CCL DESIGN

BUSCON FEATURES



SYSTEM BLOCKDIAGRAM



BUSCON SIGNALS

BUS CONTROLLER 68175

DIFFERS FROM BUSCON (68172) AS FOLLOWS:

- MASTER ONLY
- ASYNCHRONIOUS DESIGN
- DELAY LINE
- REQUIRES FAST DECODING
- ERROR RETRY SEQUENCE INCLUDED
- NEEDS ADDITIONAL ADDRESS-STROBE BIDIR. BUFFER
- 24 PIN SLIMLINE PACKAGE

VMSbus CHARACTERISTICS

- SMALL AREA NETWORK (SAN)
- SIGNAL TRAVEL TIME << BIT TIME
- NO COLLISIONS
- SIMULTANEOUS MULTISTATION ACTIVITY
- NO LINK ALLOCATION TIME REQUIRED
- SELF ARBITRATION
- ERROR DETECTION
- COMMON CLOCK
- UNIQUE STARTBIT

VMSbus APPLICATIONS

TRANSMITT MESSAGES (TALKER/LISTENER)

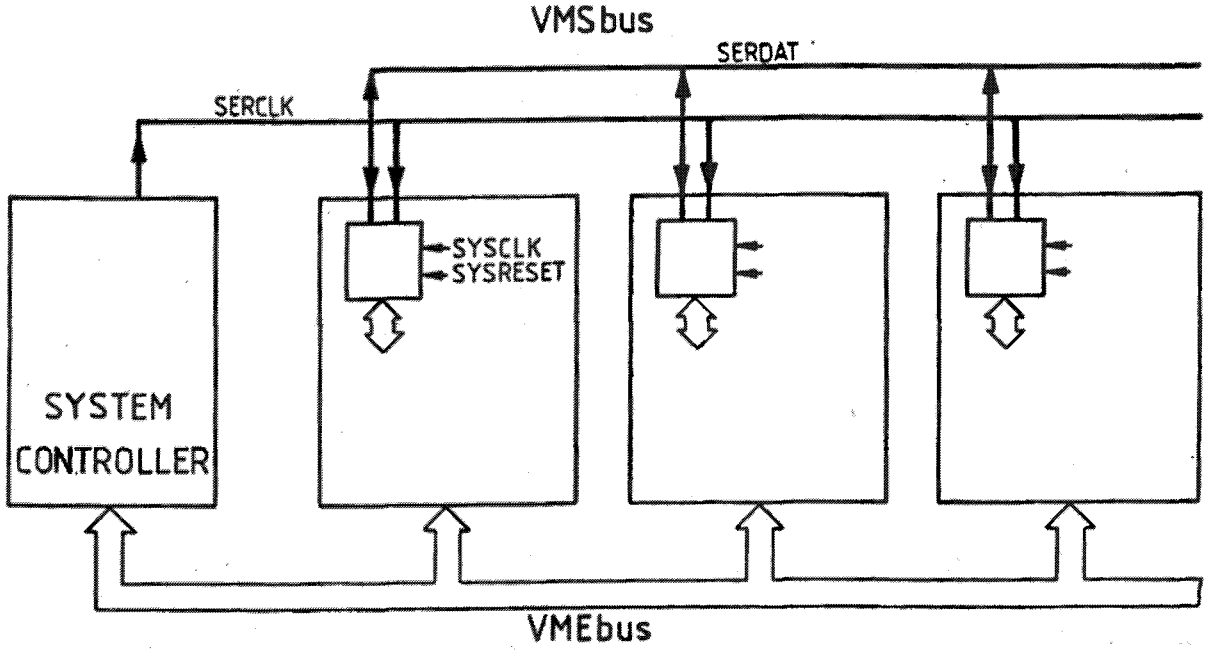
SIGNAL EVENTS, STATUS (TRACKER)

DISCONNECT

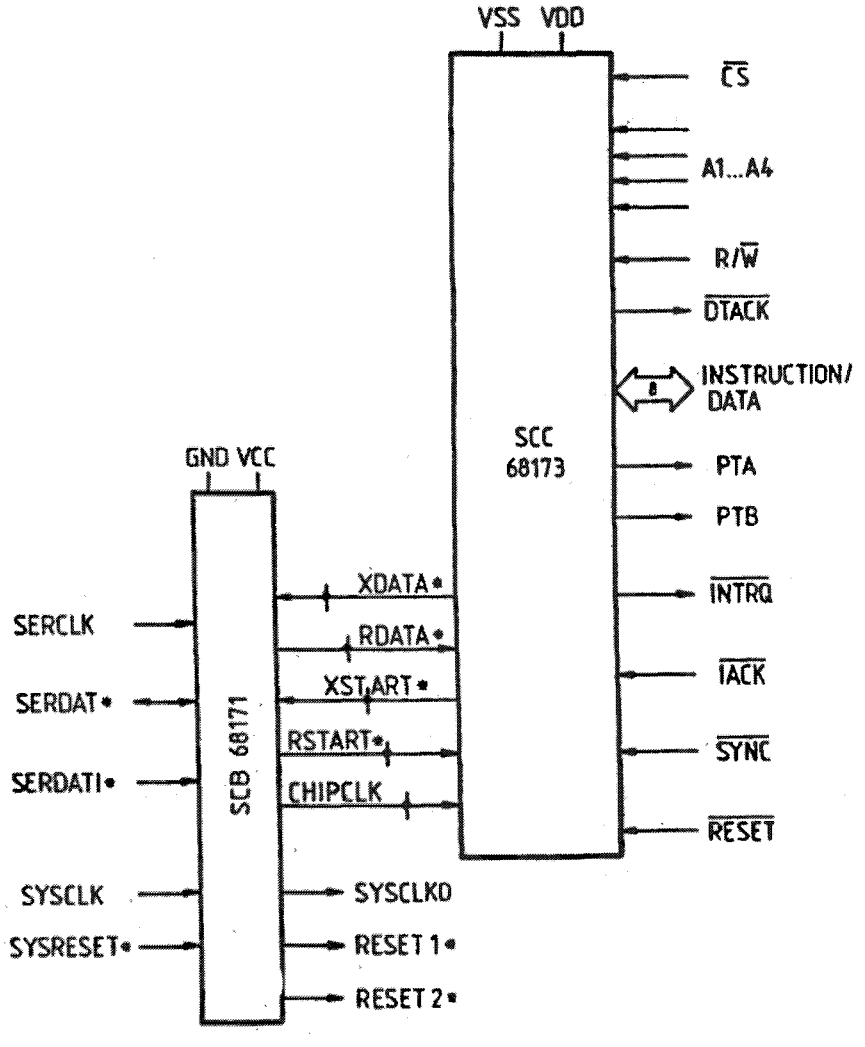
VIRTUAL SIGNAL LINES

FAULT TOLERANCE IMPROVEMENT

LOW COST ADDITIONAL LINK



VMSbus SYSTEM



VMSbus INTERFACE

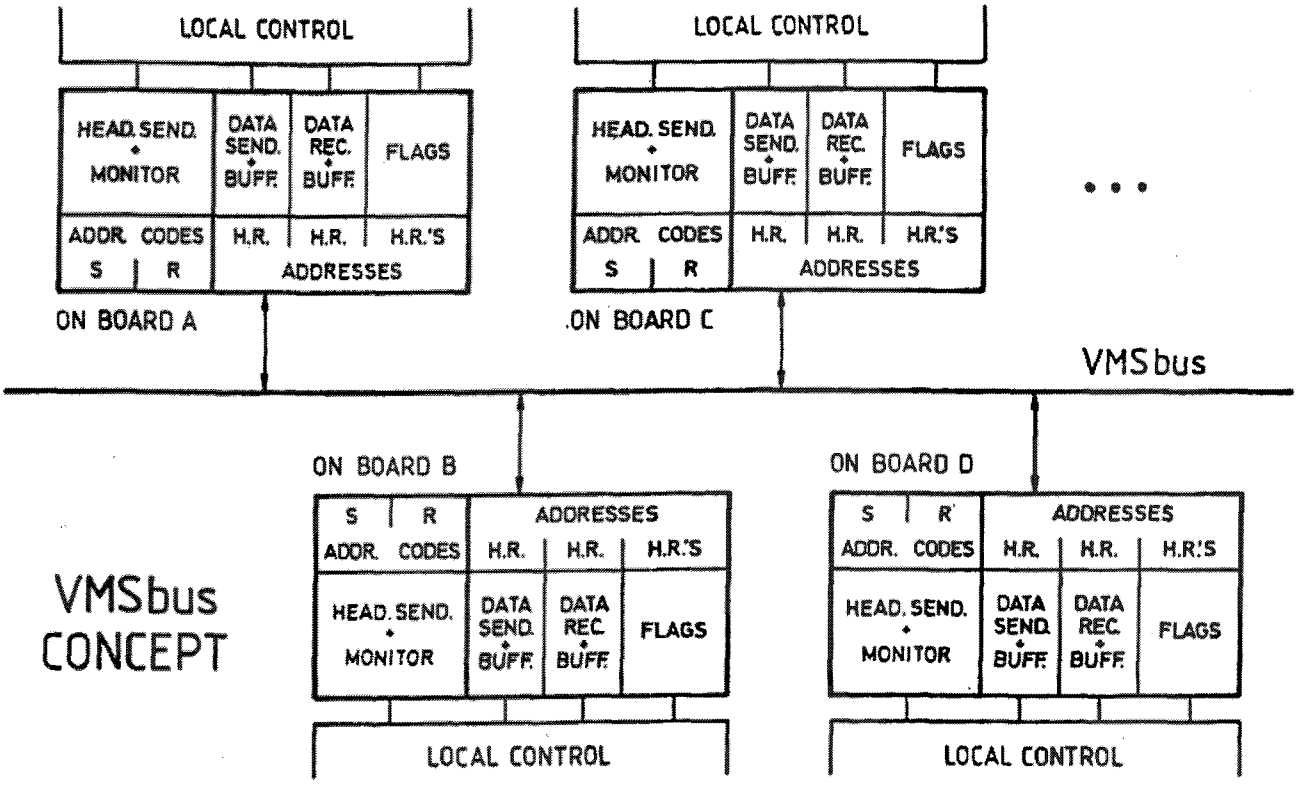
VMSbus MODULE GROUPS

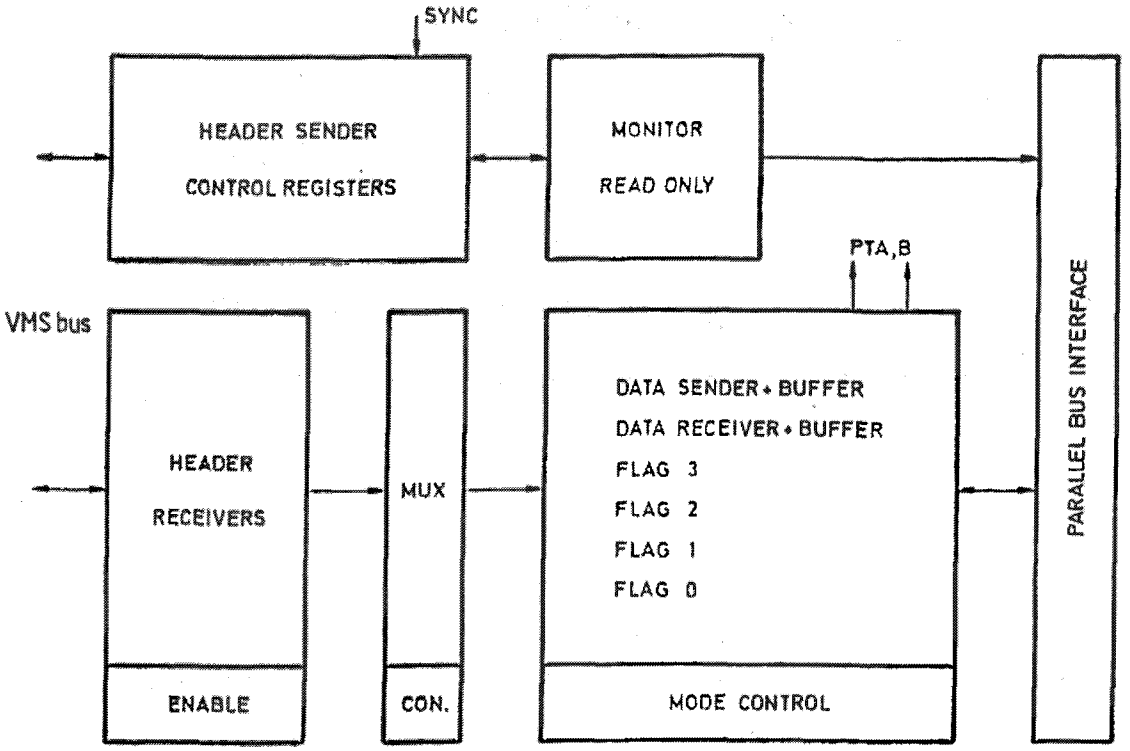
CONTROLLER : HEADER SENDER + FRAME MONITOR

FLAG : HEADER RECEIVER + LATCH

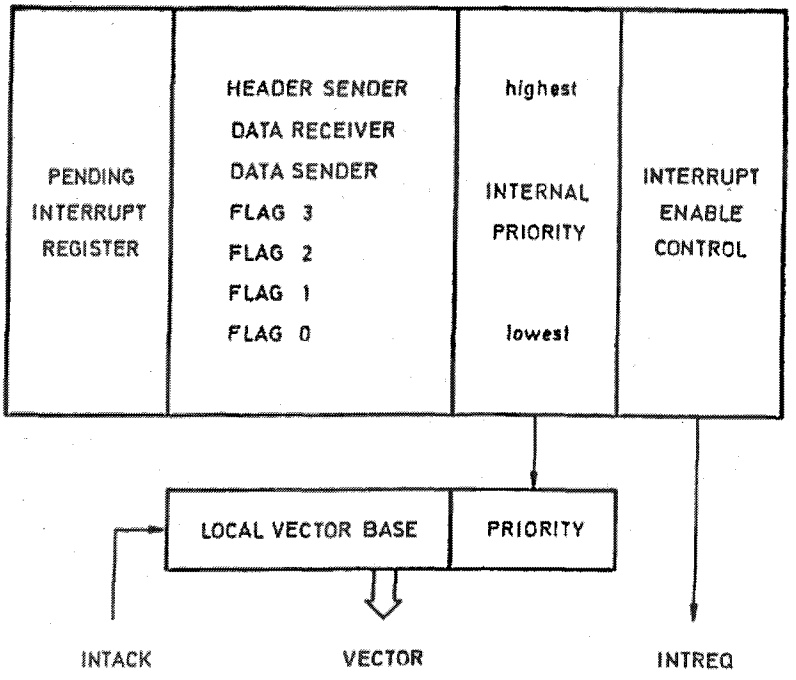
TALKER : HEADER RECEIVER + DATA SENDER

LISTENER : HEADER RECEIVER + DATA RECEIVER

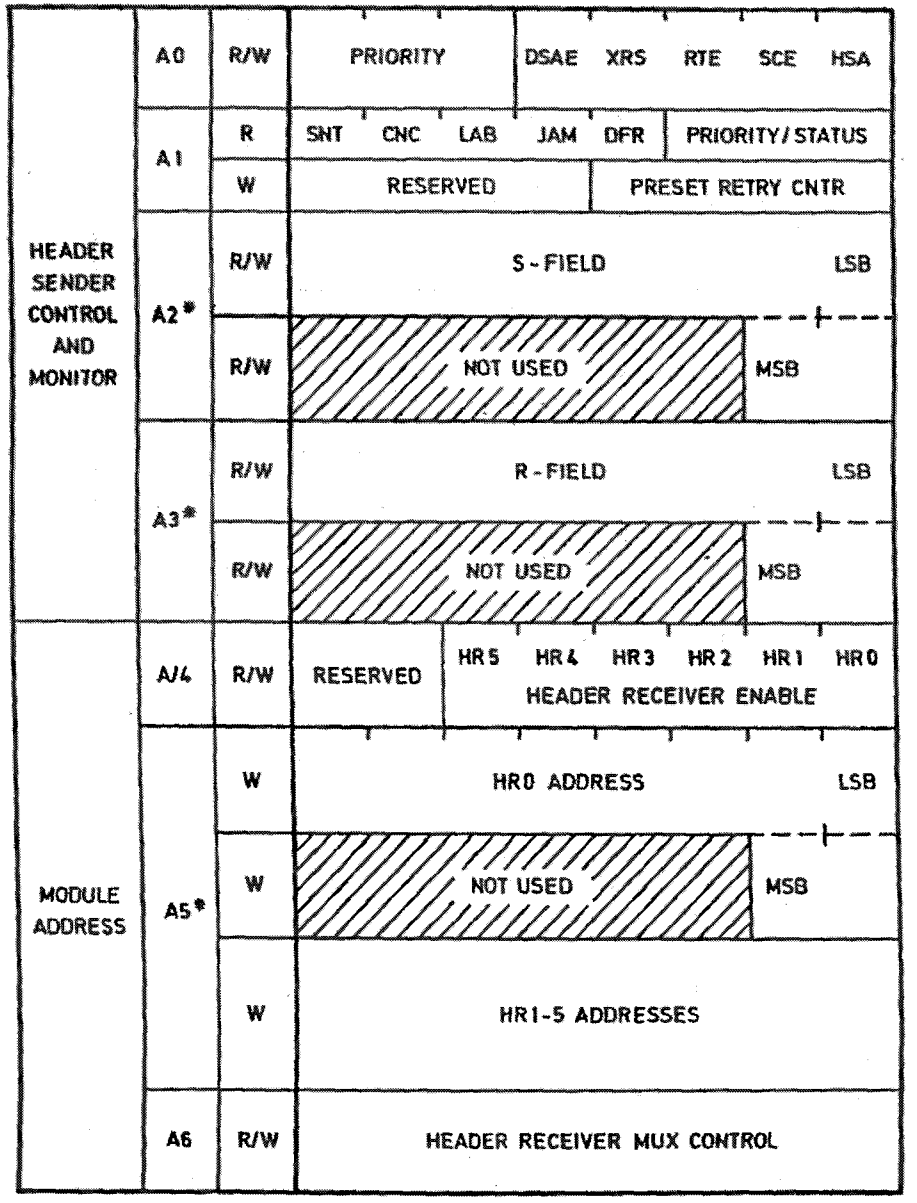




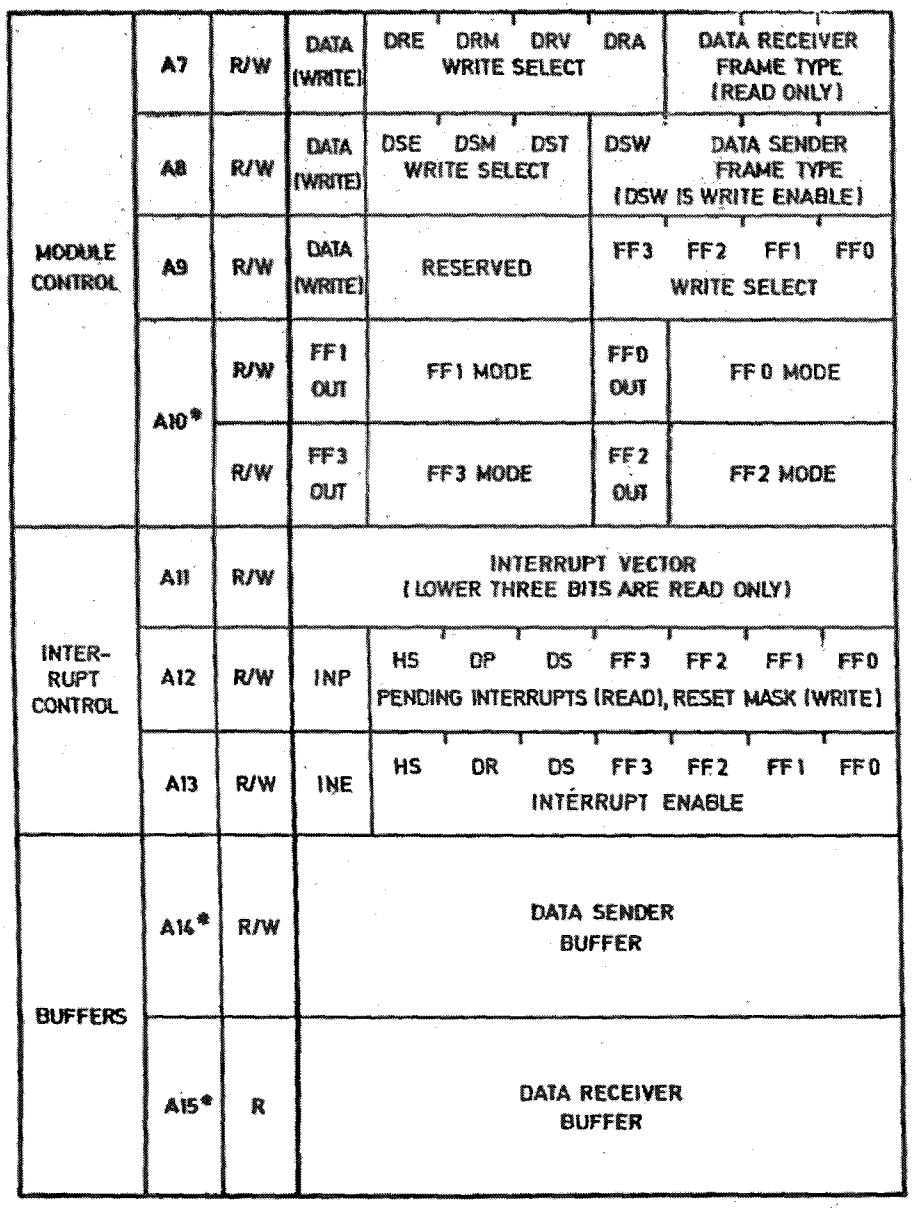
FUNCTIONAL BLOCKDIAGRAM



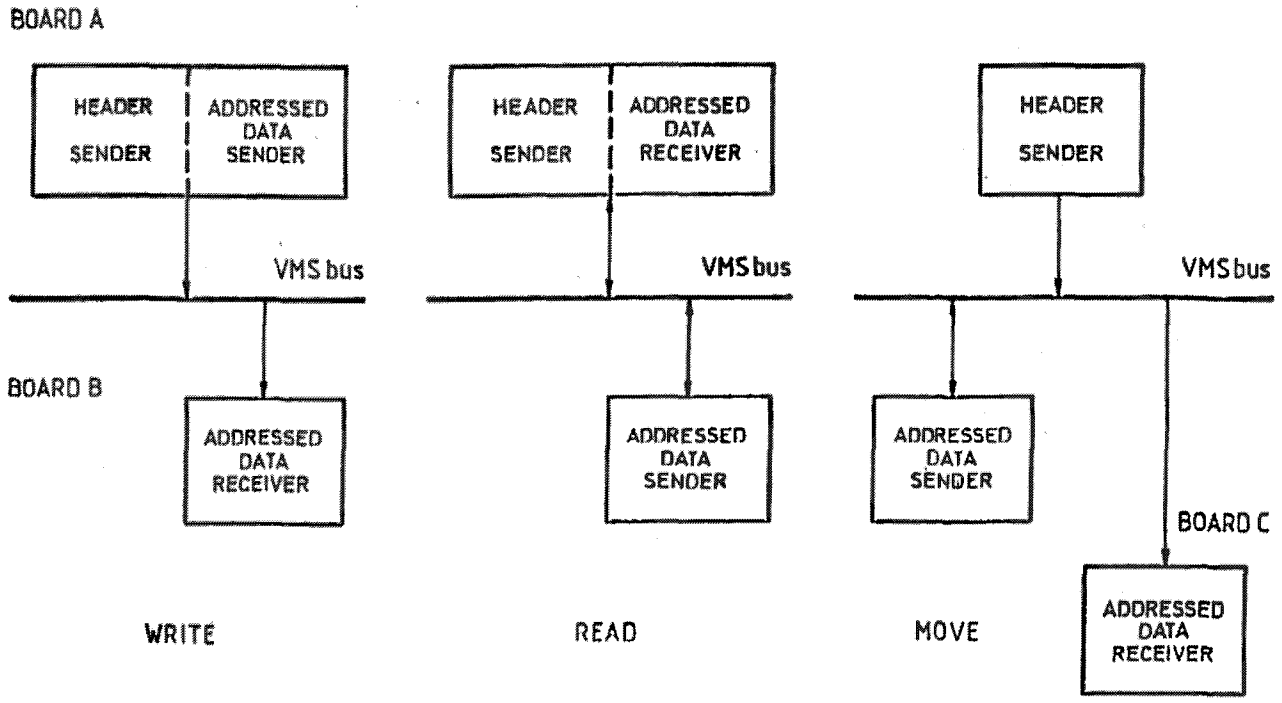
INTERRUPT



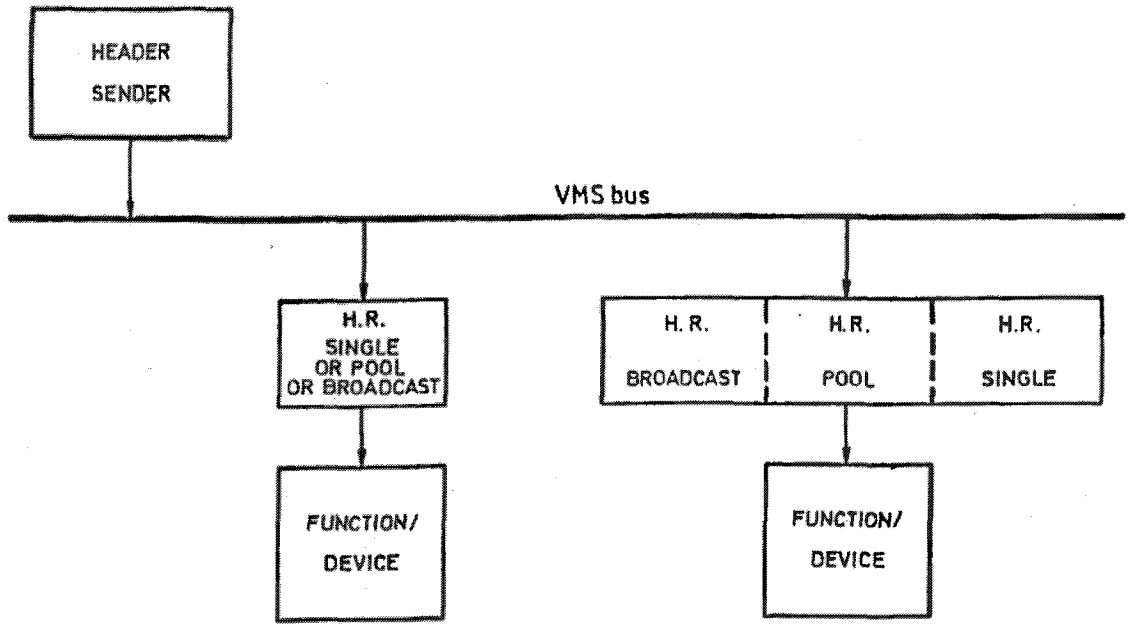
REGISTERS / CONTROL (I)



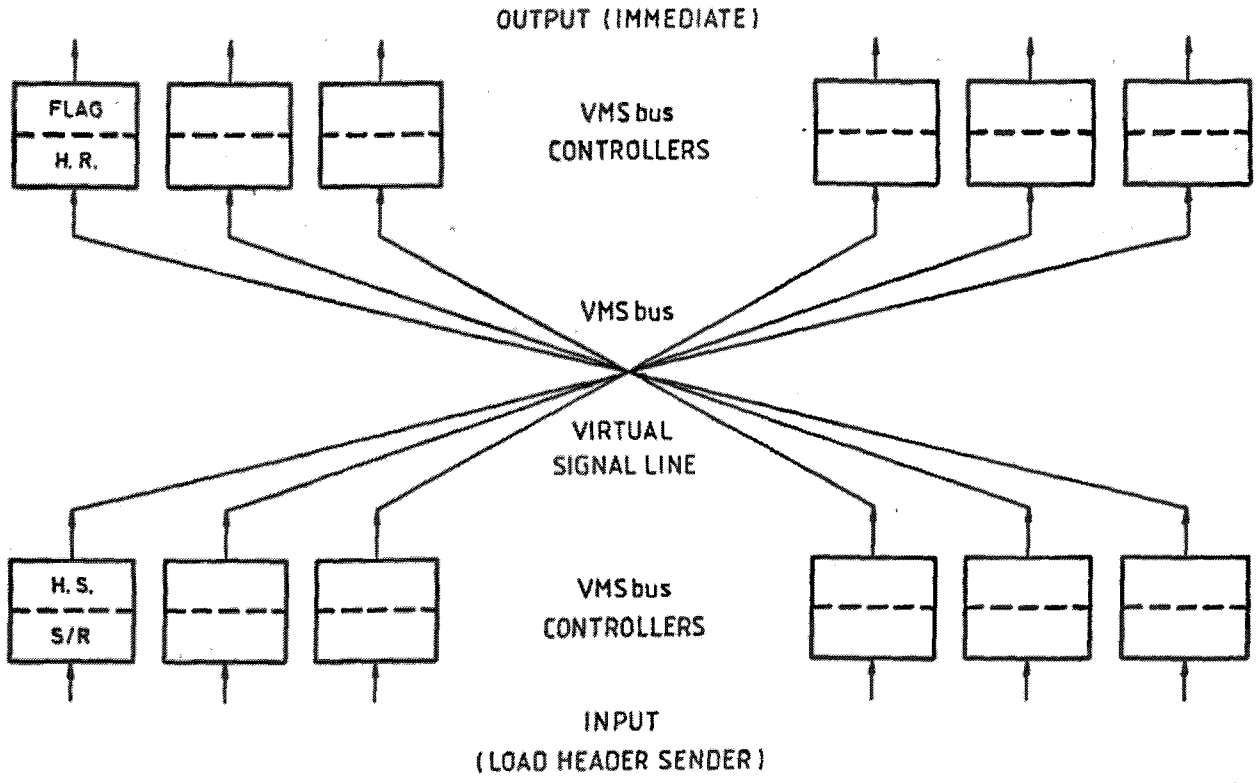
REGISTERS / CONTROL (II)



DATA TRANSFER



ADDRESSING


OUTPUT (IMMEDIATE)
FLAG
H. R.
VMS bus
CONTROLLERS
VMS bus
VIRTUAL
SIGNAL LINE
H. S.
S/R
VMS bus
CONTROLLERS
INPUT
(LOAD HEADER SENDER)

VME PROTOCOL CHIPSET USING PROGRAMMABLE LOGIC DEVICES

L.Gustafsson, P.Gällnö

DD Division, CERN, Geneva, Switzerland

Geneva, 8 October 1985

Abstract: We have investigated the availability, cost and performance of the VME protocol chipset. We have found that for most applications it is possible to use a simpler chipset built with programmable logic devices. We describe how to implement a Bus requester, Priority arbiter, Bus interrupter, Data transfer master and Slave controller, using finite state machine descriptions that easily can be transferred to programmable logic devices.

1. Introduction

We are using a VME system in order to test the main features of a general channel interface. In this interface we need a bus requester, bus interrupter, master and slave controller, see figure 1. These parts are commercially available from different manufacturers [1], but price/performance is poor compared with a functionally equivalent design using programmable logic devices (PLD). In order to test our designs in a logic simulator a rudimentary PRI arbiter and interrupt handler have been implemented in software. Similar designs have been presented [2] [3], but we hope to have found new solutions to problems in the design of VME modules. Among those problems one can mention the many straps used in some modules in order to get flexibility.

In our implementations of different protocol chips we have tried to isolate general parts from the VME protocol specification. These parts have then been transformed into finite state machine descriptions. As the VME protocol is asynchronous we have developed computer programs that can treat asynchronous machines in a race free manner and give logic equations that can be used directly in a PLD assembler.

2. The Bus requester

The first part of the protocol that has been investigated and implemented is the bus request whose finite state description is shown in figure 2. When a bus request (BR) is received from a user device a

signal (BRX) is transmitted and encoded for the request level that has been chosen. After that, it waits for a bus grant (BGR) from a local arbiter that compares the local bus request level with a bus grant level sent from the global bus arbiter. When granted the bus it also checks that AS* is high. This is done in order to pipeline with other processes. A bus busy (BBSY*) is sent back to the global arbiter in order to clear the grant level and make it possible to start a new bus arbitration cycle. When the bus grant level disappears the requester machine goes to a so called operate state waiting either for a release of the bus request, or that a bus clear (BCLR*) is received. The release modes that have been implemented are RWD and ROR. When a bus clear is received, time is given to end the current cycle and issue a new bus request. Our present implementation of the bus requester is shown in figure 3 and comprises two 20L8 PLD's. One of the PLD's houses the bus request encoder and the other implements the state machine described above. The benefit of using a PLD for the request level encoding is that the number of straps otherwise needed can be reduced, and a possibility exists to allocate request levels dynamically. In the encoder PLD we have also incorporated special switches to handle some of the metastability problems that can occur during the local arbitration [4]. In systems with frequent requests from different sources we recommend a special study before choosing the type of arbitration switches. One can level criticism against the VME specification as we have not found any recommendations when to use and not to use the system clock (SYSCLK*).

3. The PRI arbiter

As has been mentioned above we use a soft arbiter in order to test the bus requester in a logic simulator. This arbiter is described in figure 4. When the global bus arbitration is started it jumps to a state according to the bus request with the highest request level. This state produces a bus grant level and waits for a BBSY* signal to come back. This handshaking is following a complete level type protocol and specifications of waiting times etc. are not needed. If a bus request is recognised on a level higher than the current or if BBSY* goes high a new arbitration is started. A bus clear is sent if BBSY* is still low. In a hardware implementation of this arbiter it must be supplemented with parts to ensure that no spikes are generated during the arbitration phase.

4. The Master controller

The more complex chips used in VME can handle bus transfers directly but sometimes parts have to be used that have no direct interface to the transfer protocol. In figure 5 a master controller is described that can only handle the simpler transfer schemes, but that can easily be supplemented. When the current master sees that it is possible to use the bus, it decides on what type of transfer to perform. Suppose it wants to make a write to a slave. First it asserts a request out signal (REQ_OUT) which makes WRITE* go low and after a delay a local DS signal (LDS) goes high. This local DS can be used to generate the DS0*, DS1* and AS* signals. When a DTACK* or BERR* returns, LDS is taken away and a signal (RDY) is sent to the current master to check or update counters etc.. When REQ_OUT disappears the transfer cycle is ended. It is working in a similar way when reading from a slave. The only difference is that a request in signal (REQ_IN) is generated and the WRITE* signal is left high.

It is probably possible to design an even simpler controller that fulfils the same function as the one described. We see it as a strength of state machine descriptions that it is so easy to redesign a simple controller.

5. The slave controller

A controller that also has limited generality is described in figure 6. The only function that the slave controller has is to follow the timing in read and write operations. It is started on AS* low, IACK*

high and a valid address (VALAD) pending or if a STATUS/ID signal is coming from a local bus interrupter. There are two handshaking parts as we need a multiplexing/demultiplexing port. When a transfer is finished a data acknowledge (DTACK*) is sent back, which is handshaked with AS* and internal DS* both high.

The bus error generation in a slave is not possible to generalize as slaves differ too much from each other. The VME specification is also vague on this point, depending on how it is interpreted bus errors can be generated either much more or less then necessary.

6. The Interrupt requester

In the parts of the VME protocol that has been described above, complete handshaking chains exist but unfortunately that is not the case in the interrupt handling. The problem is that a qualifier is missing showing when the interrupter has taken away its interrupt request level. In the VME specification it is hoped that the interrupt level disappears at the same time as the DTACK* signal appears asserted at the interrupt handler but that is not completely satisfactory. In figure 7 an interrupter has also been implemented in PLD's. A service request (SREQ) is asserted by the user device. On this signal an interrupt request (IRQX) is generated and sent to an encoder. It then waits for an interrupt grant acknowledge (INTGR) which in its turn generates a STATUS/ID request to the slave controller. Already at this stage the interrupt request level can be taken away. It will have at least two following handshaking delays to disappear at the interrupt handler. When IACK* disappears as a result of a low going DTACK* it is necessary to send a service request end (SREND) and wait for the service request to be removed before idling.

If this part is implemented as an edge driven system it is possible to simplify the handshaking parts. The present implementation of the interrupter is using two 20L8 PLD's which is shown in figure 8. The commercially developed interrupters comprise a 8 bit status/id register, but a bigger word is often more useful.

7. The Interrupt handler

As in the development of the bus requester we have simulated the interrupter together with a soft interrupt handler that is shown in figure 9. When the arbitration is finished a bus request is started and after gaining the bus an interrupt acknowledge IACK* and IACKIN* is sent together with a 3 bit acknowledge address. A local master controller is started in order to get the STATUS/ID word transferred. If the datafield specification is not supported by the slave in question a BERR* is sent back. On the reception of a ready signal (RDY) from the master controller the service either continues or is terminated directly. It is important in the loop back to the idle state that the request has disappeared or malfunction will occur. In a hardware implementation it is necessary to guarantee metastate free outputs in the arbitration of the interrupt level that will get service. By the state assignments we are using in the state machine, we think that we have eliminated the major part of this problem. All states traversed from an instability are tied back to the idle state. In a final system it would be preferred also to have a possibility to mask interrupts.

8. Conclusions

We think that it is possible to build VME systems with many bus requesters and bus interrupters frequently asking for service and still have an acceptable runtime before failure. This can be done by paying attention to the metastability problem that can occur at the synchronization points for different

processes, especially when using PLD constructions. On some points the VME specification is still very vague and the interrupt handling is insecure from a handshaking point of view.

References

[1]. S.Baliga, Three-chip control set trims VME Bus logic for asynchronous systems
Electronic Design, January 24, 1985, p.207

[2]. R.Nilsson, The PLO 14/15/16/17 VME Bus interrupters
ISSN 0346-8887, Oct. 1983
Institute of technology, Uppsala University

[3]. W.Peterson, VME Bus requester releases bus four ways
EDN, August 8, 1985, p.241

[4]. K.Marrin, Metastability haunts VME BUS and Multibus II system designers
Computer Design, August 1, 1985, p.29

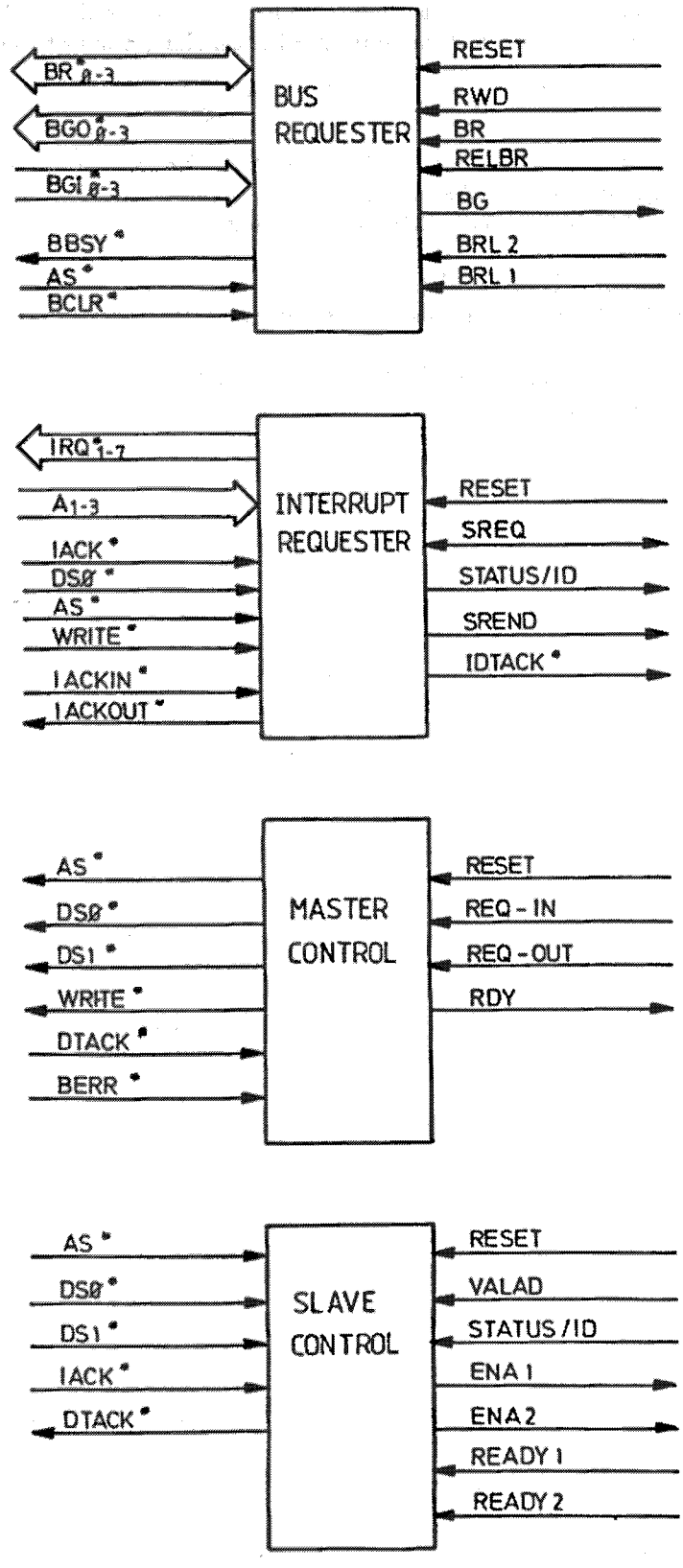

BR* Ø-3
BGO* Ø-3
BGI* Ø-3
BBSY*
AS*
BCLR*
BUS REQUESTER
RESET
RWD
BR
RELBR
BG
BRL 2
BRL 1
IRQ* 1-7
A1-3
IACK*
DSØ*
AS*
WRITE*
IACKIN*
IACKOUT*
INTERRUPT REQUESTER
RESET
SREQ
STATUS/ID
SREND
IDTACK*
AS*
DSØ*
DS1*
WRITE*
DTACK*
BERR*
MASTER CONTROL
RESET
REQ-IN
REQ-OUT
RDY
AS*
DSØ*
DS1*
IACK*
DTACK*
SLAVE CONTROL
RESET
VALAD
STATUS/ID
ENA 1
ENA 2
READY 1
READY 2

Fig. 1

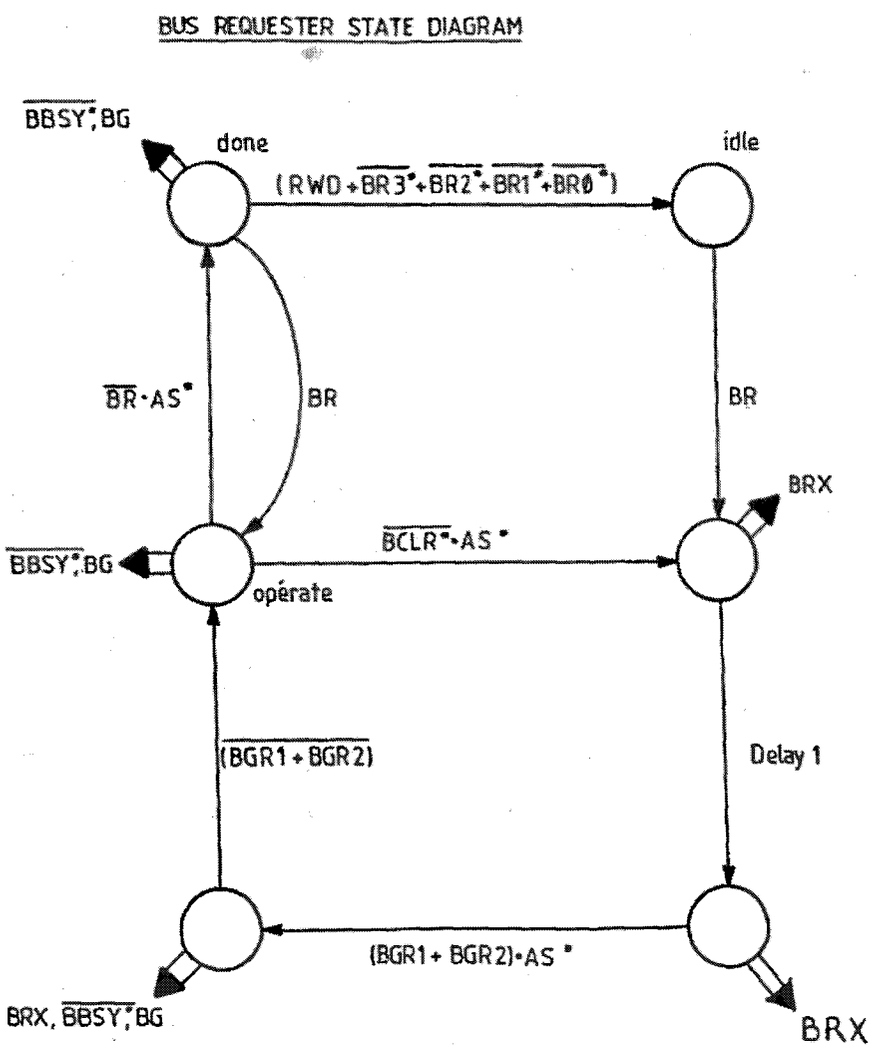

BUS REQUESTER STATE DIAGRAM
BBSY*,BG
done
idle
(RWD+BR3*+BR2*+BR1*+BR0*)
BR·AS*
BR
BR
BRX
BBSY*,BG
BCLR*·AS*
opérate
(BGR1+BGR2)
Delay 1
(BGR1+BGR2)·AS*
BRX, BBSY*,BG
BRX

Fig. 2

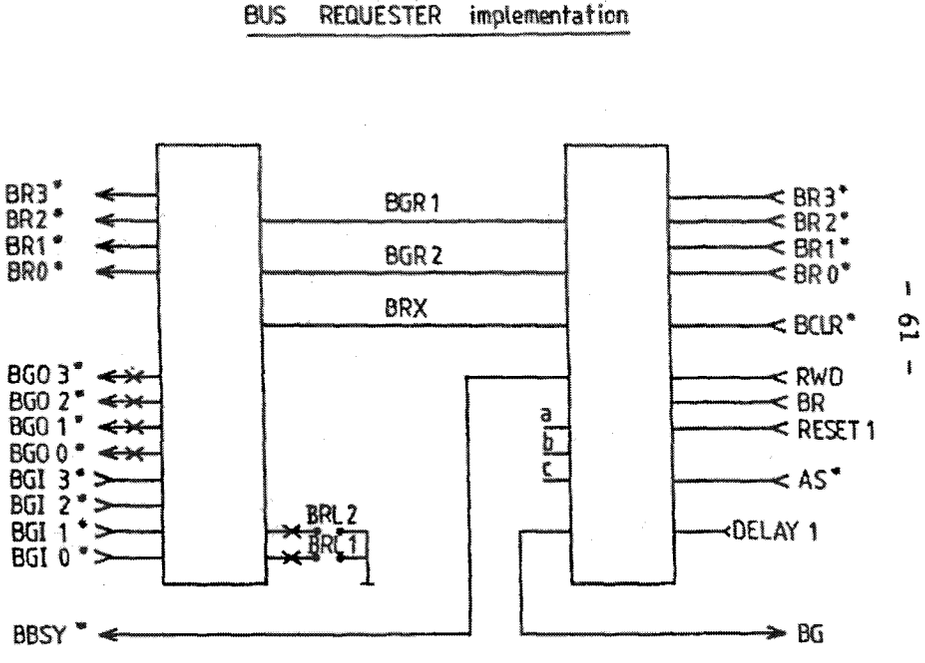

BUS REQUESTER implementation
BR3*
BR2*
BR1*
BR0*
BGR1
BGR2
BRX
BR3*
BR2*
BR1*
BR0*
BCLR*
BGO3*
BGO2*
BGO1*
BGO0*
BGI3*
BGI2*
BGI1*
BGI0*
RWD
BR
RESET1
AS*
DELAY1
BRL2
BRL1
BBSY*
BG

Fig. 3

PRI ARBITER STATE DIAGRAM

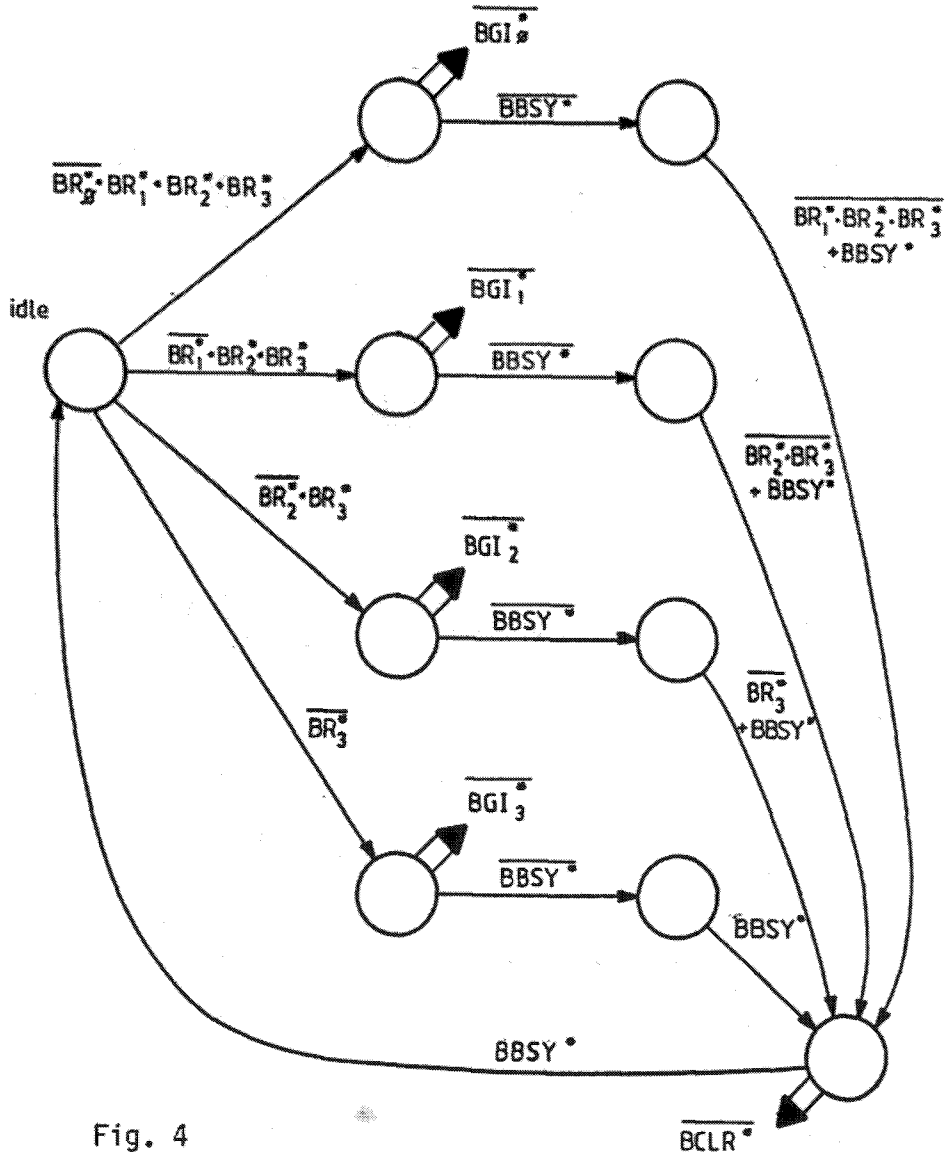


Fig. 4

MASTER CONTROL STATE DIAGRAM

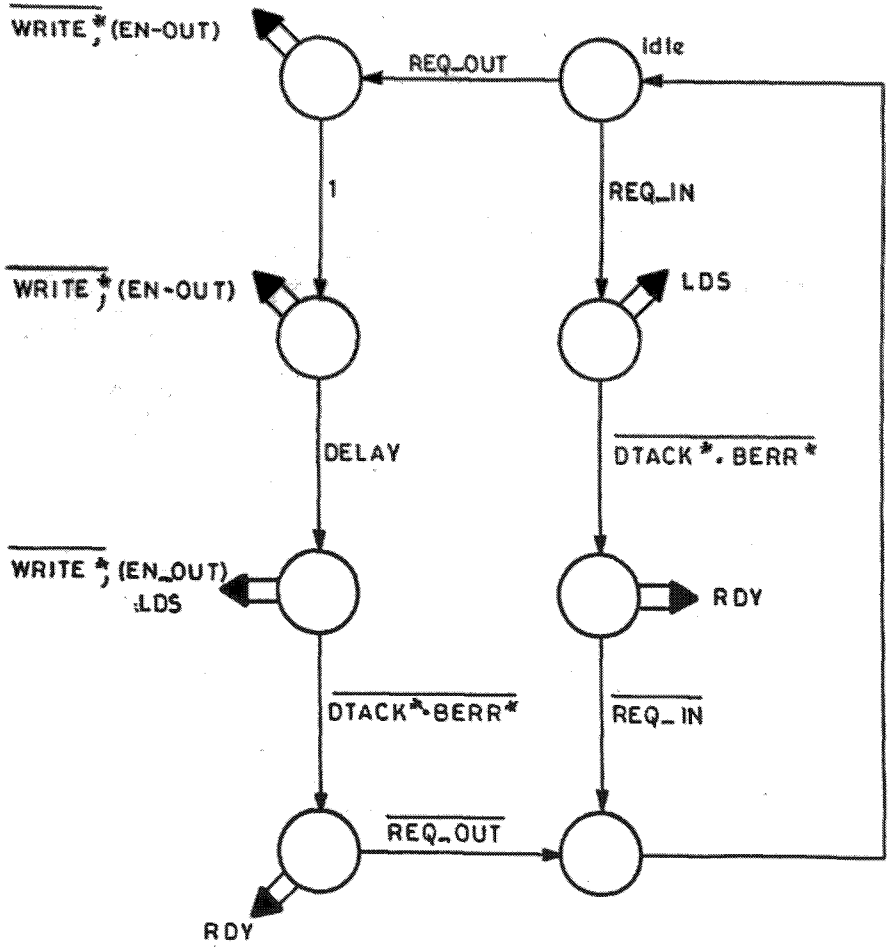


Fig. 5

SLAVE CONTROL STATE DIAGRAM

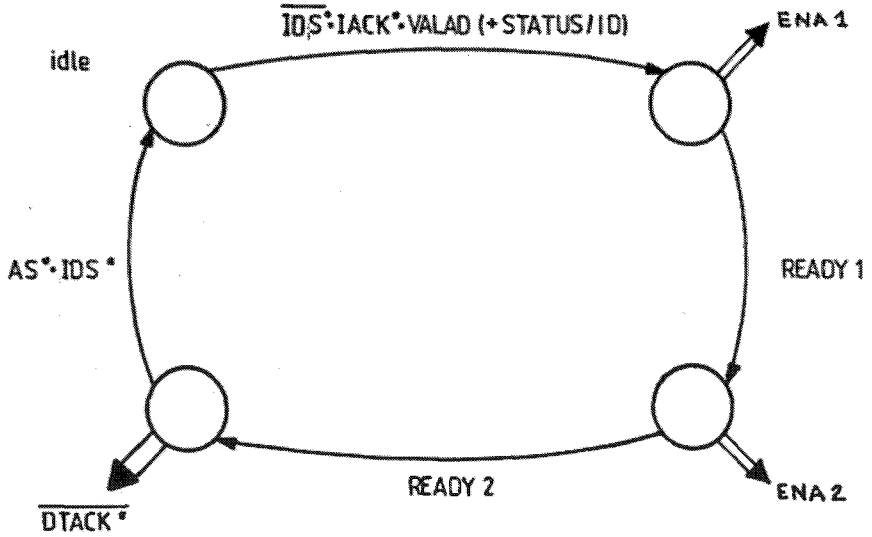


Fig. 6

INTERRUPT REQUESTER STATE DIAGRAM

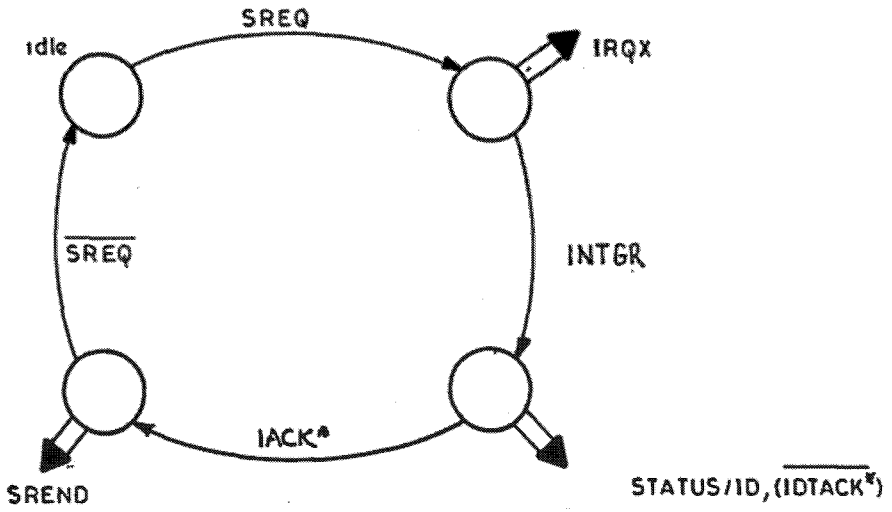


Fig. 7

INTERRUPT REQUESTER implementation

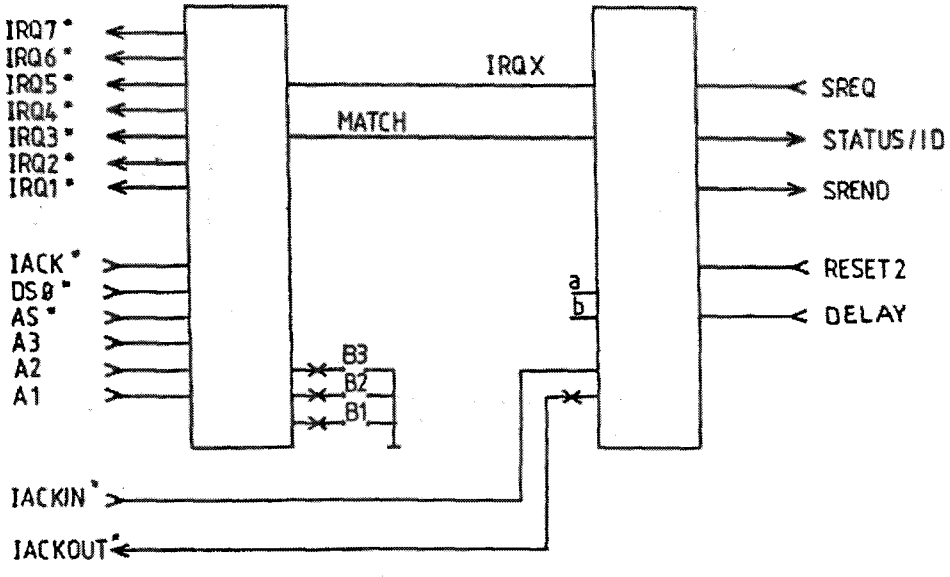


Fig. 8

INTERRUPT HANDLER

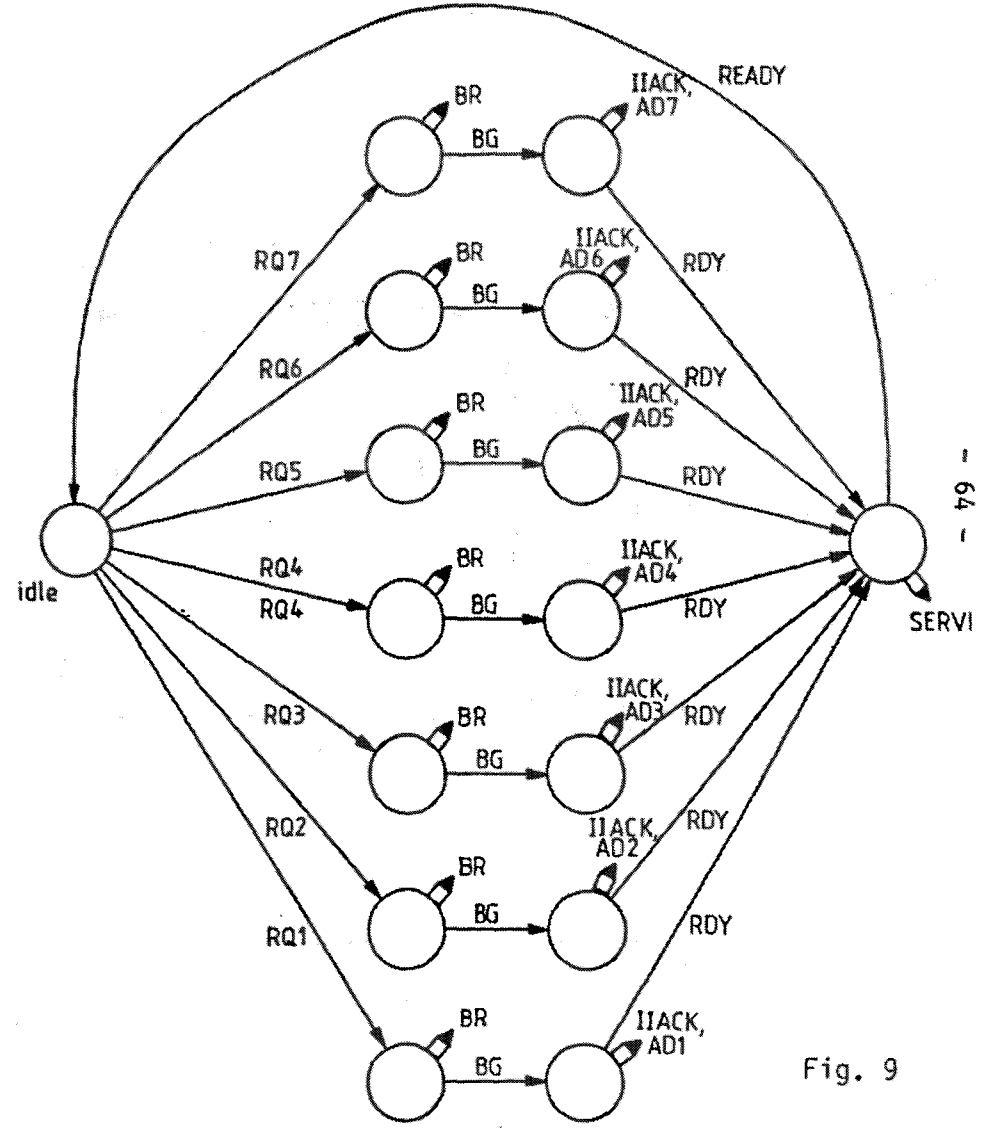


Fig. 9

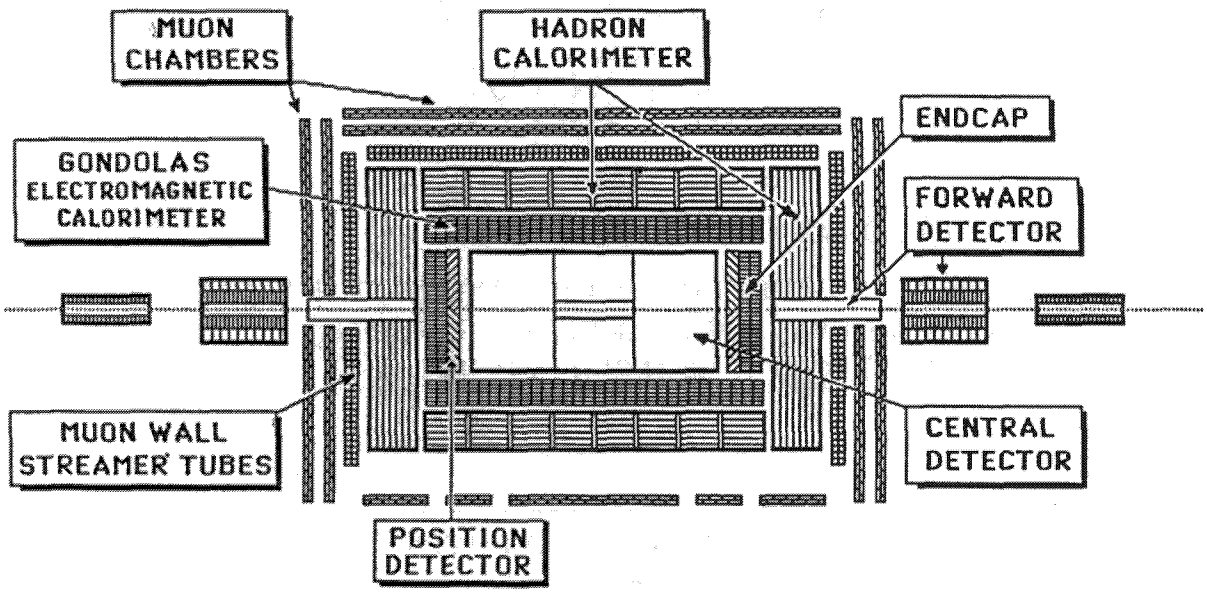
UA1 VME READOUT SYSTEM

UA1 COLLABORATION

S. CITTOLIN, M. DEMOULIN, P. GIACOMELLI, B. HAYNES,
W. JANK, P. PETTA, E. PIETARINEN, J.P. PORTE,
P. ROSSI, D. SAMYN, H. VON DER SCHMITT

Presented by S. Cittolin at the "VMEbus in Physics" Conference
held in CERN on 7-8 October 1985

UA1 EXPERIMENT LAYOUT

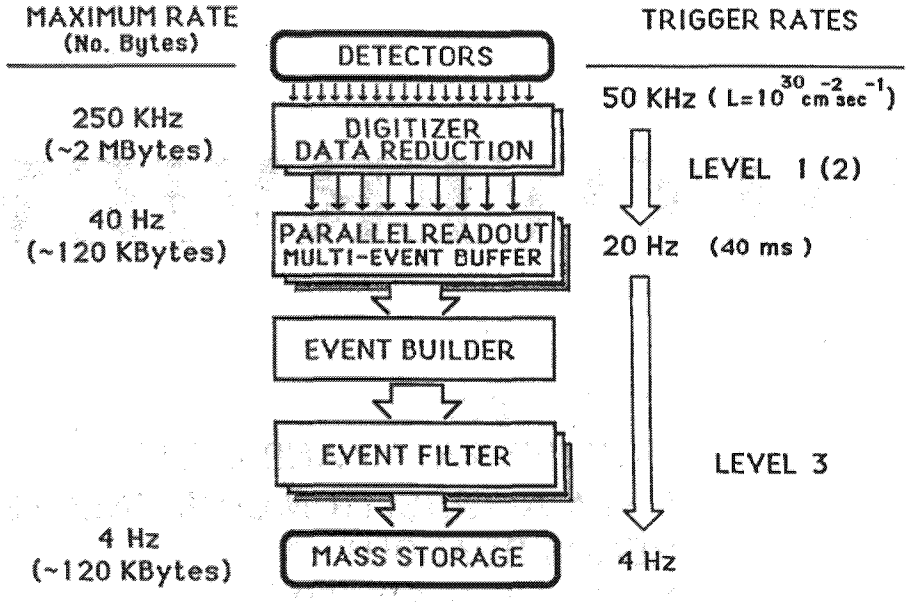


Detectors	Channels	Raw Data Bytes
Central Detector	6250	1600000
Hadron Calorimeter	1184	2400
Electromagnetic Cal.	2032	4100
Position Detector	4000	8000
Forward Chamber	2000	≈6000
Muon Chamber	6000	≈500
Muon Wall	40000	40000

UA1 OVERVIEW

DETECTORS
TRIGGER RATES AND LEVELS
DATA ACQUISITION COMPONENTS
REMUS READOUT
168E EVENT FILTER

DATA ACQUISITION STAGES AND RATES



AVERAGE EVENT SIZE	120 KByte
MAXIMUM SYSTEM THROUGHPUT	450 KByte/sec
MULTIEVENT BUFFER DEPTH	4 (16) Events
DEAD TIME	(100µsec to ≈40ms)
READOUT ERROR RATE	≤ 5 %
SYSTEM EFFICIENCY (at 4 Hz)	90 %
NUMBER OF READOUT CRATES	170
NUMBER OF PIN CONNECTIONS	$\approx 10^5$

TRIGGER LEVELS

BETWEEN BUNCH CROSSING (<4µSEC)

LEVEL 0 PRETRIGGER [HODOSCOPE COUNTER, STANDARD LOGIC]

LEVEL 1 CALORIMETER [HARD-WIRED PROCESSOR, FAST ADC]

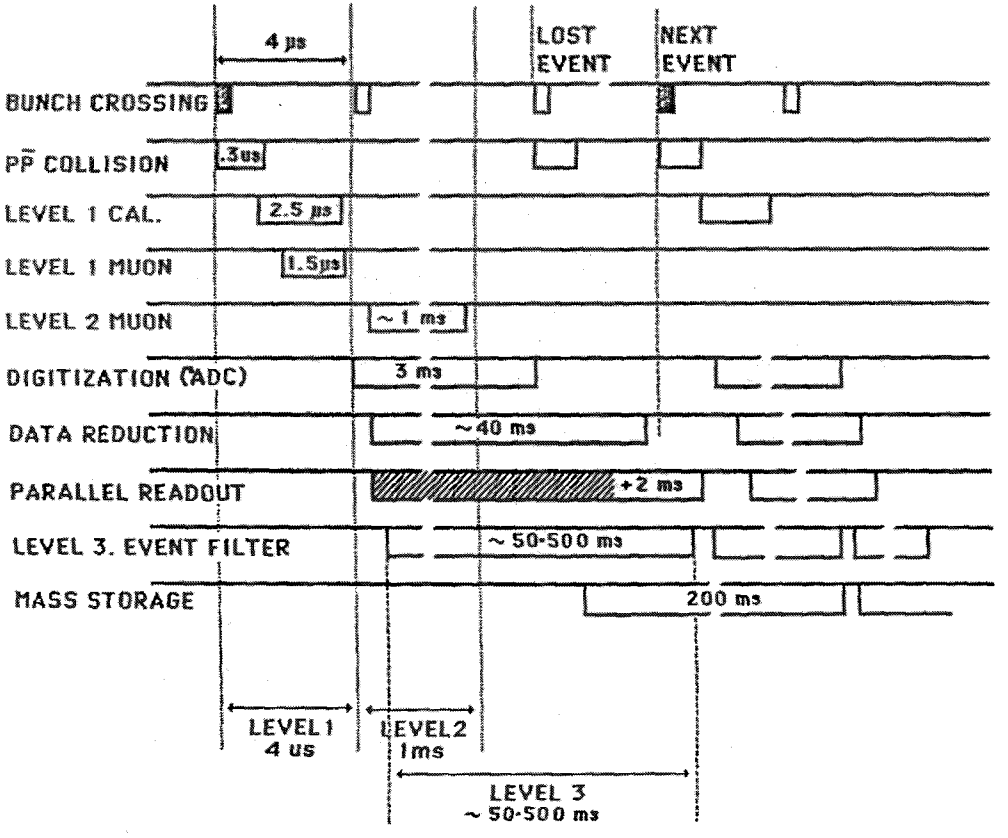
LEVEL 1 FAST MUON [MUON DETECTOR HIT PATTERN]

DURING /AFTER DATA READOUT AND REDUCTION

LEVEL 2 MUON TRIGGER [TRACK RECONSTRUCTION BY µP]

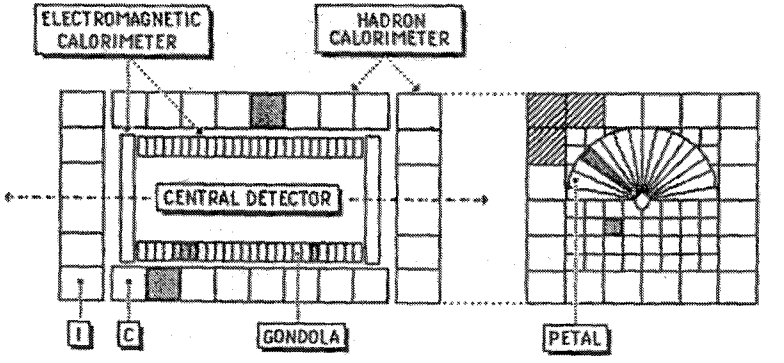
LEVEL 3 EVENT FILTER [ENERGY RECONSTRUCTION BY 168E]

TIMING OF TRIGGER LEVELS



LEVEL 1. CALORIMETER TRIGGER

TWO HARD-WIRED PROCESSORS (LOOKUP TABLE/ADDER/ COMPARATOR).
DEDICATED READOUT OF 288 CHANNELS



TRIGGER SELECTION

STANDARD CALORIMETER TRIGGER

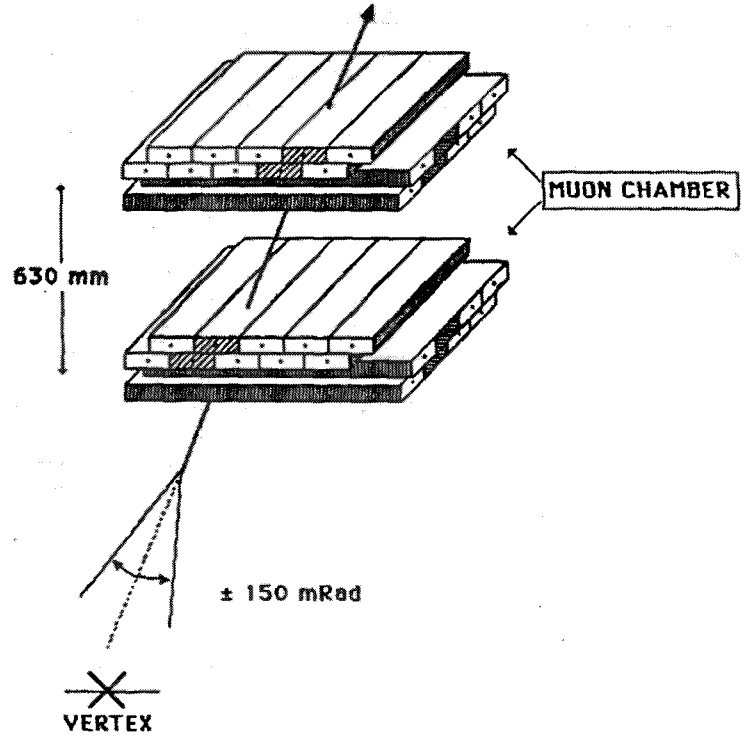
1 electron > 10 Gev
2 electron > 6 (8) Gev
1 jet > 25 (30) Gev
Σ Et [G+C] > 80 (60) Gev
Et [imbal.] > 17 / 1 jet>15 Gev

BACKGROUND TRIGGER

High Et Hadron > 8 Gev ("pion")
1 jet > 15 Gev

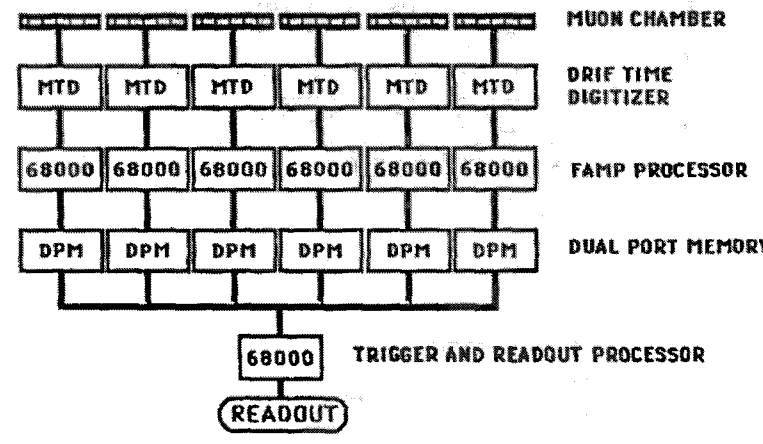
LEVEL 1. FAST MUON TRIGGER

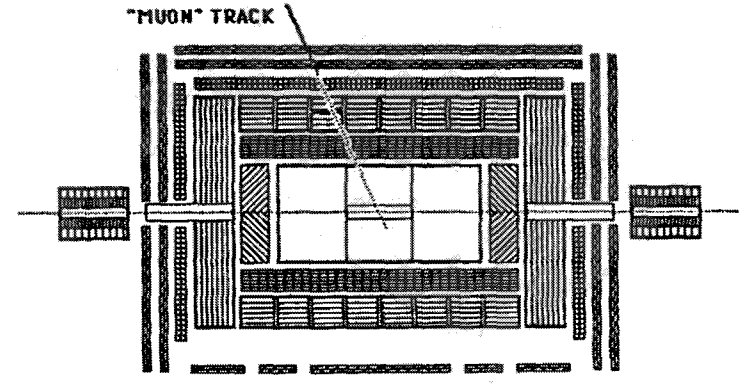
LOOK UP TABLE PATTERN RECOGNITION OF TUBE HIT.
SEARCH FOR TRACKS POINTING TO THE INTERACTION VERTEX WITHIN ± 150 mRad.



LEVEL 2. MUON TRACK TRIGGER

MUON TRACK RECONSTRUCTION BY THE FAMP M68000 MICROPROCESSOR SYSTEM.
USED TO FLAG EVENTS FOR FURTHER SELECTION.

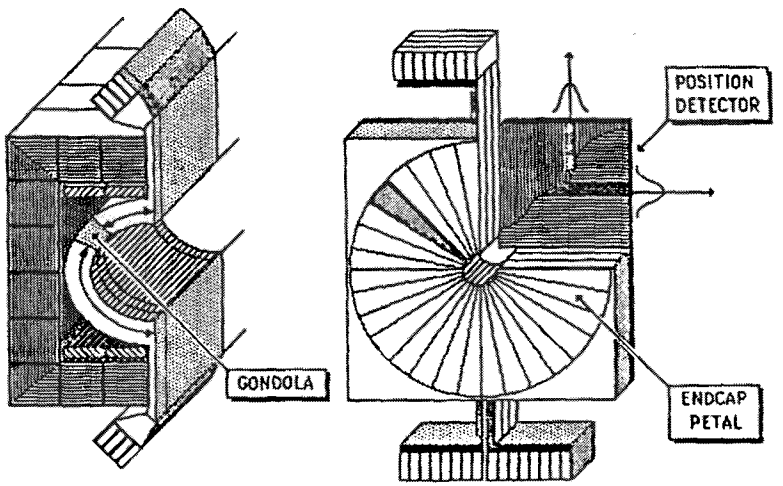




LEVEL 3. EVENT FILTER

A FIVE 168E STACK RUNNING A 25000 LINES FORTRAN PROGRAM.

- DATA STRUCTURE VALIDATION AND STATISTICS
- ELECTRON ENERGY CALCULATION USING CALIBRATION CONSTANTS, GEOMETRY CORRECTION AND CLUSTER IDENTIFICATION FOR GONDOLA AND ENDCAP CALORIMETER
- JET IDENTIFICATION
- MISSING ENERGY
- MUON TRACK RECONSTRUCTION
- BACKGROUND
- EVENT FLAG : "SPECIAL" "NORMAL" "REJECTED"



UA1 DATA TAKING RUN STATISTICS

DATA TAKING		1981	1982	1983	1984
LUMINOSITY $cm^{-2}sec^{-1}$ (Integrated L. nb)		10^{27} (1)	10^{28} (18)	10^{29} (118)	$3 \cdot 10^{29}$ (270)
COLLISION RATE Hz		40	400	4000	12000
LEVEL 1 RATE Hz		1	≤ 2	≤ 3	≤ 10
LEVEL 3 RATE Hz		—	—	≤ 3	≤ 10
MASS STORAGE Hz		1	≤ 2	≤ 3	≤ 4
RAW DATA TAPES	NORMAL	1000	1500	2700	3500
	SPECIAL	—	—	300	550

UA1 DATA ACQUISITION COMPONENTS

2	NORD 100/500 2 MByte MEMORY 3 MTU 6250 BPI 125 IPS [Data acquisition and physics monitor]
6	168E IBM EMULATORS 64 KW PROGRAM MEMORY 512 KB DATA MEMORY [Event filter and event display]
7	FAMP M68000 µP [Muon readout and second level trigger]
20	SUPER CAVIAR MICRO COMPUTERS M6800, AMD 9511 FP, 256 KB RAM, 84 KB EPROM [Equipment test and control]
200	MC68B00 µP ROP, GPMC [Readout electronics control]
110	SIGNETICS 8X300 [Central detector data reduction and formatting]
200	CAMAC CRATES
16	VME CRATES
60	CPUA1. VME MC68010 CPU 256 KB RAM, 32 KB EPROM, NS 16001 FP VME/VMX Bus [Parallel readout and event builder processors]
150	VME modules. Dual Port RAM, EPROM, REMUS branch driver, Graphics ....
24	MacVEE (Apple Macintosh with VME interface) [VME interactive interface for test, monitor and software development]
6	3081E IBM EMULATORS. 2 MB MEMORY [Event filter and off line data preprocessing]

UA1 DATA ACQUISITION PARAMETERS

EVENT SIZE 100-200 KBytes

MASS-STORAGE MAXIMUM THROUGHPUT 450 KB/s

DEAD TIME :
- IF READOUT FREE = DOUBLE BUFFER DEAD TIME
  (CD = 4us , IAROCCI = 100us, CALORIMETER = 10ms)
- DURING READOUT = ROP DEAD TIME (~40ms)
- IF MULTIEVENT BUFFER FULL > 100ms

READOUT ERROR RATE ~ 5%

THIRD LEVEL TRIGGER MAXIMUM RATE ~10Hz

MAGNETIC TAPE MAXIMUM WRITE RATE ~ 3 Hz

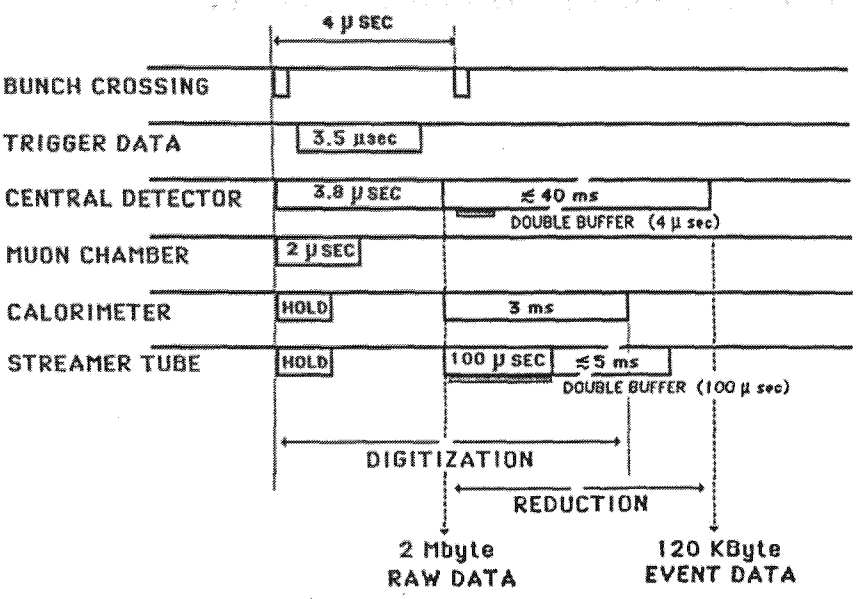
SYSTEM RUNNING EFFICENCY ~ 90%

NUMBER OF READOUT CRATES = 170

DATA DIGITIZATION AND REDUCTION

Detector	Raw data bytes	Digitizing time	Reduction time	Processors
Calorimeter	14000	3 ms	-	LeCroy 2282 (9)
Streamer tube	40000	100 µs	≤ 5 ms	VME CPUA1 (6)
Central detector	1600000	4 µs	≤ 40 ms	ROP (110)

DATA DIGITIZATION AND REDUCTION TIMING



DATA DIGITIZATION AND REDUCTION

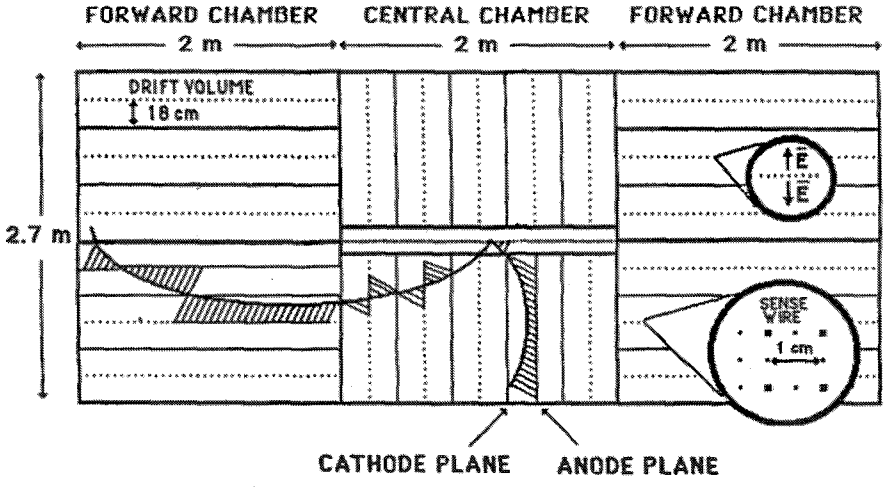
LARGE DATA VOLUME. 1600000 Byte/Event

REAL TIME PARALLEL PROCESSING. 40ms/Event

PROGRAMMABLE DATA FORMATS

CENTRAL DETECTOR

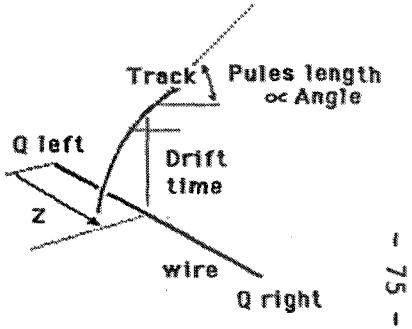
CYLINDRICAL DRIFT CHAMBER WITH "IMAGE" READOUT



CENTRAL DETECTOR DATA VOLUME	
WIRE	256 Bytes
CRATE	15360 Bytes
ALL SYSTEM	1600000 Bytes

HIT INFORMATION

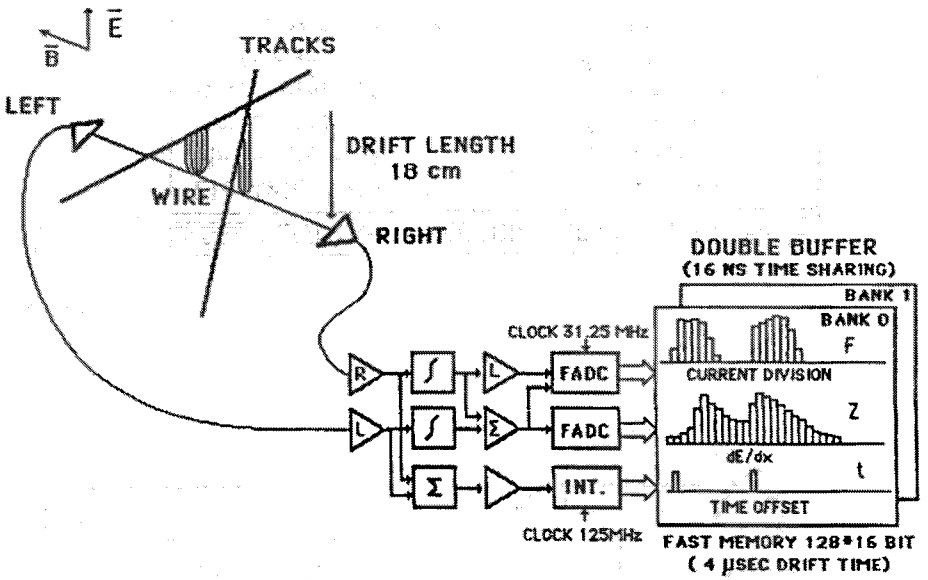
DRIFT TIME (4ns Unit)
WIRE/MODULE/CRATE
Q_{Total}
$Z = Q_{Left} / Q_{Total}$
PULSE LENGTH (32ns Unit)



CENTRAL DETECTOR PARAMETERS		
Electric field	1.5	KV/cm
Magnetic field	0.7	Tesla
Drift velocity	53	Micron/nsec
Drift angle (B = 0.7 T)	23	Degree
Drift gap	18	cm
Sense wire spacing	1	cm
Sense wires in drift volume	175 80	(Forward) (Central)
Drift volumes	46	
Sense wires	6250	

TYPICAL EVENT	
~100 TRACKS (~100 Hit/Track)	
NON ZERO DATA	300000 Bytes
HIT DATA	80000 Bytes

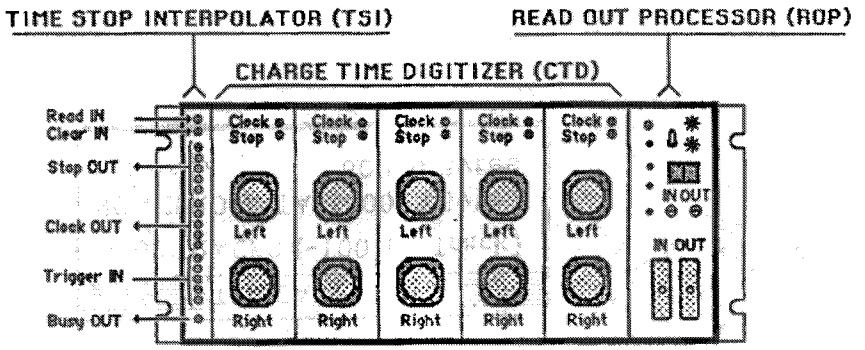
CENTRAL DETECTOR
ELECTRONICS CHANNEL BLOCK DIAGRAM



32 ns SAMPLE (16 Bits)		
$F = \frac{(Q_T + b)}{(A \cdot Q_T + b + B)}$	dE/dx.	6 Bits
$Z = Q_L / Q_T$	Curr. Division.	6 Bits
t = TDC + Time Tag (4ns)	Drift time.	4 Bits

A = .8162 compression constant
B = base line b = scale

READOUT PROCESSOR CAMAC CRATE



CENTRAL DETECTOR CAMAC CRATE

CAMAC ESONE COMPEX PROTOCOL (250ns)
TSI Time Stop Interpolator
5 CTD Charge Time Digitizers.(60 wires)
ROP Read Out Processor

UA1 CENTRAL DETECTOR READOUT

110 CRATES (6250 wires)
110 ROP (Read Out Processor) MODULES
110 TSI (Time Stop Interpolator) MODULES
500 CTD (Charge Time Digitizer) MODULES

CENTRAL DETECTOR DATA FORMATS

Data pattern data	200-1600000 Bytes
Pulse data (unpacked)	≈300000 Bytes
Packed data	≤100000 Bytes
Super packed	≤60000 Bytes

ROP DUAL PROCESSOR FUNCTIONS

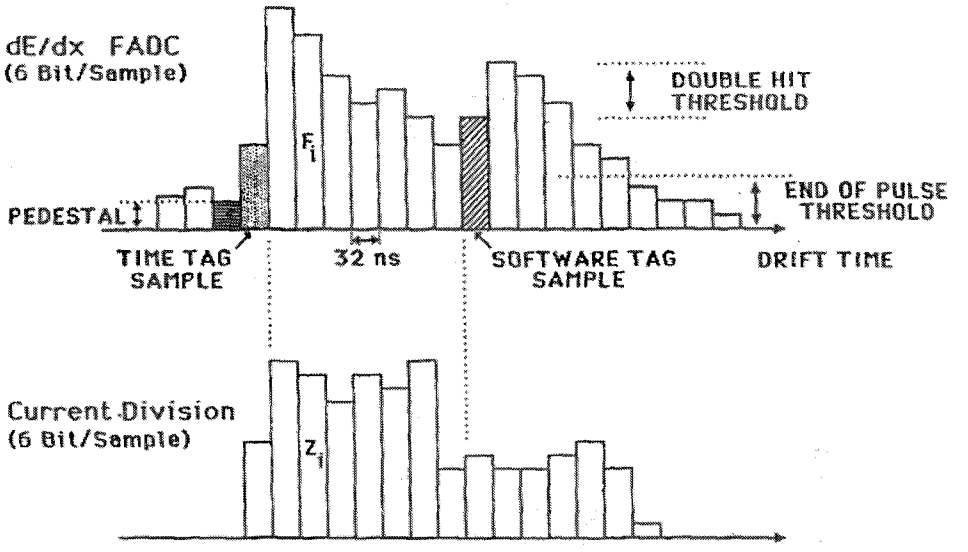
FAST DATA REDUCTION

- CTD readout, zero skipping control
- Hit parameters calculation
- Double pulse detection
- REMUS output

LOCAL SYSTEM AUTONOMOUS FUNCTION

- ROP autotest
- Front panel control
- ADLC LAN driver
- Electronics parameter evaluation
- Histogramming and statistics accumulation
- CTD internal DAC calibration and adjusting
- Data communication with supervisor

UNPACKED HIT DATA FORMAT

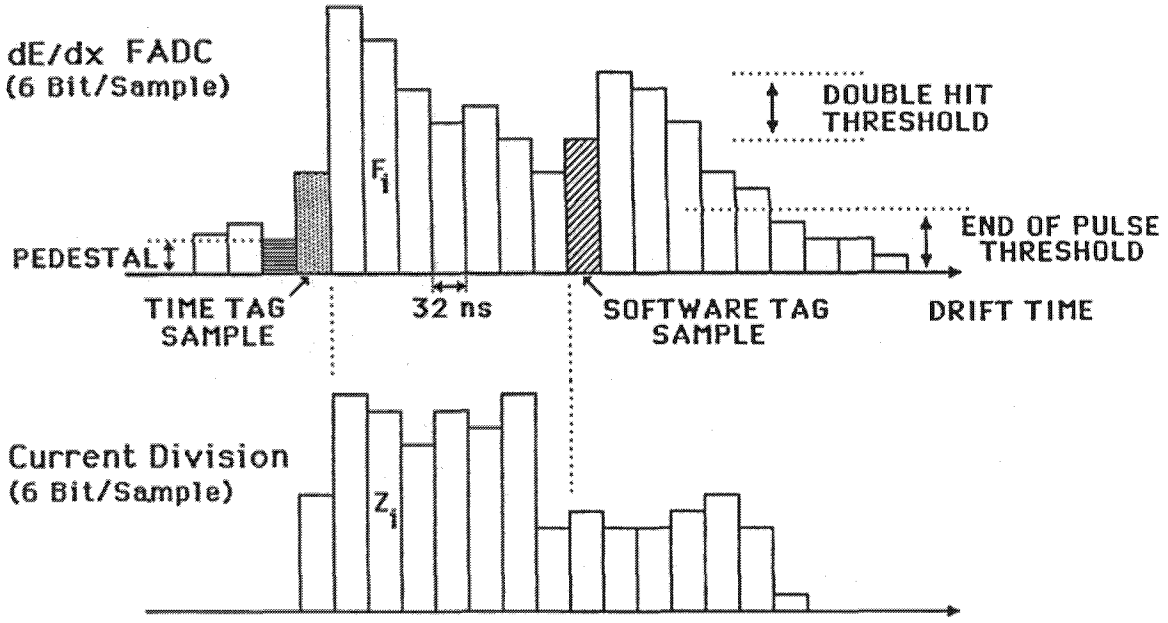


HIT UNPACKED DATA FORMAT

Hit frame bit	1	DRIFT TIME 4 ns U	S	WIRE	Hit Header
	0	F_0 sample	Z_0 sample	MOD	
	0	F_1 sample	Z_1 sample	MOD	
	0	F_i sample	Z_i sample	MOD	
	0	F sample	Z sample	MOD	
Hit frame bit	1	DRIFT TIME 4 ns U	S	WIRE	Next Hit Header

DRIFT TIME (4ns Unit) = (No. Channel)*8 + Time offset
WIRE = No. wire (1÷12)
S = Hardware (0)/ Software (1) tag
MOD = No. CTD module (1÷5)
F_i = FADC sample = (QT+B)/(A·QT+B+b)
Z_i = Current division sample = QL/QT

PACKED HIT DATA FORMAT



	HIT PACKED DATA FORMAT				
Hit frame bit	1	DRIFT TIME 4 ns U	S	WIRE	Hit Header
	0	$Q_L = \Sigma Q_L^i z_i$		MOD	Sum over n=8 samples
	0	$Q_T = \Sigma Q_T^i$		DT	
	0	$Q_L = \Sigma Q_i z_i$		MOD	Long pulse continuation
	0	$Q_T = \Sigma Q_i$		DT	
Hit frame bit	1	DRIFT TIME 4 ns U	S	WIRE	Next Hit Header

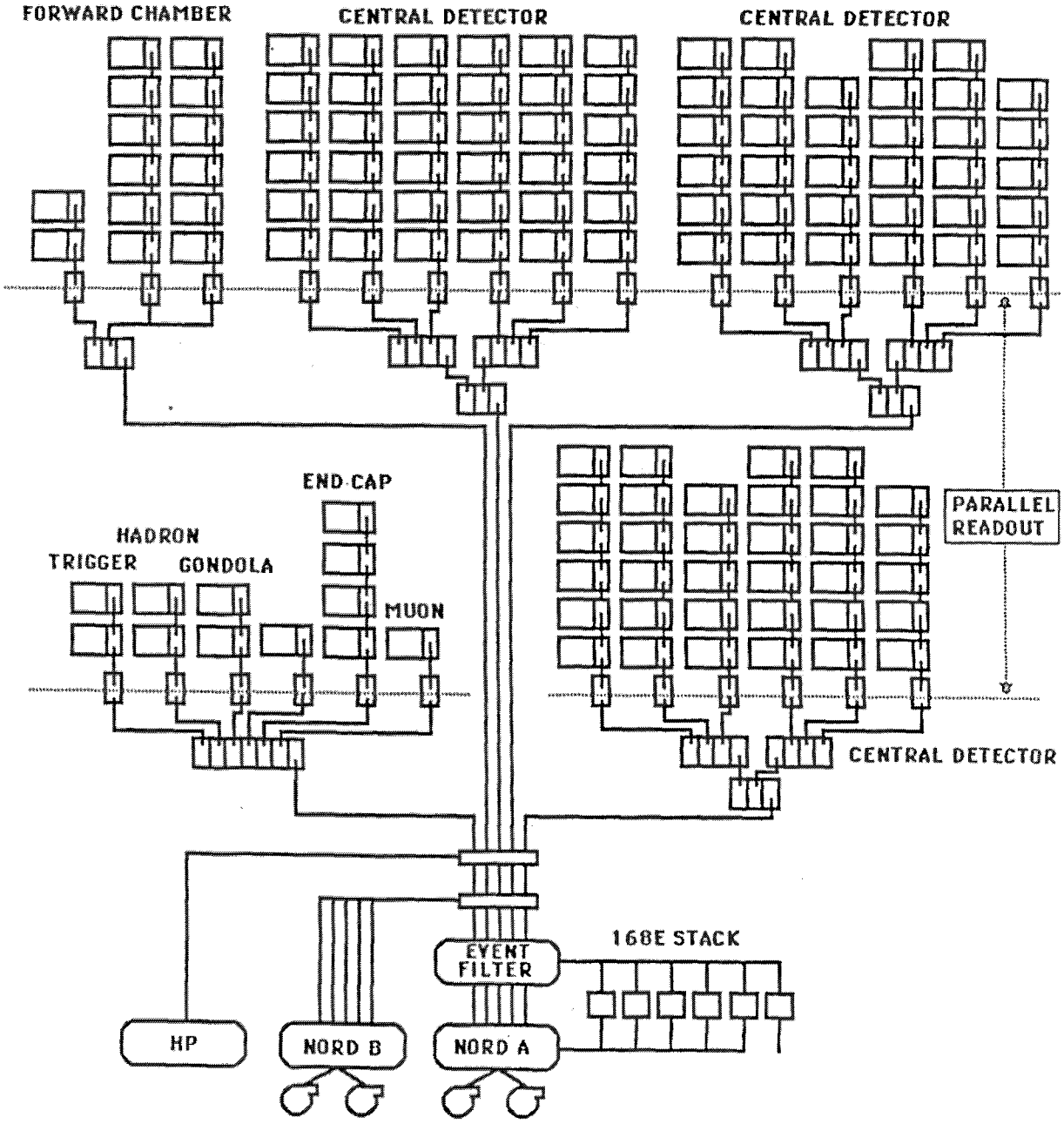
DRIFT TIME (4ns Unit) = (No. Channel)*8 + Time offset
WIRE = No. wire (1-12)
S = Hardware (0)/ Software (1) tag
MOD = No. CTD module (1-5)

$$Q \text{ Left} = \Sigma(Q_i \text{ Left}) = \Sigma \frac{64(z \cdot B(630000 - 63 \cdot A))}{((6300 - b)(63000 - A \cdot F))}$$

$$Q_T = \Sigma(Q_i \text{Left} + Q_i \text{Right}) = \Sigma \frac{B(630000 - 63 \cdot A)(F \cdot 100 - base)}{((6300 - b)(63000 - A \cdot F))}$$

DT = Pulse Length (unit 32 ns)

UA1 REMUS READOUT



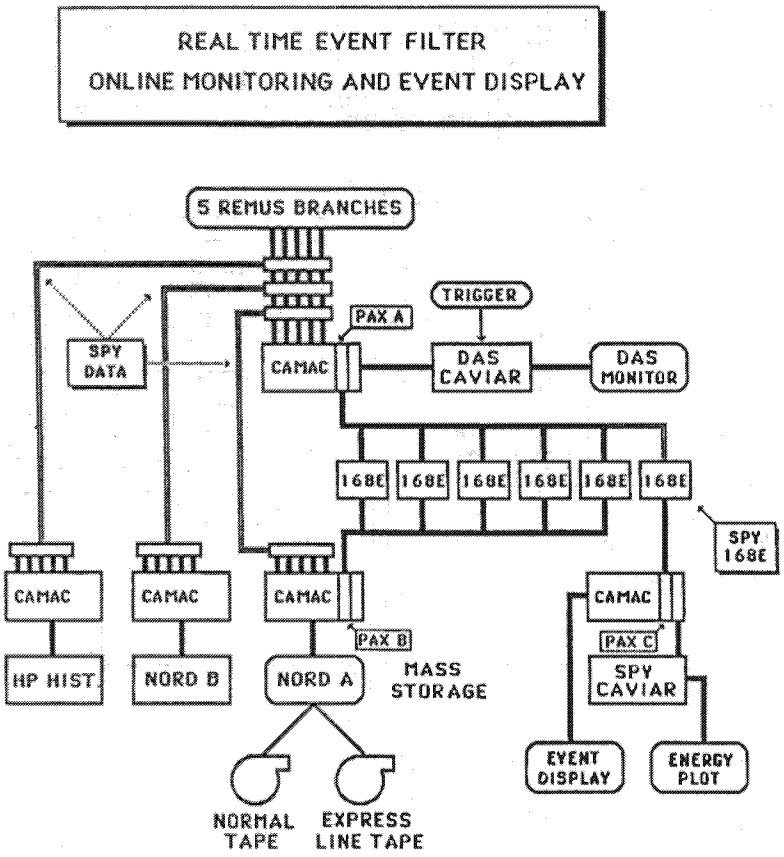
DATA READOUT

170 CAMAC CRATES READ VIA REMUS BUS

LOCAL DATA SAMPLING BY "SPY" TECHNIQUE

28 BRANCHES PARALLEL READOUT (≈50 MB/sec)

168E SYSTEM LAYOUT



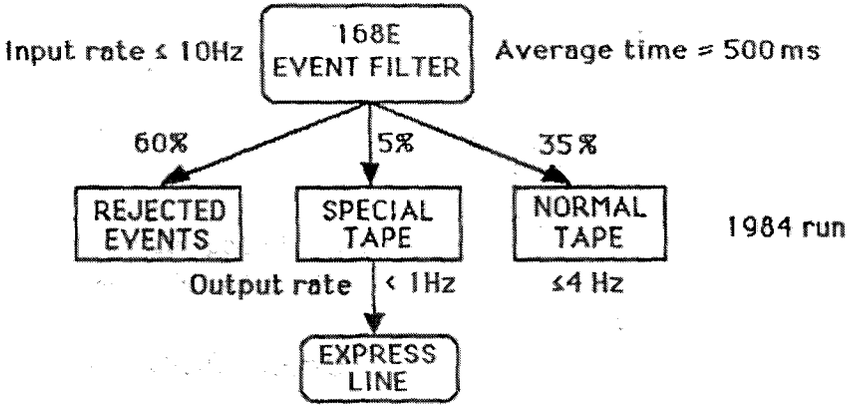
168E EVENT FILTER

ONLINE POWER OF 3 IBM 370/168 UNITS

6 168E WITH M6800 µP AS FRONT END

EVENT SELECTION AND EVENT DISPLAY

EVENT FILTER



SELECTION CRITERIA (average execution time)

ELECTRON (≈ 350 ms)
Calorimeter cell energy reconstruction.
Electron identification by E_{el} threshold cut, isolation check in space and in the hadron calorimeter energy deposition.

JET (≈ 40 ms)
Jet identification using energy vectors

ET
Total energy calculation from cell energy list

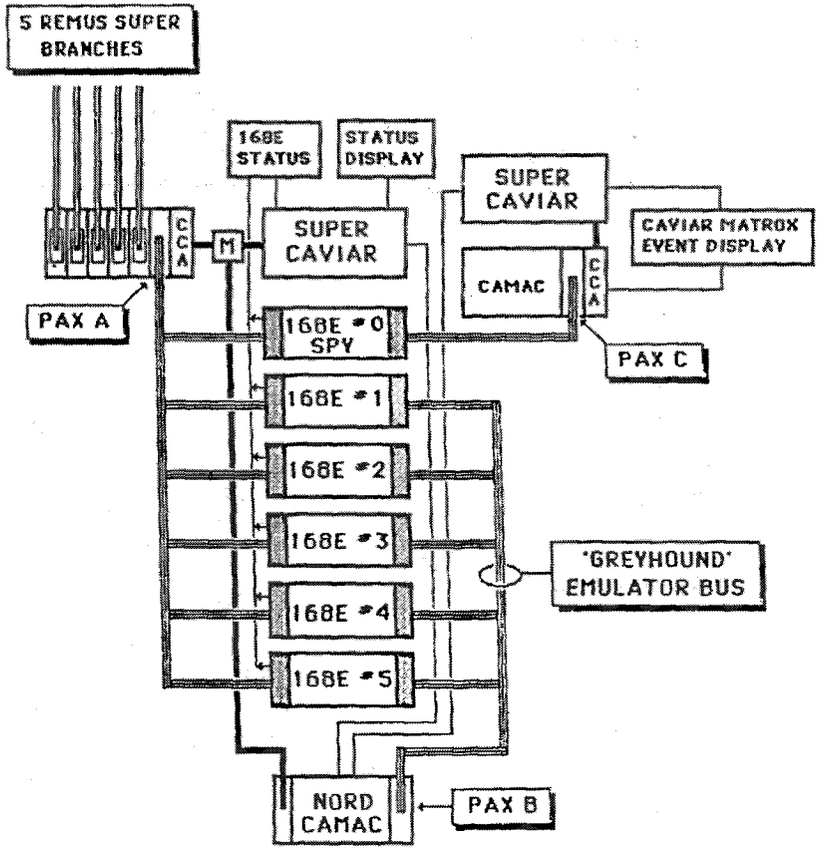
MISSING ET
Total vector energy unbalance

BACKGROUND (≈ 10 ms)

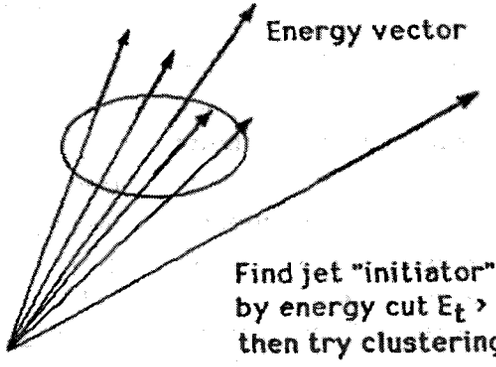
"Pion" identification by threshold cut in the E_{had} and isolation in the electromagnetic cell's space

MUON (≈ 10 (+150) ms)
Muon track reconstruction and track following in a central detector "road". Trigger selected by momentum cut.

168E EVENT FILTER

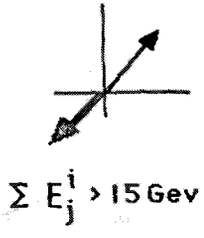


JET TRIGGER



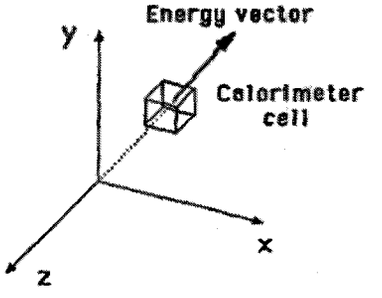
MISSING ENERGY

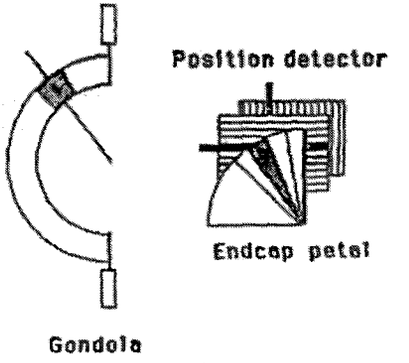
Energy vector sum cut and background test.

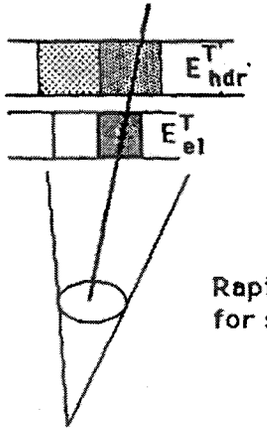


ELECTRON

Calorimeter energy cell reconstruction using calibration constants.
Electron selected by energy threshold cut and space isolation test.





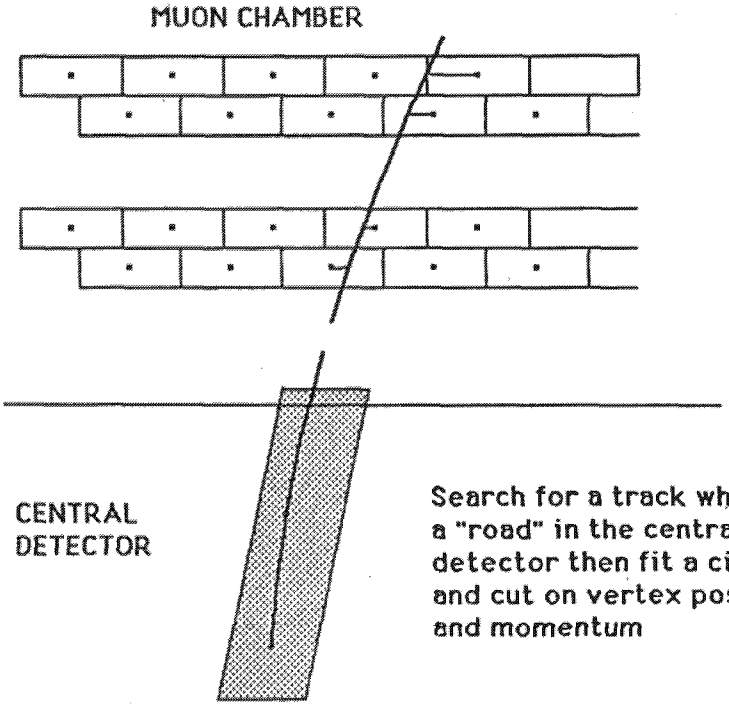


E^T_{el} cut > 6 Gev
E^T_{hdr} < 1 Gev
Isolation

Rapidity-azimuth cone (0.4) for space isolation cut

168E MUON TRIGGER

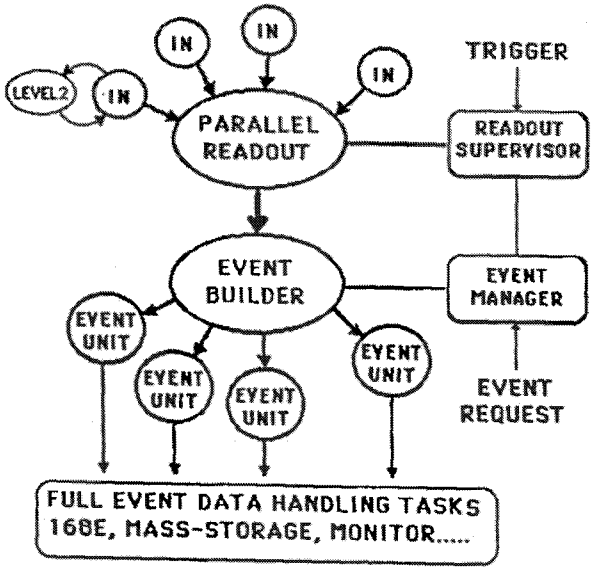
Reconstruct the muon track fro the muon chamber data



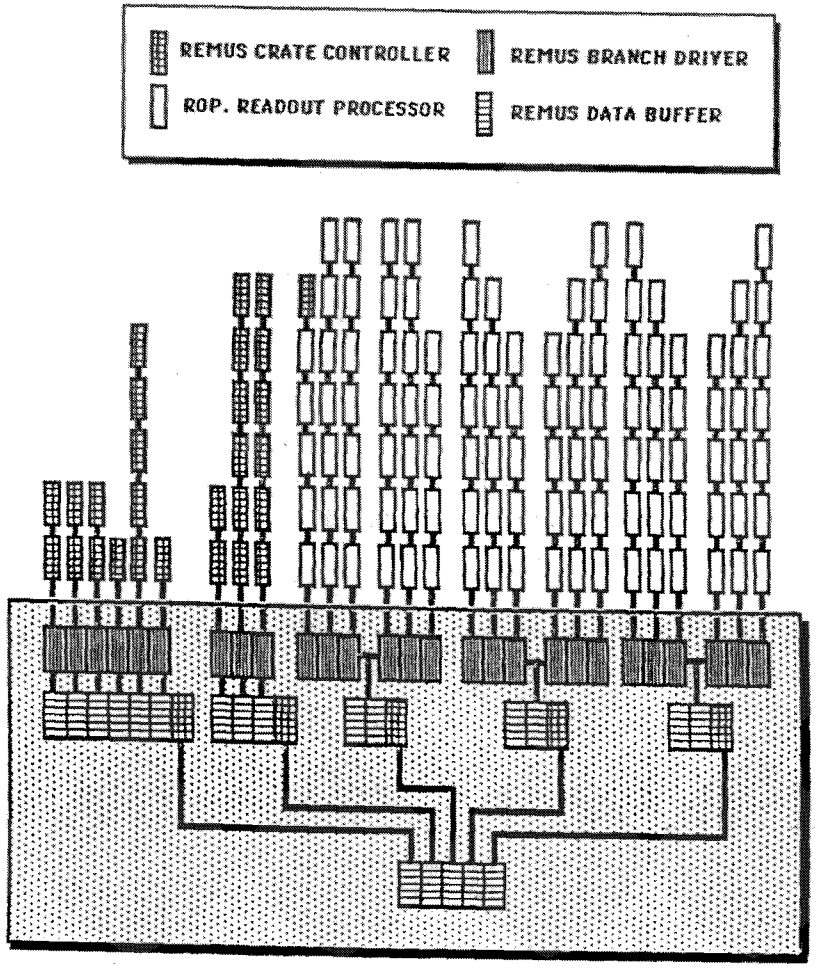
NEW STRUCTURE

DATA ACQUISITION BLOCK STRUCTURE
VME/VMX BUS

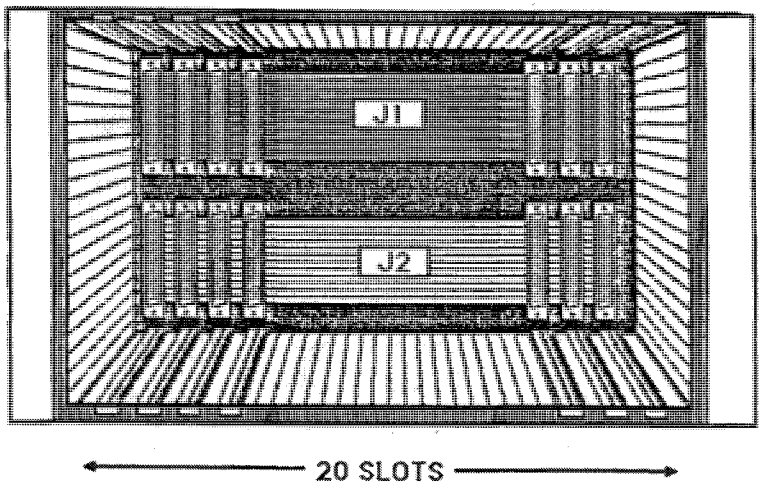
DATA ACQUISITION STRUCTURE



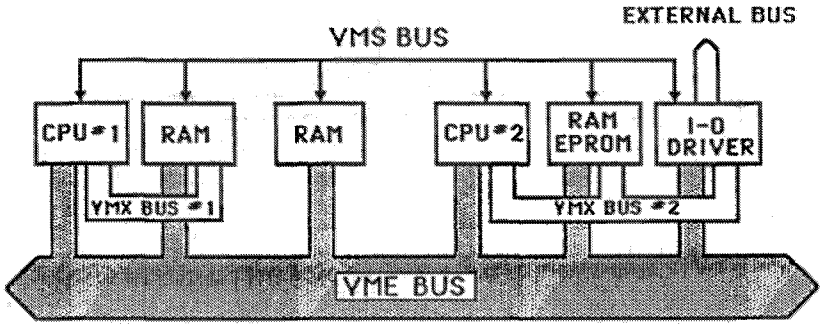
REMUS READOUT SYSTEM



VME CRATE

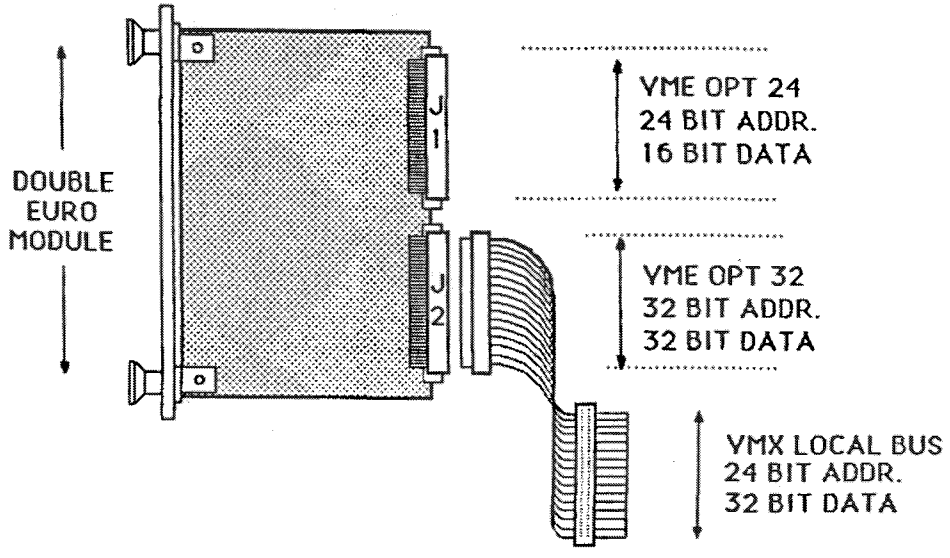


VME/VMX/VMS BUS ARCHITECTURE



VME BUS FOR GLOBAL CONNECTION
VMX BUS FOR EXECUTION AND DATA TRAFFIC
VMS SERIAL BUS FOR SYSTEM CONTROL

VME MODULE



VME BUS

- 32 BIT DATA PATH
- 4 GIGABYTE ADDRESSING
- MULTIPROCESSOR, MULTILEVEL SYSTEM
- INDUSTRY STANDARD
- 20 MB/SEC BANDWIDTH

VMX BUS

- 32 BIT DATA PATH
- 16 MEGABYTE ADDRESSING
- RIBBON CABLING BETWEEN SLOTS

VME COMPONENTS		
CPUA1	60	Data Sud
DUAL PORT MEMORY	60	Data Sud
EPROM MODULE	10	Force
CRATE INTERCONNECT	18	Helsinki
INTERUPPT GEN.	2	Saclay
PARALLEL IO	2	Lapp
168E DATA LINK	2	CERN/ED-DD
REMUS BRANCH DRIVER	40	CERN/EI-EP
VME CRATES	18	CERN/DD
3081E VME INTERFACE	12	CERN/ED-DD
VIDEO DISK SCSI INTERFACE	2	Data Sud

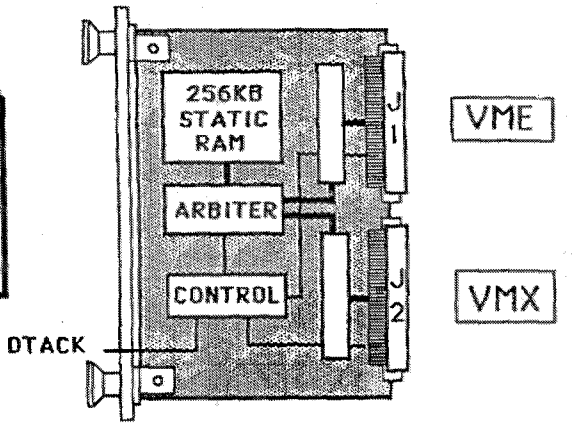
VME MODULES

M68010 CPU
VME/VMX DUAL PORT MEMORY
VME CRATE INTERCONNECT
VME INTERRUPT GENERATOR
VME/VMX DMA CONTROLLER
VME PARALLEL INPUT/OUTPUT
VME/VMX REMUS BRANCH DRIVER
3081E VME INTERFACE
GENERAL PURPOSE PERIPHERAL CONTROLLER
SCSI BUS CONTROLLER (THOMSON VIDEO DISK)

DPRX. VME-VMX DUAL PORT MEMORY (DATA SUD FRANCE)

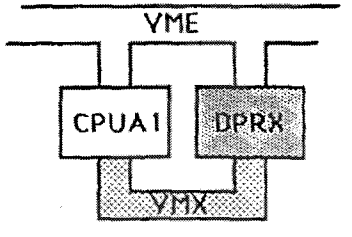
DPRX. STATIC RAM
128KB, 256KB

DPDX. DYNAMIC RAM
512KB, 1MB

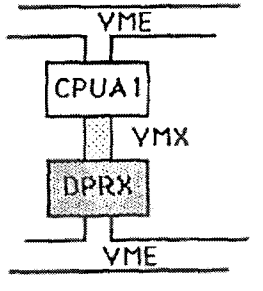


DPRX. DUAL PORT MEMORY
128/256 KByte static memory 400 ns access time Byte/Word/Long word access VME/VMX access Broadcast Write mode
APPLICATION
Parallel readout buffer, event builder data communication

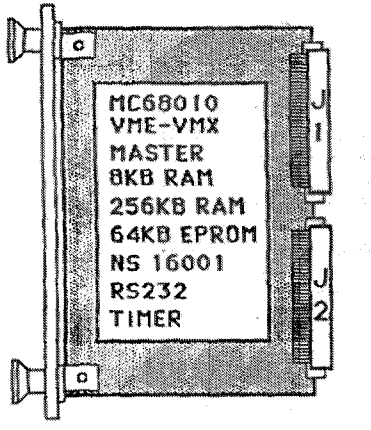
CPU LOCAL MEMORY EXTENSION



VME CRATE COMMUNICATION

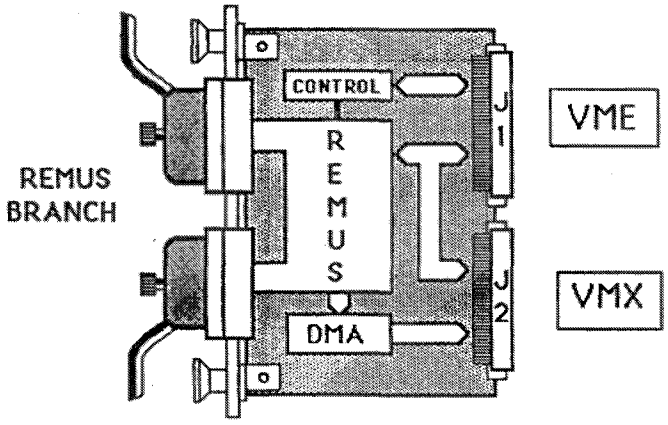


CPUA1. MC68010 VME-VMX PROCESSOR (DATA SUD FRANCE)



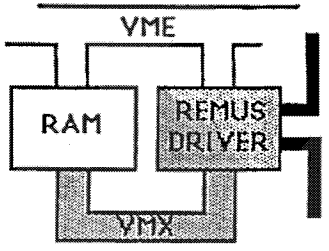
CPUA1
68010 8 MHz 256 KByte Dynamic memory 8 KByte Static RAM (dual port CPU/VME) VME Bus master (arbiter) VMX Bus primary-secondary master NS 16081 floating point processor RS 232(422) serial port Front panel signal I/O Timer. VME interrupter
APPLICATION
Readout supervisor, Readout unit Event builder, Event unit

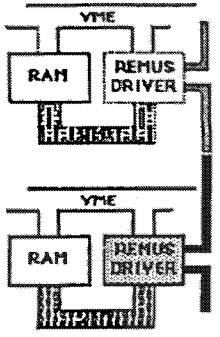
VME-VMX REMUS BRANCH DRIVER (CERN EL/EP)



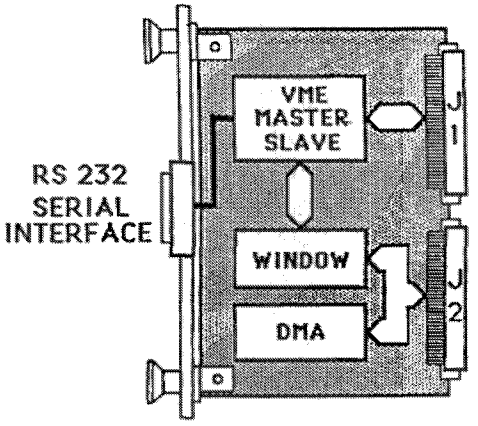
VME/VMX REMUS BRANCH DRIVER

- REMUS BRANCH INTERFACE (READ/WRITE)
- VME CONTROL
- VMX DIRECT MEMORY ACCESS
- REMUS-VME REMUS-VMX VME-VMX



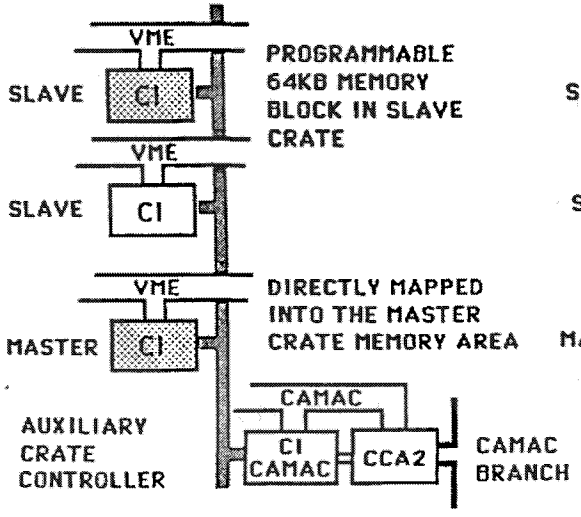


VME CRATE INTERCONNECT (HELSINKI UNIVERSITY)

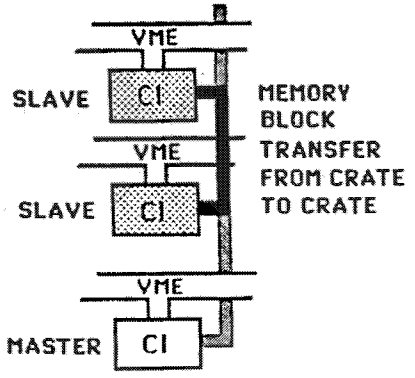


CRATE INTERCONNECT BUS
32 BITS DATA AND ADDRESS

CI WINDOW MODE

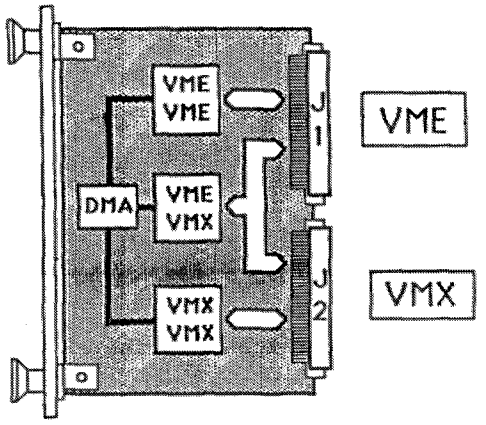


CI DMA MODE (10MB/sec)

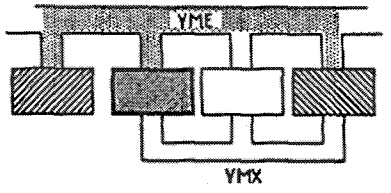


VME(VMX)/VME(VMX) DMA MODULE
(CERN EL/EP)

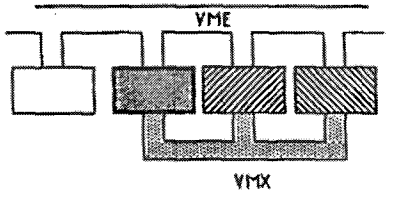
DMA 16 and 32 bit data
Source : VME or VMX
Destination : VME or VMX
Address increment : 0,2,4



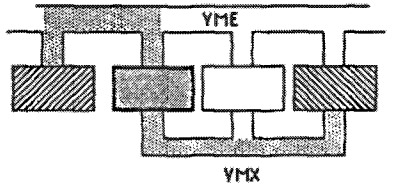
DMA VME-VME



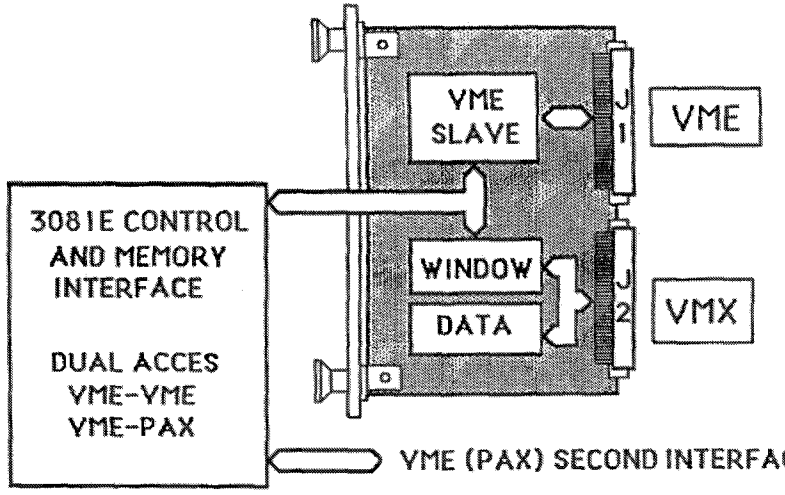
DMA VMX-VMX

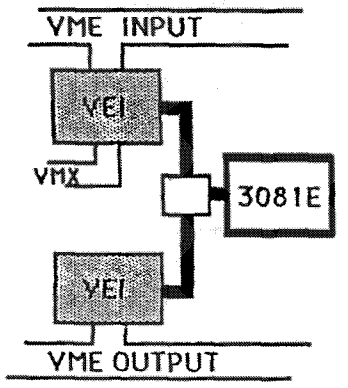


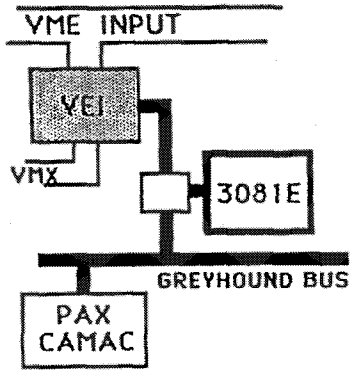
DMA VME-VMX



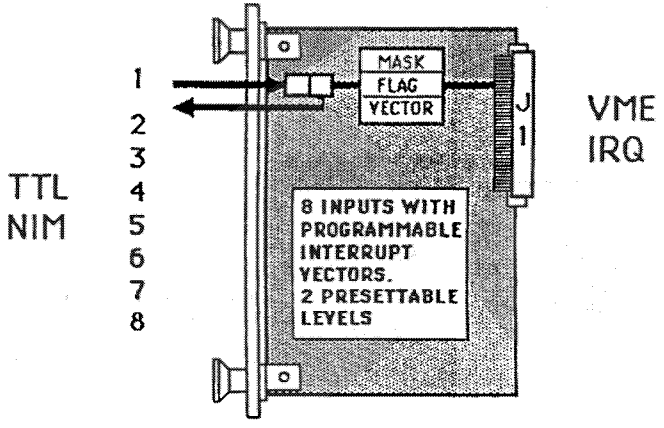
VME 3081E EMULATOR INTERFACE
(CERN ED/DD)







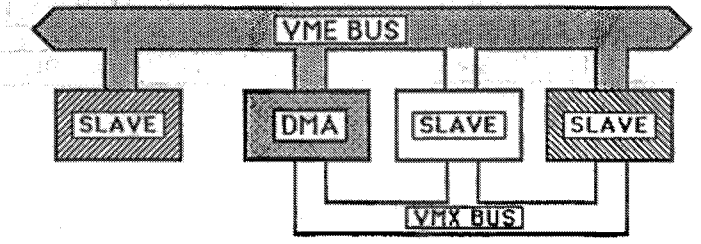
VME INTERRUPT VECTOR GENERATOR (SACLAY)



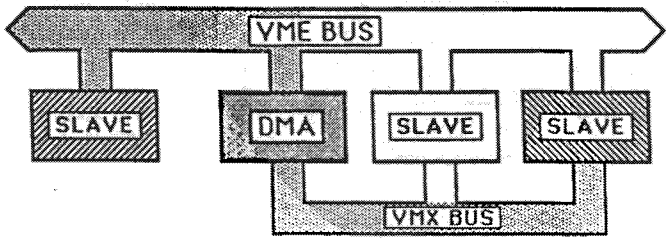
10 MB/sec DMA (CERN EL/EP)

SOURCE ADDRESS (VME-VMX)
DESTINATION ADDRESS (VME-VMX)
0, 2 and 4 Bytes ADDRESS INCREMENT

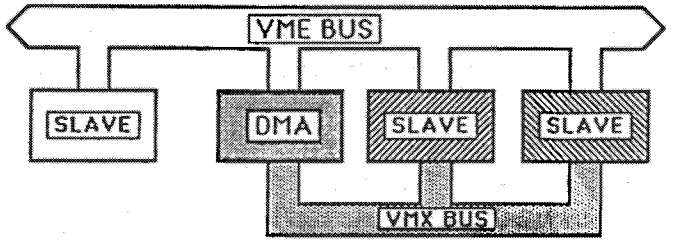
DMA VME↔VME



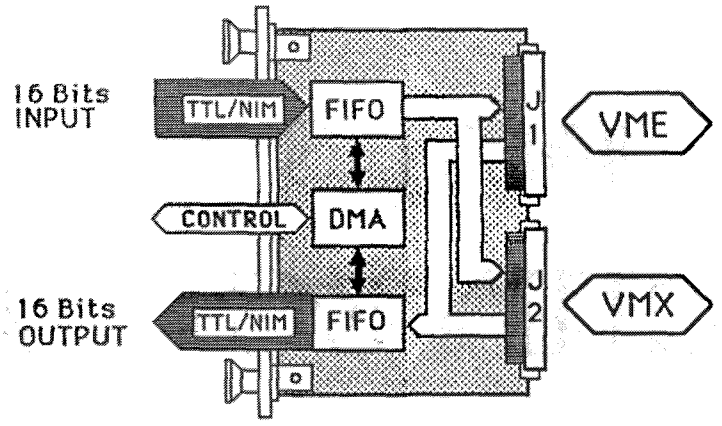
DMA VME↔VMX



DMA VMX↔VMX



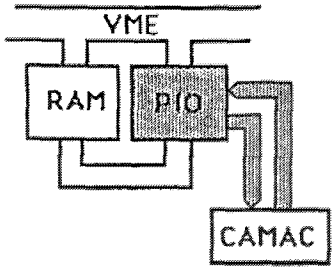
VME PARALLEL INPUT/OUTPUT MODULE (LAPP/ANNECY)



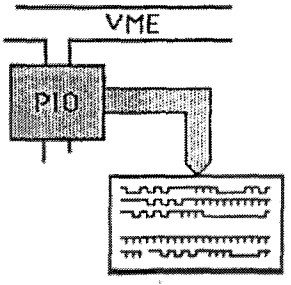
VME/VMX PARALLEL I/O

- 16 Bit INPUT (TTL-NIM)
- 16 Bit OUTPUT (TTl-NIM)
- 512 Words FIFO (200ns)
- VME CONTROL
- VMX SECONDARY MASTER

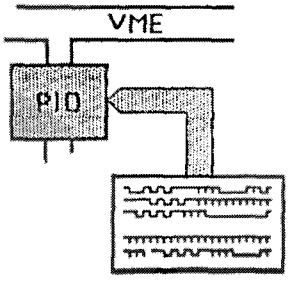
DATA COMMUNICATION



16 Bit SEQUENCER



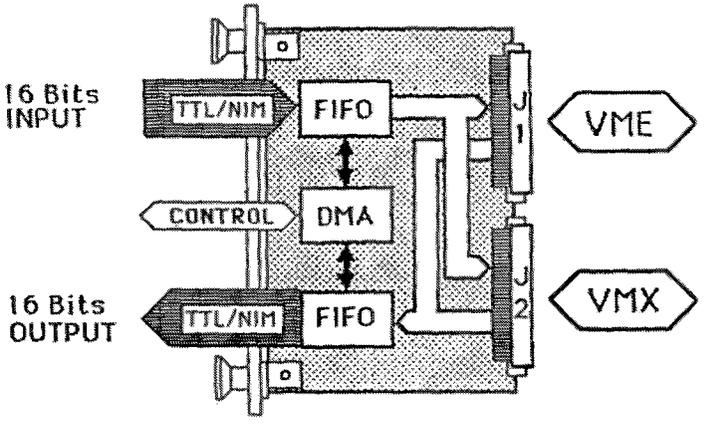
16 Bit STATE ANALYZER



READOUT ARCHITECTURE

SYSTEM BUSSES and MODULES
READOUT SUPERVISOR
EVENT MANAGER
PARALLEL READOUT UNIT
EVENT UNIT

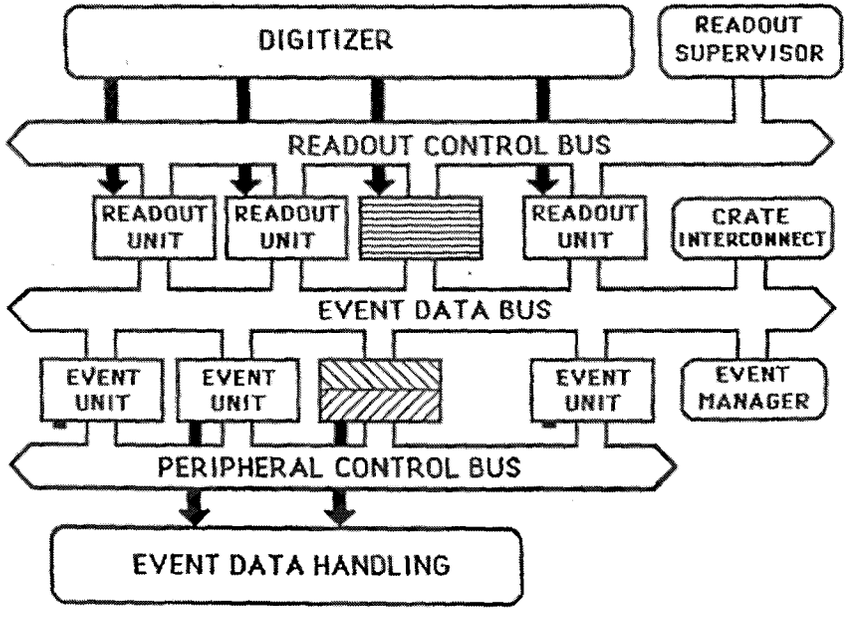
VME PARALLEL INPUT/OUTPUT MODULE (LAPP/ANNECY)



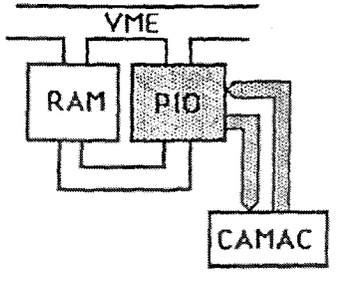
VME/VMX PARALLEL I/O

- 16 Bit INPUT (TTL-NIM)
- 16 Bit OUTPUT (TTI-NIM)
- 512 Words FIFO (200ns)
- VME CONTROL
- VMX SECONDARY MASTER

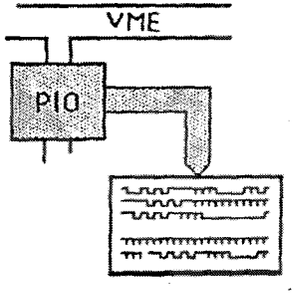
VME READOUT ARCHITECTURE



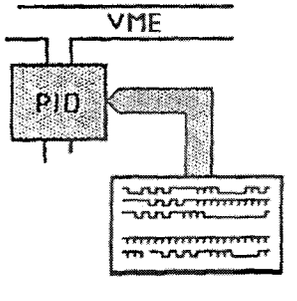
DATA COMMUNICATION



16 Bit SEQUENCER



16 Bit STATE ANALYZER



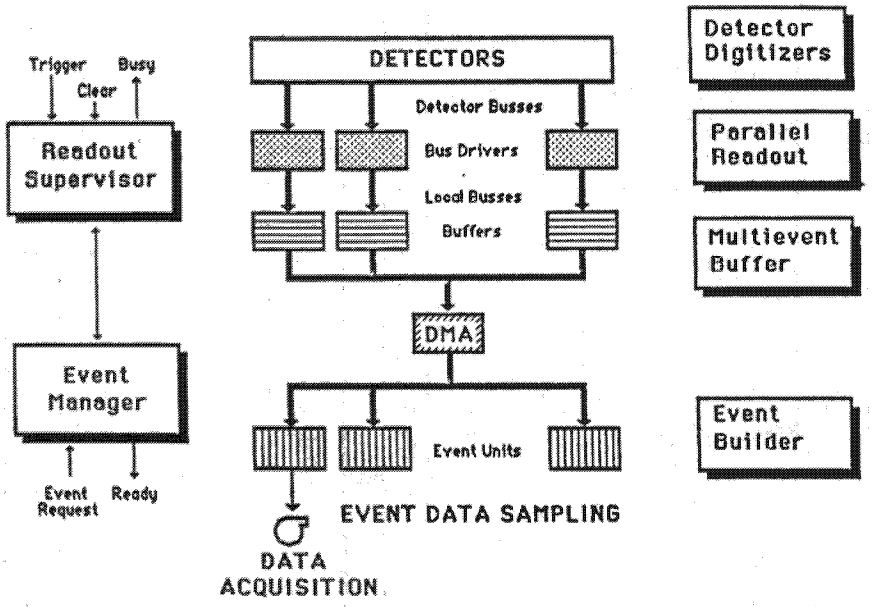
SYSTEM BUSSES

- DIGITIZER BUS
- READOUT CONTROL BUS
- EVENT DATA BUS
- PERIPHERAL CONTROL BUS
- LOCAL BUS

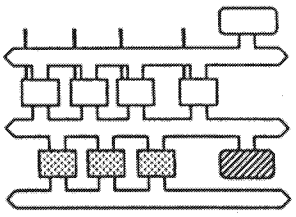
SYSTEM MODULES

- READOUT SUPERVISOR
- READOUT UNIT
- EVENT MANAGER
- EVENT UNIT
- CRATE INTERCONNECT

DATA ACQUISITION BLOCK STRUCTURE



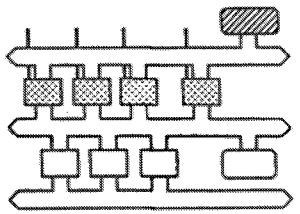
EVENT MANAGER



A VME master processor of the Event Data Bus.

TASKS:
Initialize/Test the Event Units
Process the event requests
Communicate with the Readout Supervisor
Build one event data into event unit buffer
Update multievent buffer directory
Control the Readout Supervisor

READOUT SUPERVISOR

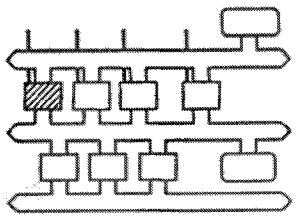


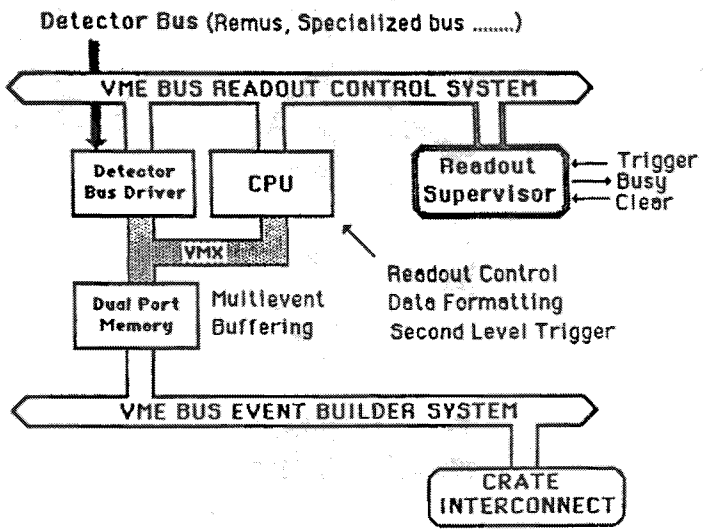
A VME master processor of the Readout Control Bus.

TASKS:
Initialize/Test Readout Units
Process Trigger signal (Read,Busy,Clear)
Start and monitor readout operations
Keep the multi event buffer directory
Communicate with event manager

PARALLEL READOUT UNIT

THE PARALLEL READOUT UNIT PERFORMES THE FUNCTIONS OF CONTROL AND READOUT OF A DETECTOR BUS AND THE DATA STORAGE INTO A MULTI EVENT DATA BUFFER
THE UNIT CAN HAVE LOCAL CPU POWER FOR DATA REFORMATTING AND SECOND LEVEL TRIGGERING

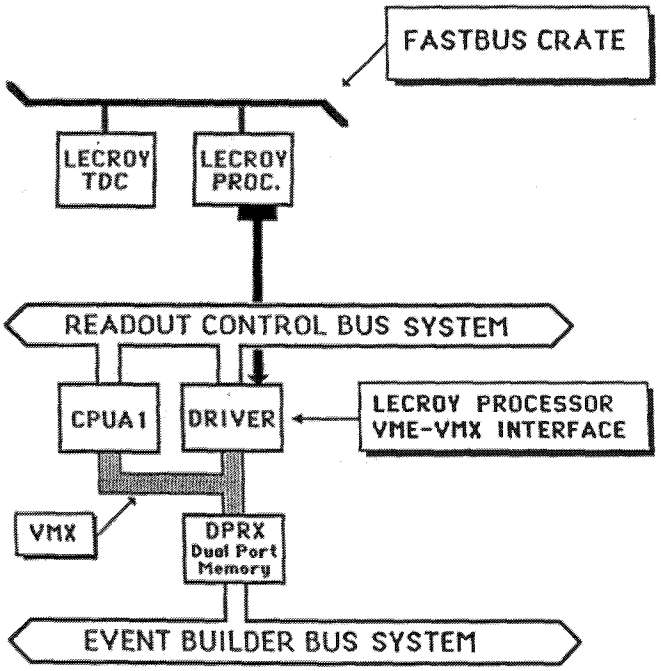




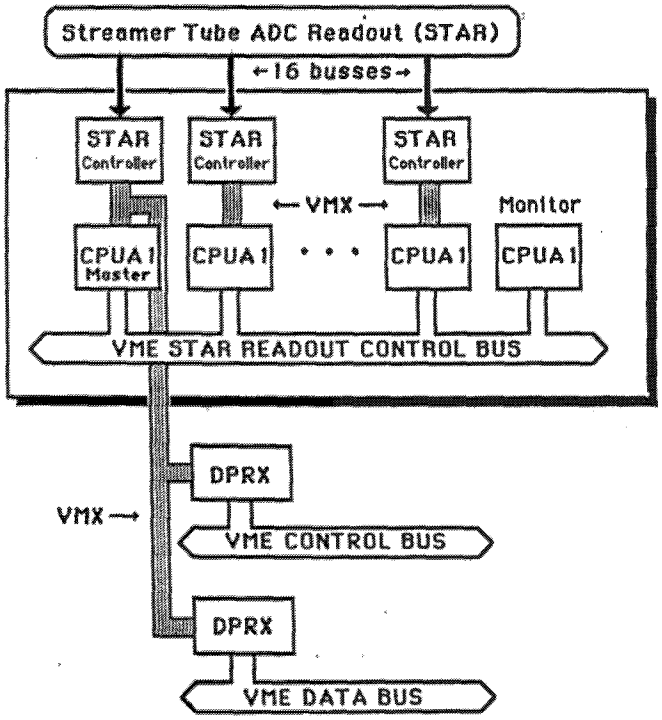
PARALLEL READOUT UNIT

REMUS READOUT
IAROCCI VME READOUT
FASTBUS VME READOUT

FASTBUS READOUT UNIT
(MICROVERTEX LECROY TDC SYSTEM)



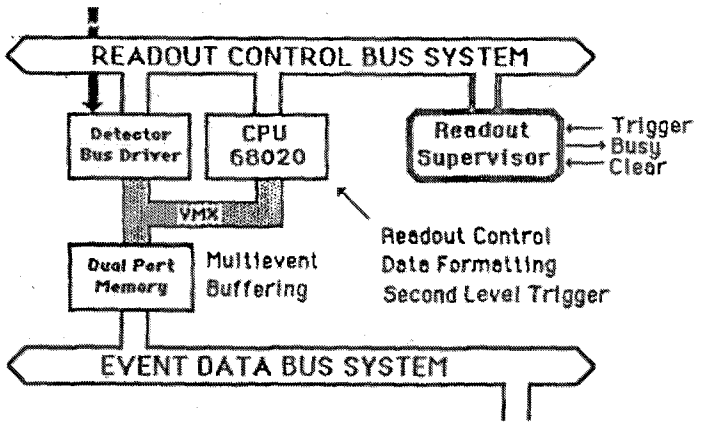
STREAMER TUBE READOUT UNIT



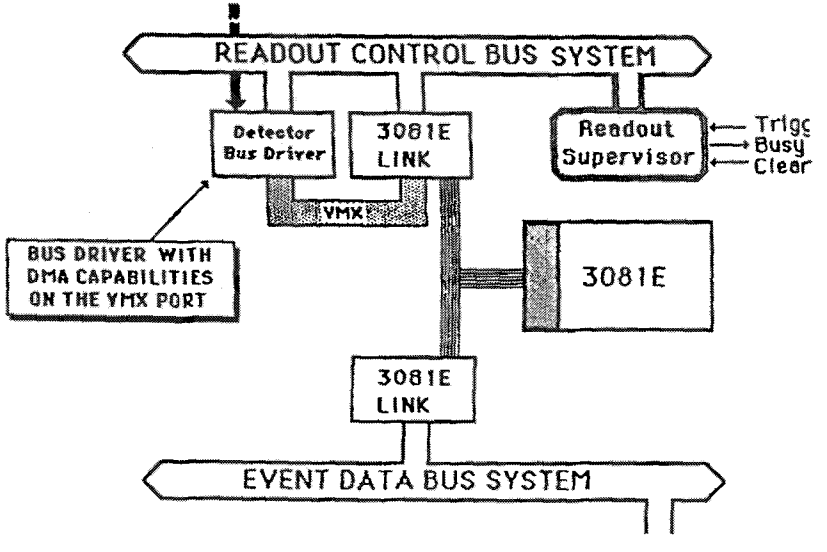
EVENT UNIT

DATA SAMPLING UNIT
168E EVENT UNIT
3081E VME SYSTEM

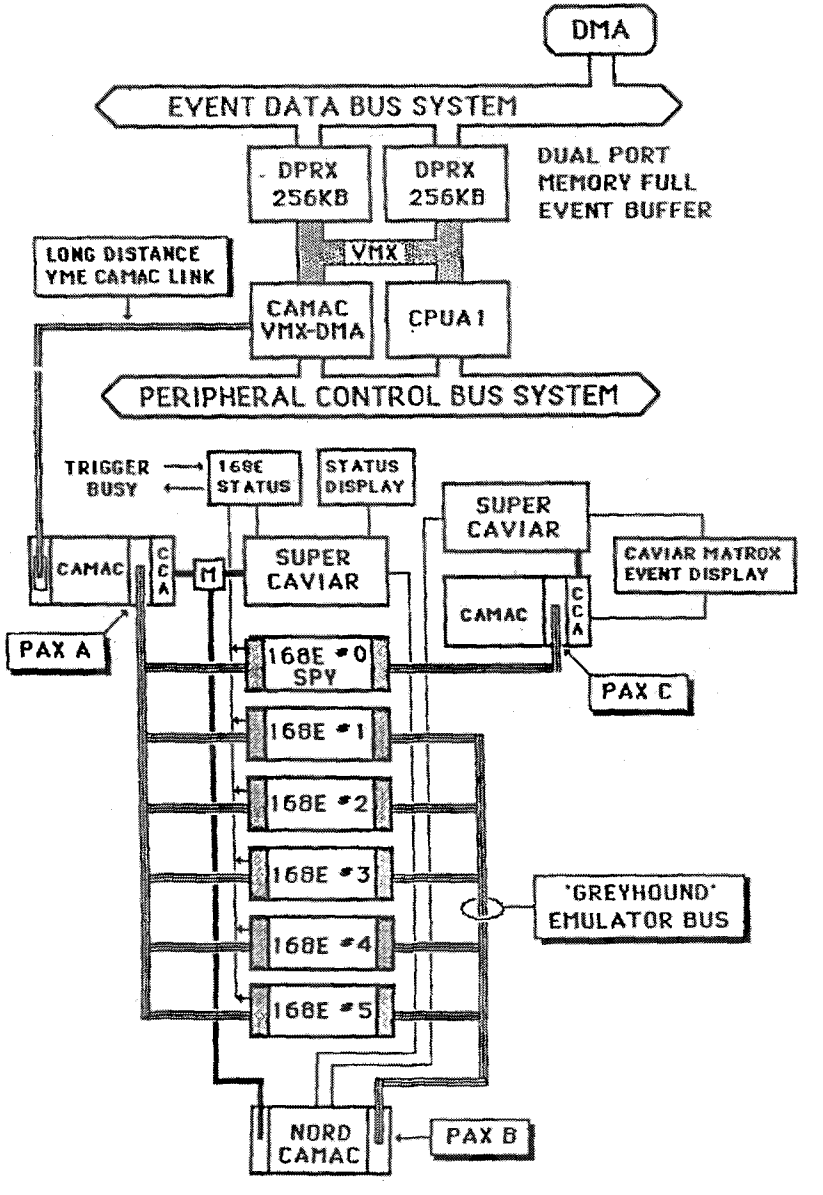
68020 SECOND LEVEL TRIGGER PARALLEL READOUT UNIT



3081E SECOND LEVEL TRIGGER PARALLEL READOUT UNIT

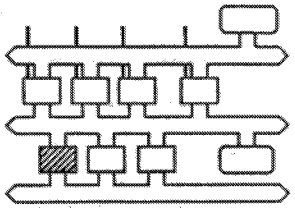


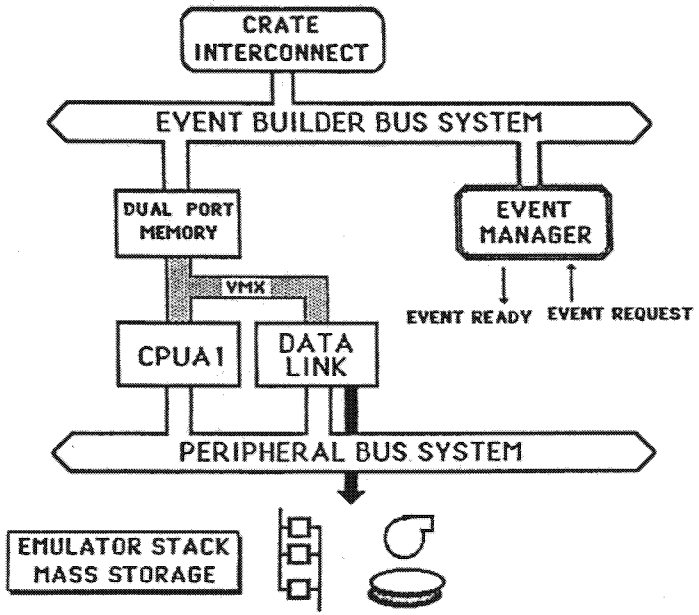
168E EVENT UNIT



EVENT UNIT

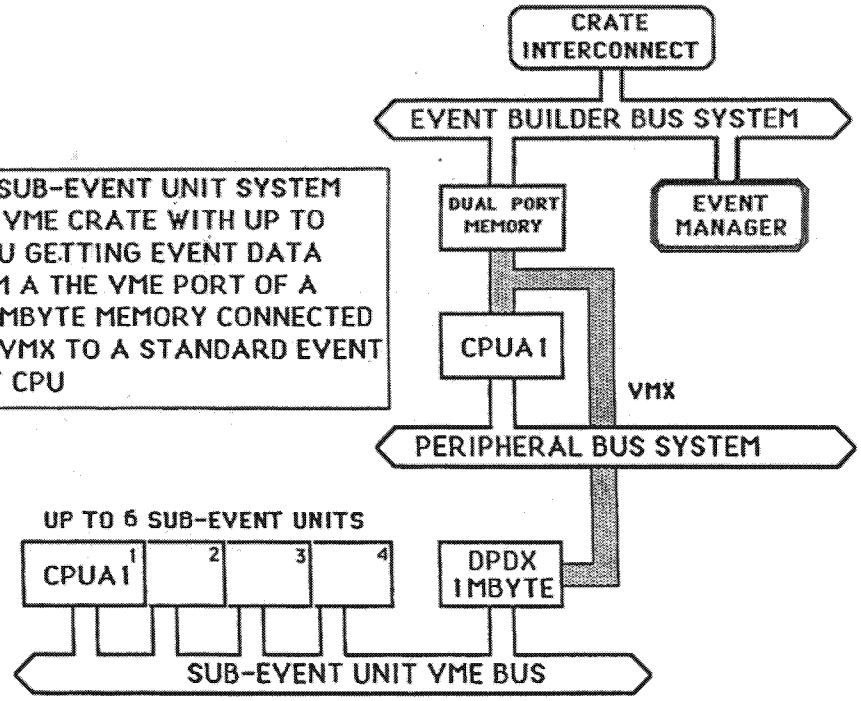
THE EVENT UNIT IS A COMPUTER SYSTEM ABLE TO COLLECT A FULL EVENT DATA BLOCK.
THE EVENT IS BUILT INTO THE UNIT'S MEMORY BY THE EVENT MANAGER.
THE EVENT UNIT CONNECTED TO THE MASS-STORAGE SYSTEM RUNS THE PRIMARY TASK



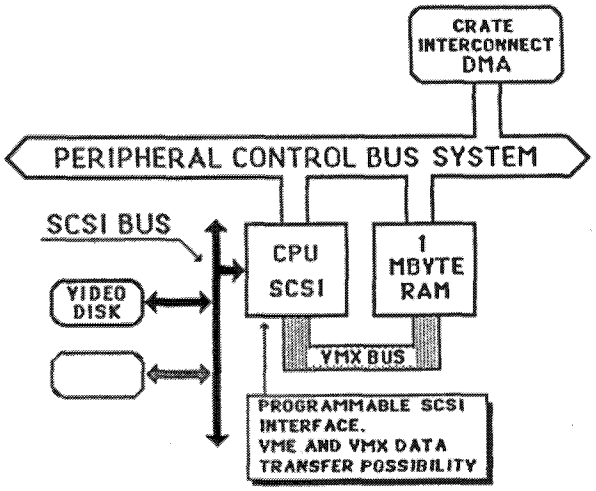


SUB-EVENT UNIT SYSTEM

THE SUB-EVENT UNIT SYSTEM IS A VME CRATE WITH UP TO 6 CPU GETTING EVENT DATA FROM A THE VME PORT OF A ONE MBYTE MEMORY CONNECTED VIA VMX TO A STANDARD EVENT UNIT CPU



VME VIDEO DISK (THOMSON G1001)



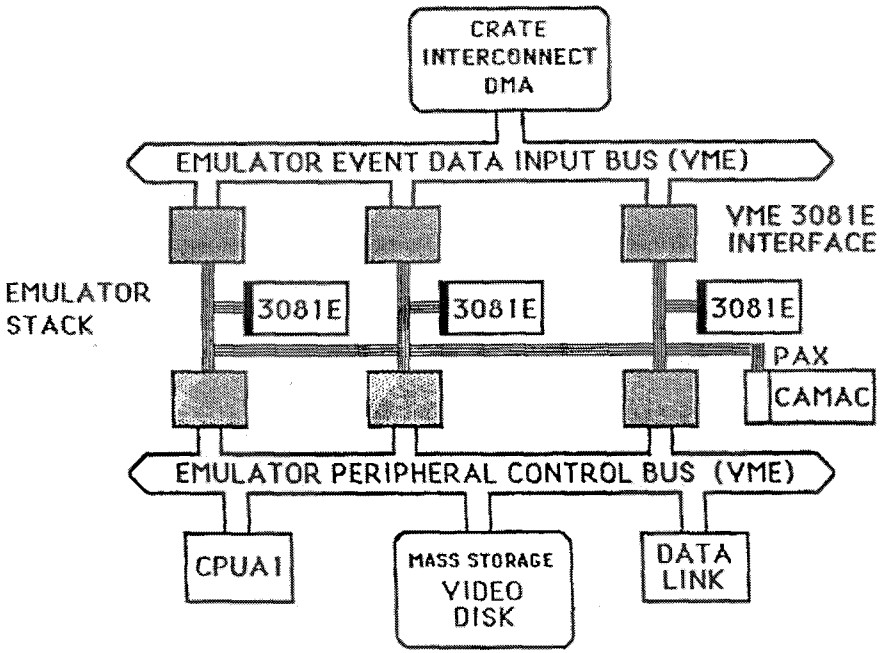
THOMSON G1001 VIDEO DISK

CAPACITY 1 GBYTE (8 6250 BPI TAPES)
SPEED 4 MBIT/SEC (6250 125 IPS MTU)
DISK CONTROLLER WITH ERROR CORRECTION
UP TO 8 DRIVES ON A SINGLE CONTROLLER
SCSI INTERFACE (Small Computer System Interface).

VME LAYOUT

VME CRATE LAYOUT
VME LOGICAL LAYOUT
SYSTEM BUSSES
SYSTEM MODULES

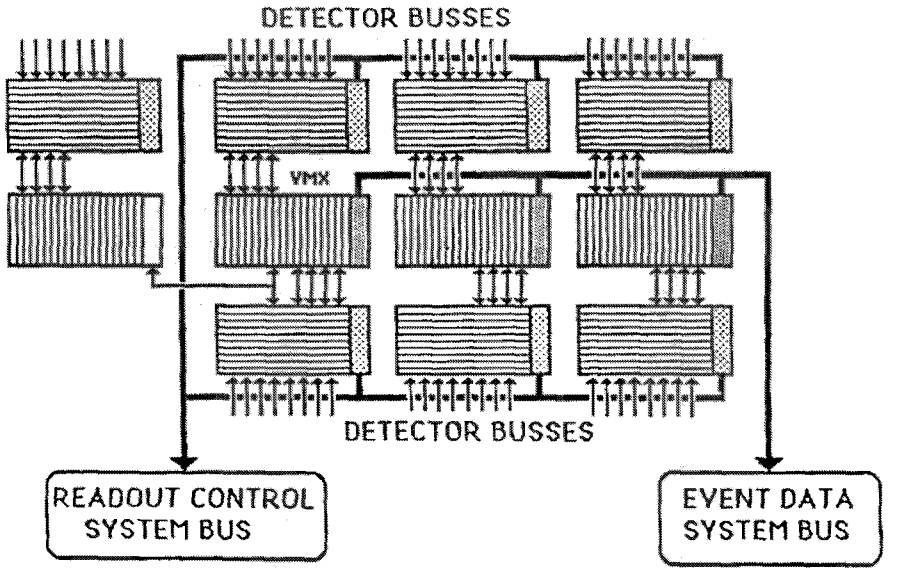
3081E STACK VME EVENT UNIT (ON-OFF LINE MAIN FRAME)

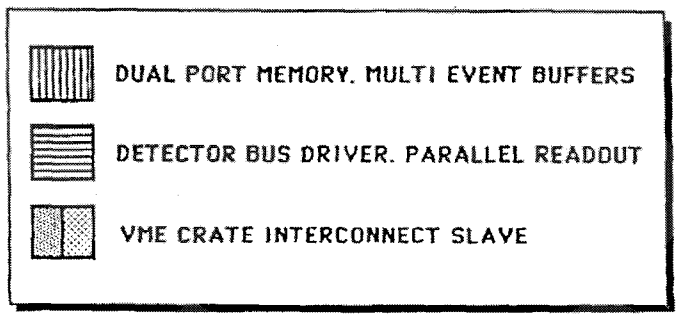


VME 3081E INTERFACE FEATURES

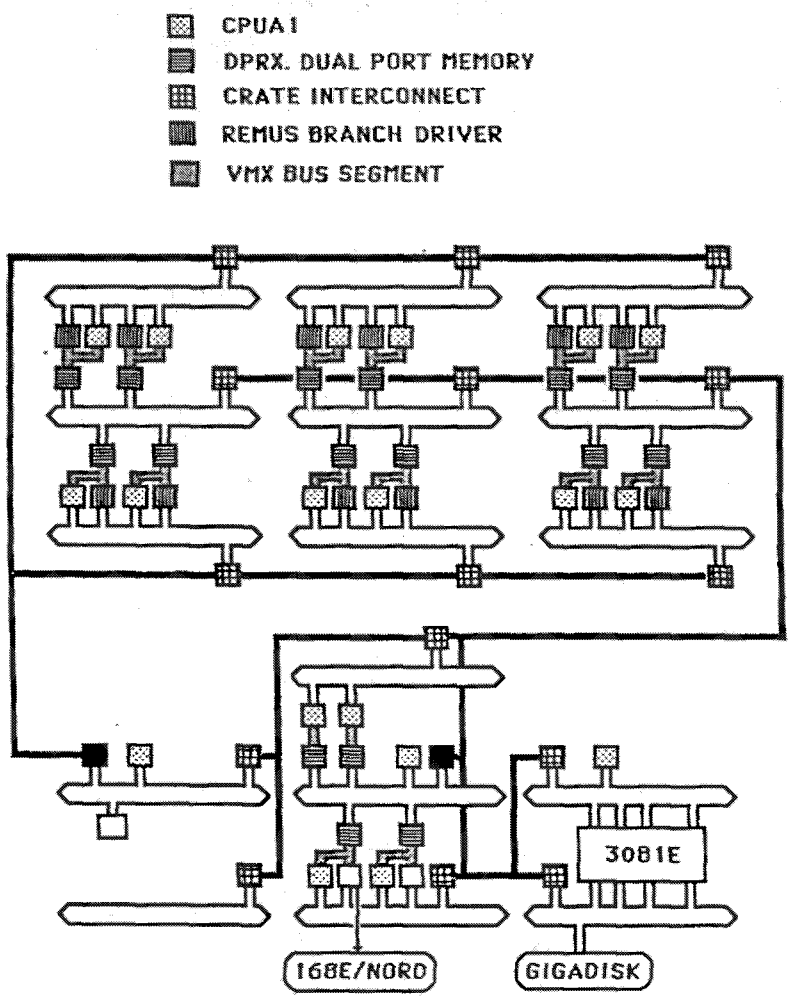
EMULATOR CONTROL
3081E MEMORY DIRECT MEMORY ACCESS
3081E MEMORY WINDOW ACCESS BY BLOCK OF
64K BYTES MAPPED INTO VME ADDRESS SPACE

VME PARALLEL READOUT PHYSICAL LAYOUT

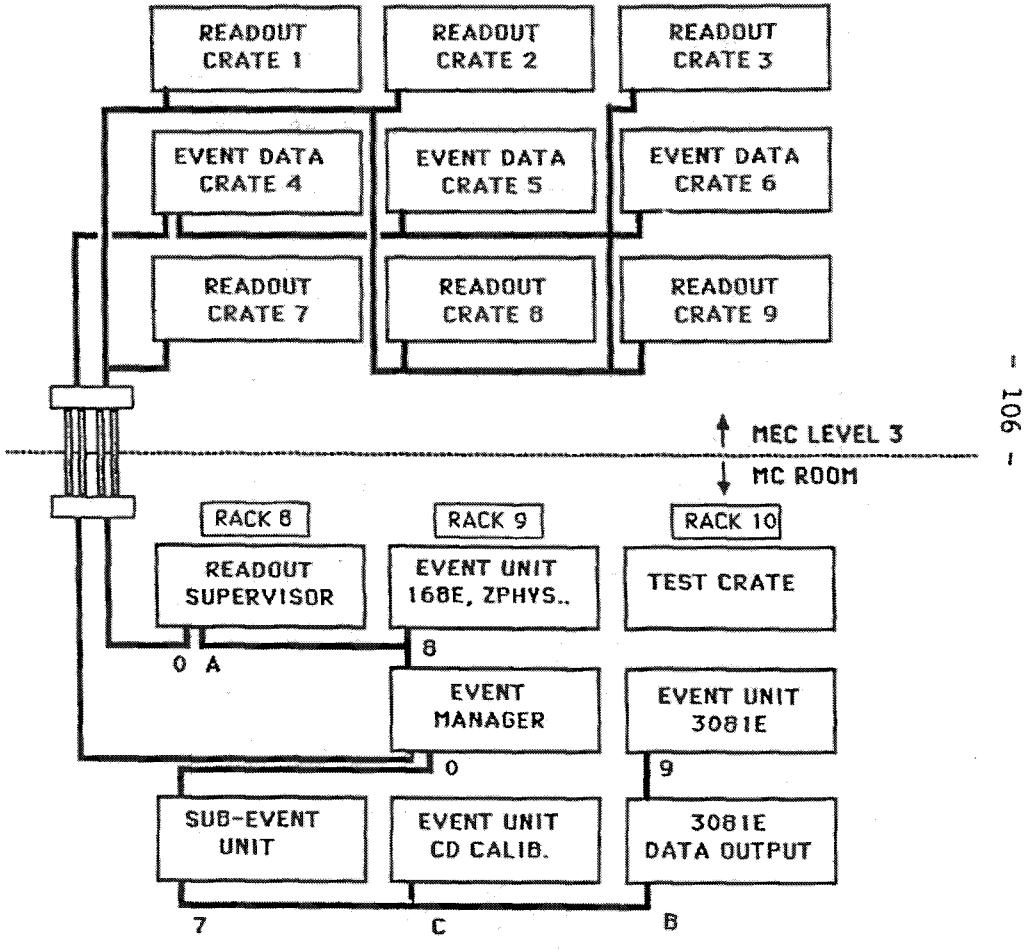




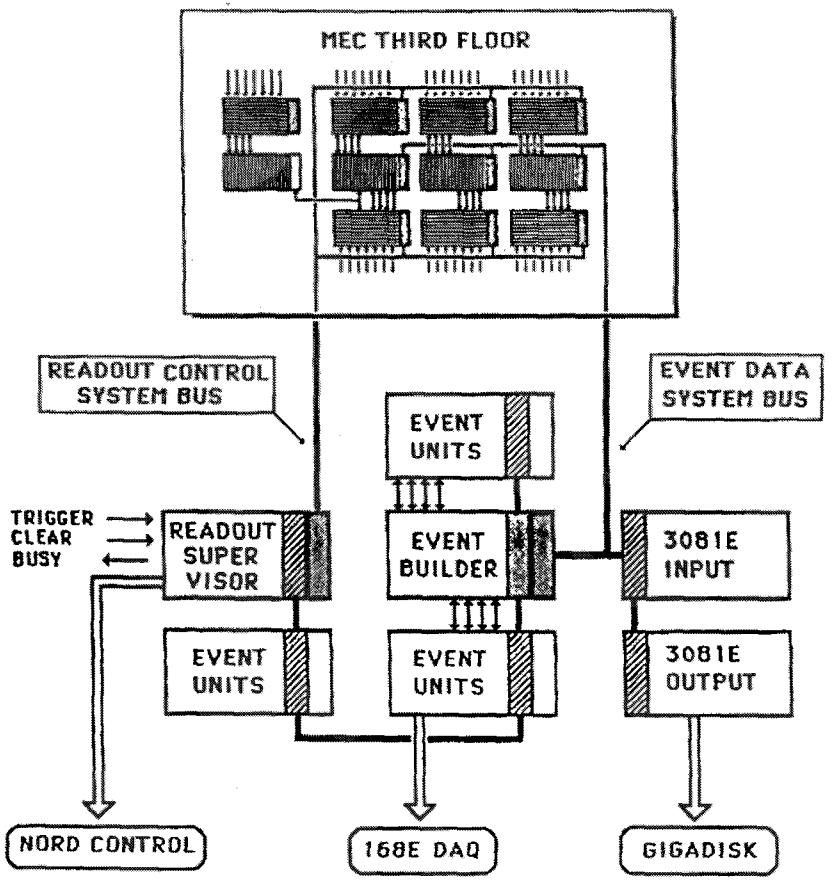
VME READOUT LOGICAL LAYOUT



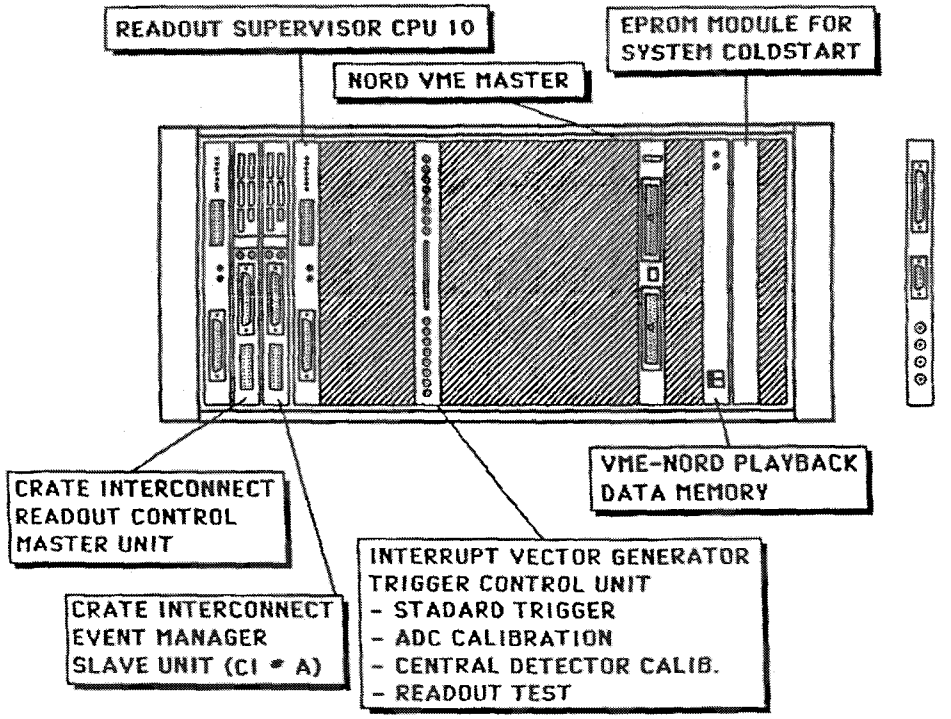
CRATE INTERCONNECT LAYOUT



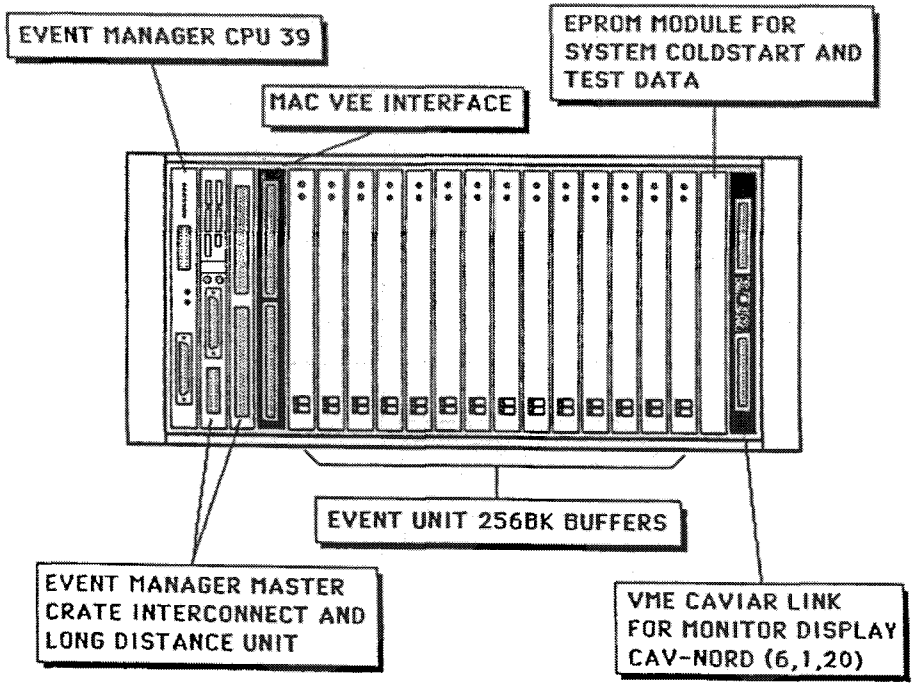
VME CRATE LAYOUT



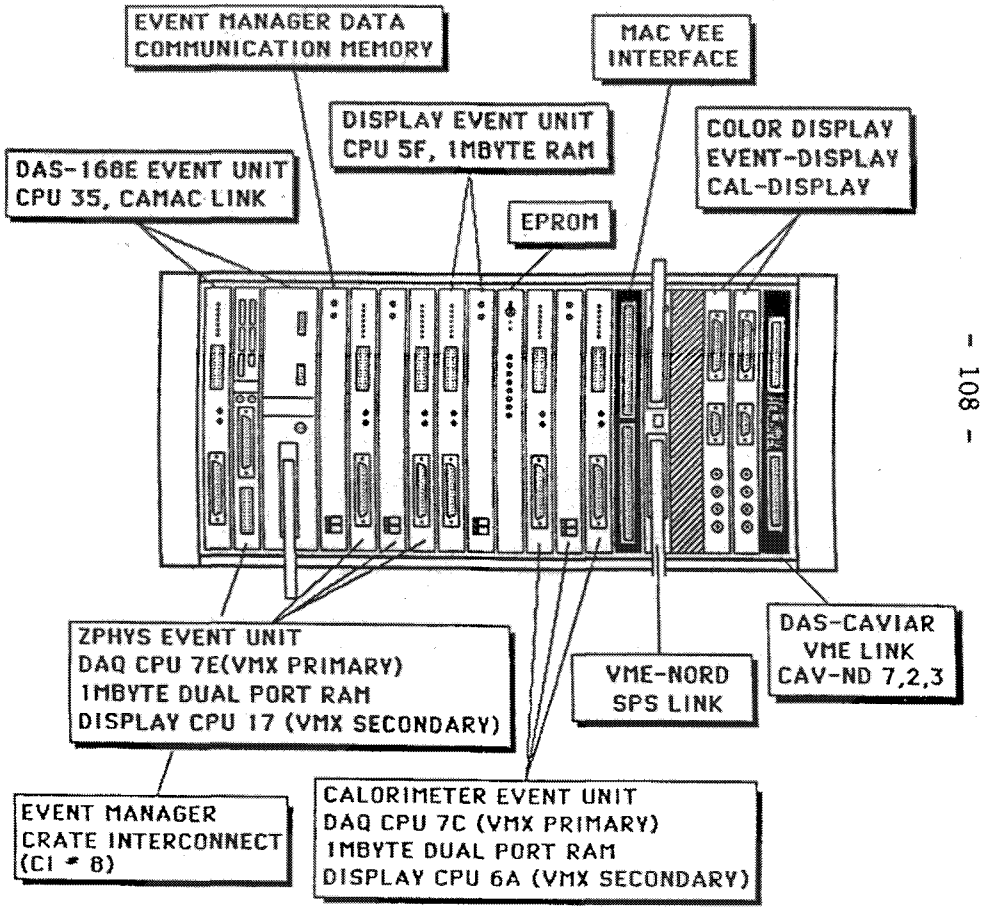
READOUT SUPERVISOR VME CRATE A

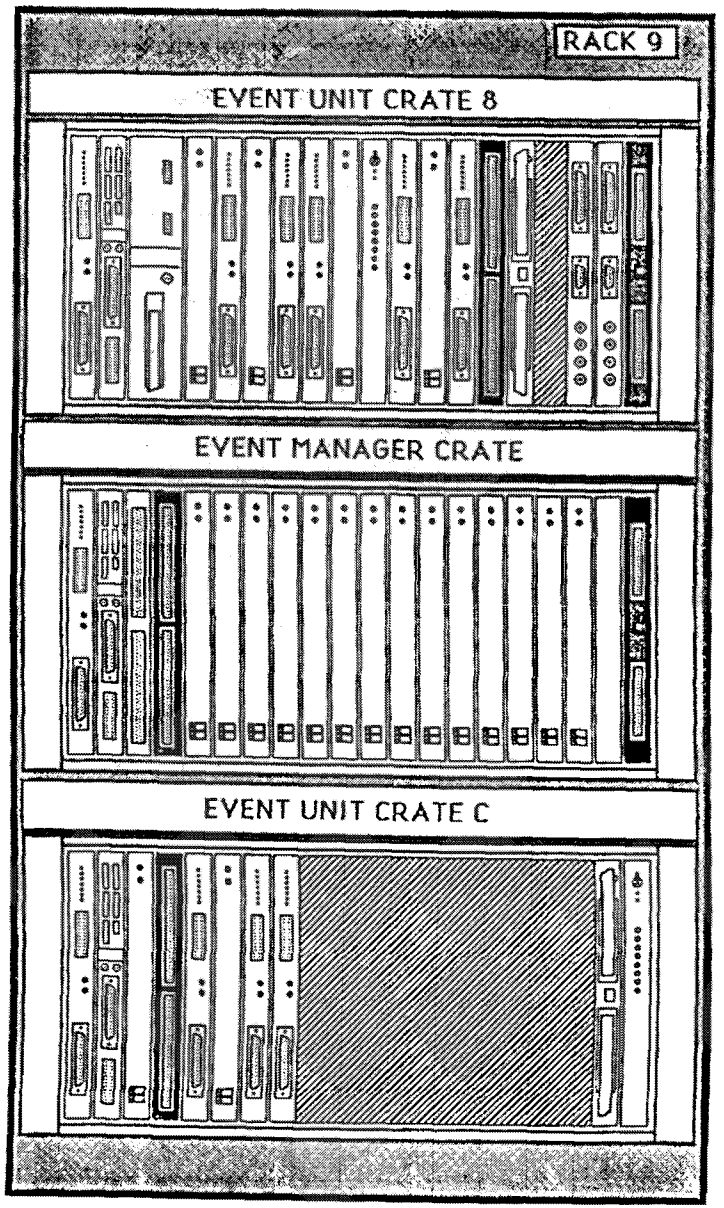


EVENT MANAGER VME CRATE 0

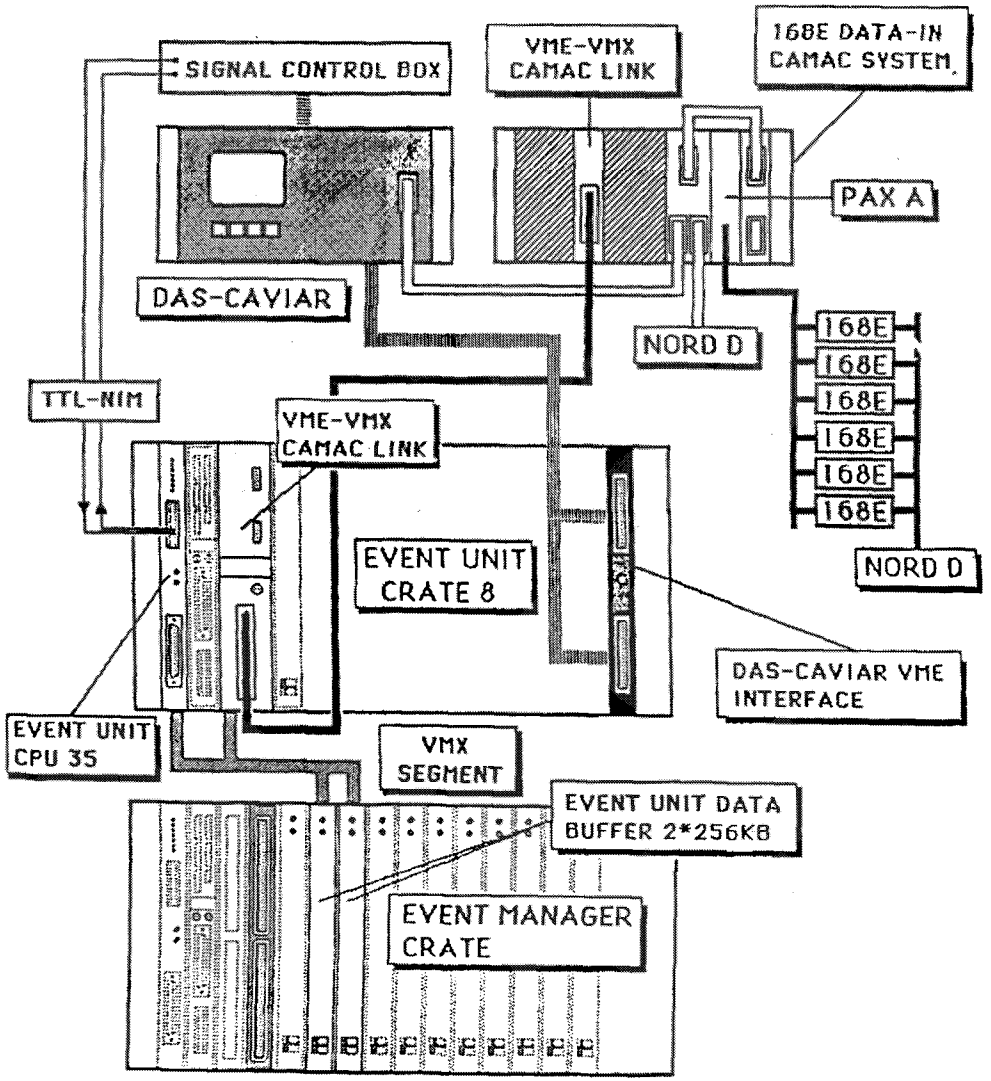


EVENT UNIT VME CRATE 8

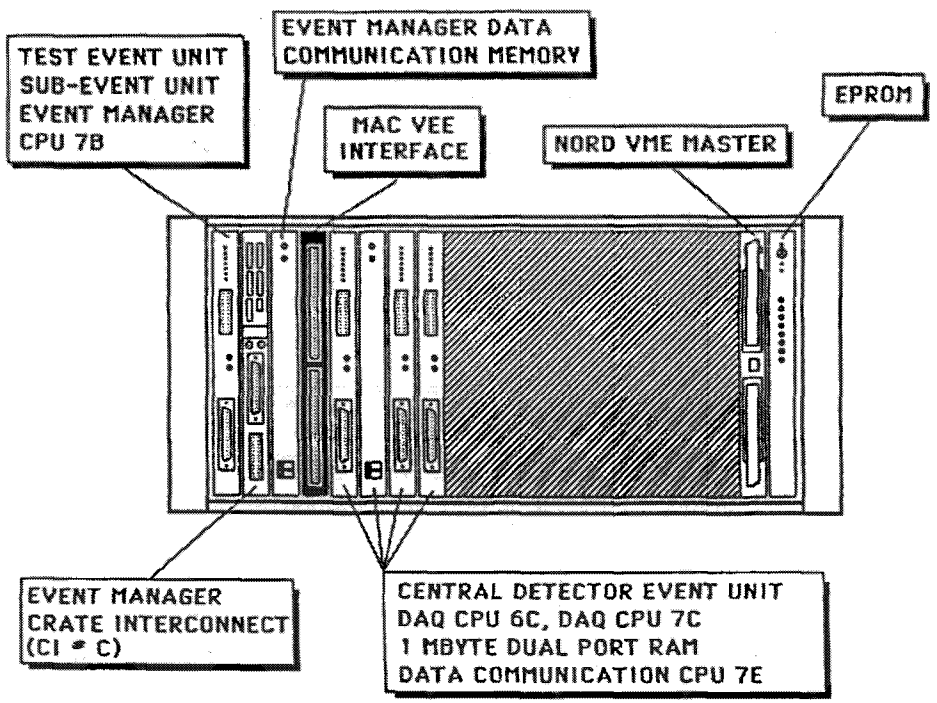




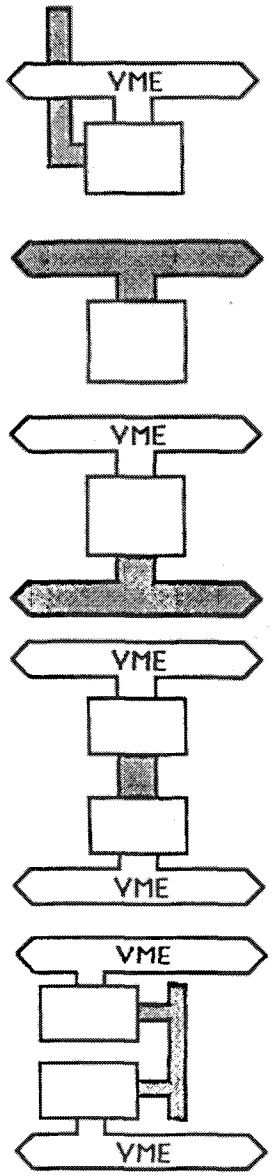
DAS-168 EVENT UNIT BLOCK DIAGRAM



EVENT UNIT VME CRATE C



VME SYSTEM BUSSES



DETECTOR BUS. DIGITIZER READOUT

The BUS connects the digitizer modules to the VME drivers such as REMUS, STAR (streamer tube) LeCroy FAST BUS 1892 processor

READOUT/PERIPHERAL CONTROL BUS

The BUS initializes and controls all the readout driver units and peripheral devices

EVENT DATA BUS

The BUS connects the parallel readout memories together, and to the event builder ones.

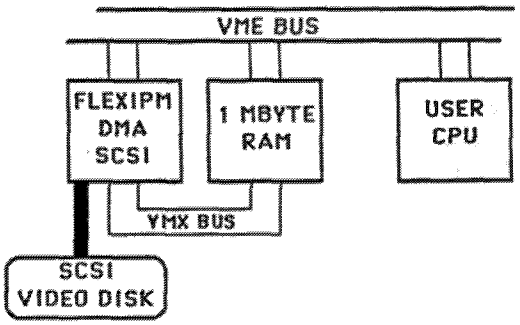
LOCAL BUS

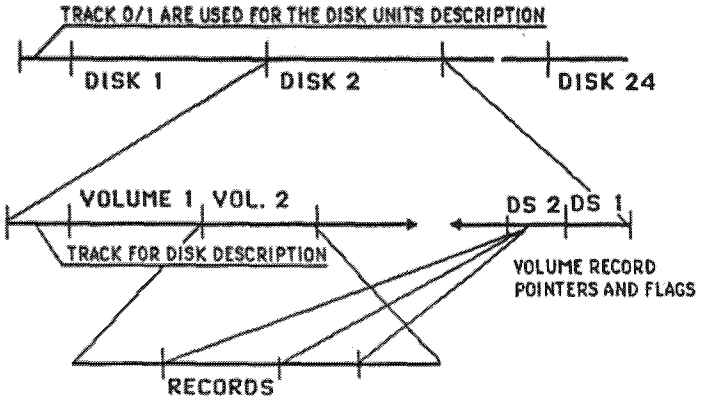
The BUSses connect the detector bus driver to the parallel readout dual port memory and the event builder event memory to the event unit CPU

CRATE INTERCONNECT BUS (CI)

The BUS connects together up to 16 VME crates, allowing memory mapping and DMA transfer.

VIDEO DISK VME LAYOUT AND DATA FORMAT

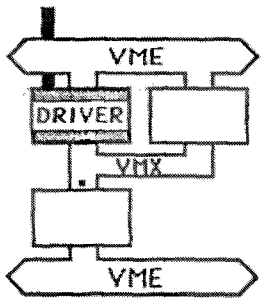




TOOLS

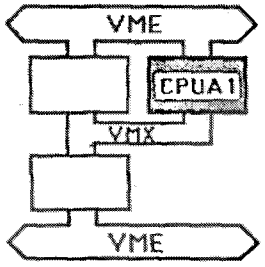
M68OXX SOFTWARE DEVELOPMENT
VME-MAIN COMPUTER DATA COMMUNICATION
VME-PERSONAL COMPUTER APPLICATION

MODULES



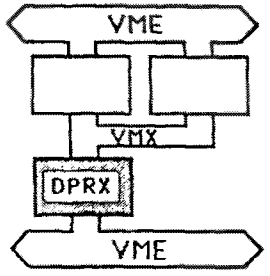
DETECTOR BUS DRIVER

The module interfaces a detector BUS either to VME or VMX parallel readout system



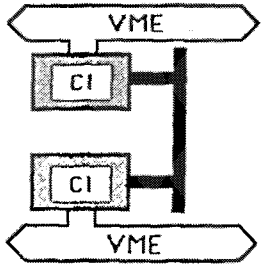
CPU MODULE (CPUA1 68010)

These modules perform the functions of readout controller, readout supervisor, event builder and event unit



DUAL PORT MEMORY

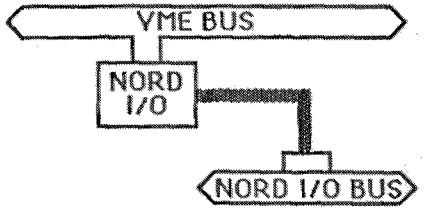
The module is used either as multi event buffer for parallel readout or full event data buffer or for data communication between VME crates



CRATE INTERCONNECT

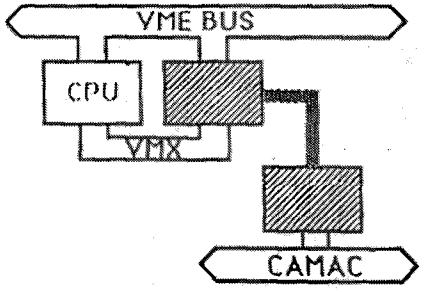
The module connects VME crates by means of a CI BUS allowing window and DMA access modes

VME-MAIN COMPUTER DATA COMMUNICATION



VME BUS MASTER MODULE DRIVEN BY NORD I/O (SPS/LEP)

NORD IOX instructions set the address, the data direction and read or write 16 bit data



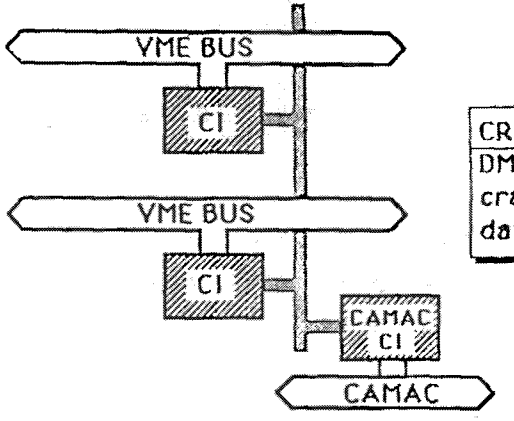
VMX-CAMAC LINK (CERN ED/DD)

Module control via VME access. VMX secondary master with a DMA channel. (4MB/sec speed)

The CAMAC module emulates a Remus branch driver

VME/VMX 16 BIT PIO (LAPP)

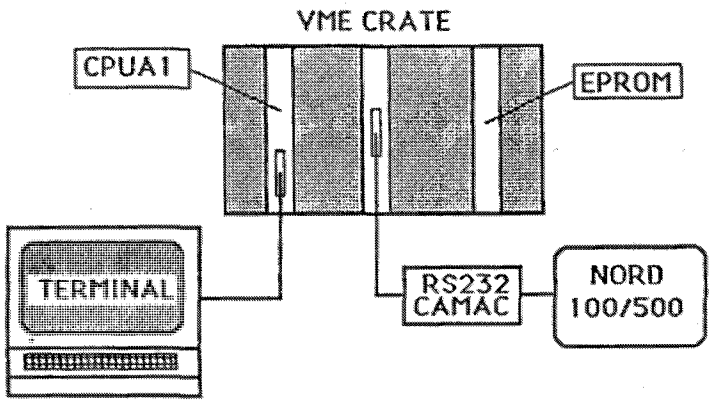
Used with a CAMAC parallel I/O register



CRATE INTERCONNECT (HELSINKI)

DMA block move between VME crates or VME and CAMAC dataway

VME SOFTWARE DEVELOPMENT



CPUA1

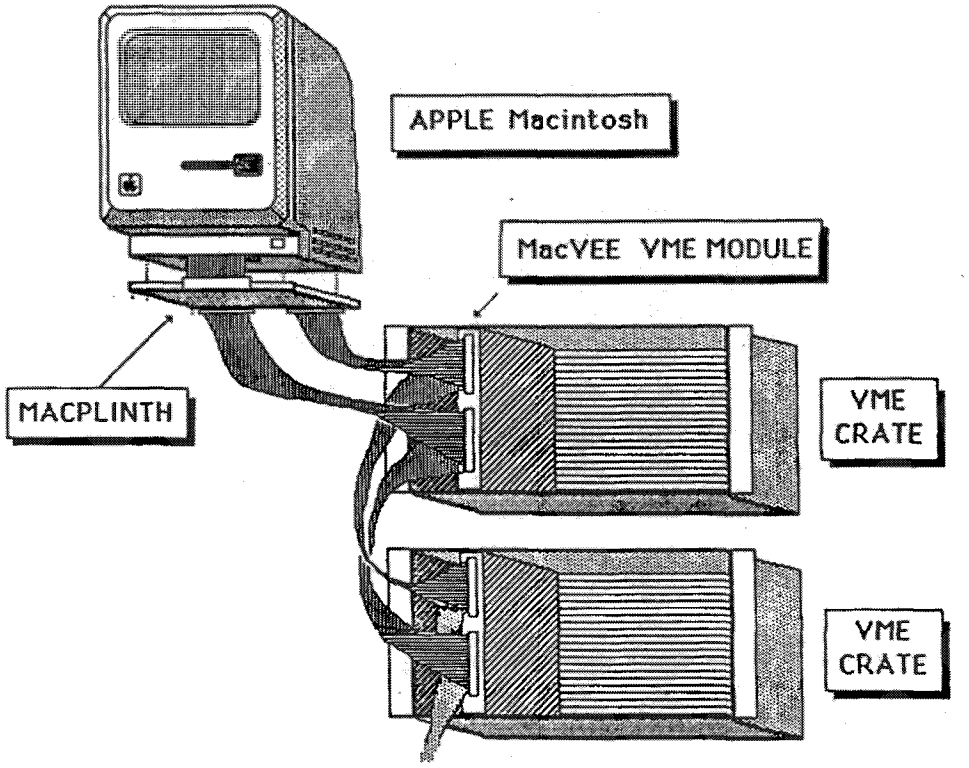
- Resident monitor. MONUA1
- VME Resident Libraries:
  - CAMAC
  - RAM test
  - Crate Interconnect
  - Remus Branch Driver
  - 168E VME-CAMAC link
  - 3081E utilities

NORD 100/500

- CERN MC68000 Cross Assembler/Linker/Pusher
- HYDS Fortran Compiler
- Standard Fortran Libraries

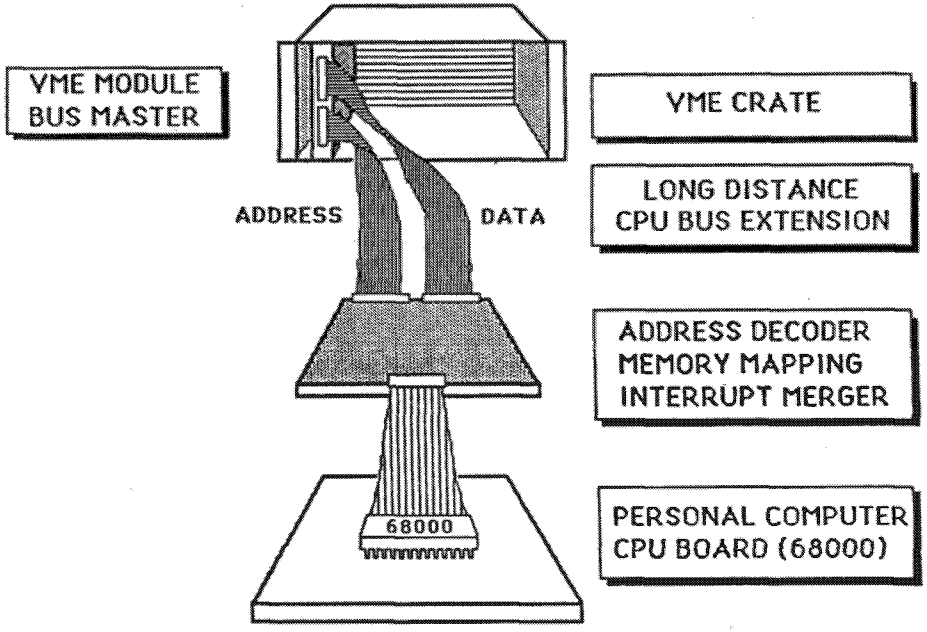
MACVEE

Microcomputer Applied to the Control of VME Electronics Equipment



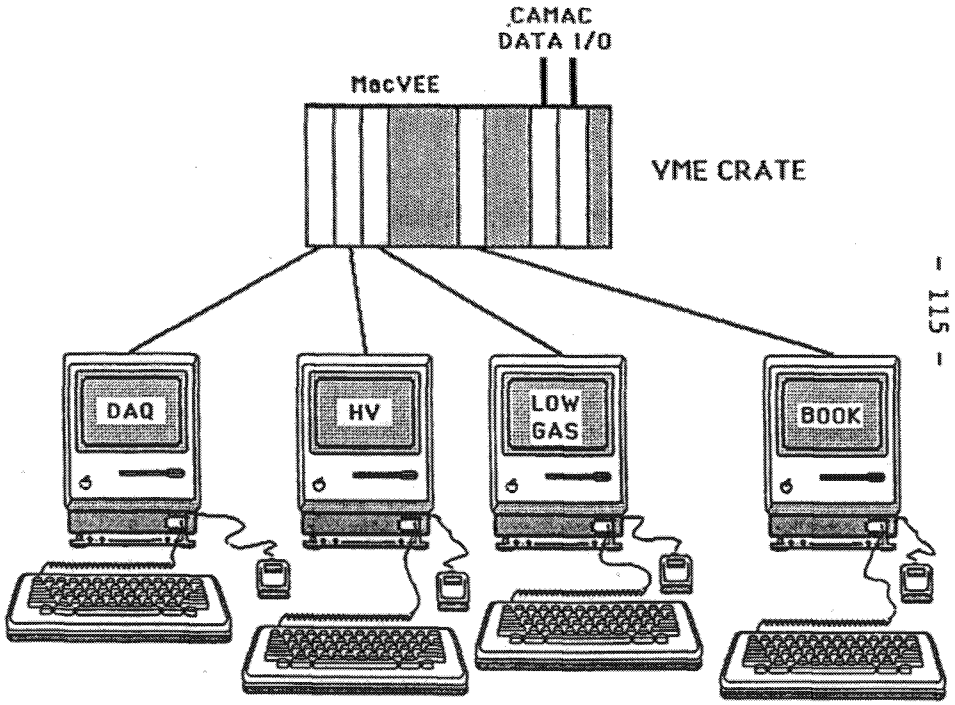
MEMORY MAPPING OF UP TO 8 VME CRATES AND 7 CAMAC CRATES

A 68000 PERSONAL COMPUTER VME INTERFACE



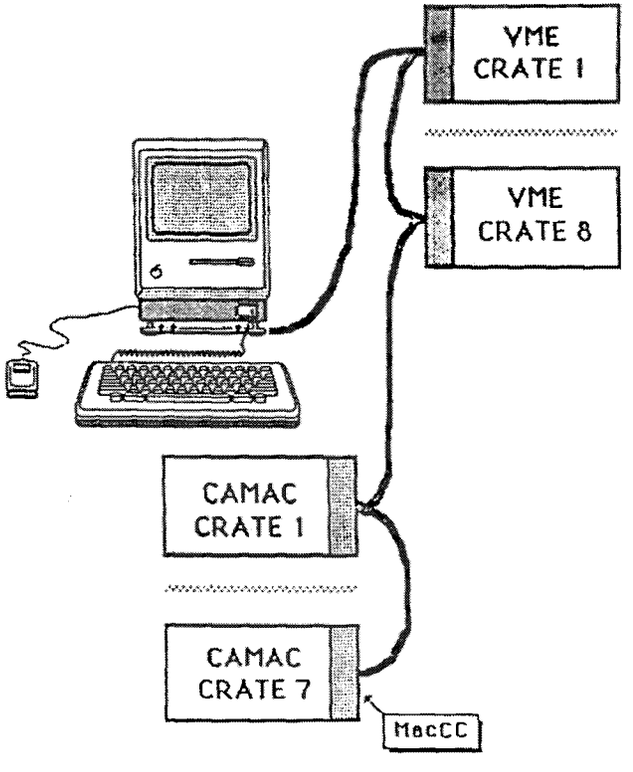
INTERFACE SOFTWARE = "NONE"

MacVEE MULTICONSOLE EXPERIMENT CONTROL SYSTEM



MacVEE APPLICATIONS

- VME/CAMAC TEST SYSTEM
- VME BASED MULTIPROCESSOR SOFTWARE DEVELOPMENT SYTEM
- MULTIPROCESSOR SYSTEM MONITOR AND INTERACTIVE INTERFACE



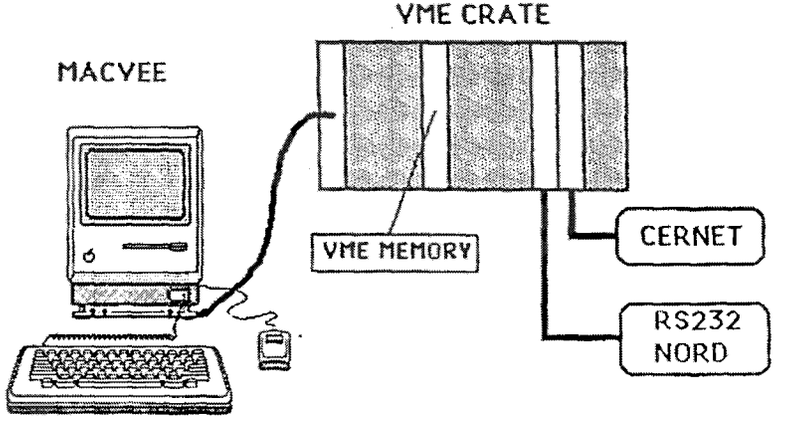
MACVEE SOFTWARE DEVELOPMENT

MACINTOSH STAND ALONE

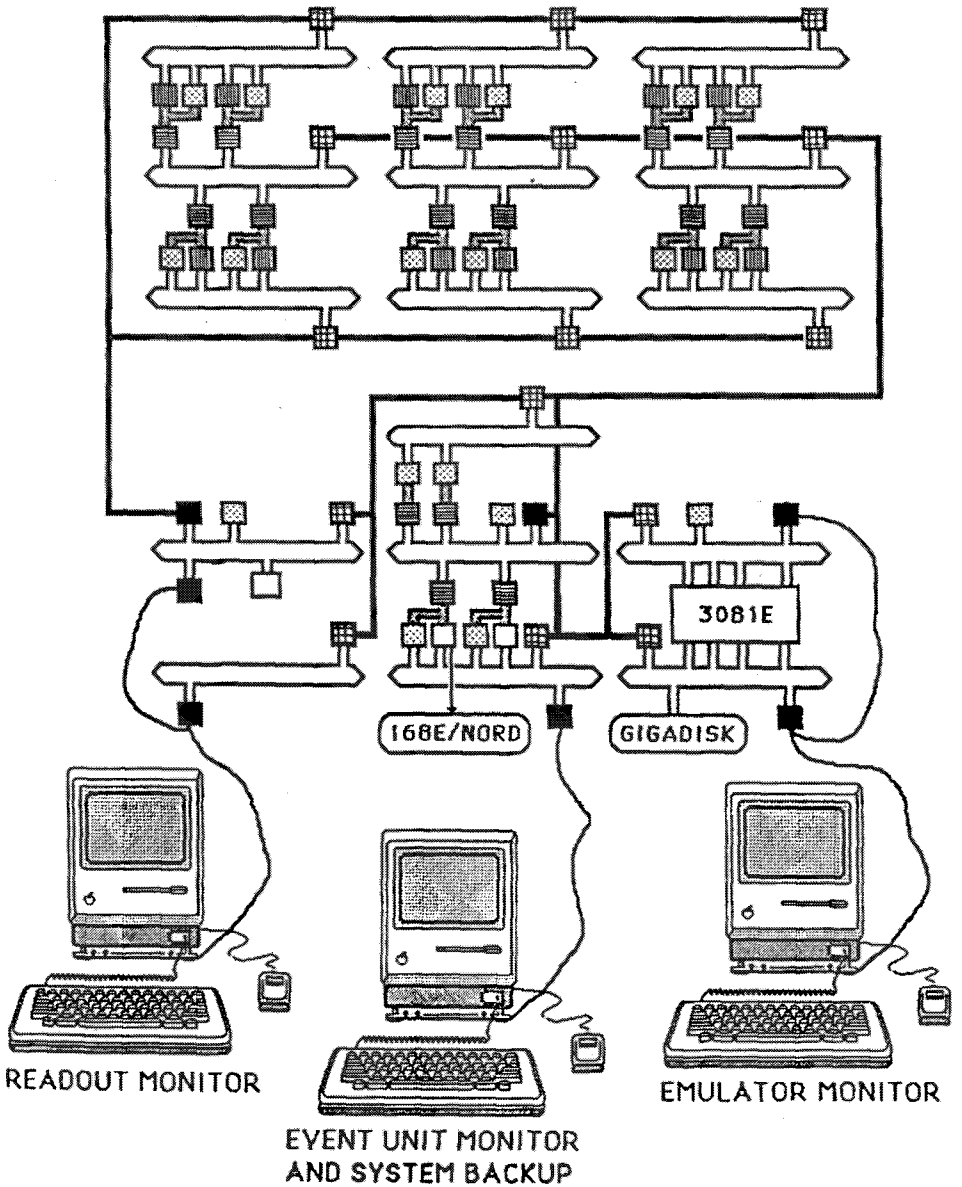
APPLE MDS.	Editor, assembler and linker
ABSOFT	FORTRAN 77
MICROSOFT	BASIC
VME utilities	
CAMAC library	(MAC-CC and DS branch driver)
UA1 System	(General user support)
UA1 Editor	(Window handling facilities)
HVDS FORTRAN	(MAC and CPUA1 development)

NORD CROSS SOFTWARE

CERN CROSS ASSEMBLER/LINKER/PUSHER
HVDS FORTRAN
FORTRAN LIBRARIES
DOWN LOADING INTO VME MEMORY



MACVEE READOUT SYSTEM FRONT END



VME READOUT

Modular structure
Separate bus system/function
Industrial standard multiprocessor bus
Cost effective
Upgradable

CONCLUSION

THE DESCRIBED SYSTEM IS INSTALLED IN THE UA1 EXPERIMENT AND IT IS RUNNING SINCE SEPTEMBER 1985.

THE OPAL VMEBUS DATA COLLECTION SYSTEM

J.C.Brisson, Ph.Farthouat, B.Gandois, F.X.Gentit, V.Hajjar, P.Le Du
E.Lesquoy, J.Mallet, P.Rougevin-Baville, S.Zylberajch

0.1 - INTRODUCTION.

I will present the non-VAX part of the data-acquisition system of the OPAL experiment, the part located between the front end detectors and the VAX cluster. My talk is divided into two parts, the first one for the hardware, the second one for the software.

0.2 - HARDWARE.

It was decided in OPAL not to impose one and only one standard of bus for the front-end part of the data acquisition. Several arguments were in favour of this attitude :

- It appeared expensive to disregard all the existing hardware in CAMAC and also all the experience accumulated with this bus.

- For the small detectors, the complexity of FASTBUS is definitely not needed.

- On the contrary, for the central detector, CAMAC was inadequate and the solution adopted of a combination of ECL crates in EUROPE mechanics and VME crates has two advantages over a FASTBUS solution: it is cheaper and leaves more freedom in the use of microprocessors

In consequence, it was decided to organize the hardware of the OPAL data acquisition according to the following principles :

1. The front-end part of the electronics may be in the CAMAC, FASTBUS or EUROPE-VME standard.

2. The upper part is in the VME standard.

3. The general structure is an inverted tree with branches and sub-branches. The tree has 5 levels. The tree appears naturally due to the fact that each proceessor in the set-up has one and only one direct master, able to read or write into its memory. Nobody else in the set-up is able to read or write into the memory of the processor.

4. The first subdivision of the tree is a subdivision into 12 detectors. Each detector has a test computer and is able to work in a stand alone mode, separated from the rest of the experiment during the test phasis.

5. The only way to communicate from one branch to another is through messages (by the PALABRE message system which will be presented later). It is safe in the sense that it is impossible to have a direct access from one branch to another or from one detector to another, so that a software error in one detector cannot cause any trouble in other detectors.

0.2.1- THE FRONT-END PART.

(see fig. 1)

The front-end part is at the fifth level of the tree. Its processors are ACC (auxiliary crate controllers). 3 cases are to be considered :

a) CAMAC crates.

(see fig. 2)

In that case, the ACC are the auxiliary crate controllers named CACs and built at Saclay : they use an M68000 microprocessor, are perfectly conform to the CAMAC norm concerning auxiliary crate controllers and are also conform to the CERN-SACLAY convention for addressing CAMAC with a M68000.

All crates are organized in branches . At the top of the branches are CBAs (auxiliary branch controller), located inside the detector system crate. We describe them in the section concerning the detector system crate.

b) FASTBUS crates.

(see fig. 3)

FASTBUS crates are used by all the detectors using charge integrating ADC (electromagnetic calorimeter, presampler, end cap calorimeter and hadron calorimeter). All these FASTBUS crates converge into a detector system crate which may be either a VME crate, or a CAMAC crate.

c) EUROPE and VME crates.

(see fig. 4)

For the JET chamber, the Z chamber and the VERTEX chamber, the front-end electronics will use a specific ECL bus in EUROPE mechanics, interfaced to the VME. Each EUROPE crate is read by a CPU board, sitting in a true VME crate, and playing the same role as the CACs in the case of CAMAC. A prototype of such a VME CAC is now in use at the JADE experiment. It is based on the CPU 07 board of MICROSYS. The 8 VME crates containing CACs are grouped into 4 branches driven by 4 VME CBA. This VME CBA will be described in the next chapter.

0.2.2- THE DETECTOR SYSTEM CRATE.

a) Bus standard for the detector system crate.

All the front-end branches of a detector converge into a detector system crate, containing CBAs. This detector system crate is

a CAMAC crate in case of a front-end electronics in CAMAC.

a VME crate in case of a front-end electronics in EUROPE mechanics.

a VME crate or a CAMAC crate in case of a front-end electronics in FASTBUS.

b) CAMAC detector system crate.

In case of a CAMAC detector system crate, the CBAs are also based on the M68000 microprocessor. They work the same as the CACs previously described, and in addition they are able to drive a CAMAC sub-branch. They follow the same norms as quoted for

the CACs. The final version of these two modules have been succesfully tested in real experimental environment.

c) VME detector system crate.

With VME, we have two problems to solve :

1. how to build the equivalent of a CBA in VME.

2. how to interconnect VME crates.

We want the VME CBA to be able to drive a branch with 16 crates. We do not want to build ourselves the CPU used in the VME CBA. We want to buy it as a commercial VME board. So a VME CBA is at least composed of 2 boards : the CPU board and the branch driver board. The connection between the CPU of the CBA and its branch driver cannot use the VME bus, but must use a private bus in order that many CBAs be able to work in parallel in the same crate.

On the contrary, if the problem is not to build a CBA, but only to interconnect VME crates then the interconnecting boards have to be operated from the VME bus.

We intend to solve the two problems (the problem of building CBAs in VME and the problem of interconnecting VME crates) by mean of a VME board, called VME Interface Port (VIP),now in development at Saclay.

i* The VIP.

(see fig. 5)

The VIP can be considered as a module with 3 physical ports:

1. A VME port.

2. A VMX32 port.

3. A branch port, for the connection to other VIPs in other crates. We will call this port the BVMX32 port, B standing for branch.

To each of the 64 lines of the VMX32 bus corresponds a differential pair in the BVMX32 branch : this in order that the BVMX32 branch be adapted and able to extend over long distances. Except for that point, the BVMX32 branch is identical to the VMX32 bus.

All the VIPs connected together by a BVMX32 branch have the same branch number. Bits 28-31 of the address are reserved for the branch number: so each time an address is emitted with bits 28-31 indicating the branch number of a VIP in the crate, then the VIP is addressed and the transaction goes through the branch.

Bits 24-27 of the address indicates the destination crate. So once the transaction is in the branch, the VIP with the crate number specified generates the transaction into its crate and give the result back into the branch.

So, if the VIPs are used for a VME-VME interconnection, the path followed by the transaction is :

The VME bus of the master crate between the CPU and the VIP of the master crate.

The BVMX32 branch between the VIP of the master crate and the VIP of the destination crate.

The VME bus of the destination crate between the VIP of the destination crate and the addressed module.

If the branch is driven by a VME CBA, then the connection between the CPU of the CBA and the VIP master of the branch is through the VMX32 bus of the crate, so that many CBAs may work in parallel in the detector system crate.

Accordingly, a VME-VME CBA will consist of 3 VME boards, communicating together through the VMX32 bus :

- A CPU board, interfaced to VME and to VMX32
- A double port (VME VMX32) memory extension board.
- A VIP.

If you need instead a VME-CAMAC or a VME-FASTBUS CBA, then it is only necessary to redesign the VIP. A VIP for VME-FASTBUS (called a FIP) is now in development at Saclay.

d) Link with the test computer.

In each detector system crate, there is one CBA which has the special role of linking the detector system crate to the test computer. This special CBA will be called the CCBA. two cases have to be considered :

i* The test computer is only an intelligent terminal (MacIntosh).

In that case, we intend to use for the connection between the CCBA and the MacIntosh exactly the solution adopted by the DD division at CERN for the VALET plus. The characteristics of this solution are the following :

- The link between the CCBA and the MacIntosh is simply an RS232 connection.
- All application programs are running in the CCBA, not in the MacIntosh
- The MacIntosh is only there for :

  Editing PILS programs, downloading them into the CCBA, saving them on disketes or printing them.

  Allow the program running in the CCBA to do I/O on its keyboard, screen and printer.

  Display of histograms on the screen.

  Allow the program running in the CCBA to use files.

In the case of a VME detector system crate, we have nothing else to do than taking hardware and software of the VALET plus (PILS,MINIGD3, HMINI). In the case of a CAMAC detector system crate, P. Scharff-Hansen and J.Petersen of the DD division of CERN have succeeded in transporting the VALET plus software on our CAMAC CBA.

ii* The test computer has a VME or CAMAC interface.

It seems to-day that in OPAL only two kinds of test computers will be used : MacIntosh and MicroVaxes. In the case of a MicroVAX test computer, we intend to provide much more than simply an intelligent terminal. In particular we ask

- that the MicroVax be able to run programs itself and execute operations into the VME (or CAMAC) crates of its detector. In the case of VME, there is no problem if the MicroVax has a VME interface: it can access any crate of the system through the VIPs of the CBAs of the detector system crate. In case of CAMAC, the problem is that the the CAMAC CBA is not transparent, so that an operation in a subbranch is only possible in an indirect way, through a dialog with the CBA of the subbranch.

But all this dialog may be done by software in a mode invisible by the user, so that the transparency is recovered by software.

- that the MicroVax be able to send or receive messages from the PALABRE message system (the PALABRE message system connects all mini or micro processor involved in data acquisition and will be described later).

0.2.3- THE VME-VMX MATRIX INTERFACE.

(see fig. 6)

When the detector's data have reached the CBAs of the detector system crate, they are read by the master of the CBAs. The master of the CBAs is called the DIF (Data acquisition processor of the InterFace) and is a processor of the VME interface. If the detector system crate is a CAMAC crate, then the link between the DIF and the detector system crate will be through a VME CAMAC branch driver (the one commercialized by C.E.S.). If the detector system crate is a VME crate, then the VIP will provide the link.

The OPAL VME interface is formed of N+2 VME crates, where N is the number of detectors. So there is one crate per detector plus one master crate, plus one extension of the master crate. In each of the crates assigned to a detector, there is one processor, the DIF and a certain number of memory boards. The DIF uses one and only one memory board to store one event of detector n. If an other event comes, an other memory board will be used. All these memory boards are connected vertically by a branch extension of the VMX32 bus (the same as described for the VME CBA). At the top of the BVMX32 branch, inside the VME master crate, is a CPU, called the FIF (Filtering processor of the InterFace). This FIF is a VME-VMX32 CPU board, and a small appendix behind the VMX32 connector transforms the VMX32 bus into a BVMX32 branch. It is this BVMX32 branch which connects vertically the FIF with all its memory boards. The FIF can read through its BVMX32 branch a whole event, that is to say N memory boards, each filled with data of a particular detector. So the FIF is the first processor of the set-up which may do sophisticated filtering, because it is the first processor handling a whole event.

In the VME master crate is a special processor called the MIF (Master of the InterFace) which is in fact the direct master of the DIFs and FIFs. It is able to read into the memory of the FIFs through the VME bus of the master crate. It is also able to read into the memory of the DIFs through the VMX32 branch connecting the MIF and the DIFs. So DIFs and FIFs are at the same level of the tree (level 3) and the MIF is their master (level 2). The master of the MIF is the VAX (level 1). The role of the MIF is the following :

1. Assign to each DIF the memory slot that it has to use for the next event. This is done by looking at the FIFs in order to find one free (that is to say having terminated its filtering process and sent its event towards the VAX).

2. Looking at the FIFs in order to allow one FIF having terminated its filtering process to send its event towards the VAX.

3. Handling of the fast gate module of the experiment.

4. Handling of the PALABRE message system.

0.3 - SOFTWARE.

0.3.1- INTRODUCTION : LESSONS FROM THE PAST.

I won't go here into the PALABRE data acquisition program itself, which is documented elsewhere. I will restrict myself to the general principles that we are trying to follow, and I will spent some time on the main characteristics of this program : the PALABRE message system.

In order to understand the background on which our ideas are based, it is of interest to have a look at the past and see how microprocessors were used in physics experiments.

Let me distinguish four periods for the use of micros in high energy physics experiments :

a) Good old days.

During these ages, high energy physics experiments were driven only by one mini-computer, without the help of microprocessors. It is however to be remembered that during these good old days mini computers were good in real time applications : a PDP 11 working under RT11 needed only 100 microseconds to react to a trigger.

b) Infancy.

Microprocessors made their appearance in experiment first as a non essential help for the mini-computer, to solve very specific problems, like

reordering or compacting of data for one apparatus.

elementary track reconstruction.

calculation of simple second order trigger.

The characteristics of this way of using microprocessor are the following :

- the microprocessor is not an essential part of the experiment : if it fails, it can be switched off and the work that it was intended to do is either done by the data-acquisition computer or left for the off-line analysis.
- the setting up of the microprocessor was very often not done by the people in charge of the data acquisition, but by the people in charge of the apparatus on which the microprocessor is intended to work. In case of a problem, the data acquisition people could only switch the micro off and wait for the arrival of the expert.

c) Babel.

A proliferation of microprocessors in experiment appeared very soon, for a lot of good reasons : they are cheap, fast and fun. An other reason, not very often quoted, for this proliferation is the degradation of the real time characteristics of mini-computers : nowadays a VAX needs 3000 to 4000 microseconds to react to a trigger.

As the way of using microprocessors did not change, the setting up of all these micros became very painful, necessitating the presence of all experts all the time. The tasks done by the micros were so numerous that it became unrealistic in case of problem to switch them off and to let the off-line people handle each case of failure.

d) Remedies : standardization and communication.

Two remedies may be and are now applied to this situation: standardization and communication.

If the level of standardization increases, it becomes more and more likely that people in charge of the data acquisition will be able to manage possible problems or bugs, because they no longer have to remember a lot of different systems.

On the other hand, if you are able to communicate between micros, you benefit from the following advantages :

- A microprocessor faced with an abnormal situation can send a message towards its master before it stops because of this abnormal situation. You are immediatly informed of which micro has failed and why. An example of this kind of failure is the case of spoiled memory : it is very easy to implement, by mean of a checksum test, a test of the portion of memory containing the code of the program, test which is activated from time to time. If this test fails, then the micro can send towards its master a message saying "my code is spoiled", and stop activity.

- As in the example just quoted, there are cases where the hang-up of the processor can be cured without human intervention, provided you have a mean of communication between processors : for instance, in the case of spoiled memory, the VAX can be asked to reload the program into the hanging processor and to restart it, all automatically.

- If you have a message system connecting all your processors, you can even debug a failing processor from any terminal of the set-up. It is not necessary to go down the pit in front of the microprocessor with a terminal under your arm.

0.3.2- PRINCIPLES ADOPTED IN THE OPAL EXPERIMENT.

a) Standardization.

In OPAL we propose to use these ideas in the following way :

1. We decide to standardize on the use of microprocessors of the M68000 family.

2. We also standardize on the use of one MONITOR for all our microprocessors : we choose MONICA because :

   Contrary to many other monitors, it is easy to transport MONICA from one processor to an other: we had no problem to transport it from the VME CPU board 101 of MOTOROLA to our CAMAC CACs and CBAs. This characteristic is essential in an experiment where different busses will be used, and even different CPUs inside each bus.

   It allows without any adaptation the use of the CERN cross-software system, allowing modular programming in PASCAL, MODULA 2, FORTRAN 77 or ASSEMBLER.

   It has a debugging facility.

   It needs less than 20 microseconds to react to an external trigger.

3. We standardize on the use of PASCAL and ASSEMBLER for the body of the data acquisition program, allowing however USER routines to be written in FORTRAN 77. A USER routine is easy to switch off and the responsability of debugging it belongs to the user.

4. We standardize on the use of the CERN-DD library for graphics and histogramming.

Although we said that one of the reason for the Babel-like situation that arose in the micro-world was the fact that the programs inside the micros were written by specialists of the apparatus and not by the people in charge of the data acquisition, we still think that these people are best placed to write the data acquisition and monitoring routines for their equipment. It is much safer that these routines are written in the form of user routines, embedded in the framework of the main program, which is written by the data-acquisition people AND WHICH IS THE SAME IN ALL MICROS.

So the last, but not the least point in the standardization of the OPAL data acquisition system is the fact that we hope to reach a situation where the programs are the same in all micros, except for the user routines and options selected.

b) Communication : the PALABRE message system.

The most important aspect of the OPAL data acquisition program in the microprocessors is the PALABRE message system.

Having recognized the necessity of a communication system between micros, we assigned the following tasks to it :

1. Fatal error messages in case of abnormal situations.
2. Non fatal error messages.
3. Periodic display on a centralized terminal of the most important flags of the data acquisition program of each processor.
4. Setting of flags inside each processor.
5. Loading or checking code.
6. Requesting or sending histograms.
7. Requesting or sending events to spy computers.

Once these specifications were made, we chose to use for the physical part of the PALABRE message system the same path as the data. It means that the PALABRE message system uses the CAMAC dataway and branch in the CAMAC part of the experiment, and the VME or VMX32 bus in the VME part of the experiment. The reasons for this choice are the following :

- In 1984, when we began writing the code, no standard had yet emerged for a LAN applied to an high energy physics experiment.
- Even if the situation has changed in 1985 with the emergence of Cheapernet and Utinet, one of the specifications of PALABRE is not satisfied by either of these LAN : the speed. The time needed to load the code inside 100 micros would be prohibitive with these LANs, but is acceptable with PALABRE on CAMAC or VME.
- By using the physical path of the data for the message system, we reduce the physical cost of the message system to zero.

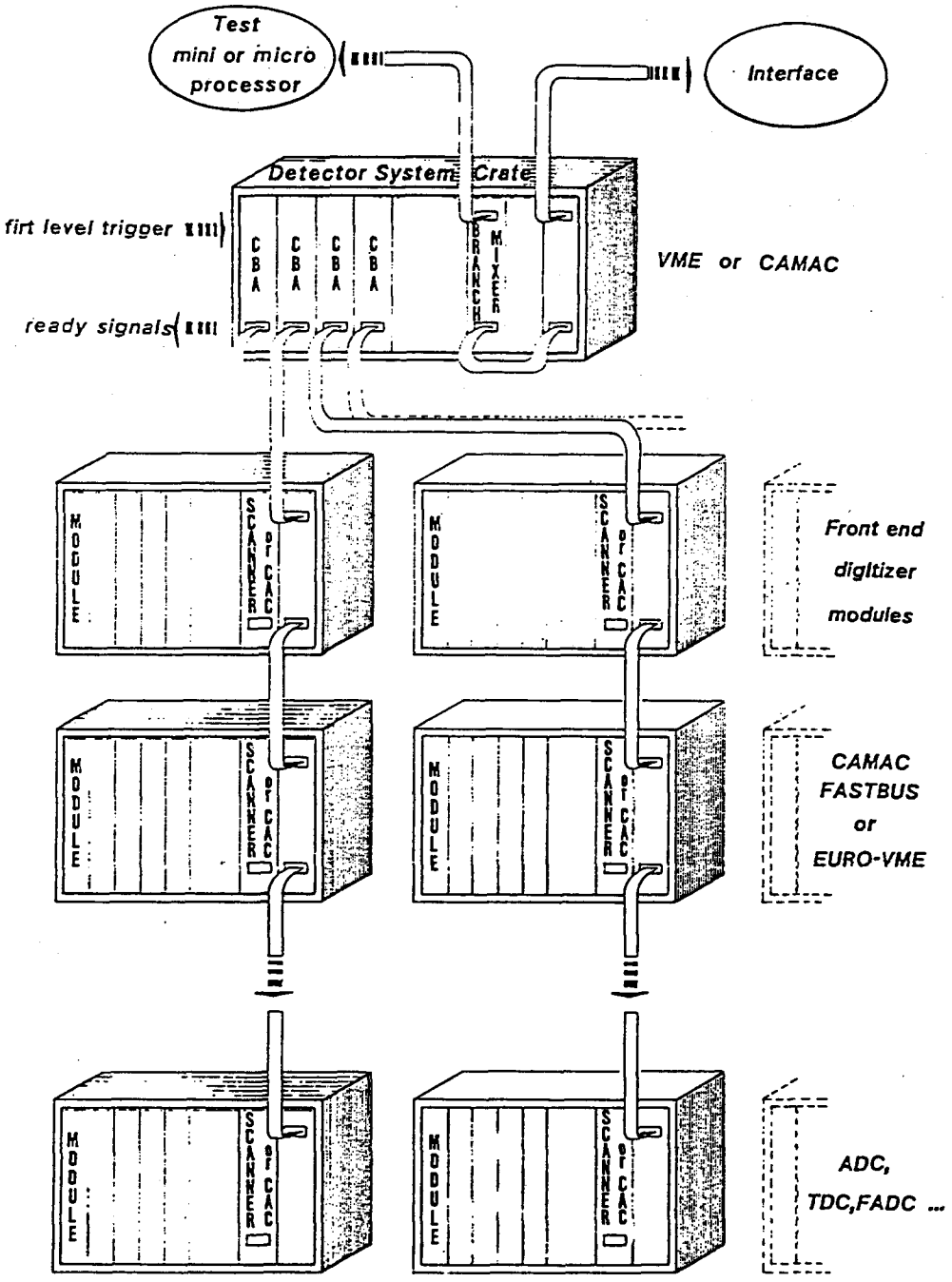
Another point should be pointed out. The PALABRE message system is implementated on an inverted tree with five levels (see fig. 7). As a consequence, the routing problem is extremely simple. It represents a routine of 40 lines of assembly code. This is to be contrasted with the cumbersome problem of geographical to logical address translation in FASTBUS, where there is no tree structure.

0.4 - CONCLUSIONS.

We had the opportunity to test a first version of the PALABRE data acquisition program in a set-up with a tree of 3 levels (VAX,CBA and CACs) during the tests of JULY 1985. In order to simulate the most complex situation compatible with these tests, we have artificially grouped into one branch of 4 CAMAC crates 4 different detectors of OPAL. Each crate had a CAC, the branch was driven by a CBA and the VAX was reading the events from the memory of the CBA.

We were happy to see that all the synchronization problems between the VAX and the micros, and also between the micros themselves were perfectly taken in charge by the Palabre message system.

These opportunities of testing the program are essential. Even if we had the chance of verifying that the general design of PALABRE is sound and working, these tests always point at a series of minor deficiencies which would not have been discovered otherwise.



CBA : Auxiliary Branch Controller (68010 Processor and Memory)
CAC : Auxiliary Crate Controller (68010 Processor and Memory)
SCANNER : Hardware Zero Skipping Module

FIG 1

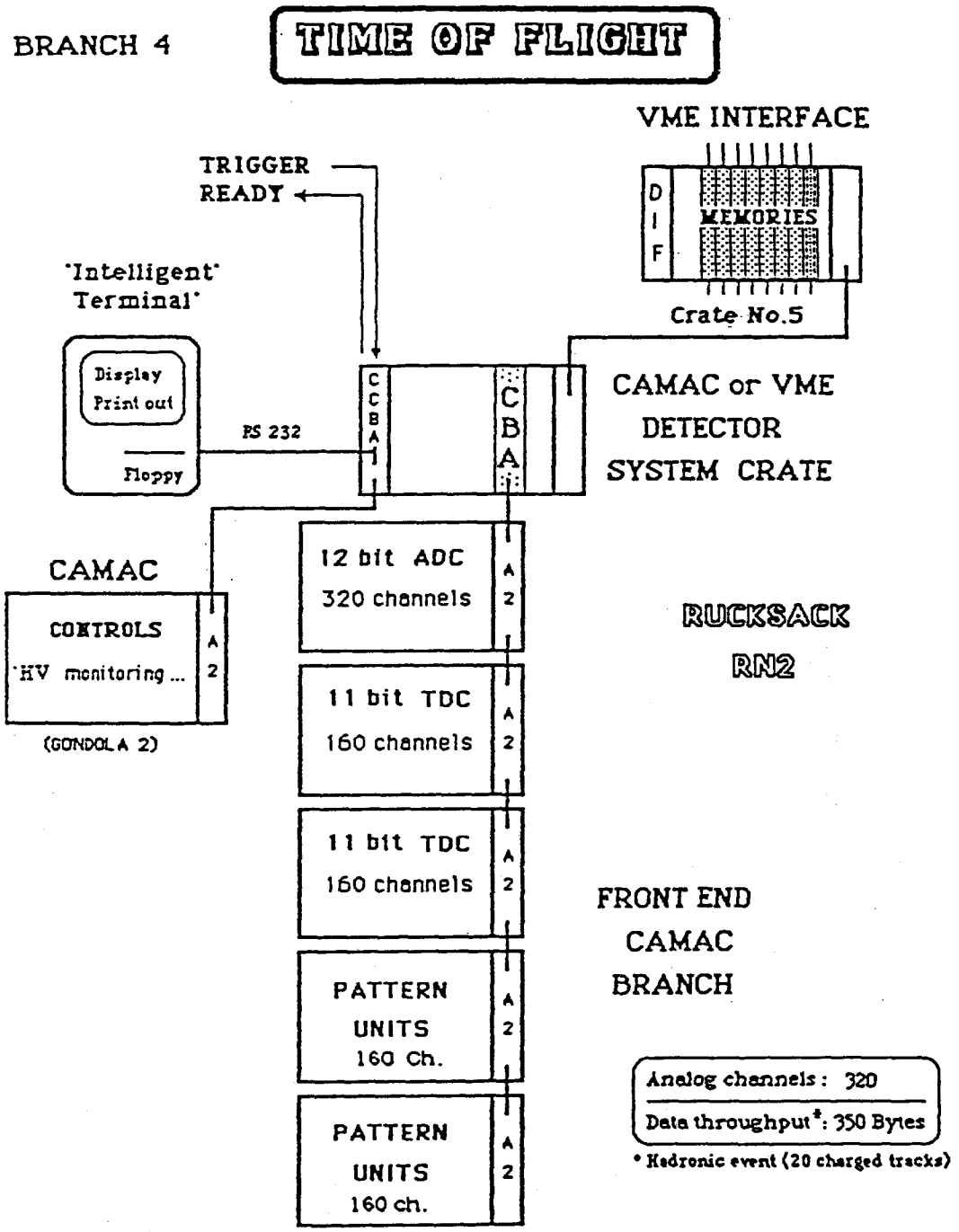

BRANCH 4
TIME OF FLIGHT
VME INTERFACE
TRIGGER
READY
D
I
F
MEMORIES
Crate No.5
'Intelligent'
Terminal'
Display
Print out
Floppy
RS 232
C
C
B
A
C
B
A
CAMAC or VME
DETECTOR
SYSTEM CRATE
CAMAC
CONTROLS
'HV monitoring ...
A
2
(GONDOLA 2)
12 bit ADC
320 channels
11 bit TDC
160 channels
11 bit TDC
160 channels
PATTERN
UNITS
160 Ch.
PATTERN
UNITS
160 ch.
RUCKSACK
RN2
FRONT END
CAMAC
BRANCH
Analog channels : 320
Data throughput*: 350 Bytes
* Hadronic event (20 charged tracks)

FIG 2

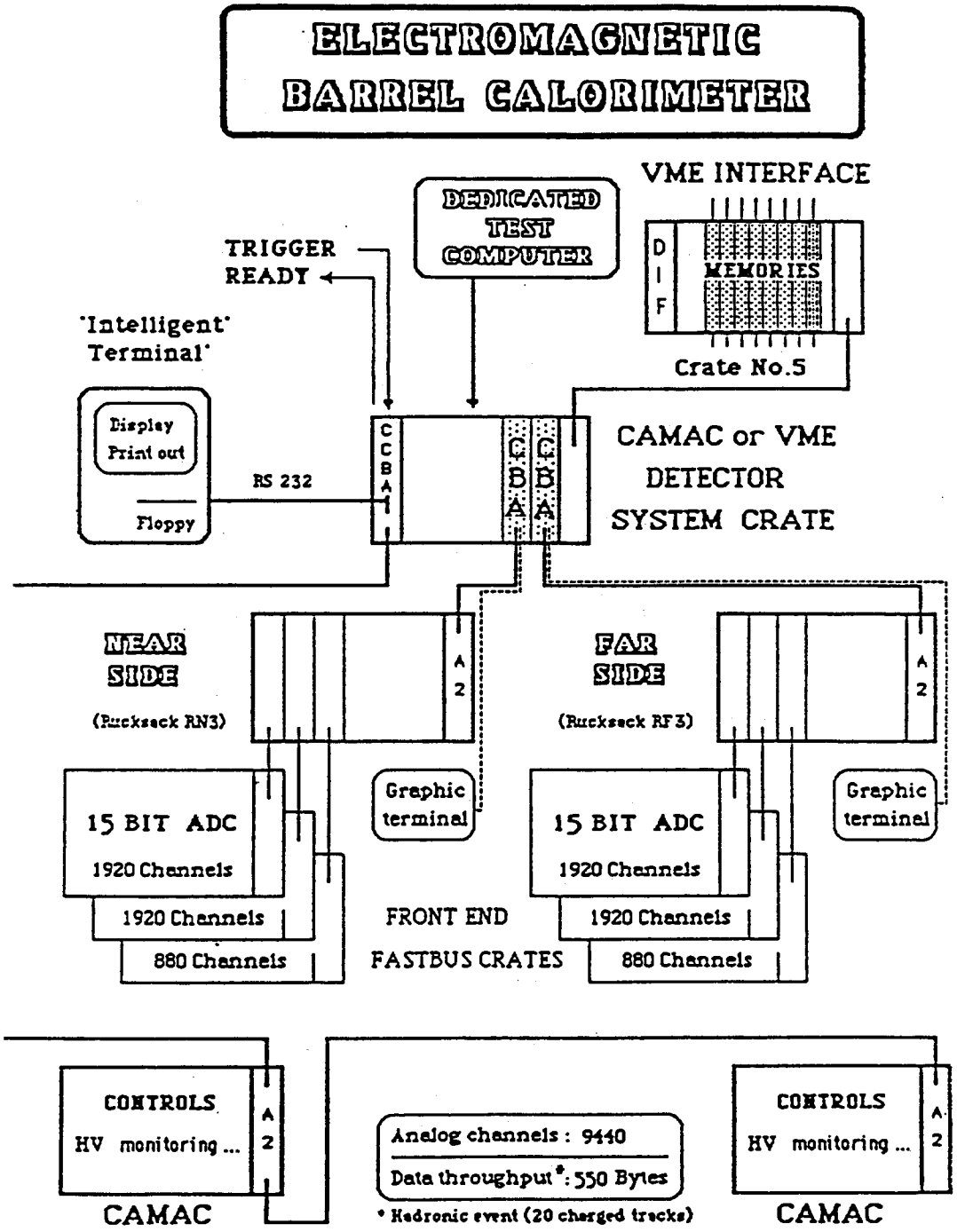

ELECTROMAGNETIC
BARREL CALORIMETER
VME INTERFACE
DEDICATED
TEST
COMPUTER
TRIGGER
READY
'Intelligent'
Terminal'
D
I
F
MEMORIES
Crate No.5
Display
Print out
Floppy
RS 232
C
C
B
A
CAMAC or VME
DETECTOR
SYSTEM CRATE
NEAR
SIDE
(Ruckseck RN3)
FAR
SIDE
(Ruckseck RF3)
A
2
Graphic
terminal
15 BIT ADC
1920 Channels
1920 Channels
880 Channels
FRONT END
FASTBUS CRATES
CONTROLS
HV monitoring ...
CAMAC
Analog channels : 9440
Data throughput*: 550 Bytes
* Hadronic event (20 charged tracks)

FIG 3

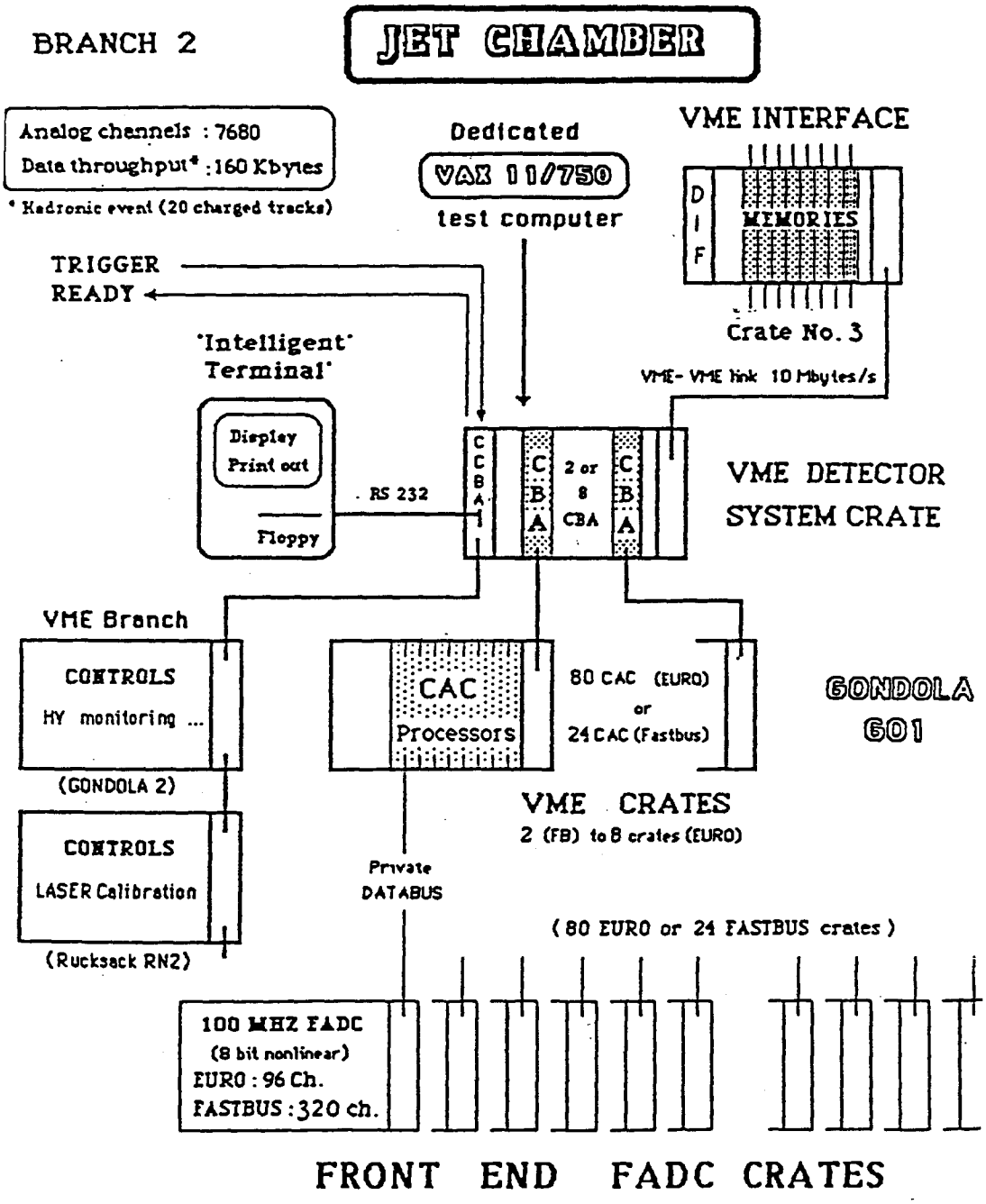

BRANCH 2
JET CHAMBER
Analog channels : 7680
Data throughput* : 160 Kbytes
* Hadronic event (20 charged tracks)
Dedicated
VAX 11/750
test computer
VME INTERFACE
D I F
MEMORIES
Crate No. 3
VME- VME link 10 Mbytes/s
TRIGGER
READY
'Intelligent' Terminal'
Display
Print out
Floppy
RS 232
C C B A 1
C B A
2 or 8 CBA
C B A
VME DETECTOR SYSTEM CRATE
VME Branch
CONTROLS
HV monitoring ...
(GONDOLA 2)
CONTROLS
LASER Calibration
(Rucksack RN2)
CAC
Processors
80 CAC (EURO)
or
24 CAC (Fastbus)
GONDOLA 601
VME CRATES
2 (FB) to 8 crates (EURO)
Private DATABUS
(80 EURO or 24 FASTBUS crates)
100 MHZ FADC
(8 bit nonlinear)
EURO : 96 Ch.
FASTBUS : 320 ch.
FRONT END FADC CRATES

FIG 4

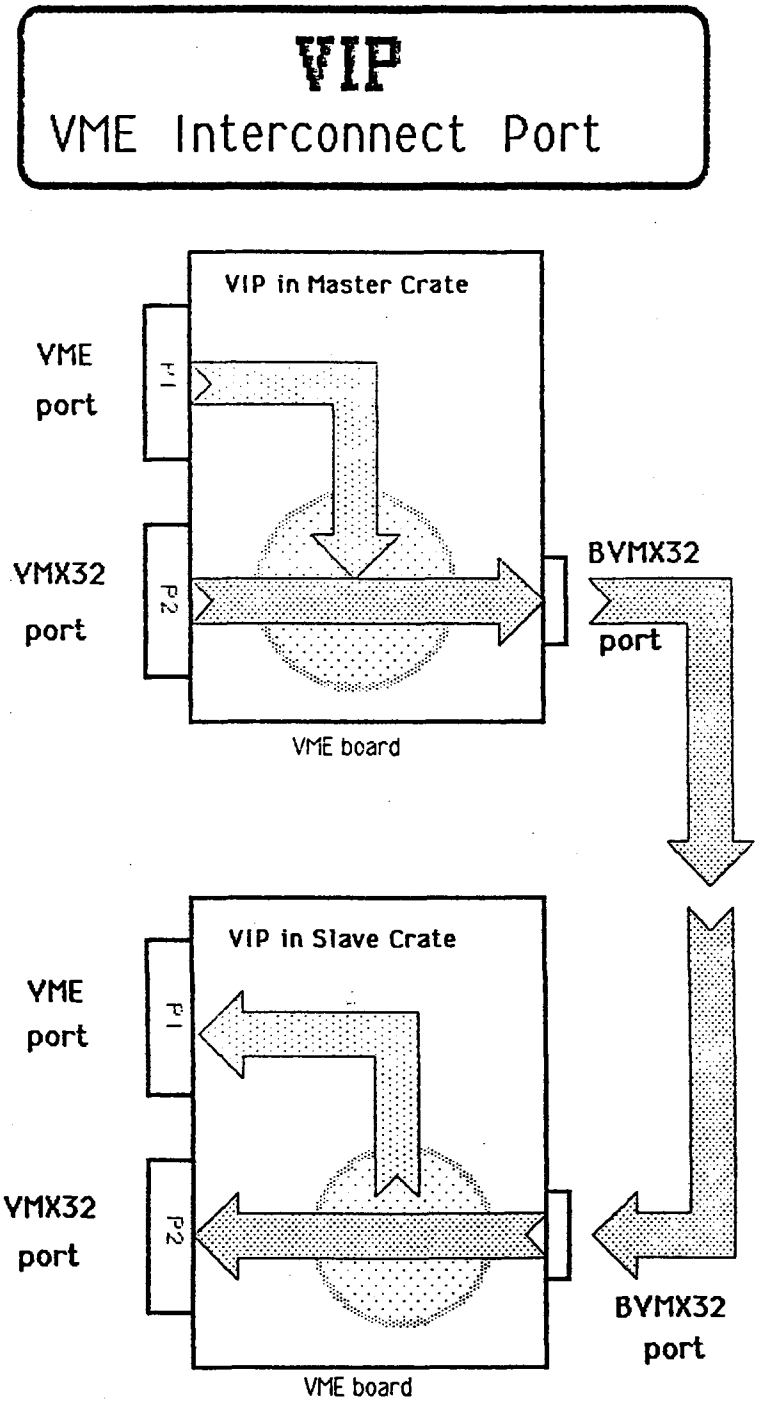

VIP
VME Interconnect Port
VIP in Master Crate
VME
port
P1
VMX32
port
P2
BVMX32
port
VME board
VIP in Slave Crate
VME
port
P1
VMX32
port
P2
BVMX32
port
VME board

FIG 5

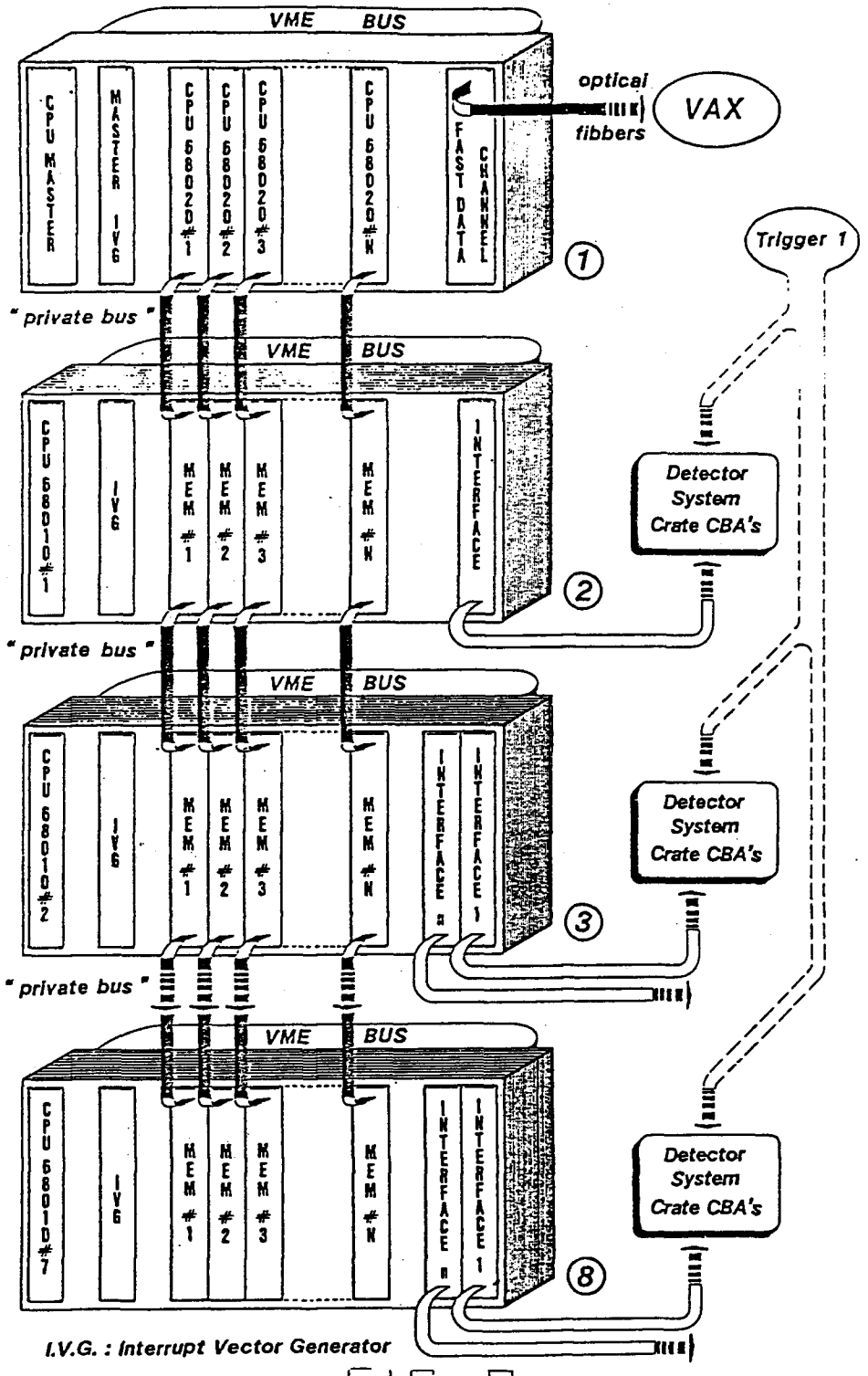
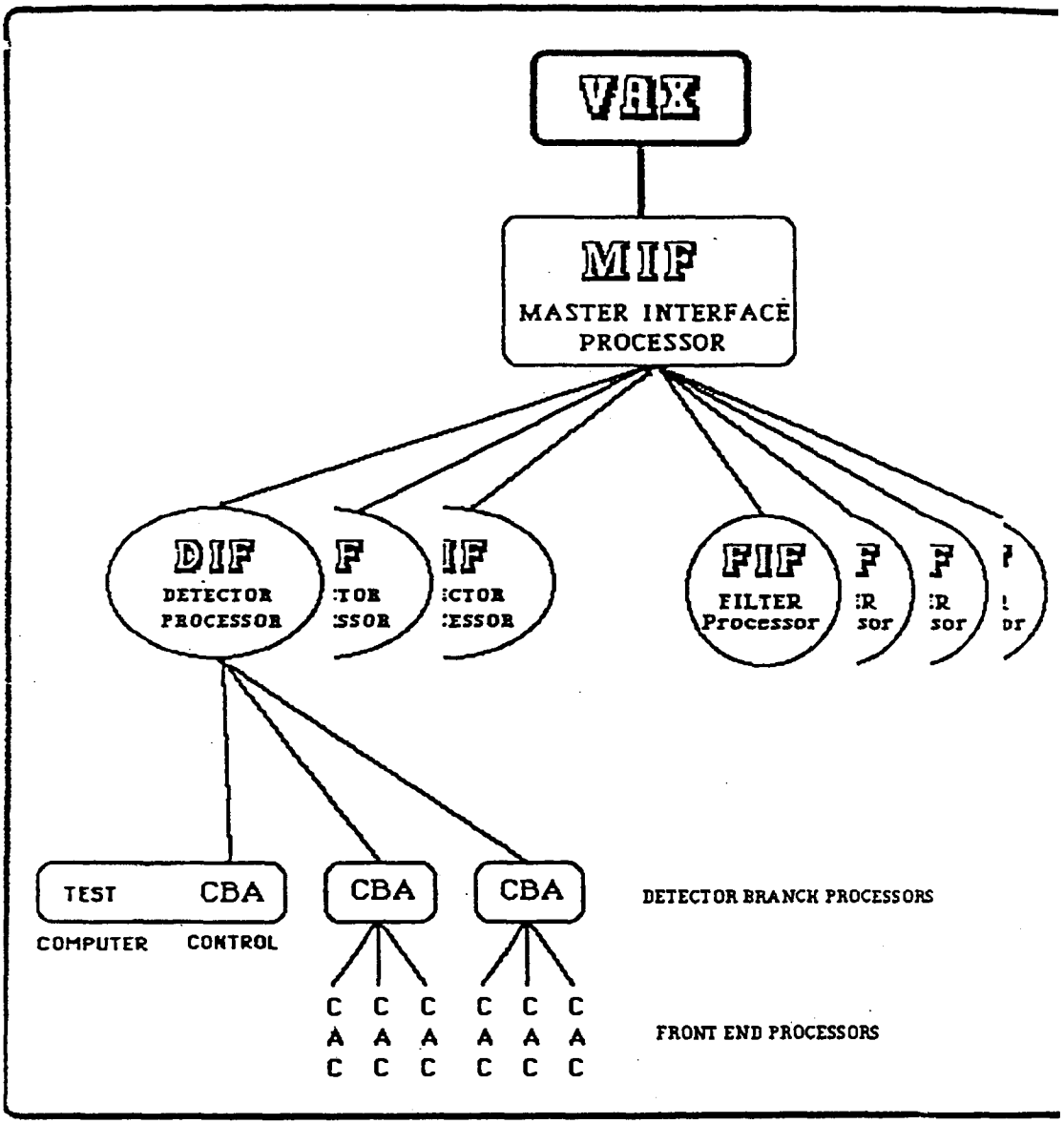

VME BUS
CPU MASTER
MASTER IVG
CPU 68020 #1
CPU 68020 #2
CPU 68020 #3
CPU 68020 #N
FAST DATA CHANNEL
optical fibbers
VAX
Trigger 1
1
"private bus"
CPU 68010 #1
IVG
MEM #1
MEM #2
MEM #3
MEM #N
INTERFACE
Detector System Crate CBA's
2
CPU 68010 #2
INTERFACE 2
INTERFACE 1
3
CPU 68010 #7
8
I.V.G. : Interrupt Vector Generator

FIG 5



PALABRE STRUCTURE

FIG 7

DATA ACQUISITION SYSTEM FOR NORDBALL

Markku Jääskeläinen

Department of Physics

University of Jyväskylä

SF-40100 Jyväskylä

Finland

and

Lars Carlen

Department of Cosmic and Subatomic Physics

University of Lund

S-22362 Lund

Sweden

Abstract

A modular data aqcuisition system based on parallel data read-out and processing and utilizing several MC68xxx based CPU's and VME-buses is described. It has been designed for a joint Nordic Nuclear physics (heavy-ion) 4π multidetector system - Nordball, but is easily applicable for other systems. Readout and preprocessing system has been implemented and full processing system is under construction.

1. Introduction

The rapid development in the experimental facilities during last few years and even more complex systems available in the future have created greater demands on data aqcuisition systems of a nuclear physics laboratories. Especially multicoincidence and multiparameter experiments used in nuclear physics research can only be matched with a multiprocessor data aqcuisition and processing system. The joint Nordic multidetector system (NORDBALL) requires an efficient data aqcuisition system, but also experiments with other equipments are becoming more complex. An efficient system designed for such a detector arrangement with tens of parameters can also easily be adapted for smaller scale experiments and used in smaller laboratories.

The basic concept of the NORDBALL project is that many different types of detectors needed for the experimental program are designed with the restrictions given by the modular NORDBALL frame.

The basic structure gives room for 20 large BGO shielded Ge-detectors placed in the hexagons and 12 smaller detectors placed in the pentagons.

It is planned that at least one frame is installed in connection with each accelerator in the Nordic countries. Many groups are taking part in the design of special equipment for their own primary use, which then becomes available to other users too. The following detector systems are considerd at the moment.

1. Anticompton spectrometers
2. Calorimeter for sum energy and gamma multiplicity and with time of flight (TOF) discriminations against neutrons

3. Beta-spectrometers of the orange type
4. Neutron TOF detectors
5. Particle telescopes
6. Recoil detectors
7. Position sensitive detectors doe Coulomb excitation experiments
8. A plunger facility for lifetime measurements
9. A catcher facility for experiments with delayed radiation

A typical example of the NORDBALL equipped with 20 anticompton spectrometers and a BaF_2 calorimeter for high spin spectroscopy is shown in fig. 1.

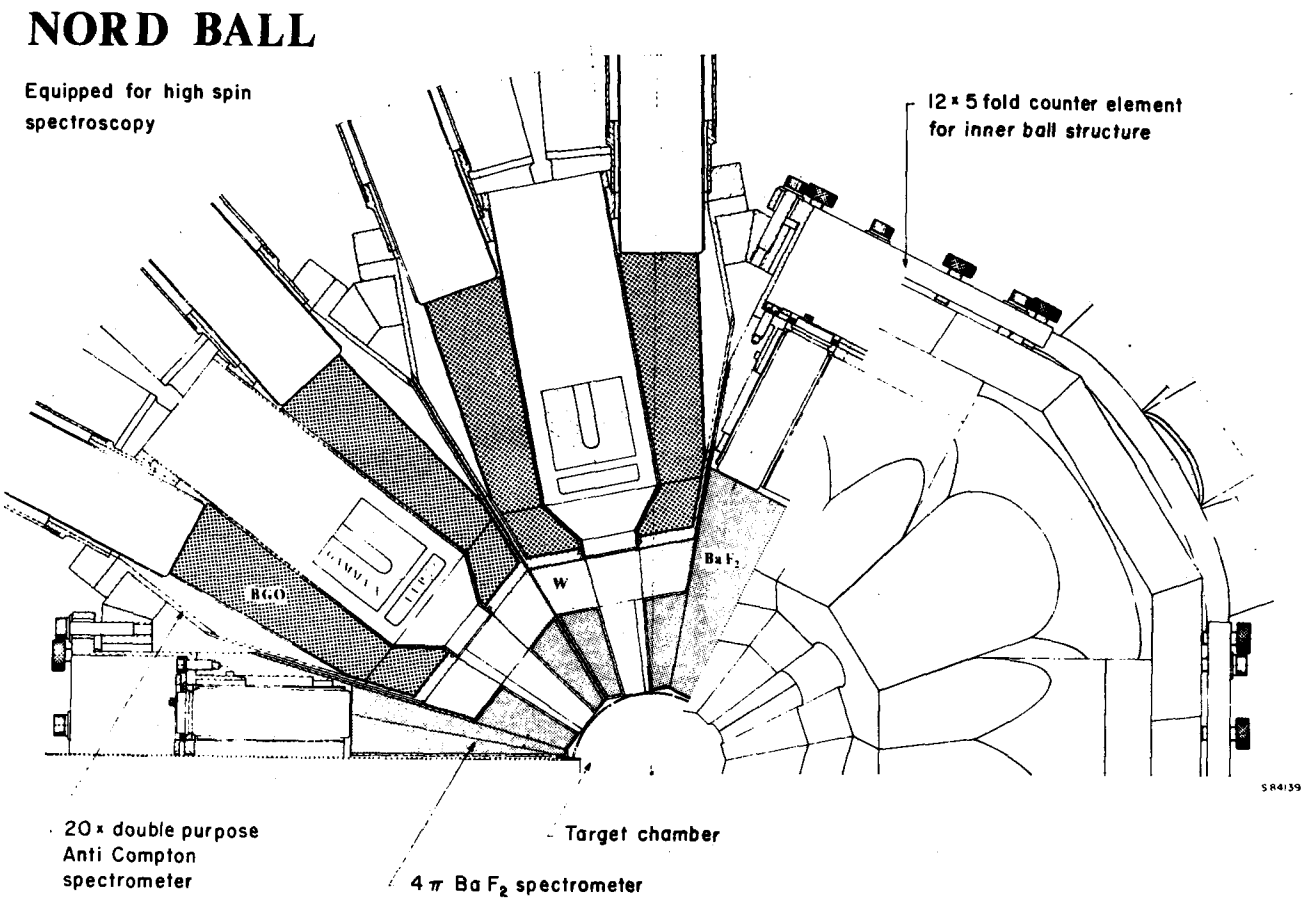


Fig. 1. A cut through the Nordball equipped with detectors for a typical high spin experiment. The set up includes 20 Anti-Compton spectrometers and a 4π BaF_2 calorimeter around the center.

The most important detector, the basic element in NORDBALL detector system is the anticompton spectrometer.

The design is compact and easy to "pack" around the target to obtain optimum solid angle for each detector. The triple coincidence rate is proportional to the 6th power of the distance to the target. A good response function with a high peak/total ratio is also very important such that the remaining background can easily be removed by simple unfolding procedures. It may then become feasible to unfold 3- and maybe 4-dimensional matrix spectra without erroneous "blowup".

The main use of the data acquisition system for NORDBALL is to collect data from the experiment and to store it for final data analysis. However, also on-line sorting is essential due to large amount of events. The data acquisition system has been designed

1) to collect event-by-event data up to 500 kB/s rate without considerable dead time.
2) to preprocess events according to users criteria(by calibrating, gating etc.)
3) to sort or presort some type of events on-line into computer memory or to disk
4) to store unsorted events (or all events if so desired) on tape or disk
5) to monitor some critical parameters during the experiment.

In order to fullfil these requirements a modular VME-based system have been designed. The whole data acquisition system is modular both in hardware and in software design. It is constructed in two steps and thus there will be two generations of the system. It has also been designed so that it can be used at different size of Nordic laboratories together with the NORDBALL or for some other experiments.

2. First generation

In the first generation, which has been constructed, the data collection and preprocessing is implemented on a multiprocessor VME-system. At this point the on-line sorting and storage of the event-by-event data is done by a host computer which has disk and tape units. The existing laboratory computers in each laboratory (PDP, VAX) are used as the host computer. A schematic diagram of a hardware setup of the first generation data acquisition system with one VME-crate and a VAX host computer is shown in Fig. 2. As seen from the figure the system can logically and physically be divided into three separate parts: the LAB-system, the VME-system and the host computer.

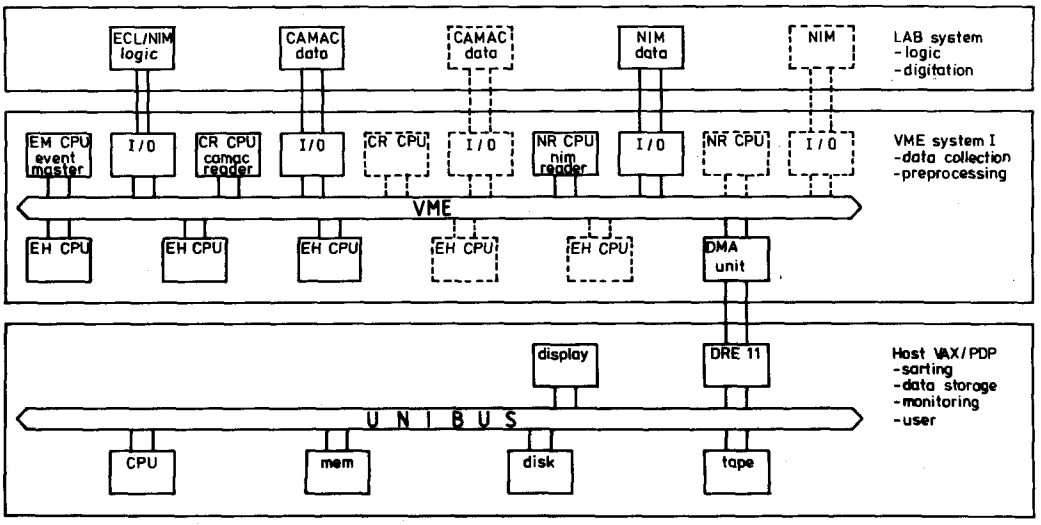


Fig. 2. A block diagram of the hardware of the first generation data acquisition system.

The LAB-system consists of the data digitation devices (ADC's, TDS's, etc.), their housing and interfacing. The system is designed so that different kind of laboratory equipment and data transfer connections can be used according to local and experimental requirements. Presently these include interfaces to NIM and ECL logic, CAMAC crates and direct interface of NIM ADC's to the VME-bus. Later connections for Fastbus, IEEE-488 bus etc. can be implemented. The system is designed to be fully modular so that any combination of the laboratory interfaces can be used.

The VME system I is a multiprocessor system based on one global VME-bus, which can also be distributed into several VMEbuses, several MC68000 (or MC68010) microprocessor cpu's and local buses. This system is used to manage and perform the event-by-event collection from the LAB-system using parallel readout, to perform event-by-event preprocessing (calibration, event recognition, gating, event formating etc.) and to pass the data to the host computer via a fast parallel I/O link. The use of several cpu's on the system enables parallel readout and preprocessing thus utilizing the full power of the VME-system to receive a fast data throughput.

The host VAX computer is used to enter the user control commands, to control the data acquisition, to sort events on-line and to store the sorted spectra into memory or on disk and to write the events on tape as desired. The fast parallel (16 bit) DMA link to the host computer for VAX and PDP computers is done with DRE11 or DR11W units.

The VME-system I used for data collection includes one cpu that will control the event-by-event data readout and preprocessing. Here this cpu is named EM (event master). It is a master cpu only in a logical (software) sense because each of the cpu's in the system will

be the actual bus master whenever it will be using the bus. In addition to the general management of the data EM will handle the communications to the external event logic system. The direct interfacing of the NIM ADC's to the readout VME-bus are done with a 15-channel (16 bits/channel) input registers JYVME-I developed at University of Jyväskylä. The data from these interfaces are read with NIM reader cpus (NR). Depending on the actual number of NIM interfaces and data rate one or more NR cpu's can be used in the system.

Also several CAMAC crates can be interfaced to the VME system. Each crate will be connected using the CAMAC-crate controller developed at the University of Lund and one cpu, (CR). The data from each CAMAC crate and NIM interface are read by the corresponding reader cpu and passed to the system via the dual ported memory of the preprocessing or event handler cpu's (EH).

The preprocessing of the event-by-event data is done using several parallel cpu's connected to the VMEbus and having dual ported memory for data transfer. These cpu's are called event handlers, EH's. The data processing is done by the cpu's using a local bus without disturbing the VME-bus and then passed forward, in block mode using, the fast DMA link.

2.1. Event collection

The external ECL logic is used to create an event signal (trigger) and to interrupt the event master (EM) externally. It also provides a master event pattern and other information about the event

for EM cpu. Once interrupted EM cpu will create an external busy to block further events during the readout and make up the reader cpu's to start the parallel readout. The event master cpu (EM) selects a free event handler cpu (EH), during data digitation and instructs the reader cpu's to press data that cpu.

After the readout is finished and the data is stored into the dual ported memory of the EH cpu each reader cpu informs EM cpu and get ready for the next event. At this point the data has been transferred from the LAB-system into the VME-system and thus EM cpu clears the data registers (JYVME-I, CAMAC TDC's, ADC's and IR's, etc.) externally and then allows new events by releasing the busy signal. It should be noted that using this readout system the parameters are transferred parallel and thus effective transfer rate of several megabytes per second can be reached.

After releasing the LAB-system for the next event, CPU EM will interrupt the processing (EH) cpu locate a new free event handler and wait for the next event.

2.2. Preprocessing

The preprocessing cpu's manipulate the event-by-event data according to user's definitions. At the preprocessing system parameters can be selected, new parameters can be defined, whole events or individual parameters can be gated, some parameters can be calibrated etc. The preprocessing cpu's process the events using local bus without disturbing the VME bus and place the events into a buffer in dual

parted memory. Whenever the output buffer is filled the cpu initiates a DMA transfer from dual ported memory into the host computer using the VME bus, and DRE11 link. The number of the preprocessing cpu's required depends on the event data rate and it can vary from experiment to another. The system is designed to use one to eight preprocessing cpu's without any changes. Typically one to three cpu's are enough. Since the event handler cpu all do the same event formating handling separate events they are independent of each other and the number of them can vary from experimental to another.

2.3. Host computer

The maximum data collection speed is received if each laboratory interface used in the experiment is controlled with its own reader cpu. However, one cpu can manage several interfaces and in an experiment with low counting rates the event master (EM)can do also the readout. Futhermore the addition of different readout channels (ex Fastbus) are relatively easily incorporated to the system of this modular design.

The software for the data collection system is written so that it is adaptable for systems of different sizes from a small system with one cpu up to a multiparameter experimental setup. The data rate received with DRE11 can be easily obtained with the VME system described above. Actually the processing load of the host computer can be decreased by adding more preprocessing power to the system and implementing maximum amount of processing into this step of the data acqui-

sition. The on-line sorting of the events and writing of event-by-event data on tape will be done by the host computer.

In addition to the parallel DRE11 connection between the VME system and the host computer a serial RS232C connection is used for the user control of the experiment. The definitions and the control of the experiment is done by using a control program in the host computer. The same control program is used in the host computer to define the form of data storage. At the first generation the existing tape and disk units are used in the host computer. Eventually a new on-line sorting system will be developed for the host VAX computers to create spectra on disk. However, since the readout from DRE11 is formally equivalent for reading data from event tapes any existing off-line sorting system can be easily modified to be a temporary on-line sorting system. The existing on-line sorting system at PDP11/44 in Jyväskylä has been modified for the new VME based data acquisition system and the modification of the on-line sorting system on VAX 11/780 at Niels Bohr Institute (Risø, Denmark) is under way. An experimental version of the first generation of the data acquisition system has been developed at JYFL during the spring 1985 and is used for experiemnts at Jyväskylä Cyclotron Laboratory. The system will be implemented using a VAX host computer during the and fall 1985. A small version of the Nordball will be operational early 1986 at Risø.

3. The second generation

The second generation will include all the parts of the first generation and a second VME-system will be added and the on-line data

sorting will be done by this VME system II. A schematic diagram of the second generation data acquisition system is shown in fig. 3. The host computer will still control the system (user communication program). The VME system II will be based on full 32 bit address and data space and at least at some parts use MC68020 based CPU's. The data sorting will be done by several cpu's (DS) which are controlled by one data master CPU (DM), (parallel processing).

The sequence of the data sorting at this stage will be done much the same way as the event preprocessing is done in the data collection at VME system I. Here, however, the master CPU will receive blocks from the VME system I via the crate interconnect units and the processing cpu's (DS) will sort the events one block at a time. The master cpu (DM) will distribute the blocks for the slave cpu's se-

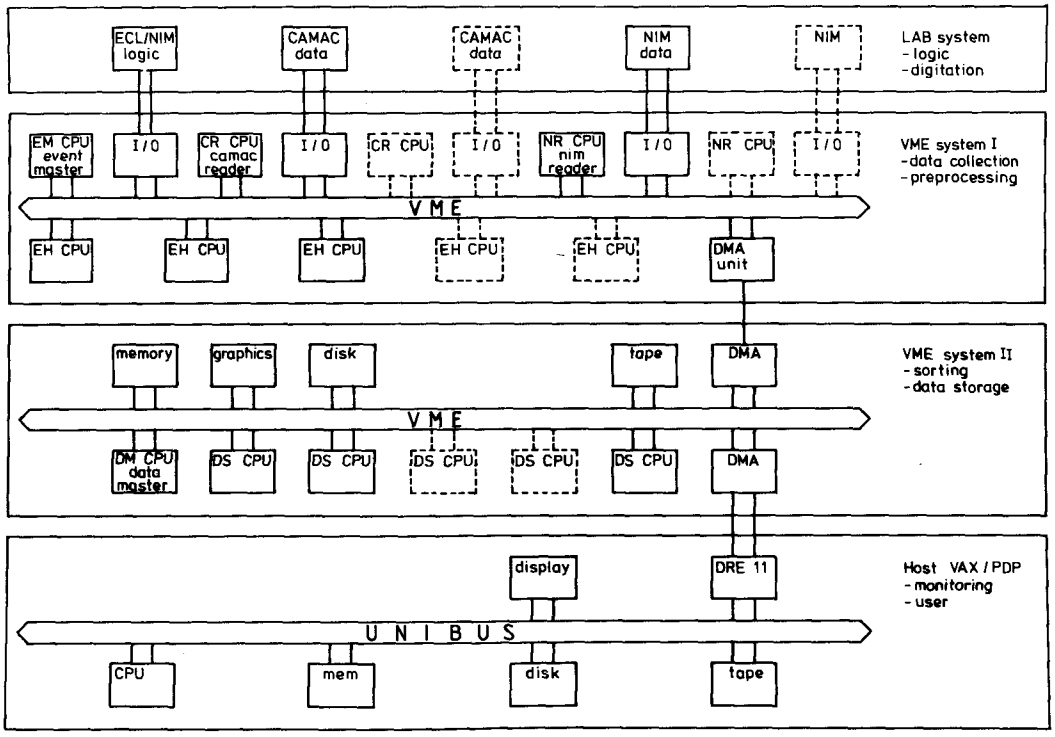


Fig. 3. A block diagram of the hardware of the second generation data acquisition system.

quentially. Also here the amount of the cpu's depend on the amount of computing power required. Since all the slave cpu's are performing the same functions doubling of the number of cpu's will double also the computing power.

Because of the large address space (4GB) even big matrices can be created in memory. In Nordball experiments at least one E_γ vs E_γ spectrum can be created into the memory. In addition to the on-line sorting a graphics system for on-line data display in the VME-bus system will be developed. The development of the VME system II have been started in fall -85 and it will be finished during the spring -86.

The VME system II will also be made modular in both hardware and software so that it can be implemented with different amount of cpu's according to the experimental and laboratory requirements. It will also be designed so that it can be used for off-line data analysis (data playback) to sort the stored events. At this mode the events will be fed from the host computer via DRE11 system or read from a storage unit directly connected to the VME system II.

Acknowledgement

The Nordball project is a collaboration of laboratories at Risó (Denmark), Jyväskylä, Helsinki (Finland), Oslo, Bergen (Norway) Gothenburg, Lund, Stockholm and Uppsala (Sweden).

The authors would like to thank all collaborators for discussions and their help. Especially we would like to thank Dr. A. Holm for valuable discussion about future development. For financial support we would like to thank NOAC (Nordic Accelerator Committee) and the Finnish Academy.

FRONT END PROCESSING FOR A 100 MHZ FLASH-ADC-SYSTEM

G. Eckerlin, E. Elsen, H.v.d. Schmitt, A. Wagner, P.v. Walter
Physikalisches Institut
Universität Heidelberg
Philosophenweg 12
D-6900 Heidelberg

Abstract

An intelligent interface for readout of a high speed (100 MHz), multichannel Flash-ADC System [1] is described. 3072 FADC channels are controlled and read by a system of 34 microprocessors M68000 placed at two different hierarchical levels. In addition to the readout itself, the processors perform a detailed pulse shape analysis neccessary for a compact and manageable data format.

The purpose of the system is to exploit the good double track separation and time resolution provided by Flash-ADCs in conjunction with large drift chamber detectors such as JADE at PETRA [2] and OPAL at LEP [3]. Details of the system presently being installed at JADE are reviewed.

Introduction

In conventional drift chamber electronics the drift time is measured with a discriminator of fixed threshold and a fast time to digital converter (TDC). With this type of electronics the accuracy of the drift time measurement is mainly determined by diffusion, especially for large drift distances, and often by the accuracy of the TDC. In pictorial drift chambers, where genuine space points are recorded, the third coordinate is frequently obtained by charge division. This requires in addition to the timing measurement a charge integration and digitisation at both ends of each signal wire. In the case of multihit electronics one needs furthermore a charge storage. The measurement of the charge provides also information about the ionization loss and thereby about the particle type. The two-track resolution is affected either by the size of the drift cell or by the finite, prefixed integration time required for an accurate charge measurement.

Recently a new technique for the simultaneous precision measurement of drift time and signal charge has been proposed, which provides at the same time an unsurpassed two-track resolution. This system is based on Flash-ADCs which record the development of the drift chamber signal in steps of 10 ns [4]. The detailed analysis of the pulse shape leads to a timing accuracy with small contributions from the electronics and with a strongly reduced influence of the diffusion for large drift distances. The pulse shape analysis allows furthermore a good charge measurement even for overlapping hits. Values of 90 µm for space and 2.5 mm for two-track resolution have been achieved in a prototype cell of the JADE drift chamber (16 wires) and it has been shown that a sophisticated on line pulse shape analysis is possible even in a system with many channels. The implementation of such a system at the JADE detector is described in detail.

Setup

The JADE central tracking device consists of a jet chamber with 1536 wires arranged in 48 concentric layers parallel to the beam axis. The readout electronics used up to now is sketched in Fig.1. The time measurement is performed on the sum of the signals from both wire ends using a discriminator and a RAM as TDC, while the charges are found by analog integration. Up to 8 hits can be stored for a wire [5].

The new readout scheme employs a multichannel transient recorder and subsequent digital signal discrimination followed by detailed software pulse analysis (Fig.2). The final data format is left unaltered: 2 signal amplitudes and the drift time are recorded for each hit. The data compression which starts from raw data of 1536 wires • 2 channels/wire• 256 bytes/channel = 0.75 Mbyte is partly done in hardware and partly in software. Three different hardware layers can be distinguished (Fig.3):

The first layer houses the digitising electronics (FADC-modules) and the hardware processor for zero suppression (SCANNER) in a double height eurocrate (DL300-system).

The second layer contains 2•16 Front End Processors (FEP) which are placed in two VME crates. Each CPU (M68000) is connected to the associated DL300 crate through an interface which resides on a custom made piggyback card on the CPU board. The CPUs read the relevant data of the DL300 system and perform the pulse analysis in order to determine the drift time and the signal charge for each hit.

The third layer, placed in a VME crate, contains the EVent Processor (EVP) which assembles the event and controls the transfer to the host computer of the experiment via a single width CAMAC module. The same VME crate houses another processor, the MOnitor Processor (MOP), which performs checks on data passing through the EVP, determines the calibration constants necessary, and distributes the programs at system startup.

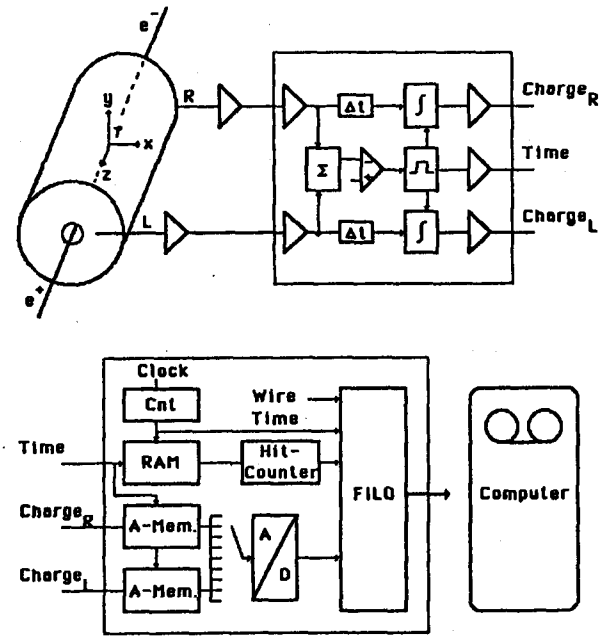


Fig. 1 The multihit electronics of the JADE experiment using analog pulse discrimination and charge integration [5].

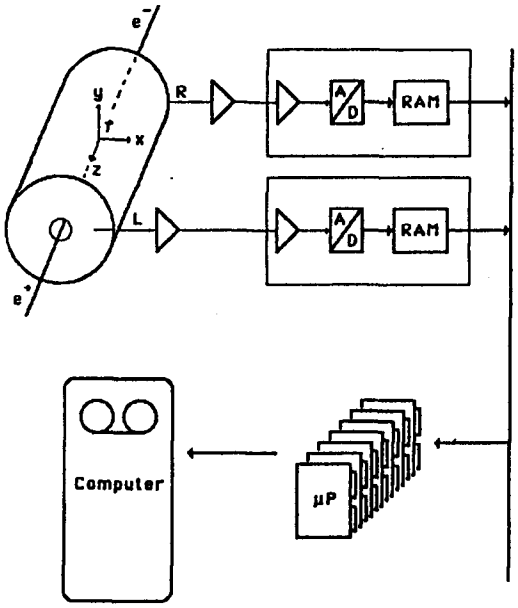


Fig. 2 The readout of the JADE drift chamber with Flash-ADCs followed by microprocessor pulse analysis.

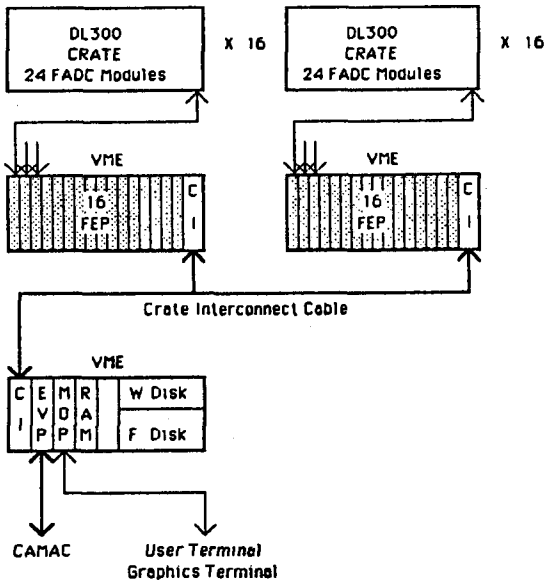


Fig. 3 The three hardware levels of the readout system.

Components

The main components assembled in a DL300-crate and their functions are summarized below, details have been described elsewhere [1].

(i) The FADC-Module is a fast (100 MHz), 4 channel analog to digital converter based on monolithic 6-bit-flash-ADCs followed by 256 word deep memory. The FADCs are operated in a mode with a non-linear response function resulting in an extension of the dynamic range by a factor of 4 in the lower amplitude region. 24 FADC-modules for 48 wires (96 channels) reside in one crate.

(ii) The SCANNER- and HIT DETECTOR-Module is the programmable control unit in a DL300 crate and can operate in two basic modes: The 100MHz SAMPLING-mode for digitisation of the drift chamber signals and the 25MHz FASTSCAN-mode, where all data in the FADC modules are inspected for valid hits.

(iii) The VME-Interface-Module is the interface to the M68000 processor for data and address transfers. The SCANNER signals "End of sampling" and "Hit detection" are available in a status word and generate an interrupt at the connected CPU.

The FEPs [6] are commercial single width VME units with a 128 kbyte RAM. The access to the associated DL300 crate is provided by a piggyback card which plugs into the CPU board and uses the P2 connector at the back of the module. The memory of the FEPs is dual ported and placed at non overlapping VME base addresses. A memory access in conjunction with a special VME address modifier is used to generate an interrupt at the parallel port chip of each CPU (mail interrupt).

The connection to other VME crates is established by the Crate Interconnect module (CI) [7] which is used to open a 64kbyte window into the memory of another crate.

The EVP consists of a CPU of the same type used for the FEP. It is connected to the module driving the Auxiliary Crate Controller Interface (ACCI) [8] presenting the data in their final format in a CAMAC unit to the host computer of the experiment.

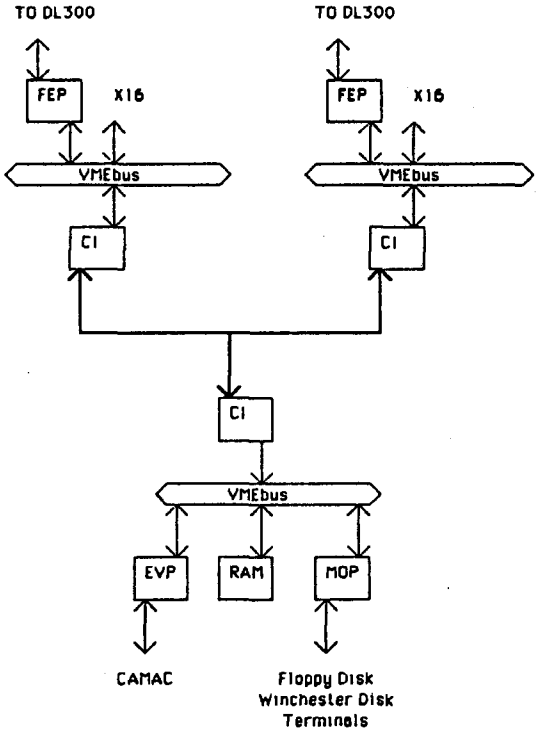


Fig. 4 The components of the system and their connection to the different VME buses.

The MOP is housed in the same VME crate as the EVP. Its task is to supervise the data flow and to collect and store data and calibration constants. While otherwise identical to the other processors, this CPU-board holds a memory of 256 kbytes which is further increased by a separate 0.5 Mbyte VME-RAM unit contained in the crate. This CPU is connected to a 27 Mbyte Winchester disk and a floppy disk drive. It runs under CP/M68K while all other CPUs do not require an operating system. An ASCII-terminal and a graphics

terminal are connected to the serial ports of the MOP.
Figure 4 shows the different buses of the system.

Readout Scheme

Under the control of the SCANNER- and HIT DETECTOR module, the hardware FASTSCAN [1] starts at the end of the SAMPLING phase to look for valid hits in the memories of all FADCs. On detection of a valid hit, the FASTSCAN is paused and a CPU read cycle is invoked for all data bins related to the hit. The SCANNER continues the search for new hits while the CPU itself reads the data. The total transfer time is about 3ms, where the exact value depends on the number of hits, and is entirely dominated by the time required to read the data into the FEP. In the case where no hits are found in a given crate, the FASTSCAN is completed within 0.5 ms.

The pulse shape analysis is the most time consuming part of the readout chain requiring 1-3 ms per hit. However, once the pulse has been read into the dual ported memory it is accessible to all 16 CPUs in the same VME crate. To profit from the computer power available we have adopted the solution where each CPU can analyse not only its own hits but can help to analyse all hits stored in the other CPUs of the same crate. In this case only the average hit population of 48 wires determines the analysis time per crate, while in the case where each processor only deals with data from its own front end crate the highest hit population for a group of 48 wires is relevant for the processing time. The gain amounts in JADE to a factor of 4, leaving 8 hits on average for each processor. To exclude interference between different processors each hit has to be protected by a semaphore (Fig.5). Even though the hits of one CPU may have been analysed by different CPUs the results are stored back into the memory of the originating CPU, thus facilitating the ordering of hits and assembly of the event record after pulse analysis (Fig.6).

The JADE event format implies a fixed readout sequence for the wires and hence for the data from the processors. Therefore, the selection of the next hit to be analysed follows this predefined order and gives the highest priority to the analysis of the data stored in the first processor to be read.

The EVP event readout and transfer to the host computer of the experiment starts synchronously with the ongoing hit analysis.

Optionally the MOP may request the full Flash-ADC information of selected channels for debug and performance tests of the hardware and algorithms. The additional CPU load placed on a FEP by such a request is easily compensated by the other processors in the same VME crate.

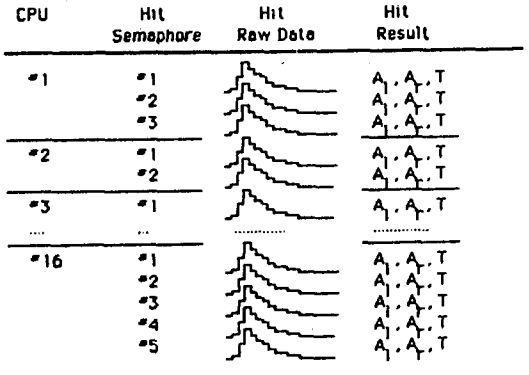


Fig. 5 Data Structure for raw and result data in the different CPUs and the protection by a semaphore.

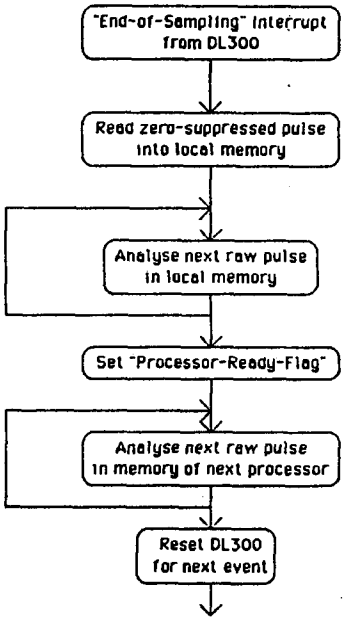


Fig. 6 The different readout stages for the FEP processors.

Pulse Analysis

Several pulse shape analysis algorithms have been investigated [1,9,10,11]. We have restricted our selection to those algorithms that calculate a weighted center of gravity of a small number of samples around a characteristic bin of the FADC pulse distribution. A bin showing high stability against fluctuations of the pulse shape is, for example, given by the bin with the largest positive derivative. To achieve the best resolution we empirically determine the optimum set of constant weights for the time estimate. Qualitatively, the result of such minimization emphasizes the early part of the pulse which contains the information of the first arriving electrons. However, the detailed distribution of weights, especially the inclusion of the falling part of the distribution has to be optimized with chamber data itself. 4 bins (40 ns) are sufficient to deduce a drifttime estimate accurate to 2 ns (Fig.7). The intrinsic time resolution of the FADC is better than 1 ns as has been demonstrated by comparing the drift times measured at both ends of a wire.

The DOS (difference of samples) algorithm [10] is based on calculating the weighted center of gravity of the differentiated pulse. After optimizing the weights for this method we arrive at a resolution which is equal or even better than that for the center of gravity of the direct distribution. Since this method is fast and has excellent properties for detecting a second hit we adopt this method for calculating the time of the pulse.

The amplitude is found by integration of the pulse for a fixed number of bins (150 ns). For overlapping hits the individual amplitudes are determined in the following way: A norm pulse is fitted to the unobstructed part of the first hit and extrapolated into the region of the overlapping second hit. The extrapolated part of the fitted pulse is then added to the charge integral of the first hit and subtracted from the second hit.

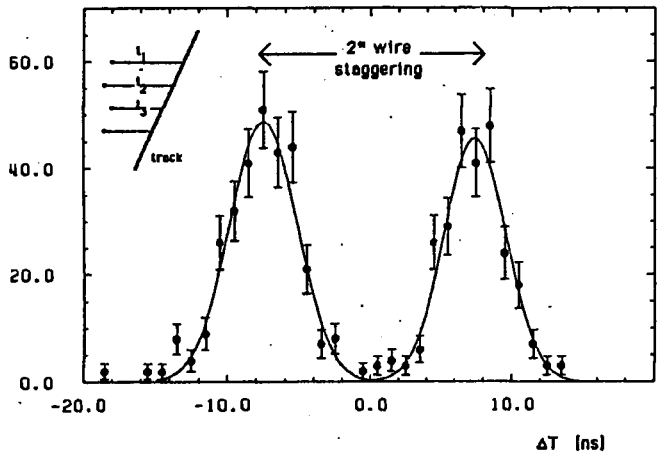


Fig. 7 The distribution of the residual $\Delta T = \frac{1}{2}(t_1+t_3)-t_2$ for three wires as measured with the prototype of a JADE cell. The double structure is caused by the wire staggering and shows that the left/right-ambiguity can already be resolved on the basis of a wire triplet.

Communication between processors

Communication between processors is necessary at several stages:

Parallel processing of all hits requires easy access to all hits read out by the CPUs. With all of the memory being dual ported, access to data in another CPU appears as a simple memory access.

The event building processor (EVP) uses the Crate Interconnect module to open up a transparent 64kB window at an arbitrary memory position of a front end processor to transfer the results of the individual pulse analyses.

The monitoring processor (MOP) uses the same module to distribute programs at system startup.

An important aspect of the communication between processors is the possibility to issue mail interrupts to an outside CPU. We use a VME bus memory access at a certain address in conjunction with a special address modifier to generate the interrupt used to initiate a certain task in such a CPU (mail interrupt).

System Startup

The system is started from the MOP either through operator intervention or automatically through a system reset initiated by the host computer. Downloading of the different programs and data components into the other processors proceeds according to a routing table held on the disk of the MOP. This table in ASCII format can easily be edited with the system editor and thus be adapted to the different program configurations necessary for system calibration, debug and standard data taking. An initial mail interrupt, issued sequentially to the CPUs involved, starts execution of the individual programs after download. The time required to setup the entire system amounts to a few seconds only.

Software Aspects

All code is written in high level FORTRAN 77 [12] especially extended for real time applications. Among the many hardware oriented extensions the most important ones for our purposes are:

(i) Support of memory mapped I/O with special language constructions for both static and dynamic assignment of variables to memory positions and

(ii) Reentrancy of the code to allow for sharing of libraries and direct use of interrupt routines without the extra overhead of an operating system.

Many other extensions with respect to semaphore handling, explicit register access, full M68020 and coprocessor support are available for the user.

Program Development

The programs are written, compiled, linked and executed on the VME-system itself. Both the VAX-like editor and the compiler are written in FORTRAN 77. File access is provided by CP/M68K. Using only one language for both system and application software greatly simplifies the development of the readout system. The actual programming effort for the entire system could only be kept at comparatively low level due to the availability of a known high level language and the valuable extensions necessary for real time applications. Standard software packages like Mini-ZCEDEX, Mini-GD3 and Mini-HBOOK are running on the system.

Conclusion

The JADE experiment is presently being equipped with fast Flash-ADCs for drift chamber readout. The huge amount of data produced by such a system can effectively and fast be reduced online by a combination of hardware zero suppression and additional pulse shape analysis in a system of VME processors. The architecture of the VME bus allows full use of the available CPU power for parallel processing, while the implementation of such a system is simplified by a high level FORTRAN 77 compiler with real time extensions. Owing to these two aspects of the system the effort on the VME side of the readout could be kept at a very moderate level.

Acknowledgements

We would like to acknowledge the contributions of all members of the electronics workshop of the Institute where this project was realized, in particular R. Rusnyak and K. Bielefeldt. We wish to thank Profs. J. Heintze and G. Weber for the continual support that made this project possible. This work was supported by the Bundesministerium für Forschung und Technologie.

References

[1] P.v. Walter, G. Mildner, "A Multichannel 100 MHz Flash-ADC System with Fastscan Zero Suppression", IEEE Trans. Nucl. Science NS-32, 626 (1985)
[2] J. Heintze, "The Jet Chamber of the JADE Experiment", Nucl. Instr. and Meth. 196 (1982), 293
[3] "The OPAL Technical Proposal", CERN, LEPC 83-4
[4] W. Farr, R. Heuer, A. Wagner, "Readout of drift chambers with a 100 MHz Flash-ADC System", IEEE Trans. Nucl. Science NS-30, 95 (1983)
[5] W. Farr and J. Heintze, Nucl. Instr. and Meth. 156 (1978), 301
[6] CPU 07, MicroSys Electronics GmbH, Munich, W. Germany
[7] E. Pietarinen, "VME-bus Interconnect Manual" Internal note, Dept. of Physics, Univ. of Helsinki
[8] V. Mertens, "68CAMICRO - ein leistungsfähiger Experimentrechner auf Mikroprozessorbasis", Bonn IR-83-10
[9] P. Bock et al., "Drift Chamber Readout with Flash-ADCs: 1) Time and space resolution", to be published in Nucl. Instr. and Meth.
[10] D. Schaile et al., CERN-EP 85-95 (1985)
[11] J. Spitzer, JADE Internal Note
[12] H.v.d. Schmitt, RTF/68K, Real Time Fortran 77, Manual of Compiler and Run-time library.

MLLE - A DATAFLOW CONTROLLED MULTIPROCESSOR SYSTEM

G. Beier, L. Kappen, R. Lutter, K. Schöffel, B. Stanzel, K. Steinberger, and H. Wilhelms
Beschleunigerlaboratorium der Universität und Technischen Universität München
D-8046 Garching, Federal Republic of Germany

Abstract

MLLE is a VMEbus based multiprocessor system for data acquisition and processing in nuclear physics applications. A large memory allows megachannel histograms. The system interfaces CAMAC via a special auxiliary crate controller. The DAMOS operating system features dataflow controlled processing using about 10 microprocessors operating in a non hierarchical mode.

Introduction

In 1977 we started in our laboratory with a multiprocessor system called MADAME (Million channel Analyser for Data Aquisition of Multiparameter Events) [1]. It is still used for online accumulation of large multidimensional histograms. Since MADAME allows a maximum of not more than 4 ADCs and programming is done completely in assembler (TMS9900), the range of applications for MADAME is too small. This fact, and the age of our current data aquisition system (PDP15) lead us three years ago to the decision to develop the MLLE [2].

To avoid waisting manpower each time when changing from a previous data aquisition system to the following, for system implementation only high level languages and, if possible, standard operating systems should be used. We chose C as programming language and UNIX as operating system. Application software can also be written in FORTRAN 77.

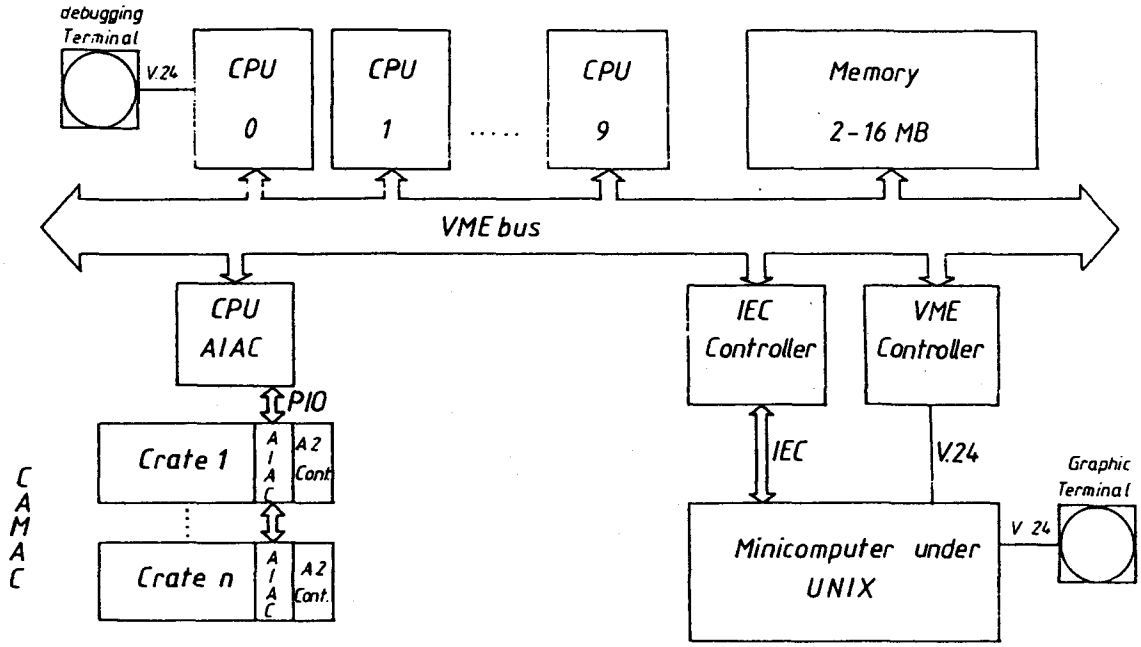


Figure 1: Blockdiagram of the MLLE

Hardware

MLLE is realized as a VMEbus system (Fig. 1) and contains a VMEbus system controller, up to 10 processing units, an IECbus controller, and up to 16 Mbyte memory used by all CPUs as a global memory. The AIAC processor is necessary to interface CAMAC. At present we use MVME121-boards. Each board contains a MC68010 CPU with 10 Mhz clock rate, 512 Mbyte local memory, up to 128 Kbyte EPROM, a MC68451 MMU, a serial debugging port, 4 timers, and 4 switches on the front panel for board identification. Communication between host (CADMUS 9230) and MLLE is done via the IEEE 488 bus link.

DAMOS

The software of MLLE consists of several components organized in different layers. The main components are ULRIKE [3] (UNIX like runtime kernel), the application software divided in tasks, and DAMOS.

DAMOS is a multiprocessor operating system and runs on all processors in identical form. There

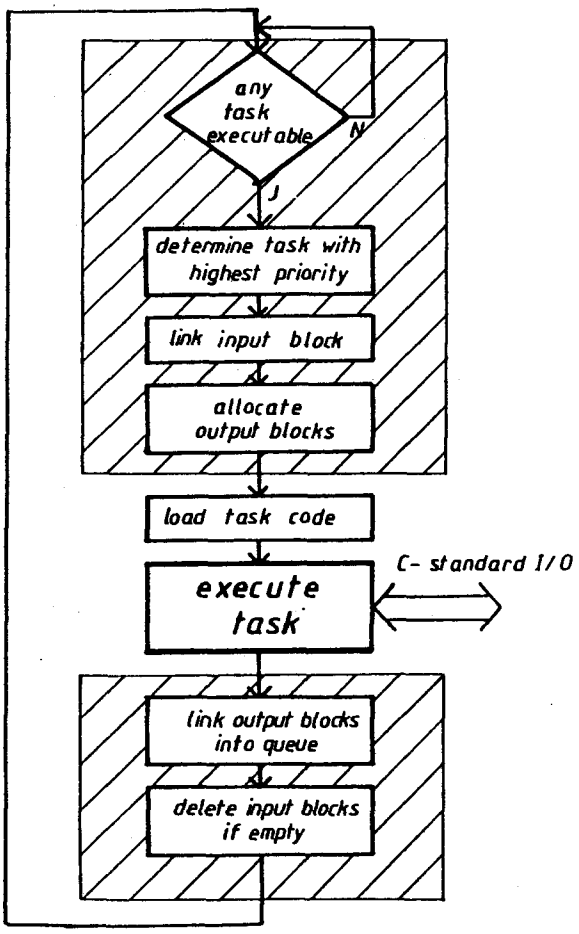


Figure 2: DAMOS processor loop

is no master in the system. One processor handles the communication with the debugging terminal for printing error messages or giving START or STOP commands. It is also possible to have the debugging terminal at the system controller or via the system controller at the host.

All processors are running in the processor loop (Fig. 2). The decision which task may run next cannot be done by several processors in parallel. With the indivisible TAS (test and set) instruction it is possible to use semaphores for processor synchronisation. With this method it can be achieved that the critical part of the processor loop will be executed by only one processor at a time.

In the normal case each task reads incoming data and writes outgoing data, thus functioning as a consumer and producer of data packets. If the executable task with highest priority has been found, from each input queue one data block is retrieved and linked to the task. The task code will be copied from the global to the local processor memory only, if it isn't already there.

It is possible, that one task will be executed by several processors simultaniously, each working on a separate block from the input queue and producing its own data for the output queue. If the output buffer is full or the task finishes processing, the output blocks will be linked to the output queue.

Example of a simple net

Let us consider as a nuclear physics application a γ-γ-coincidence between two detectors. Fig. 3 shows the graphical representation of the corresponding net. The net contains queues of data packets, tasks, and histograms. These items are all stored in the global memory.

The net describes the dataflow through MLLE:

> The raw data from CAMAC reaches, the global memory via the AIAC CPU in the MLLE. Now the task **hostcomm** sends the **rawdata** on the IECbus to the host to save it on magtape for later analysis. The **rawdata** will also be processed immediately by the task **filter**, where window conditions will be tested and data will be sorted into the queues **d2** and **d3**. Finally these data packets will be accumulated to two-dimensional histograms by the tasks **accu2** and **accu3**. The task **accu1** accumulates the **rawdata** directly to one-dimensional histograms.

The **rawdata** appears three times as queue in the diagram but it is stored physically in the global memory only once. Therefore a data packet in the queue **rawdata** will not be physically removed before the tasks **accu1**, **filter** and **hostcomm** have finished processing of this data.

In contrast to usual multitasking systems, in which each task has a fixed priority, DAMOS computes the task priority dynamically: Priority increases with the number of data packets in the input queue and decreases with the packets waiting in the output queue. If in the given example the task **filter** needs too much processing time the queue **rawdata** would increase,

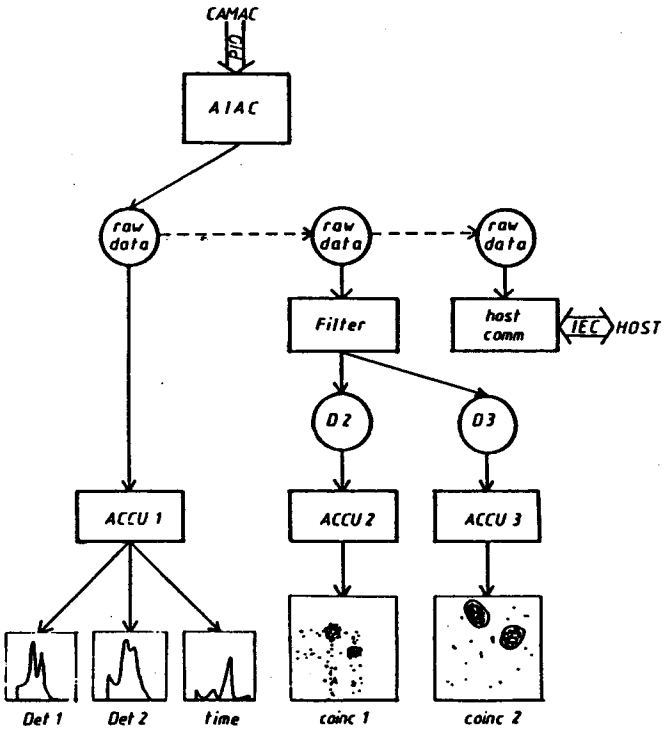


Figure 3: Graphical structure of a simpl net

```
aiac > rawdata

host < rawdata

accu1 < rawdata > det1 det2 time

filter < rawdata > d2 d3

accu2 < d2 > coinc1

accu2 < d3 > coinc2

#spec det1   1 2 0,2047
#spec det2   1 2 0,2047
#spec time   1 2 0,1023
#spec coinc1 2 2 0,511 0,511
#spec coinc2 2 2 0,511 0,511
```

Figure 4: Description of this net

the queues d2 and d3 would become empty. But if this happens, the priority of this task would increase fast and therefore several other processors would run this task.

The way to define such a net is very simple (Fig. 4). The first name in a line gives the name of the task and the task code file in the host. Names after "<" are the names of input queues, after ">" of output queues. Lines beginning with #spec define histograms. The arguments are: dimension, bytes per channel, first and last channel for each dimension.

The MLLE may also be able for later analysis of the raw data stored on tape. In this case it is only necessary to cancel the line:

aiac > rawdata

and to change the line:

host < rawdata to **host > rawdata.**

Literature

[1] B.Stanzel et al., IEEE Transactions on Nuclear Science, Vol. NS-28, No 5, Oct.81, Page 3892

[2] G.Beier et al., IEEE Transactions on Nuclear Science, Vol. NS-32, No 4, Aug.85, Page 1426

[3] G.Beier et al., Cross Software Development in a UNIX Environment, talk given at this conference.

USE OF A MOTOROLA 68000 VMEbus BASED SYSTEM FOR FAST READ-OUT OF A RETICON PHOTOSENSITIVE ARRAY

D.R.N. Jeffery,P.Buttner,R.S.Gilmore,T.K.Gooch,
W.L.Kwan,T.J.Llewellyn,I.C.McArthur,J.Malos,
J.P.Melot,R.J.Tapper.

H.H. Wills Physics Laboratory, Royal Fort,
Tyndall Avenue, Bristol, England.

October 1985

Abstract

The read-out system is built on 9u size boards in a Eurocrate with VMEbus J1 and J2 dataways and uses a Motorola 68000 processor to control the read-out of a Reticon based camera. The read-out normally runs asynchronously at a regular rate to give a constant noise from charge leakage and gives a fast rejection of each frame read unless it is accompanied by an external accept signal.

Accepted events are transferred to an LSI 11/23 computer for display after background subtraction.

Frames are read by DMA directly into the memory at a rate of one pixel per five hundred nanoseconds, which is limited by the speed of available memory, and then only selected events of interest are transferred at relatively low rate to the display. The system has been tested by selecting events from an optical read-out avalanche chamber where a cosmic ray track has passed through the active region. It will be used for the optical read-out of scintillating fibres.

1 Introduction

At Bristol we have developed a fast two dimensional optical readout system. This has been developed to study the light from an electron avalanche chamber and also from scintillating fibres.

The avalanche chamber was constructed as part of a Cerenkov ring imaging detector [1], for which the camera readout system was initially developed. Light from the Cerenkov radiator was focused onto the calcium fluoride input window of the avalanche chamber. The avalanche chamber acts as an image intensifier. Photoelectrons are multiplied to produce avalanches which emit light from atoms excited by the avalanche [2]. This light was then focused onto the photocathode of a Mullard XX1500 type image intensifier. A Reticon integrated photodiode array was coupled to the phosphor screen of the image intensifier via fibre optics and the digitised data from the pixels was read out with the VMEbus 68000 system, Fig 1.

Further developments have lead to the camera being used in a prototype detector constructed from scintillating glass fibres [3]. At present a bundle of fibres made from GS1 glass is used as an interaction target. Light from the fibres was fed away from the beam to a diode type image intensifier. This is an intensifier without a channel-plate; it consists of a thin alkali metal photocathode and a phosphor screen. Electrons liberated at the photocathode are accelerated in an electric field and collide with the phosphor screen to generate light [4]. This diode type image intensifier has a good quantum efficiency and a clean single photoelectron peak, but has fairly low gain. Coupled to the phosphor screen of the diode intensifier is a high gain gated channel-plate type

intensifier, which is normally gated off to decrease any photocathode noise. Coupled to the exit window of this intensifier will be the Reticon device.

2 The data acquisition system

The data acquistion system consisted of an LSI 11/23 computer and a VMEbus based 68000 system, which acted under program control of the 11/23. Communication between the two computers was by means of a 16 bit bidirectional parallel interface.

2.1 The solid state camera

The camera is based on an EG+G Reticon device RA100 or RA256, Fig 2. This is a silicon device with a square array of photodiodes, which can be read out sequentially and is in operation unlike a CCD. The rows of photodiodes are accessed sequentially and each row is shifted only once into a BBD type output register. This circumvents the problems that CCDs have of light arriving while the shifting process is in operation, and consequently appearing in the wrong place in a reconstructed image. The RA256 device has a pixel centre to centre spacing of 40 microns and a total active area of 12mm by 12mm.

The Reticon device had a fibre optic coupling piece bonded to its surface to allow uniform illumination of the whole active surface. Previous use of a CCTV lens had proved inadequate due to severe transmittance variations across the lens surface. The fibre optic protuded about 1 cm from the ceramic chip package to allow easy optical mating with the device to be viewed.

The active elements of the Reticon device are reversed bias photodiodes, Fig 3, with a reset charge stored its inbuilt capacitance while in the reversed biased state. Electron-hole generation by photons falling onto the pixel create a charge deficit on the reset level and this is the signal sensed at the output. Electron-hole pairs are generated thermally, and in the absence of any signal there is still a leakage current across the photodiodes. This gives rise to a fixed pattern noise in the output (assuming the readout time is constant). The problem of this pattern noise was minimised by first taking a background frame (by averaging 16 frames) and subtracting this (online) from every frame transferred to the LSI 11/23.

The signals from adjacent pixels are fed to two output stages on the chip, and then onto an "OUTPUT PROCESSOR" board. This board buffered and amplified them and provided correlated doubling sampling. The signal emanating from the chip consists of the wanted signal and a bias voltage superimposed onto the bucket-brigade reset voltage, which was found to drift with time. To eliminate this drift from the final signal the technique of correlated double sampling was employed. Here this technique was implemented by the first section of the circuit, shown in Fig 4. After the output amplifier of the Reticon has been reset and before the next signal charge is sampled the point B is reset to a clamp voltage by pulsing on the mosfet T1. Thus when the next signal charge is transferred into the output amplifier the point 'A' goes to the voltage Vreset-(Vbias+Vsig) and the point 'B' goes to Vclamp-(Vbias+Vsig); hence the term Vreset is eliminated from the signal amplified by further stages in the "OUTPUT PROCESSOR" board, Fig 5.

Clock and control signals for the chip were provided by a separate camera controller board which was connected to the camera by ECL level twisted pair cable. The clock signals were buffered by a "DRIVER" board before being applied to the chip, bias levels were also provided by this board.

The signals were then fed to a TRW TDC1048E1C flash ADC card, which fed its digitised output down an ECL twisted pair to the camera controller.

The two digitised values for the pixels were latched on the camera controller before being passed via another ECL twisted pair to the VME system. When running the frames from the camera were read continuallly into the VME memory.

2.2 The VMEbus system

The VME system is a mixture of commercially available components and special units built at Bristol. The prototype units were constructed on standard double extended eurocards, but their size proved to be a severe limitation on the functionality of a board. One unit, the DMA controller, was built on two double extended eurocards, interconnected with a ribbon cable, which was far from satisfactory. So a new size, 9U, was adopted for construction.

To allow double extended eurocard components to be used the VME crate was divided into two sections. The left hand half was built to take these cards, and the right hand half to accommodate 9U boards.

The backplane specifications limit the current that can be drawn from the 5 volt line for a single slot to 3 amps, which for a 9U size board filled with TTL logic would prove

insufficient. So an extra power connector was added below the J2 connector, and a separate 'power bus added to the crate. This also allowed for separate analogue power lines to be supplied to a board.

The units for the VME system consisted of an 8 MHz M68000 cpu board and a 1/2 Mbyte dynamic RAM card. The following were specially constructed for this application :

DMA controller.

16 bit parallel interface.

NIM trigger unit.

2.2.1 The DMA Controller

This provided the interface between the camera controller and the VME memory. The board contained logic for bus mastership acquisition and, for handshaking to control the flow of data from the camera controller board, and to write the data into the system memory. The mastership of the bus was gained by the circuit in Fig 6. Under program control the Q output of flip-flop U1 was set. this was fed to the VME signal BR1* to request mastership of the bus. The 68000 responded with the BGACK* signal which was latched by U2 when AS* was high (at the end of current bus cycle). This ensured that BBSY* was not asserted during the bus cycle by the CPU which originally set BR1*.

The DMA was a block transfer of a single frame from the camera, which for the RA256 device consisted of 64 Kbytes. To increase data throughput the data was transferred as words, and at the end of a frame or upon an error condition occurring the bus mastership was relinquished.

The flow of data from the camera controller to the DMA controller was controlled by 'handshake' signals. The handshaking allowed an asynchronous flow of data from the camera to the VME system, the data rate being limited by the system memory speed. In practice the rate was determined by the time taken for a memory write cycle when a refresh cycle was in progress, which in our case was 1 microsecond. The data rate could therefore be vastly improved by using a fast static memory board for the ring buffer data area.

The handshaking between the camera and the controller consisted of four signals, Fig 7.

Two signals 'ENABLE' and 'SEND' are driven by the DMA controller and the 'RECEIVE','EOF' signals by the camera controller. The 'ENABLE' signal signified that a frame of data was required, the 'SEND' signal requested a 16 bit word of data from the camera controller. In response to the 'SEND' signal, the camera controller placed a word of data on the cable, and, after a delay to allow for propagation time on the cable, asserted 'RECEIVE'.

After latching the data on the DMA controller board 'SEND' was removed, which in turn allowed 'RECEIVE' to be removed, and a bus cycle was then initiated to write the data to memory. Upon completion of the bus cycle 'SEND' was again asserted to request the next word of data.

The 'EOF' signalled the end of a frame and arrived after the last pixels were transferred.

Fig 8 shows the relationship between the handshake lines and the data flow.

2.2.2 Parallel interface

This device was also constructed to provide a bidirectional interface between the VME system and the 11/23. A commercially available DRV-11 interface was purchased for the 11/23. The interface constructed interface has TTL logic levels and contains a four line handshake system to control the data flow at a rate of 100 Kwords per second.

2.2.3 NIM trigger unit

As NIM logic is used extensively in high energy physics, a NIM level input/output module was constructed . It consists of four self-latching inputs and four pulsed outputs. The unit can be configured to interrupt the 68000 upon reception of a combination of input pulses (defined by a loadable bit mask) or to latch the lines so the unit can be polled.

In practice the unit was polled because of problems associated with the asynchronous nature of interrupts.

3 Conclusion

The apparatus has been operating reliably for almost 18 months. The readout time for one pixel is 500 nS and a frame from the RA256 can be read in 35 mS. The transfer time of a full frame to the 11/23 is approximately 2 seconds, a compressed frame is transferred considerably quicker.

There are limitiations with its operation : the lack of ability to have a triggered read and the speed of the readout.

It would prove very difficult to provide a triggered read due to the design of the Reticon chip itself. Although it has

a frame-reset control this is not used due to timing requirements in its application.

With the current DRAM in the VME system, the readout speed is limited to 1 microsecond per 16-bit word. This is much slower than the access time of the memory, but allows for increased access time when a refresh cycle is in process.

Cosmic ray tracks and pair production from cosmic rays have been seen in the avalanche chamber, but the camera has not yet been successfully used on the scintillating glass detector.

4 Future improvements

A fast-access static memory board is under construction which will allow the reticon to be clocked at its maximum rate of 5 MHz. A unit which will allow background subtraction and frame compression 'on the fly' is also under construction, and this will speed up the data rate considerably. A stand alone VMEbus system with its own integral 40 Mbyte Winchester hard disk drive and graphics capability is being developed. This can boot an operating system from the hard disk and will allow data acquisition to be independant of the LSI 11/23.

5 References

1. R.S.Gilmore et Al. An optical readout for accurate positioning of UV photons in an avalanche chamber. NIM 206 p189 (1983).

2. T.K. Gooch Ph.D. Thesis. Bristol University (1984).

3. T.K. Gooch et Al. A multistep avalanche chamber with optical readout for use as a 2-dimensional vuv photon or particle track detector. To be published in NIM.

4. R.Ruchti et Al. Scintillating glass fibre-optic plate detectors for active target and tracking applications in high energy physics experiments.

5. Proxifier BV 25 reference sheet. Proxitronic, Rudolf-Diesel StraBe 23, D-6108 Weiterstadt.

6. Reticon RA256 preliminary data sheet. EG+G Sunnyvale California.

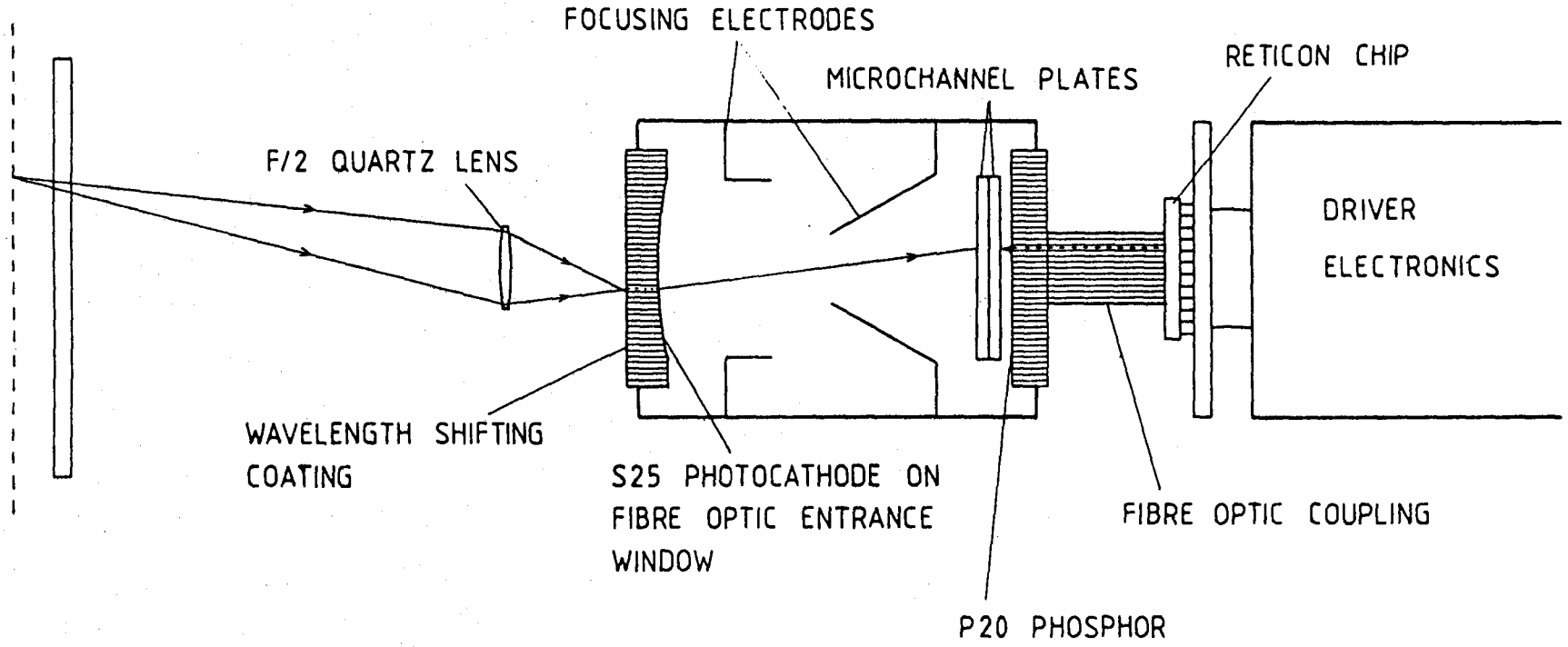


IMAGE INTENSIFIER TUBE

Fig. 1

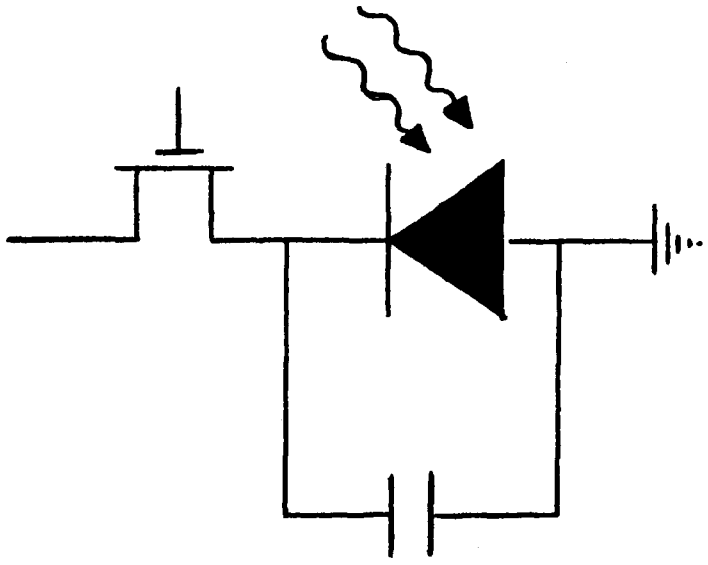


Fig. 3

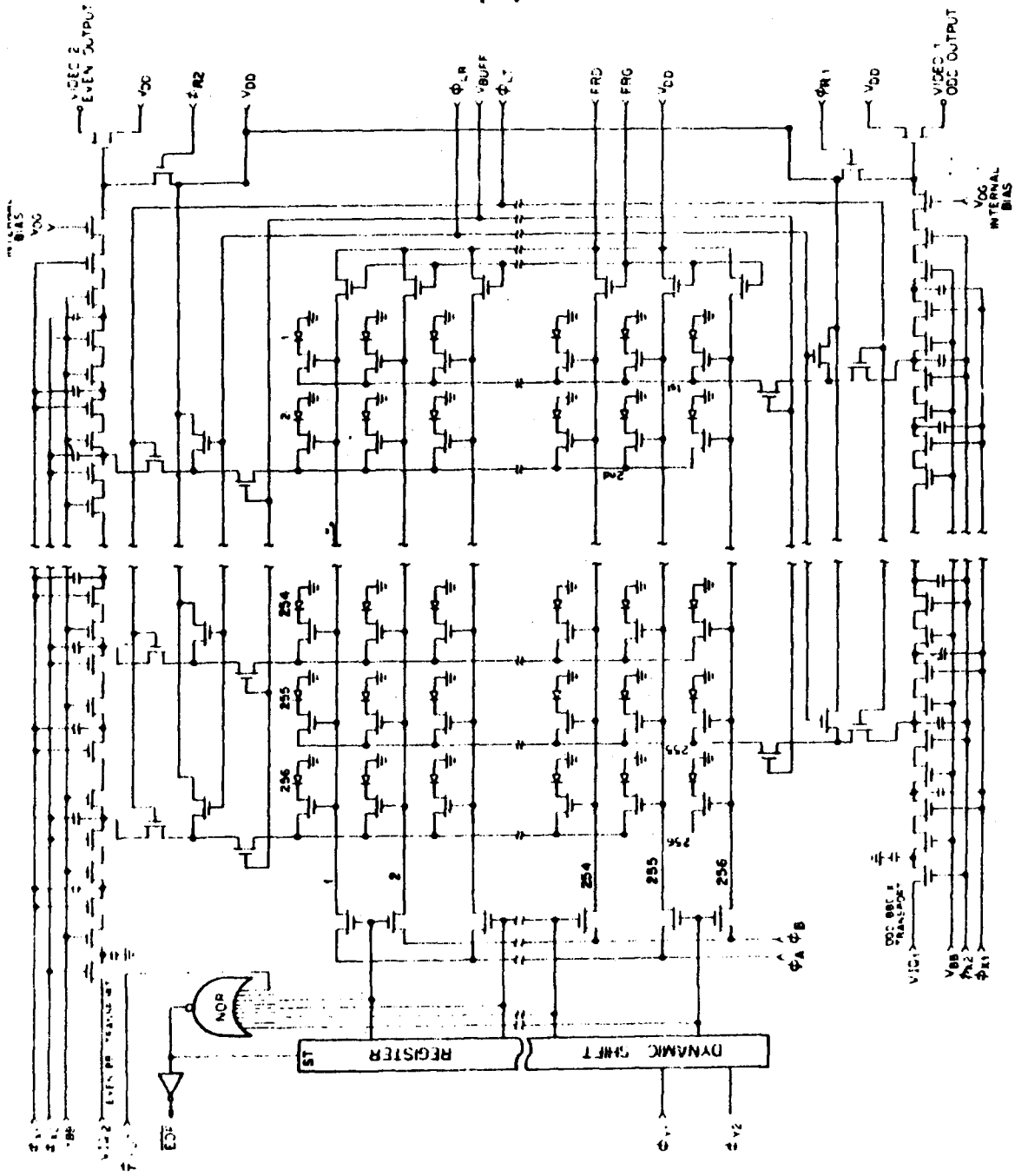


Fig. 2

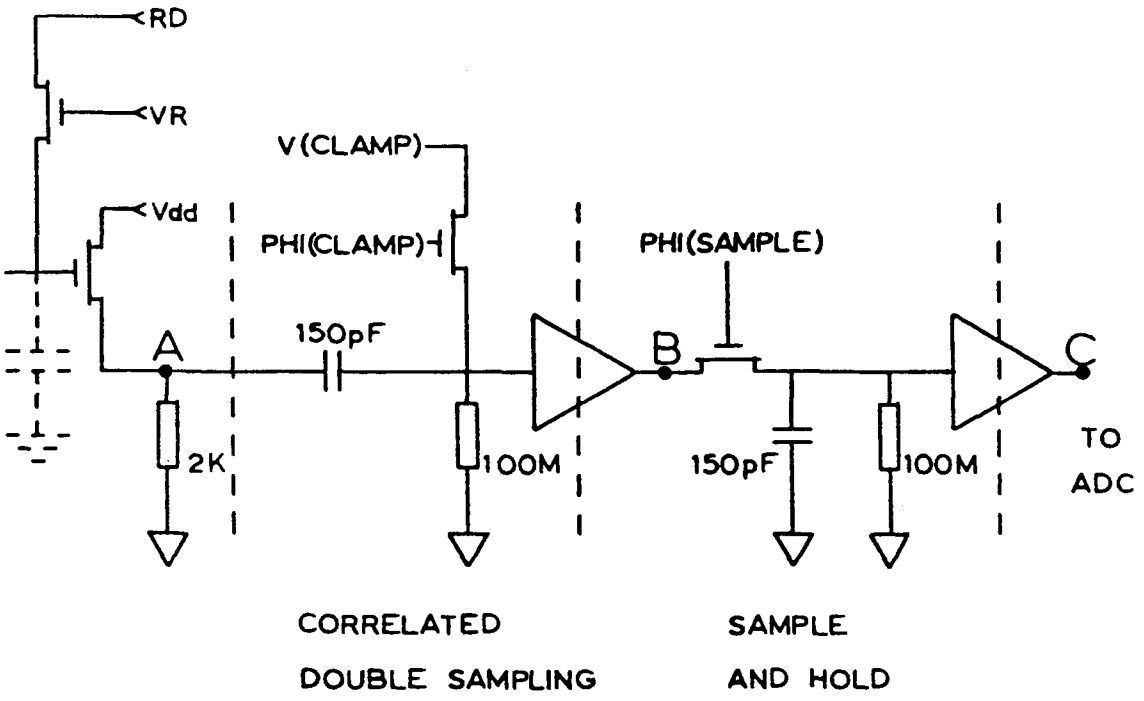

RD
VR
V(CLAMP)
Vdd
PHI(CLAMP)
PHI(SAMPLE)
A
150pF
B
C
2K
100M
150pF
100M
TO
ADC
CORRELATED
DOUBLE SAMPLING
SAMPLE
AND HOLD

Fig. 4

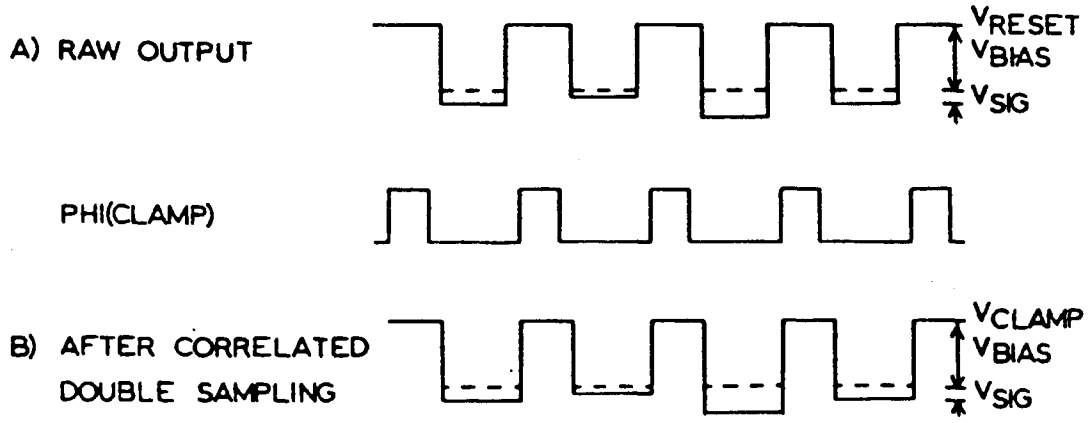

A) RAW OUTPUT
VRESET
VBIAS
VSIG
PHI(CLAMP)
B) AFTER CORRELATED
DOUBLE SAMPLING
VCLAMP
VBIAS
VSIG

Fig. 5

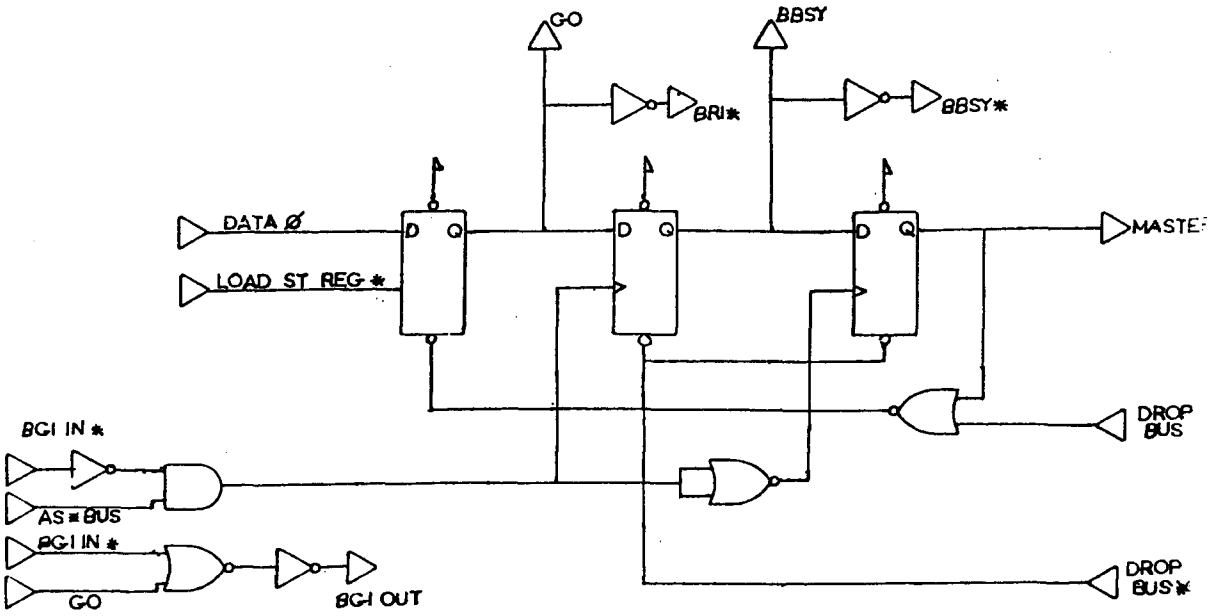


Fig. 6

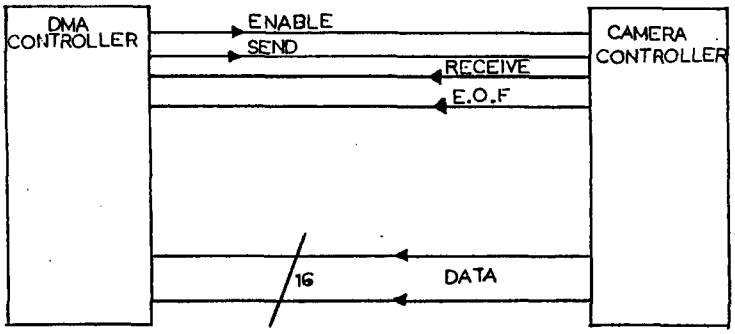


Fig. 7

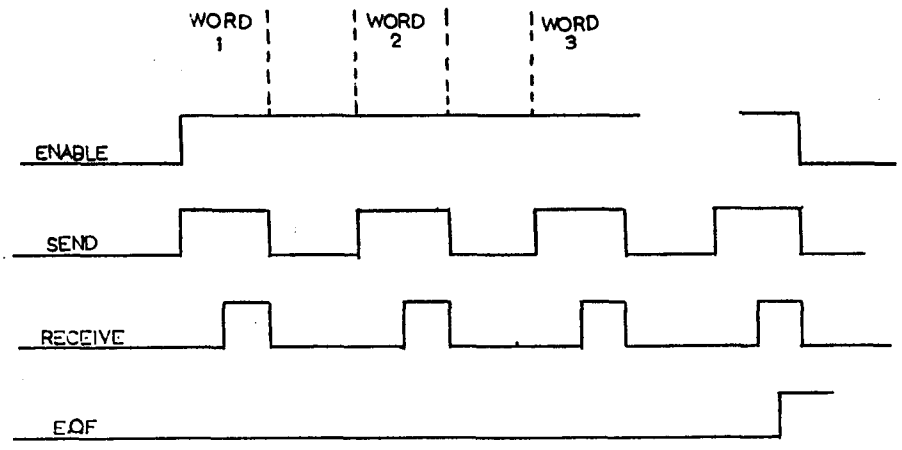


Fig. 8

LOSS-FREE GAMMA-RAY COUNTING ON THE VMEbus

M. M. Minor, E. B. Shera, and J. W. Lillberg
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

ABSTRACT

An add-N histogram memory controller has been designed to histogram gamma-ray data using conventional pulse-height spectroscopy ADCs into VMEbus memory. The weight factor, N, is derived from the instantaneous counting losses present in the system.

1. Introduction

Loss-free gamma-ray counting is a technique of correcting for system counting losses in real-time. The technique is particularly useful when measuring mixed radionuclides with very short half lives. In order to perform accurate quantitative pulse-height analysis measurements of short-lived neutron activation products under conditions of rapidly varying count rates, we have designed a loss-free counting module which interfaces pulse height nuclear ADCs to VMEbus memory. Several techniques for real-time correction of counting losses have been developed.[1-3] All employ add-N histogram memory where the integer weighting factor, N, is derived from the instantaneous counting losses present in the system.

The loss-free counting module provides the basis for a replacement control and data acquisition system at the Los Alamos Omega West Reactor. The system which has been in use for a number of years has been described elsewhere.[4] In short, the system is designed to automatically control sample irradiations and gamma-ray counting. The system utilizes a PDP-11* computer, a UNIBUS* programmable CAMAC branch driver, and several CAMAC crates. It is planned to replace all the above with a VMEbus based microcomputer. A VMEbus

* PDP-11 and UNIBUS are trademarks of the Digital Equipment Corporation.

system provides the versatility and excellent mechanical specification desirable for a replacement system. Most VMEbus system components required for this system are now commercially available; a suitable ADC interface is not.

2. Real-Time Correction of Counting Losses

Real-time correction of counting losses requires the generation of consecutive weighting factors W(t) of high statistical accuracy in millisecond time intervals. The weighting factors are generated from a clock signal (4MHz SERCLK) and a gating circuit which transmits clock pulses to a preset counter only during live-time intervals of the pulse-height analysis system. The number of clock ticks accumulated during this time in a second counter is equal to the value of the preset counter divided by the system live-time. The instantaneous weighting factor is just the accumulated clock ticks divided by the preset (gated) counter value. In order to generate an integer weight, the integer part of the weighting factor is added to histogram memory and its fractional remainder is added to the next weighting factor thus forming a new integer approximation of the true weight with each stored conversion.

As with any live-time generation system, the accurate derivation of system busy is crucial. We have included provisions for generating an extended busy signal as described by Westphal[5] in his Virtual Pulse Generator Method, and have utilized the same weight generation methods.

3. Loss-Free Counting Module

An add-N histogram memory controller was designed to histogram gamma-ray data using conventional pulse-height spectroscopy ADCs with a VMEbus computer system. As shown in Fig. 1, the controller is designed to interface to VME/VMXbus compatible dual ported memory. Our application utilizes a commercially available 128K byte static RAM module.[6] A block diagram of the module is shown in Fig. 2. The module employs a single VMEbus control register which specifies storage memory offset, ADC enable lines, and mode of operation bits. Mode bits allow add-1 or add-N histogramming and enable or disable extended busy generation.

The module uses an 8-bit adder with carry increment in order that memory may be incremented by weight factors of 1 to 255 in the modify portion of the

VMXbus Standard Indivisible Single Address Cycle. This allows the loss-free counting techniques described above to be utilized in addition to conventional add-1 histogramming. The weight generation circuitry is shown in Fig. 3. A new integer weight, N, is generated with each ADC conversion.

The controller is a VMXbus D32, IMA Master which accesses 32-bit longword locations in the accompanying memory. Since the ADCs are typically set for use at their maximum 13-bit conversion gain, 32Kbytes (8K longwords) of memory are utilized per spectrum, and up to four spectra can be accumulated in the histogram memory.

4. Summary

With the inclusion of multiple detector gamma-ray histogram capability on the VMEbus, a VMEbus based microcomputer system is being configured which will replace a PDP-11/CAMAC based data acquisition system. It is expected that UNIX** System V operating system software has all the "real-time" implementations necessary for our applications. Specifically shared memory segments, memory locking, and semaphores will allow software similar to our current RXS-11-based data acquisiton software to be written.

** UNIX is a trademark of Bell Telephone Laboratories.

REFERENCES

1. J. Harms, Nucl. Instr. Meth. 53 192 (1967).

2. G. P. Westphal, Nucl. Instr. Meth. 146 605 (1977).

3. G. P. Westphal, Nucl. Instr. Meth. 163 189 (1979).

4. M. M. Minor, W. K. Hensley, M. M. Denton, S. R. Garcia, Radioanalytical Chem. 70 459 (1982).

5. G. P. Westphal, Radioanalytical Chem. 70 387 (1982).

6. The VME/VMXbus Dual Port Memory, DSSEDPRX, Sept., 1984. Data Sud Systems/U.S., Inc.

* * *

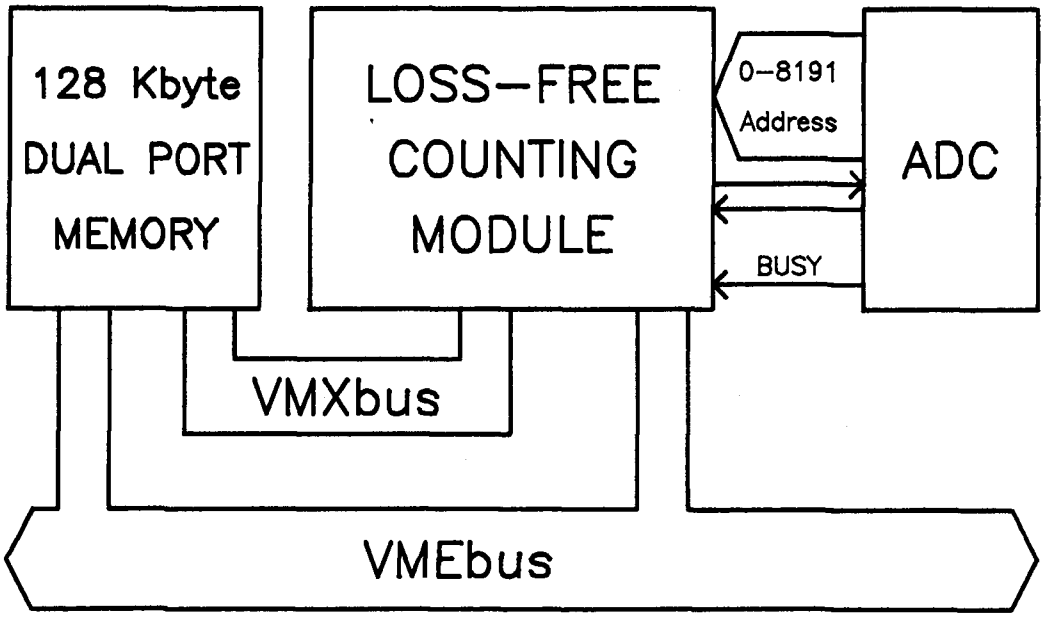


Fig. 1. Loss-Free Counting Module.

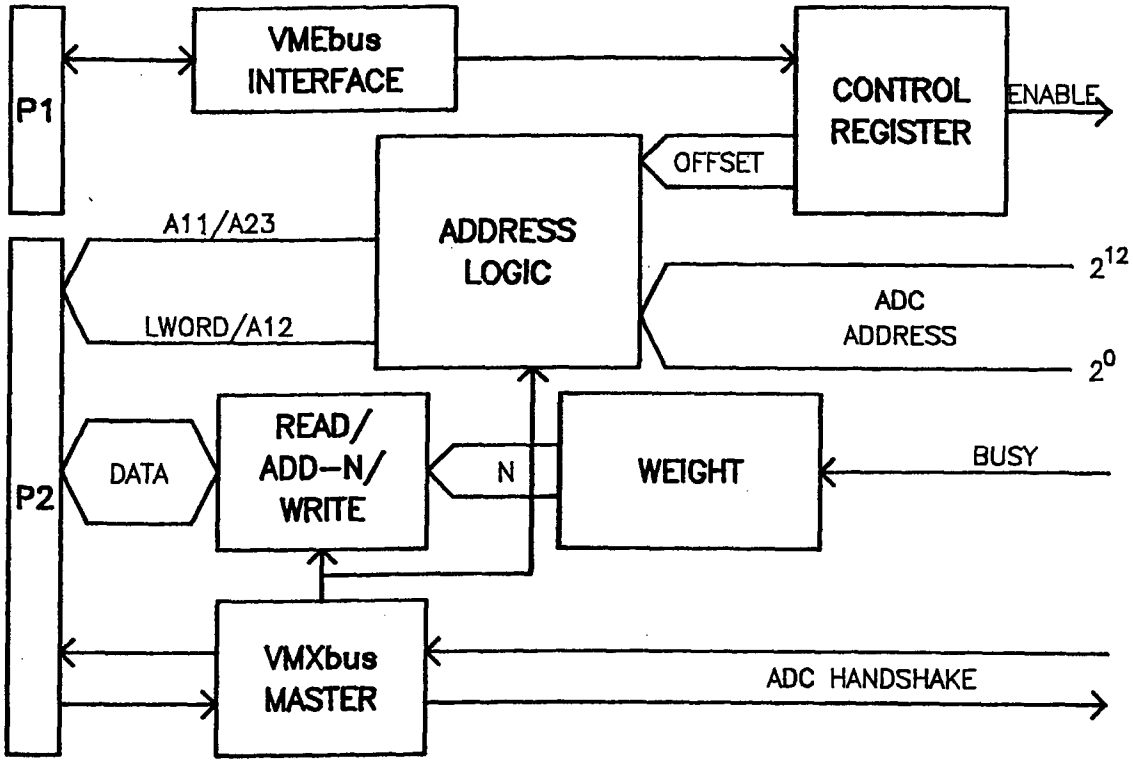


Fig. 2. Los-Free Counting Module block diagram.

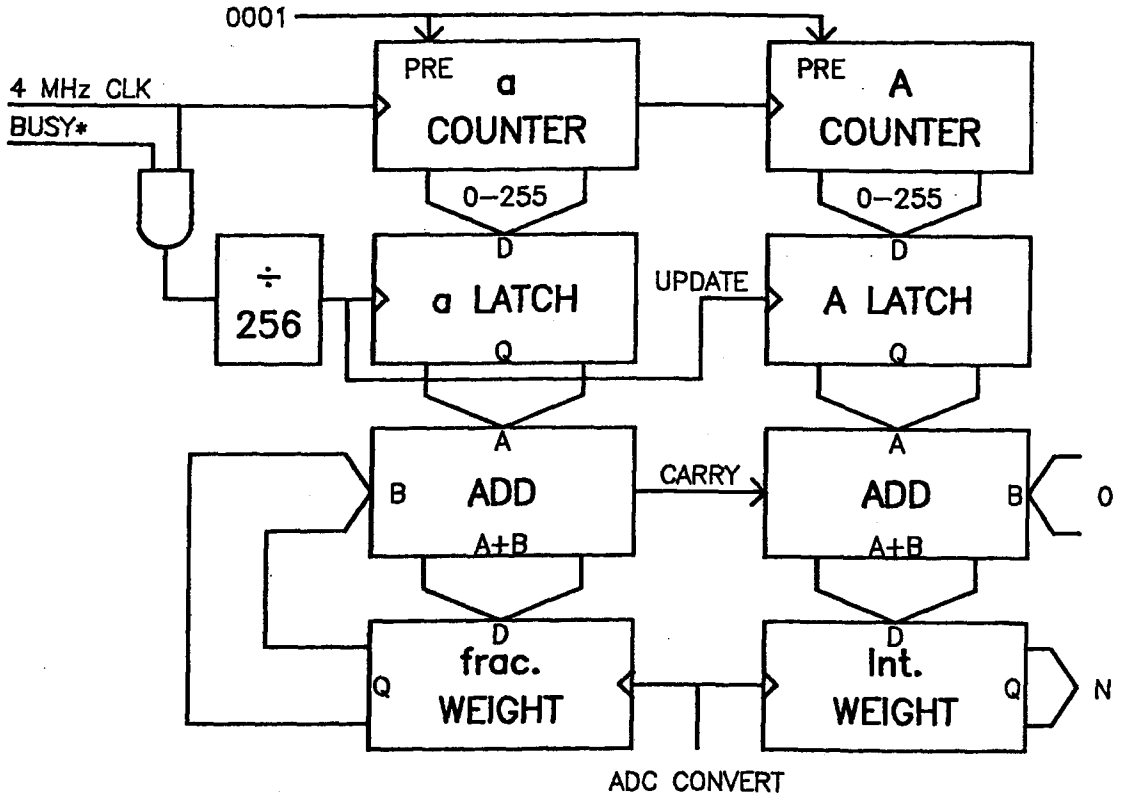


Fig. 3. Weight factor circuit block diagram.

A VMEbus APPROACH FOR THE CONTROL OF THE CLOSED ORBIT CORRECTION POWER SUPPLIES WITHIN THE SPS SUPERCYCLE

P. Martinod, G. Mugnai, J. Savioz, P. Semanaz

SPS Beam Monitoring Group

CERN, Geneva, Switzerland

Abstract

As an injector for LEP, the SPS will operate with a supercycle consisting of 5 elementary cycles (1p, $2e^+$, $2e^-$). During the e^+ and e^- cycles, the closed orbit has to be corrected everywhere during the cycle, thus the corresponding power supplies have to be pulsed with a different function for each elementary cycle.

This note describes a flexible solution using the VMEbus-68000 µp standard as a multi-output function generator to control COD & POD/MDP power supplies for closed orbit correction within the SPS supercycle. A VMEbus crate will house up to 36 independent, 16 bit resolution function generators.

Basically this approach consists of a VMEbus structure, linked to the SPS computers through the MIL-1553-B network, using a CPU in a mono-master configuration followed by nine dedicated VMEbus hardware units, 68000 (10 MHz) µp based, housing up to four function generators each.

1. INTRODUCTION

At present as a proton accelerator whether for fixed target physic or ppbar collider experiments, the Super Proton Synchrotron (SPS) works on a elementary cycle basis, as illustrated in Fig. 1.1.

As an injector for the Large Electron Positron machine (LEP) the SPS while continuing to accelerate protons as in a normal cycle, will have to receive as well electrons (e^-) and positrons (e^+) from the Proton Synchrotron (PS) at 3.5 GeV and accelerate them up to 20 GeV then finally injet them into the LEP.

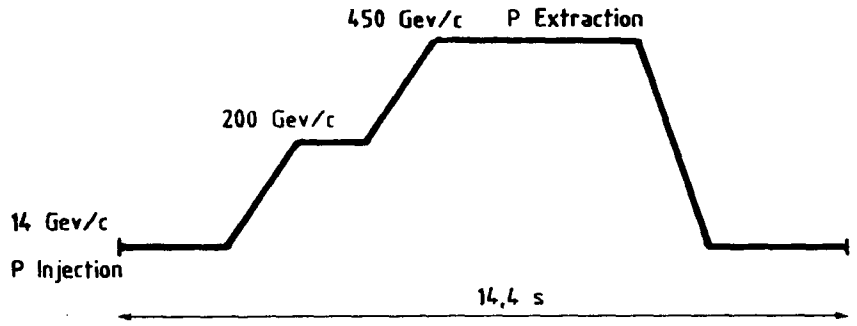


Figure 1.1 - The SPS normal proton cycle

Hence the new SPS cycle becomes a Supercycle (see Fig. 1.2) made up of five elementary cycles : 1p, $2e^+$, $2e^-$

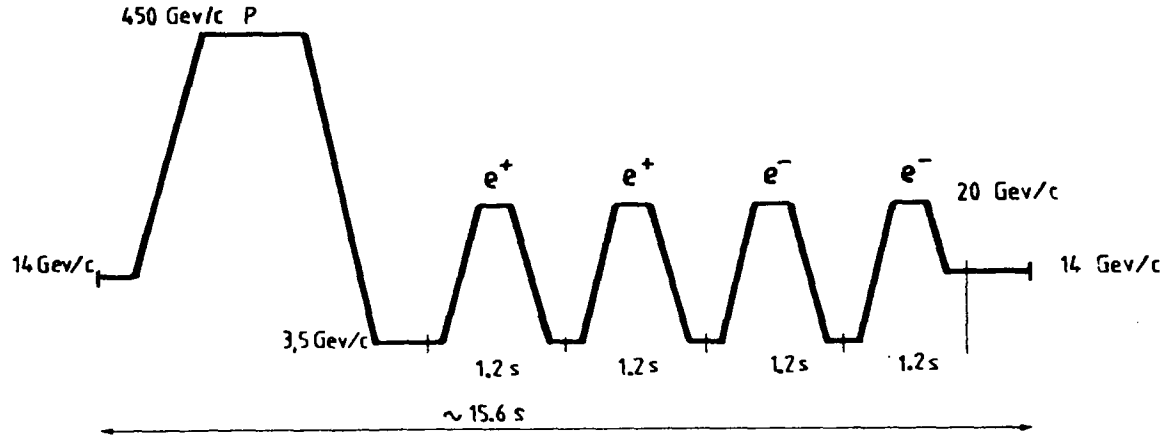


Figure 1.2 - The new foreseen SPS supercycle

The much greater complexity of the SPS supercycle, as compared to a normal proton cycle, requires additional and more sophisticated controls over several hundreds of magnetic elements and their associated power-supplies distributed around the accelerator ring.

During the e^+ and e^- elementary cycles, the beam closed orbit must be corrected everywhere during the whole cycle. This is due to the low injection energy of 3.5 GeV and the need to keep the orbit centered at the top energy of 20 GeV in order to catch all the synchrotron radiation with the tungsten diaphrams foreseen for this purpose. This is entirely different from the present mode of fixed target operation in which the orbit is only corrected at injection with DC current in the laminated closed orbit correctors.

Furthermore about sixty existing elements (i.e. RF control, Chromaticity adjustement, Extraction, etc...) currently controlled by AUXPS through first generation function generators (FG) must be pulsed with a different function for each of the five elementary cycles.

In this situation it is therefore necessary using present day technology and standards in particular large use of fast 16 bit microprocessors, VMEbus and analog converters, to develop for the control of approximately 250 closed orbit correction power supplies together with the sixty elements mentioned above, a new high resolution, powerful, flexible and easily implementable function generator called Multi-Cycle-Function Generator (MUGEF).

By taking into account the SPS and CERN guide lines such as modularity, the use of defined standards and that of commercially available units whenever possible, the new FG will cover all the SPS and we believe all CERN requirements; thus allowing uniformity in the hardware and software, easier control operation, and last but not least low-cost development, maintenance and piquet service.

2. FUNCTIONS REQUIREMENTS

The functions required as input reference for the power supplies are analog functions of time, which can be fitted and generated by a succession of straight lines, called vectors.

The resulting function accuracy depends essentially on the amplitude, slope and duration resolutions of the vectors; an ideal function generator having infinite vectors parameters resolution

A function is therefore defined from a user point of view as a set of dots each one having its amplitude Ai and duration Di; function discontinuities being allowed as well, see Fig. 2.1.

The required characteristics are as follows :

- Function definition : Ai, Di 1x16 bit word each
  Di are expressed in absolute time, i.e. :
  $Di \geq Di_{-1}$
- Amplitude A : $-32768 \leq A \leq 32767$ bit ($-10v \leq A < 10v$)
- Amplitude resolution : 16 bit DAC or 3.10^{-5}

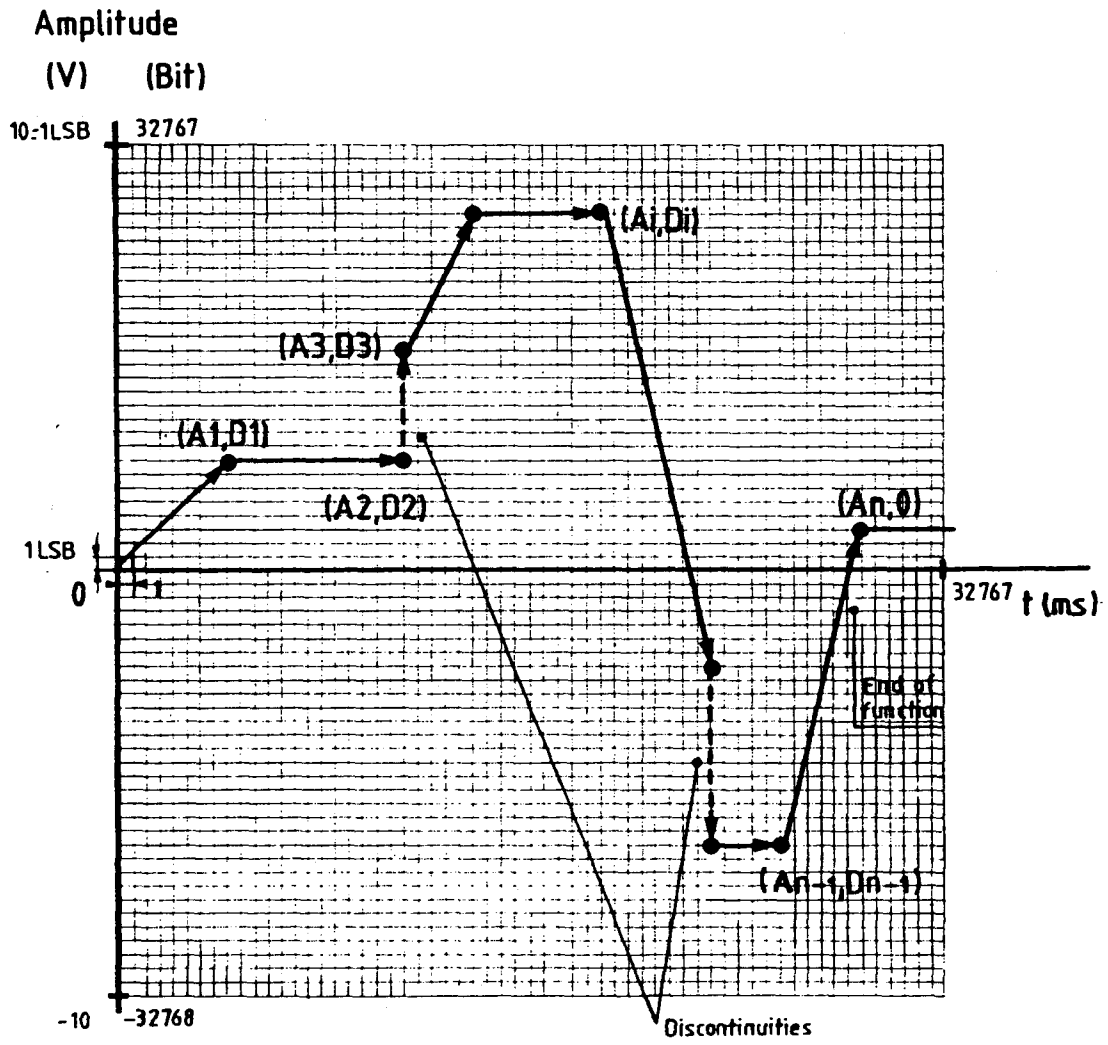


Figure 2.1 - Function requirements and definitions
(Amplitude and time not on scale)

- |Δ Amplitude| : $|\Delta A| = |Ai - Ai_{-1}|$
  $0 \leq |\Delta A| \leq 65535$ bit or $0 \leq |\Delta A| \leq$ (20v - 1LSB)
  It follows that a vector can be linked between <u>any</u> two dots of the matrix Amplitude - Time as shown in Figure 2.1.
- * Function length : 0 to 32767 ms
  the max. foreseen length of the SPS supercycle is 15.6 s
- Function length resolution : 1 ms
- Number of vectors : ~ 800
- Number of vectors/function : 1 to 800
- " " resident functions/FG : 1 to ~ 10

2.1 Vectors characteristics

The main characteristics of the vectors are essentially those of the function itself, this being made up of 1 to 'n' vectors. There is no upper limit in theory to the value of 'n'. In practice 'n' is limited by the memory size of the function generator.

The required characteristics are as follows :

- Amplitude Ai : $-32768 \leq Ai \leq 32767$ bits ($-10v \leq A < 10v$)
- Amplitude resolution : 16 bits (3.10^{-5})

* The function length is limited to 32767 ms (i.e. 2^{15} - 1 ms) due to the definition of the durations (Di) in absolute time using a 16 bit word, thus : $Di \geq Di_{-1}$

The max. function length allowed by the MUGEF function generator itself is much greater, the duration in hardware format being defined by 2 x 16 bit words, that is :

MUGEF Max. Function Length = $(2^{32}-2)T$ with durations Di in absolute time

Example : for a sampling-time T = 100 µs we have

| MUGEF Max. Function Length ≃119 hours. |

- Duration Di : $1 \leq Di \leq 32767$ ms
- " resolution : 1 ms

- Discontinuities : ±1 bit/0 ms to ±65535 bits/0 ms
- " resolution : ±1 bit

- Slope P : ±1 bit/32767 ms to ±65535 bits/ms
  and discontinuities allowed

- Slope resolution : this is one of the most important parameters in the generation of a function, which is better illustrated and explained in Fig. 2.2, rather than by analytical formulas.

It results that within the maximum amplitude and duration limits, the number of possible individual vectors is in the order of ~ 2.10^9, which show a very high slope resolution indeed.

Furthermore, for slopes |P|<1, the amplitude resolution is always 15 bits + SGN whatever the vector duration is. For slopes |P|>1, the amplitude resolution is optimised so as to have the maximum obtainable value.

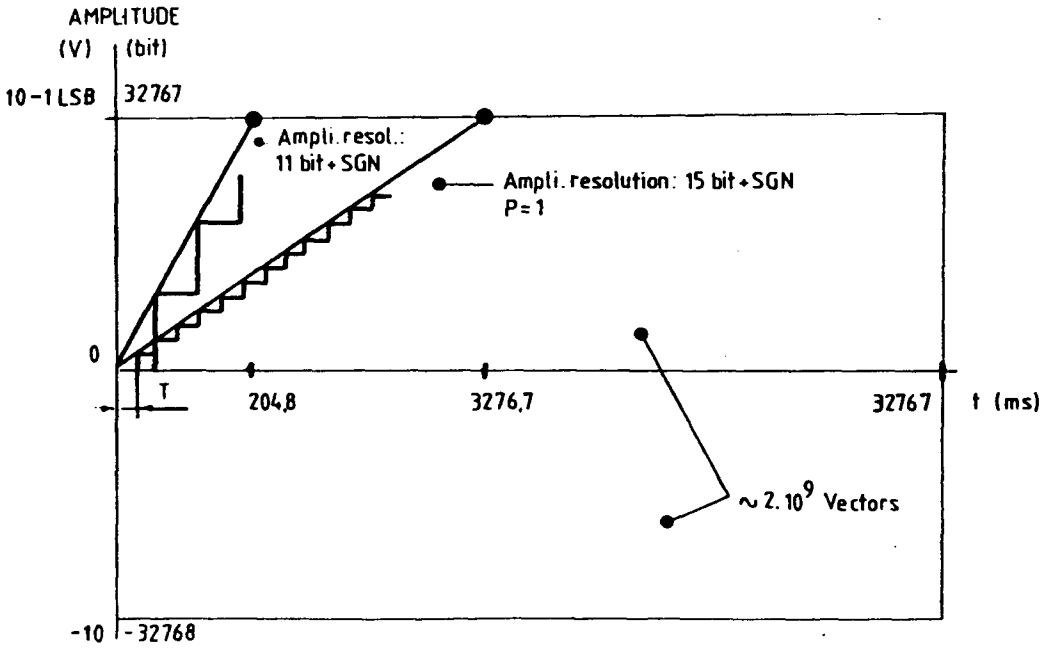


Figure 2.2 - Slope resolution of the vectors
(Amplitude and time not on scale)
(T = 100 µs for the values here shown)

2.2 Ramp generation

The precision and the stability versus temperature and time of the functions to be generated, require a digital rather than analog approach.

Indeed taking into account the function specifications and the high resolution and stability of present day quasi-monolithic DAC, a digital integrator is much more suitable in our application rather than a conventional analog integrator.

In a very simple form a Ramp Generator (RG) consists of a processor, namely a µP, where the vectors are computed, followed by a DAC converter. The DAC output is therefore a step function as illustrated in Fig. 2.3.

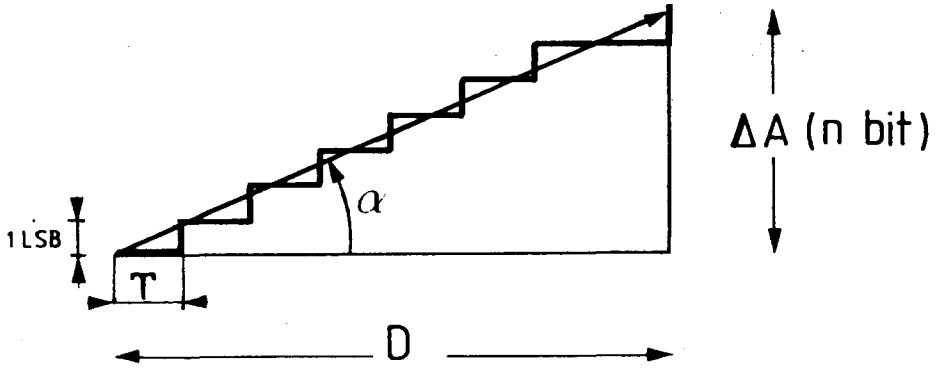


Figure 2.3 - Generation of a digital ramp of amplitude ΔA and duration D

The parameters specified by the user are the amplitude ΔA and the duration D.

The parameters requested by the digital RG are the sampling period T, the slope P = tgα = ΔA/D and the amplitude resolution ≥ 1 LSB.

The main problem is the determination of the sampling period T, which is used to increment or decrement the DAC output according to the sign of the slope to be generated.

T = D/ΔA no integer in most of the cases
thus T = f (ΔA/D, Amplitude resolutiom, Amax, Dmax)
T = 5 μs is the minimum value required for all the SPS function generators

The solution of this function, taking into account the minimum requested sampling time T = 5 μs, requires the use of a 68000-10MHz μP, with very fast dedicated algorithms entirely developped in assembler, and making use of 16 and 32 bits multiplications and divisions.

The digital RG acts therefore like an automatic and sophisticated frequency synthetiser within an overall accuracy of ± 1 LSB.

3. SYSTEM DESCRIPTION

3.1 System structure and layout

The function generators and the power supplies to be controlled are located in six Auxiliary Building (BA) placed around the 7 km SPS ring.

The schematic function generators layout for each BA is shown in Fig. 3.1.

Two independent VMEbus crates (1 for 36 CODs and 1 for 15 POD/MDP) in each BA, linked to the SPS computers through the MIL-1553-B networks, allow the entire control and analog acquisitions of the 250 COD and POD/MDP power supplies for closed orbit correction within the SPS supercycle.

We may distinguish two sections for each VMEbus crate :

- The VMEbus general crate control
- The ramp generators and power supplies control section.

The first section contains three VMEbus units, developped at CERN by the SPS/LEP Control Group and it represents the minimum base structure of the VMEbus crate, regardless of the process functions to be carried out in the other section.

We have in the order :

- VME/MIL-1553-B Interface
- TG3 Timing module. This unit allows the synchronisation of the functions to be generated to the SPS supercycle. It generates in particular the Start and Stop of the functions. The module is 6809 or 68000 µp based.
- CPU HAMAC 1. This board, 68000 µp based is the master of the VMEbus and it controls all the communications of the crate.

The second section deals with the process equipment dedicated units for the control of 36 independent Power Supplies. These are shown in the following order :

- 1 or 2 ON/OFF & Status board. This unit being too specific therefore is being developped at CERN.

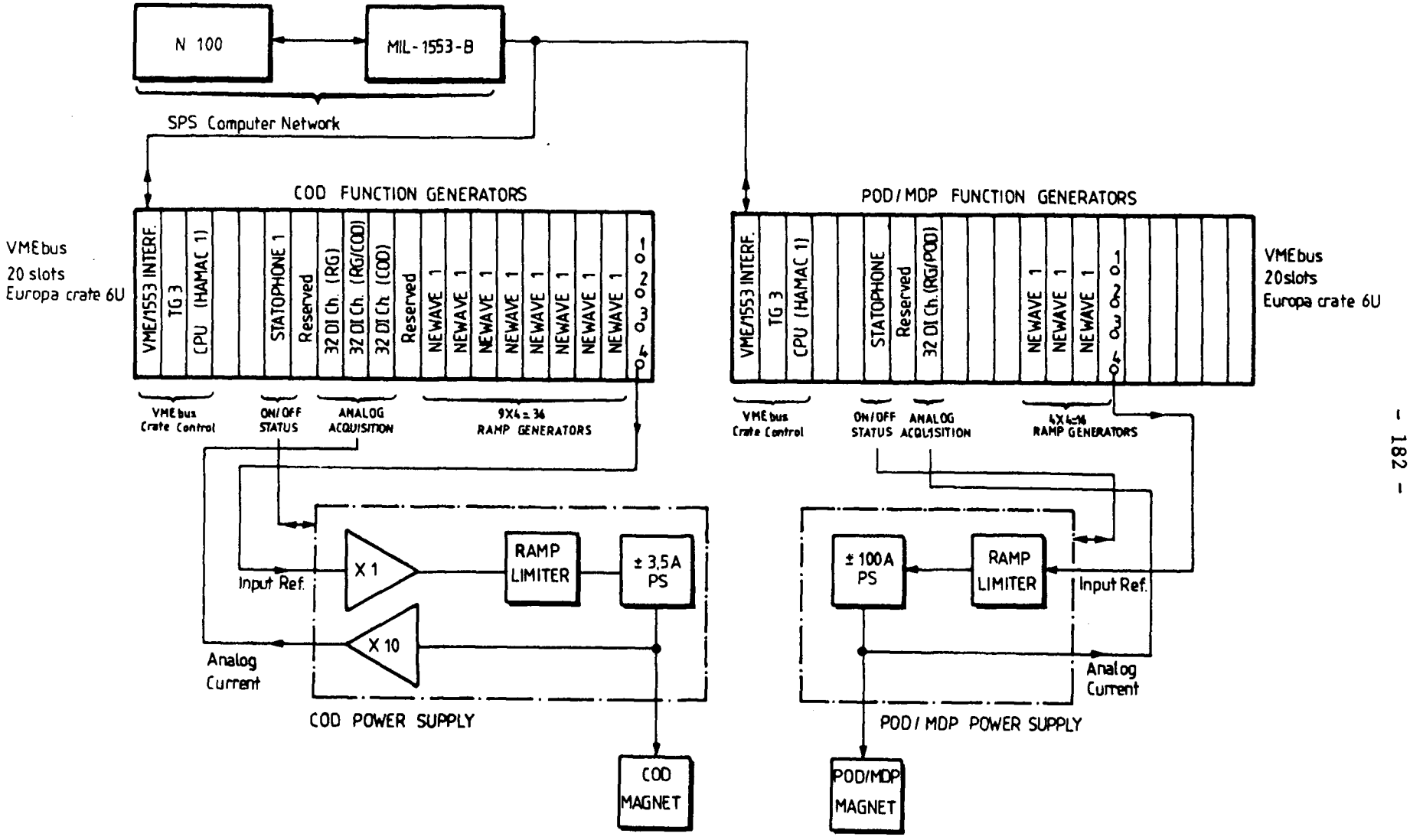


Fig. 3.1 SCHEMATIC CONTROL LAYOUT FOR COD & POD/MDP IN EACH BA

- 1 to 3 Analog Input board. 12 bit ADC resolution, 50 μs conversion time. 64 Single-Ended or 32 Differentials input channels. These units will allow the analog acquisition of the 36 Ramp Generators output (i.e. the Power supplies' reference voltage) and the 36 Power Supplies' current. This are no intelligent peripheral modules (i.e. no μp on board) and therefore their control is entirely made by the master CPU.

  It follows that the analog acquisitions of all channels cannot take place simultaneously but in sequence, one after the other in time.

  These modules are getting commercially available, e.g. XVME-560 XYCOM

- 1 to 9 Ramp Generators boards. These units are the heart of the FG and they carry out the real time tasks of the system.

  Essentially a ramp generator module is an intelligent hardware dedicated unit, 10 MHz, 68000 μp based, driving up to 4 independent 16 bit DACs supplied by individual DC/DC converters for noise and common mode rejection. The functions are carried out by very fast dedicated algorithms. Thus a VMEbus crate will house up to 36 independent FGs.

  From a function loaded in its memory in 'hardware format', the μp computes in a time-sharing fashion the 4 independent ramps and sends every 'T' μs the new current amplitude to the corresponding DACs.

  It should be noted that the sampling time T = 100 μs is largely suitable for all the SPS power supplies [1)]. Therefore, a ramp generator with 4 DACs can pratically cover all the SPS function generator requirements. If particular applications requiring steeper and higher amplitude resolution ramps should arise, we can of course go as low as T = 5 μs, just by inserting 1 DAC instead of 4 DACs into the modular Ramp Generator, with no change on the application software.

  In this way all the SPS FG requirements can be largely covered by the MUGEF solution.

The ramp generator module is made up of a "Mother + Daughter board" technique, thus allowing modularity, flexibility and cost-saving. Indeed, by changing the daughter card, one can define another function (e.g. ADC, Scalers, etc...), the mother card remaining the same.

The mother board called FLEXIPM is a commercial unit from DATA-SUD-FRANCE, as shown in Fig. 3.2. It should be noticed that the "Analog Outputs" and the "Interrupts Inputs" pass through the P2 connector.

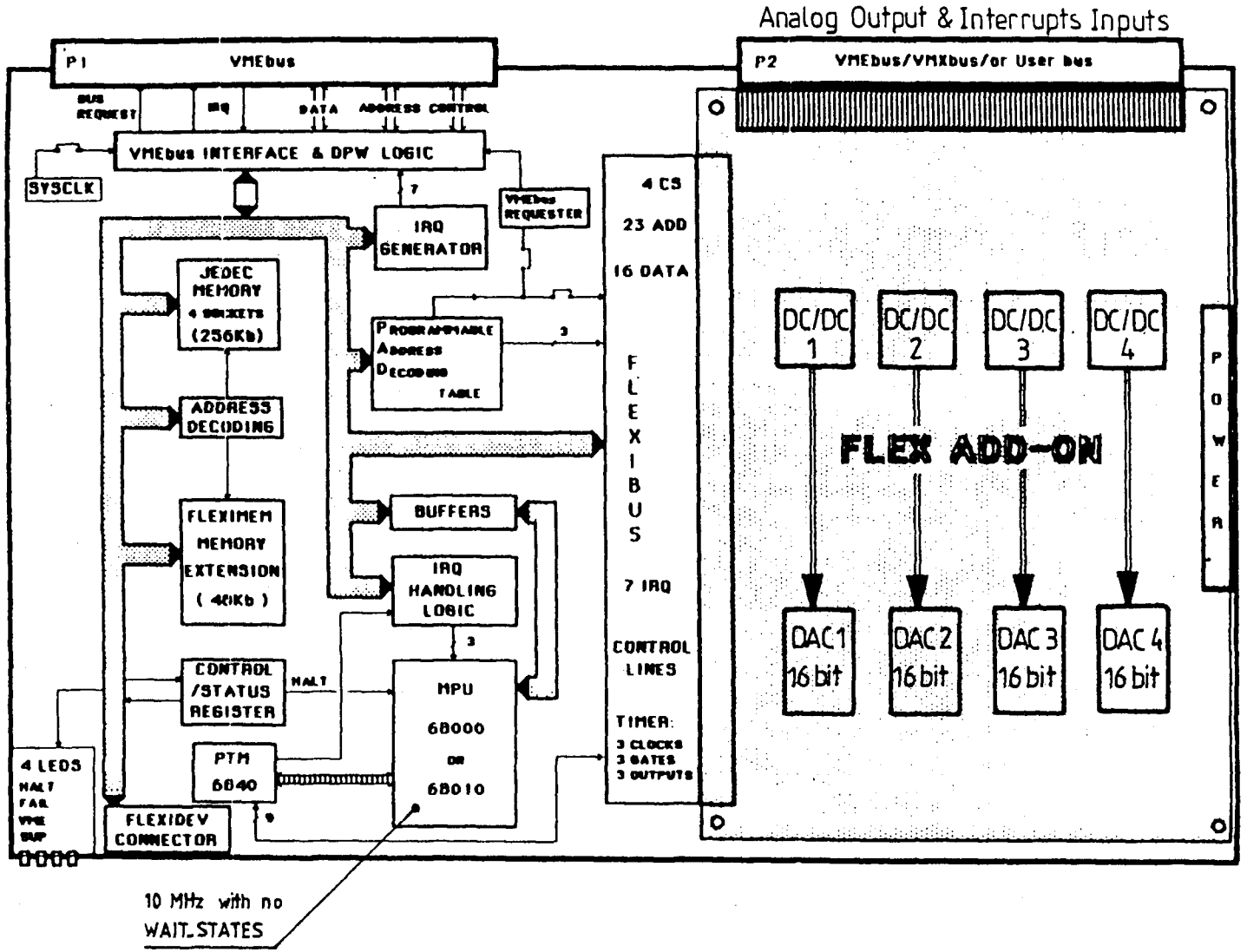
Remarks :

a) An 8 channels - 16 bit DAC - daughter card is currently being developped. The unit should allow a considerable cost and volume saving reduction per FG channel.

b) In view of the decreasing cost of memory, the memory size of the RG could be increased by using new expected, JEDEC compatible, 256/512 KB RAM & EPROM in order to have more vectors/function and a greater number of resident functions.

c) No booster output amplifier is required. The DAC being used is capable of driving at least 70 m of twisted pair cable with a good margin of gain and phase stability. This is a very important point to highlight, as it allows a considerable reduction in the volume of the electronics involved, hence an important cost-saving too.

d) Function length and resolution could be both increased and also software programmable.



FLEXIPM BLOCK DIAGRAM

Fig. 3.2 RAMP GENERATOR CARD DIAGRAM

3.2 Function generation and system configuration

The system is a VMEbus mono-master structure configured in a multiprocessor environment (up to 11 68000 µp/VMEbus crate). The CPU is the master of the crate.

All data transfer and commands N100-VMEbus take place in the CPU, through the VMEbus/MIL-1553-B interface, using an appropriate communication package.

The functions in user format are sent from the N100 to the CPU, where they are converted in a new format, called hardware format, fitting the Ramp Generator requirements. Up to 36x10 = 360 different functions of 20 vectors each, both in user and hardware format can be resident in the CPU.

Functions in hardware format are sent in real time to the RG in the time interval Stop-Start, according to Fig. 3.3.

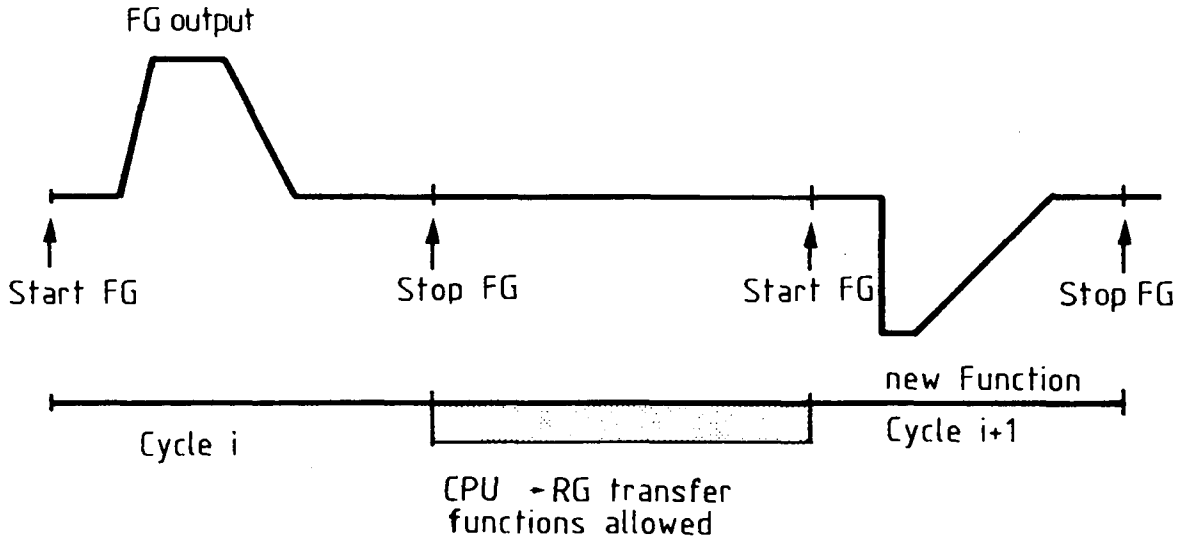


Figure 3.3 - Function generation and timing signals
(Start FG - Stop FG intervals not on scale)

Transfer time from CPU to RG memory for 800 vectors is less than 10 ms.

The functions are then generated in real time by the ramp generators which are entirely self independent and they cannot be interrupted

until the next Stop timing signal from the TG3 module or from other external sources.

The analog acquisitions and ON/OFF & Status board are entirely controlled by the master CPU.

4. SOFTWARE IMPLICATIONS

- CPU. A data module subroutine 'DMS MUGEF' resident in the CPU will provide the software data base both for the communication control between the SPS computer network and the CPU, and also the management of the VMEbus crate.

  A function editor developped around the Motorola 68000 µp monitor has been finalized and tested in the CPU.

  Having the CPU connected to a standard video terminal via its Host/Terminal RS-232 port, the function editor allows a inexpensive way to work on the ramp generators without the need of an external hardware/software development system.

  Specific application programs concerning survey and faults diagnostics have yet to be written.

  Programming languages : MODULA-2, Assembler.

- Ramp Generator. Very fast algorithms have been written in assembler, both for the 1 DAC/RG version (T = 5 µs fastest sampling time) and for the 4 DAC/RG version (T = 100 µs). Real time tasks.

5. PRELIMINARY RESULTS

A first RG prototype with 4 channels - 16 bits DAC - has been developped and tested. The results are in agreement with the expected values, in particular concerning the functions generation, accuracy, noise crosstalk and hardware modularity.

The following photos show the high degree of flexibility we may obtain with the 'MUGEF' system.

Photo 5.1 shows the summing output on the oscilloscope, of the functions carried out by the 4 channels, photos 5.2 to 5.5. The 4 DACs have completely different function each other, both in amplitude and duration.

Also to be noticed are the function discontinuities.

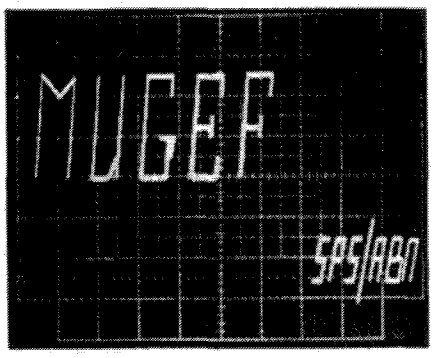


Photo 5.1

Σ DAC 1 to DAC 4 outputs
on the oscilloscope
V = 2 v/div.
H = 500 mv/div.

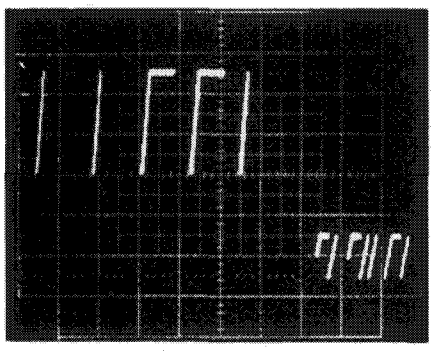


Photo 5.2

DAC 1 output

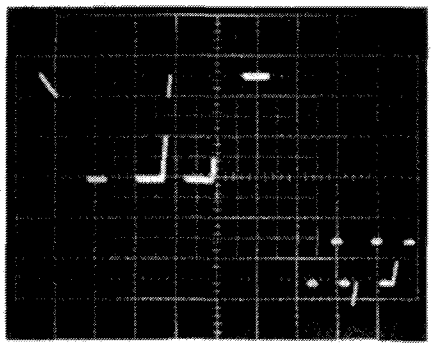


Photo 5.3

DAC 2 output

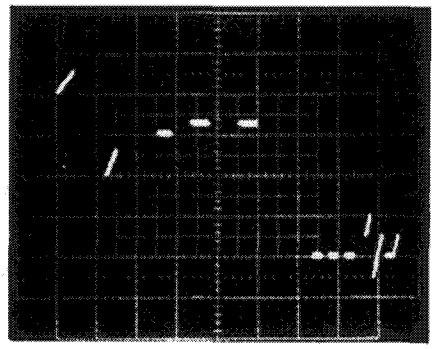


Photo 5.4

DAC 3 output

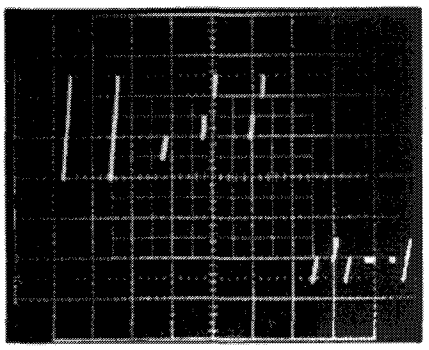


Photo 5.5

DAC 4 output

7. CONCLUSIONS

The VMEbus standard appears to be very suitable for the development and the construction of the new SPS multicycle function generators, both for technical and economical reasons.

Thanks to the speed and resolution of the modular ramp generators, the functions applied as references to the power supplies will be identical to the desired current ouputs. The operation and fault diagnostics of the function generators and power supplies will be simpler and more transparent for the MCR and for the piquets.

By inserting 1 to 4 DAC in the ramp generator card, according to the steepness and to the amplitude resolution of the ramps required, the proposed 'MUGEF' solution is modular, flexible, cost saving and very suitable for all the function generators at the SPS and, we believe, at CERN as well.

8. REFERENCES

1) A possible solution to control POD & MDP Power Supplies for Closed Orbit correction within the SPS supercycle.
SPS/ABM/Note/85-46 - 7.3.1985
G. Mugnai, J. Savioz

P2 CONNECTOR - PANEL DISCUSSION

Chairman: D. O. Williams (CERN)

Participants: M. Lösel (Force Computers, Munich), R. Rausch (CERN), J. Regula (Ironics Inc., U.S.A.), J. Werdehausen (Motorola GmbH, Munich) E. Owen (Daresbury Laboratory, U.K.),

Abstract

The outer two rows of pins on the VMEbus P2 connector have been a source of controversy since they were allocated as "User I/O Pins".

Several implementations of auxiliary buses on these pins have been made, notably: VMXbus (which has passed through two revisions), MVMX32bus (from Motorola), VMCbus (in use at CERN), not to mention many others from different manufacturers.

Can we converge onto one standard, if so what should it be and what effect does this choice (or lack of choice) have on associated equipment, especially in experimental physics?

Chairman's Summary: The Use of the User I/O Pins on the P2 Connector

INTRODUCTION

It is clear that a full technical discussion on the merits of various different approaches to the use of the "User I/O" pins would have required at least one day. It is also clear that we did not have a whole day available for such a discussion at this two-day conference. I hope, nevertheless, that the audience and the panellists felt that it was useful to try to review the major ideas that arise in this area.

As Chairman I briefly introduced the speakers.

Jürgen Werdehausen is Technical Marketing Manager with Motorola Microsystems, Munich. Max-Eberhard Lösel is Vice-President, Technology and Engineering with Force Computers in Munich. Raymond Rausch is with the SPS Division at CERN in charge of the hardware aspects of the LEP/SPS Control System. He also has been involved in ESONE standardisation activities related to microprocessors. Ted Owen is at Daresbury Laboratory, U.K.; he is Chairman of the ESONE Technical Committee, and has recently taken on the job of Chairman of the VITA User's Committee in Europe. Jack Regula is with Ironics Inc. of Ithaca, New York, U.S.A., where he has designed an MC68020 processor board using VMXbus Revision B, and is on the VITA Technical Committee.

Then I gave a brief overview of the problem.

I showed a telex received about two weeks prior to the meeting where a company primarily building physics instrumentation had been enquiring what CERN was recommending as a VMXbus specification.

Basically the VMEbus specification defines the two outer rows (a and c) of the second (P2) connector as "User defined". However in October 1983 a draft specification of VMXbus was issued, which required the use of these pins. Several users (notably the UA1 experiment at CERN) and several companies (notably, but not exclusively Data Sud) then made designs to this draft specification. When the specification was changed from draft (Revision A) to final (Revision B) in December 1984, there were major modifications, and all Revision A designs were made obsolete.

During autumn 1984 it became clear that Motorola were not very enthusiatsic about VMXbus Revision B, and that they proposed to design all of their MC68020 VMEbus board products using a new specification, called MVMX32.

Question: Why do we need a second bus anyway? Surely VMEbus can do everything that we need?

Answer: This is an issue of system design, and more especially one of performance. VMEbus is fine as a system bus, but we commonly find that each processor in any mutliprocessor system needs a private bus to access its private equipment (private memory, special I/O, etc.). If this private equipment is connected via the system bus then the bus can get saturated and become the factor that limits the overall performance.

Question: Can this private bus be limited to memory extension only?

Answer: In my view any such limitation is too much of a constraint. Systems designers in fields like particle physics need to interface very high data rate equipment to microprocessors. This sort of high rate equipment is exactly the sort of thing that will saturate the system bus, and so should be directly attached through a private bus. However we should not limit this private bus uniquely to passive memory access.

The minimum physics requirement is that it should be possible to install a DMA controller on the private bus which permits data coming from phyics readout systems to pass into a dual port memory on the private bus, and where that data is then accessible to the controlling processor.

A set of physics requirements that are slightly more extensive, but which I consider still reasonable, include:

1. Extension from a single secondary master (DMA controller) to multiple secondary masters. This essentially permits greater flexibility to control multiple devices from a single CPU.

2. Possibilities to send interrupts from the secondary masters to the CPU. This permits faster and more predictable response to events such as DMA completion, as compared to sending the interrupts over the system bus.

3. The possibility for the private bus to prevent devices attached to dual port memory via other ports from accessing this memory. If this facility is missing it is not possible (or, more exactly, not simple) to ensure that the view of the data as seen through two ports of the memory is consistent.

4. The possibility to prevent data in attached memory from entering the cache of a processor that has a data cache.

5. 32 bit data (no argument) and 32 bit addresses. I see the need here as being a strong preference to avoid base registers, and to keep the addressing on the system bus and private bus compatible, as far as possible.

Problems: One way of looking at this issue is to say that we are asking for the functionality of the VMEbus (32 bit address and data, multiple masters, interrupts, etc.), while only having limited real estate and 64 pins available to provide this functionality

It is also crucially important to emphasise that the users need to mix and match boards from different manufacturers. The ability to do this has been one of the real success stories of VMEbus, and I sometimes think that the suppliers do not realise how important this aspects is for the users.

Accordingly it is vital for the future of VMEbus that VITA puts some order into this situation and provides a single recommendation for a private bus.

VMXbus Revision B

As far as I can see this rather poor for use in physics, since:

a. The absence of a LOCK line (which was present in VMXbus revision A) means that safe access to a dual-port memory cannot be provided.

b. The absence of a CACHE signal will make it difficult to keep volatile data out of the cache of processors like the MC68020.

MVMX32

This design seems to solve the two major problems that I see with VMXbus Revision B. However I then pose two other questions:

a. The original design comes (if I understand things correctly) from a pure memory access bus, the RAMbus. Is the MVMX32 specification mature enough for the multiple master situation?

b. Is it too tied to the MC68020?

PANELLISTS' PRESENTATIONS

Juergen Werdehausen

Emphasised that the users must remain free to use the "User I/O" pins for anything they need, and pointed out that Motorola saw MVMX32 as an optimal bus for the MC68020, and not necessarily in conflict with VMXbus Revision B. Since MVMX32 has evolved out of RAMbus, it has a proven architecture, and Motorola will provide VLSI support chips for MVMX32 interfacing.

Max-Eberhard Lösel

Also emphasised the freedom for users to use these pins in any way they like. Force use a special connection ("FLME" – Force Local Memory Extension) between processsors and local memory,

which allows the processor to be attached to one memory board on each side. These boards are interconnected using on-board DIN connectors that are aligned parallel to the backplane. This set of interconnected cards can then be inserted or removed from the VMEbus backplane as a single unit.

The overall architecture which Force see as appropriate for automated manufacturing is shown in the attachments.

Raymond Rausch

First of all explained the reasons which led his team to concentrate some of their work on the VMCbus, which is a definition of the M68000 pinout onto the 64 "User I/O" pins.

He considered that the main problems with VMXbus Revision A had been:

a. Lack of any interrupt specification

b. Limitation to a single secondary master

c. Unusual address/data multiplexing. 32 bit data is supported, but the address is multiplexed with twice 12 bits.

He saw the main advantages of VMCbus (C for CERN) as having

a. Full M68000 bus extension

b. Allows the logical and physical extension of the rather limited VMEbus board space

c. Allows use of M68000 peripheral chips

d. Allows easy no wait state memory extension

e. Provides vectored interrupts

f. Supports multiple secondary (DMA) masters

g. Facilitates the monitoring of the activity of the M68000, since special monitoring hardware can be located on separate cards.

Note that VMCbus supports 32 bit addresses and only 16 bit data. It was emphasised that VMCbus is NOT an alternative to VMXbus.

Raymond's list of desirable features, from the user's point of view, for the extension bus included:

a. Full 32 bit address and data, which automatically implies multiplexing, in view of the limitation to 64 pins.

b. Mulitple secondary masters and

c. Vector interrupt capability, which together imply a daisy chained arbitration.

d. Full memory extension with no wait states

e. LOCK and CACHE facilities to support dual-port memory and caching processors

f. Full support for the MC68020, including the coprocessor architecture

g. Simple interfacing for other processors, including the NS32032 and Intel 80286/80386

h. Asynchronous

He summarised by giving his opinion that we have not seen the end of the iterations around the VMXbus specifications, and advised users to wait and see, while encouraging the suppliers and VITA to come to the correct conclusion as fast as possible.

Ted Owen

Complained that the Chairman had stolen his lines, and then emphasised the need, as seen by the users, for a single agreed VITA specification for the private bus. A standard would bring the well known benefits which come from competition amongst suppliers of compatible equipment, and would assits system design by removing the choice of an extension bus from the decisions that have to be made.

Jack Regula

Reported that VMXbus Revision B, the official party line, was relatively easy to use, and that he didi not think that there were real problems with it. VMXbus is good for 16 bit applications and adequate for 32 bit processors. With the coming of big (1 Mbit) memory chips, the emphasis for the extension bus would be on its use for I/O, rather than memory extension. With an on-board cache memory, the need for a zero wait-state memory bus would largely disappear.

General Discussion

At this point the session was thrown open for user questions and comments. I think that the most important was the brief report made by Mira Pauker (Philips, France), who is the French delegate to the IEC committee. If I have the picture clear (I have no notes on the point) the IEC decided that neither VMXbus nor MVMX32 should be endorsed as a bus related to VMEbus, and set up a small working team to try to get the functionality of VMXbus Revision B and MVMX32 into a unifed VME System Bus, or "VSB". If you are optimistic, this work might be in reasonable shape by the end of 1985!

In closing the session I pointed out that the conference, with its attendance of over 300, with 60 people from industry, had shown the vitality of the VMEbus marketplace. The confusion over VMXbus presents a serious challenge for both the users and the suppliers, and I sincerely hope that VITA will succeed to bring some measure of order in the near future.

VMEbus P2 Connector Assignments
==========

Advanced computer and microcomputer architectures require a powerful bus structure for the various types of processing and I/O handling capabilities.

The VMEbus in that respect is seen as a future oriented bus structure for 16 and 32 bit processing unit and represents the "primary bus" in the system enviroment.

To allow the system designer, system integrator and user to utilize all the power of a processor it is required to add "subsystem busses" to the primary bus. Therefore there is a strong need for a VMXbus and even the VMSbus within the VME system enviroment.

The P2 connector in the VMEbus enviroment had been layed out to conform with these requirement and to carry out two directions from a system design viewpoint:

a) to provide an user I/O on P2 for dedicated customer I/O interface solutions.

   It has been used so far to assign parallel I/O interface lines (i.e. line printer or equivalent) as well as serial I/O interfaces lines or proprietary bus structures servicing 8 bit board products to this port.
   P2 connector may optionally standardized for RS 232 serial interface I/O and/or printer interface.
   Our recommendation is to leave it as user I/O esp. in a 16 bit enviroment.

b) to provide an interface for a secondary or subsystem bus. The VMXbus was and still is our recommendation for the P2 connector under this viewpoint. VMXbus, Rev.B, has been released now and has so far been the common activity of the manufacturing group. There is no major need to change the P2 connector assignment. The VMX on P2 serves the designers need having the secondary bus structure available in a more complex computer envoriment.

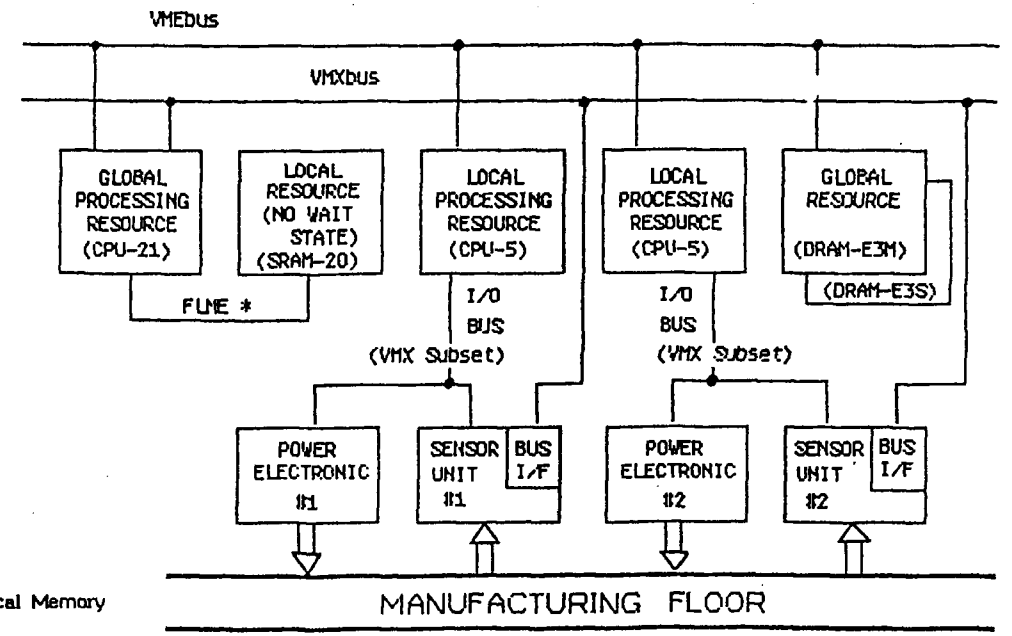
The secondary bus should not serve a special need of a particular microprocessor chip structure and should be structured as much as possible to support system enviroment configurations. VMX can do the job.

FORCE's FLME structure is not destroying the above described model in contrary FLME is even supporting it. There is no limitation built into a board design by using FLME. FORCE is going to announce products soon which will demonstrate this functionality.

FORCE COMPUTERS GMBH

Max-Eberhard Lösel
Exec. VP R + D

FORCE 's 32 Bit/16 Bit System Structure with Distributed Front Ends for Automated Manufacturing



* FORCE Local Memory Expansion

18.9.1985

VMCbus - A USER I/O LOCAL BUS

R. Rausch
CERN - LEP/SPS Controls Group

The conclusions of the CERN interdivisional working group on bus systems and microprocessors made in 1982 recommended the standardization of future equipment on the VMEbus and the 68000 microprocessor family.

At that time the VMXbus Specification was awaited and the first publication, revision A, was received by October 1983. After a study of the document several serious technical limitations were identified amongst which:

a) No interrupt mechanism for I/O modules.
b) Only one secondary master possible.
c) Address/Data multiplexing for the 68000 family which is not multiplexed.

For the LEP/SPS control system a function to function architecture has been adopted for the VME multiprocessors requiring close coupling of the CPU with private memory extension boards and with private Input/Output cards. Thus for this type of application the LEP/SPS Controls Group decided to assign the full 68000 microprocessor family to the USER-I/O pins of P2. A detailed specification manual, on the so-called VMCbus has been written and made available to VME users and manufacturers. The main advantages of this approach are the following:

- Full 68000 microprocessor bus extending physically and logically the VME board size and functionality with up to 5 VMC cards.

- Possibility to use directly all 68000 peripheral chips; the addressing, the timing and control signals over the VMCbus being those of the 68000 processor.

- Fast memory extension without WAIT states up to 12.5 MHz and no multiplexing/demultiplexing logic.

- The 68000 vectorized and autovectorized interrupt mechanism is provided over the VMCbus allowing each VMCcard to interrupt directly the microprocessor.

- The 68000 Direct Memory Access mechanism is supported over the VMCbus allowing multiple DMA masters to communicate with the microprocessor's private memory.

- Each VMC board may be interfaced both to the VMEbus and to the VMCbus.

- A powerful diagnostic tool is provided by plugging a logic analyser directly onto the VMCbus to monitor the microprocessor's activity on its private internal bus without perturbation.

It should be noted that Private 32-bit data transfers could be done over the VMCbus, using row b of P2, but this would exclude 32-bit VMEbus transfers. Nevertheless, in many applications, it is more important to use efficiently a 68020 with a large private memory than to transfer 32-bit data amongst processors.

The VMCbus is not meant to be an alternative to the VMXbus as its use is restricted to the 68000 microprocessor family. It allows an easy, fast, powerful and modular extension of the VME board.

ABOUT VSBBUS (VME SUBSYSTEM BUS)

IEC Special Working Group nominated by the National Committee delegations during last IEC General Meeting (May 85)

Is preparing a specification which is *technically a superset* of both VMXbus and MVMX32 bus

First meeting took place on a 6−9 sept, at Los Gatos

Present experts:

Nominated by	USA Nat. Com.	Arlan Harris	(Hamilton/Mostek)
		Doug Graft	(MOTOROLA)
Nominated by	U.K. Nat. Com.	Paul Borrill	(London University)
Nominated by	France Nat. Com.	Mira Pauker	(PHILIPS CTI)
Convenor		Robert Davis	(IEEE MSC Chairman)
Present observers:		Shlomo Pri−Tal	(MOTOROLA)
		Craig Mac Kenna	(SIGNETICS)

It is anticipated that the preliminary specification will be available within a short period of time.

VMEbus CRATE INTERCONNECT

E. Pietarinen
Department of Physics, University of Helsinki,
Siltavuorenpenger 20 D, SF-00170 Helsinki

ABSTRACT

The UA1 VMEbus based read-out system uses crate interconnect modules for communication and data transfer between VMEbus crates. A memory mapped transparent access through a 64 kbyte movable window allows an easy control access to the remote crates, while a separate block transfer facility moves data between VMEbus crates at rates up to 10 Mbytes/sec.

1. The main features on the VME crate interconnect and its use in the UA1 VME readout system

The VMEbus specifications do not prescribe any definite rules to be followed when building VME multicrate systems. Standard VME I/O modules and datalinks are usually restricted to transfer speeds below 1 Mbyte/sec, which is not adequate for high energy physics data acquisition systems.
In the UA1 experiment the off line processing capacity and the speed of data storage (tape writing, laser discs) limit the final data rate below 0.5 Mbyte/sec, but much higher peak rates have to be handled internally during the event readout.

The task of the VME crate interconnect module is to provide the required control and data transfer facility to create a VME multiprocessor network suited for large data acquisition systems.

The functioning of the system can be best understood by studying the block diagram of the UA1 data acquisition structure in Fig. 1 (ref.1). The readout supervisor has complete control over all the readout crates. Most of the readout modules at present are Remus branch drivers (RVMEX, ref. 2). When a readout trigger occurs, the digitized data are read in parallel via the VMX ports into VME/VMX dual port memories. The event appears sliced into several (~40) fragments in different memory modules and even in separate VME crates. The event manager sets up the crate interconnect modules to collect the data together and transfer them

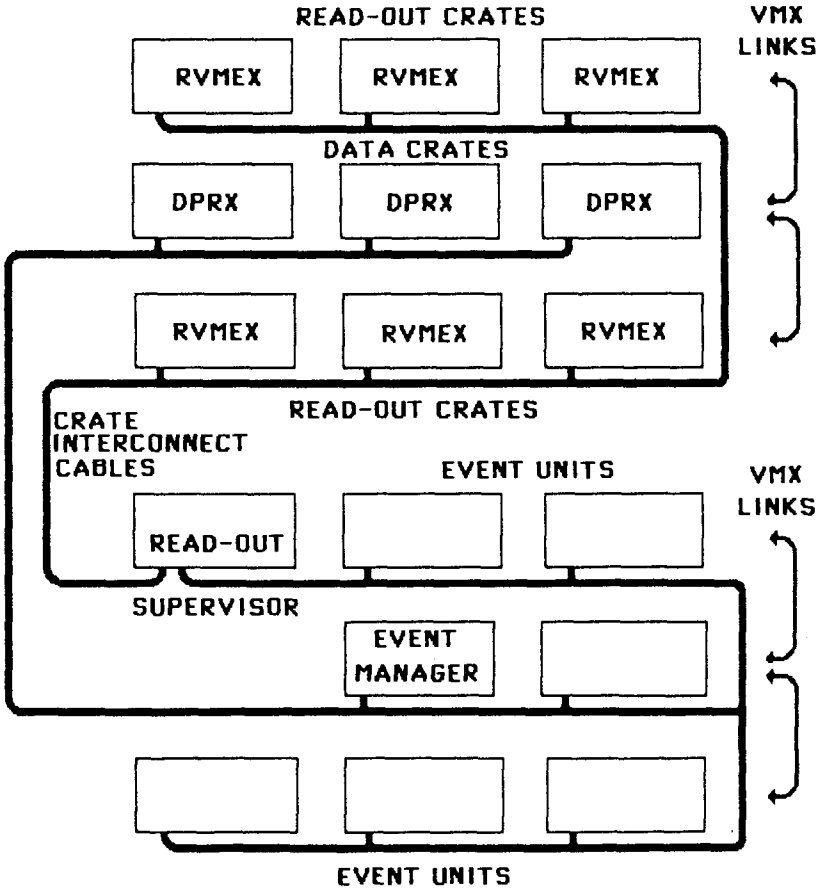


Fig. 1. The UA1 VME readout system

simultaneously into those event units which have requested an event. These event units may even be located in different crates. Due to such a broadcasting mode the same event can be made available to several tasks without significant overhead to the event manager or decrease in the data flow rate.

The most frequent tasks of the crate interconnect module are clearly the control access and the block transfer of data. These operations are controlled hierarchically. The event manager is a master over the data and event unit crates and it determines the way the block transfers are performed. Correspondingly the readout supervisor is a master over the readout crates. Actually the readout supervisor is accessible by the event manager and using two crate interconnect branches in chain, the event manager can in principle directly access the readout crates.

For control access the master VME crate interconnect can set one of the remote crate interconnects into a window mode, which maps a 64 kbyte segment of the remote crate VME address space to a fixed location in the master VME crate. VME cycles accessing the window area in the master crate are translated into similar VME cycles in the remote crate (see Fig. 2). The base address of the window in the remote crate is contained in a register and it can be altered with a single instruction.

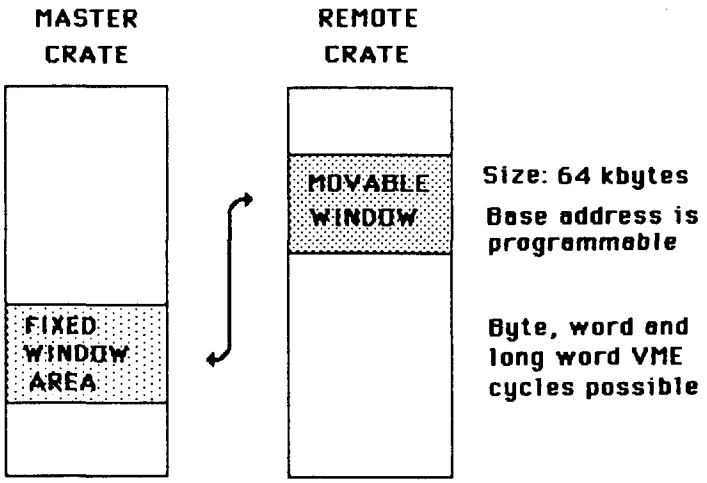


Fig. 2. Crate interconnect window mode

Since the window mode is normally used to refer to control registers, descriptor blocks or CPU mail boxes, the window size is adequate allowing non multiplexed fully handshaked access through the crate interconnect cable.

For block transfer a CPU initialises via the master crate interconnect the source and destination interconnects into direct memory access (DMA) mode writing values to the start address and byte count registers. After a start command the modules transfer data autonomously until stopped by the byte counter or an error condition. The correctness of data is controlled by a 32-bit parity check as well as the number of bytes transferred. A software initiated retransmission is done in case of an error. It appears that no transmission errors have been seen during tests involving transfers of several hundred Gbytes.

In the UA1 experiment the readout and data crates are at the side of the detector separated from the counting room and the rest of the VME system by cables of about 150 meters. A special long distance interface module is used to convert the crate interconnect cable signals to differential form as used e.g. in RS-422 specification. Peak transfer rates of 8-10 Mbytes/sec can be achieved depending on the transfer distance and the memory access times.

The crate interconnect contains also an RS-232 interface to allow a common multiplexed host connection for all CPU's in a crate for program development and testing. This interface is independent of the other functions of the device. For visual monitoring of the activities in each crate there is also a front panel LED display of VMEbus data lines D0-D15, addresses A1-A23 and the WRITE-line.

2. Theory of operation

The block diagram of the VME crate interconnect is shown in Fig. 3. Table 3 shows the crate interconnect cable signal

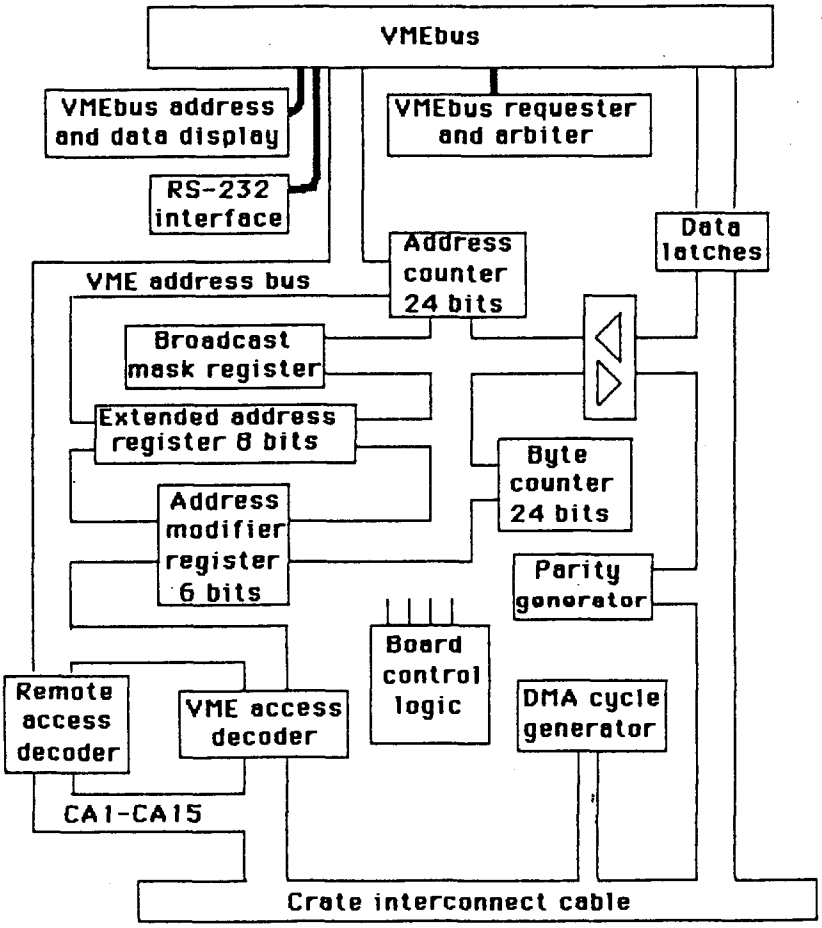


Fig. 3. Crate interconnect block diagram

assignments and their description. A complete manual with full technical details is available (ref. 3).

The data between the VMEbus and the crate interconnect cable are routed via Am 29843 high performance latches, which in the window mode operate transparently. Address and control lines are interfaced via 48 mA drive capability transceivers. All the internal logic is realised using programmable logic devices (16 circuits). The serial interface uses the enhanced programmable communications interface MC 68661 or SC2661. Sockets are provided on the crate interconnect board to plug in termination resistors. With appropriate terminations window mode of operation is possible over 100 m and DMA mode up to 15 m without long distance interfaces.

The VME access decoder has a jumper presettable VME base address, which determines e.g. the window address in the master crate, location of the module registers, and the address of the RS-232 interface. Only a part of the registers and the RS-232 interface are accessible from a slave crate interconnect module.
The individual crate interconnect registers are accessed using an offset proportional to the crate interconnect number, which is jumper presettable between 0 and 15 (restricted by fan-out requirements). The available commands and the registers are described in Tables 1 and 2. Commands arriving to a remote module (strobes MSEL and DATA, Table 3) are digitally filtered to remove any noise before an operation is performed or a remote VME cycle is started. Similar filtering occurs while receiving the data transfer acknowledge signal (CDACK).
An address counter is a write only register on each crate interconnect module for defining the start address of DMA block transfers and to hold the window base address. In the latter case the lower address lines are taken from CA1-CA15 on the crate interconnect cable.

A byte counter is a read/write register, which is incremented in steps of 2 (16-bit DMA mode) or 4 (32-bit DMA mode) until a carry out is produced stopping the DMA operation. Both the addresses and the byte counter are 24-bit wide only allowing a maximum of 16 Mbyte block transfer with a single initialisation. A32 addressing is possible using an extended address byte register and the appropriate value in the address modifier register. Extended addressing has not yet been used in the UA1 VME readout system.
A VMEbus requester and a single level arbiter is realised using PAL20X8. In normal use it is recommended that the crate interconnect is set on the arbiter priority level 3 and other VME masters on a lower level to ensure sufficient performance of the data transfers and control. In DMA mode the crate interconnect starts in release-when-done (RWD) mode and switches to release-on-request, if no other bus requests are

seen. As soon as another bus request is detected during DMA operation, the requester switches to RWD mode and releases the bus within two DMA cycles. The time-out counter during DMA waits for delays due to arbitration and triggers only to accesses where no data transfer acknowledge is received. Due to the on board arbiter and own system clock the crate interconnect can be used alone in a VME crate without other VME masters, as it appears in the data crates of the UA1 readout.

A TAS-register is a single bit location to control the access to the interrupt register. It is used to arbitrate interrupt requests among processors in the same crate. Releasing of the interrupt access is done by clearing the register. The bit is again set by the next TAS instruction executed after which the access is transferred.

An interrupt register is a 4-bit wide register. The normal contents are 0, but writing a value between 1 and 15 causes an interrupt via the crate interconnect cable signal line CINTR. This interrupt is served by the master CPU. The crates can be connected by a daisy chain cable connected to the front panel of the crate interconnect, this enables the master CPU to read the interrupt number with a single instruction. In the case of several interrupters in different crates, the first crate will be serviced first with other interrupts pending. One of the possible uses of such an inteerupt scheme is the transfer of crate interconnect cable mastership in a non-hierarchical environment. In that case mastership requests via interrupts are serviced by the old master after it has completed the activities with the network and the mastership is released under software control to the new master. Only a few microseconds in addition to the 68000 interrupt response time are needed for mastership transfer, provided the old master can release the mastership.

Table 1. Crate interconnect commands and registers

(commands dependent on module number in Table 2)

COMMAND OR REGISTER	ACTION OR EXPLANATION
CCLEAR (cable clear)	Stops DMA, ends window mode
Request mastership	Module becomes cable master
Start DMA	Source and destination modules start transfer
RS-232 registers: data, status, mode, command	Available in all modules
Module and DMA status	Indicates ongoing DMA and window modes
TAS-register	For interrupt register arbitration
Interrupt register	Interrupts master with a vector 1-15
Read interrupt source	Activates the interrupt register daisy chain

Table 2. Commands dependent on the crate interconnect module number

COMMAND OR REGISTER	COMMENT
Set window mode	Only one module at a time
Set longword mode	For D32 VME slaves
Set DMA source mode	Only one source
Set DMA destination	For destination crates
Send interrupt	Interrupt in remote crate
Reset remote crate	Removed by CCLEAR
Write address register	For DMA or window
Read/write byte counter	
Write broadcast mask	Defines modules for broadcast, also used for special DMA features
Write address modifier and extended address	For A32 mode and special address modifier values

Table 3. Crate interconnect P2-connector pin assignement and explanation of signals

Pin number	Row C	Row B	Row A
1	CINTR	+5 V	MSEL
2	GND	GND	DATA
3	CCLEAR	-	START
4	CPARITY	BA24	GND
5	DS0	BA25	GND
6	DS1	BA26	GND
7	CDACK	BA27	GND
8	CDSTR	BA28	READ
9	CA15	BA29	CA14
10	CA13	BA30	CA12
11	CA11	BA31	CA10
12	CA9	GND	CA8
13	CA7	+5 V	CA6
14	CA5	BD16	CA4
15	CA3	BD17	CA2
16	CA1	BD18	GND
17	CD16	BD19	CD0
18	CD17	BD20	CD1
19	CD18	BD21	CD2
20	CD19	BD22	CD3
21	CD20	BD23	CD4
22	CD21	GND	CD5
23	CD22	BD24	CD6
24	CD23	BD25	CD7
25	CD24	BD26	CD8
26	CD25	BD27	CD9
27	CD26	BD28	CD10
28	CD27	BD29	CD11
29	CD28	BD30	CD12
30	CD29	BD31	CD13
31	CD30	GND	CD14
32	CD31	+5 V	CD15

Signal	Active level	Explanation
CINTR	L	Interrupt line
CCLEAR	L	Initialisation
CPARITY	H	32-bit parity line
DS0	L	VME data strobe
DS1	L	-
CDACK	L	Data transfer acknowledge
CDSTR	L	DMA data strobe
CA1-15	H	Cable address bus
CD0-31	H	Cable data bus
MSEL	L	Master command strobe
DATA	L	Window data strobe
START	L	Start-DMA signal
READ	H	Data direction (except DMA)
GND		VME-board ground
+5 V		VME power line
BA24-31	H	VME extended address
BD16-31	H	VME data lines

3. Further developments

A Unibus interface for the crate interconnect has been developed (ref. 4) to enable data transfers to and from e.g. VAX computers at the maximum rate allowed by Unibus (DRE-11).
A prototype module of a CAMAC-crate interconnect interface also exists to transfer data between the VME system and a CAMAC computer at the maximum rate allowed in CAMAC. The interface will also act as an auxiliary crate controller in a CAMAC crate.
A geographical crate management system has been designed for the 68020 based processors (ref. 5) to be installed to the UA1 experiment. A small modification of the VME access decoding of the crate interconnect allows one to generate a bus error for accesses outside the 64 kbyte window, after which a new window can be set up by software. With this modification the access to remote crates can be made completely transparent to the user.
A fiber optics cable offers several advantages as a high speed link between computer systems. An implementation of the circuitry to transfer the crate interconnect cable signals in bit serial form over an optical fibre is underway. The module will have two fibre inputs and two outputs with bit rates over 150 Mbit/sec. The intention is to implement the control part as a 68020 coprocessor to be used in conjunction with general purpose 68020 cpu boards (e.g. ref. 5).

References

(1) S. Cittolin, The UA1 Data Read-out System, Talk in the conference "VMEbus in Physics", 1985

(2) C. Engster and L.G. van Koningsveld, REMUS-VME/VMX branch driver type V385, Talk in the conference "VMEbus in Physics", 1985

(3) E. Pietarinen, The VMEbus interconnect manual, University of Helsinki, Department of Physics, preprint, 1985

(4) D. Pascoli, P. Rossi et al., University of Padua (in preparation), 1985

(5) VME020 manual, Robcon Oy, PL 9, Helsinki, Finland

Acknowledgements

I am grateful to Mr T. Vehviläinen for valuable help during the assembly and installation of the crate interconnects. I am also obliged to Dr W.J. Haynes for writing a complete package of test programs for the crate interconnect.

VERTICAL BUS FOR MULTI-CRATE VMEbus SYSTEMS - VMV

Authors: J.Bovier,F.Wora
Creative Electronic Systems SA, GENEVA
CP 107 Petit-Lancy SWITZERLAND

A Multimaster VMEbus Extender: The CES concept.

Foreword:

The CES vertical bus - called VMV in this document - has been designed to provide an efficient and simple solution to the data handling problem in large data acquisition systems requiring several VME crates. The transfers on the VMV bus are handled in completely transparent mode for the local CPUs and are executed at the maximum VME speed. The system utilizes a tree structure with up to 15 crates at each level of the tree

The hardware:

The VMV interface units consist of a pair of single width VME boards.A solution with two separate units allows the best preformance/price ratio in each application, because in mosts of the cases only one board is required per crate.

- The receiver unit : VBR 8212

These units are similar to CAMAC Crate Controllers.These Master VME boards receive the cycles coming from the VMV bus and transform them into VME cycles in the crate where they reside. A VMV Slave Crate requires only a VBR 8212.

- The transmitter unit : VBE 8213

These units are similar to CAMAC Branch Drivers.They receive the VME bus cycles from the crate where they reside and transform them into VMV bus cycles for other VME crates. A VMV Master Crate requires both a VBR 8212 and a VBE 8213.

The transfers:

The transfers on the VMV bus are of the asynchronous type as in the VME standard. The VMVbus works on 32 bits addresses and 32 bit data.All the VME parameters are transmitted between the source crate and the destination crate. All types of transfers

- i.e. D8, D16 , D32 - are supported for the following transactions:

- READ
- WRITE
- READ MODIFY WRITE
- BLOCK MODE
- ASCENDING MODE

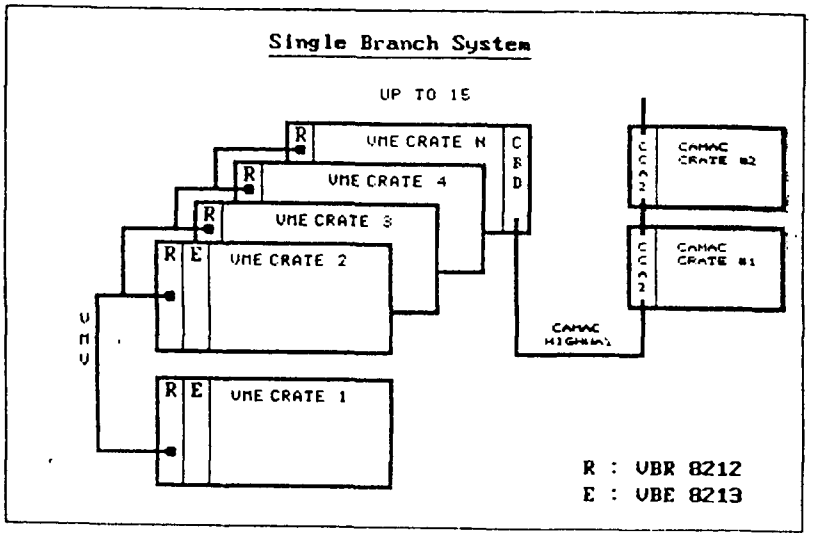
The Handshake between the Master Crater and the Slave Crate is maintained during the complete transaction.

Performances:

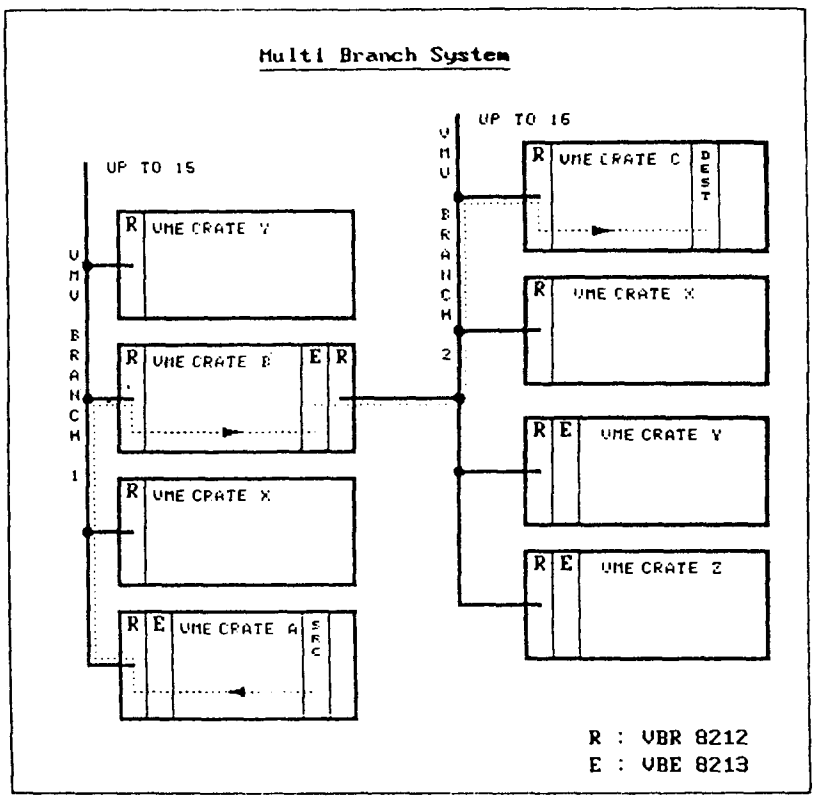
A special emphasis has been given to the speed and smoothness of the transactions. The system bandwith is superior to 10 Mbytes for short distances (one rack) with a full handshake. The maximum length of the bus is around 100 meter. The signals are distributed in differential form and allow "wired-or" connections. Each line is terminated on its characteristic impedance.

Typical systems:

Systems can be monobranch or multibranch. A typical mono-branch system consists of one crate equipped with a VBR and a VBE , and of N crates equiped with VBR units. A typical multi-branch system consists of one crate equipped with a VBR and a VBE dealing with a first branch of N1 crates equipped with VBR units where one - or more- of these crates is equipped with another pair of VBR/VBE units dealing with a second branch of N2 crates equipped with VBR units.



In this system, crate 1 and crate 2 may reach all other crates through the VBE units. The crate N contains a CAMAC branch driver to drive CAMAC crates from VME.



In this system crate A connected on Branch 1 may reach crate C connected on branch 2 through the relay units residing in crate B.

Crate Addressing:

Crate addressing is implemented "geographically". Up to 15 receiver crates can be connected to the VMV bus;the actual number beeing defined by a rotary switch on the front-panel of the VBR 8212 unit.

Two modes of crate addressing are provided by the vertical bus:

- standard mode where a single crate is selected.
- multi-crate mode in which all on-line crates of the branch are selected.

Multi-crate mode is allowed during "DIRECT" or "VME-VME" cycles and is possible because "wired-or" inherent in the vertical bus structure. Multi-crate mode is selected by crate address "0".

Physical addressing:

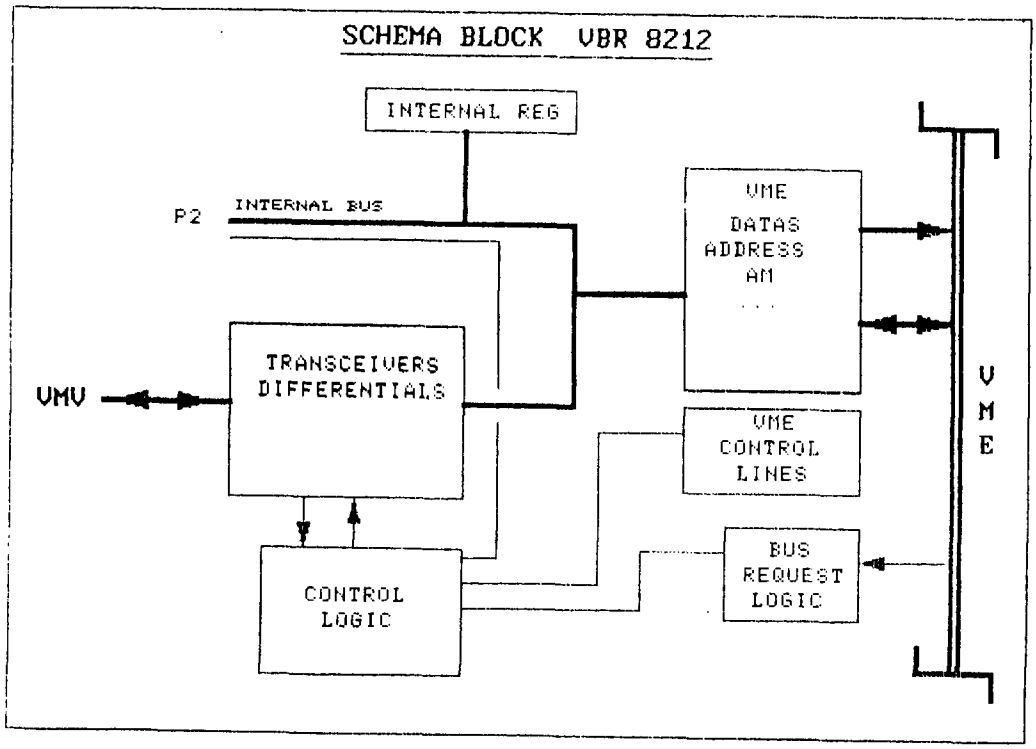
There are two distinct ways to transmit a VME cycle between source and destination crates.

a) "window mode" An 1 Mbyte window mapped in the source crate's address field allows access to the other crates in the branch. Destination missing information - high address, crate number, is obtained directly from buffer registers.

b) "auto-routing mode" The destination crate number is mapped directly within the address space of the source crate. (i.e. 4Kbyte to 16Mbyte are direcly accessible for each destination crate.)

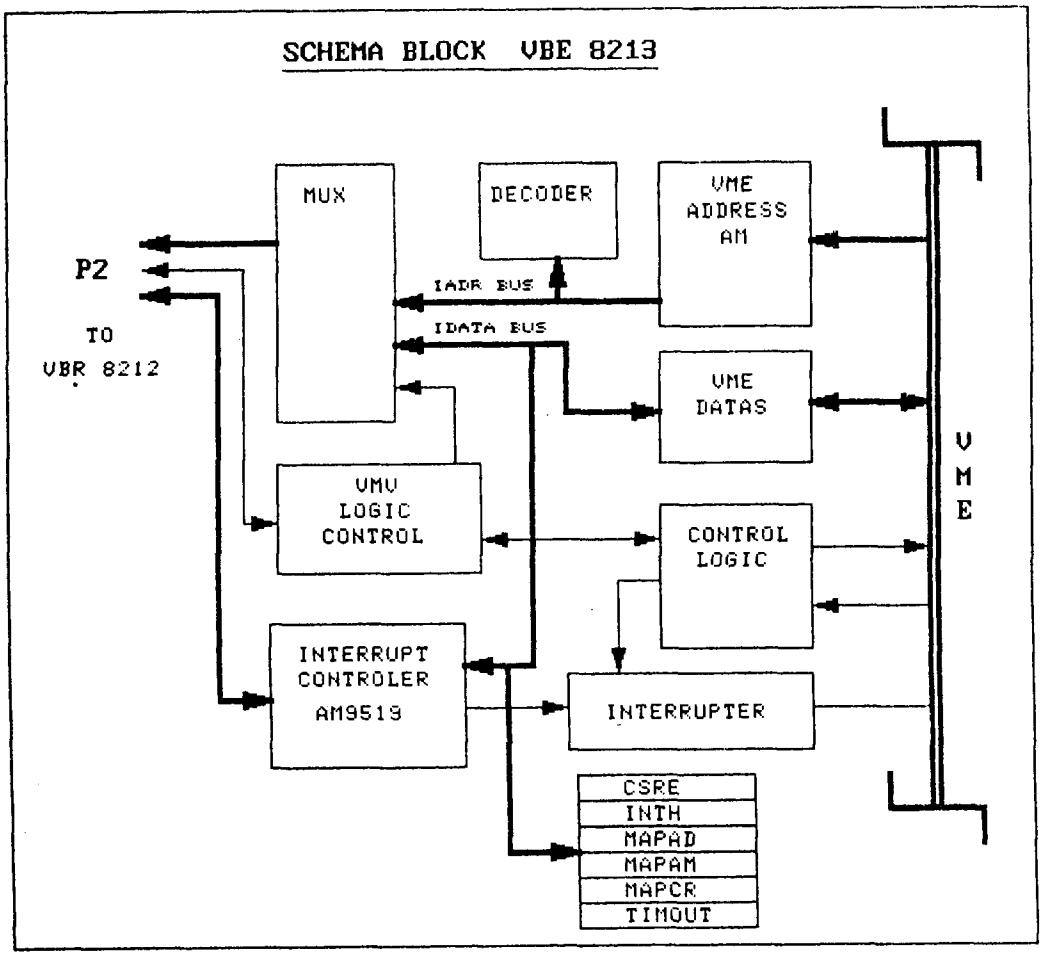
The Vertical Bus Receiver

The VBR 8212 is the Crate Controller for Vertical Bus cycles; it converts VMV protocol to VME standard cycles, implementing all VME cycles (A32,D32,D16,D8) in all modes. This is the only unit furnished with a differential interface for the vertical bus. Transceivers are positionned on the lower part of the card.P2 is a connector for an adjacent VBE 8213 Emitter.
A piggy-back bus terminator, BVT 8214 is required for each of the VBR 8212 units at the ends of the branch (and on no others). The VBR 8212 is a VME master (like a CPU), possessing arbitration logic and controlling the VME cycles which it issues. It also posseses interrupt handling abilities.



Vertical Bus Emitter

The VBE 8213 is a VME slave unit.It converts transfers destined for other crates from VME to VMV protocol.It contains no differential interface, thus can only work in conjunction with an adjacent VBR 8212 which is connected via rows A and C of plug P2. The unit is equipped with arbitration logic so that severeal VBE 8213 may share the same vertical bus.
Vertical bus cycles are generated automatically for transfers destined for other crates.The VBE 8213 interrupts its own crate at a programmable level,stimulated by an internal event or by another crate of the system.
The unit's internal registers are only addressable as 16 bit words.



CSRR :

VMEM	ACF	-	ARB1	ARB0	IS2	IS1	IS0	IN4	IN3	IN2	IN1	ID3	ID2	ID1	ID0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CSRE :

-	VARB1	VARB0	VBOK	NOREP	ITO	ADM	AMM	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

A VME MULTIPROCESSOR ARCHITECTURE FOR THE LEP/SPS CONTROL SYSTEM

J. Altaber, P.G. Innocenti and R. Rausch

LEP/SPS Controls Group

Abstract The control system for the LEP collider follows the concepts developed for the SPS accelerator but making use of present-day technology. Conventional minicomputers are replaced by Process Control Assemblies constructed in the VME standard using the 68000 μp family. LEP and SPS equipment will be controlled by intelligent interfaces built in G64 or VME crates and linked to the Process Control Assemblies by the MIL-1553-B multidrop bus. These Process Control Assemblies will be interconnected by ring networks following the IEEE 802.5 token-ring protocol. The paper presents the advantages of a VME multiprocessor functional architecture compared to the conventional mini-computer approach and discusses the merits and limitations of the VMEbus for the implementation of flexible multi-processor systems.

1. INTRODUCTION

CERN is constructing a Large Electron Positron (LEP) collider on the Swiss/French border. The new machine has a circumference of 28 kms and is located in an underground tunnel at 60 to 120 m depth below the surface of the ground. Equipment will be installed into the tunnel and be controlled from underground alcoves and from surface buildings located at eight equidistant areas. LEP is expected to start operation early 1989.

The LEP control system will use a fully distributed computer system made up almost entirely of closely coupled multi-microprocessor assemblies[1,2].

The LEP machine will be operated from the same control room as the SPS accelerator, in operation since 1976. It follows the same philosophy of functional and geographical distribution but takes advantage of to-day's technology. Instead of conventional mini-computers the LEP control system will use assemblies of microcomputers built in VMEbus standard[3] linked by a token passing ring network conforming to the ISO P.8802.5 standard protocol, Fig. 1.

The existing SPS control system will be progressively upgraded by using the same VMEbus multiprocessor architecture and the token passing ring network. Already new equipment required for the e^+e^- operation of the SPS uses VME instead of CAMAC.

CERN is currently consulting computer manufacturers to identify and chose a standard VME distributed multi-microprocessor system together with a distributed real-time operating system industrially supported[4].

The purpose of this paper is to describe the multi-microprocessor architecture being developed in VME at CERN, Fig. 2. The objective of this project is to verify the validity of these concepts for real-time control of a large accelerator and to assess its relative merits and limitations compared to a conventional implementation[5,6,7]. The technical requirements to be satisfied by a multi-processor architecture, the interprocessor communication, and the reservation and protection mechanisms implemented are discussed.

The merits and limitations of the VMEbus for constructing a true distributed multimaster-multiprocessor are presented. Several improvements to overcome these limitations have been implemented and are described in this paper.

2. THE PRINCIPLES

In the past, several multiprocessor architectures have been proposed differring mainly by the degree of coupling and ranging from a network of distributed processors to tightly coupled micro-computers connected to a high speed multimaster parallel bus[8,9]. The CERN project described here is of the closely coupled type and uses the high bandwidth of the standard VMEbus for interprocessor communication without the aid of a central controller.

Modularity and expandibility, the primary goals in the hardware design, are ultimately aimed at drastic reductions in design time and cost for customized systems of varying degrees of complexity and software simplification. Although the microprocessor is a relatively slow device, the parallelism obtained in closely coupled multiprocessors greatly enhances system speed and throughput. In addition, future expansion of the system is accomplished easily by adding more microprocessors without any significant increase in the overall cost.

Another attractive capability of the multiprocessor approach over the single multitasking minicomputer is the potential for increasing the reliability. Reliability can be increased by allowing more than one microprocessor to perform the same critical tasks in parallel.

3. THE MULTIPROCESSOR ARCHITECTURE

The multiprocessor architecture CERN has developed for its pilot-project, and for which CERN is currently consulting VME manufacturers, is of the fully distributed multimaster-multiprocessor type. This means in particular that all VME processors in a crate are real masters of the VME backplane, thus up to 20 CPU modules can take the mastership of the VMEbus for inter-communication. This approach is to be distinguished from the more popular concept of a single multitasking processor controlling a number of intelligent modules containing a microprocessor but each behaving like a slave module on the VMEbus. This latter arrangement also is sometimes claimed to be a multiprocessor system. In CERN's fully distributed multi-master-multiprocessor system each VME processor module contains a resident real-time software kernel and an interprocessor communication mechanism.

3.1 Function-to-Function Architecture

A Function-to-Function Architecture (FFA)[10] has been implemented for the CERN multi-microprocessor project. The FFA is a generation of memory-intensive microcomputer boards and software modules completely interchangeable within the system structure. FFA is the technique for distributing functionally powerfull computing elements throughout a system. This architecture provides familiar structures and simplified procedures allowing users to benefit from highly advanced microcomputer technology.

In a conventional computer and input/output system, the three functions, shown as F1, F2 and F3 in Fig. 3, are executed in turn by multiplexing the hardware resources between the functions as required. Thus only one function at a time can be active. Even if one CPU was dedicated to each function, the overall system performance would not improve appreciably because all activity is bound by the bandwidth of the bus.

The function-interconnect bus of Fig. 3 represents the form of FFA, and resolves the bus crowding problem. Each functional module contains sufficient processing, memory and I/O capability to execute the required

functions collected within the module. The effective bandwidth of an FFA system can be as high as the sum of its individual microcomputer bandwidths. The bandwidth of the bus that interconnects these microcomputers is a function of the volume of inter-processor messages and shared data traffic. Modular computing functions can contain the hardware and software for implementing one or several pre-programmed functions.

The multiprocessor architecture is aimed at the control of the LEP and SPS accelerators. A large number of these multiprocessor systems will be used and they will act as Process Control Assemblies (PCA). Each PCA contains several functionally dedicated processing units, some of them are associated privately with Input/Output cards or private memory extension boards, Fig. 4. All processing units are identical in the current implementation, using a Motorola 68010-12.5 MHz processor and show to the function-interconnect bus the same logical multimaster structure, Fig. 5. More powerful processors, 68020 with 68881 arithmetic coprocessor, will be used in the future if required for some specific function but the physical and logical connection to the bus will be identical to the present one.

Some of the functional modules are:

- A Data-Link unit (DL) interfacing each PCA to the Token Passing Ring network conforming to the ISO-P.8802.5 protocol.

- A Supervisory Unit (SU) managing the activities in a PCA.

- One or more NODAL Interpreter modules (NI) interpreting the high level commands.

- A FORTRAN Processing unit (FP) when fast data processing is required.

- An Equipment Directory unit (ED) which acts primarily as the local data-base.

- Several Multidrop Bus controller (BC) modules conforming to the MIL-1553-B protocol linking many Equipment Control Assemblies (ECA) to the PCA for control and monitoring.

Additional functions may be added to a PCA to complement the basic functionality, i.e. graphics, man-machine interfaces, mass storage drivers, etc. Modules can evolve at their own pace changing and improving with advances in technology; this function-to-function architecture provides a clear path to 68020 32-bit multiprocessor configurations.

3.2 Message Passing Protocol

The function-to-function architecture implies a message passing protocol between microcomputer boards for data exchange. This concept is extensively supported in the MULTIBUS II architecture[11]. Two intermodule communication protocols are generally used in multiprocessor systems "pass-by-reference" and "pass-by-value". In "pass-by-reference" the communicating modules exchange pointers to gain addressability to the shared data structures, while in "pass-by-value" modules exchange a copy of the data structure. Both methods have their relative merits and inconveniences.

The CERN multiprocessor pilot project uses a flexible event interrupt method to communicate the type of message to be transferred and the "pass-by-value" method for the data exchange between the source processor and the destination processor.

The ability of all processors to share portions of physical memory is fundamental to the design of a closely coupled multiprocessor system. CERN's design allows a processor access to a local physical memory that, while reserved primarily for its own use, is also shared among other processors. Arbitration logic resolves competition for use of each processor's memory bus. The principle of a safe "circuit-switching" whereby a direct signal path is established between a source memory and a destination processor is implemented, Fig. 6. The shared local memory is partionned in small blocks of 256 words each allocated to one corresponding source processor. The memory mapping and the address generation and decoding is arranged by hardware logic in such a way that any processor can physically only access its own predefined and reserved area in the shared memory of any other processor. Thus corruption of data is prevented by hardware protection and moreover is reinforced by a "Read mostly" approach of the messages out of shared memory. This organization of the memory mapping allows the maximum number of pluggable processors (20) in a VME crate. The multicrate architecture developed at CERN for the VMEbus extends theoretically this capability up to 256 processors.

3.3 Resource Allocation and Protection

With functional partitioning of the tasks only the data that must be shared by individual processors passes over the function-interconnect bus. Resources may be either private, thus only accessible by a single processor or resources may be global and shared by several processors. In this latter case a reservation mechanism by semaphore and TEST and SET functions associated with a source signature mechanism has been implemented following the ESONE E3S standard system specification proposal[12]. One byte of the 24-bit address on the communication bus, representing the source processor identifier, is automatically loaded into a signature register of the resource during a successful reservation access. From this instant on, this signature becomes part of the full 24 bit address decoding while before reservation the resource responded to 16 bit address. Subsequently only the successful processor can access this resource or release it. Deadlock situations are monitored by the supervisory unit.

3.4 Software aspects

The above description of the hardware structure for the LEP/SPS control system has an important consequence on the conception of the software.

Such a structure allows the cutting of the global control problem into units of practical and manageable size which can be solved separately. This fragmentation capability is firstly exploited by assigning a PCA to each geographical/functional subsystem such as the beam instrumentation, accelerating system, vacuum, power converter. Each of these PCA's will communicate via the token ring essentially by means of messages containing programs or data in a canonical form.

A further level of fragmentation is achieved within a PCA via the function-to-function architecture. This architecture, based on autonomous General-purpose Processing Units (GPU) emulates a conventional real-time multi-tasking operating system. The communication between the units is made over the VMEbus by means of messages which contain requests for service and data.

This implies that the elementary software package is a function which, when it is well defined, is simple to develop and debug. The overall distributed operating system is made by combining, in a PCA, the appropriate functions.

The essential element for such a system to work is a well defined communication convention between GPU's in a PCA, and between PCA's. The information which flows on both communication systems is basically of the same nature, they are messages. The ISO Open System Interconnection reference model is well suited for this kind of problem; it offers a coherent framework for this implementation of a uniform scheme for the data flow in the overall system[13].

Another important aspect of the hardware structure comes from the fact that a function being logically defined is independent of the hardware module which realizes it physically. If tomorrow a better hardware module is found for performing a specified function we could use it without any major change in the design.

For this statement to be fully true it requires that the software which was in the module can be mostly re-used. The achievement of this requires the use of a high level language widely accepted in order to guarantee the availability of compilers. It has been decided to use the Modula-2 language which offers all the necessary features for writing the operating system programs.

4. THE VMEbus USED FOR MULTIPROCESSING

4.1 Advantages and limitations

In 1982 CERN choose the VMEbus and the 68000 microprocessor family for all future projects following the recommendations of an internal Working Group. Thus the multiprocessor pilot project for the LEP/SPS control system has been implemented in VME. The large industrial VME support allowed us to concentrate our effort on the implementation of the distributed multi-processor concepts and to use standard products available from many VME manufacturers for all common facilities like: SASI drivers, display functions, diagnostics modules, memory extensions, crates, busses and power units, etc.

While the VMEbus specification provides good facilities for a few master-processors plugged in a crate its architecture is not ideal for true distributed processing with up to twenty closely coupled microcomputers communicating by message protocol in multimaster mode. To achieve its goal CERN has studied and developed new facilities fully compatible with the VMEbus specification. In the future some of the missing facilities will be

provided by the serial VMSbus chip set. These compatible developments and operation procedures include a distributed arbitration mechanism, a programmable interrupt system, a specification for an extension of the microprocessor private bus over P2 connector and a multimaster-multicrate highway linking up to seven VME crates. All these extensions allow the use of standard VME manufacturers' products.

4.2 Distributed arbitration

A fully distributed multimaster-multiprocessor architecture must provide an arbitration mechanism suitable for an unlimited number of independent processors in the same crate. Most new multiprocessor busses have adopted the distributed arbitration mechanism: *FASTBUS*, *SMBUS*, *MULTIBUS II*, *NUBUS*, P896 FUTURE BUS. Distributed arbitration offers position independence and the possibility to implement easily fairness and dynamic high priority amongst an unlimited number of independent processors. The fairness algorithm ensures that every board gets its fair chance to access the bus; no board is forever locked-out by a higher priority master. The high priority algorithm allows the designer to define a priority structure among all of the boards in the system. Modules handling real-time events are usually given the highest priority while less critical boards get a lower priority. In the CERN pilot project this capability is used extensively for passing urgent messages and event-like interrupts between processors. Changing between the two algorithms can be done dynamically; the designer is free to choose one or the other, or a combination of the two. Distributed arbitration avoids the burden of manipulating numerous daisy-chain jumpers on the backplane in case of system reconfiguration, as processor's relative priority is position independent. But the biggest advantage given by distributed arbitration is the diagnostic possibilities. While it is difficult with the daisy-chain mechanism to know at any instant which processor has generated the current erroneous bus cycle it is straight forward with distributed arbitration. Each bus cycle is accompanied with the priority code of the current master processor thus, bus activity monitoring can be flexible and very selective; a bus error can be trapped and attributed to the faulty master.

CERN's development allows to mix in the same VME crate modules working with the daisy chain arbitration and modules using a distributed arbitration approach in conjunction with a special system controller plugg in slot 1[14].

The implementation of the VME compatible distributed arbitration uses three of the four pairs of Bus-Request Bus-Grant In/Out lines. BR3 and BG IN/OUT3 are not touched but are reserved for simultaneous daisy-chain VME arbitration. All three BG IN/OUT pairs of pins are strapped at all module slots including the first and the last positions. This gives six lines (BR0, 1, 2 and BG 0, 1, 2) which are terminated on each side of the dataway by the standard VME resistor network. These six lines are used for priority encoding; five lines for priority 0-31 and the 6th for exceptional high priority access, typically for urgent messages or event-like interrupt cycles as explained later. Processors developed for the CERN multiprocessor pilot project support either arbitration methods by replacement of a macro-component mounted on a small daughter board. To provide full mixing capability the system controller in slot 1 combines the daisy-chain arbitration BR3 with the distributed arbitration and leaves the BR3 chain for top priority as specified by the VME standard.

4.3 Module and crate addressing

The VMEbus specification allows for Standard Address Modifier codes (AM) and User Defined AM codes. For the design of a modular multiprocessor system it is very useful to introduce the concept of Crate (C), Module (M) and Register (R) addressing similary to CAMAC. For our multiprocessor pilot project, the ESONE - E3S proposal for a standard system specification has been implemented[12]. This document recommends a particular memory and peripheral addressing, for which two VME User Defined AM codes have been allocated in addition to the usual AM codes. A homogenous internal register addressing for processor and peripheral modules is recommended, also free or protected access methods to common memory or peripheral resources are specified. More information about the E3S implementation for this project is given in Ref. 13.

Having specified a crate number and a module number several advantages can be gained. It becomes easy to define a general source and destination concept for interprocessor transactions in single or multiple crate systems. It is also simple to protect hardware wise access to resources by a source signature mechanism carried over the VMEbus by each individual read or write cycle. In addition, a generalized programmable event-like interrupt method (described in section 4.4) can be implemented and last, but not least,

multiprocessor systems can be debugged more rapidly and more efficiently by identification of the processor's number which is the current master of the VMEbus.

4.4 Programmable Interrupt System

The VMEbus provides seven Interrupt Request lines IRQ1-7 and a single daisy-chained Interrupt Acknowledge line. In a multiprocessor system the seven IRQ lines may be activated each by any processor or slave module and each request line ends in one of seven processor boards. Unless several interrupt sources share a given interrupt line using software polling, and the same interrupt line is connected to several destination processors, seven is the maximum number of bus-master processors one can use in a VME crate system. Each interrupted processor has first to become the master of the bus before it can acquire the interrupt vector. This is a serious limitation for the implementation of a flexible multiprocessor architecture. When the VMS chip set become available this VME problem will partially be solved. The CERN project has therefore developed a programmable and addressable interrupt-event concept which is nowadays becoming used on other standard busses[11].

The principle is the following: when a source processor needs to interrupt a destination processor it requests bus mastership with high priority, i.e. its normal priority code plus activation of the sixth priority code line. The source processor performs a programmable write cycle associated with a 16 bit interrupt vector data word; the high byte contains the source processor code (3 bits for crate number and 5 bits for the processor number) and the low byte contains the interrupt vector code. The 16 bit address is generated with the standard short I/O address modifier code. All processors contain an Interrupt FIFO (16 bit wide, 256 words deep) receiving these interrupt vector codes. Thus each processor can interrupt or be interrupted by up to 20 processors in a single crate system and by up to 140 processors in a multicrate system with seven VME crates. The multimaster-multicrate highway carries these interrupt vector words as any other programmed Input/Output transaction.

This simple but very powerful interrupt mechanism uses the dynamical high priority arbitration implemented either with the four-level daisy-chain priority or with the distributed arbitration as explained above. It allows straight forward interrupt source identification provided that a coherent

module and crate memory address-mapping has been adopted during system configuration. This method can be seen as memory-mapped interrupts where the source module performs a write into the register address space of the destination processor to trigger an interrupt. This creates so many "virtual" lines that the designer has not to worry about running out of interrupt lines among modules.

4.5 Local Bus

In the CERN multiprocessor project the VMEbus is used mainly as a function to function interconnect bus carrying messages from source to destination processor. Considering the small size of the VME card and the functionality required from the various microcomputer boards an extension capability is often required for additional memory or private I/O. This extension can be done via rows a and c of the P2 connector, either by following the VMXbus specification or by using these 64 pins freely as User I/O pins according to the VMEbus specification. For private memory extension on a local bus the full microprocessor speed must be usable without introducing wastfull wait states and additional logic between the microprocessor and its memory. As VMXbus specifies multiplexing of address and data it has been considered too stringent to multiplex A/D at twice the microprocessor speed to benefit from the full microprocessor power. Thus to satisfy CERN's requirements a straight forward expanded local bus for the 68000 processor family has been specified on the 64 User I/O pins and named VMCbus[16].

The VMCbus is an extension of the 68000 processor bus to provide a high speed secondary path which is optimised for connecting up to six boards in a subsystem configuration. This subsystem can transfer data from board to board over its VMCbus interface without waiting for and without burdening the primary bus (VMEbus). A set of two or more modules interconnected via the VMCbus behaves like one big VMEbus module with its own internal bus, freeing the VMEbus for other transfers.

The 68000 vectorised and autovectorised interrupt mechanism is provided over the VMCbus allowing any card connected to the VMCbus to interrupt the processor card directly, thus saving unnecessary mastership arbitration in a multiprocessor system over the VMEbus with improved response time. The 68000 Direct Memory Access mechanism is supported as well over the VMCbus and multiple DMA masters can communicate over the VMCbus with the processor's private memory.

The VMCbus specification permits to use directly all 68000 family peripheral chips as the timing and control signals and the address and data lines are those of the 68000 processor.

A board that is inserted from the front of the card rack may have a VMEbus interface, a VMCbus interface, or both. For example, a global I/O board might have only a VMEbus interface, a CPU or a memory board might have both; while a private memory, a private I/O or a math-processor might have only a VMCbus interface. Any board that has a VMCbus interface uses the two outer rows of the J2/P2 connector.

The use of a ribbon cable or alternatively a small PC backplane to bus the P2 connectors allows any group of up to six adjacent slots to function as a subsystem. The user can install two or more P2 cables or backplanes to create several VMCbuses in a single card rack. Each of these VMCbus subsystems can operate independently of the primary system bus (VMEbus) and independently of each other. It also permits some slots to be used for other purposes such as I/O signals.

4.6 Multimaster-multicrate highway

An accelerator control system needs provision for connecting numerous equipments to the local process computer. At the SPS for example, up to eight CAMAC crates are controlled by the minicomputer I/O bus and up to eight serial multidrop busses (MPX system) connect 32 crates each to serial drivers located in CAMAC crates. In addition local intelligence has to be associated logically with some I/O subsystems, similarly to CAMAC-ACC's. As a consequence for the LEP/SPS control system an homogenous communication amongst these intelligent VME process and I/O crates is necessary for large and powerful systems. This example demonstrates the need for a multicrate highway allowing several VME crates to interface equipment either directly in a point to point distribution or via a MIL-1553-B multidrop serial bus over long distances.

The LEP/SPS multiprocessor assemblies must be expandable over more than one VME crate. To satisfy this requirement a VME multimaster-multicrate highway has been developed[17].

For the multimaster-multicrate highway (Fig. 7) the following design criteria have been fulfilled:

- All standard VMEbus cycles with all address modifiers are supported.

- Full 32-bit address and data are transferred over the highway in an A/D multiplexed mode.

- Differential drivers and receivers according to RS-485 use a twisted pair line for each signal allowing interconnection of seven VME crates over distances of 30 meters.

- Each VME crate can support several processor modules capable of gaining mastership of the highway and accessing memory or I/O resources in any other crate. A daisy-chain arbitration organises the priority amongst the VME crates. Each highway bus linker module has top priority in the crate it is located to minimise arbitration time in the destination crate. Crates not concerned by a data transfer over the highway do not participate in the arbitration; only the source and destination crates are involved in a transaction.

- Data transfer can be fully interleaved at a cycle by cycle basis, but sending small data blocks saves arbitration time.

- Interrupt-like events are sent over the highway by the method described in section 3.2. Any slave or any processor can send its interrupt vectors with source identification to any other processor in a single high priority transaction.

- When a processor tries unsuccessfully to access a remote resource over the highway due to heavy traffic it uses the microprocessor's re-run facility to try again later. Bus error conditions and deadlock situations have been taken care of.

5. MERITS OF THIS VME MULTIRPOCESSOR ARCHITECTURE

Compared with conventional computer systems the function-to-function multiprocessor architecture implemented in VME for the LEP/SPS controls system offers a number of present and future advantages:

- This architecture allows simple designs of complex systems by permitting system requirements to be broken down into identifiable, manageable tasks associated with real world needs (DL, BC, SU, ED, MS, NI, FP modules, etc.).

- It provides excellent system throughput, with very low bandwidth over the function-interconnect system bus (VMEbus) and very high bandwidth through the private local bus (VMCbus) of each processing module.

- Low cost, high density electronics following VLSI improvements is offered. Tomorrow's components will permit to build functional modules tailored to an application using just the components necessary for processing, memory and I/O operations. (Replacement of VMC and VME cards by a single equivalent VME module.)

- Functional modules can be grouped into clusters or duplicated to raise significantly the performance level of a particular function. (Several NI modules in a PCA allow parallel interpretation of commands expressed in high level language.)

- Functional modules can be duplicated to increase a function's reliability. (In a PCA two DL modules allow connection on a dual communication network for redundancy.)

- This architecture allows hardware/software transportability of functions without impacting system operation. Versions of the same function can be implemented in software, embedded in firmware with a microprocessor, or for highest performance, designed in hardware. (In the future, a language processor module, NI or FP, may be implemented by a special microprocessor with an instruction set optimized for this language.)

- Designs are simple to upgrade or modify provided the VMEbus interconnection logic remains the same. Functions litterally can be plugged into or be removed from an existing system with no adverse effects on control or scheduling operations.

- Alternatively FFA functions are bus independent. Functions developed for the VMEbus can operate without modification on other computer system buses. Only dedicated bus-coupling circuitry adjusts for differences.

- This architecture offers economy by permitting functions to be implemented in any technology that can perform the desired function.

This also applies to software and hardware. A function constructed in software can be converted to hardware to satisfy the higher performance requirements of a particular part of a system.

6. CONCLUSIONS

It is most interesting to compare the mini-computer based architecture of the SPS controls system built in 1972 and the VME-based multimaster multiprocessor architecture planned for LEP and SPS in 1987-1988.

The first SPS generation consisted of a NORD-10-CPU with 16 kword core memory, and a 128 kword drum filling a 1.5 meter height, 19 inch rack, a separate crate for the point-to-point star network connection and seven CAMAC crates driving up to eight MPX multidrop links in command/response mode.

The initial LEP installations will be based on a single VME crate with up to 20 processors of the 68000 family each having at least 1 Megabyte of memory and a token-ring network interface module. In addition, up to eight MIL-1553-B multidrop bus controllers will plug in that VME crate; each controller will drive in message mode a cluster of 32 equipment crates containing as well at least one microprocessor.

A new generation of mini-computers (NORD-100) had to be installed in 1980 to upgrade the SPS control system. Today the 16-bit CPU architecture shows its limitation and a new upgrade is to be envisaged by the end of the decade.

The main advantage expected from using VMEbus is the fact that it is an OPEN SYSTEM available from many manufacturers, controlled by an international standardization body and supported by a strong User's Group. We are confident that its lifetime will be much longer than that of any PROPRIETARY SYSTEM and that a smooth upgrading to full 32-bit multi- processor systems can be achieved by this VME functional architecture, thus saving the large capital investment of the future LEP/SPS control system.

REFERENCES

1. J. Altaber and R. Rausch, A Multiprocessor Bus Architecture for the LEP Control System, Proc. National Particle Accelerator Conference, Santa Fé, 1983.

2. M.C. Crowley-Milling, The Control System for LEP, Proc. National Particle Accelerator Conference, Santa Fé, 1983.

3. J. Altaber, M.C. Crowley-Milling, P.G. Innocenti and R. Rausch, Replacing Minicomputers by Multi-microprocessors for the LEP Control System, Proc. National Particle Accelerator Conference, Santa Fé, 1983.

4. Technical Specification for the Supply of Software and Hardware for the Multiprocessor Control Assemblies, Tender I-1398/LEP, CERN, Geneva, July 1985.

5. J. Altaber, F. Beck, M.C. Crowley-Milling and R. Rausch, Suggested Principles for the Control of Future Accelerators, IEEE Trans. Nucl. Sci., Vol. NS-26, No. 3 (1979).

6. J. Altaber, F. Beck and R. Rausch, Accelerator Control Systems without Minicomputers, XIth Intern. Conf. on High-Energy Accelerators, CERN, Geneva, 1980.

7. J. Altaber, F. Beck, M.C. Crowley-Milling and R. Rausch, Truly Distributed Control Using one Microprocessor per Real-Time Task, IVth International Conference on Trends in On-Line Computer Control Systems, Warwick, 1982.

8. C. Ali, A.D. Hirschman and R. Swan, Standard Modules offer Flexible Multiprocessor System Design, Computer Design, May, 1979.

9. J.P. Barthmeier, Multiprocessing System Mixes 8 and 16-bit Microcomputers, Computer Design, February, 1980.

10. M. Conrad,W.D. Hopkins, L. Spry, A. Orban, A. Nawaz and I. Ansari, Functional Architecture Threatens Central CPU's, Electronic Design, September, 1981.

11. J. Beaston, Intel Corp., Multibus II - An Architecture for 32-bit systems, Electronic Engineering, March, 1984.

12. E3S - ESONE Standard System Specification, Proposal submitted to the ESONE General Assembly, Berlin, 1983.

13. J. Altaber, V. Frammery, C. Gareyte, R. Rausch and P. van der Stok, The Message Architecture of the LEP Control System, Nuclear Science Symposium, Orlando, 1984.

14. R. Rausch, J.M. Sainson and G. Surback, Distributed Arbitration Compatible with the VMEbus Standard, internal report CERN SPS/ACC/Note/83-18 (1983).

15. E3S for the LEP multiprocessor control system, CERN, october, 1984.

16. VMCbus Specification Manual (1984), internal report CERN SPS/ACC/Tech.Note/85-2 (1985).

17. R. Rausch, Bus Multichassis-Multimaître compatible VME, Conférence sur les Systèmes à Microprocesseurs, Ecole Polytechnique, Paris, 1984.

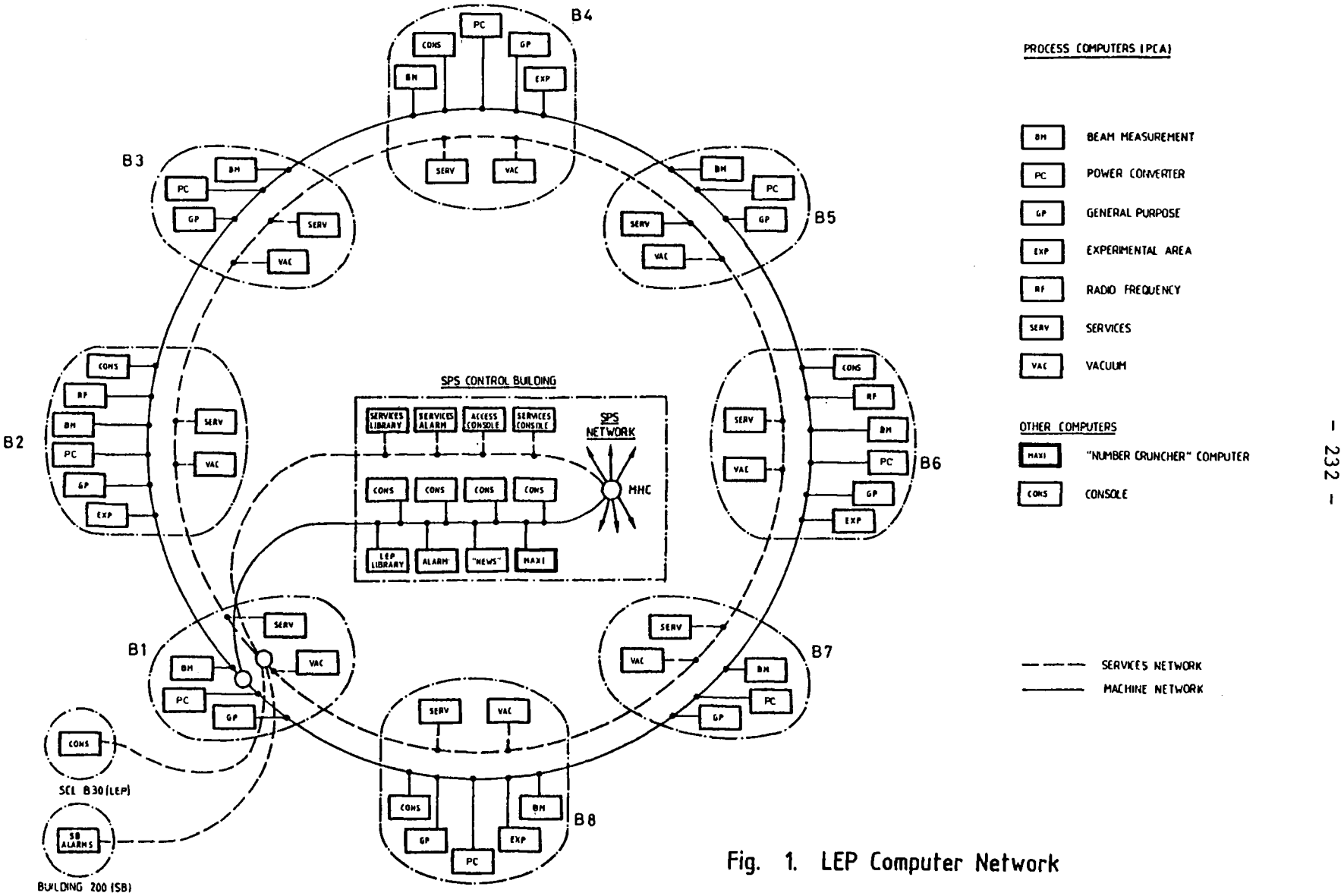


Fig. 1. LEP Computer Network

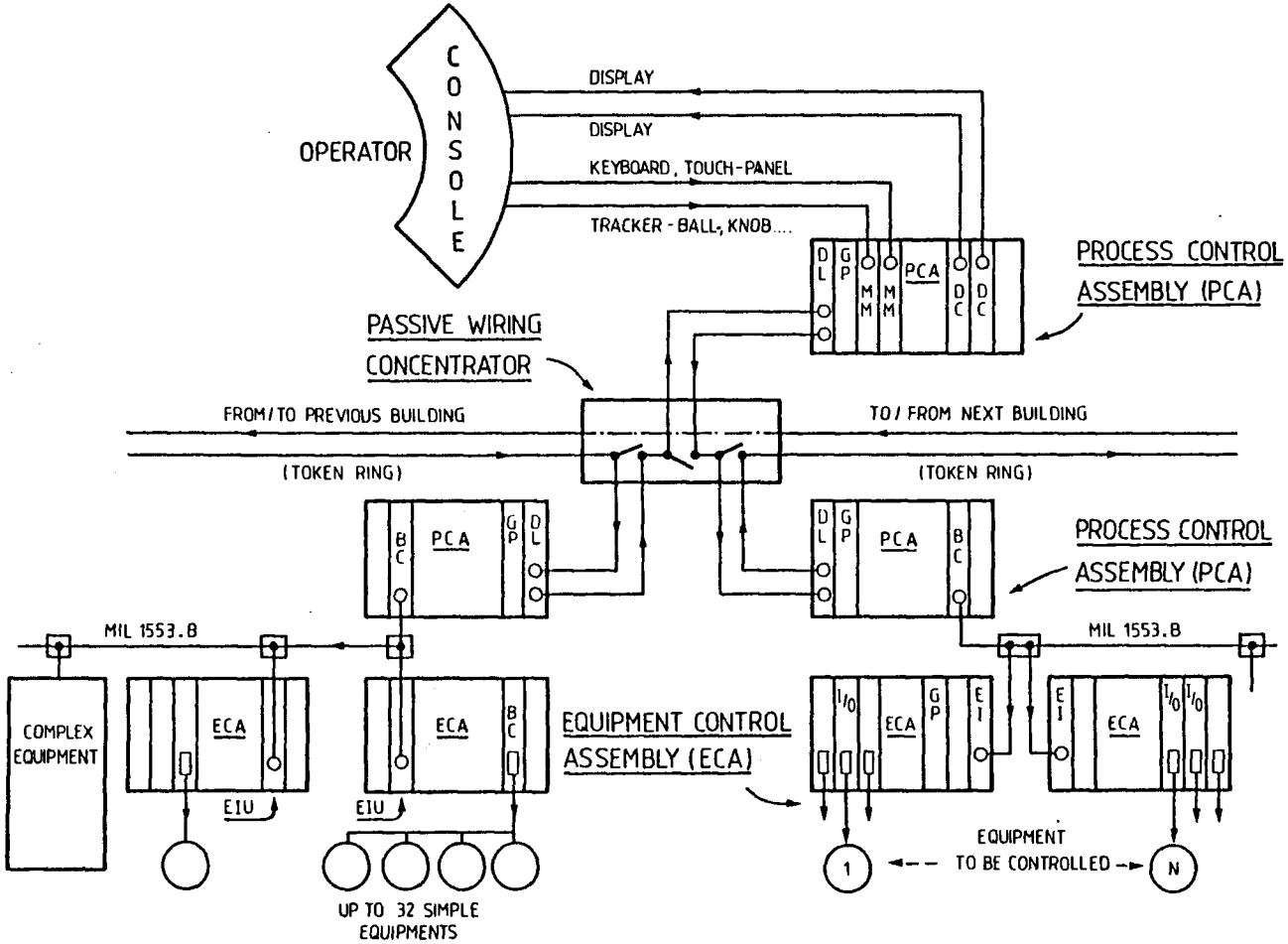


Fig. 2. The LEP architecture

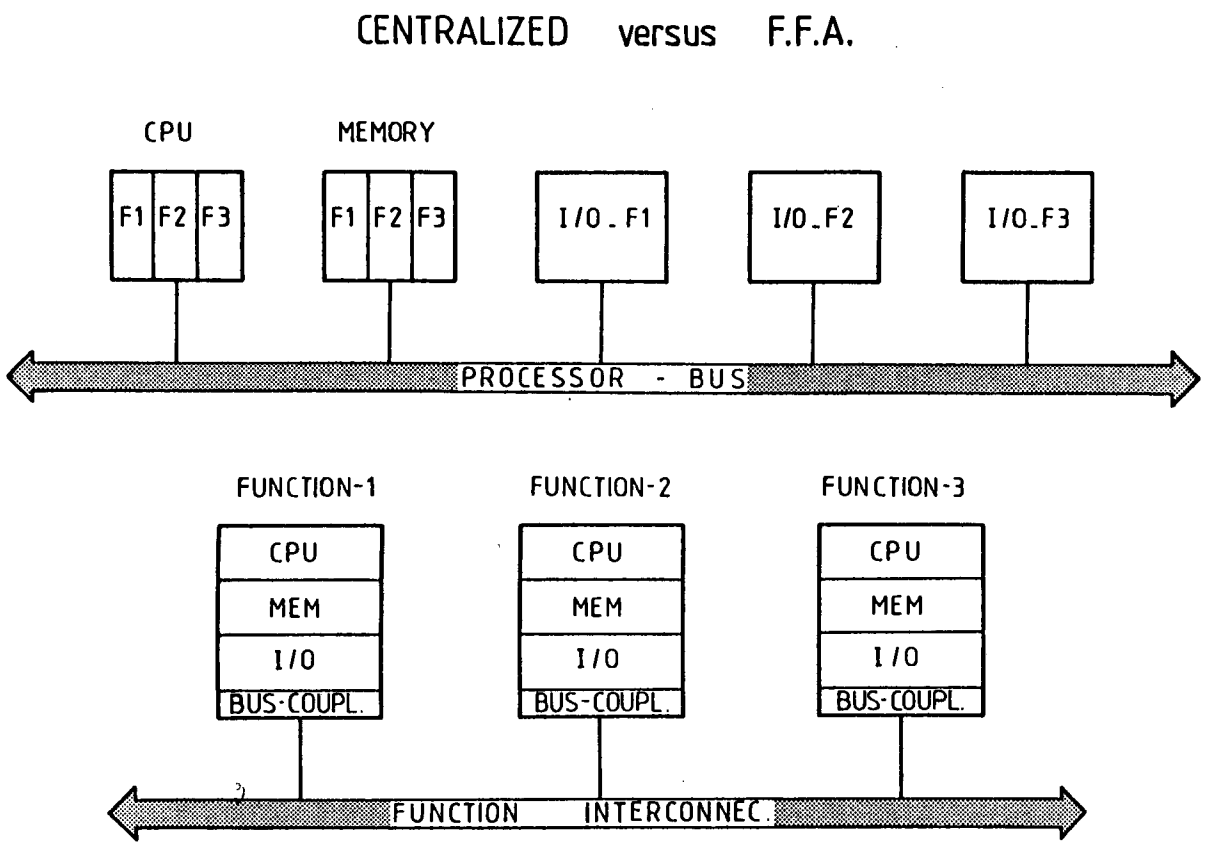
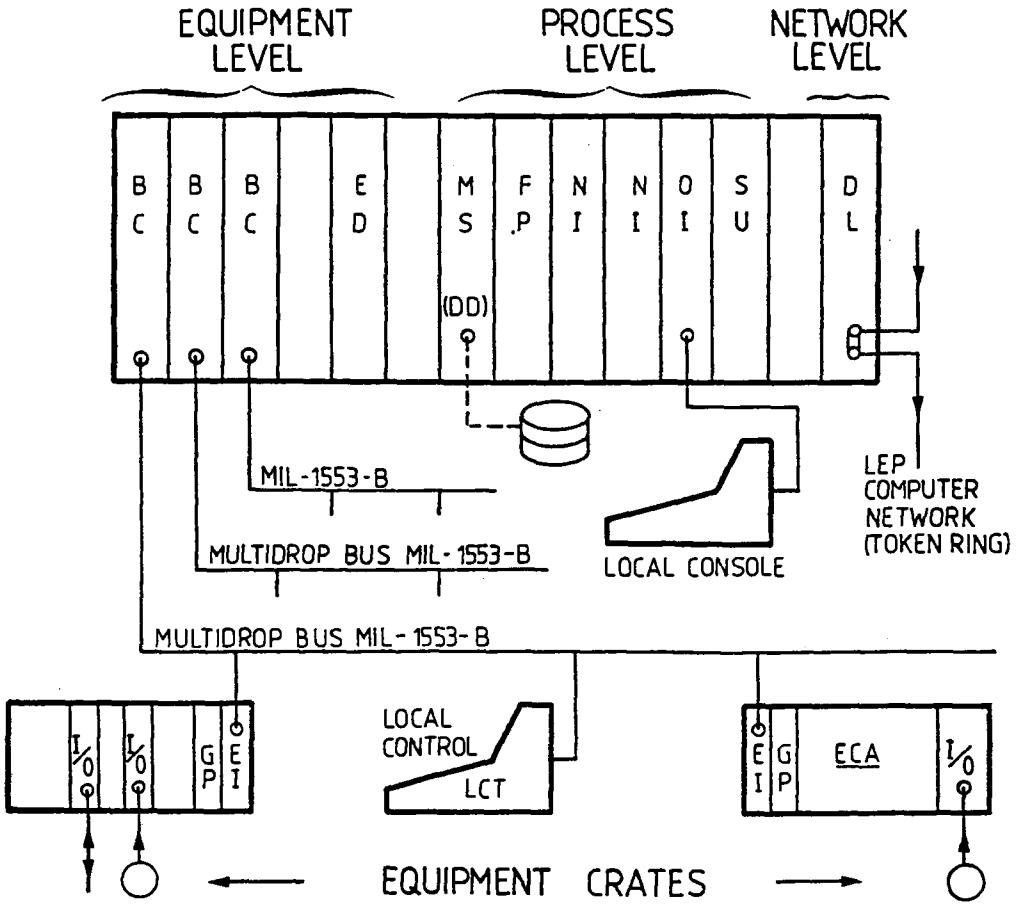


Fig. 3. The function-to-function architecture



- BC Bus Controller
- DD Disk Driver
- DL Data Link driver
- EI Equipment Interface

- GP General purpose Processor can act as:
  - SU SUpervisor
  - NI NODAL Interpreter
  - MS Mass Storage (semic.)
  - OI Operator Interface
  - ED Equipment Directory
  - FP FORTRAN Processor
  - GP General Processor

Fig. 4. Process Control Assembly

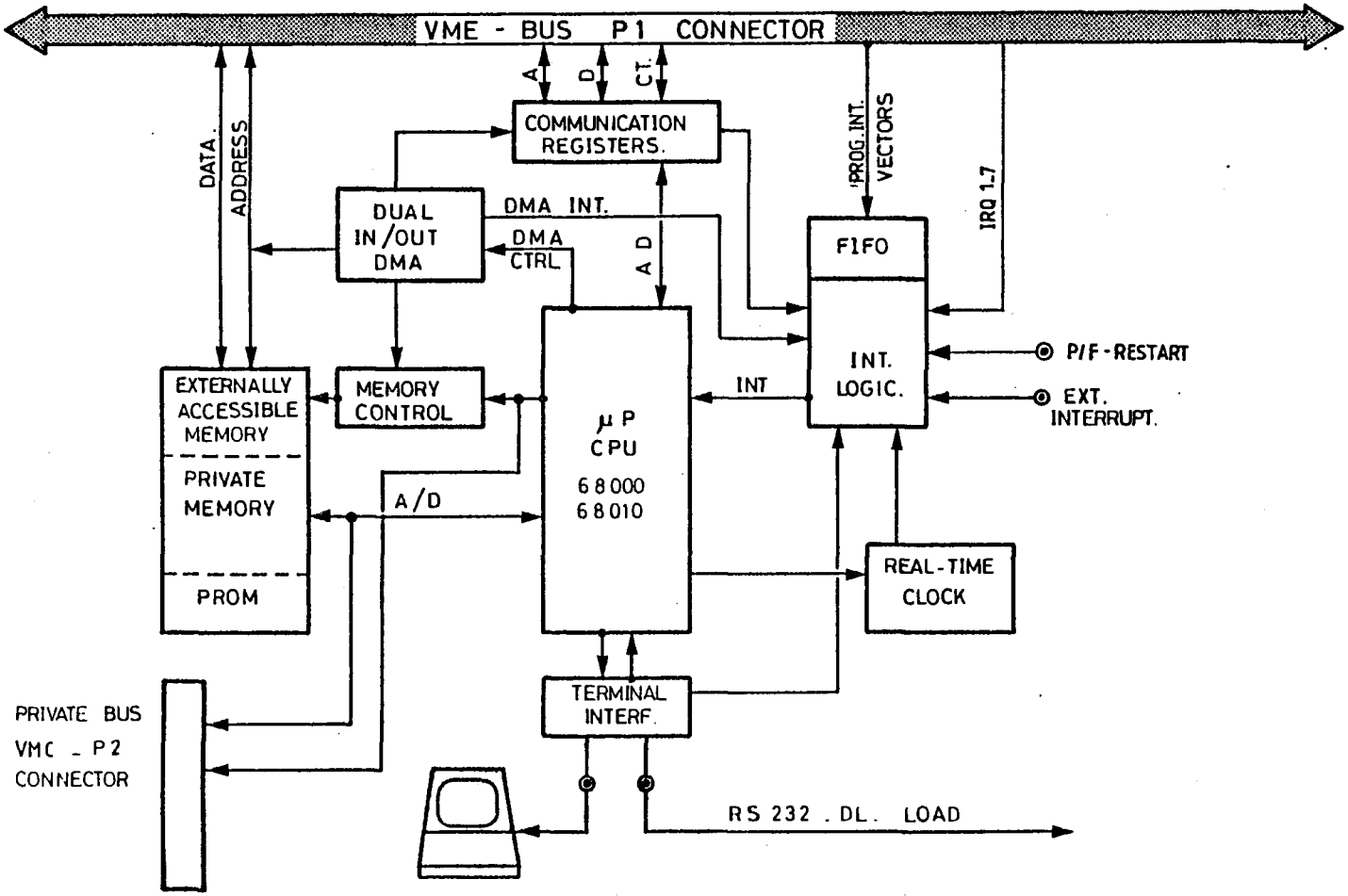


Fig. 5. General Purpose Processor Unit

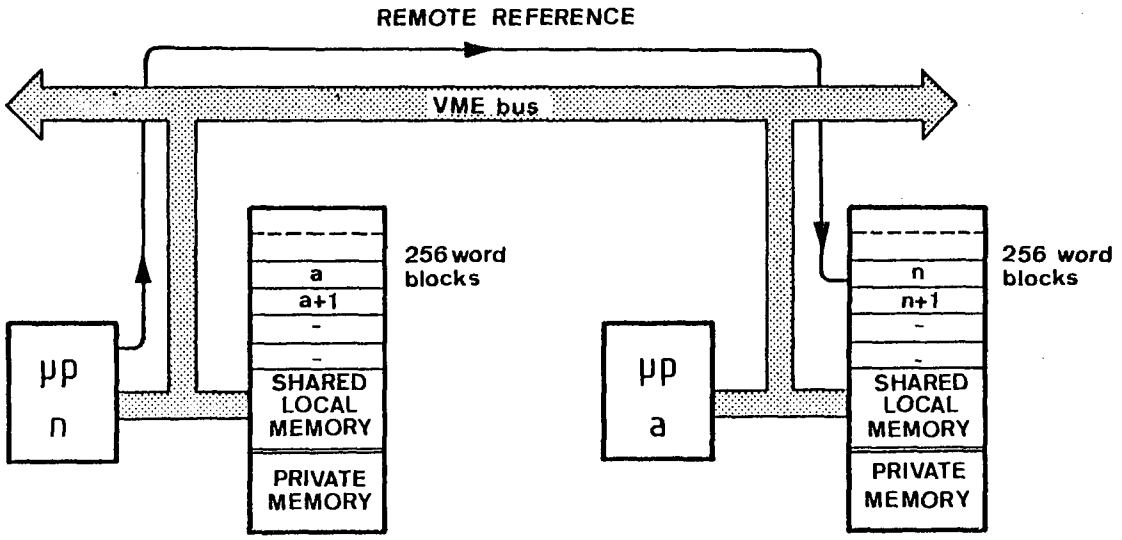


Fig. 6. Remote Reference Memory

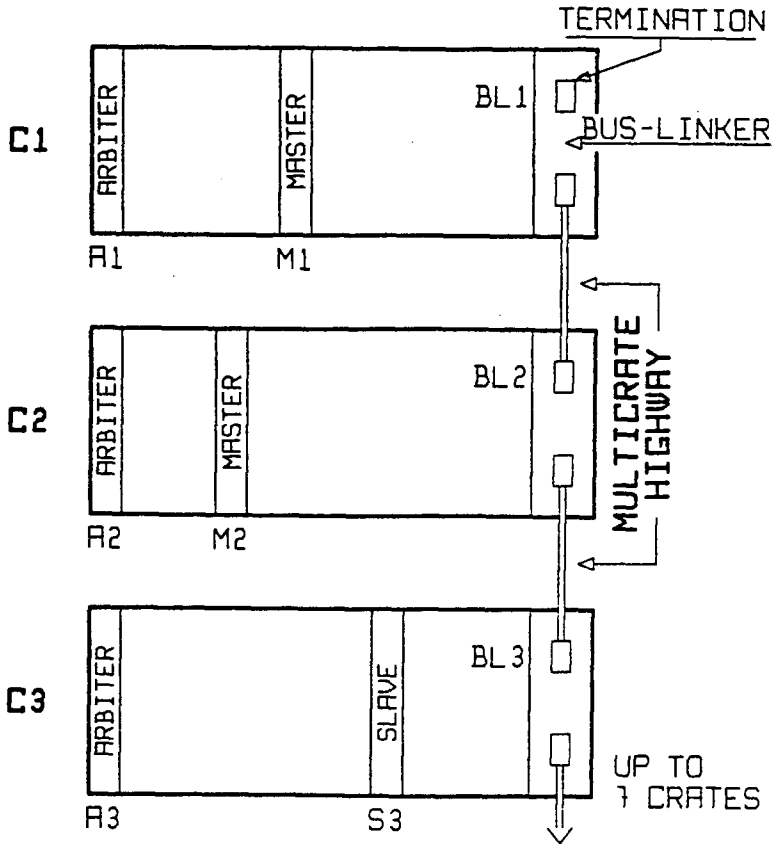


Fig. 7. Multicrate-multimaster highway

DRM SYSTEM
A DISTRIBUTED REALTIME MULTIPROCESSOR SYSTEM

Hugh Maaskant

Nederlandse Philips Bedrijven B.V.
I&E, Industrial Data Processing
Eindhoven, The Netherlands

Based on extensive research efforts Philips has developed an integrated software package for distributed, real-time applications, called DRM System. The DRM System comprises an application development environment and a distributed target environment.

The central concept of the DRM System is called SOMA (for software machine), which allows the construction of hardware independent, distributed, application programs. These programs consist of a number of SOMAs which communicate via message passing.

INTRODUCTION

Because of the growing interest in distributed computer architectures the Philips Research Labs began, in the early 1970's, a research project on the hardware and software aspects of distributed systems. As one of the results of this project a prototype system, called PHIDIAS, was built. This system has been evaluated in a robot control research project and is still in use today.

In 1983 the Philips divisions Elcoma (Electronic Components and Materials) and S&I (Scientific and Industrial Equipment), in close cooperation with the Research Labs, started work on a distributed computing system based on the PHIDIAS research effort and experiences with the robot application. The resulting product is the DRM System, which stands for Distributed Real-time Multi-processor System; it is now commercially available.

The remainder of this paper first briefly discusses the product DRM System, then it introduces the concepts used within the DRM System. The following sections give an overview of the application development tools and the top level design of the operating system. The last section reports on the current status of the product and the development plans for the near future.

THE PRODUCT

The DRM System is a software package consisting of:
- A configurable operating system for loosely coupled distributed computing systems.
- A set of tools to develop application programs to be run under the control of the operating system.
- A set of standard application programs.

Within the DRM System we recognise two different environments: the development environment and the target environment. The development environment is a conventional, interactive computer system on which the usual tools like editors and a number of DRM System specific tools can be used to develop application

programs. The target environment consists of the actual distributed computer system under control of the DRM System's real-time operating system. The development and target environments may be connected through an appropriate link to allow files in the development environment to be accessed from the target environment.

CONCEPTS

One of the PHIDIAS research goals was to develop a methodology that would allow programmers to construct parallel and distributed programs to a large extent independent of the actual hardware configuration. The resulting concept is called a SOMA, which is an acronym for software machine. The SOMA concept is also the basic DRM System concept. Other concepts are HAMA, configuration, application and job.

SOMA

A SOMA is an entity consisting of one or more (typically 1) sequential processes and zero or more mailboxes (see fig. 1). The SOMA's processor cannot directly access data belonging to another SOMA but they may communicate by sending messages to and receiving them from mailboxes. A mailbox is a typed data structure belonging to a particular SOMA. It consists of a fixed, non-zero, number of slots where each slot can contain exactly one message of the same type as the mailbox.

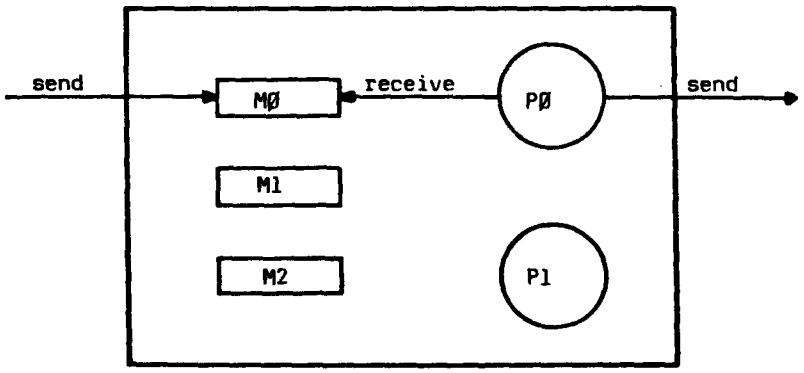


fig. 1, graphical model of a SOMA

To support communication an asymmetric pair of primitives is available called send and receive. With the receive primitive a process obtains a message from the specified mailbox (which must belong to the same SOMA). If, at the time of invocation of the receive primitive, no message is available in the mialbox the invoking process will be suspended until the arrival of a message. The send primitive is never suspended.

The send operation is transparent to the location of the destination mailbox, i.e. the sending process does not need to be aware of the HAMA on which the mailbox has been loaded.

HAMA

HAMA is an acronym for hardware machine; a HAMA consists of one or more processors with a common memory. A HAMA's processor cannot directly access another HAMA's memory but HAMAs may communicate through communication links. HAMAs may have I/O ports.

Configuration

A collection of one or more HAMAs connected through communication links is called a configuration. The DRM System imposes no requirements on the interconnection structure, so a meshed network may be used.

Application

An application is a collection of one or more SOMAs that are designed to be controlled as a unit.

Job

When an application is loaded on a configuration it is called a job; a job is the dynamic image of the application template. When loading a job the SOMA is the unit of distribution. One SOMA cannot be distributed over HAMAs.

APPLICATION DEVELOPMENT TOOLS

The major development tools are the Application Builder, the Distribution Handler and the Operating System Builder. They all run on the development environment while the Distribution Handler also runs on the target environment.

Application Builder

The Application Builder takes as input a text file containing a description of the application. This file contains among other items the declarations of SOMAs with their processes and mailboxes, and it names the files comprising the sequential processes. From these files the Application Builder generates a load file which contains a coded application description and the relocatable images of each SOMA.

Distribution Handler

A distribution is the mapping of the application's SOMAs onto the HAMAs in the configuration. The Distribution Handler generates a coded set of permitted distributions from a user provided distribution description file.

The information in the load file and the distribution file together is sufficient to start an application on a configuration (i.e. a job).

Operating System Builder

The Operating System Builder is analogous to the Application Builder but instead of generating an application load file it generates an operating system load file. As input it requires a file containing a partial description of the configuration and a file containing information specific to the HAMA.

OPERATING SYSTEM DESIGN

Each HAMA contains a specific copy of the operating system, therefore HAMAs can operate autonomously and they can be added to or removed from a configuration dynamically. After booting each HAMA will attempt to establish a connection with all other HAMAs.

The Operating System (OS) has been divided into two parts: the local OS and the global OS. The local OS consists of a conventional kernel that implements the SOMA concept and provides preemptive, priority based process scheduling, interrupt handling and timer functions. The global OS consists of three functional elements: the Network System, the Job Control System and the File Management System. Each of these elements has knowledge of and can cooperate with the corresponding element on the other HAMAs. Unlike distributed systems based on existing operating systems, the DRM System has its communication layer at the bottom of the global OS so that the Job Control System and File Management System can make use of the Network System services, allowing for an efficient operation.

Network System

The Network System provides reliable inter HAMA message transport which is implemented through packet switching. Packets are dynamically routed using a least cost algorithm based on link costs. Another function of the Network System is to detect disconnections.

Job Control System

The Job Control System controls the starting and stopping of jobs. To start a job a load file and a distribution file must be specified. The Job Control System examines each distribution in turn until a valid one (i.e. one for which all resources are available) has been found. Then the job is loaded according to that distribution.

A job is called severed if, due to a disconnection, any of its SOMAs cannot be reached by any of its other SOMAs. Severed jobs are automatically stopped by the Job Control System. This means that, within a job, no unreachability handling or resynchronisation is required.

File Management System

The File Management System provides a uniform view of files and devices, which is compatible with UNIX (*). Files can be accessed independently of the location (HAMA) of the devices on which the files reside. In addition the File Management System provides access to the files belonging to the development environment.

STATUS AND NEW DEVELOPMENTS

Release 1.Ø of the DRM System is now available. It implements a HAMA on a series of VMEbus/68000 CPUs with up to ½ Mbyte of local memory. The links supported are VMEbus and V11/V24. The supported development environment is VAX/VMS (**).

In the near future UNIX will be supported as a development environment. Furthermore Ethernet (***) communication will be supported.

(*) UNIX is a registred trademark of AT&T Technologies

(**) VAX/VMS is a registred trademark of Digital Equipment Corporation

(***) Ethernet is a registred trademark of Xerox Corporation

BUFFERED PIPE PROTOCOL

J. Werdehausen

Technical Marketing Manager
Motorola Microsystems Europe
Taunusstrasse, 51
8000 Munich 40
Federal Republic of Germany

Abstract

A standard is presented for a processor to processor communications mechanism. The procedure is applicable to any two generic processors (tasks) which can access a common address space, though it is primarily intended for use in a CPU - IPC environment. Conceptually, information is passed as a fixed length message in an envelope through a pipe shared by two processors. The processors must supply the envelope when sending a message, and are given an envelope when receiving a message. The message is typically a pointer to a data structure, but may be any application defined data.

1. INTRODUCTION

1.1 Overview

This document is but one of several describing various aspects of the CPU-IPC interface. A mechanism is described here which allows messages to be passed from one processor to another with a minimal set of restrictions. Other documents will treat such issues as the content of the messages.

1.2 Definitions

Busy Interface	In this context, an interface between two boards which may require one of the two processing elements to wait idly (in a software loop) for the right to access some resource.
CPU	Central Processing Unit; this document uses the acronym to refer to a microprocessor-based board-level product whose function is to ru application software and, in so doing, direct peripheral controllers to control devices.

IPC — Intelligent Peripheral Controller; a microprocessor-based board-le product whose function is to control peripheral devices under the direction of one or more CPUs.

1.3 Assumptions

For all practical purposes, the IPC must be microprocessed-based. The interface proposed here involves queue manipulations algorithms, currently executable only in software.

The use of this interface assumes the existence of a relatively large body of RAM (typically many Kbytes) which is accessible to both the processors being interfaced. Ordinarily this will be what is usually called "system RAM", which is RAM accessible by any bus master on the system bus.

2. REQUIREMENTS

2.1 Features

- *Range of use:* the interface is intended to be (within reason) operating system- and board-independent. Both UNIX and VERSAdos can make fine use of the interface, and it will serve for virtually any IPC and CPU.
- *Supports both interrupt-driven and polled modes of operation:*
- *Low hardware requirements:* essentially all that is required is that both boards be able to access a nontrivial region of shared RAM. For interrupt-driven operation, each processor must have the ability to interrupt the other.
- *"Nonbusy" handling:* when one processor accesses the interface, the other processor is not prevented from accessing the interface at the same time.
- *"Infinitely long" queue of messages:* there's always rooms for another message in the interface, without waiting.
- *Low, constant overhead:* message sending and receiving is always accomplished by the same small set of instructions. There are no "worst cases" to worry about which consume substantially more time than the "normal" case.
- *Ease of use:* there are only a few simple procedure calls in the package, and their operation is straightforward. Only one data structure - a relatively simple one - is defined by the protocol.
- *Completely symmetrical interface:* the same mechanism can be used for virtually all communication between the two processors.
- *Complete flexibility in message definition:* this protocol in no way restricts the content of the information sent.
- *Order preservation:* Even though messages are being exchanged between two asynchronous processors, each processor will receive messages in the order in which they were sent to it. Equally important, though, is the fact that the receiving processor is in no way constrained to process the messages in that same order.

2.2 Functional Description

Conceptually, we would like the interface between a CPU and a device on an IPC to consist of a set of pipes: some for the host to send commands to the IPC, and others for the IPC to send status back to the host.

The pipes should be able to hold an arbitrarily large number of messages, so that the case never arises in which an urgent message cannot be delivered. In order to prevent arbitrarily long delays in which a driver waits for the right to access data structures, the pipes should be constructed in such a

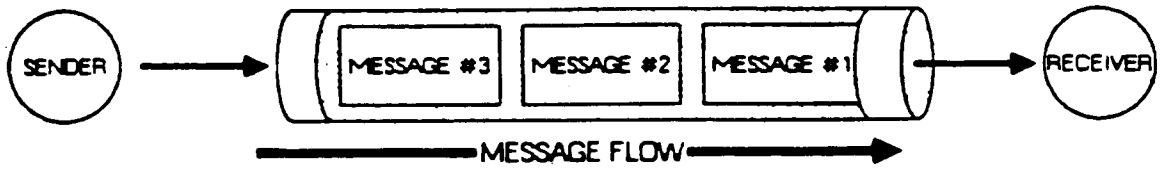


Figure 1. The concept of a pipe.

way that they are always available to both the CPU and the IPC; that is, software trying to access a pipe (to put a message in or take a message out) must never be turned away because the pipe is "busy".

Implementing the pipe as a simple queue, in which entries are connected as a forward linked list, would satisfy the requirement that the pipe have an "unlimited" capacity for messages. However, without special assistance from hardware it is difficult for two processors to simultaneously access a queue (one enqueueing an entry while the other is dequeueing an entry) in a reliable manner.

The problem centers around the fact that the routines must behave differently depending on whether or not the queue is empty. A processor must observe whether or not the queue is empty and then take an action appropriate to that case. But since the observation and action cannot be performed as an indivisible operation (without special hardware), the observation may be invalidated by the other processor by the time the action is taken. Hence use of a simple queue would require that both processors treat the instructions which access the queue as a critical section and somehow ensure that the two processors are never in their critical sections at the same time. Typically this is done by having a semaphore represent the right to enter the critical region, and having each processor loop on a TAS instruction until the condition codes say "it's your turn". This "spin lock" carries with it certain liabilities, however:

- Since it is impossible to predict how many times the loop must be executed before gaining access to the resource, the containing algorithm has a non-deterministic execution time. This is undesirable in real-time systems.
- As a lockable resource, the data structure would introduce possibilities for deadlock (e.g., several processors may be consuming the total bus bandwidth trying to win the TAS, while the processor which has control of the resource can't get to the bus to release it).
- It requires that both processors have a compatible TAS-like instruction. If an IPC possesses a more modest MPU such as a 6809, or if the MPU's only access to the bus is through a DMA controller, then no such capability exists.
- It requires that the code accessing the structure be able to disable interrupts, perhaps requiring that the code run in supervisor mode. If a module were to win the TAS with interrupts enabled, servicing an interrupt at that point might cause the structure to be tied up for milliseconds.

Fortunately it is possible to construct a queue which never becomes empty, and therfore requires no observation prior to taking action. In a nutshell, the technique involves keeping a dummy entry at the end of the queue at all times. The receiver knows never to dequeue the dummy entry. The sender adds to the queue by first adding a new dummy entry and then using the old dummy entry to hold usefull information. Because the dummy entry buffers the queue from going empty, the data structure into which the messages are put is called a "buffered pipe".

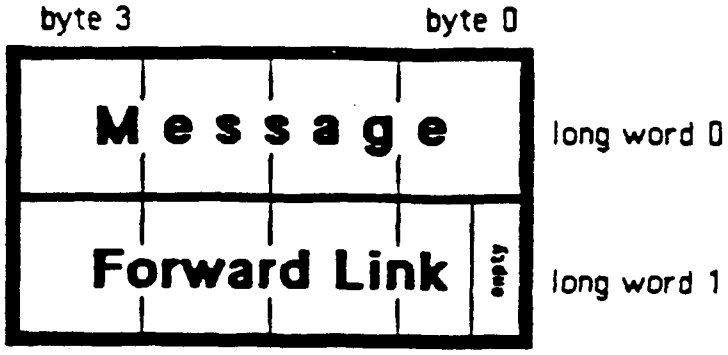


Figure 2. Buffered Pipe Envelope (BPE).

Messages cannot be queued directly on the queue. Instead, a data structure called a "buffered pipe envelope" is enqueued, and this envelope may contain a message. While the message could be of any length, for efficiency reasons they are fixed at four bytes for this interface. The envelope, called a BPE, is diagramed in figure 2.

If the envelope is full, the message field contains a valid message. If the forward link field is even, then the envelope is full (it may be dequeued) and the forward link is the address of the next BPE in the queue. However, if the forward link field is odd, then the envelope is empty (i.e. it is a dummy entry) and should not be dequeued; this entry marks the end of the pipe. Note that the null address convention, as commonly used in the C language, is not employed here. Access to an address value can not be guaranteed to be an indivisible operation, but access to the least significant bit of an address is guaranteed to be indivisible. Since it is very possible that both the sender and receiver processors may be accessing the same forward link field simultaneously, it is essential that the consistency of the data structure be maintained even as the sender is updating the forward link and the receiver is testing it.

Note: BPE's are required to begin on an even address.

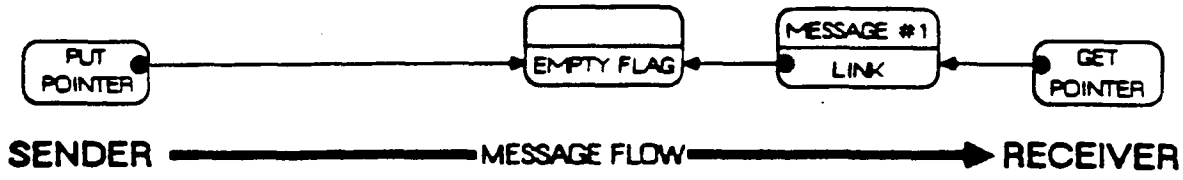


Figure 3. Implementation of a buffered pipe.

It is important to keep the concept seen by the user, the buffered pipe, distinct from its physical implementation as queue. What the user of the interface sees is a virtual data structure called a buffered pipe into which can be put a 4-byte message; the pipe is sometimes empty. The physical implementation, as seen in figure 3, is a queue of BPE's, there is always at least one BPE on the queue, and the last BPE on the queue is always marked "empty".

A buffered pipe is initialized to have a dummy entry (an empty envelope) on its queue. The following figures and pseudocode describe the sending and receiving of messages on the pipe:

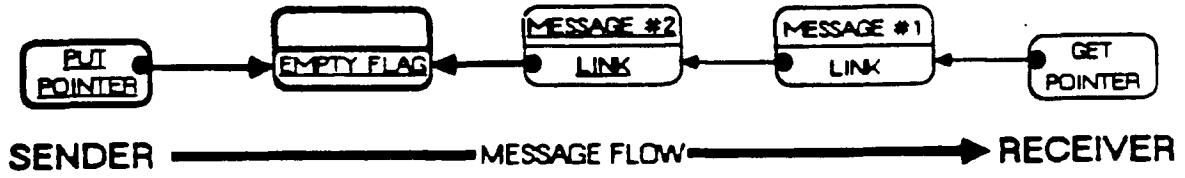


Figure 4. Putting a message in a buffered pipe

To put a message in the pipe:

Add an empty envelope to the queue.
Put the message in the preceeding envelope (which is empty) and mark it full.

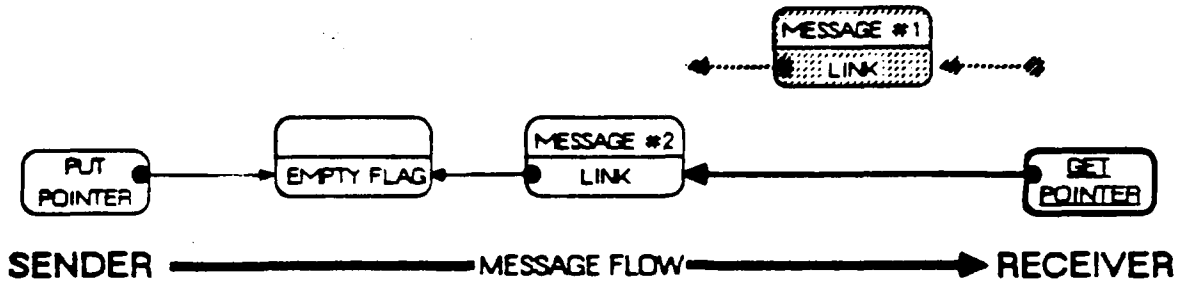


Figure 5. Getting a message from a buffered pipe.

To get a message from the pipe:

If the next envelope on the queue is empty then
    There are no messages to receive.
Else
    Dequeue the envelope and take out the message.

3. AN EXAMPLE USE OF THE PROTOCOL

While the definition of the protocol was intentionally left as general as possible so that it could be used in many different environments, it is possible to discuss a simple set of circumstances surrounding the use of the protocol and to discuss its use in that context. Those circumstances are:

- A new microprocessor-based IPC is produced, and a driver for an operating system such as VERSAdos or UNIX must be written to support it. The IPC controls eight independent devices (e.g. serial ports).
- Two buffered pipes are used: one for the CPU to send commands to the IPC, and the other for the IPC to send status back to the host.
- The interface between CPU and IPC is interrupt-driven.
- Messages are actually just pointers to memory-resident data structures, which we will call "packets".
- Packets are fixed in length.

- A pool of packets is created by the driver at initialization time, as well as a pool of BPEs.
- Status for a command is returned in the same packet in which the corresponding command was received. This can simplify housekeeping for the driver.

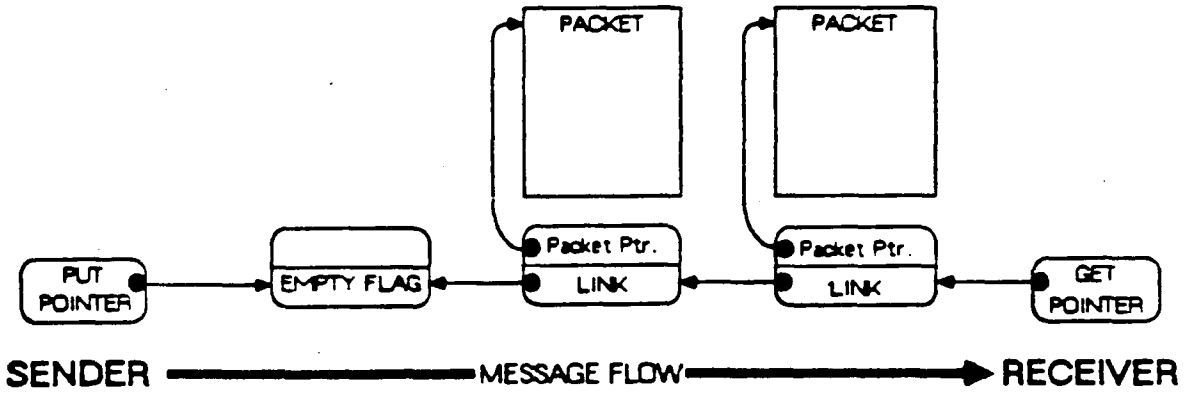


Figure 6. Example: Messages in the BPE's are packet pointers

3.1 CPU Function

The CPU has the following functionality in its driver:

In the initialization routine:

- Get memory from the OS kernel and use it to make a pool of N packets (all of which are of the same size), and a pool of N + 2 BPEs.
- Get a BPE from the BPE pool and initialize a pipe for sending commands to the IPC.
- Get a BPE from the BPE pool and initialize a pipe on which the IPC will send status to the CPU.
- Pass the address of the BPE's use to initialize the two pipes to the IPC, using some mechanism (other than the pipes) appropriate to the particular IPC.

In the user request handler(s):

- Get a packet from the packet pool.
- Get a BPE from the BPE pool.
- Fill in the packet according to the user request.
- Put pointers to the task and its request block in the packet, or someplace associated with it.
- Send the address of the packet as a message to the IPC on the command pipe.
- Interrupt the IPC.

In the interrupt service routine, or a routine activated by it:

- Try to get a message from the status pipe.
- While there is a message to get,

    Return the obtained BPE to the BPE pool.

    Using the message as a pointer to the packet,
    extract the status from the packet, as
    well as pointers to the task that made the request
    originally and to the block representing the request.

    Use the info to signal I/O completion to somebody.

Try to get another message from the status pipe.

3.2 IPC Function

The IPC contains the following functionality:

In the initialization routine:

- Receive pointers to the command and status queues via some mechanism appropriate to the IPC.
- Initialize a pool header for BPEs (the pool is empty).
- Initialize queue headers for holding incoming commands when they arrive, as appropriate to the functionality of the IPC.

In the handler for interrupts from the CPU:

- Try to get a message from the command pipe.
- While there was a message to get,

    Add the BPE to the IPC's BPE pool.

    Using the message as a pointer to the packet,
    look at the command packet.

    Determine the packet's priority and do something
    appropriate with it, like putting
    it in one of the IPC's command queues.

    Try to get another message from the command pipe.

In the logic that recognizes a command has been completed:

- Put status in the packet that the command came over in.
- Get a BPE from the IPC's BPE pool.
- Put a pointer to the packet (as a message to the CPU) into the status pipe.
- Interrupt the CPU.

PERSONAL COMPUTER ACCESS TO THE VME BUS

B.G. Taylor

CERN, 1211 Geneva 23, Switzerland

Abstract

The marriage of a mass-produced personal computer with the versatile VMEbus environment creates a cost-effective solution to many laboratory instrumentation requirements. This paper describes a novel means of providing direct memory-mapped access from a 68000-based personal computer to single or multiple VMEbus and CAMAC crates which are interconnected by a ribbon cable bus. The system is called MacVEE (Microcomputer Applied to the Control of VME Electronic Equipment).

In an implementation for the Apple Macintosh, the bus is driven by an electronics plinth called MacPlinth, which attaches to Macintosh and becomes an integral part of the computer. The total external address space accessed via MacPlinth is over 100 Mbytes, in up to 8 VME crates, or up to 7 VME crates and up to 8 CAMAC crates, in any mix. MacVEE applications can be programmed in any of the Macintosh resident languages.

Introduction

In recent years, the growth in the demand for personal computers has justified investment in highly automated facilities for their manufacture, and the most popular machines are now produced on assembly lines with capacities exceeding 1000 computers per day. The application of consumer electronics mass production techniques makes such personal computers available at prices which compare very favourably with those of microcomputer systems produced in small quantities for professional instrumentation applications.

The large market for low-cost personal computers stimulates the creation of a wide range of compatible hardware and software products by third-party vendors. Numerous software developers in educational or research environments contribute public domain programs and utilities for the popular personal computers, whilst the large user base is a factor in thorough debugging and influences positively the quality of the documentation available.

The M68000-based Apple Macintosh, the architecture of which is indicated in Fig. 1, has enjoyed considerable popularity since its introduction in January 1984 and over 500,000

machines are now estimated to be in service. Some 25 software houses have generated a range of resident programming languages for the Macintosh which includes 68000 Assembler, Basic, BCPL, C, Cobol, Forth, Fortran-77, Lisp, Logo, Modula-2, Neon, Pascal, Prolog, Scheme, Simula-67 and Smalltalk-80.

The language support, compact size, fast high-resolution graphics and friendly Smalltalk-inspired user interface of the Macintosh are attractive features for an instrumentation-development and experiment-monitoring microcomputer, but the Macintosh is only provided with 2 serial ports for external device connections. Furthermore, there are already several candidates for these ports, such as Imagewriter or Laserwriter printers, digitizers, Winchester disk units, modems, or the Appletalk local area network.

To provide a high performance direct link between the Macintosh and VMEbus and CAMAC crates, a memory-mapped system called MacVEE (Microcomputer Applied to the Control of VME Electronic Equipment) has been developed. In a MacVEE system, selected VME or CAMAC crates simply appear within the address space of the Mac's M68000 microprocessor, so that no special software drivers are required to access them. There is no address translation, so that MacVEE is not limited to the execution of position-independent code, and VME-resident software has direct access to all Macintosh resources and the powerful toolbox ROM.

MacPlinth

The VME or CAMAC crates in a MacVEE system are interconnected by a ribbon cable bus, which is driven by an electronics plinth called MacPlinth (See Fig. 2). The total external address space accessed via MacPlinth is over 100 Mbytes, in up to 8 VME crates, or up to 7 VME crates and up to 8 CAMAC crates, in any mix. In a VME system, Mac can execute programs in VME RAM or EPROM, and programs resident in one crate can access facilities in any of the others.

The MacPlinth pc board mounts below the Macintosh cabinet, from which it is thermally isolated by a heat shield. It is powered by a miniature high-efficiency primary-switching power supply, which dissipates very little heat, and has a fringing magnetic field below that which would cause any detectable disturbance of the CRT display.

Two short ribbon cables interconnect MacPlinth with the Mac logic board, on which a number of pc traces are cut to permit signal merging and masking. The plinth directly controls the enabling of the pair of Macintosh toolbox ROMs, which otherwise drive the microprocessor data bus when no other internal devices are accessed. MacPlinth can accommodate 32 - 128 Kbytes of local EPROM for permanent library enhancements.

Adequate crosstalk margins are achieved with a limited total number of conductors by appropriate signal grouping, and the provision of ground isolation only for selected strobe lines. In addition to microprocessor signals, line and frame timebase and video signals are transmitted to MacPlinth, for the generation of a composite video signal for external remote monitors able to accept the line scan frequency of 22.25 kHz. A high resolution monitor with a video bandwidth exceeding 16 MHz is required for a sharp display comparable with that of Mac's internal screen.

An abort switch is provided for program development use, as well as separate internal and external reset switches, so that Mac can be reset without disturbing a running VME multiprocessor system.

MacVEE Memory Mapping

In the MacVEE memory mapping scheme, the 16 Mbyte address space of the Macintosh M68000 is divided into 256 segments of 64 Kbytes each. A 4-bit descriptor for each segment is programmed in a Schottky 'emmental' PROM, which maps the segments to internal Macintosh space, MacPlinth EPROM, external system space or external common space. A typical memory map is shown in Fig. 3.

Substantial address space in Macintosh is occupied by the images created by the incomplete address decoding of internal hardware facilities, such as the SCC (Serial Communications Controller), IWM (Integrated Woz Machine disk controller) and VIA (Versatile Interface Adaptor). SCC read and write functions, for example, occupy 1 Mbyte each.

The emmental PROM allows 'holes' to be assigned to the addresses actually referenced by the system software, while the image spaces are mapped to VMEbus 'cheese'. This technique increases the external addressing range by 4.5 Mbytes per map. The emmental PROM pattern can be readily changed to adapt to other personal computers, or possible address changes in future versions of Macintosh.

A second level of mapping, which can be programmed through a control register on MacPlinth, allows the external system space of over 12.5 Mbytes to be allocated dynamically to any of up to 8 VMEbus crates. Up to 8 Mac-CC dedicated Macintosh CAMAC crate controllers, which occupy an address space of only 64 Kbytes each, can be connected in place of one of the VME crates. The control register is accessed at address $14, the unused most significant byte of the M68000 zero divide exception vector.

Common Area

Any segment or group of segments in the external system space may be defined by the emmental PROM to be common address area. References in that area are then executed in map 1, independent of the current selection in the map control register.

Macintosh software which must access resources in all VME crates may reside either in Mac internal RAM, in Plinth EPROM, or in VME crate no. 1.

Bugs

Program or compiler bugs in normal Macintosh applications could result in spurious references to MacVEE external system address space. Such bugs could escape attention because Mac's internal DTACK generator does not verify that all addresses accessed are in fact valid. However, if these accesses were mapped to the external system, they would timeout if the corresponding VMEbus address space is unallocated, or could cause an undesired action there if it is occupied.

To allow pure-Macintosh applications containing such bugs to be run on MacVEE without this inconvenience, the entire external address area is mapped to Macintosh internal space except when an external enable bit in the MacPlinth control register is set. This bit is set by a MacVEE application which requires external access, and cleared otherwise.

In the case of a spurious access to a non-implemented Macintosh address, data bus pull-ups ensure that MacPlinth returns the same (invalid) data as an unmodified Mac. Hence if such a bug does not cause a problem on an unmodified Mac, the application runs on MacVEE also.

MacVEE VMEbus Master

VME crate access in MacVEE systems is achieved via a low-cost VMEbus master module (see Fig. 4) incorporating a release-on-request (ROR) data transfer bus (DTB) requester and a 3-level interrupt handler.

This 6U-4HP module includes slot-1 functions (system clock, option-1 bus arbiter, and a global crate data-transfer timeout). It can be employed as a normal DTB master in a multi-processor system, or as a system controller. As far as possible, the design is personal-computer independent.

Mac-CC

CAMAC crates equipped with A1 or A2 controllers can be accessed by MacVEE via a VME 4600 branch driver or a SuperCAVIAR VME interface, located in one of the VME crates. Alternatively they can be accessed directly via Mac-CC dedicated crate controllers, which can be intermixed with MacVEE VMEbus master modules on the same bus from MacPlinth.

Mac-CCs also allow CAMAC crates to be accessed without an intermediate VME crate and branch driver in pure CAMAC (CAVIAR replacement) environments. In this case the CAMAC library subroutines can be accommodated in MacPlinth EPROM. The MacVEE bus uses RS485 differential transmission, and permits greater ranges than a 4600 branch.

Mac-CC is a memory-mapped dedicated crate controller (see Fig. 5) with address assignments in accordance with CERN recommendations[1]. Additional functions have been chosen to be as compatible as possible with the VME CAMAC branch driver and Type A2 crate controllers, as appropriate. All Mac-CCs are accommodated within map 8, and the addresses which they occupy remain free for VMEbus use in maps 1 - 7.

Mac-CC is equipped with an EUR 6500 auxiliary controller bus, supporting multiple controllers in a CAMAC crate with either R/G (request/grant) arbitration or ACL (auxiliary controller lockout). It is compatible with CERN standard LAM graders.

MacVEE Interrupt Handling

Macintosh internal interrupt codes are decoded on MacPlinth, merged with external interrupt sources, and re-encoded before application to the M68000.

MacPlinth assigns three interrupt levels to Macintosh internal auto-vectored (AV) interrupts, three levels to external user-vectored (UV) interrupts, and one non-maskable level to external AV interrupts. These assignments can be altered by jumpers on MacPlinth, but only if appropriate system software changes are implemented. The Mac M68000 executes in supervisor state continuously, not just during exception processing.

During a Macintosh IACK bus cycle, MacPlinth decodes the interrupt level being acknowledged from the LS bits of the address bus. If the level is one which has been assigned to external system UV interrupts, the assertion of VPA* by the Mac interrupt circuitry is inhibited, and a vector number is fetched from the VME interrupter via the MacVEE VMEbus master module.

In a multi-crate VME system, MacPlinth automatically accesses only the crate containing the source of the interrupt, independent of the map no. currently in its control register.

M68000 UV interrupts would normally be vectored through RAM locations $100 - $3FF, which are used by Macintosh system software for global variables. On MacPlinth, the 4 MS bits of the vector number are masked, and an offset is added to translate the 16 possible vector numbers to the range $30 - $3F. These vector numbers cause the M68000 to fetch its vectors from RAM addresses $0C0 - $0FF, which are not used by Mac software.

Interrupts produced by the optional interrupt actuator, which may be installed at the left side of a Macintosh cabinet, can be at levels 4, 5, 6 or 7. The MacPlinth Abort switch, which interrupts at level 7 only, is used instead and the optional interrupt actuator removed when MacPlinth is fitted.

Software

MacVEE provides direct multi-crate VMEbus and CAMAC access to any resident Macintosh language capable of reading and writing at known absolute addresses. At CERN, extensive use is made of the Macintosh M68000 resident development system (MDS) incorporating an editor, an assembler, a linker, an executive, and a program for creating resource files.

MDS includes two families of debug monitors. MacDB provides the most powerful facilities using a two-Macintosh configuration, while the Macsbug family comprises a set of one-machine debuggers using part of the internal screen, or an external terminal, for data display. The MDS allows MacVEE tests or applications in assembly language to be written, run and debugged entirely on the Macintosh.

Microsoft Basic can prove a useful tool for MacVEE systems. MBasic 2.1 is fully integrated with the Macintosh, is comprehensively documented, and has excellent facilities for editing, syntax-checking and program-tracing. It supports calls to toolbox ROM routines and allows a Macintosh environment to be created from within a Macintosh environment. Frequently-used programs can easily be implemented as Macintosh applications with customized pull-down menus, multiple windows, and dialog boxes with unlimited numbers of option selections.

For MacVEE applications programming or testing where execution speed is more important than interactive facilities, Microsoft resident Fortran (formerly Absoft MacFortran) may be used. The compiler is a full implementation of Fortran-77 with some extensions. It is rapid, and generates a Macintosh application which can interface with assembly language routines and access the toolbox.
MacFortran has been used successfully with the CERN Fastbus library and Fortran test programs, and with the ESONE standard CAMAC library for Mac-CC stored in MacPlinth EPROM.

Conclusion

The MacVEE system described provides an intimate connection from the Apple Macintosh to multiple VMEbus and CAMAC crates. It allows a low-cost personal computer with a wide range of commercially-supported resident programming languages to be used as a development tool for new instrumentation, and as a control and monitoring device in high energy physics experiments.
A more detailed description of the system is given in the MacVEE Hardware User Manual[2], while the equipment itself is now manufactured by European industry.

Acknowledgments

The author is indebted to C. Rubbia and S. Cittolin for their support of the development of MacVEE, and to A. Contin, M. Demoulin, P. Giacomelli, W. Haynes, P. Petta, J.-P. Porte, D. Samyn and other members of the UA1 collaboration, to G. Cabras, A. Marchioro and W. von Rüden of ALEPH, and to H. von der Schmitt of OPAL, for their valuable assistance.

References

1 Rimmer, E.M., CERN Implementation Recommendations for M68000-based CAMAC Port Controllers, Version 1, DD Division, CERN. (September 1983).

2 Taylor, B.G., The MacVEE Hardware User Manual, EP Division, CERN. (October 1985).

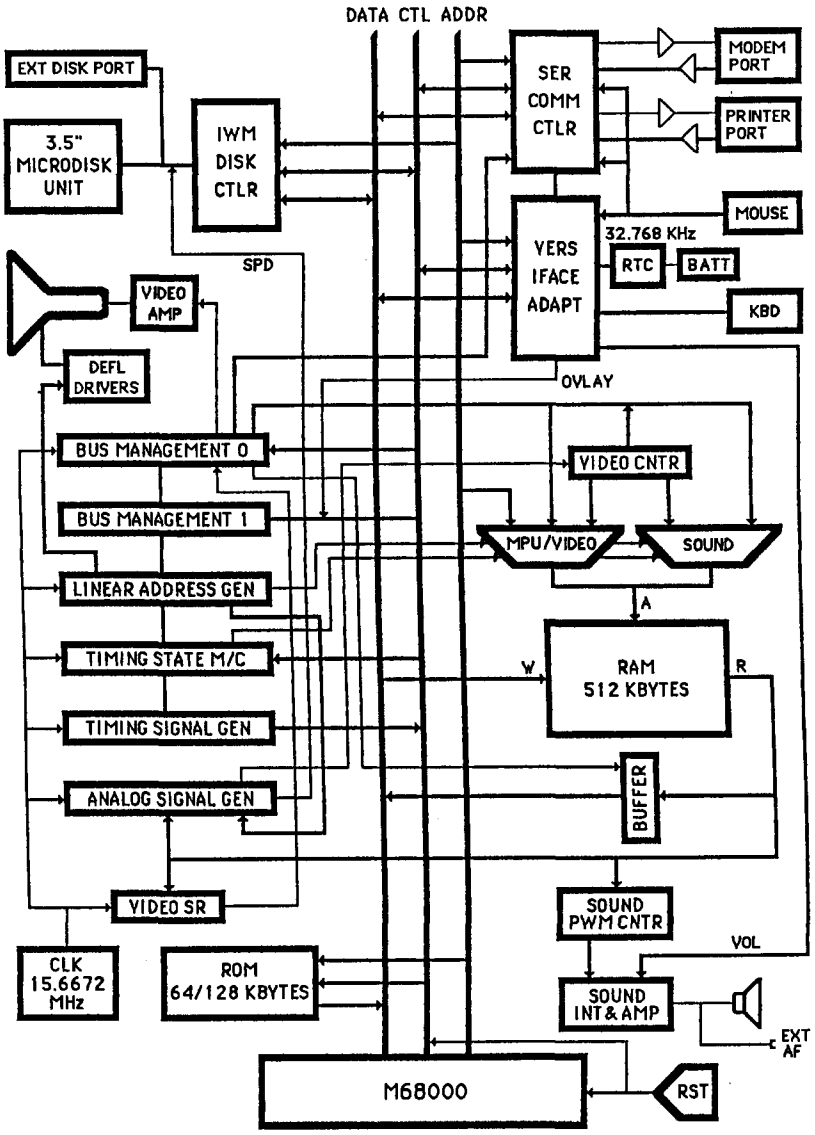


Fig. 1 Macintosh Block Diagram

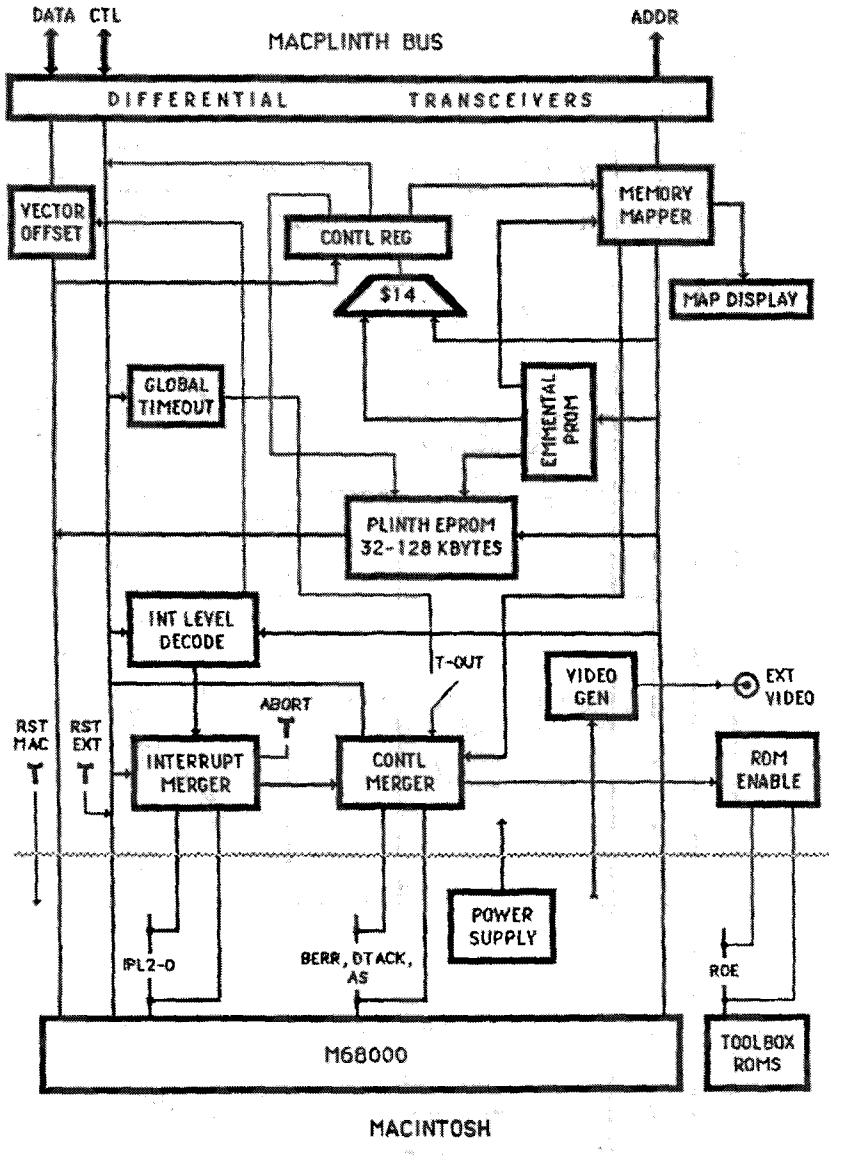


Fig. 2 MacPlinth Block Diagram

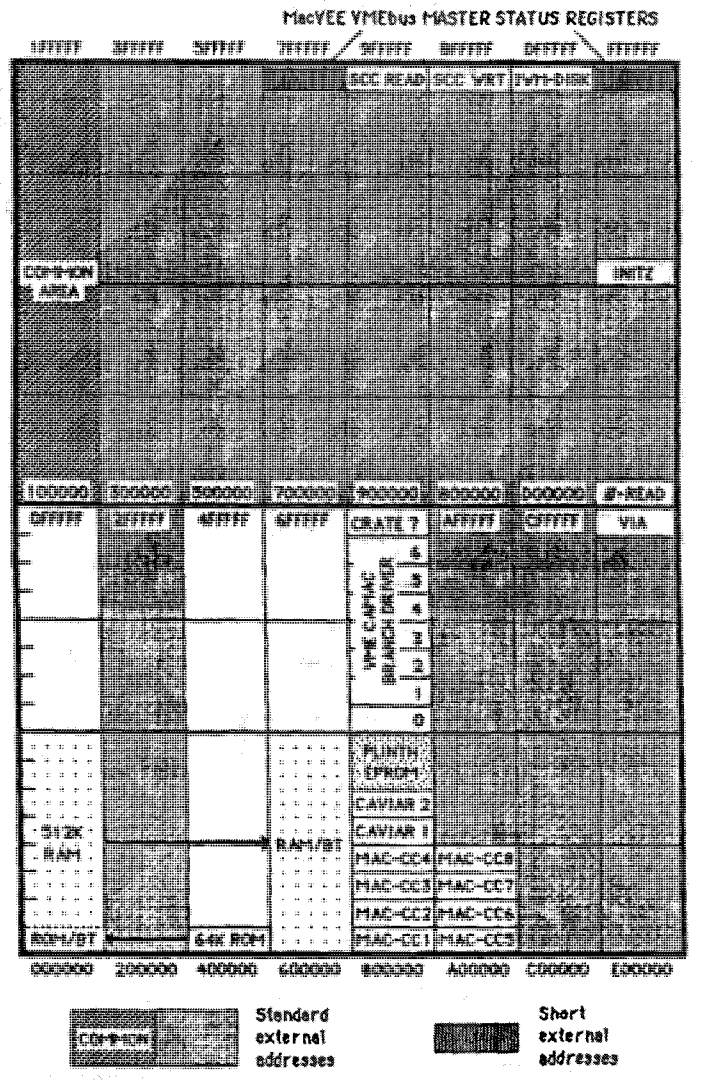


Fig. 3 MacVEE Address Map - Typical Allocations

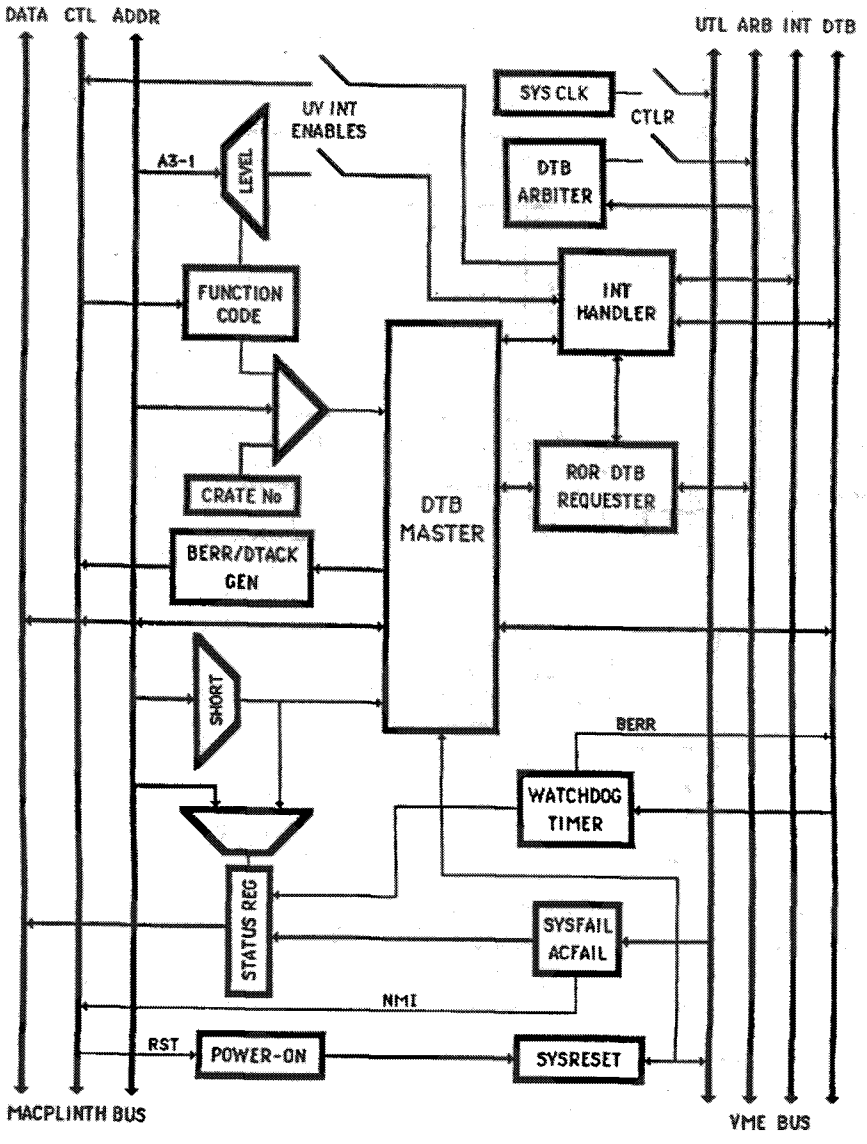


Fig. 4 MacVEE VMEbus Master Module Block Diagram

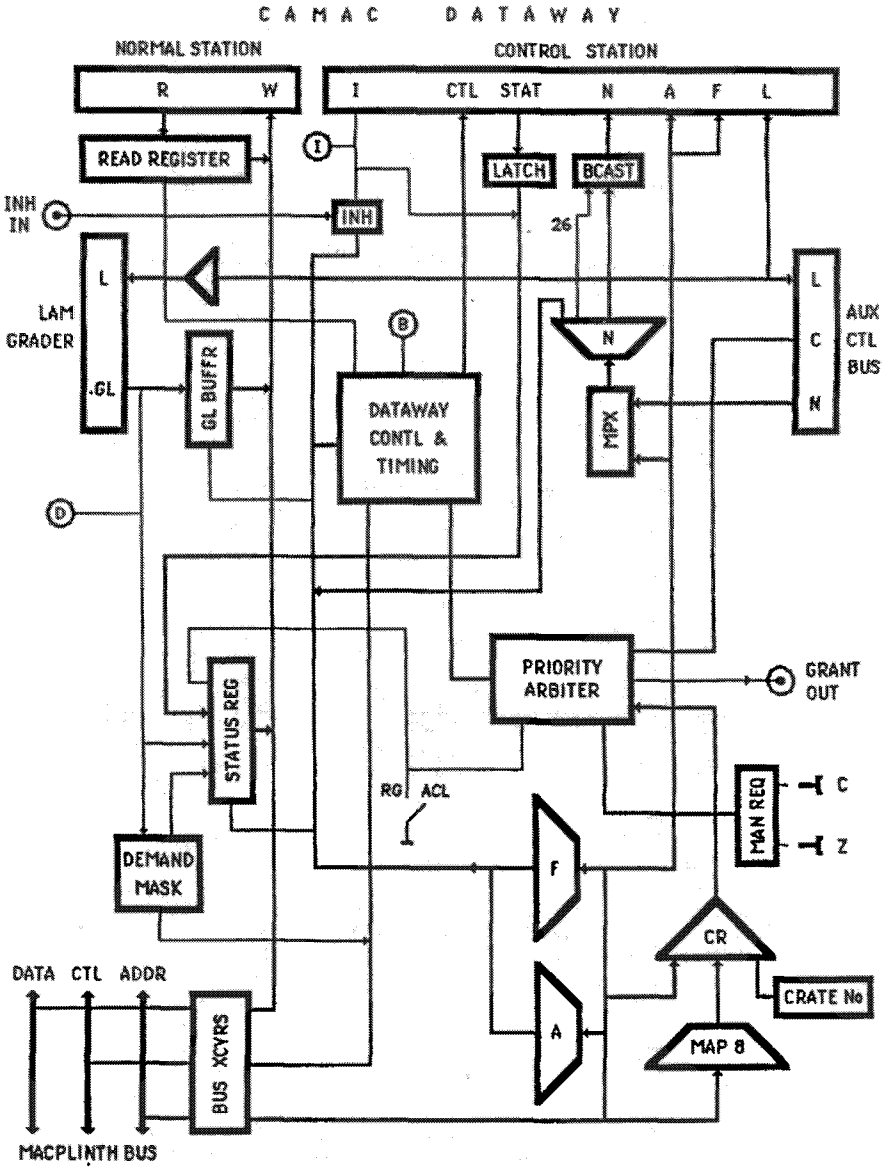


Fig. 5 Mac-CC Block Diagram

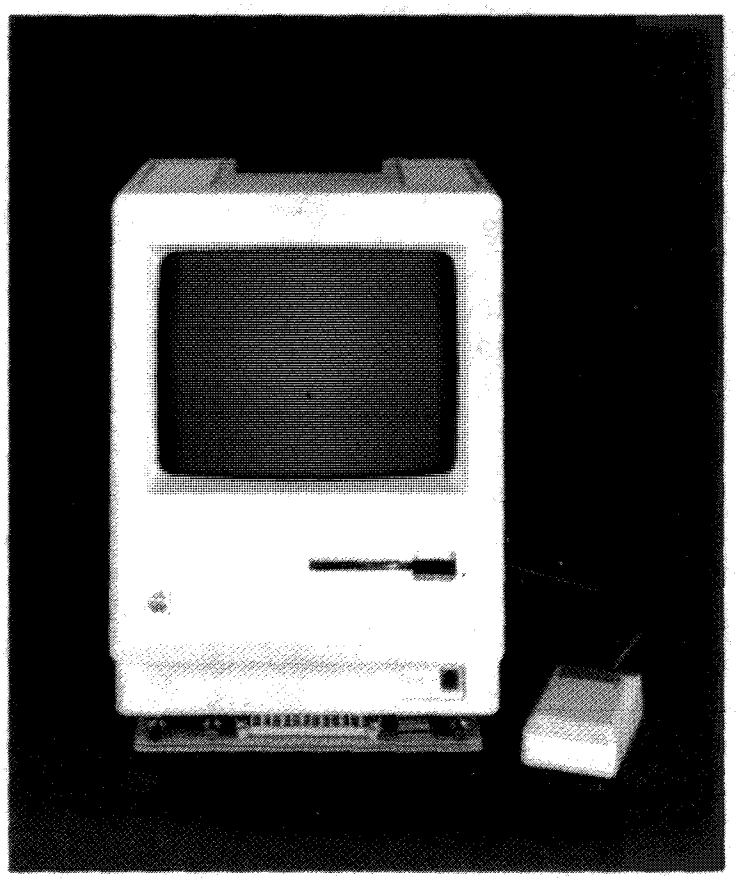


Fig. 6

Macintosh equipped with MacPlinth

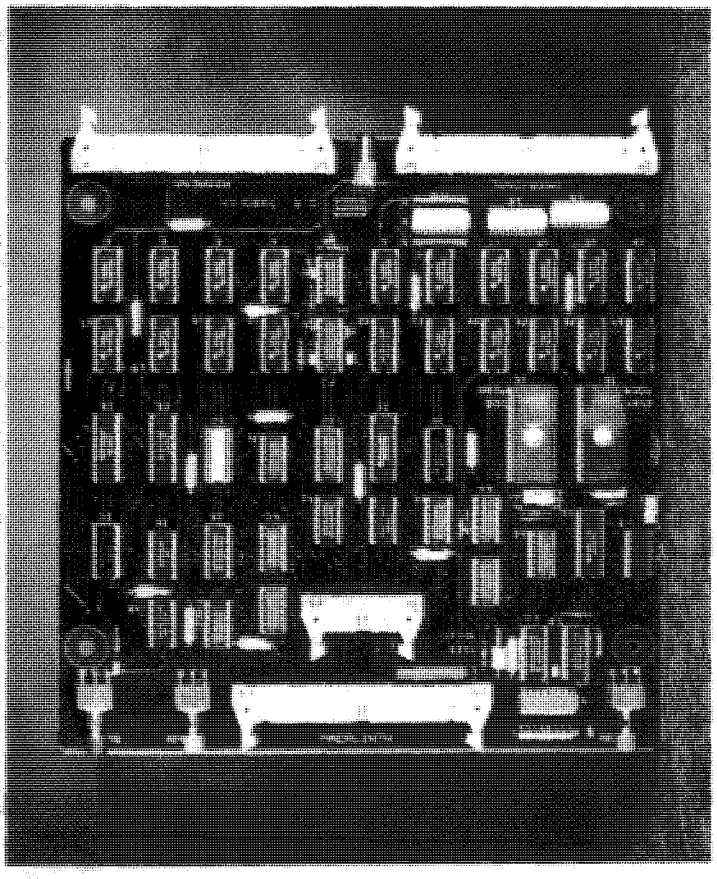


Fig. 7

MacPlinth

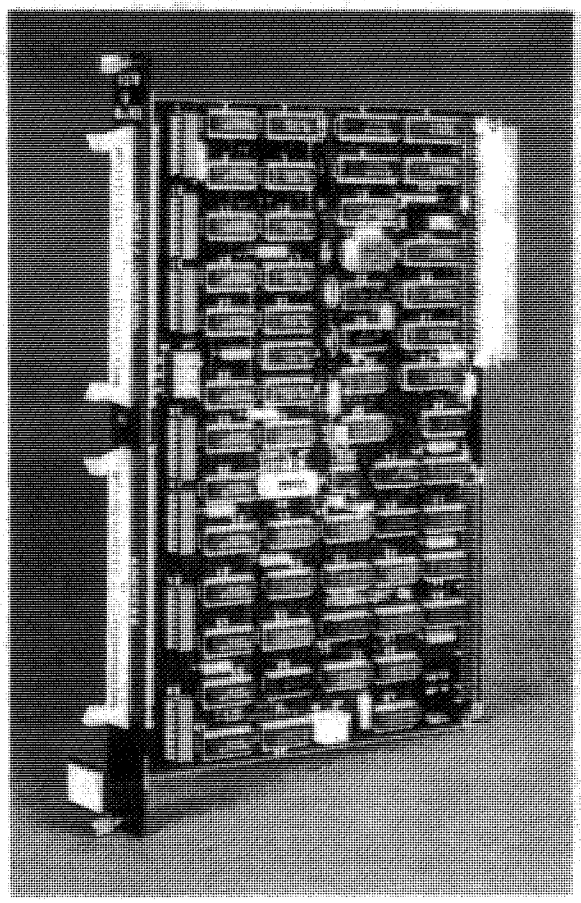


Fig. 8

MacVEE VMEbus Master Module

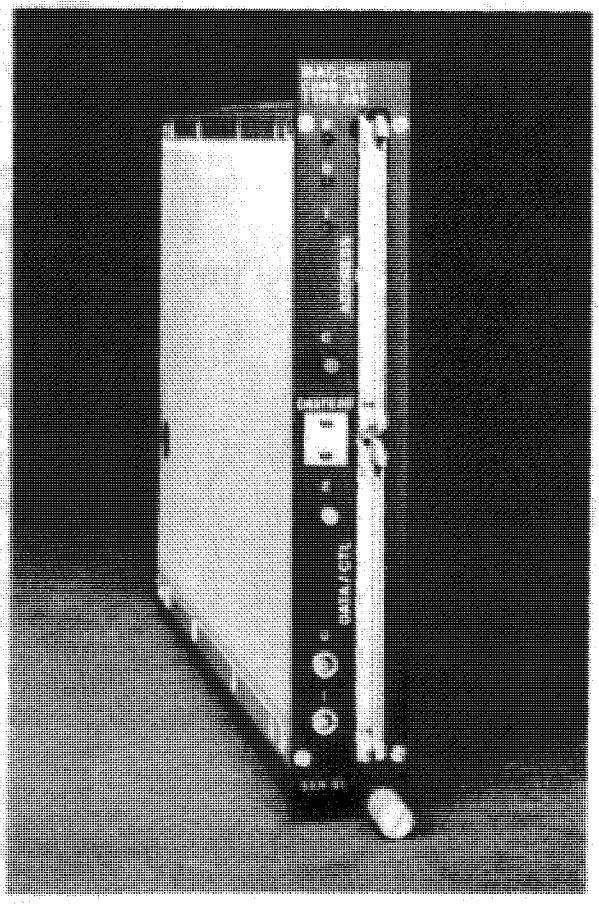


Fig. 9

Mac-CC CAMAC Crate Controller

VALET-PLUS

A VMEbus SYSTEM FOR ELECTRONIC EQUIPMENT TESTS USING YOUR FAVOURITE PERSONAL COMPUTER

C.Parkman, Y.Perrin, J.Petersen,
E.M.Rimmer, P.Scharff-Hansen, L.Tremblet
On-Line Computing Group, DD Division, CERN
CH-1211 Geneva 23
Switzerland

October 1985

Abstract

Electronic equipment testing in High Energy Physics (HEP) requires a low cost, easy to use, I/O efficient system interfaced to the HEP standard buses such as CAMAC, FASTBUS, VMEbus and GPIB. This system should replicate the test environment found on larger computers used in experiments. VALET-Plus is a system which meets these criteria. It consists of a VALET (VMEbus Applied to Laboratory Equipment Test) PLUS a personal computer.

This approach isolates the HEP-specific hardware and software in a VMEbus crate which is driven by a M68000 processor. This processor runs application programs using ROM based standard libraries and PILS (Portable Interactive Language System), and accesses the HEP buses with minimum overheads. Peripheral support for keyboard, screen, printer, and disc, and the interface to control the application processor are provided in a user transparent way, by a computer connected to the M68000 by means of a standard communications link, presently RS-232-C. The firmware running in the VMEbus M68000 is independent of the type of computer chosen. The communication software "Bridge" in the connected computer is written in PASCAL and is easily portable. This allows a variety of choices and expansion, both on the VMEbus and the computer side, without loss of previous investment.

1. INTRODUCTION

The needs of today's large and complex physics experiments together with the increasing maintenance cost of aging mini-computers obliged CERN in 1983 to review its strategy for electronic equipment and detector test facilities. The technical requirements which emerged included a small computer test system with a large address space, easy and efficient physics I/O (CAMAC, FASTBUS, VMEbus, IEEE-488), modularity, and potential portability of test software. Economic factors such as low cost, competitive price/performance ratio over a reasonably long time and manufacturing origin in CERN member states also had to be taken into account as well as a good support in CERN collaborators' home institutes.

2. THE VALET-PLUS APPROACH

Given these aims, the approach adopted for the VALET-Plus takes into account today's economic and manpower situation as well as the rapid evolution of technology. The present trend is towards a widespread ownership of personal computers with associated peripherals and commercial software packages. It makes sense to reuse this functionality by adding to it a "naked" application, in this way retaining the low cost, modularity and maintainability.

Following this approach, the VALET-Plus architecture consists of:

The VALET: An optimum environment for the application and the physics I/O

> The application software and libraries execute in a VMEbus M68000 processor which has direct control of VMEbus and of all HEP buses interfaced to it. Consequently physics input and output are made directly with minimum system overhead. Rapid expansion in the VMEbus market ensures upgrade possibilities according to user needs.

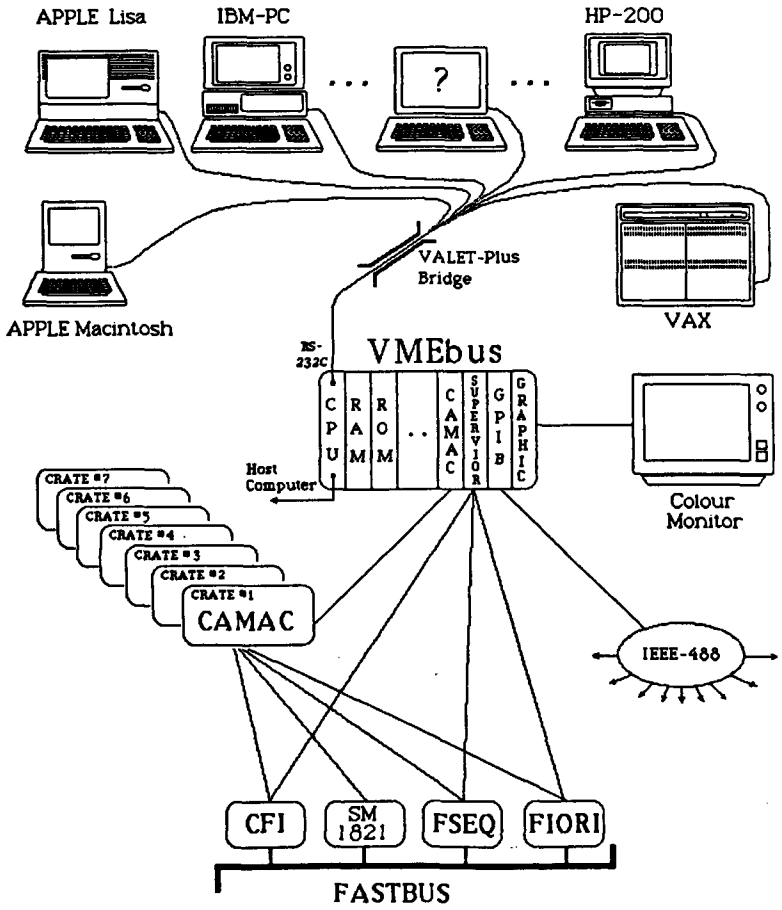
The Personal Computer: A cheap peripheral server and user interface

> Console and standard peripheral access needed in application programs are provided by a personal computer via an RS-232-C asynchronous connection to the VALET M68000. This connection is supported by a BRIDGE program running in the personal computer. As RS-232-C connections are to be found on most PC's, there is little or no restriction on the user choice: hence the "favourite" in the title of this paper. Since most test application software is not disc intensive, the bandwidth limit of RS-232-C is rarely an inconvenience.

3. THE HARDWARE

A VALET-Plus consists of:

- A fully equipped VMEbus crate
- Three mandatory VMEbus modules
- Optional VMEbus modules
- A personal computer



3.1 VMEbus Crate

Recommended VMEbus crates featuring adequate power, cooling, power fail handling and J1 as well as J2 backplanes are available from two commercial sources. Special care has been taken to leave rear access to all J2 connectors free to allow I/O cabling and possible extension bus connections and to permit daisy chain jumpering (Bus Grant, IACK) from the front face.

These chassis are available in three sizes, 5, 9, 20 slots and in table top or rack mounting versions.

3.2 Mandatory Modules

3.2.1 Processor Module

The VALET processor module is the MVME101 General Purpose Monoboard Microcomputer. Our experience with it to date has been good. It is a low cost, reliable, easily available module with good documentation. Nevertheless, new modules are under study to take advantage of the on-going improvements.

The VALET uses the following features of the MVME101:

- 8MHz M68000 processor
- 8 x 28-pin JEDEC compatible memory sockets (2 equipped with 16 kbytes of RAM and 6 equipped with 192 kbytes of EPROM)
- 2 x programmable RS-232-C ports (up to 19.2 kbaud)
- A VMEbus single-level arbiter and seven-level interrupt handler

3.2.2 Read Only Memory

For convenience and efficiency the VALET-Plus software and the run-time system are in EPROMs. In the present configuration, these are installed on both the Motorola MVME101 processor module (128 kbytes in addition to the monitor/debugger) to take advantage of zero-wait state memory access and on a Force SYS68K/RR-1 (or the Plessey PME CRR-1 equivalent) memory module. The Force SYS68K/RR-1 universal memory module (or the PME CRR-1), is not fully used when equipped with 32 kbyte EPROMs. There is, therefore, some room for growth in the code. However, for operational reasons a module capable of using EEPROMs (Electrically Erasable and Programmable Read Only Memories) is obviously desirable. Whilst such devices are not yet of sufficiently high density, trends are being closely watched.

3.2.3 Random Access Memory

The VALET-Plus is configured with the Force SYS68K/DRAM-1 module equipped with 512 kbytes of random access memory.

3.3 Optional Modules

3.3.1 FASTBUS Connection

The Super-VIOR is a VMEbus module used to connect to FASTBUS interfaces. It has been specified by CERN, and marketed by ANTARES. It is a dual 16-bit input/ output register module with a DMA capability. It can drive the FASTBUS interfaces commonly used at CERN, namely, the FIORI, the CFI, and the Fast Sequencer.

3.3.2 CAMAC Interfaces

The VALET-Plus can be equipped with a CAMAC Branch driver complying with the CERN recommendations for CAMAC/VMEbus address mapping. Two modules are in use, the Data-Sud DSSECAMAC and the Creative Electronic Systems C.E.S. CBD 8210. The CBD 8210 is foreseen to have a DMA option available at some later date.

3.3.3 Graphics

The VALET-Plus is capable of graphics display on either the screen of the personal computer or on a colour monitor driven by a VMEbus graphics controller module, or both. The Eltec GRAZ Colour Graphics Controller is based on the Thomson 936x series of Graphic Display Processors. For the VALET-Plus the GRAZ is used to give a resolution of 512 x 256 pixels and reasonable, flicker-free pictures even when used with low-cost monitors.

3.4 The Personal Computer and its Link to the VALET

The connection between the VMEbus system and the personal computer is made at present by means of an RS-232-C connection at 19.2 kbaud, using one of the 2661 USARTs on the MVME101 processor module. Consequently, the constraints put on the personal computer are few. The hardware must include:

- 1 RS-232-C port, preferably running at 19200 bauds
- A disc drive
- A printer
- Graphics

The software must have:

- A PASCAL compiler
- Full program control of the serial port

4. SOFTWARE

4.1 In the VALET

All software modules in the VALET and the tools to develop programs to be executed in the VALET-Plus are based on CERN standard software packages supported independently from VALET-Plus.

4.1.1 MoniCa

MoniCa is a monitor for the M68000 microprocessor family. It provides run time support for Assembler and higher level languages such as C, Fortran, Modula-2 and Pascal. Powerful debugging aids are available. A logical channel concept and interrupt driven input/output operations are implemented. MoniCa is written in Assembler and Pascal and is in widespread use at CERN.

4.1.2 PILS

PILS, a Portable Interactive Language System, is designed to improve portability of application software and programming environment. It has been installed with identical application libraries on VAX, NORD, and VALET-Plus systems. PILS contains ANSI Minimal BASIC as a subset, with additional features which create a more powerful language.

A PILS compiler, which improves execution speed, is presently available for the VAX and will be released for the M68000 in the coming months.

4.1.3 The Application Libraries

Standard libraries also available on VAX and NORD systems are installed with PILS:

- ESONE/IEEE/NIM CAMAC Subroutines
- CERN FASTBUS Subroutines

- CERN Histogram package (HMINI)
- CERN Graphics (PIGS with a Mini-GD3 Interface)
- Math and Trigonometric routines
- ISA Bit manipulation routines

Two additional libraries have been implemented specifically on VALET-Plus:

VMEbus Specific Routines

This is a set of six routines to put or get data to or from any VMEbus address. The routines can handle 8, 16 and 32 bit data. These simple routines are included to allow convenient testing and debugging of almost any VMEbus module from PILS.

GPIB (IEEE 488) Library

A library is available, though not yet released, for the Motorola MVME300 controller, which has DMA transfer rates of up to 500 kbytes/sec.

4.1.4 The Communication Facilities

VALET-Plus offers several possibilities for communication with other computers.

RS-232-C

As previously mentioned, one of the serial ports of the VALET M68000 CPU module is used to communicate with the personal computer. The second serial port can be used to connect the VALET to remote Hosts, allowing the personal computer to be used as a remote terminal and ASCII file transfers between the personal computer disc and a remote VAX/VMS system.

The CERN Network (CERNET)

For installations with a CAMAC link to CERNET, a MoniCa command permits downloading of absolute programs into the VALET processor from any CERNET host. A utility program handles ASCII file transfers between the personal computer disc and any CERNET host.

4.2 In the Personal Computer

4.2.1 The BRIDGE

The BRIDGE is the utility program which allows the personal computer to act as:

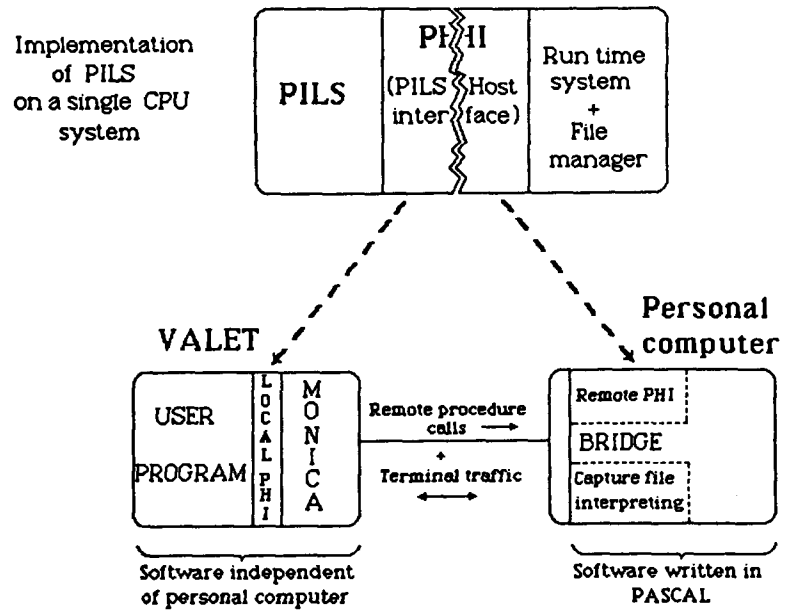
- A terminal emulator
- A server of the remote procedure calls sent by the VALET through the serial link

Depending on the type of personal computer, it can also be programmed to provide special user interface features such as menus, windows, etc.

The remote procedure calls are of two kinds:

- File management requests issued by the PILS Host Interface (PHI) in the VALET M68000 and served by a corresponding "Remote PHI" in the personal computer BRIDGE.

- Graphics requests containing records of the "metafile" issued by the VALET graphics library to be interpreted and plotted on the personal computer screen.



The BRIDGE has been written in PASCAL for portability. System specific code such as control of the serial port, file and graphic management is well isolated and easily modifiable between implementations.

The BRIDGE utility has been installed on:

- Apple Macintosh
- Apple Lisa (MacXL)
- IBM-PC/MS-DOS and compatibles
- Hewlett Packard 200/300 Series
- VAX/VMS

4.3 Programming in Languages other than PILS

Though primarily offered as a PILS machine, VALET-Plus can be programmed in Assembler, PASCAL or FORTRAN 77, by means of a suite of CERN cross-software.

4.3.1 The CERN 68000 Cross-Software

The CERN cross-software for the M68000 family includes Assembler, PASCAL, FORTRAN 77, C and Modula 2. With the exception of the FORTRAN 77 compiler, the cross-software is written in

PASCAL. A FORTRAN 77 compiler written in PASCAL is under development. The compilers generate a common relocatable format CUFOM for which link editors and loaders exist. The output of this chain is an absolute image of the program represented as an ASCII file of S-Records, which can be loaded by the VALET monitor MoniCa. The cross-software is available under VAX/UNIX-BSD4.2, VAX/VMS, IBM/MVS and on several other minicomputers and mainframes. When substantial non-PILS program development is required, the cross-software should be installed, if possible, on the local personal computer, with good hard-disc support. It has been installed on the HP200 family under the PASCAL Work Station System PWS (with the exception of the cross FORTRAN compiler).

A command file allows users to link FORTRAN programs to the libraries which are resident in the VALET EPROM. This reduces the number of S-Records, and therefore the load time, required for main programs which make extensive use of libraries.

5. DEVELOPMENTS

5.1 PILS

A PILS compiler for the M68000, to improve run-time execution speed, is in its final test phase and will be available in the coming months. Compilation will not require any disc access, and PILS programs developed interactively will be compilable at any time. A PILS error recovery facility is under test. It allows a PILS program to branch to a user written error recovery procedure when a run time error is detected, a necessary safety feature of any process control system. PILS event handling will follow, to allow execution of a specified sequence of PILS code on receipt of an external event such as a CAMAC LAM interrupt.

5.2 Communications

ETHERNET is being introduced at CERN and although several personal computers do not support direct interfaces to ETHERNET, it is possible to provide LAN services through a VMEbus connection. Such possibilities are under investigation.

5.3 VALET-Plus to Personal Computer Link

The speed of the link between the VALET and the personal computer could be improved by a factor 2 to 3 by using a synchronous mode of transfer over the serial line. In addition, an ETHERNET / CHEAPERNET connection between a VALET module and a personal computer supporting ETHERNET will be considered for cases where:

- The VALET is far from the PC
- Geographically distributed control of the VALET is required
- ETHERNET access is necessary both on the PC and on the VALET for other reasons.

Support of binary transfers and transmission error recovery capabilities will also be studied.

5.4 New Hardware (MC68020, MC68881, VMXbus, etc.)

The MVME101 CPU and SYS68K DRAM boards could be replaced with a single CPU board with a large on-board memory, bringing the following advantages:

- A saving of one VMEbus slot
- Increased performance because of faster CPU access to (local) memory
- Reduced VMEbus load

High performance CPU modules with MC68020 processor and MC68881 floating point co-processors are being evaluated. These processors increase integer processing speed by a factor of 3 to 5 and floating point processing by orders of magnitude. This is likely to be an expensive option but will provide VALET-Plus users with high performance if required.

5.5 Real time capabilities (RMS68K)

Studies are being carried out to offer RMS68K, the MOTOROLA Real Time kernel, selected by CERN for M68000 applications. Integration of this product with the standard CERN 68000 cross-software package is being done and its availability on VALET-Plus should permit the porting of a basic data acquisition package required by more demanding test applications.

5.6 Data Recording

A binary data recording capability, essential to all data acquisition and even to some test systems, is not available for present VALET-Plus. Two options, the use of a LAN connection to remote hosts with conventional peripherals, and support for a VMEbus based, low cost recording medium are being investigated.

6. STATUS

Six months after the release of VALET-Plus, more than 40 requests have been registered from scientific laboratories and institutes in 10 countries. Most systems incorporate an APPLE Macintosh or an IBM-PC; a few systems use APPLE Lisa, HP200, or VAX.

A complete set of documentation is available on a self-help basis on CERN IBM WYLBUR service. A VALETNEWS scheme based on electronic mail has also been implemented under WYLBUR to communicate with the user community.

7. CONCLUSIONS

The current interest in VALET-Plus, the spectrum of personal computers used, the high competition on the VMEbus market and the recognized benefit of real portability of application programs seem to indicate that the concept is valid and the choice of standard basic hardware and software components is right.

Interest in the concept have also resulted into implementations with other 68000/MoniCa based designs and porting of the VALET-Plus environment has been achieved on the FASTBUS General Purpose Master (GPM) by the DELPHI collaboration and on the CAMAC "Controlleur de Branche d'Acquisition" (CBA) by the OPAL collaboration. A feasability study of a generalization of this concept might be initiated to benefit from the user-friendly interfaces available on low cost personal computers from a PC independent Command/Menu package running in a separate application processor.

Acknowledgements

Many colleagues in the high - energy physics community and in the CERN Data Handling Division have helped directly or indirectly to refine our ideas on the VALET-Plus architecture. H.Von Eicken and R.Nierhaus are thanked for their help with MoniCa and graphics respectively and the DD-SW group for their support with cross software matters. C.Mazzari, J.C.Pinasseau, J.Bourges, the DD Division Electronic Workshop and K.Protoulis have provided invaluable technical and administrative assistance.

References

1. VALET-Plus Overview - OCDOC GA/NOTE17 C.Parkman, Y.Perrin, J.Petersen, E.M.Rimmer, P.Scharff-Hansen, L.Tremblet OC GROUP - GA/NOTE17 CERN Data Handling Division.

2. VALET-Plus Operations Guide OC GROUP - GA/MAN7 CERN Data Handling Division.

3. VALET-Plus Hardware Guide OC GROUP - GA/MAN6 CERN Data Handling Division.

4. PILS Reference Manual R.D.Russell, L.Tremblet, D.O.Williams OC GROUP - GEN/MAN2 CERN Data Handling Division.

5. PILS Installation Manual R.D.Russell, L.Tremblet, D.O.Williams OC GROUP - GEN/MAN3 CERN Data Handling Division.

6. Standard CAMAC Subroutines OC GROUP - GEN/MAN5 CERN Data Handling Division.

7. Users Guide to CERN FASTBUS Routines OC GROUP - CERN Data Handling Division.

8. HMINI Common Histogramming Library OC GROUP - GEN/MAN8 CERN Data Handling Division.

9. MiniGD3 User's Guide DD/US/90 CERN Data Handling Division.

10. STANDARD FORTRAN MATHEMATICAL ROUTINES (For example: VAX or NORD FORTRAN Reference Manuals)

11. BIT MANIPULATION ROUTINES ANSI/ISA Standard S61.1 For Bit Manipulation Routines

12. CERNET SIMPLE FILE TRANSFER Network Project Note 83 CERN Data Handling Division.

13. PRIAM UNIX SERVICE PRIAM/VAX/84/7 CERN Data Handling Division.

14. MONICA CERN Data Handling Division.

15. CERN Universal Format for Object Modules DD/US/83 CERN Data Handling Division.

A STAND ALONE VMEbus 68k TEST SYSTEM RUNNING CP/M

Authors: Y. Bertsch, A.Degre, J. Lecoq
Address LAPP, BP 909, 74019 ANNECY CEDEX FRANCE

H. Von Der Schmitt
Physikalisches Institut Universitat HEIDELBERG

ABSTRACT

We describe a test system we have set-up to be used in LEP experiments. Under VME standard, "open", stand-alone under CP/M, VAX "like", it is very powerful in CAMAC, VME and FASTBUS applications. An optimized multi-user possibility is set-up just by adding CPU cards.

The FORTRAN 77 native compiler developped by one of us, processes all standard packages available at CERN (CAMAC - FASTBUS- MZCEDEX- HMINI-HPLOT....) with very good performances and makes the system user friendly. PASCAL and C native and cross compilers are available.
An ETHERNET link with a VAX VMS has been developped above level two in C allowing a file transfer of 100 kbits/second.

Performances are competitive with larger configurations. Cheap price makes such a system very promising for future large experiments which need more distributed CPU power, specially for test and monitoring.

1 INTRODUCTION

The most popular computers associated to test systems generally don't meet well the requirements for fast hardware applications. But the simultaneous availability of 68K microprocessors and of the VME bus allows setting-up of new, cheap and well suited configurations. Preparation of LEP experiments leads laboratories in charge of electronic tests to update their system. Several set-ups can be considered, provided they meet CERN standard requirements for CAMAC and FASTBUS both in hardware and software aspects.

L.A.P.P has the responsability to develop CAMAC and FASTBUS hardware units and to set-up a test system of the second level trigger of the L3 experiment.

To achieve our goals we need a powerful "hardware oriented" tool verifying the following requirements:

- Stand alone .
- Fast in hardware debugging and tests.
- User friendly for non informatician people (VAX like) .
- Supporting a fast native FORTRAN 77 compiler .
- Able to use the CERN PRIAM cross software .
- With an efficient connection to ethernet allowing file transfers with a VAX VMS.
- Easily upgradable.
- Cheap (less than 25000 SF totally equipped).

Because VME has been designed to work in a multi-CPU environment, these considerations have oriented our choice to a mono-task mono-user disk operating system. Therefore, a multi-user configuration is set-up just by adding extra CPU cards,in such a way that the multi-user is taken care directly by VME hardware i.e without system overhead for hardware performances.

The popular CP/M system enriched by the Fortran-77 compiler developed by one of us (Hans Von Der Schmitt) fits well our requirements.

The ETHERNET connection will allow the system to be integrated in the L3 ON-LINE configuration to perform trigger tests under VAX's control.

2 SYSTEM OVERVIEW

2.1 HARDWARE

The hardware, built around VME bus, allows to define an "open" system, compatible with a wide range of modules, without excessive manufacturer dependance. We chose the 12 Mhz "ELTEC" CPU driven by the CP/M disc operating system for the following reasons:

This CPU was the fastest 68K available. It includes a 1 Mbytes memory on the board avoiding unnecessary bus access. The same board includes also the floppy disk interface and two RS232 ports. In a multi-CPU environment, these features increase the performances.
The bus arbitration unit is not implemented on our CPU. The Motorola MVME 101, introduced for some extra reason, takes care of this function.

In order to boost the system performances we have added two memories:
The first one is a RAM memory, which emulates a disk, used exactly like a fast disk.
The second one is a ROM module containing utility programs (actually Fortran 77 and libraries).
To avoid the associated back-up constraints, we chose not to use hard disks as mass storage but only two 1 Mbyte floppy disks. While slow, this solution is sufficient.

CAMAC is driven via the DATA-SUD interface.

FASTBUS is driven via the VIOR-FIORI interface and soon via the CFI interface.

ETHERNET is connected via the LRT interface which is a master.

XOP, a fast processor developped at CERN, is driven via its VME interface.

To be compatible with MONICA applications, we have added a MVME 101 CPU with its external RAM memory. As stated before, this CPU is also used as our VME arbitration unit.
The VME address space of the entire configuration is described in figure 1.

2.2 SOFTWARE

In High Energy Physics, the most commonly used machines are VAX and IBM. To avoid users to learn a new system, we have created an environment as close as possible to the VAX context. It has been possible since the very primitive operating system we have selected, let say CP/M, is easily hidden for the most common applications.

But primitive does not mean not efficient. On the contrary, the mono-task/mono-user feature is a great advantage for speed and efficiency.

The multi-cards possibility being arbitrated by hardware directly on the bus, we take advantage of this feature.The multi-users possibility is implemented just by adding extra CPU cards. So, in a multi-users configuration, speed penalty will appear only during conflictual access to the bus. It is reduced to a minimum by using CPU cards with a large memory on board and with direct access to peripherals.

If a file has to be edited, a full screen editor behaving just like the VAX EDIT is invoked by the command: EDT Filename.

Cross-compiling is a very useful tool, but we have to remember that object file has to be downloaded after each modification of the source code. Downloading of large files via RS232 line, can reach up to one hour.Such a procedure becomes unusable for development of large programs. As FORTRAN is the most common language in our applications we feel important to use a native compiler.

The FORTRAN 77 native compiler developped by one of us (HVS) ensure compatibility with CERN utility standard packages as:
- MZCEDEX : command processor and menu facility.
- HMINI and HPLOT : standard graphic packages.
- CAMAC : standard ESONE calls
- CERN FASTBUS :

This compiler is available in the native mode and as cross compiler on the VAX VMS.It satisfies almost all F77 requirements except real*8 and direct access for files manipulations. Its run-time performances have been particularly developed.
Some performances of the compiler are summarized in table 2.

As for an example, the development of a F77 program will be achieved as follows:

FOR test (compilation: produces ASM code, then object code)

LINK test (links program with FORTRAN libraries)

TEST (starts execution of program TEST)

If the program has to be recalled from VAX, the sequence available is the following one:

HOST (transparency on host)

SUP (initiates on VAX the facility for file transfer)

XFER (on CP/M initiates file transfer)

is the host a VAX ? (Y/N)
host file name:
local file name:
host to local ? (Y/N)

The file transfer is initiated and is available in both directions, either for text files or 68k images which are decoded from S format to binary format to be stored on the CP/M mass storage.

For small tests, a basic-like language may be useful, we intend to install PILS on our system before the end of 85.

In the LEP context with ETHERNET environment, it is obvious that a fast connection with the usual host (VAX) will be a major feature and will make possible a new range of applications during setting up of the experiment . A task to task link above ETHERNET has been realized and is described in a following subsection.

3 DEVELOPMENTS AT LAPP

3.1 VME

Initially, we chose to built the test system around the VME bus, because VME is an industrial standard, able to manage multi-master configurations with good performances. We have never foreseen to perform development on VME, but we'll certainly be obliged to understand why mixing units from different manufacturers in a same crate is not straightforward. We have to understand and solve this problem because we need at least three masters in the same crate. We hope the problem will be solved by using a more efficient general purpose arbitration unit... which has been missing up to now.

As many modules are provided with fixed addresses, especially in the short I/O range, we often had to reconfigure our VME space.

3.2 CAMAC

With the VME-CAMAC interface, we have run CAMAC under excellent conditions: Being our system a mono-user one, we can switch CAMAC on and off at any time.

After optimisation of the CAMAC library, CAMAC cycles (standard ESONE calls), have been measured at less than 60 microseconds with the 12 MHZ CPU.

In our FORTRAN compiler , the CAMAC interface is a part of the VME memory.This allows FORTRAN programs to run CAMAC cycles in a few microseconds.

3.3 FASTBUS

Developments will increase in the next months at LAPP. The L3 trigger is largely based on FASTBUS. FASTBUS is actually running on our VME system and the last FB package has been implemented. The commonly used test programs(FBMON,FDMTST) are being implemented. We have tested that we are really able to compile and link FBMON locally as well as with the cross-compiler (4500 FORTRAN lines giving 19000 assembly code lines).

The first large FASTBUS prototype, a 500 components module developped by CAD, with a local developed integrated circuit on board, is just completed. Hardware debugging and maintenance tests will be performed with this system.

3.4 NETWORKS

ETHERNET is the most popular LAN used at CERN. Most mini's such VAX's, APOLLO's UNIX VME systems can communicate on ETHERNET via TCP/IP. TCP/IP however is UNIX oriented and except for the VMS VAX, not adapted outside UNIX environment. Another problem is the cost of the connection. Among the ETHERNET cards presently on the market, there are intelligent ones, with firmware on board up to level 4 (transport in ISO model). This solution is attractive but expensive ($4000) and in fact only adapted to UNIX systems. Moreover, it requires to install the TCP/IP package on VMS VAX ($4000) which doubles the overall price. In fact for a VME UNIX or UNIX like system the connection would actually equal the total price of the system.

For small, low-cost systems another solution was suitable. A classical scheme, particularly developed at CERN is the SERVER/CLIENT couple.
A SERVER, installed on the HOST is waiting for requests from different CLIENTS and must be able to satisfy several of these requests.

We have realized such a connection above level 2 of ETHERNET, with the present restriction that the SERVER on the VMS VAX is able to satisfy only the request from one VME's CLIENT.
The software has been derived from the existing package developed at CERN (Ben Segal, private communication) for the link between the PRIAM VAX UNIX and the MOTOROLA MVME CPU running MONICA in VME.

This sofware, written in C had to be adapted to other machines (VAX VMS and CP/M VME 68k). The development, started last spring is now completed. It has been facilitated by the portability of the C language and the support we found at CERN. However this implementation has obliged us to considerably modify the original software.

The encountered difficulties were both on software and on hardware.

For the hardware the board delivered by LRT at the end of july 85 presented a serious bug. This bug did not appear systematically on all cards delivered at CERN, and consequently slowered its debugging. After having delivered a packet the card was unabled to "hear" any incoming packet unless a carrier was present on the cable just before.

This has been temporary solved by preceeding each VME destination packet by any packet bringing the required carrier shot.
The sofware also carried up some difficulties. The native VME/C compiler delivered by WESTERN DIGITAL comprises some bugs and does not implement some features of the C language.
All these considerations have obliged us to modify the code written for a VAX (we had previously developed the SERVER/CLIENT couple between two VAX's).

This solution, which exhibits a connection cost of about $1000 (plus transceiver)allows a 100 kbits/sec file transfer between the VAX VMS and a VME system.

3.5 L3 DEVELOPMENTS

Our main applications with this system are done on L3 experiment and specially on the second level trigger
In this framework we are developping some new, fast FASTBUS interface (slaves and master),as well as some CAMAC modules.
Using VME CP/M system in this context allows us to replace heavy slow VAX's by little, fast, self configurable 68000 cpu's. One of these systems will stand on the L3 experiment as the second level survey and maintenance system.

4 CONCLUSION

We presented a stand-alone 68K test system running the standard FORTRAN CERN packages.
Very good performances obtained in CAMAC and FASTBUS meet well requirements needed for hardware developments.
The new coming tools as : native F77 compiler, VAX compatible screen editor, remote login, file transfer... make the system powerfull and user friendly for program developments and hardware tests.
The ETHERNET connection will allow the system to be integrated in the L3 online structure to perform second-level trigger tests "in situ" under VAX's control.

This system has been running for six months without serious problems. Its low cost, (less than 25000SF totally equipped), with all needed interfaces needed in CERN experiments, boosted by 5 Mbytes memories (RAM+mass storage) makes it very attractive. If desired, the multi user feature can be obtained virtually by adding extra CPU's, or by using the CERN cross software on the VAX and downloading S records from VAX to VME.
To go further, we have to investigate the multi-master aspect. This work has pointed out the problem of hardware compatibility between some VME manufacturers.

ELAPSED TIME TO COMPILE MZCEDEX AND MZCOMM ON ELTEC VAX

MZCEDEX.FTN
MZCOMM.FTN
|| RTFMAIN
v

RAMDISK
RTFMAIN (EPROM)

	12 MHZ	8 MHZ	NATIVE	CROSS
MZCEDEX.S MZCEDEX.LIS	5'45	9'50	1'00	8'50
|| AS68 v MZCEDEX.O JADDR.O	1'00	2'20	----	----
|| L068 v MZCEDEX.68K	0'10	0'15	----	10'

2496 LINES - 68K

CROSS SOFTWARE DEVELOPMENT IN A UNIX ENVIRONMENT

G. Beier, L. Kappen, R. Lutter, K. Schöffel
B. Stanzel, K. Steinberger, H. Wilhelms
Beschleunigerlabor der Universität und
Technischen Universität München
D-8046 Garching

Abstract

For stand-alone applications of 68000 processor boards a runtime library was developed. It consists of routines for I/O to serial, parallel, and IEEE bus ports, exception handling, floating point emulation, and most function calls available under UNIX. The major part of it is written in C and follows the UNIX calling conventions. The library, which may be stored in EPROM, was used with VMEbus cpu-boards from different manufacturers. Program development (C and assembler) is done on a UNIX host system.

Introduction

At the accelerator laboratory of the Munich universities, we use VMEbus based 68000 processors for two kinds of applications:

- in dedicated systems (e.g. a CAMAC auxiliary controller interfacing our old PDP-15 online stations)
- in a multiprocessor system for data acquisition and analysis (MLLE) [1,2]

For these purposes, we needed

- a powerful software production environment (programmer's workbench)
- a unified runtime system for processor modules from different manufacturers

Furthermore, the minicomputer used for program development also works as the user interface of the MLLE multiprocessor system. This leads to demanding similar environments (e.g. equal languages) for programs on the minicomputer and on the VMEbus board.

We found, these requirements could be best served by using UNIX (System III and V) as our host system and writing ULRIKE, a UNIX like Runtime Kernel for the 68000 microprocessor.

Some words about UNIX as a development system:

- Multi-user timesharing system:
  running on a wide scale of computers (from micros to mainframes) with some interesting features with respect to the development of large program systems

- Hierarchical, tree structured file system: very well suited to keep an order among numerous source files

- Source Code Control System (SCCS):
  Keeps track on the development of source files. All changes are added as *Deltas* to the original version and labelled with version numbers. Any previous version may be recovered with a simple command. Further it is effectively guaranteed that only one user at a time makes changes on a specific version of a file.

- *Make* utility:
  Can be used to describe even very complex building processes of target files from numerous source files. Works well together with SCCS. Only those files are processed (compiled, assembled, etc.) that are necessary due to recent changes on the source files. Prerequisites of files may be declared, so e.g. untouched files will be newly compiled if changes were made on their include files.

- Programmable user surface (shell):
  Procedures may be written in the command language which compress complex lists of commands into shorthand. These procedures may use variables from a user-specific environment. The functionality is comparable to the VAX DCL.

- Availability of source code:
  UNIX system sources are available at moderate costs for educational institutions. A lot of our microprocessor software is directly taken from the UNIX System V sources.

Cross Software for the Motorola 68000 (languages)

All software development for microprocessors is done on the UNIX host computer. Our present microprocessor software is entirely written in C, with only a few percent of assembler code in the runtime kernel. We used (until recently) a C compiler written at Siemens which was adapted to our PDP-11/55 UNIX System III computer, and a self-written assembler. The assembler has no real macro features, but uses the standard C preprocessor which allows a lot of useful things like include files, symbolic constants, conditional assembly/compilation etc. We found a macro assembler unnecessary, because complicated things are better done in a high-level language. Assembler and C both produce relocatable binaries in UNIX standard format. Linking is done with a modified version of the UNIX *ld* link editor which allows relocation to given physical addresses. Base addresses may be given independently for text (=code), (initialised) data, and bss segments (=uninitialised data), respectively.

More languages

Since recently we also use a 68000 based UNIX system (CADMUS of PCS). Because we use the standard UNIX object format, all compilers available on this machine (C, FORTRAN, PASCAL, MODULA-2 etc.) may be used for cross software production. Till now we tried FORTRAN-77 besides the native C compiler.

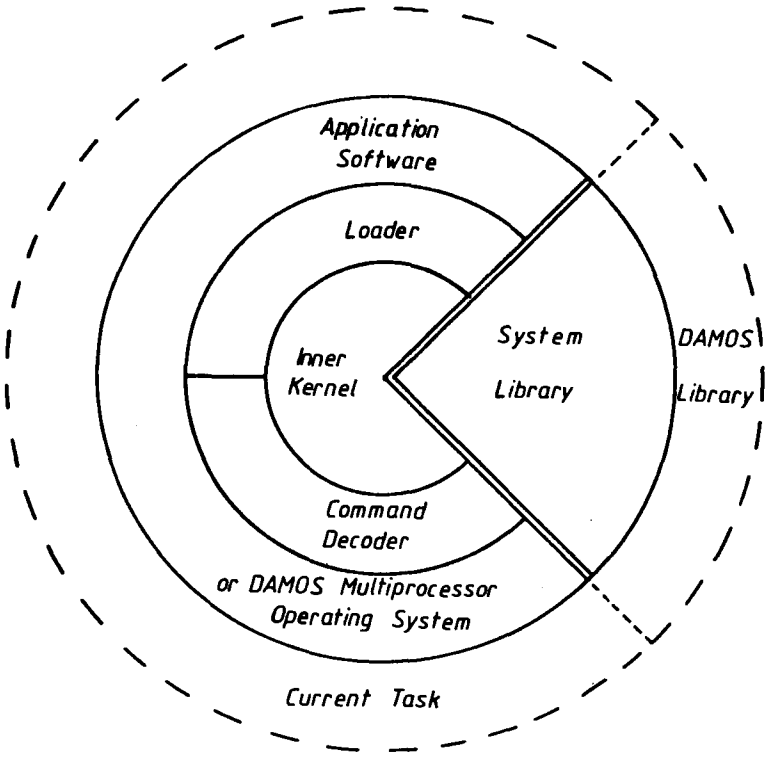
ULRIKE - the UNIX like Runtime Kernel

Our ULRIKE system consists of four basic components,

- an inner kernel which contains startup routines, exception handlers and device drivers,
- a system library with functions for all layers of software,

- a downline loader to load programs from the host system, and
- a command decoder to interface the intrepid user.

The figure shows a layer diagram of ULRIKE. The top layer(s) are formed either by application software in a single processor, or the DAMOS operating system [2] with its current task in a multiprocessor configuration.



Software Layers of ULRIKE

We do not support multitask programming on a single processor, because we think that every task should have its own processor. However, as several programs may reside in core simultaneously, they can be started sequentially by the command decoder.

The Inner Kernel

The inner kernel performs the very basic system functions:

On power-up or reset, all devices are initialised, stacks and trap tables are set up, and control passes to the command decoder.

Exception handling is either done by standard exception handlers, or control is passed to user-defined routines. The assignment of new exception routines is made at runtime by entering the new start addresses into the appropriate tables. In C, this can be done very simply, because the start address of a function is accessible within a program as a pointer to that function.

The terminal driver is compatible with UNIX System V. It consists of two parts, one generic with all the special character handling like XON/XOFF, rubout etc., and the other one device dependent, containing all the special features of the utilised hardware. The latter was adapted to several commonly used serial I/O chips (Valvo DUART 68881, Motorola ACIA 6850, Rockwell 68560, Mostek MFP 68901 and SIO 68564). Only few definitions have to be changed (base address etc.) to fit the driver to a new CPU or serial I/O-board, if it uses one of the already implemented chips.
As a special feature we implemented virtual terminal lines, which can connect any two processors in a multiprocessor system. This communication uses the common VMEbus memory or, if available, the VMSbus. By choosing the appropriate line mode, this link can be used for very different applications in a multiprocessor system:

- error message output to a single terminal or host computer,
- system initialisation and shutdown,
- inter-processor task communication and synchronisation.

A modified UNIX driver serves the IEEE 448 bus. We use a MIZAR interface with a TMS9914 chip from Texas Instruments. Also, a simple straightforward driver for the Motorola PI/T (MC68230) parallel interface chip is available.
Realtime clocks accessible with the standard UNIX calls work with the timer part of the Motorola PI/T, Mostek MFP (MK68901), and National Semiconductor's battery buffered MM58167A realtime clock chip.
Access to CAMAC is done with a dedicated processor (AIAC), briefly described elsewhere [2]. We also intend to use a VMEbus CAMAC branch driver.

At the moment, we don't plan to connect other peripheral devices, especially no file structured devices like floppies or winchester disks. We think, they are better handled by the operating system of a host computer. On the long run, however, bulk storage devices like streamer tapes or optical disks might be interesting for the fast storage of raw data.

System Library

The system library offers all the routines that make the runtime environment look very similar to UNIX.
It holds

- the standard C I/O library
- Motorola's fast floating point library
- emulations of all UNIX system calls applicable to a single-task system (routines to handle timers, semaphores, signals, etc.)
- the UNIX libraries of mathematical and string handling functions
- miscellaneous other functions, mostly for system programming purposes (such as priority setting, linking of exception routines, etc.)

All these functions are reentrant and reside, except for debugging purposes, in EPROM. They can be called from every layer of software. Application programs are linked only with a symbol table describing all entry points and execute the code directly from EPROM. This keeps the size of the application program small (and downline loading fast).

The Loader

With the downline loader, code files are loaded from the UNIX file system to the microprocessor memory. We either use a serial line or an IEEE 488 bus for program loading. In any case, code is transferred in slightly modified UNIX loadmodule format, using 8-bit transfers. So far, no problems with transfer errors on RS232 lines at 9600 baud arose, even at program sizes of about 256 Kbyte. The transfer speed is about a factor of 2.5 faster compared to ASCII-coded S-records.

The Command Decoder

The command decoder parses the input string of a communication line and dispatches to the appropriate routine.
A command line consists of a command name, an optional set of switches, and a list of arguments. Possible argument types are integer, floating point, and character string. Command and switch names may be abbreviated and are tested for uniqueness. A command may be connected to several routines using different argument lists. The decoder provides function calls to add new commands and to define their argument lists. The

default command list includes standard commands like *help*, *initialise*, and *show*, and some facilities known from the Berkeley shell such as *alias* and *history*.

* * *

[1] G. Beier et al., IEEE Transactions on Nuclear Science, Vol. NS-32, No 4, 1985, p. 1426

[2] H. Wilhelms et al., MLLE, A Dataflow Controlled Multiprocessor System, talk given at this conference

AN APPROACH TO PROGRAMMING MULTI-MICROPROCESSOR BUS SYSTEMS

Janusz Zalewski
Institute of Atomic Energy
PL-05-400 Otwock/Swierk
Poland
+48(22) 799516

Abstract

General view to program standardized multiprocessor bus systems from the user level is presented, covering: configuration management, ligical-to- physical mapping, resource sharing, and interprocess communication. The project goal is to establish conventions which allow for programming idependent of the architectural differences and for producing portable application software. Emphasis in examples is given to programs expressed in Ada.

1. INTRODUCTION

In programming standardized bus systems like CAMAC, Fastbus, Multibus, VMEbus, it is extremely important to have a consistent set of rules governing the access to modules and transfers of data. It was recognized in the early days of CAMAC that widely accepted routines are of substantial assistance to the programmers. First such rules have been prepared long ago and published as the "Subroutines for CAMAC". Their application in practice has led to the high increase of programmers' productivity. Numbers reported from CERN said of about 7 times increase, at least in certain application. From my own experience I know that the complexity of programming was reduced so drastically that all application software for quite sophisticated control systems was possible to be done by technicians.

This trend is still vital and the next generation of subroutines has been published - "Standard Routines for Fastbus" [1]. However, this document when presented may cause head-ache to the non-experienced users as being very complicated, despite the explanation that it is complicated proportionally to the complexity of Fastbus. The users' need is evident: even if a hardware architecture is far more sophisticated than ever before, software must be simple. What we are suggested to do then is to simplify the programming principles.

The importance of simplicity issues was confirmed by an earlier language selection study [2]. The investigation which language is best suited to programming modular interfacing systems led to the

conclusion that even with properly selected but purely technical criteria one cannot diversify languages like C, Modula-2 and Ada

- a statement known also from more recent publication [3]. However, if we take into account one meta-criterion - simplicity of use -the choice is more straightforward.

To explain further this philosophy it would be instructive to repeat a statement by Harlan D. Mills of IBM:

"Software stands between the user and the machine".

In terms of Fig. 1 it means that the portability has two faces. The term stands not only for a mobility from one machine to another but also - from one user to another. The users who want to have friendly systems, in addition to simplicity of languages and simplicity of calls would prefer an operating system to be transparent. The approach taken here starts at the user level and does not go down to the operating system kernel.

We identified four in principle different problems, which cause most severe difficulties to an average user when programming multi- microprocessor bus systems. Two of them are rather static in nature:

- configuration management, that is identification and initialization of boards before any application task is running

- logical-to-physical mapping, relying on partitioning the overall system tasks among the available processors. During the execution of tasks, when a bus traffic is concerned, two kinds of things may happen in general, connected with the distribution of processors along the bus:

- communication between tasks located on different processors

- collision in their access to certain resources.

The next sections will discuss proposed technical solutions.

2. CONFIGURATION MANAGEMENT

The purpose of a configuration management is to allow software to identify, configure, initialize, and diagnose the boards. Regarding the complexity of all contemporary bus system architectutes, automatic configuration management is essential for the user. Configuration manager should be able first of all to recognize the current system configuration, installed processors, memories, and other modules and resources.

The solution is to have a table for configuration information. This may be a software table, a configuration ROM or a whole space for storing this kind of data. Fastbus and NuBus, for instance, have separate control spaces in addition to the normal data or global memory space. In Multibus II there is a separate interconnect space, where all this configuration information is contained. The space splitting concept is most useful in that it resembles in hardware data typing in programming languages. If there is no separate space the hardware environment may be described by means of a small configuration table.

The report on configuration status (status selfcheck) should contain all relevant information and provide it to the application program. Except of submitting and updating the configuration status, the

configuration manager should be able to perform initializations and tests or initiate self-test routines of the configured equipment. Whenever it is run from a terminal, a console panel or a program, before any application task is executed, the user should know all the available resources and their location, and be sure this is safe and consistent information.

Unfortunately, due to the diversity of architectures, the unification of the form in which this information should be presented is hard to achieve. One bus architecture (NuBus) may contain 64 or more contiguous bytes of information on each board in the system, and for another (Multibus II) this is well-organized for each module as an interconnect template consisting of a header, function and EOT record. Therefore, a variety of configuration information should be available to the user via standard software databese being an interface to application programs.

Visibility of configuration registers to the users is then limited. Users do not know what information is located where in the registers and need access to the database only. A system database may contain names for boards in the system, their addresses (logical and geographical), connection tables, and other relevant information [1]. It seems, however, that there is no common practice in this respect. The Fastbus Software Working Group has left the issue for further revisions of the standard.

Essential of the proposed database organization follow the work of the IEEE P896 System Architecture Subgroup [4], although may be incomplete for non-conventional architectures. The principle is that the information is splitted into 2 categories:

- administrative, like serial number, part number, vendor id, board type, revision level, etc., which is mostly read only
- program-related, like device driver, diagnostics and flag register offsets, resource type, bus options etc. (acces to this portion of database is strictly controlled).

3. LOGICAL-TO-PHYSICAL MAPPING

The issue of allocating tasks to processors is certainly a central one in programming multiprocessor bus systems. It is desirable that all mapping be done automatically. Given the device/processor connections map to the busses, software should allocate tasks onto the available hardware.

Conceptually this can be done in a language or by a tool in an environment. However, in contemporary high-level languages there are no explicit mechanisms for the control of the distribution of tasks among processors. There is no common programming language in terms of which one can describe target configuration and allocation of program units to physical nodes according to application requirements.

If one drops a solution based on a language extension, and there does not exist a compiler which translates any of the program texts into code mapped to the individual nodes (hardware modules), the conversion is still possible of a single program to multiple programs by a preprocessor, which will use a representation specification to convert code. This also looks like working with a tool. Therefore a clear need exists to create an environment for developing and executing software that is targeted to a distributed system.

The most comprehensive study related to this subject was conducted by the Ada community [5] (this is the point where we tie to Ada, since no respective investigations have been done for other languages). The main result is that preference should be given to implement the application software as one portable program, and that the tasks should be grouped on processors to match their functions, and to minimize code duplication and remote communications.

It is required that the user provide - except of a source code - the following specifications (Fig. 2):

- task groups composition
- target system description
- composition of nodes (consisting of processors)
- allocation of task groups to nodes.

Decisions should also be made regarding the static or dynamic allocation. If objects created at run-time are involved, allocation could be left to the responsibility of the user, but not recommended. In a real time environment there is a heavy target dependence of tasks, because of the unavoidable representation specifications in programs. This helps to automate the allocation process and to make it static. For dynamic mapping the information from the configuration database can be used to map tasks onto nodes. The description of the distribution of tasks should have no effect when the program is compiled for a single processor, thus allowing exactly the same text to work on uni- or multiprocessor.

An elementary example of Ada program distribution among processors was given in [6]. The criterion for partitioning is the minimization of remote data transfers. For a procedure P (Fig.3) to be executed on 4 processors, A, B, C, and D, a module is expressed in extended Ada (Fig. 4) and mutual references between objects are voted as a basis for allocation.

4. RESOURCE SHARING

Multiple processors in a bus system may share a global resource. Examples include sharing common peripherals, high speed math boards performing floating point operations, network communication controllers etc. As far as a kernel can provide software regions to prevent collisions between tasks sharing resources, numerous languages equally allow expressing mutual exclusion by lock operations and resource sharing by semaphores with read-modify-write sequence. Bus lockin is typically used for indivisible test and set instructions used to implement semaphores for interprocess communications. Nevertheless on the language level it is highly desirable to have a single construct to express the intention of resource sharing in a program. This is rarely provided even in most advanced languages, where the mutual exclusion is typically achieved by creating a critical section, i.e. a code segment that once begun must complete execution before it, or another critical section that accesses some shared resource, can be executed.

An interesting solution is the addition of a LOCK statement to Modula-2 [7], giving a simple syntax for mutual exclusion:

LOCK <mutex> DO statement_sequence END

where <mutex> is a designator of type Mutex provided by additional module THREADS, which also provides another type Condition used in connection with some mutex, and several procedures for operation on mutexes and conditions. This solution is both more safe and simpler for multiprocessor applications than the original constructs in Modula-2.

5. INTERPROCESS COMMUNICATION

An important aspect of a multiprocessor architecture is the exchange of data between processors. In fact, the data transfer is the main objective of setting up a bus system. Although it has much common

with sharing resources along the same bus, both problems are conceptually different (Fig. 5). As far as sharing the resources should be explicitly expressed in the application program, the communication between processors may not be. This is because of the preference given here to write an application software as one portable program. Then the user has no a priori knowledge of the distribution of communicating tasks (processes) along the processors, and the single construct in a language is only needed for intertask (interprocess) communication.

Though the interprocess communication is separated from the interprocessor communication, both may overlap when defined groups of processes are placed on several processors. The implementation of an Ada rendezvous for this purpose reveals a number of dangerous situations, which may cause severe difficulties in processing as results of:

- idenfinite waiting for a response message (timed or conditional entry call after the start of the rendezvous will not avoid the problem)
- unlimited block transfer length
- remote procedure calls, and others.

The asynchronous nature of tasks execution on different processors means that the operating system must have some facility for buffering information, e.g. through a mailbox structure. Another assumption is that it should handle the communications between processes on different processors in a way transparent to the user. Although there are other solutions, message passing with strictly defined protocol is the most natural choice for interprocessor communication. Essential for message passing and all communication interchange is data typing. Sending and receiving only messages of the associated type may eliminate a lot of the danger.

6. SUMMARY

In a contemporary multi-microprocessor bus systems it is necessary for the user to provide something more in the program than the conventional data, address and operational information related to the system response or status. We identified the additional areas, where establishing conventions is essential to facilitate the users to write distributed programs. These are: configuration management, logical-to-physical mapping, resource sharing and interprocess communication.

As they are related to the application program not only during execution but also in the static phase, it would be helpful to have a configuration language to deal with the specification of software components and their interconnections. The Ada language may serve the purpose after being augmented as a Program Design Language [8]. Its private data types hide the representation and generics may circumvent more serious differences in physical systems.

The language approach to programming multi-microprocessor bus systems cannot be regarded as generally superior than building application software on the top of the real time kernel. It seems that in a short time scale the former is more costly. The economic justification is in favor of it for longer periods due to the better portability of programs. Whenever a community of users realize that they can, with minor effort devoted to learn a highly efficient language, master the programming of sophisticated systems and exchange the programs as they are, this approach would survive.

7. REFERENCES

1. Fastbus Software Working Group: Draft Specifications for Standard Routines for Fastbus, March 1985

2. Zalewski J.: Language Selection for Modular Interface Systems. Part 1, Proc. IFIP WG2.4 Working Conf. on System Implementation Languages, P. King, ed., North-Holland, 1985. Part 2, Proc. Europhysics Conf. on Software Engineering Methods and Tools in Computational Physics, Computer Physics Communications, Vol. 39, 1985

3. Williams D.O.: Software and Languages for Microprocessors. 6th Summer School on Computing Techniques in Physics, Nove Mesto na Morave, Czechoslovakia, 17-26 September 1985

4. Laws G.: A Proposal for a 32-bit System Architecture Based on the NuBus. Tentative Draft, IEEE P896 System Architecture Subgroup, 27 February 1984

5. CISE, TXT, SPL International: A Feasibility Study to Determine the Applicability of Ada and APSE in a Multimicroprocessor Distributed Environment. Final Report, March 1983

6. Cornhill D.: Distributed Ada Project. Honeywell Systems and Research Center, Minneapolis (MN), 1985

7. Rovner P. et al.: On Extending Modula-2 for Building Large Integrated Systems. DEC Systems Research Center, Palo Alto (CA), 26 December 1984

8. Ada as a PDL Working Group: Ada as a Design Language. IEEE P990 Draft, 28 May 1985

* * *

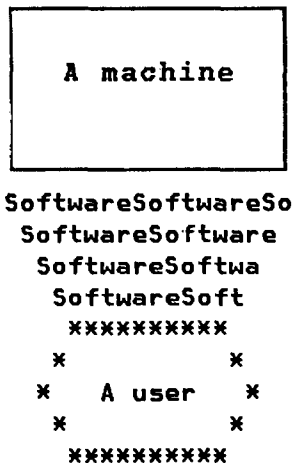


Fig. 1. Software is portable between machines and between users.

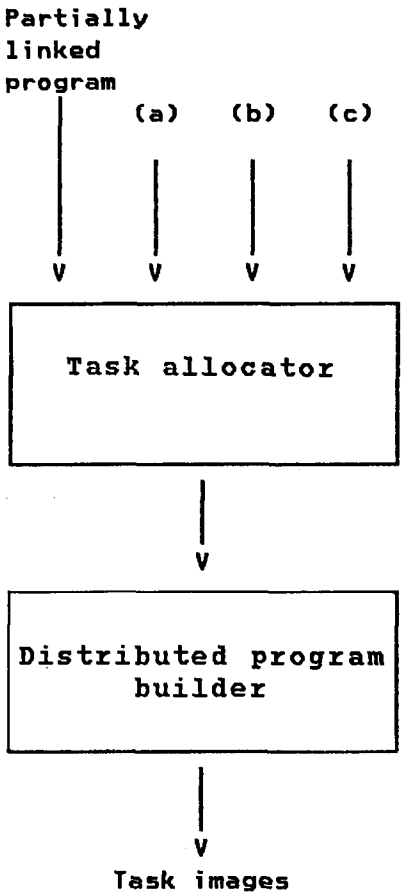


Fig. 2. Information provided by a user for proper task distribution: (a) target description and node information, (b) task group composition, (c) allocation instructions.

```
procedure P is
   X,Y : INTEGER := 0;
   function F(I : INTEGER) return INTEGER is
   begin
      return I + 1;
   end;
begin
   Y := F(X);
end;
```

Fig. 3. Procedure to be partitioned.

```
procedure P at A,B is
   Y : INTEGER at C,D;
   function F(I : INTEGER) return INTEGER at C,D
      return INTEGER at C,D;
   access
      use Y : INTEGER
         do VOTE;
      use function F(I : INTEGER) return INTEGER
         do VOTE;
end P;
```

Fig. 4. Expressing a procedure when voting for distribution.

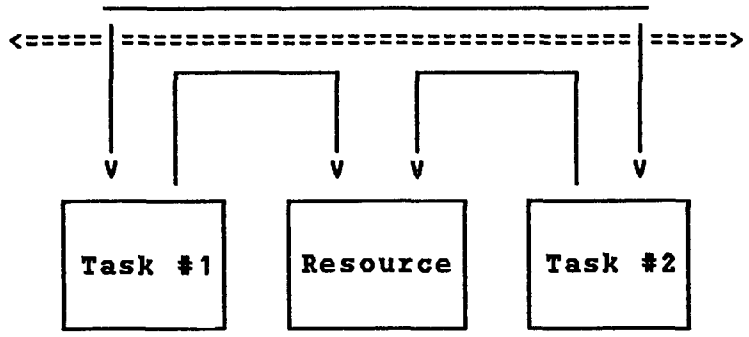


Fig. 5. Sharing resources and intertask communication.

A VME INTERFACE TO AN IBM MAINFRAME COMPUTER

J. Alexander
Daresbury Laboratory, Warrington WA4 4AD, England

Abstract

The development of a VME interface to an IBM 360/370 I/O channel is described. The interface is capable of emulating IBM I/O devices to minimise the software development required in the IBM computer. Burst data rates in excess of 750 Kbytes per second have been achieved. The use of the interface, together with a VME to Ethernet interface board, to allow minicomputer-based data collection systems to exchange files directly with an IBM mainframe is also described.

Introduction

At Daresbury Laboratory we have a requirement to connect our central IBM-compatible computer to an Ethernet local area network, so that data acquisition stations on the laboratory's Synchrotron Radiation Source can transfer data directly to the mainframe for analysis and long term storage. It was decided to do this by developing an IBM to VMEbus interface to be used in conjuction with a commercially available VME to Ethernet interface. In this way it should be possible to connect other networks and devices to an IBM in the future with minimal hardware development. This paper describes the IBM to VME interface design, and outlines the way in which it is to be used in the IBM to Ethernet interface project.

Interface Design Philosophy

The interface is designed to connect to a 'standard' IBM I/O channel (referred to as a channel throughout this paper) without using any of the additional features, such as 16-bit-width data transfers or data streaming, specified in reference 1. It is able to emulate different IBM I/O control units since it contains a 68000 microprocessor which determines the way in which it interacts with the channel. It can operate on all three channel types: Selector, Block Multiplexer or (Byte) Multiplexer; although in the last case it can only emulate a control unit interfacing a single device.

The block diagram of the interface, Figure 1, shows that it consists of four sections: the microprocessor and its associated program and data memory, an interface to the channel, an interface to the VME bus, and a fast data path between the last two. The fast data path allows data to be

transferred between the channel and a unit on the VME bus (such as a memory card or a data link controller), without the involvment of the microprocessor other than to set up the transfer. The microprocessor controls, monitors and exchanges data and parameters with the other sections of the unit via memory-mapped registers.

This architecture was chosen to give the microprocessor overall control of the interface, while not loading it or its bus (by use of a DMA chip) with end-to-end data transfers, - to maximise the end-to-end data transfer rate. This assumes that the microprocessor does not need to process the data, which is generally the case in the communications environment for which the interface is intended. However the capability is provided for the microprocessor to transfer data directly to or from the channel, and it is perfectly feasible to transfer part of a block this way and part via the fast data path. This might be used, for example, if the block consisted of a header section containing control information and a separate data section. Alternatively, if the fast data path is used to transfer data between the channel and a VME memory module, the microprocessor can directly access the data in the memory.

The last is achieved by mapping most of the microprocessor's address range onto the VME bus, as shown in Figure 2. Any microprocessor access to an address above 64 Kbytes automatically generates a VME bus cycle with the same address (via the VME interface section). The Address Modifier code is determined by the microprocessor's Function Code outputs and by whether the address is in the top 64 Kbytes, the latter determining whether the 'Standard' or 'Short' address codes are used.

Communication between the on-board microprocessor and another VME based processor is expected to be via VME memory accessible to both. Synchronisation between the processors is achieved by the unit's ability to generate a VME interrupt; and to accept a 4-byte 'message' written to it as a VME slave. The message is stored for the microprocessor to read later, and the microprocessor interrupted.

Channel Interface Section

The channel interface section is built around a PROM-based sequencer, which responds to the handshake sequences of the 'Tag lines' used by the channel to communicate with control units. As it does so it loads information from the channel (addresses, command codes and data) into registers which can be read by the microprocessor; and transmits information to the channel (addresses, status and data) from other registers which are loaded by the microprocessor. The microprocessor controls the order of Tag sequences performed, and monitors their progress, via control and status registers. It also loads registers to specify to which channel addresses the sequencer will respond, and to specify the Initial Status byte returned to the channel for all possible commands. Thus the sequencer handles the (fixed) hardware level protocol of the channel, while the microprocessor handles the higher level protocols determined by the programs running in the channel controller. The microprocessor can be programmed to emulate standard I/O control units, to obviate the need to write channel control programs specifically for the interface.

Completion of a transfer via this path is signalled to the microprocessor when the transfer length counter has been decremented to zero and the FIFO in use has emptied. It is also signalled (as a different event) if the channel uses the STOP sequence to indicate that its byte count is exhausted.

VME Interface Section

This section receives, and arbitrates between, requests to use the VME bus by the microprocessor and by the fast data path. Upon receipt of such a request it arbitrates for mastership of the bus (if it has not already got it), and upon gaining mastership gates the address, data and timing signals between the selected source (microprocessor or fast data path) and the bus, inserting appropriate delays to comply with the VME timing specifications. The VME 'Release When Done' algorithm is used to release the bus if the fast data path is not in use, and the 'Release On Request' algorithm if it is (to achieve maximum data transfer rates). The arbitration level used is selected by on-board patching.

This section will also respond as a D16 slave to three consecutive 'double-byte' addresses in the VME Short Address range. (The addresses it recognises can be set by on-board patching.) Two of these addresses are used to accept a 'message' to the on-board microprocessor. The third can be used to access the fast data path as a slave, but only if the on-board microprocessor permits this via the contents of a control register. Without such permission an access to this address causes a Bus Error response; as does a read if the fast data path is set up for an IBM read operation, and vice versa.

The interface generates a VME interrupt, on a patch selectable level, when the microprocessor loads a 'vector' into one of its registers. The vector is transmitted to the VME bus during the subsequent interrupt acknowledge bus cycle.

Fast Data Path

The fast data path contains a FIFO (First-In First-Out) de-randomising buffer for each direction of data transfer, and logic to convert between the 16-bit words transferred over the VME bus and the 8-bit bytes transferred over the channel. It also has a VME address register/counter and a transfer length counter, which are loaded by the microprocessor to specify a DMA-style transfer over the VME bus. The operation of the fast data path is described for an IBM write command; a read is simply a reversal of this process.

When enabled by the microprocessor, the channel interface loads data bytes received from the channel into the write-FIFO as long as this has space available and the transfer length counter is non-zero. (If space is not available the channel handshake is witheld until it is.) The transfer length counter is decremented for each byte loaded. As bytes become available at the VME end of the FIFO they are extracted and paired to form 16-bit words. The order of bytes within a word is determined by a control register loaded previously by the microprocessor. Each word is presented, together with the VME address, to the VME interface section for transfer over the bus; after which the VME address is incremented, decremented, or held unchanged - this also being determined by the control register.

Test Facilities

To facilitate initial testing and to allow confidence checking of the unit prior to going online to the channel, a data loopback path at the channel interface is made whenever the unit is offline. This allows the microprocessor to check its data, address, status and command paths to and from the channel. It can also exercise the fast data path by writing bytes into the write-FIFO or reading them from the read-FIFO (using test addresses), while at the VME end transfers occur on the bus as normal. (The microprocessor can become bus master to address the fast data path as a VME slave if the slave mode is enabled!)

Current Status of the IBM to VME Interface

A prototype unit has been built and commissioned. It occupies 3 VME boards interconnected by a front panel bus, with a further small board located on the remote panel supporting the channel interface connectors. (The last holds circuitry to buffer the channel signals, and a relay to provide an onward path for the daisy-chained Select Out signal when the interface is offline or powered down.) When tested on the laboratory's central NAS (IBM compatible) computer, the data rate via the fast data path, on a 15 metre long channel cable, was measured to be over 850 Kbytes/sec during the data transfer phase of a channel operation. The overhead time for the channel to issue a command, the sequencer to respond with an initial status byte and to inform the microprocessor, the microprocessor to set up a fast data path transfer, the transfer termination to be indicated, and the microprocessor to direct the sequencer to send 'ending status' to the channel was measured to be 130 microseconds. This gave an average data throughput rate of 770 Kbytes/sec when transferring 1 Kbyte data blocks.

The IBM to Ethernet Interface Project

As stated earlier, the IBM to VME interface was designed as part of a project to connect data acquisition stations on the Synchrotron Radiation Source to the laboratory's central computer. There are currently 23 stations, with some 10 more projected, each controlled by a PDP 11/04, LSI 11/23 or (in the future) microVAX computer. These currently communicate with a VAX 11/750 computer which carries out some data analysis and has a network link to the central computer for onward transfer of data. The direct communication links between the experimental stations and the VAX are currently being converted to use a single Ethernet network. This carries a Remote Procedure Call (RPC) protocol designed, at Daresbury, to permit the reliable and rapid transfer of large volumes of data with minimal software overhead (references 2 and 3). It is intended to extend the Ethernet to the central computer, so that data acquisition stations can transfer data directly to it. This will permit a greater overall rate of data throughput, free the VAX to concentrate on analysis work, and facilitate interactive working with the central computer by a user at a data acquisition station.

The IBM to Ethernet interface is currently undergoing software development; its hardware configuration is shown in Figure 3. It uses the IBM to VME interface described here, together with a commercially available Ethernet interface, memory card and CPU card in a VME crate. The 68000

microprocessor on each of the IBM and Ethernet interfaces will simply direct the transfer of data to and from buffers within the memory card, while the 68000 on the CPU card handles the details of the RPC protocol. The CPU card, together with a small amount of software in the central computer, implements the RPC 'manager' process for this Ethernet node. Software will be written to implement 'server' processes within the central computer: initially file servers to write incoming data to disk.

It may eventually prove feasible to dispense with the CPU card by amalgamating the RPC manager process with processes on one or both of the interfaces. Initially, however, it was felt that the comparatively small extra cost of a separate CPU card was justified by making the software easier to develop and de-bug, since in this way each microprocessor handles a well defined and separate task.

Acknowledgements

The IBM to VME interface is an adaptation of an interface, developed at Daresbury, for the GEC 4190 'front end' computer currently handling all communications with the central computer (reference 4). Thanks are due to E. Owen and P. Kummer for discussions concerning the architecture of both interfaces; to them and to T. Daniels and P. Havercan for help in elucidating the complexities of an IBM channel; and to G. O'Neill and I. Lazarus for hardware design work on the GEC interface.

* * *

References

1) "IBM System/360 and System/370 I/O Interface Channel to Control Unit Original Equipment Manufacturers' Information"
IBM document number GA22-6974-5

2) "The Use of a Remote Procedure Call Protocol for High Speed Data Transfer on an Ethernet"
by R Tasker, F Rake, P Kummer and D Hines
To be published in 'Interfaces in Computing'. Daresbury Laboratory preprint number DL/CSE/P15

3) "Remote Procedure Call for Ethernet, Protocol Specification Version 1.0"
by P Kummer and R Tasker Daresbury Laboratory internal document number DL/CSE/TM35.

4) "Programmers' Guide to the IBM Channel Interface for the GEC DMA-bus"
by J Alexander Daresbury Laboratory internal document.

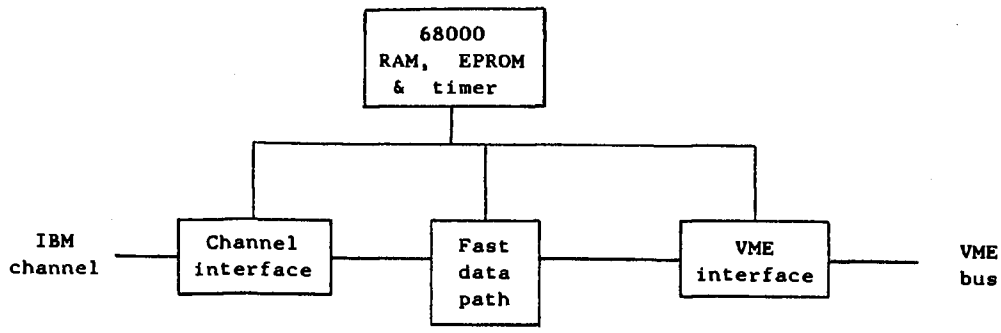


Figure 1: Block diagram of the IBM to VME Interface

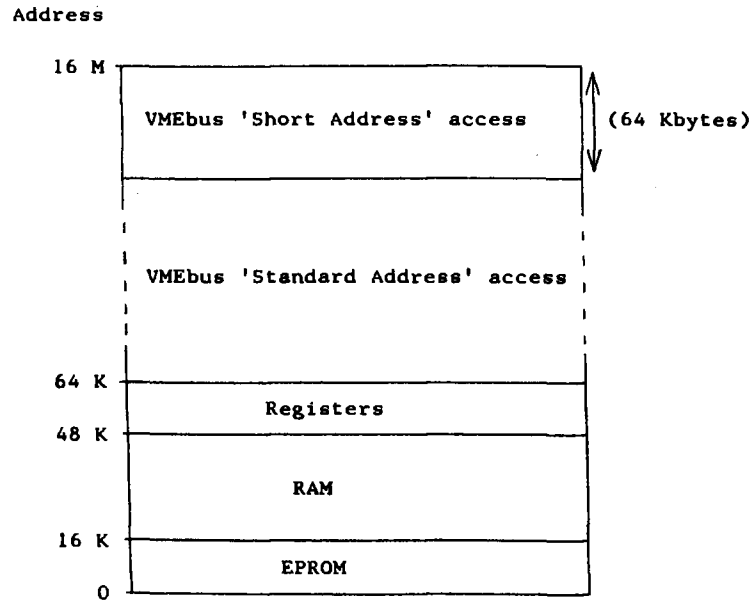


Figure 2: Memory map for the microprocessor in the IBM to VME interface

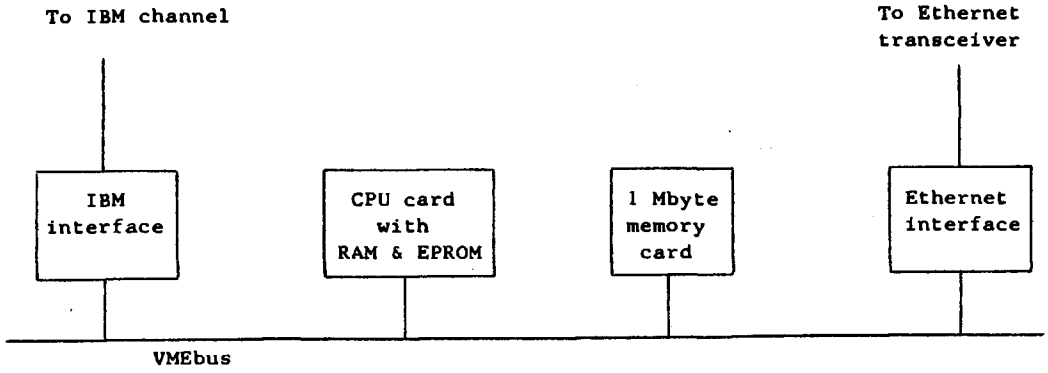


Figure 3: Configuration of the IBM to Ethernet Interface

REMUS-VME-VMX BRANCH DRIVER (RVMEX) TYPE V385

C. Engster and L.G. van Koningsveld

October 23, 1985

ABSTRACT

The RVMEX is a D16,A24 VME SLAVE with a D16,SEQ VMX MASTER and 2 "REMUS" front panel connections. The card can be used as a CERN RWBD[1] type 283.

Controlled via VME , the card may perform autonomous data transfers between a REMUS branch and a VMX bus. The RVMEX can be a VMX PRIMARY MASTER if alone or a VMX SECONDARY MASTER if used in conjunction with another VMX PRIMARY MASTER. Only 16 bit block transfers with sequential addressing and auto-increment are allowed.

The REMUS branch can also be read by VME under program control. The RVMEX has 2 REMUS connectors which allows it to be used as a branch DRIVER if a RVMEX DRIVER TERMINATOR is connected to the upper connector, or as a branch SPY if connected within a branch.

When the card is initialized to "VME-VMX" then block transfers from VME to VMX for both read and write are possible.

1. INTRODUCTION

CAMAC equipment is widely used in high energy physics experiments.

The reading of multi-crate, multi-branch systems in a tree structure has been satisfied by the REMUS concept.

---

[1] see EP-Electronics Note 80-01, February 1980

REMUS is used in the parallel readout of the UA1 experiment[2] at CERN where 170 CRATES are grouped in 28 BRANCHES.

Due to the upgrading of the UA1 experiment the need of a specific REMUS BRANCH DRIVER in a VME environment has arisen.

2. APPLICATION AND OUTLINE

This new module , the RVMEX (Remus VME VMX Branch Driver) type V385 is described in this paper. For the UA1 event builder a VME system has been chosen. The REMUS readout for the CAMAC crates in the experiment was kept. Thus the VME bus for control and the REMUS branch for data readout were imposed. As a local DMA bus the VMX bus (Rev. A) has been chosen. The Rev. A version of the VMX bus is well suited for large data block transfers with unknown block length, since the auto-increment addressing mode allows the full VMX address space. Since REMUS data are 16 bits wide and the CPUA1 contains a 68010 both the VME port and the VMX port were limited to 16 bits. This allowed the implementation of VME-VMX transfers with few additional electronics. Controlled via VME , the card may perform autonomous DATA transfers between a REMUS branch and a VMX bus. In the UA1 experiment the VMX bus is connected to a CPUA1 and a DPRX (128 or 256 kbyte dual port memory).See fig. 1. The RVMEX can be a VMX PRIMARY MASTER if alone or a VMX SECONDARY MASTER if used in conjunction with another VMX PRIMARY MASTER. The REMUS branch can also be read by VME under program control. The RVMEX, which has 2 REMUS connectors, can be used as a branch DRIVER, if a RVMEX DRIVER TERMINATOR is connected to the upper connector, or as a branch SPY if connected within a branch.

When the card is initialized to "VME-VMX" then block transfers from VME to VMX, for both read and write, are possible.

Due to the dedicated use of the RVMEX some restrictions to the VME specifications have been made. Only WORD access and only 24 bit addressing are allowed. The card will respond to Standard Supervisory Data Access or to Standard Non-Privileged Data Access or to both, depending on the programming of a PAL on the card. Normally the RVMEX is programmed to respond to both.

The REMUS branch driver is restricted to "single word" operation. This means that the mode of operation in which the N-lines of the REMUS branch are transfered together with every data word is NOT supported (see RWBD type 283). The REMUS functions are generated by writing a bit pattern into a register via VME. Since the generation of the REMUS "read-in" signal is also programmable, the "Prepare & Go" can be connected to any sub-address.

---

[2] see The UA1 Data Read-Out System in this proceedings

The card contains a 512 word FIFO between REMUS and VMX or VME.

Up to 15 RVMEX cards may be set at the same coarse-base-address. A hexadecimal switch on the front panel allows to differentiate the actual base-addresses of the individual RVMEX cards from 1 to F. This is at the same time the BRANCH NUMBER. Branch number zero corresponds to a broadcast address to which all cards respond to write functions. This allows the generation of REMUS functions on up to 15 branch drivers simultaneously. An EVENT count is sent together with the BRANCH NUMBER in order to verify the synchronization of data belonging to the same event.

3. MODULE DESCRIPTION

3.1 COST REGISTER

The COntrol and STatus register controls the data flow and the type of operation of the module. See fig. 2. The COST can be accessed while autonomous REMUS to VMX transfers are taking place.

3.2 REMUS FUNCTION REGISTER

The REMUS function register is used to generate the REMUS functions on the branch. When the RVMEX is in TEST mode then the REMUS Functions are generated but only used inside the card.
When used as a SPY then the REMUS Functions are not generated by the RVMEX (even if programmed) but are spied from the REMUS BRANCH.

3.3 TEST REGISTER

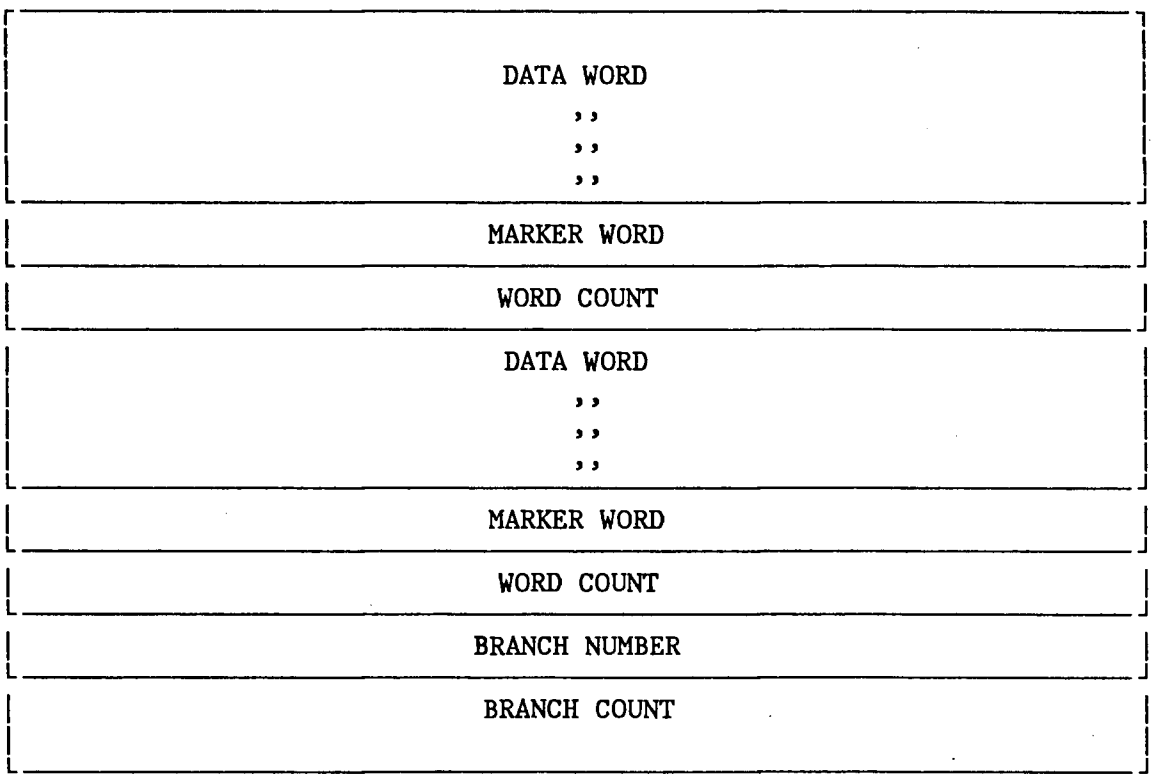
The TEST register can be used to simulate a REMUS branch. The same register serves as data register for REMUS write functions.

3.4 COUNTERS

The branch driver has to generate count words which have to be inserted in the REMUS data structure. Therefore a WORD COUNTER and a BRANCH COUNTER are part of the RVMEX hardware. A sequencer takes care of the correct insertion of the count words.

3.5 REMUS DATA BLOCK FORMAT

BLOCK FORMAT:



There are no reserved bits to distinguish between the different types of words used within a REMUS data block. Therefore the data has to be sorted out scanning backwards through the data block, knowing the total length (BRANCH COUNT) and the individual data block length (WORD COUNT). The last four words are always as indicated in the block format and the word before a WORD COUNT is always a MARKER WORD. DATA WORDS and MARKER WORDS are read from the REMUS BRANCH while WORD COUNT, BRANCH NUMBER and BRANCH COUNT are generated by the RVMEX. WORD COUNT and BRANCH COUNT are 16 bits wide and include themselves. The BRANCH NUMBER generated by the RVMEX has a special format since it also contains data of an EVENT COUNTER.

4. ACKNOWLEDGEMENTS

We would like thank S. Cittolin and M. Demoulin for their definition and support of this project. Software support has been provided by W.J. Haynes, whose ability to find the last bugs in the RVMEX has been of great help for us.

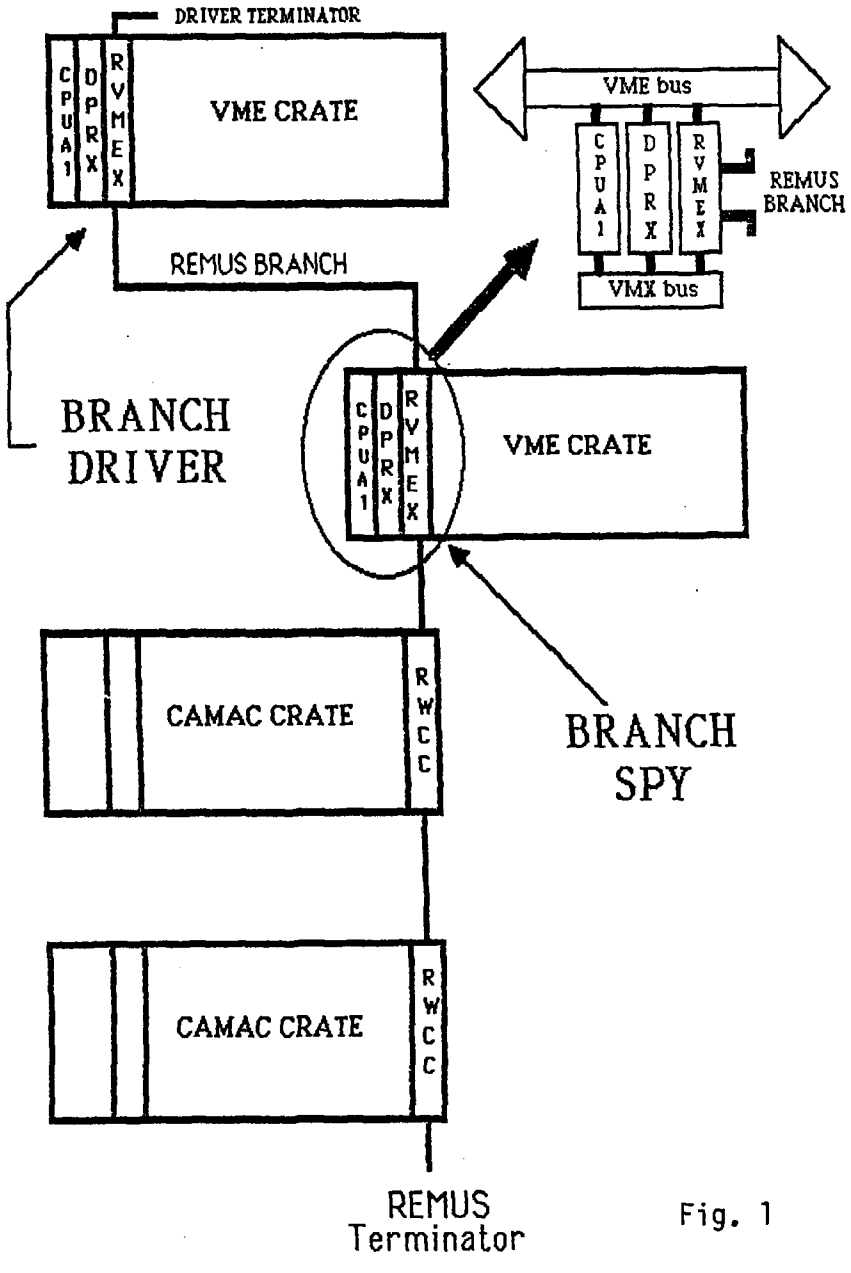
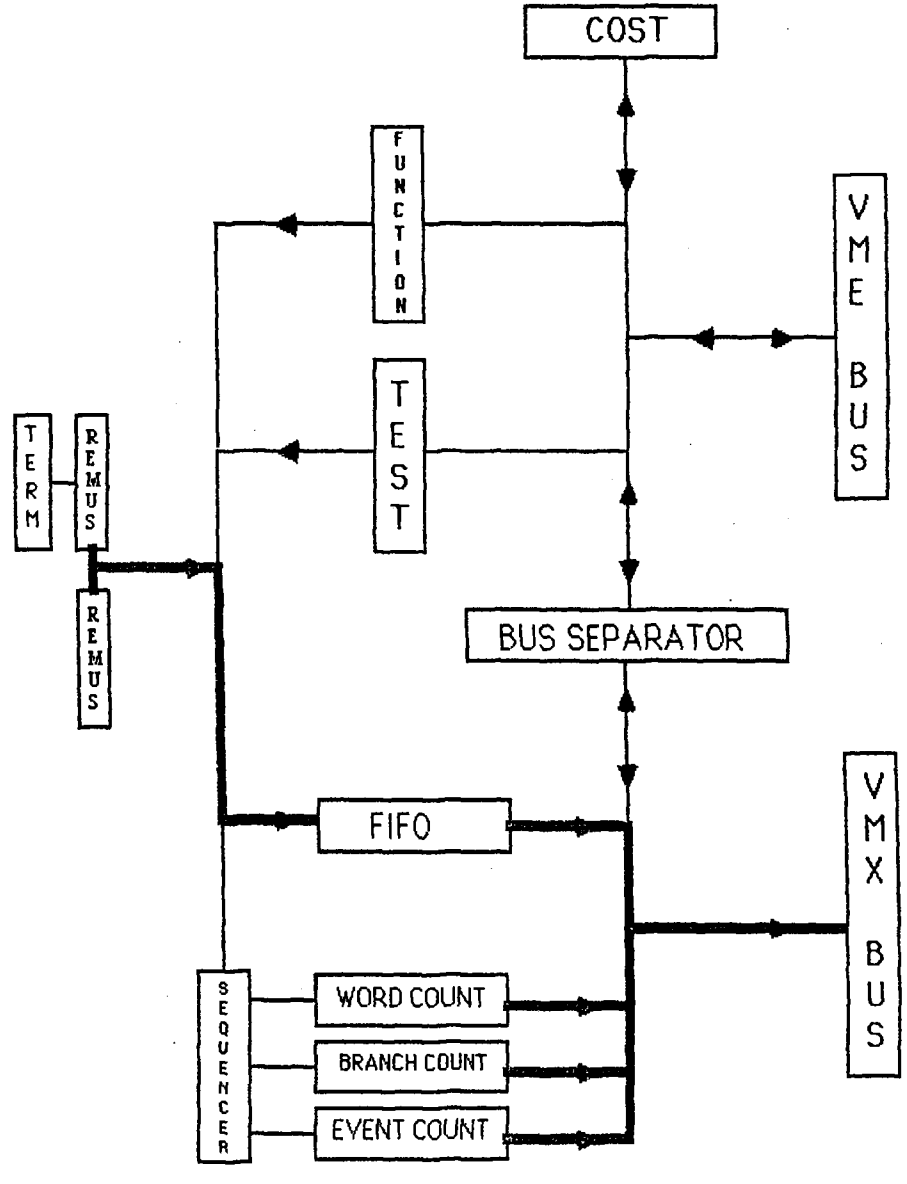


Fig. 1



REMUS-->VMX

Fig. 2

THE NEW CONTROL SYSTEM OF THE SACLAY LINEAR ACCELERATOR

J.F. Gournay, G. Gourcy, F. Garreau, A. Giraud, J. Rouault
Service de Physique Nucléaire- Accélérateur Linéaire, CEN Saclay
91191 Gif-sur-Yvette, France

A new control system for the Saclay Linear Accelerator designed during the two past years is now in operation. The computer control architecture is based on 3 dedicated VME crates : one crate with a disk-based operating system runs the high level application programs and the database management facilities, another one manages the man-machine communications and the third one interfaces the system to the linac equipments. At the present time,communications between the VME micro-computers are done through 16 bit parallel links. The software is modular and organized in specific layers, the database is fully distributed. About 90% of the code is written in Fortran. The present status of the system is discussed and the hardware and software developments are described.

INTRODUCTION

The ALS, in operation since 1969 [1], was primarily manually controlled, a computer was introduced into the accelerator control system in 1974 for centralization of informations, automatic surveillance of the main parameters and control of the beam switchyard [2,3]. During the next 8 years this system was expanded with a second computer and with ponctual local processing power (8 bit micro-processors), but it became obvious in 1982 that the system had to be replaced very soon. The old technology computers were completely obsolete, their maintenance was difficult and expensive and the software developments very tedious. In this paper we describe the solutions that we have adopted for the new system.

HARDWARE SYSTEM

For the new control system it was decided to take advantage of the progresses in the micro-processors technology to ensure reliability, flexibility, versatility and a good cost effectiveness. A solution based on a standard and modern bus with powerful CPU and I/O boards available from many different manufacturers looked very attractive. Among the different possible choices, the VMEbus and the MC68000 microprocessors family seemed to be the best solution for the process computers.

Process computers

The functions of the system are distributed between 3 dedicated VME crates (fig.1) :

. the "HOST" crate with 1MByte of memory, a 20 MByte disk drive, a floppy disk for backup. This station is running under a disc-based operating system.

. the "LINA" crate with 512kByte of memory and all the interfaces to the equipements of the linac.

. the "OPER" crate with 1MByte of memory and all the interfaces to the man-machine communication devices.

The LINA and OPER stations have no disk and their software is downloaded from the HOST station. The three stations are close enough to be interconnected with parallel links, the protocol used enables 50 kBaud data transfer rates on these links. The stations are equiped with "first generation" VME CPU boards: 8MHz MC68000, small in-board memory size, no MMU, no floating-point coprocessor and a limited private bus reserved for I/O.

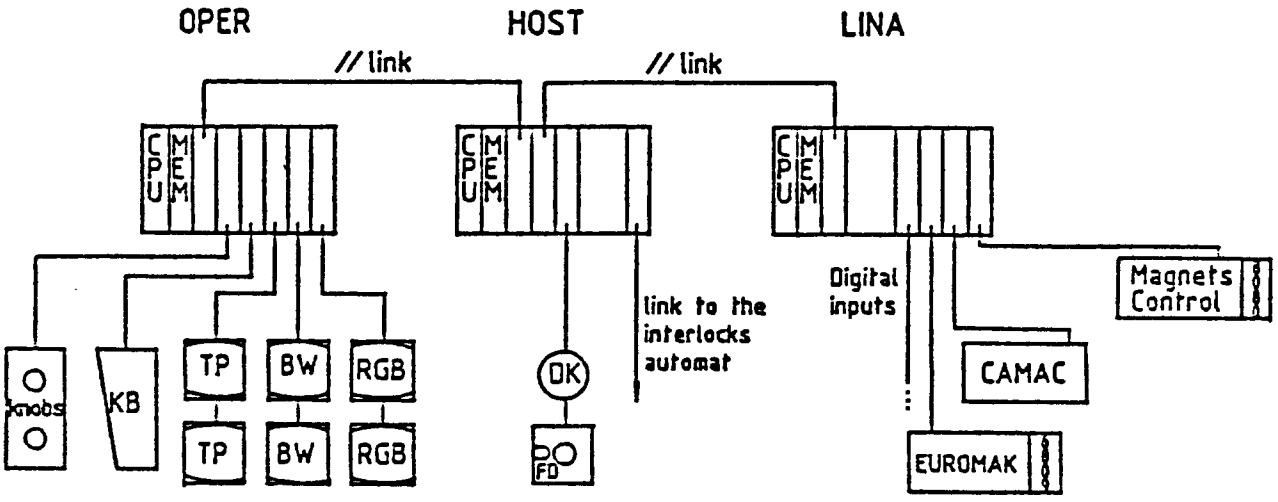


Fig.1 - Hardware Configuration

Interface to the equipments

All the equipments were previously interfaced through CAMAC modules. This solution suffers important disadvantages (expensive, relatively fragile, nuclear but no industrial standard) compared to its advantages (international recognised system, great variety of modules), so it was decided to progressively replace CAMAC by more modern standards.

One solution could be to interface the equipments directly to the VMEbus but this solution is rather expensive due to the complexity of the bus. So direct I/O interfaces on the VMEbus are reserved for the console equipments where fast response is primordial and for digital inputs where the VME module can directly interrupt on the CPU board. This last module was designed in our laboratory. For the console operation most modules were commercially available (high resolution colour graphic boards, touch panel and keyboard interfaces) except a shaft encoder interface that we have also designed. This module handles a pair of encoders and provides two convenient features: a software controllable brake for limiting the operator actions and 2 software controllable LEDs for warning the operator of the encoder(s) assignment.

For machine equipments interfacing we prefered to use a low cost 8-bit standard bus with industrial cards mechanically and electrically robust. The French EUROMAK bus was chosen : it uses single EUROcards, MC6809 micro-processors and 15 slots crates. A lot of industrial modules are available from the manufacturer. A first experience with EUROMAK is now acquired with the digitization of about 300 analog input channels splitted in 3 crates each containing multiplexers and ADC boards. The LINA station and the EUROMAK crates are temporarily linked with serial RS232C channels.

Future improvements

The major improvements planned in the next few months concern the hardware links between the VME crates and between the VME and EUROMAK crates. An ETHERNET network is now available in the VME standard and will replace our parallel links, the data transfer rates will be much higher and it will be easy to interconnect the VME crates to the micro-computer development system (MOTOROLA EXORmacs) for software down-loading.

It is planned also to interconnect our system to the nuclear physicist computers through the ETHERNET network, in order to make the linac parameters directly available to the physicists. Due to the bus structure of this network, it will be straightforward to add new VME crates if they are needed to increase the power of the system. The objection that this network is not very suitable for a control system since a maximum response time is not guaranteed [4] is not founded in our case: we have a small number of stations and the stations exchange only non time critical messages (updates of local and central data bases, operator commands, display lists).

The same type of network could be used to interconnect the VME and the EUROMAK crates, however as we planned in the future to increase the number of EUROMAK crates, an ETHERNET network will become prohibitively expensive. Following the choice of CERN for the LEP project [4] the aircraft MIL/STD-1553B network will be used for this purpose. The 1553B interfacing to the EUROMAK bus was done in our laboratory, two boards are used to manage the communications: the bus controller (BC) and the remote terminal unit (RT), one BC can control a network of up to 30 RTs. Following arrangements with the manufacturer the EUROMAK 1553B network is now commercially available.

To interconnect a VME master crate with EUROMAK slave crates equipped with 1553B RTs a VME compatible BC must be furnished. One such module for the military and aerospace market is available, but its price is incompatible with the type of applications aimed here. Therefore, taking advantage of the experience gained with the EUROMAK modules the VME BC module was also designed in our laboratory. As shown in figure 2, it is built around the SMC COM 1553B chip and is controlled by a MC6809 microprocessor which unloads the main processor from the 1553B hardware protocol (management of the 32 words 1553B messages, of the command and status words). Two dual-ported 16 Kbyte RAM and two semaphores are used to manage the exchanges between the local MC6809 and the main processor. These two areas can be used completely independantly: while the MC6809 performs a list of commands in one area, the host processor can prepare a new list or can read response to a previous command list in the second one. An interrupt is sent on the VME bus to signal the end of processing of a list by the local MC6809.

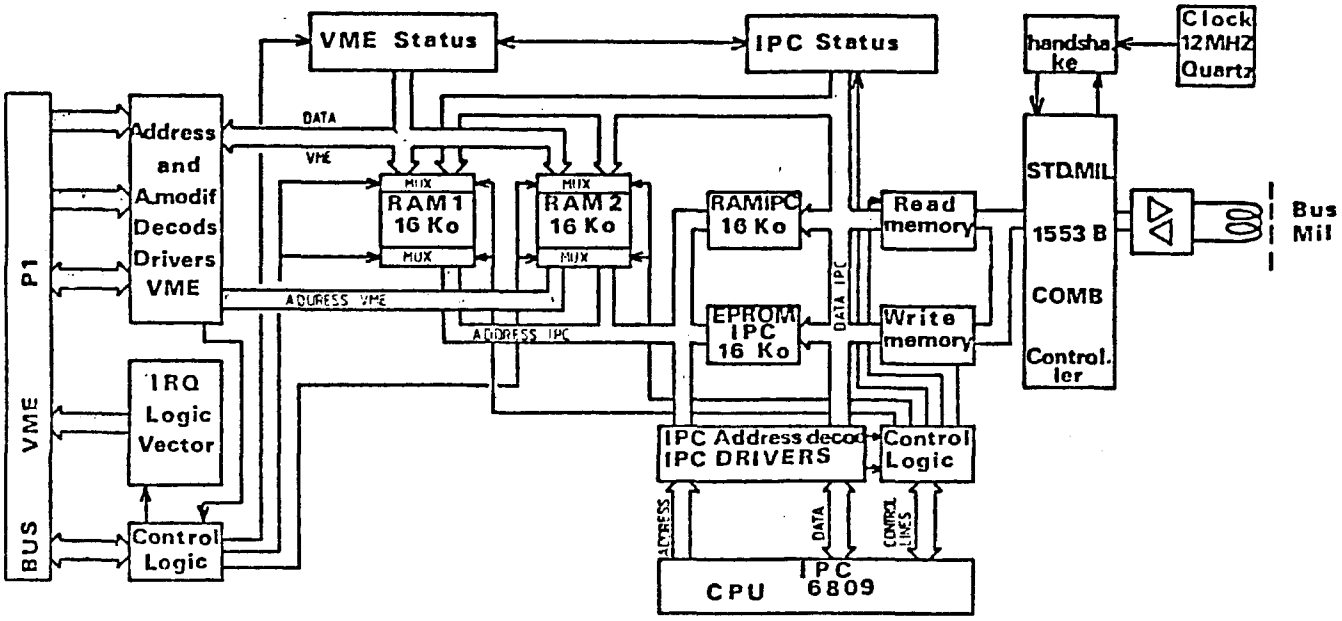


Fig.2 - MIL STD-1553B Bus Controller Block Diagram

SOFTWARE SYSTEM

A proportion of 80% of the old system software was written in three different assembly languages and therefore must be completely

rewritten. An high level language was needed to rewrite rapidly all the software developed during 8 years and FORTRAN was chosen because it is widely used and understood in our Laboratory. Furthermore many of its disadvantages against more structured languages have vanished with the 77 versions.

The ease to write and maintain the programs with a high level language is paid by their least efficiency : the response times are increased by a factor from 4 to 6 regarding our previous system. Improvements are expected with the new generation of VME boards and also with the availability of a better FORTRAN compiler.

The software developments are done with an EXORmacs micro-computer from MOTOROLA. It supports easily 4 to 6 simultaneous users with 1 Mbyte of memory.

3 layers can be distinguished in the control system software:

Operating system

The EXORmacs and HOST station run the same VERSAdos disk-based O.S. The LINA and OPER stations use the RMS68k real-time kernel. VERSAdos is built around this kernel. This feature is very convenient : the programs are almost completely debugged on the development system before their final integration on the target systems.

Service software

This level concerns all the general purpose services which are furnished to the high level application software.

Intertask Communication : A message system has been developped which handles in the same way local messages in one station as well as messages exchanged between two different stations. The receiver of a message is addressed with a symbolic name :

name = STAT, SSTA, TASK

where STAT is the name of the VME station
SSTA will be used to address a substation connected to the 1553B network
TASK is the name of the receiving task in a VME station or an EUROMAK substation.

The management of the transfers can be modified by attributes specified by the receiving task:

- the receiving task can be blocked until a message is sent to it (default option) .
- the receiving task can test if there is a pending message for it.
- the receiving task can receive only messages of selected sending tasks,and in this case, discard or delay the other messages.

Data base management system : A distributed data base contains all the informations concerning the accelerator parameters. The data base management system was strongly influenced by the system used at SLAC for the SLC control system [5,6]. The accesses to the data base are done through standard functions with symbolic parameters. The data base is structured in subsets belonging to each station. The HOST station maintains a copy of the whole data base. Each subset is structured in blocks, the first block contains all the pointers necessary to access the data contained in the other blocks. In the four other blocks, the data are structured regarding their types : stable parameters, HOST write only parameters, HOST read only parameters, HOST only parameters. A Fortran-like format with or without a repeat specification is associated with each piece of information : integer 16 or 32 bits, real, character string.

Man-Machine communication facilities : A graphics support has been provided to display informations on the console screens. The output commands (draw a set of vectors, display a string of characters, fill an area) are passed to the graphics software and are stored in device independent graphic segments. Another set of commands is used to manipulate the segments (initialization, deletion, drawing, erasing, windowing) and to send them to specific device drivers.

The operator commands are communicated to the system through the use of touch panels and their operation are defined by symbolic files compiled by a specific program. This compiler allows the specifications of the location of a button, its label, and the actions to be taken when it is touched : e.g. the command:

```
COMMAND:05,07,     ,RUN SURVEILLANCE,SURV,RUN-SURV
```

generates a button at location 5,7 with the label RUN SURVEILLANCE and when it is touched, the message "RUN-SURV" will be sent to the task SURV. The command:

```
MENU   :05,08,   M,INDEX            ,IDLE,MENUINDX
```

generates a button at location 5,8 with the label "INDEX" in a medium size and, when it is pushed, the menu called "MENUINDX" will replace the current one.

Error handling facility

Based on the existing VERSAdos Error Messages Handler, a facility is provided to edit the warning or error messages in a standard way on the terminal attached to the HOST station. The messages appear always with the same format : date, originating task , message (with or without variable fields) - For example :

```
05/07  9h30m LINA-....-LACQ CAMAC ERROR IN CSSA  C=3  N=10  A=0  F=0
05/07 10h15m HOST-....-DBEX Protect code error (+4)
```

The most recent 200 messages are also logged on the disk for later analysis

Application software

The LINA station runs all the acquisition and control software under the control of interactive programs in the HOST station. It does also autonomous cyclic acquisition and surveillance. The interactive tasks running in the HOST station are disk resident except two tasks which display, on two dedicated screens, permanent informations about the accelerator status.

All the basic tasks available on the old system are now implemented.

CONCLUSION

According to the schedule, the system described here is operational since September 1985. Less than one month after the system start-up, it is obvious that some unavoidable youth problems are still present (minors bugs in the network software and in some application tasks, poor quality of our VME power supplies). Also as stated before the time responses must be decreased to improve the operator comfort. Nevertheless, with more than one year of experience in using VME microcomputers we can draw some conclusions:

. The choice of the VME standard, 2 years ago, was relevant : the number of vendors is extremely large and the module available are more and more performant, but the software furnished (compilers, utilities, system generation tools) is not always flawless or sufficient.

. The number of hardware failures is drastically decreased compared to our old technology system.

. The use of FORTRAN and of the software services mentioned above have been very successful to write efficiently the appplication programs.

. The hardware and software distributed architecture gives the system a great flexibility. This point will be particularly valuable for the extension of the control system to the ALSII project [7].

REFERENCES

[1] H.LEBOUTET et al., "First operation of the high duty cycle Saclay electron linac (ALS)", Proc. of the 1969 Particle Accelerator Conference Washington, Trans. Nucl., NS16, n°3 (1969) 299.

[2] G. BIANCHI et al., "A flexible dialogue with the computer in the control room of the Saclay's linac", Proc. of the 1976 Proton Linear Accelerator Conference, ChalkRiver (1976) 369.

[3] J.F. GOURNAY, "Logiciel du système de controle de l'Accélérateur Linéaire de Saclay" déc. 1979, Internal report 79-428.

[4] M.C. CROWLEY-MILLING,"Distributed Digital control of accelerators" Proc. of the 1983 Computing in Accelerator Design and Operation Conference, Berlin, Springer-Verlag (1984) 278.

[5] R.E. MELEN, "Design and performance of the Stanford Linear collider control system", Proc of the 1984 Nuclear Science Symposium, Orlando, Trans. Nucl. Sci., NS-32, n°1 (1985) 230.

[6] M. BREIDENBACH, N. PHINNEY, Private communications.

[7] B. AUNE et al., "ALS II : The Saclay proposal for a 2 GeV, CW electron facility" Proc. of 1983 Particle Accelerator Conference, Santa Fe, Trans Nuc. Sci. NS-30, n°4 (1983) 3296.

INTEGRATION OF VMEbus MODULES

IN A SM90 MINICOMPUTER RUNNING UNIX ON A M68000

G. Fontaine, L. Guglielmi
Laboratoire de Physique Corpusculaire
Collège de France, Paris

ABSTRACT

SM90 is a multiple bus, multiprocessor structure commercially available in France.

A simple adaptator has been developped to allow the use of slave VMEbus cards on local buses. VMEbus standard CAMAC branch driver and intelligent graphic controller have already been used and easily integrated into the software.

More complex applications can be tackled by a dual port memory module linking the VMEbus to the SM global bus.

SM90 native compilers (C and F77) have been extended and merged with CERN cross-software to produce object code running under MoniCa in the satellite VMEbus crates.

SM90

The SM90 is a multiple bus, multiprocessor structure developped in 1981 by CNET for communication needs. It can run under various operating systems, most of them developped by CNET and INRIA. The machine is now commercially available from TELMAT and BULL-SEMS and we are using it since 1983 (ref. 1).

Its structure is based on two buses : the global bus is a multimaster system bus devoted to inter-processor communications,

---

CNET : Centre National d'Etude des Télécommunications

INRIA : Institut National de Recherche en Informatique et Automatique

UNIX is a trademark of Bell Laboratories.

while the local bus is used by each CPU for fast access to its local own ressources such as memory or specific interfaces (see fig. 1).

This structure anticipated the VME/VMX organisation, but did not use it, since the VME/VMX standard did not exist at SM90 design time. However, this machine uses double euro-mechanics which helps for adapting VMEbus cards.

The processor boards use the MC680XX family : MC68000, MC68010 and soon MC68020, with a CNET designed MMU (Memory Management Unit) and optional FPU (Floating Point Unit). A lot of intelligent I/O controllers connected to the global bus are now available : SASI, Serial I/O, Ethernet, X25, magnetic tapes, graphics... The operating system running on our machine is SMX 4.3, a multiprocessor implementation of UNIX V7, with additions from Berkeley 4.2.

Local Bus - VME Adaptator

The local bus, a single master bus, is in fact a subset of the MC68000 bus, and as well a subset of the VMEbus. We have developped a very simple hardware adaptator, which allows the use of slave VME cards on the local bus. The main goal was to use the Data-Sud VME-Camac interface on SM90, but other kinds of VME cards can be, and have been, used, such as a graphic card or simple memory cards.

For the I/O cards, such as Camac interface, the implementation can be very easy, just by mapping physical addresses of the card into the process logical addresses. A system call provides this facility. This method does not need a sophisticated driver, but does not provide interrupt possibilities. We have implemented in this way a subset of the ESONE Camac routines, written in C language. It gives a block data transfer rate of about 20 µs per Camac word, not so bad for a high level language implementation under UNIX.

In addition to the Camac Interface, we have also installed a VME Data-Sud 512 x 512, 8 color graphic card (DSSE512-chroma8) accessible through GKS.

If one needs to deal with interrupt capabilities, it is necessary to write a specific driver, and to link it into the operating system. This driver has then to deal with the single level of auto-vectored interrupts, provided on the local bus.

Global Bus - VME Interface

A more powerful interface has been implemented by INRIA and the french firm DIGITONE, enabling the dialogue between the SM90 and a full VME system. A dual port memory is connected on one side to the global bus (or system bus) of the SM90, and on the other side to the VMEbus. The memory, with a capacity of 256 kbytes or 1 Mbyte is located in the SM90 chassis. Twisted pair cables link the card to the adaptator located in the VME crate.

Writing at the first address of the card, by any SM90 master, triggers a vectorized interrupt on the VMEbus. The vector number is software selected and the interrupt level is selected by straps. In a similar way, writing at the same address by a VME master, triggers an interrupt on the SM90 global bus.

A UNIX driver for that card has been written by INRIA, following the SM90 dialogue conventions between masters and slaves. At Collège de France, we are working on VME drivers for this link, running under MoniCa or OS-9.

This makes a full VME system act as a SM90 standard I/O processor, with the full power and versatility of the VMEbus. The VME crate can then be used for data-acquisition and prefiltering in a real time environment. Data, in this case are transmitted via the fast interface to the SM90 where they can be analysed, displayed and stored on the SM90 mass storage.

Software Development

Programs to be run in a VME processor can be developped on the SM90 and quickly downloaded into the VME crate.

We have already installed on the SM90 most of the CERN's cross software :

- CUFOM processors (linker, pusher)
- M68MIL assembler
- SIEMENS PASCAL compiler.

In addition, SM90 native compilers (C and F77) have been extended and merged with CERN cross software. They can now produce CUFOM output and the generated code can be run under the control of MoniCa.

A PILS Host Interface can be implemented through the SM90-VME Interface, enabling high speed file transfers between PILS, running in the VME and the SM90.

Many other utilities are available on the SM90 :

- CERN's M680X assemblers
- Prom and Pal programmers accessible via the local bus Camac Interface
- PATCHY
- Most of the CERNLIB
- GKS
- ...

Conclusion

These two VME interfaces enhance the power of the SM90 through real-time data acquisition possibilities, and make it a very suitable work-station for electronic tests and small data acquisition systems.

* * *

Reference

1 "Stations de travail intégrées sous UNIX pour le développement de logiciels et le test d'équipements" (LPC 83-35), and in proceeding of "Forum sur la micro-informatique en physique nucléaire et physique des particules" Paris 1983 (LPC 83-36).

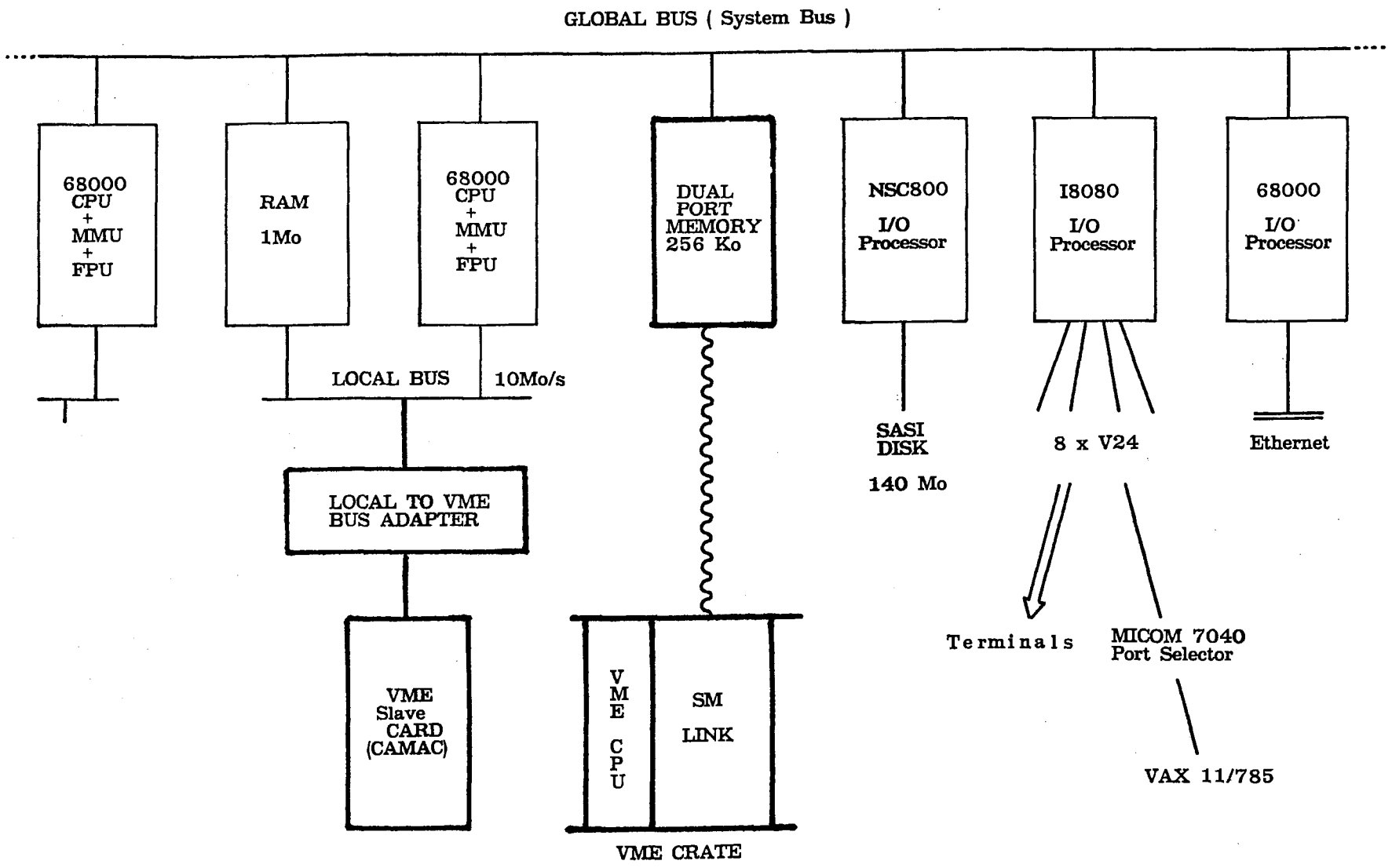


Fig 1. S M 90 - V M E Links

USE OF OS9/68K - A UNIX LIKE, REAL TIME OPERATING SYSTEM - IN PARTICLE PHYSICS VMEbus APPLICATIONS

G. Fontaine, L. Guglielmi
Laboratoire de Physique Corpusculaire
Collège de France, Paris

ABSTRACT

OS-9/68K is an industrial operating system for real-time multitasking applications of M68000 processors.

It is a modern, UNIX-like, system offering most of the high level facilities needed for software development, and gaining efficiency through an assembly coded kernel and an original memory management structure.

OS-9/68K has rich real-time possibilities due to the simplicity and efficiency of inter-process communication schemes. It is very modular and can be easily configured, extended or ported to almost any M68000 system.

This presentation will be based on our experience of system transport to several VMEbus configurations, and on the use of a similar OS-9/6809 system in a CERN test beam for CAMAC data acquisition.

OS-9/68K is the MC680XX version of the OS-9 operating system originally developped for the MC6809 microprocessor as a joint effort by Microware and Motorola. Since its introduction in 1979, OS-9 has become one of the most popular systems in industrial automation and communication applications. It is used by our laboratory to provide testing tools for high energy physics equipment, and has been and will be used in some experiments. This paper describes briefly the way in which these application requirements are well matched to OS-9 features and how OS-9's

---

versatility makes it the present best choice for small and medium size real-time systems.

A UNIX-Like System

Unix, now established as a standard, has proved to be a very effective operating system for the development of computer software.

OS-9 provides a Unix-style environment, with essentially all the functions which make the programmer's job a lot easier. Like Unix, it is a multi-tasking, multi-user operating system featuring a unified device independant I/O scheme. It also provides redirection mechanisms and pipes allowing the use of filter tools. The hierarchical file management system too is very similar to Unix with tree-structured directories, and the human interface is done via a replaceable independant task whose usual implementation is the Shell.

However, OS-9 also has some important differences from Unix (fig. 1). In addition of being a real time system with more modularity (see below), it requires much fewer hardware ressources : OS-9 can be effectively used on systems with medium size memory, small disks or low cost floppy disks, or even on systems without any mass storage at all (such as ROM-based control systems). It offers additional features such as file locking and record locking, and eliminates the fearsome disk recovery problems by using a more reliable "crash proof" disk management technique. Finally, OS-9 runs faster since the assembly coded kernel is specific to the MC6809 or MC680XX, and its high modularity makes it very easy to port to new hardware.

A Real-Time System

The OS-9 kernel provides a comprehensive assortment of multitasking functions, including creation (via a fork mechanism), scheduling (time slicing for active, sleeping or waiting tasks) and prioritization.

Task synchronisation can be done on the real-time clock or on external events causing interrupts. Inter-task communica-

tions are provided by several mechanisms : signals, semaphores, pipes or shared memory modules.

Real-time performance is achieved by keeping all tasks resident (no swapping) and by an assembly-coded kernel. An interrupt can be serviced in less than 100 µs (measured on a 8MHz CPU with 350 ns access time memory). OS-9 can be ROMed for high security control systems or for small single board controllers.

An Open System

Versatility is achieved by a unique modular design and support for sophisticated modular software techniques. OS-9 programs (as well as OS-9 itself) are built up from groups of memory modules. These are named objects having a standard format. They can contain program code, tables, subroutine packages or any other kind of data. Code modules contain position independant code and use separate data areas allocated at run time by the system. Code modules are thus ROMable. All modules are under control of the system kernel using a module directory for memory space management. Modules can locate each other by means of system calls and can be shared simultaneously by several tasks.

This technique as well as reentrant coding, are supported by all languages under OS-9 and provide a very efficient use of memory space. It allows complex software (system or application) to be broken into smaller, more manageable modules that can be individually written, tested and maintained. For instance, it is possible to dynamically replace an existing module, even in ROM, by an updated version, located elsewhere, bearing a higher revision level in its header.

The operating system itself is also modular and is broken into a dozen memory modules (fig. 2). It can thus be easily configured or customized by the user without any need to access the system source code.

Adding support for a new I/O device may require only the creation of a new table module (device descriptor), or in the worst case the writing of a driver module. Both can be installed without a new system generation and without stopping the system, by a simple load into memory, OS-9 hooking them automatically in its tables while it is running.

Programming Languages

OS-9 is supported by a relocatable assembler and a series of high level language compilers providing reentrant and position independant ROMable code in memory module format. All languages use a common IEEE standard floating point library making easier the integration of a floating point coprocessor.

The macro assembler is Motorola compatible and supports OS-9 system calls.

The C compiler is a complete implementation of the Kernighan and Ritchie specifications and is compatible with application software written in the C language for Unix System V and BSD 4.2 Unix. (The C library supports both OS-9 and Unix system calls).

Basic09 is much more than a Basic ; it is a structured language providing Pascal-type loop constructs and typed data structures. It is procedure oriented (with argument passing) and supports the OS-9 modular software environment. Being compiled, it runs rather fast, while maintaining interactive symbolic debug facilities.

Pascal follows the ISO 7185.1 standard and includes a large number of extensions. Two different compilers are now available, one from Microware, the other from Omegasoft.

A Fortran-77 compiler is announced to be available by the end of 1985.

For graphic applications, a VDI (Virtual Device Interface) standard binding package is available for C and Basic09. A GKS library (level 2B), implemented in C, is in progress.

Versions

OS-9 exists in two flavours :
- Level 1 with software memory management,
- Level 2 for systems with hardware memory management.

MC6809 versions (Levels 1 and 2) have been in use for several years in industry.

MC68000 Level 1 version was released in mid'84, it is regularly updated, and Level 2 should be available now. A MC68020 version is announced for the end of 1985.

A networked version is also under development for both MC6809 and MC680XX series. It incorporates, in the modular OS-9 structure, an hardware independant Network Manager associated with drivers for various links such as Ethernet, X25, RS232C, GPIB...

Use of OS-9

Our laboratory has been and is still doing software development on a Unix computer for various microprocessor targets including VME systems (ref. 1). This kind of environment has been very much appreciated but all these nice facilities were absent at run time since the applications were either too small or real-time oriented, thus excluding Unix from their domain.

OS-9 seemed attractive to provide such a continuation with the required flexibility and performance.

Our first exercise has been to write a disk driver for a 6809 version running on a Motorola Exorset 165. This being completed we added a Camac branch interface and a graphic controller. It was then possible in a few days to write and debug an acquisition and monitoring program which included histogramming and graphical data presentation, for a real test of a particle shower position detector in test beam X5 at CERN. The system was easily prepared and installed for acquisition of several LRS 2280 48 channel ADC. It worked as expected, giving us confidence in OS-9 qualities.

A similar system is now prepared for a control application. It will provide user friendly monitoring of a high voltage system in a $\bar{p}$ p collider experiment.

For OS-9/68K on VME hardware, we also started with a commercial system (Eltec Eurocom 3) and were soon followed by two other groups of our lab buying the same system. Given the wide variety of applications considered (including the data acquisition for a neutrino oscillation experiment) we did not want to stay dependant on a single hardware vendor, and thus needed to be able to transport the OS-9 system to any mixed configuration.

This transport is best done with a port-pack license, available at reasonable cost from Microware. It includes the few modules that have to be adapted (Boot module, clock driver, terminal driver, disc driver) in source code for various popular hardware items. Software tools are also provided to help starting the process, and it is quite easy to modify the supplied modules to tailor them to specific needs.

This work has been done for various VME cards that can now be used in any combination. These include four CPU cards :

- Thomson EFD-CPU1
- Motorola MVME 101 and MVME 110
- Data-Sud DSSECPUA1 (in progress)

and two disk interfaces :

- Data-Sud DSSEFDCONT (floppies)
- Motorola MVME 315 (floppies + SASI hard disk, in progress).

This choice, together with the three Eltec VME cards Eurocom 3 will enable the physics groups involved to choose the configuration that best fits their needs for OS-9/68K applications.

Acknowledgements

The authors wish to thank Mr M. Vignes and P. Wolstenholme from CERN for very fruitful discussions concerning OS-9 system.

* * *

Reference

1 "Stations de travail intégrées sous UNIX pour le développement de logiciels et le test d'équipements" (LPC 83-35), and in proceeding of "Forum sur la micro-informatique en physique nucléaire et physique des particules" Paris 1983 (LPC 83-36).

FEATURE	OS-9/68K	UNIX	CP/M-68K
General Features			
Multitasking	Yes	Yes	No
Multiuser	Yes	Yes	No
Kernel Size	24K	96K	24K
Minimum RAM	64k	256K	64K
Minimum Disk	None	20 Megabyte	512K
Source Language	Assem. and C	C	C
ROMable	Yes	No	No
Real-time functions	Yes	No	No
Modular Memory Management	Yes	No	No
User Configurability	Easy	Difficult	Easy
Disk File System			
Multilevel Directories	Yes	Yes	No
Sequential Files	Yes	Yes	Yes
Random Files	Yes	Yes	Yes
Max. Record Size	None	None	256
Record Locking	Yes	No	No
File Security	Yes	Yes	No
I/O System			
Max. Number Terminals	No Limit	No Limit	1
Max. Number Printers	No Limit	No Limit	2
Max. Number Disks	No Limit	No Limit	16
User-written drivers	simple	complicated	simple
Interrupt-driven I/O	Yes	Yes	No
Print Spooling	Yes	Yes	No
I/O Redirection	Yes	Yes	No

Fig.1. Comparison of OS-9/68K features with those of some popular operating systems.

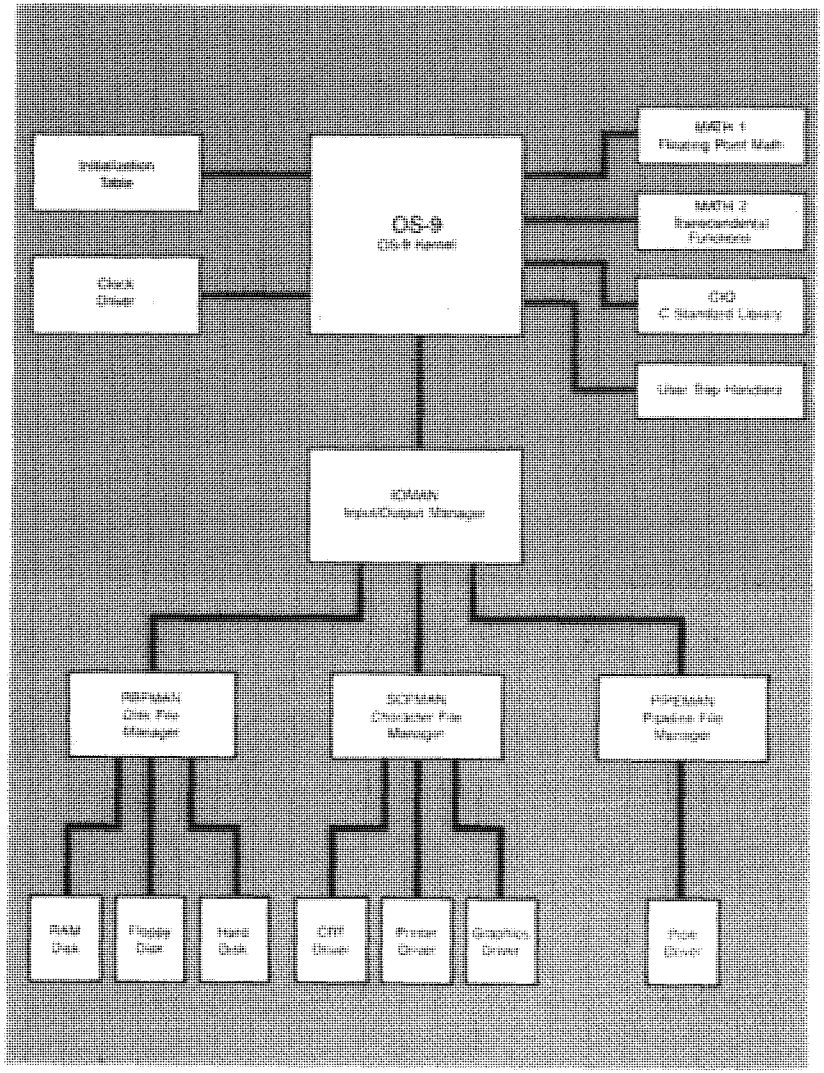


Fig.2. Modular structure of OS-9/68K operating system.

A 1 Mbit/s Communication Interface for a VMEbus System

P. Heimann
Max-Planck-Institut für Plasmaphysik (IPP),
EURATOM Association, D-8046 Garching

The communication interface presented in this paper is built around the VLSI device WD2511 manufactured by Western Digital Corporation /1/. It handles bit oriented, full duplex data communication which conforms to CCITT X.25 level 2 LAPB (link access protocol balanced). Those protocols include zero bit insertion and deletion, automatic appending and testing of frame check sequences (CRC) and automatic appending of address and control fields for level 2 protocols. The device also contains two direct memory access (DMA) channels, one for transmit and one for receive, to gain access to data buffers. Serial transmission rates of up to 1.1 Mbit/s in full duplex operation can be obtained.

16 on-chip I/O registers are used to control and monitor the operation of the controller. Two control registers provide means to initiate link set up, set the receiver to ready condition and start sending data. Three status registers contain fields for the transmit and receive counters, link status information, and interrupt causing state. Severe changes in link state or failures on the link level are indicated in the error register. Other registers are used to set the timer and retransmission counter, load the address field, and set the pointer to the cyclic buffer management queue.

Transmit data is accessed by the WD2511's DMA channels through cyclic buffer queues, one for transmit (TLOOK) and one for receive (RLOOK). Both queues consist of 8 elements containing buffer addresses and transmission counts.

At a serial transmission speed of 1 Mbit/s (and neglecting zero bit insertion/deletion) the controller would access a new data byte every 8 microsecond for either transmit and receive. This would introduce a heavy load on the computer bus, especially when several DMA devices are present on the same bus. To avoid this problem a dual port memory is provided on the board to separate the computer bus and the serial transmission operation. Communication between CPU and serial transmission line is therefore only possible via this dual port memory by moving messages between the main computing memory and the on-board memory.

The block diagram of the controller board is shown in figure 1. It can be devided into several functional modules. The VMEbus interface provides access to the computer bus signals. Only those signals are used that are necessary for slave data transfer operation /2/. Service requests to the CPU are initiated by the interrupt requester. All 7 interrupt levels may be selected by jumper option. Interrupt acknowledge cycles are answered by the slave device with the assertion of an interrupt vector that is also settable by jumpers.

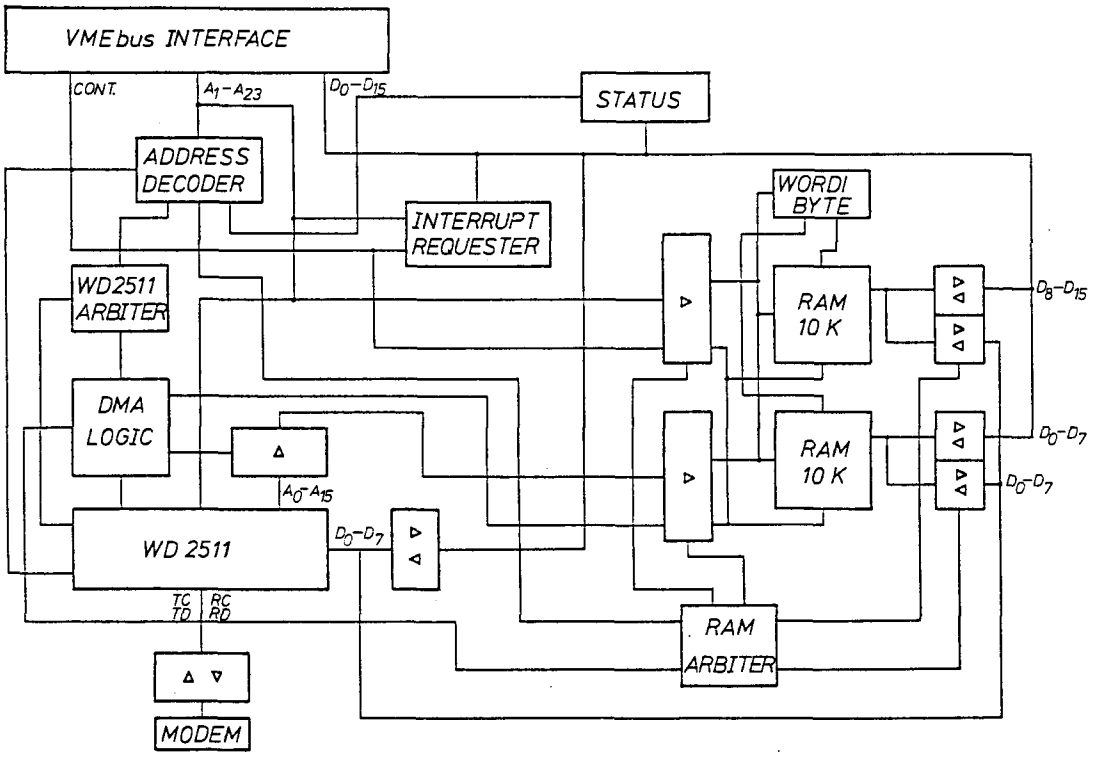


Figure 1: Block diagram of the communication interface

Access to the registers of the WD2511 and the 20 kByte on-board memory is accomplished through two dual port controllers. They arbitrate requests from the VMEbus and the DMA controllers of the WD2511 on a first come first served basis. A total of 32 kBytes address space on the VMEbus (the start address is jumper selectable) is necessary to operate the board. The status register is used for modem control and modem status information /3/.

Programming of the board is simple. After initially setting the registers of the WD2511 and establishing the TLOOK and RLOOK queues the receiver is set ready and link set up is initiated. Data to send is moved to a free transmit buffer and the corresponding entry in TLOOK is made ready. Then the WD2511 is commanded to send the packet. After successful transmission a block acknowledge interrupt is generated and the TLOOK entry is freed again. Received data packets are stored into a free buffer from RLOOK and a received packet interrupt is generated.

An application example of the controller board is shown in Figure 2. In a data acquisition system /4/, where large amounts of data have to be pro-

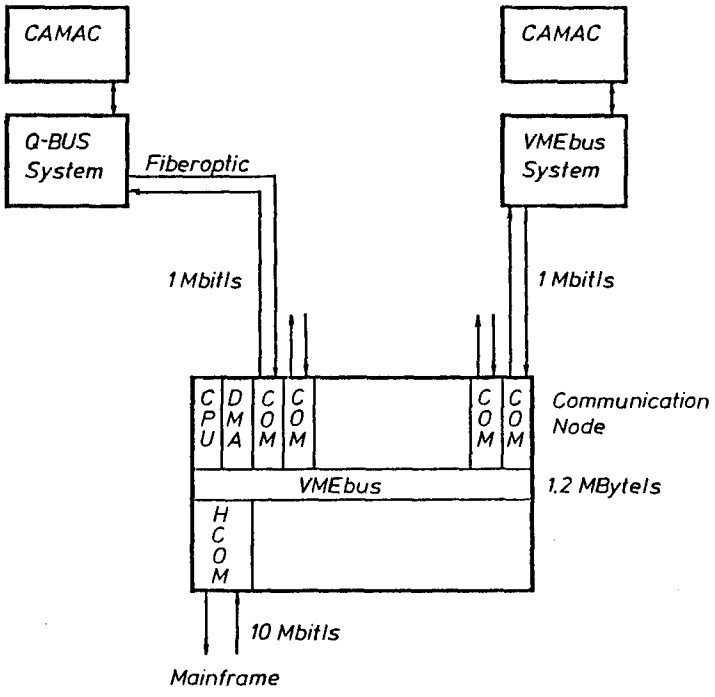


Figure 2: High bandwidth communication node in a data acquisition system

cessed in short time intervals, a mainframe computer is used for data analysis. The communication net connecting the various components of the system has to transfer the measured data as fast as possible to satisfy the user requirement of a sufficiently short response time.

The front end computers, that may be Q-Bus or VMEbus systems (both controller board versions are available), sample data from CAMAC crates. The preprocessed data are then transferred to a communication node via a fiber optics link built with the 1 Mbit/s communication controller. Incoming data packets interrupt the CPU. According to the routing information contained in the packets, data is transferred by the DMA controller (maximum transfer rate 1.2 MByte/s) from the receiver memory to the memory of the destination device. When data is sent to the mainframe computer the destination device is a 10Mbit/s communication board. In this example a throughput of 1 MByte/s may be achieved in the communication node.

* * *

References:

/1/ WD2511 X.25 Packet Network Interface (LAPB), data sheet, Western Digital Corporation

/2/ VMEbus specification manual, VMEbus Manufacturers Group, Rev. B, Aug. 1982

/3/ P. Heimann, A 1 MBit/s communication interface for VME/Q-Bus systems, AMOS development note D105.0, Max-Planck-Institut für Plasmaphysik, Informatik

/4/ F. Hertweck, AMOS/D the data acquisition system for ASDEX-Upgrade, AMOS development note D100.0, Max-Planck-Institut für Plasmaphysik, Informatik

Problems and Chances of Real-Time Data Processing in UNIX-like Operating Systems

Peter Buchheim / Johannes Schacht

Institut fuer Angewandte Informatik
Technische Universitaet Berlin /
Electronic Modular Systems

ABSTRACT

In our age of automation and data communication real-time data processing has encreased considerably. Formerly users implemented their application on naked hardware. Today they demand real-time operating systems so that safer and quicker development becomes possible - which of course implies a decrease of expenses. Following this trend a UNIX-like operating system with real-time features seems to stand a fair chance of selling extremely well.

There are various ways of trying to develop such an operating system. This essay describes the difficulties and introduces the different methods of approach. Afterwards one system is described more closely.

1. Introduction

1969 UNIX (TM)* was developed as an operating system in Bell Laboratories. Because of its clear structure which proved to be of great advantage to the developers of programs, UNIX circulated quickly in Bell Laboratories, and soon spread to the universities.

However, it was not before the end of the Seventies that UNIX was really wide-spread - after interested firms and then AT&T itself had distributed it with the necessary support. This development was encouraged by the decision of leading software firms to use UNIX as standard development system.

---

* UNIX is a trade mark of Bell Laboratories

UNIX was not planned as a real-time operating system. The developers of UNIX probably never thought of using their operating system in such a way. The wish to be able to develop and operate real-time systems under UNIX rather originated with thrilled UNIX-users who did not want to miss the valuable development environment when working on real-time tasks. The wish can be explained by the fact that there is no real-time operating system available on the market that combines any of the advantages of UNIX, as for instance portability, powerfulness and wide circulation.

From a technical standpoint it is difficult to give reasons for equipping UNIX with real-time features. Seen technically, the best solution would be the development of a new operating system with real-time features. It would be wise to evaluate the experiences made with UNIX and other operating systems during the last 15 years, and make use of them when developing such a new operating system. That would solve the problem of having to be compatible on one level (e.g. the system call level). If the demands of real-time data processing are taken into consideration when planning the new operating system, the realization should give more satisfaction than the supplementary remodeling of an existent operating system. However, this is not the subject of the essay, but I do think these aspects should not be ignored.

2. Real-Time Requirements

The expression 'real time' has been mentioned several times. It is used frequently nowadays. People demand the real-time capability of UNIX. Operating systems with real-time kernels are presented. But what does 'real-time capability' mean? I shall examine the question in this chapter without claiming to undertake an all-round investigation.

2.1. Response Time

DIN 44 300, No.161 defines real-time operation as the operating of a computer system, where programs for processing data are ready for service permanently and the results are at disposal after a certain period of time.

This period of time is also called response time. Put differently, the response time is the time the real-time system needs to react to an event in the real world or in the computer itself.

The response time is determined by the specifications of a system. It may be the result of physical laws or of an optional determination. To be able to guarantee the response time, it is necessary that the user's program runs fast enough and is scheduled as soon as the event has occurred. The operating system is responsible for this. Here we have come upon the first approach for thinking about real-time capability.

A general-purpose real-time operating system should, of course, enable response times which are as short as possible. And there is a further demand: To be able to check whether a system keeps to a certain specification it must be possible to calculate the maximum response time exactly.

2.1.1. Context-Switch Time

The Context-Switch Time is that characteristic of a real-time operating system which is spread most widely. The period of time defined by the expression 'context-switching' is the time between the end of the performance of one, and the beginning of the performance of another process. This period of time is generally not added to one or the other process. It is needed to identify the new process, to safeguard the context of the old process and to generate the context of the new process. Therefore users of a real-time operating system want those periods of time to be as short as possible.

2.1.2. Critical Regions

Critical regions are program parts which must not be interrupted whilst working. Critical regions are unavoidable in programs which compete for mutual resources and/or communicate with one another. An operating system contains critical regions. No switch of context is allowed when such a program part is being executed. Therefore it is important to know the length (period of time) of the longest critical region existing.

2.1.3. Memory Residence

The knowledge about the context-switch time and the length of the longest critical region is not sufficient to guarantee a certain response time. If the process image is not resident in the memory it has to be loaded first. The amount of time needed for such an operation is too large for several applications. Nevertheless loading must not exceed a certain period of time even when dealing with applications which have long response times. It is difficult to calculate the time, especially if one has to load from a disk. The period of time varies amazingly. Its length depends on hardware characteristics, on the position of the heads and on the physical distribution of the process image on the disk. Therefore it seems sensible to ask for real-time operating systems with the ability of keeping the processes memory resident.

2.2. Programming Environment

Real-time systems have to work on various tasks simultaneously. Accordingly they are always implemented as multi-tasking systems. Thus demands arise which a developer of purely sequential programs does not have.

2.2.1. Interprocess Communication and Synchronization

Interprocess Communication is needed for data transfer (e.g. status or measure data) between co-operating processes. Interprocess Communication has to be reliable and should be capable of eliminating error situations, i.e. in case of an error (e.g. awaiting of a message which will never come) a process must not be delayed endlessly.

When competing for mutual resources processes have to be synchronized. When competing for a printer, for instance, a good operating system should handle synchronization. However a general-purpose real-time

operating system cannot manage peripherals which are different from system to system. This task must be handled by the user's programs. The synchronization of processes is another problem for which an interprocess communication is necessary.

2.2.2. Time Management

The operating system must provide a real-time clock and be capable of executing tasks at specified times.

2.3. High Reliability

Real-time operating systems must be extremely reliable. They should be implemented with a tolerance towards errors. After having detected an error they should eliminate it (if it's a temporary error like a transmission error). If elimination should prove impossible (as with a permanent error) the effects of the error should be reduced to a minimum.

Quite a number of errors -especially those detected in connection with the peripheral- can only be handled satisfactory by users' programs. A real-time operating system should be prepared for this possibility.

2.4. Hardware Access

A typical requirement in the programming of real-time systems is the direct access to the hardware. The hardware of conventional operating systems consists of terminals, disks, tapes and the like. For safety reasons the operating system does not allow any of the user processes to access the hardware directly, and serves the user an abstract model of these devices instead. (In UNIX, for example, all devices appear as (special-) files to a user process.) The abstraction is possible, because the devices are more or less similar.

In real-time applications, however, the hardware components that have to be controlled often vary extremely. For that reason the operating system cannot anticipate the structure of the I/O system, and the application process has to be provided with the means to access the hardware directly. This immediately creates a conflict with the safety aspects of an operating system, and is completely unacceptable to a multi-user operating system like UNIX.

3. UNIX as Real-Time Operating System

The ordinary UNIX like version 7, system III or system V cannot be used as a real-time operating system without modification, because the response time cannot be guaranteed. To achieve real-time capability UNIX either has to be modified or run in a different environment.

3.1. Rewriting UNIX

All UNIX processes are the same: They have a variable priority assigned dynamically by the kernel. The user only has little influence on

the priorities via NICE-value. UNIX would have to be modified in a way enabling the user himself to determine priorities.

The UNIX scheduling concept seems very suitable for real-time activities, since context-switching takes place after every interrupt if necessary. There is, however, one exception: There is no context-switching if a process is running in kernel mode when the interrupt occurs.

Processes in UNIX alternate between user mode and (in case of a system call) kernel mode. Kernel mode processes cannot be interrupted unless they call up 'sleep' themselves. Some UNIX system calls are of a considerable length. For that reason it is intolerable for real-time applications that they cannot be interrupted. Hence there must be modifications to enable interrupts of kernel processes but excluding certain critical regions from this operation.

These are the most important alterations of those necessary. Especially the alteration mentioned last represents a severe interference with the kernel. It is impossible to anticipate all consequences straightaway. Thus the anticipation of the expenditure of work becomes very difficult. There is another disadvantage: The whole procedure has to be repeated every time a new version of UNIX is issued.

3.2. Writing UNIX New

The best solution would be the development of a new operating system with real-time features. The final product could not be called UNIX any longer, but it would possess UNIX features on a level specified before (syscall level or command level). Expenditure would be enormous, and the compatibility with later UNIX releases could only be guaranteed at great expense, too. The market has seen some examples of this approach, but they can either not satisfy in a real-time area or in respect to UNIX, because of great compatibility problems.

3.3. UNIX as Real-Time Process

Another possibility would be the implementation of UNIX as a process of a real-time operating system. The real-time processes would then be managed by the real-time operating system directly, and one of the processes scheduled would be UNIX itself as a whole (including the kernel and the user processes). This scheme is similar to IBM's VM operating system where different operating systems (DOS/CMS) work under one supervisor operating system (CP).

If this concept is followed strictly all real-time tasks are separated from UNIX (they don't appear in a 'ps' command), and run at a higher priority than the UNIX processes. The result would be more or less the same as that presented in the next chapter - except for the fact that the real-time processes run on separate processors there.

3.4. The Shifting of Real-Time Processes to a Slave Processor

This method of approach does not alter the UNIX kernel. As usual the kernel runs on a processor. This approach allows an additional processor which has a real-time kernel and works on real-time tasks.

On the hardware level the processors can communicate via interrupts and shared RAM. On the process level UNIX and real-time processes communicate via special file. A special device driver has to be integrated for that purpose.

This method of approach has several advantages:

(-) UNIX is not altered. The system running is the original UNIX. So there are no problems when a new version of UNIX is released.

(-) Real-time processes do not become a burden to UNIX.

(-) If a good real-time kernel is chosen for the second processor all problems created by the rewriting of UNIX can be avoided. Real-time processes run in a most favourable environment.

(-) Experiments (like the testing of real-time software) can be made during the phase of development without running the risk of the whole system crashing, if hardware means have been established to prevent the real-time processor from accessing the rest of the system.

4. A Multi-Processor Example

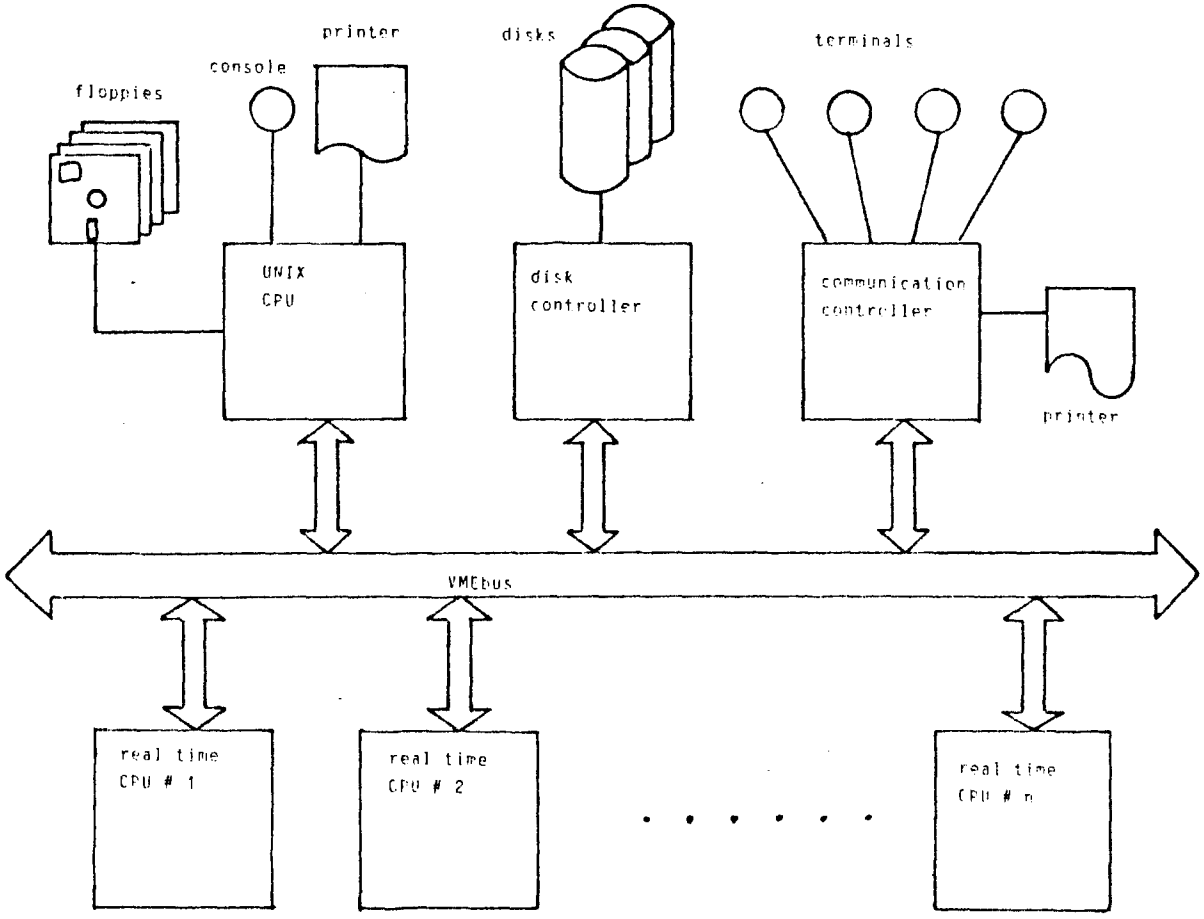
This chapter presents a model of a real-time system implemented by moving the real-time task to specialized processors, and leaving the overall system control and maintenance task under the control of the UNIX operating system. The division of a whole application into two parts (where one part can be implemented to run under a real-time operating system, and the other -which does not need real-time response- can run under UNIX) can often be achieved very easily. As an example take a measuring system: The manual operator can start a measuring task (UNIX), then the system has to sample a huge amount of data in a very short time without data loss (real time), and afterwards the data is evaluated (again UNIX).

4.1. Hardware

The system in question is a VMEbus system. Since you can connect several processors to the VMEbus it is ideal for realizing the project. The UNIX system consists of a CPU board with local RAM, a hard-disk controller and a board for serial and parallel I/O.

To be able to tackle real-time tasks one or more additional CPU-boards are put onto the VMEbus. These CPU- boards are also equipped with a local RAM and a real-time clock.

The real-time boards can handle interrupts as well as receive interrupts in a special way. There is an interrupt register on this kind of board. When this register is written into, an interrupt is generated on the board. This interrupt mechanism is used for the communication between the UNIX- and real-time CPU.



Each real-time CPU can generate an interrupt on the VMEbus too. This mechanism is used when a real-time CPU wants to interrupt the UNIX CPU.

The memory of all CPU boards can be addressed by way of the VMEbus. The UNIX CPU can read and write into the memory of each real-time CPU. Real-time CPUs have similar possibilities, but a certain address space can be blocked for them by a PROM. This is important to protect the UNIX memory from being written over by a real-time process.

4.2. Real-Time Kernel

The system described here uses the real-time kernel pSOS-68K (TM)* from Software Components Group Inc., Santa Clara, to run on the 'real-time CPUs'. pSOS is a multi-tasking operating system kernel. It offers the following facilities:

(-) Process Management

(-) Memory Management

(-) Interprocess Communication

---

* pSOS is a trade mark of Software Components Group

With pSOS the user has an enriched message-passing-model from Hansen at his disposal. Here messages are not bound to processes, but sent per exchanges. Several exchanges may exist. One or more processes can send messages to one exchange or receive them. Generally spoken, this model represents an n-to-m process communication. All typical interprocess communication tasks (sending/receiving data, synchronization and mutual exclusion) can be implemented easily with this scheme.

A process is also associated with seven events. Each process can indicate one or more events by means of signal_v. A process can await one or more events by calling up wait_v.

The event system is not as flexible as the message-passing-system, but easier to handle and more efficient.

(-) Time Management

(-) I/O System

Before talking about the connection between the two (operating) systems a brief description of the pSOS real-time kernel is necessary, since the latter is not so well known as the UNIX operating system.

4.2.1. Process Management

From the pSOS standpoint a process is the smallest program unit which can compete for resources itself. pSOS establishes an environment for processes in which they can use all resources of the system without regard for other processes.

Processes can be created dynamically. Process priorities are determined by the user and can even be changed during execution. Process scheduling is tackled by round-robin procedure, priority basement or both. Processes are sorted in respect to their priority, so that no search has to be undertaken to get the next process to a running state, if a context switch occurs.

4.2.2. Memory Management

pSOS-Memory-Management is realized by three calls: alloc_seg, free_seg and assign_seg. Memory can be called up with alloc_seg, and is distributed in accordance to the first-fit algorithm. Free_seg sets a segment of the memory free, and assign_seg hands over the segment to another process.

4.2.3. Interprocess Communication

Two different mechanisms for interprocess communication are available. There is a **message-passing system** and an **event system**. Strategies for avoiding deadlocks are not part of the pSOS-kernel. They have to be realized by the user.

4.2.4. Time Management

The time management provides the processes with date and time. Moreover, it realizes the timeout times of the system calls, and places an absolute wait call at the systems disposal.

4.2.5. I/O System

The I/O system in the pSOS kernel is a very thin layer. It only serves to define a standardized interface between user processes and device drivers. The structure of the drivers is similar to the structure of UNIX drivers. A driver is selected by a major device number. A minor device number is passed on to the driver by the call. A driver consists of six routines: device_init, device_open, device_close, device_read, device_write and device_control.

5. The UNIX-pSOS Connection

This section describes in detail the connection between the UNIX operating system and the pSOS real-time kernel over the VMEbus. The description is divided into three parts: Part One discusses the various possibilities of connecting operating systems. Part Two gives an idea of how the connection can be used, and the final part discusses implementation details.

5.1. Basic Design of the UNIX-pSOS Connection

Three possibilities of connecting the operating systems have to be considered: A) a connection by means of the interprocess communication facilities of the operating system, B) a connection over the networking facilities of the operating system, and C) a connection of the I/O systems.

5.1.1. ICP-Connection

It seems natural to connect the interprocess communication facilities, because communication takes place between two processes, and because data can be transferred from one process's address space to the other process directly in a tightly coupled system like the connection via a computer bus.

The interprocess communication facility of pSOS is the message exchange. A pSOS process sends and receives messages (4 longs of net data) to/from a certain exchange. Interprocess communication facilities have been established in UNIX recently - in release V. They include three different means of communication: shared memory, semaphores and message passing.

The connection of the interprocess communication facilities of the operating system would have the following effect: When a pSOS process sends data to a certain exchange, the data appears as a message in a certain message buffer of UNIX. This concept has several disadvantages. 1) pSOS uses its message exchanges to solve three different kinds of interprocess activities (data transmission, synchronization and mutual exclusion), whereas UNIX has at least two different kinds of interprocess com-

munications for that purpose (messages and semaphores). 2) The implementation includes a modification of the UNIX kernel to enhance the semantic of the interprocess communication operations, and 3) the interprocess facilities are not used by standard UNIX tools and programs currently. It is not possible for a UNIX utility to redirect its input from a message queue.

5.1.2. Network Connection

The connection of operating systems using the network facilities appears to be an attractive solution, however, pSOS does not offer any network features, and UNIX does not offer a unique network concept. The socket concept in the Berkeley-UNIX environment could be used to implement a pSOS network connection. Such a connection would have the following advantage: The same communication scheme could be used, if the two systems were connected by means other than the VMEbus. However, as the system we are considering includes the standard AT&T-UNIX, we have decided to connect the two operating systems by way of their I/O system.

5.1.3. I/O Connection

The connection of the operating systems over their I/O systems means that each OS takes the other as a peripheral device. In both operating systems peripheral devices are characterized by a major and a minor device number specifying a device type and a unit respectively. In UNIX the device numbers are obtained from the file system as a 'special file'. For the connection of the I/O systems the write operation of a process in one system is interpreted by the OS in a way that a read operation in the other OS gets the data that has been written.

Advantages of this communication scheme are threefold: The I/O system of an OS (UNIX and pSOS are no exceptions) can be modified easily. Modifications of the UNIX I/O system like adding a device driver can be achieved without modifying the kernel, and additionally, it does not require an AT&T source licence - a commercially important point. As all devices are accessed through the UNIX file system, the UNIX file protection mechanism applies to the UNIX-pSOS connection.

Another advantage of the I/O system is the fact that the data is moved by the operating systems. If, for example, the message passing system of pSOS is used, the amount of data sent or received at once is four long words. If large amounts of data are to be moved within pSOS, the information passed normally consists of pointers which point to data themselves. Moving these pointers from one system to the other causes a problem, as local addresses within a system (e.g. a VMEbus board) differ from global addresses (the base address of the VMEbus board itself must be added as an offset). The whole problem disappears, if the I/O system is used for communication, since data is moved directly.

The discussion of the pros and cons of the different approaches may lead to the wrong impression, that the decision for one of them excludes the other possibilities completely. In fact, only the basic interface between the operating systems is affected by this decision. If, for example, it becomes necessary to use the message-system for communication, it can easily be put on top of the I/O system. A pSOS process would send data to

a specialized message exchange which is served by a process that sends the message to the other operating system by way of the I/O system. This implementation is completely transparent to the sending process.

5.2. Using the UNIX-pSOS Connection

Assume, a pSOS process generates ascii data and sends the data to the I/O port connected to the UNIX system. In the UNIX system the data can be received with **cat </dev/pSOS** assuming that /dev/pSOS is installed with the appropriate major/minor device numbers. To bring the data onto the printer the simple command sequence **pr < /dev/pSOS/lpr** is sufficient.

For large-scale control tasks a UNIX program can simply use the UNIX 'open(2)' system call to open the channel to pSOS, and then read from and write data to the pSOS system.

From the pSOS side the communication works much the same. pSOS processes open the device attached to UNIX and execute read/write supervisor calls. It will often be necessary for pSOS processes to have access to resources of the UNIX system. As those resources most often are an integral part of the UNIX file system (data files, printer, terminals, IEEEbus), a simple server process giving access to the UNIX file system is sufficient for a large group of applications. If a pSOS process wants to open a UNIX file, it calls the (library-) function 'u_open', that sends the open request (via UNIX-pSOS connection) to a UNIX process serving this request. The server process executes the 'real' UNIX system call and passes the result back to the pSOS process. Read/write requests are executed in a similar way. More demanding requests from pSOS to UNIX, say queries from a data base, must be served by more specialized server processes.

Another task often required in a multi-processor application is process synchronization which can be achieved directly by the I/O link. A process on one side of the link (either UNIX or pSOS) can wait for an event on the opposite side by simply reading from the I/O link. If there is no data, the process is deferred until a write from the other side occurs.

More sophisticated synchronization tasks include semaphores and mutual exclusion. They can be implemented by a specialized message exchange on the pSOS board. A server process in pSOS allows UNIX processes to send and receive messages to resp. from a pSOS exchange. If multiple pSOS systems are connected to each other, each operating system wanting to access an exchange remotely has to have a remote server process. The remote exchange scheme provides all synchronization means necessary in a multi processor environment.

A last feature of the UNIX-pSOS connection should be described separately: the so-called pROBE channel. pROBE is a debugger with special enhancements to debug pSOS applications. If processes are executed under the control of pROBE, the operator has the possibility to look at process states, message queues and the like. When a UNIX user connects his terminal to the pROBE channel logically, he has complete control over the real-time processes. This feature is especially useful during the test phase of program development for the real-time system, as debugging takes place without having to use a serial channel of the real-time processor.

5.3. Implementation of the UNIX-pSOS Connection

5.3.1. Moving Data

The read/write operations of each operating system are the central action of information transfer. When a process executes a read system call, the events following depend on two circumstances: (a) whether a process on the opposite side already has a write operation (with enough data to satisfy the read request) on the same channel, and (b) whether the channel was opened to operate in blocking or non-blocking mode.

A **read** request in both operating systems specifies the channel to be used (coded by major/minor device numbers), the number of bytes to be read, and the addresses of data location. If sufficient data is available from a previous write on the other side of the channel, the data is transferred to the specified address, and the read operation is terminated. If no (or not sufficient) data is available, further action depends on the channel mode. In non-blocking mode the read system call returns immediately indicating that only part (or none) of the read request could be satisfied. It is then up to the process that executed the read to proceed appropriately (e.g. retry). If the channel operates in blocking mode, process execution is suspended until enough write operations from the other side have occured, so that the read can be satisfied. Afterwards the waiting process is prepared for running again.

The **write** system call operates in a similar way. If a read request on the other side is already pending for the amount to be written, the system call is executed. If no read is pending, the channel mode flag decides, whether the writing process is to be suspended or should return indicating its disability to transfer data.

It should be obvious from the discussion above that successful data transfer is impossible if both sides try to operate the channel in non-blocking mode, as it is very unlikely that both sides execute the system calls at exactly the same moment. The proper set-up of the channel is ensured during the **open** system call to the channel. The open system call returns successfully, if a process on the other side is already waiting for the channel to be opened. In case there should be no process waiting on the other end, the open request either will return without having been successful or will block - depending on the mode specified for the channel (blocking/non-blocking).

If a process at one end of a channel is waiting (due to a read, write or open request), it has to be informed that the other side of the channel has acted. This is done by interrupting the operation at the other end. The interrupt service procedure can then wake up the process waiting. More about interrupts in a later paragraph.

5.3.2. Global Considerations

Assuming the co-existence of a UNIX system and several pSOS boards on one VMEbus, communication should be possible between UNIX and each pSOS board, as well as between all pSOS boards. Since addressing requires a major/minor device number only, it is completely transparent to the com-

municating processes whether they are accessing another pSOS board or the UNIX system. Each system is processed by one VMEbus board. Some boards are configured to have access to other VMEbus boards; others are configured for local resources. For the latter there is only one way of attracting the attention of other systems: by executing a VMEbus interrupt.

If two systems (i.e. two VMEbus boards) want to communicate, at least one of the boards needs global VMEbus access in order to move the data from one system to the other.

The distinction of links and channels has been introduced to manage the different aspects of inter-system communication. A link can be set up between two system boards. Whether it is possible to set up a link to another system depends on: the accessibility of the system (physical availability, system software initialized and running), getting the permission from the other side, and as mentioned above the VMEbus access of at least one of the systems. When establishing the link the decision is made as to which of the two systems is responsible for the actual data transferring.

Max. 16 channels can be operated on each link. A channel is the main communication path between two processes. It can be opened, closed and operated independent from all other channels. Thus several independent communication processes can take place between two systems. A link is established automatically by the first open of one of the channels between two systems. At the same time the place of residence is fixed for the channel descriptors. The system **not** responsible for actual data transfer has to have the channel descriptor inside his local memory. The channel descriptor contains various information e.g. whether a process is waiting for data to arrive.

5.3.3. Initialization

During the powering up of the system the UNIX kernel (or more exactly the initialization procedure of the device driver) reads in the physical characteristics of the VMEbus system from the file system. The information includes the VMEbus addresses of the real-time boards, and information on the capability of a board to access the VMEbus.

With this scheme the addition, removal or reassignment of addresses to boards does not imply the recompilation of either operating system. The only thing necessary is the editing of a UNIX file.

If one of the pSOS operating systems executes its start-up phase (which may happen either before or after UNIX has come up), the communication initialization procedure gives an interrupt to the UNIX system. Some time after its own initialization is completed UNIX is ready for processing the interrupt, and sends the information on the VMEbus configuration to the requesting board. So after all boards have come up, they have enough information to establish links to one another.

5.3.4. Interrupts

The most critical part of UNIX-pSOS connections are the interrupts. It is possible that several processes on a board try to access one or more

VMEbus boards at the same time. However, it may be impossible to execute an interrupt immediately, because the interrupt handler procedure on each board takes quite some time to be completed. For this reason, processes may be deferred until they gain access to their local interrupt resource. Hereby an acknowledge interrupt is automatically introduced with the purpose of waking up sleeping processes.

Another limitation that has to be overcome is the number of interrupt levels on the VMEbus. The VMEbus only has 7 interrupt levels, and some of them could already be being used by the UNIX system for its own resources (terminals, disks etc.). Therefore each real-time board has a local (mailbox) interrupt, which is triggered when local memory location 1 is written to by another VMEbus board. Unfortunately, this mechanism can only be used by processes that have global VMEbus access.

A special interrupter process must be introduced into the system. The process (which can be located in any of the systems) can be interrupted by a VMEbus interrupt, and will then post that VMEbus interrupt to another board as mailbox interrupt.

IBM / VME CHANNEL
HIGH SPEED PARALLEL INTERFACE

B. Merry, A. Moreton, A. Smith

Department of Physics
University of Liverpool
Liverpool
United Kingdom

HARDWARE FEATURES

** HIGH SPEED DATA TRANSFER RATES - 700 K BYTES/SEC ACHIEVED ON A BLOCK MULTIPLEXER CHANNEL.

** SIMPLE PHYSICAL CONNECTION TO CHANNEL VIA STANDARD CONNECTORS, AND RIBBON CABLE TO VMEBUS P2 I/O CONNECTOR.

** THE INTERFACE SUPPORTS ALL STANDARD CHANNEL I/O INSTRUCTIONS; (SIO HALTIO TIO); PLUS RESET AND STACK STATUS CONDITIONS GENERATED BY THE CHANNEL.

** INTERRUPT FACILITIES PROVIDED TO THE CHANNEL ALLOWING INTERRUPT DRIVEN PROGRAMS TO RUN ON THE IBM AND REMOVING THE NEED FOR TIME CONSUMING POLLING.

** 16K DUAL PORTED MEMORY ADDRESSABLE FROM THE VMEBUS AND APPEARING TO THE IBM CHANNEL AS FOUR BYTE SERIALLY ACCESSIBLE BLOCKS OF 4K BYTES.

** COMPLETION STATUS BITS SET AFTER EACH BLOCK HAS BEEN ACCESSED BY THE CHANNEL.

** A MICROCODED STATE MACHINE HANDLES ALL COMMUNICATIONS WITH THE CHANNEL WITHOUT INTERVENTION BY A VME CONTROL PROGRAM.

** SWITCHES SELECT ANY CHANNEL ADDRESS OR VMEBUS BASE ADDRESS.

HARDWARE

GENERAL

THE INTERFACE APPEARS AS A DUAL PORTED MEMORY WHICH IS RANDOMLY ACCESSIBLE BY THE VMEBUS, AND IS BYTE SERIALLY ACCESSIBLE IN FOUR 4K BYTE BLOCKS BY THE IBM CHANNEL.

ALL CHANNEL CONTROL AND DATA-TRANSFER SEQUENCES ARE HANDLED BY A MICROCODED STATE MACHINE (SM). THIS APPROACH MINIMISES CHANNEL OVERHEAD: BY RESPONDING QUICKLY TO CHANNEL SEQUENCES; AND BY REQUIRING NO INTERVENTION FROM A CONTROL PROGRAM.

OPERATION

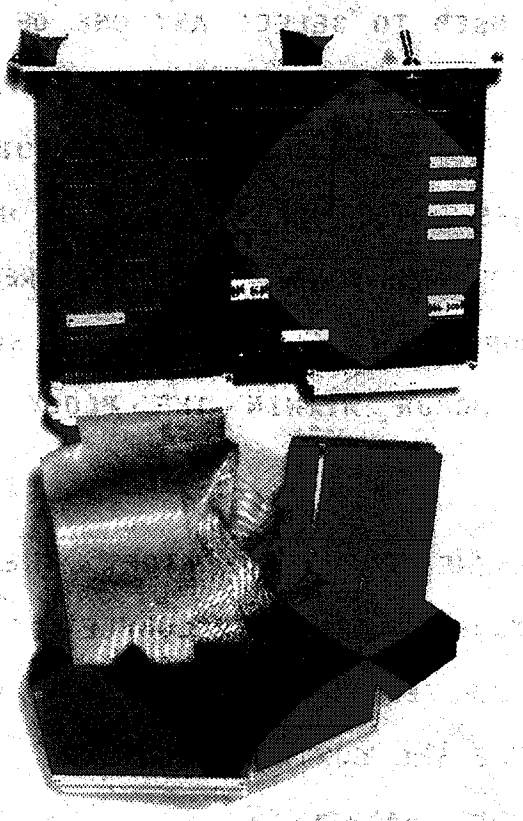
THE SM CONTINUOUSLY MONITORS THE CHANNEL FOR ITS OWN DEVICE ADDRESS. WHEN THIS IS DETECTED IT CAPTURES THE CHANNEL AND WAITS FOR A COMMAND. WHEN A COMMAND IS RECEIVED, IT IS LATCHED AND DECODED TO PRODUCE A VECTOR TO A MICROCODE ROUTINE WHICH HANDLES A PARTICULAR COMMAND. A DATA-TRANSFER-DIRECTION BIT AND A BLOCK-SELECT BIT ARE ALSO LATCHED AT THIS TIME. INITIAL STATUS IS NOW PRESENTED; THIS INFORMS THE CHANNEL THAT THE OPERATION IS COMPLETE, THE DEVICE IS BUSY, OR THAT A DATA TRANSFER IS ABOUT TO START.

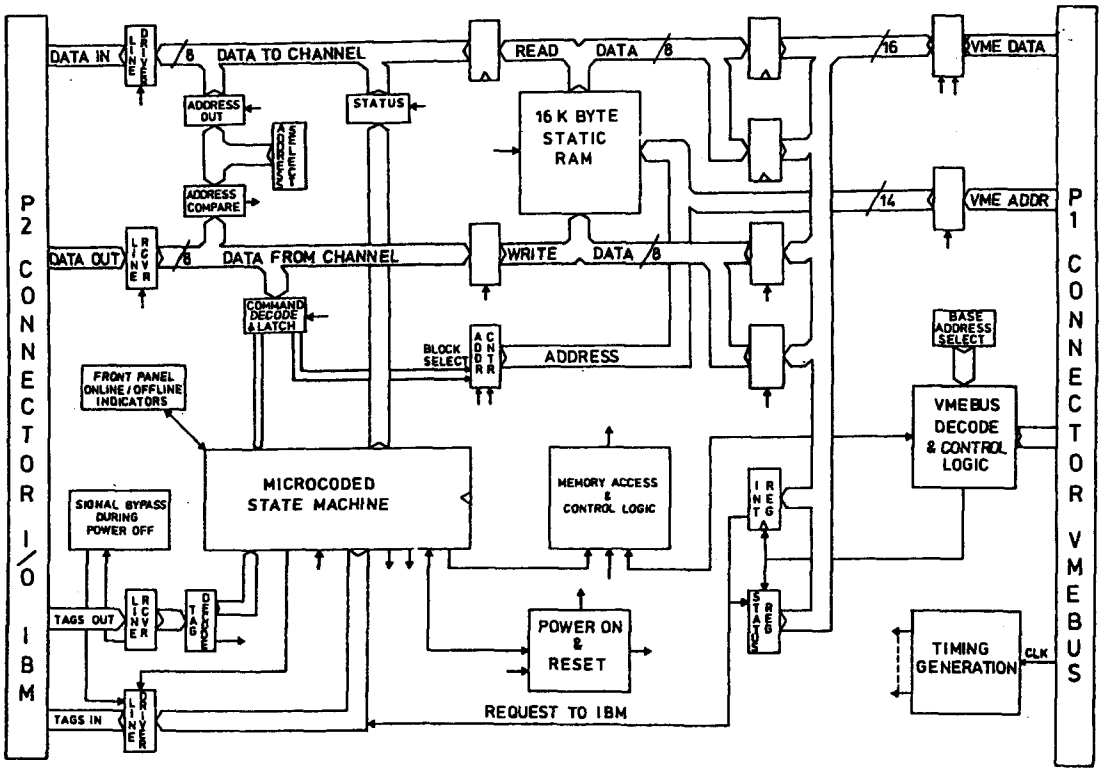
IF THE COMMAND IS ACCEPTED AND IS A READ OR WRITE COMMAND, THE ADDRESS COUNTER IS CLEARED, AND A HANDSHAKING-DATA-TRANSFER SEQUENCE IS STARTED. TWO BITS OF THE COMMAND ARE USED TO SELECT ANY ONE OF FOUR 4K SEGMENTS IN THE MEMORY. AFTER EACH BYTE IS TRANSFERRED, THE ADDRESS COUNTER IS INCREMENTED AND ANOTHER TRANSFER IS REQUESTED. THIS SEQUENCE IS REPEATED UNTIL THE CHANNEL INDICATES THAT IT HAS TRANSFERRED THE REQUIRED NUMBER OF BYTES. IF MORE THAN 4K BYTES ARE SENT OR RECEIVED, WRAPAROUND WILL OCCUR WITHIN THE BLOCK SELECTED BY THE COMMAND.

WHEN THE CHANNEL INDICATES STOP, THE SM STOPS ISSUING TRANSFER REQUESTS; A TRANSFER-COMPLETE-STATUS BIT IS SET FOR THE BLOCK SELECTED BY THE COMMAND; (THIS MAY BE READ AND CLEARED BY THE VME CONTROL PROGRAM); AND ENDING STATUS IS PRESENTED TO THE CHANNEL.

AN INTERRUPT-REQUEST REGISTER (WHICH MAY BE WRITTEN TO, AND MONITORED BY THE VME CONTROL PROGRAM), ALLOWS INTERRUPTS TO BE SENT TO THE CHANNEL. WHEN THE CHANNEL RECEIVES A REQUEST, IT POLLS THE ATTACHED DEVICES. IF THE INTERFACE HAS A PENDING REQUEST WHEN IT DETECTS A POLLING SEQUENCE, THE SM CAPTURES THE CHANNEL AND PRESENTS ITS DEVICE ADDRESS. WHEN THE ADDRESS HAS BEEN ACCEPTED, STATUS IS SENT TO THE CHANNEL. WHEN STATUS IS ACCEPTED, THE REQUEST IS CLEARED.

THE STATUS PRESENTED CONTAINS ATTENTION, DEVICE END, AND ANY ONE OF FOUR POSSIBLE COMBINATIONS OF THE UNIT-EXCEPTION AND STATUS-MODIFIER BITS. THESE BITS ALLOW THE VME CONTROL PROGRAM TO PASS INFORMATION ABOUT MEMORY AVAILABILITY ETC., WITHOUT THE NEED FOR A DATA TRANSFER.





PHYSICAL ATTACHMENT

THE INTERFACE BOARD IS INSTALLED IN A 9 SLOT VME CRATE, AND IS ATTACHED TO THE IBM4331 ON A BLOCK MULTIPLEXOR CHANNEL SHARED BY 2 MEMOREX TAPE DRIVES. THE VMEBUS CRATE ALSO CONTAINS A 68000 SINGLE BOARD COMPUTER, A 6 CHANNEL I/O CARD, AND A GPIB CARD.

SOFTWARE

THE IBM SIDE OF THE INTERFACE IS CONTROLLED BY A PROGRAM RUNNING UNDER CMS ON A DEDICATED VIRTUAL MACHINE. IT PACKS USER DATA INTO A POOL OF 4K BYTE BUFFERS FOR TRANSMISSION TO THE VMEBUS PROGRAM.

THE VMEBUS SOFTWARE RUNS ON THE 68000 SINGLE BOARD COMPUTER. IT TAKES 4K BYTE BUFFERS FROM THE INTERFACE, UNPACKS THEM AND QUEUES THE DATA FOR THE APPROPRIATE DRIVERS. DATA FROM THE DEVICES IS PACKED IN AN IDENTICAL MANNER AND TRANSFERRED TO THE IBM.

OUTPUT BUFFER CLEAR AND INPUT BUFFER FULL SIGNALS ARE PASSED BY ATTENTION INTERRUPTS GENERATED BY THE VMEBUS PROGRAM.

A SET OF FORTRAN CALLABLE ROUTINES ALLOW IBM USERS ACCESS TO DEVICES ON THE VMEBUS THROUGH THE STANDARD IBM COMMUNICATIONS PACKAGE IUCV. THE USER CAN REQUEST THE ATTACHMENT OF A DEVICE AND THEN WRITE TO IT, AND REQUEST INPUT VIA READ PROMPT, AN INPUT QUEUE, OR INTERRUPT CONTROL INTO A SPECIFIED FORTRAN EXIT ROUTINE.

ALTERNATIVELY USERS MAY LOGON TO THE IBM IN LINE EDIT MODE. THE LOGON SOFTWARE USES THE STANDARD IBM PACKAGE LDSF WITH RUTHERFORD LABORATORY MODIFICATIONS FOR LINE EDIT TERMINALS.

THE INTERFACE IS CURRENTLY DRIVING A SIGMA GRAPHICS TERMINAL ON A 38.4K BAUD SERIAL LINE, AND AS CALCOMP PLOTTER ON A 9.6K BAUD LINE. WORK IS IN PROGRESS ON A GP1B DRIVER FOR A MEGATEK GRAPHICS DISPLAY AND A PARALLEL INTERFACE TO A 370/E IBM EMULATOR.

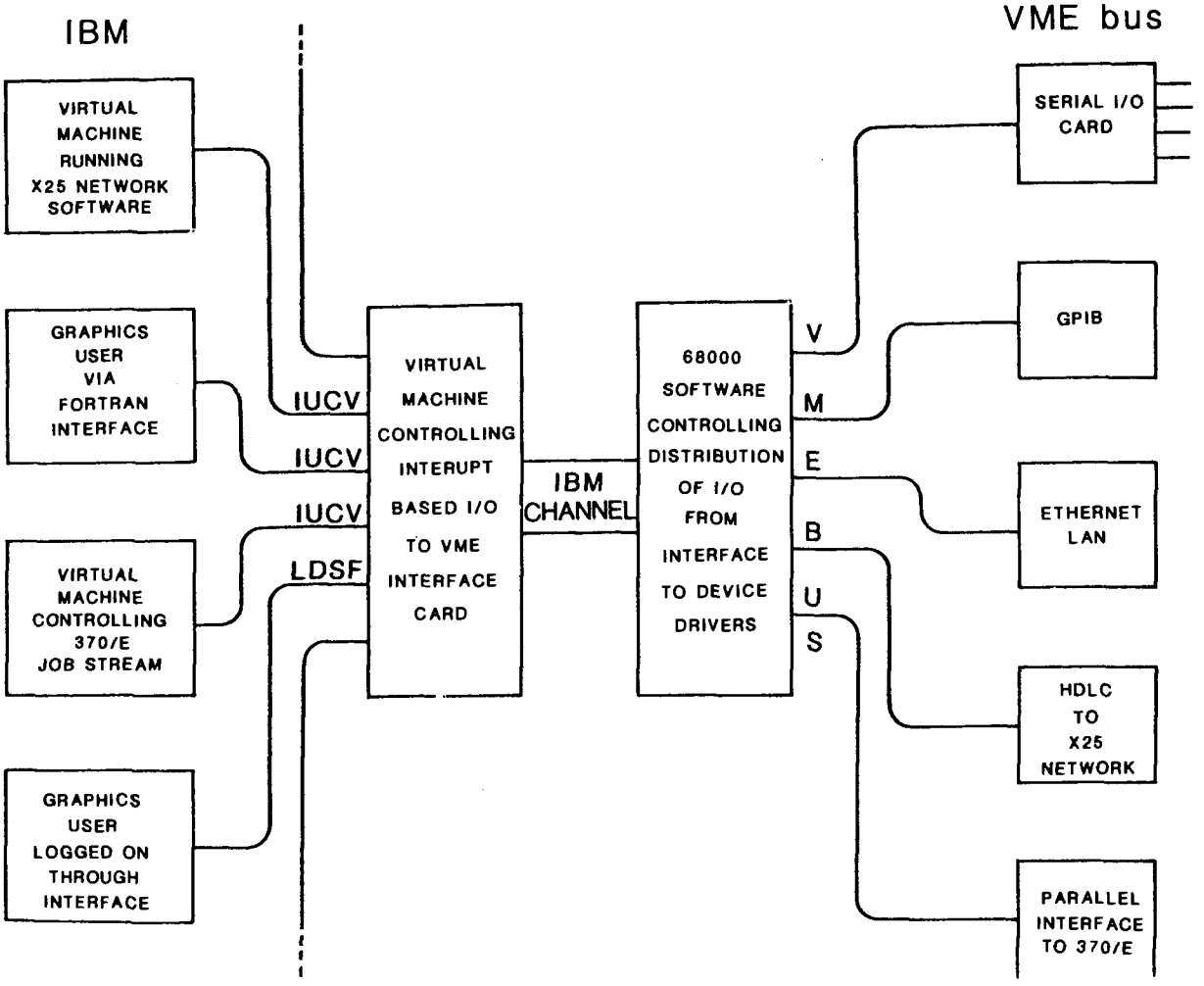
THE DIAGRAM SHOWS SOME OF THE OTHER AVAILABLE VMEBUS HARDWARE THAT MIGHT BE USEFULLY ATTACHED TO AN IBM THROUGH

THE INTERFACE. THE MOTOROLA ETHERNET BOARD FOR A DEPARTMENTAL LAN AND AN HDLC BOARD FOR CONNECTION TO THE NATIONAL PACKET SWITCHING NETWORK ARE OF PARTICULAR INTEREST TO THE DEPARTMENT.

ANY FUTURE DEVELOPMENT WORK ON THE INTERFACE WILL BE CONCENTRATED ON THE HDLC AND ETHERNET CONNECTIONS.

* * *

PRESENT STATUS
& FUTURE POSSIBILITIES



Part 3
LIST OF PARTICIPANTS

Total number of registered participants: 294

Breakdown:

- Industry: Total 52

- Member states: Total 188
  Austria: 1
  Belgium: 4
  Denmark: 4
  France: 52
  Germany (Federal Republic): 40
  Italy: 26
  The Netherlands: 11
  Norway: 7
  Switzerland: 23
  Sweden: 5
  United Kingdom: 15

- Other Countries: Total 22
  Canada: 2
  Finland: 3
  Iraq: 2
  Poland: 5
  U.S.A.: 10

- CERN: Total 84
  DD Division: 27
  EF Division: 8
  EP Division: 34
  LEP Division: 3
  PS Division: 2
  SPS Division: 10

AARSNES, K
UNIVERSITY OF BERGEN
DEPARTMENT OF PHYSICS
ALLEGT 55
5000 BERGEN
NORWAY

ABBES, M
COLLEGE DE FRANCE
LAB. PHYSIQUE CORPUSCULAIRE
11 PLACE MARCELIN BERTHELOT
75231 PARIS CEDEX 05
FRANCE

ACTRON
(W. OSSEL)
MARKERKANT, 1206
POSTBOX 3000
1300 EA ALMERE-STAD
THE NETHERLANDS

ADAMEC, F
SIN
5234 VILLIGEN
SWITZERLAND

ADAMS, M
HAID-UND-NEU-STRASSE, 10-14
7500 KARLSRUHE
WEST GERMANY

AGEHED, K
RESEARCH INST OF PHYSICS
FRESLATIVAGEN 24
10405 STOCKHOLM 50
SWEDEN

AGHION, F
LABORATORIO CICLOTRONE
VIA CELORIA, 16
20133 MILANO
ITALY

AGUER, MME M
C.E.N. SACLAY
DPH N ME
91191 GIF SUR YVETTE CEDEX
FRANCE

AL-BAKER, M
IAEC
NUCLEAR RESEARCH CENTER
TUWAITHA, BAGHDAD
IRAQ

AL-CHALABI, Z
IAEC
NUCLEAR RESEARCH CENTER
TUWAITHA, BAGHDAD
IRAQ

ALEXANDER, J
DARESBURY NUCLEAR PHYSICS LAB.
DARESBURY, WARRINGTON WA4 4AD
UNITED KINGDOM

ANDERSSON A. /CERN - DIVISION DD

ANTARES AG
(F. ZAHORKA)
SCHULSTRASSE, 3
5415 NUSSBAUMEN/BADEN
SWITZERLAND

ANTHONIOZ-BLANC, J. /CERN - DIVISION DD

ASTESAN, F
PNHE
TOUR 32 R DE C
4 PLACE JUSSIEN
75230 PARIS
FRANCE

ASTROL SA
(B. STORNI)
4, VIA STAZIONE
6596 GORDOLA
SWITZERLAND

BAGNARA, R
INFN BOLOGNA
VIA IRNERIO 46
40126 BOLOGNA
ITALY

BARRELET, M
LPNHE
ECOLE POLYTECHNIQUE
91128 PALAISEAU CEDEX
FRANCE

BECK, F. /CERN - DIVISION EP

BEE, C
PHYSICS DEPARTMENT
LIVERPOOL UNIVERSITY
LIVERPOOL
UNITED KINGDOM

BEIER, G
BESCHLEUNIGERLABOR
DER UNIVERSITAT UND
TECHNISCHEN UNIVERSITAT MUNCHEN
HOCHSCHULGELANDE
8046 GARCHING
WEST GERMANY

BELLEMAN, J. /CERN – DIVISION EP

BERGOZ SA
(J. BERGOZ)
CROZET
01170 GEX
FRANCE

BERNAUDIN, P
LAB. ACCELERATEUR LINEAIRE
BAT 200
91405 ORSAY
FRANCE

BERTSCH, Y
LAP - ANNECY
CHEMIN DE BELLEVUE
B.P. 909
74019 ANNECY CEDEX
FRANCE

BICC-VERO ELECTRONICS LTD
(A. TIMMINS)
UNIT 5, INDUSTRIAL ESTATE
FLANDERS ROAD,
HEDGE END,
SOUTHAMPTON,
HAMPSHIRE SO3 3LG
UNITED KINGDOM

BICC-VERO ELECTRONICS LTD
(M. HEDGES)
UNIT 5, INDUSTRIAL ESTATE
FLANDERS ROAD
HEDGE END
SOUTHAMPTON
HAMPSHIRE SO3 3LG
UNITED KINGDOM

BIESER, F
LAWRENCE BERKELEY LABORATORY
COMPUTATION DEPARTMENT
BLDG 50 ROOM 250
BERKELEY, CA 94720
USA

BIRSA, R
INSTITUTO NATIONALE
FISICA NUCLEARE
VIA A VALERIO 2
TRIESTE
ITALY

BLAKE, J. /CERN – DIVISION DD

BOGAERTS, J. /CERN – DIVISION DD

BOHRINGER, T. /CERN – DIVISION EP

BORER ELECTRONICS
(M. RYSER)
POSTFACH
4501 SOLOTHURN
SWITZERLAND

BOS, K. /CERN – DIVISION EP

BRACCO, A
DIPARTIMENTO DI FISICA
UNIVERSITA DI MILANO
VIA CELORIA, 16
20133 MILANO
ITALY

BRISSON, J
DPHPE-SEIPE
CEN SACLAY
91191 GIF SUR YVETTE CEDEX
FRANCE

BRONSON, M
LAWRENCE BERKELEY LABORATORY
COMPUTATION DEP
BLDG 50 ROOM 250
BERKELEY, CA 94720
USA

BRONSTAD, K
DEPARTMENT OF PHYSICS
ALLEGT 55
5000 BERGEN
NORWAY

R. L. BROOMFIELD (ELECTRONICS)
(R. BROOMFIELD)
28, RUE DE VILLAGE
1295 MIES
SWITZERLAND

BUDINICH, M
DIPARTIMENTO DI FISICA
UNIVERSITY OF TRIESTE
VIA VALERIO 2
34127 TRIESTE
ITALY

BURCKHART, MME D. /CERN - DIVISION DD

BURCKHART, H. /CERN - DIVISION EF

CAIRANTI, S. /CERN - DIVISION EF

CAMTO LTD
(F. THOMSEN)
LINDEENGEN, 10
2740 SKOVLUNDE
DENMARK

CAMTO LTD
(W. ADERSEN)
LINDEENGEN, 10
2740 SKOVLUNDE
DENMARK

CARLEN, L
UNIVERSITY OF LUND
DEPARTMENT OF PHYSICS
SOLVEGATAN, 14
223 62 LUND
SWEDEN

CARLES, G
CEN-SACLAY
DPHPE/SEIPE
91191 GIF SUR YVETTE CEDEX
FRANCE

CAVALLARI, G. /CERN - DIVISION EF

CECCHET, G
UNIVERSITY OF PAVIA
INFN
V BASSI 6
27100 PAVIA
ITALY

CHIEFARI, G. /CERN - DIVISION EP

CHRISTENSEN, P
RISOE NATIONAL LABORATORY
ELECTRONICS DEPARTMENT
4000 ROSKILDE
DENMARK

CHRISTER, L. /CERN - DIVISION DD

CHRONOLOGIC SA
(C. BUFFAM)
13 CHEMIN DU LEVANT
01210 FERNEY-VOLTAIRE
FRANCE

CIAPALA, E. /CERN - DIVISION LEP

CITTOLIN, S. /CERN - DIVISION DD

CLEMENT, J
GANIL
B.P. 5027
14021 CAEN CEDEX
FRANCE

COMYN, M
TRIUMF
4004 WESBROOK MALL
VANCOUVER, B.C., V6T 2A3
CANADA

CREATIVE ELECTRONICS SYSTEMS
(J. BOVIER)
ROUTE DU PONT BUTIN 70
1213 PETIT-LANCY/GENEVA
SWITZERLAND

CREATIVE ELECTRONIC SYSTEMS
(F. WORM)
ROUTE DU PONT BUTIN 70
1213 PETIT-LANCY/GENEVA
SWITZERLAND

CROSETTO, D
UNIVERSITA DI TORINO
ISTITUTO DI FISICA
CORSO M D' AZEGLIO 46
10125 TORINO
ITALY

CUTTONE, G
LABORATORIO CICLOTRONE
VIA CELORIA, 16
20133 MILANO
ITALY

D'ANONE, I
INFN
46 VIA IRNERIO
40126 BOLOGNA
ITALY

DARBO, G. /CERN – DIVISION EF

DARNAUER, S
20 YORKTON CT
ST PAUL MN 55055
USA

DATACARE AG
(H. MITTENDORF)
UNTERE BAHNOFSTRASSE, 19
9500 WIL
SWITZERLAND

DE CLERCQ, C
VRIJE UNIVERSITEIT BRUSSEL
DIENST ELEM
PLEINLAAN 2
1050 BRUXELLES
BELGIUM

DEGRE, A. /CERN – DIVISION EP

DEHAVAY, C. /CERN – DIVISION PS

DESCHAMPS, J
UNIVERSITE PARIS-SUD
INSTITUT DE PHYSIQUE NUCLEAIRE
SERVICE ELECTRONIQUE PHYSIQUE
BATIMENT 102
91406 ORSAY
FRANCE

DI GIOVANNI, P. /CERN – DIVISION SPS

DICK, P
SIN
5234 VILLIGEN
SWITZERLAND

DORENBOS, T. /CERN – DIVISION PS

DORNIER
(R. WILCKE)
MUHLENKAMP 32
2000 HAMBURG 60
WEST GERMANY

DUCORPS, A
LAB. ACCELERATEUR LINEAIRE
GROUPE CHAMBRE A BULLES
91405 ORSAY
FRANCE

DUMONT, G. /CERN – DIVISION EP

ECK, C. /CERN – DIVISION DD

EHS
(M. SADELER)
INDUSTRIE STRASSE
8000 MUNICH
WEST GERMANY

ELTEC ELECTRONIC
(J. BULLACHER)
POSTFACH 65
GALILEO GALILEI STRASSE,
6500 MAINZ 42
WEST GERMANY

ELTRADE SCHRODEL AG
(K. SCHRODEL)
SEESTRASSE, 323
8038 ZURICH
SWITZERLAND

ELECTRONIC MODULAR SYSTEMS
(M. SCHACHT)
ACKERSTRASSE, 71-76
1000 BERLIN 65
WEST GERMANY

ENGSTER, C. /CERN – DIVISION EP

ERNI + CO AG
(M. HUGELOHOFER)
STATIONS STRASSE, 31
8306 BRUTTISELLEN
SWITZERLAND

EYRING, A. /CERN – DIVISION EP

FABBRO, B
C.E.N. SACLAY DPH-N/ME
91191 GIF SUR YVETTE CEDEX
FRANCE

FABRIMEX AG
(J. SILHAVY)
KIRCHENWEG, 5
8032 ZURICH
SWITZERLAND

FABRIMEX AG
(C. BAERISWYL)
KIRCHENWEG 5
8032 ZURICH
SWITZERLAND

FAIVRE, J
C.E.N. SACLAY DPH-N/ME
91191 GIF SUR YVETTE CEDEX
FRANCE

FARTHOUAT, P
CEN-SACLAY
DPHPE, B P 2
91191 GIF SUR YVETTE CEDEX
FRANCE

FEYT, J. /CERN - DIVISION EF

FONTAINE, G
LABORATOIRE DE PHYSIQUE CORPUSCULAIRE
COLLEGE DE FRANCE
11 PLACE MARCELIN BERTHELOT
75231 PARIS CEDEX 05
FRANCE

FORCE COMPUTERS GMBH
(L. PAYERL)
DAIMLERSTRASSE, 9
8012 OTTOBRUN
WEST GERMANY

FORCE COMPUTERS GMBH
(M. LOESEL)
DAIMLERSTRASSE, 9
8012 OTTOBRUNN
WEST GERMANY

GALLICE, P
DEIN/SIR
CEN SACLAY
91191 GIF SUR YVETTE CEDEX
FRANCE

GALLNO, P. /CERN - DIVISION DD

GAMBA, D
UNIVERSITA DI TORINO
ISTITUTO DI FISICA
CORSO M D' AZEGLIO 46
10125 TORINO
ITALY

GANDOIS, B
CEN-SACLAY
DPHPE, B P 2
91191 GIF SUR YVETTE CEDEX
FRANCE

GENTIT, F
CEN-SACLAY
DPHPE, B P 2
91191 GIF SUR YVETTE CEDEX
FRANCE

GIACOMELLI, P. /CERN - DIVISION DD

GIOVE, D
INFN
SEZIONE DI MILANO
VIA CELORIA, 16
20133 MILANO
ITALY

GOSMAN, G
NIKHEF
P.O. BOX 41882
1009 DB AMSTERDAM
THE NETHERLANDS

GOURCY, G
CEN-SACLAY
DPHN/AL
91191 GIF SUR YVETTE CEDEX
FRANCE

GOURNAY, J
CEN-SACLAY
DPHN/AL
91191 GIF SUR YVETTE CEDEX
FRANCE

GRASSO, A
ISTITUTO FISICA GENERALE
UNIVERSITA DI TORINO
CORSO M D'AZEGLIO 46
10125 TORINO
ITALY

GRILLO, A
INFN
CASELLA POSTALE, 13
00044 FRASCATI
ITALY

GRINDHAMMER, G
DESY F36
NOTKESTRASSE, 85
2000 HAMBURG 52
WEST GERMANY

GRUNG, B
UNIVERSITY OF BERGEN
DEPARTMENT OF PHYSICS
ALLEGT 55
5014 BERGEN
NORWAY

GUGLIELMI, L
COLLEGE DE FRANCE
PHYSIQUE CORPUSCULAIRE
11 PLACE MARCELIN BERTHELOT
75231 PARIS CEDEX 05
FRANCE

GUILLAUME, C. /CERN – DIVISION SPS

GUO, Y. /CERN – DIVISION EP

GUSTAFSSON L. /CERN – DIVISION DD

HAGLUND, R. /CERN – DIVISION EP

HAMEL, J
LABORATOIRE NATIONAL SATURNE
CEN SACLAY
91191 GIF SUR YVETTE CEDEX
FRANCE

HANSEN, S. /CERN – DIVISION LEP

HARROCH, H
INSTITUT DE PHYSIQUE NUCLEAIRE
B P 1
91405 ORSAY CEDEX
FRANCE

HAYNES, W J. /CERN – DIVISION EP

HECK, B. /CERN – DIVISION EF

HEIMANN, P
MAX-PLANCK INSTITUT FUR PLASMAPHYSIK
8046 GARCHING
WEST GERMANY

HERTWECK, F
MAX-PLANCK-INSTITUT
FUR PLASMAPHYSIK
BEREICH INFORMATIK
8046 GARCHING
WEST GERMANY

HERVE, C
LABORATOIRE NATIONAL SATURNE
CEN SACLAY
91191 GIF SUR YVETTE CEDEX
FRANCE

HEWLETT-PACKARD GMBH
(M. WIPPERFELD)
RCD-RND
WIRT 3
HERRENBERGER STR 130
7030 BOBLINGEN
WEST GERMANY

HEWLETT-PACKARD GMBH
(W. BRAUN)
BCD-RND
WIRT 3
HERRENBERGER STR 130
7030 BOBLINGEN
WEST GERMANY

HOEY-CHRISTENSEN, P
NIELS BOHR INSTITUTE
TANDEM ACCELERATOR LABORATORY
4000 ROSKILDE
DENMARK

HONSCHEID, K
PHYSIKALISCHES INSTITUT
DER UNIVERSITAT BONN
NUSSALLEE, 12
5300 BONN
WEST GERMANY

HORKY, V
SIN
5234 VILLIGEN
SWITZERLAND

HUET, M
DPHN/HE
CEN - SACLAY
91191 GIF SUR YVETTE CEDEX
FRANCE

HUGHES-JONES, R
UNIVERSITY OF MANCHESTER
H E P GROUP
SCHUSTER LAB.
MANCHESTER M13 9PL
UNITED KINGDOM

INTERACTIVE CIRCUITS SYSTEMS LTD
(S. DEWAR)
3101 HAWTHORNE ROAD
OTTAWA
ONTARIO K1G-3V8
CANADA

JAASKELAINEN, M
UNIVERSITY OF JYVASKYLA
DEPARTMENT OF PHYSICS
S40100 JYVASKYLA
FINLAND

JACQUET, G
INST DE PHYSIQUE NUCLEAIRE
43 BD DU 11 NOV 1918
69622 VILLEURBANNE CEDEX
FRANCE

JEFFERY, D
H H WILLS PHYSICS LABORATORY
UNIVERSITY OF BRISTOL
TYNDALLS AVENUE
BRISTOL BS8 1TL
UNITED KINGDOM

JOHANSSON, T. /CERN - DIVISION EP

JOVANOVIC, P
UNIVERSITY OF BIRMINGHAM
PHYSICS DEPARTMENT
P.O. BOX 363
BIRMINGHAM B15 2TT
UNITED KINGDOM

KERZENDORF, W
MAX-PLANCK INSTITUT
FUR PLASMAPHYSIK
RECHENZENTRUM
8046 GARCHING BEI MUNCHEN
WEST GERMANY

KISIELEWSKI, B
INST OF NUCLEAR PHYSICS
UL KOWIORY 26A
30-055 KRAKOW
POLAND

KLESSE, R
ILL GRENOBLE
B.P. 156
X CENTRE DE TRI
38042 GRENOBLE CEDEX
FRANCE

KLOK, P
UNIVERSITY, PHYSICS DEPARTMENT
TOERNOOIVELD
6525 ED NIJMEGEN
THE NETHERLANDS

KO, W. /CERN - DIVISION EP

KUZMINSKI, J. /CERN - DIVISION EP

LE CROY RESEARCH SYSTEMS SA
(M. VINCELLI)
101 ROUTE DU NANT-D'AVRILL
1217 MEYRIN 1
SWITZERLAND

LE DU, P
CEN-SACLAY
DPHPE, B P 2
91191 GIF SUR YVETTE CEDEX
FRANCE

LECOQ, J
LAPP
CHEMIN DE BELLEVUE
B.P. 909
74019 ANNECY CEDEX
FRANCE

LESQUOY, E
CEN-SACLAY
SECB
91191 GIF SUR YVETTE CEDEX
FRANCE

LEYENDECKER, P
EMBL
POSTFACH 10-22-09
6900 HEIDELBERG
WEST GERMANY

LIGUORI, C
INFN - SEZ DI MILANO
VIA CELORIA, 16
20133 MILANO
ITALY

LOVDE, J
INSTITUT DE PHYSIQUE NUCLEAIRE
UNIVERSITE DE LAUSANNE
BATIMENT DES SCIENCES PHYSIQUES
DORIGNY
1015 LAUSANNE
SWITZERLAND

LUONG, T
GANIL
B.P. 5027
14021 CAEN CEDEX
FRANCE

LUTTER, L
BESCHLEUNIGERLABOR
DER UNIVERSITAT UND
TECHNISCHEN UNIVERSITAT MUNCHEN
HOCHSCHULGELANDE
8046 GARCHING
WEST GERMANY

MACAVERO, E
ISTITUTO DI FISICA
UNIVERSITA DI MILANO
VIA CELORIA, 16
20133 MILANO
ITALY

MADICHARD, M
CEN SACLAY
SAP
91191 GIF SUR YVETTE CEDEX
FRANCE

MANDRIOLI, G
ISTITUTO DI FISICA
46 VIA IRNERIO
40126 BOLOGNA
ITALY

MARBOT, R
ECOLE POLYTECHNIQUE
LPNHE
ROUTE DE SACLAY
91128 PALAISEAU
FRANCE

MARKT UND TECHNIK
(R. HUTTENLOHER)
HANS PINSEL STRASSE, 2
8013 HAAR BEI MUNCHEN
WEST GERMANY

MARON, G
INFN
LABORATORI NAZIONALI LEGNARO
35020 LEGNARO
ITALY

MARZANO, F
INFN
VIALE REGINA ELENA 299
00161 ROMA
ITALY

MATTSSON, S
UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF PHYSICS
41296 GOTEBORG
SWEDEN

MAZZANTI, M
ISTITUTO DI FISICA
UNIVERSITA DI MILANO
VIA CELORIA, 16
20133 MILANO
ITALY

MCLAREN, R. /CERN - DIVISION DD

MCPARLAND, C
COMPUTATION DEPARTMENT,
BLDG 50, ROOM 250
LAWRENCE BERKELEY LABORATORY
BERKELEY, CA 94720
USA

MENCIK, M
LAB. ACCELERATEUR LINEAIRE
BAT 200
91405 ORSAY CEDEX
FRANCE

MERCER, D
HIGH ENERGY PHYSICS GROUP
SCHUSTER LABORATORY
THE UNIVERSITY, BRUNSWICK ST
MANCHESTER M13 9PL
UNITED KINGDOM

MME E MERONI,
ISTITUTO DI FISICA
UNIVERSITA DI MILANO
VIA CELORIA, 16
MILANO
ITALY

MERRY, B
PHYSICS DEPARTMENT
LIVERPOOL UNIVERSITY
LIVERPOOL L69 3BX
UNITED KINGDOM

MERTENS, V
DESY-F12
NOTKESTRASSE, 85
2000 HAMBURG 52
WEST GERMANY

MIDTTUN, G
INSTITUTE OF PHYSICS
UNIVERSITY OF OSLO
OSLO 3
NORWAY

MINOR, M
LOS ALAMOS NATIONAL LABORATORY
P.O. BOX 1663, MS G776
LOS ALAMOS, NM 87545
USA

MJORNMARK, U. /CERN - DIVISION EP

MONTANARI, P
INFN BOLOGNA
VIA IRNERIO 46
40126 BOLOGNA
ITALY

W. MOOR S.A.
(P. BARDET)
C.P. 57
1026 DENGES
SWITZERLAND

MORANDO, M
UNIVERSITA DEGLI STUDI
ISTITUTO DI FISICA
VIA MARZOLO 8
35100 PADOVA
ITALY

MORETON, T
PHYSICS DEPARTMENT
LIVERPOOL UNIVERSITY
P.O. BOX 147, OXFORD STREET
LIVERPOOL L69 3BX
UNITED KINGDOM

MORGUE, M
INST. DE PHYSIQUE NUCLEAIRE
43 BD DU 11 NOVEMBRE 1918
69622 VILLEURBANNE CEDEX
FRANCE

MOSTEK INTERNATIONAL
(W. OSSWALD)
51 RUE DE MOULIN A PAPIER BTE 3
1160 BRUXELLES
BELGIUM

MOTOROLA MICROSYSTEMS
(M. RUDYK)
TAUNUSSTRASSE, 51
8000 MUNCHEN 40
WEST GERMANY

MOTOROLA MICROSYSTEMS
(J. WERDEHAUSEN)
TAUNUSSTRASSE, 51
8000 MUNICH 40
WEST GERMANY

MUGNAI, G. /CERN - DIVISION SPS

MULLER, H. /CERN - DIVISION EP

NAPPEY, P. /CERN - DIVISION EF

NATIONAL INSTRUMENTS
(W. NOWLIN)
12109 TECHNOLOGY BLVD
AUSTIN, TEXAS 78729
USA

NAVARRE, J
LABORATOIRE NATIONAL SATURNE
CEN SACLAY
91191 GIF SUR YVETTE CEDEX
FRANCE

NIEUWENHUIS, C. /CERN - DIVISION EP

NORSK DATA AS
(S. JOHANSEN)
BOX 25 BOGERUD
0621 OSLO 6
NORWAY

NORSK DATA AS
(E. KRISTIANSEN)
BOX 25 BOGERUD
0621 OSLO 6
NORWAY

NOTZ, D
DESY F1
NOTKESTRASSE, 85
2000 HAMBURG 52
WEST GERMANY

NYFFENEGGER P. /CERN – DIVISION DD

OWEN, E
DARESBURY NUCLEAR PHYSICS LAB.
DARESBURY, WARRINGTON WA4 4AD
UNITED KINGDOM

PAIN, J
C.E.N. SACLAY
DPH-N/ME
91191 GIF SUR YVETTE CEDEX
FRANCE

PARKMAN, C. /CERN – DIVISION DD

PASSENEAU, MME M
PNHE
TOUR 32 R DE C
4 PLACE JUSSIEN
75230 PARIS
FRANCE

PAWLAK, T
INSTITUTE OF PHYSICS
WARSAW TECHNICAL UNIVERSITY
UL KOSZYKOWA 75
00-662 WARSZAWA
POLAND

PERRIN, Y. /CERN – DIVISION DD

PERYT, W
INSTITUTE OF PHYSICS
WARSAW TECHNICAL UNIVERSITY
UL KOSZYKOWA 75
00-662 WARSZAWA
POLAND

PETERSEN, J. /CERN – DIVISION DD

PHILIPS EXPORT BV
(M. KLEWER)
BUILDING TQ III-2
5600 MD EINDHOVEN
THE NETHERLANDS

PHILIPS CTI
(MME M. PAUKER)
4-16 AVENUE DU GENERAL LECLERC
92260 FONTENAY AUX ROSES
FRANCE

PHILIPS INDUSTRIAL DATA PROCESSING
(H. MASSKANT)
BUILDING TQ III-1
P.O. BOX 218
5600 MD EINDHOVEN
THE NETHERLANDS

PHILIPS INTERNATIONAL
(A. SCHELLEKENS)
DEP. EXTERNAL STANDARDISATION
5600 MD EINDHOVEN
THE NETHERLANDS

PICCINELLI, G
DIP FISICA UNIVERSITA
PIAZZALE A MORO
2-00185 ROMA
ITALY

PIERSCHEL, G
PHYSIKALISCHES INSTITUT III A
PHYSIKZENTRUM
SOMMERFELD STRASSE, 4
5100 AACHEN
WEST GERMANY

PIETARINEN, E
DEPARTMENT OF HIGH ENERGY PHYSICS
UNIVERSITY OF HELSINKI
SILTAVUORENPENGER 20C
SF-0170 HELSINKI
FINLAND

PIGNARD, C. /CERN – DIVISION SPS

PIJLGROMS, B. /CERN – DIVISION EP

PIQUET, B
CEN SACLAY
91191 GIF SUR YVETTE CEDEX
FRANCE

PLESSEY MICROSYSTEMS LTD
(A. WAIGHTS)
WATER LANE
TOWCESTER
NORTHANTS NN12 7JN
UNITED KINGDOM

PLESSEY MICROSYSTEMS LTD
(P. BURNLEY)
WATER LANE
TOWCESTER
NORTHANTS NN12 7JN
UNITED KINGDOM

PLUTA, J
INSTITUTE OF PHYSICS
WARSAW TECHNICAL UNIVERSITY
UL KOSZYKOWA 75
00-662 WARSZAWA
POLAND

PONTING, P J. /CERN – DIVISION EP

POVEL, H
INSTITUT FUR ASTRONOMIE
ETH-ZENTRUM
8092 ZURICH
SWITZERLAND

PROME, M
GANIL
PB 5027
14021 CAEN CEDEX
FRANCE

PROTOULIS, K. /CERN – DIVISION DD

RABANY, M. /CERN – DIVISION SPS

RAINE, B
GANIL
B.P. 5927
14021 CAEN CEDEX
FRANCE

RAUSCH, R. /CERN – DIVISION SPS

REHLICH, K
DESY F 52
NOTKESTRASSE, 85
2000 HAMBURG 52
WEST GERMANY

REICHART, W
PHYSIK INSTITUT DER UNIVERSITAT
SCHONBERGGASSE 9
8001 ZURICH
SWITZERLAND

REITHLER, H
PHYSIKALISCHES INSTITUT III A
PHYSIKZENTRUM
A SOMMERFELD STRASSE
5100 AACHEN
WEST GERMANY

REYNAUD, S. /CERN – DIVISION EF

RIBORDY, L
INSTITUT DE PHYSIQUE
PEROLLES
1700 FRIBOURG
SWITZERLAND

RICHTER, H
MAX-PLANCK INSTITUT FUR PLASMAPHYSIK
8046 GARCHING
WEST GERMANY

RIJLLART, A. /CERN – DIVISION EP

RIMMER, E. /CERN – DIVISION DD

RISTORI, C
CENTRE D'ETUDES NUCLEAIRES
38 AVENUE DES MARTYRS
38041 GRENOBLE CEDEX
FRANCE

ROBCON OY
P.O. BOX 9
00391 HELSINKI
FINLAND

ROCH G. /CERN – DIVISION DD

ROOSEN, R
MHE
VRIJE UNIVERSITEIT BRUSSEL
PLEINLAAN 2
1050 BRUSSELS
BELGIUM

ROSSA, E. /CERN – DIVISION LEP

ROUGEVIN-BAVILLE, P
CEN-SACLAY
DPHPE, SEPH
91191 GIF SUR YVETTE CEDEX
FRANCE

RUHM, W. /CERN – DIVISION EP

SAINSON, J. /CERN – DIVISION SPS

SALA, S. /CERN – DIVISION EP

SANDOVAL, A. /CERN – DIVISION EP

SCHAER, M
ECOLE POLYTECHNIQUE FEDERALE
INSTITUT DE GENIE ATOMIQUE
ECUBLENS
1015 LAUSANNE
SWITZERLAND

SCHARFF-HANSEN, P. /CERN - DIVISION DD

SCHMAL, N
UNIVERSITAT KOLN
INSTITUT FUR KERNPHYSIK
ZULPICHER STRASSE, 77
5000 KOLN
WEST GERMANY

SCHMICKLER, H
PHYS INSTITUT
DER UNIVERSITAT BONN
NUBALLEE, 14
5300 BONN
WEST GERMANY

SCHUTT, J. /CERN - DIVISION EP

SCHWAB, H
ILL GRENOBLE
B.P. 156
X CENTRE DE TRI
38042 GRENOBLE CEDEX
FRANCE

SCIUBBA, A
DIP FISICA UNIVERSITA
PIAZZALE A MORO
2-00185 ROMA
ITALY

SEMMEL, P
INSTITUT FUR KERNPHYSIK DER UNIVERSTAT
J J BECHERWEG 33
6500 MAINZ
WEST GERMANY

SENDALL, D. /CERN - DIVISION DD

SEYERLEIN, J
MAX-PLANCK-INSTITUT FUR ASTROPHYSIK
FOHRINGER RING 6
8000 MUNCHEN 40
WEST GERMANY

SHEA, M
FERMILAB
P.O. BOX 500
BATAVIA IL 60510
USA

SHERA, E
LOS ALAMOS NATIONAL LABORATORY
P.O. BOX 1663, MS D443
LOS ALAMOS, NM 87545
USA

SIMONS, W
20 YORKTON COURT
ST PAUL MN 55424
USA

SINTRA
(A. AMIEL)
74 AVENUE GABRIEL PERI
92230 GENNEVILLIERS
FRANCE

SINTRA
(G. PEROT)
74 AVENUE GABRIEL PERI
92230 GENNEVILLIERS
FRANCE

SITRUK, MME M
COLLEGE DE FRANCE
PHYSIQUE CORPUSCULAIRE
PLACE MARCELLIN BERTHELET
78005 PARIS
FRANCE

SKAALI, B
UNIVERSITY OF OSLO
INSTITUTE FOR THEORETICAL PHYSICS
BLINDERN
OSLO 3
NORWAY

SMITH, T
PHYSICS DEPARTMENT
LIVERPOOL UNIVERSITY
LIVERPOOL L69 3BX
UNITED KINGDOM

SPHICAS, P. /CERN - DIVISION DD

STEINBERGER, K
BESCHLEUNIGERLABOR
DER UNIVERSITAT UND
TECHNISCHEN UNIVERSITAT MUNCHEN
HOCHSCHULGELANDE
8046 GARCHING
WEST GERMANY

STUCKENBERG, H
DESY
NOTKESTRASSE, 85
2000 HAMBURG 52
WEST GERMANY

STUMPE, B. /CERN - DIVISION SPS

TAGESEN, S
INSTITUT FUR RADIUMFORSCHUNG UND
KERNPHYSIK DER AKADEMIE
BOLTZMANNGASSE 3
1090 WIEN
AUSTRIA

TAVERNIER, S
VUB DIENST ELEM
PLEINLAAN 2
1050 BRUSSELS
BELGIUM

TAYLOR, B. /CERN - DIVISION EP

TELELOGIC
(R. NILSSON)
DAG HAMMARSKJOLDS VAG, 31
75237 UPPSALA
SWEDEN

TELELOGIC
(A. FLODIN)
DAG HAMMARSKJOLDS VAG 31
75237 UPPSALA
SWEDEN

THENARD, J
LAP - ANNECY
CHEMIN DE BELLEVUE
B.P. 909
74019 ANNECY CEDEX
FRANCE

TORELLI, M
INFN
VIALE REGINA ELENA 299
00161 ROMA
ITALY

TRAINITO, G. /CERN - DIVISION DD

TREMBLET, L. /CERN - DIVISION DD

TROSTER, D. /CERN - DIVISION EP

TUUVA, T. /CERN - DIVISION EP

VALVO APPLIKATIONS LABORATORIUM
(N. NISSEN)
VOGT-KOLLN STRASSE, 30
2000 HAMBURG 54
WEST GERMANY

VAN DER VLUGT, C. /CERN - DIVISION EP

VAN KONINGSVELD, L. /CERN - DIVISION EP

VANUXEM, J P. /CERN - DIVISION EP

VERMEULEN, J
NIKHEF
P.O. BOX 41882
1009 DB AMSTERDAM
THE NETHERLANDS

VERWEIJ, H. /CERN - DIVISION EP

VITA
(L. HEVLE)
SUITE #E
10229 N SCOTTSDALE ROAD
ARIZONA 85253
USA

VITA
(M. KLEWER)
BEGIJNSTRAAT, 10
5303 XW VELDHOVEN
THE NETHERLANDS

VMES BENELUX BV
(S. POPPENK)
POSTBUS 3000
1300 EA ALMERE
THE NETHERLANDS

VMES BENELUX BV
(F. ESSEN)
POSTBUS 3000
1300 EA ALMERE
THE NETHERLANDS

VON DER SCHMITT, H. /CERN - DIVISION DD

VON EICKEN, H. /CERN – DIVISION DD

WALTHER, H
INSTITUT FUR KERNPHYSIK
SAARSTRASSE, 21
6500 MAINZ
WEST GERMANY

WATSON, E. /CERN – DIVISION EP

WATTS, S
PHYSICS DEPARTMENT
BRUNEL UNIVERSITY
UXBRIDGE
MIDDLESEX
UNITED KINGDOM

WAWER, W
HAHN-MEITNER-INSTITUT
BEREICH DIE
GLIENICKER STRASSE, 100
1000 BERLIN 39
WEST GERMANY

WERNER, P. /CERN – DIVISION EP

WEYMANN, M
FAK PHYSIK
UNIVERSITAT FREIBURG
HERMAN-HERDER STRASSE, 3
7800 FREIBURG
WEST GERMANY

WIJNEN, T
UNIVERSITY OF NIJMEGEN
PHYSICS DEPARTMENT III EXP HEP
6525 ED NIJMEGEN
THE NETHERLANDS

WILD HEERBRUGG AG
(R. SCHULER)
9435 HEERBRUGG
SWITZERLAND

WILHELM, R. /CERN – DIVISION SPS

WILHELMS, H
BESCHLEUNIGERLABOR
DER UNIVERSITAT UND
TECHNISCHEN UNIVERSITAT MUNCHEN
HOCHSCHULGELANDE
8046 GARCHING
WEST GERMANY

WILLIAMS, D. /CERN – DIVISION DD

WINKLER, U
PHYSIKALISCHES INSTITUT
PHILOSOPHENWEG, 12
6900 HEIDELBERG
WEST GERMANY

WOLSTENHOLME, P. /CERN – DIVISION SPS

WYLIE, H
OFFICE FOR SCIENCE AND TECHNOLOGY
AUF DER HOSTERT 3
5300 BONN 2
WEST GERMANY

ZALEWSKI, J
INSTITUTE OF ATOMIC ENERGY
DEPARTMENT E-4
05-400 OTWOCK/SWIERK
POLAND

ZANELLO, L
INFN
VIALE REGINA ELENA 299
00161 ROMA
ITALY

ZIEGLER, P
SIN
5234 VILLIGEN
SWITZERLAND

ZIEM, P
HAHN-MEITNER-INSTITUT
BEREICH KERNPHYSIK
GLIENICKER STRASSE, 100
1000 BERLIN 39
WEST GERMANY

ZYLBERAJCH, S
CEN-SACLAY
DPHPE, B P 2
91191 GIF SUR YVETTE CEDEX
FRANCE