USING CAMAC HARDWARE FOR ACCESS TO A PARTICLE ACCELERATOR

by

J.N.J. Truter

SUBMITTED TO THE FACULTY OF COMPUTER SCIENCE OF THE UNIVERSITY OF CAPE TOWN IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (COMPUTER SCIENCE)

September 1987

ABSTRACT

The design and implementation of a method to software interface high level applications programs used for the control and monitoring of a Particle Accelerator is described.

Effective methods of interfacing the instrumentation bus system with a Real time multitasking computer operating system were examined and optimized for efficient utilization of the operating system software and available hardware. Various methods of accessing the instrumentation bus are implemented as well as demand response servicing of the instruments on the bus.

ACKNOWLEDGEMENTS

I wish to express my thanks to all those who assisted in bringing this thesis to completion. I am particularly indebted to

Professor K.J. MacGregor of the University of Cape Town for acting as my supervisor for this project,

Mr H.F. Weehuizen of the National Accelerator Center, Faure for the encouragement and technical help given during the course of this project,

Dr G.F. Burdzik of the National Accelerator Center, Faure for his encouragement and help given during the course of this project,

to the other members of the Control Group for their assistance in general,

to my wife Nelet for her patience and encouragement during the preparation of this thesis at home, and

the National Accelerator Center of the Council for Scientific and Industrial Research for the facilities.

Table of Contents

CHAPTER 1

INTRODUCTION

The controlling and monitoring of the different subsystems of a particle Accelerator facility (consisting of many complex mechanical, electrical and electronic components) is essentially a task for computers using a real time operating system which allows multitasking. This computing facility can ideally be provided by technical minicomputers.

To interface the computers to the various incompatible instruments of the facility, the CAMAC (Computer Automated Measurement and Control [1]) bus interface method was selected. This decision was based on the cost and availability of standard interfacing components and the possibility to extend the interface bus over considerable distances remote from the computer. In addition CAMAC accesses are very fast, and the CAMAC hardware system has a high addressing capability regarding the different types of CAMAC modules for instrumentation interfacing.

The method of connection for the CAMAC system forms a typical star topology. As the current trend is to decentralize the Accelerator Control system function and incorporate intelligent nodes at the different subsystems to be controlled, the star topology is not ideally suited to this environment where the emphasis will be toward process-to-process communication between intelligent nodes (containing the Accelerator instrumentation to be controlled or monitored). The CAMAC branch system was the state-of-the-

art interfacing mechanism at the time when the Particle Accelerator was being constructed and is also used by other similar establishments overseas. Computer networks came into existence during the time that the interfacing to CAMAC was in progress. Computer networking hardware as well as instrumentation hardware connected together on a network was however not available. On the other hand, the required hardware for the computer to CAMAC connection did exist.

No software interface exists to interface the CAMAC bus system to the selected minicomputers. The most needed immediate requirement is to provide the computers with a method to be able to transfer data to or from the CAMAC bus system in a way that the operating system requirements for the selected minicomputers, be satisfied. The requirements of the user (the particle Accelerator) must be examined and analyzed in terms of the usage of CAMAC in the subsystems. These requirements together with the predetermined operating system specifications will determine the methods needed to interface the CAMAC system with the minicomputers.

The objective of this thesis is therefore to provide a software interfacing technique for the control computers and CAMAC for a particle Accelerator.

The dissertation is structured as follows :

In chapter 2 the introductory description of the CAMAC definition and the CAMAC hardware used in the Accelerator environment is discussed. This includes a discussion of the Parallel and Serial Branch system. Also described are the computer systems used to control the Accelerator system and the specialized hardware to connect the computer I/O bus to the CAMAC Executive crate.

The Accelerator Control System is described in chapter 3 in order to analyze the requirements to provide the most efficient methods to connect the CAMAC hardware to the computers used for controlling the Accelerator subsystems.

In chapter 4 follows a description of the method of connecting the CAMAC crates together. The two possible methods of accessing the CAMAC hardware from the computers and a possible method to service interrupts occurring in the CAMAC crate system are discussed. The actual implementation method for Privileged Subroutines is discussed. Then follows a description of a general device driver for the computers. This leads on to the description of the device driver written for the CAMAC system. The three main sections of the device driver are discussed in turn. The final part of this chapter is used to describe the servicing of demand interrupts occurring in the CAMAC crate system by using the privileged driver method and special techniques used to announce the interrupts to the operating system for further high level servicing by programs or by the CAMAC device driver in the case of a Serial Branch transfer.

An evaluation of the performance for the three methods described in the previous chapter is made in chapter 5. The performance figures for the Privileged Subroutine method is given. The method to access the crates on the Serial Branch and an alternative method is evaluated. The interrupt service implementation is evaluated in terms of the rate of service and the comparison of two methods of scheduling the high level service programs is discussed.

Chapter 6 contains a summary of the various implementation methods and the resultant conclusions.

CHAPTER 2

THE HARDWARE

2.1 DESCRIPTION OF CAMAC

Due to the complexity of high-energy and nuclear physics experiments, the need for data acquisition systems to be bus structured rather than point-to-point connections existed during the 1960s. Many different incompatible data-acquisition busses were developed. The European Standards On Nuclear Electronics (**ESONE**) committee recognized the need for a standard and between 1966 and 1969, the system (**CAMAC** - **C**omputer **A**utomated **M**easurement and **C**ontrol) was defined and the basic standard (EUR 4100) was published [2].

CAMAC is thus a standard mechanism for connecting signals to a computer. Although it can be used to connect just about anything to a computer, it is in practice used more often to connect instrumentation and experiments to data-acquisition and control computers.

The most basic system consists of a crate, a dataway and a plug-in module. The crate houses the modules and provide it with power. The dataway provides a means to allow a controller of the modules (also resident in the crate) with a method to transfer data and control signals to or from the modules.The dataway consists of 4 bussed signal groups namely :

Data, Control, Status and Power busses. In addition, the control module has access to 2 more bussed groups. These are the Look-at-me lines and the Station-addressed lines (see figure 2.2). This is a standard

laid down in the publication 'A Modular Instrumentation System for Data Handling' (IEEE 583)[1].

The method to connect a computer to a CAMAC crate is to use a CAMAC controller that has a computer Input/Output bus plugged into it from the front panel. This system can be expanded by adding multi-branch branch couplers for parallel or serial branches. Such a crate is called a System crate or Executive crate.

The CAMAC system crate consists of a housing to contain CAMAC plug-in units or modules, an Executive Controller module, Branch Coupler modules and the interface modules (Program Units or controller as above) to connect the mini-computer input/output bus to the system crate [20]. The executive controller is the control module for the program units and branch couplers [21]. It controls all the modules in the system by means of commands received via the system Data Highway (short form Dataway - carries data, control signals and power in the CAMAC crate). It also allocates usage of the dataway to requesting program units, selects branch couplers and controls the resultant branch operation timing when a program unit commands a branch operation. A program unit can be one of four groups :

a) Programmed Transfer Interface [18].
b) Interrupt Vector Generators [10].
c) Autonomous Memory Channels [19].
d) Autonomous Control Units.

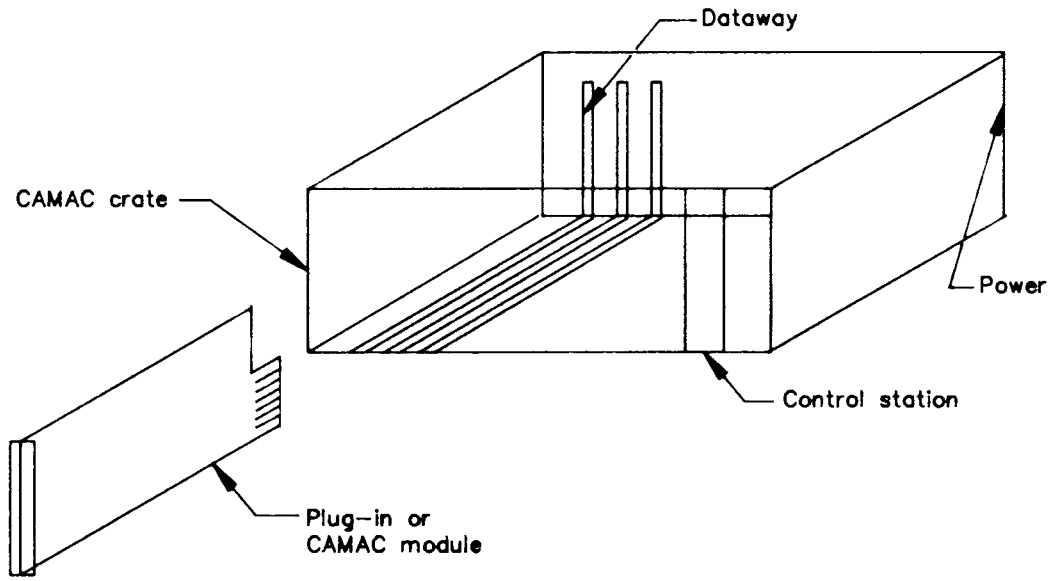


Fig 2.1 THE BASIC CAMAC STANDARD

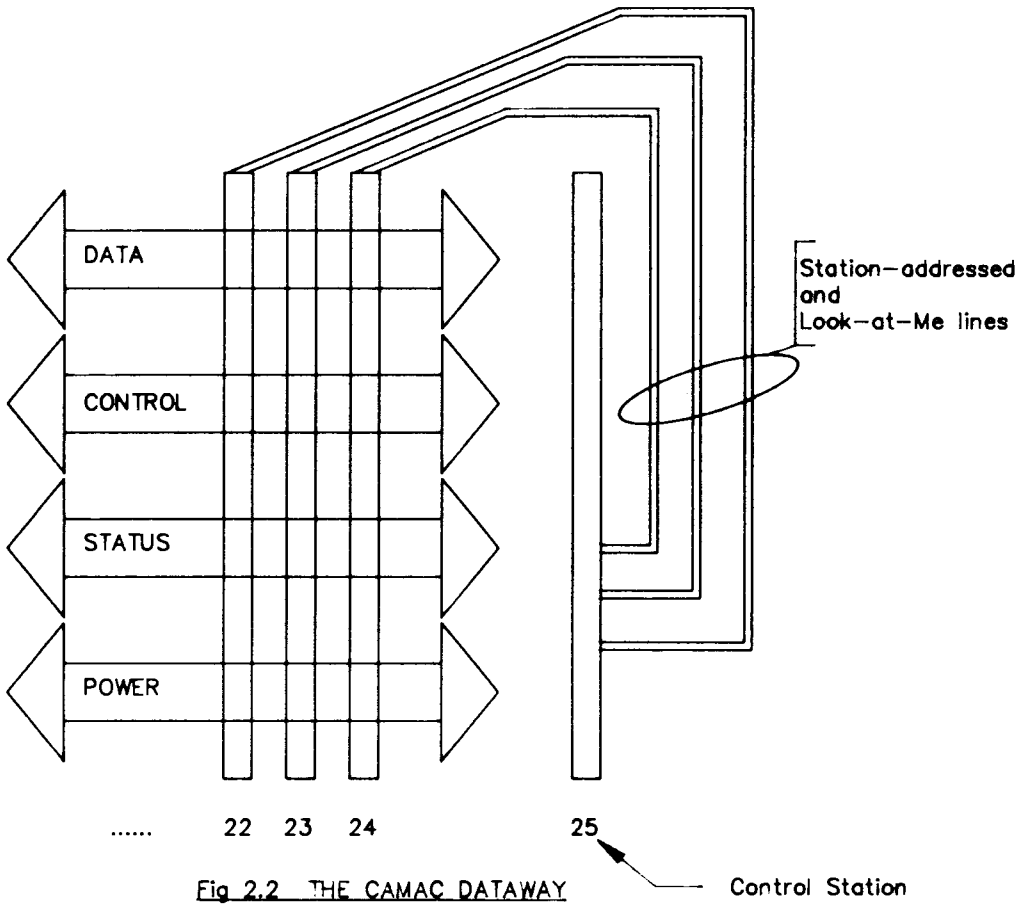


Fig 2.2 THE CAMAC DATAWAY

Program Transfer Interface (PTI) [18] :

This interface is accessed by the computer by using program input/output transfers to load the System Crate command into buffers in the interface units and transfers associated data and/or status information to the computer.

The PTI uses a block of six consecutive I/O Select Codes. The address of interface card in the interface bus of the computer, is known as a Select Code (SC). The base select code for the PTI is selected on the PTI via switches.
The PTI contains six registers namely :

I/O Select code	Operational register	Associated flag
xØ	Control and Status (CSR)	Error
x1	CAMAC Command (CCR)	NOT Q
x2	CAMAC Function (CFR)	Function complete
x3	MSB data (8 bits) (DHR)	Busy
x4	LSB data (16 bits) (DLR)	Data
x5	Interrupt handler (IHR)	Demand

x above corresponds to the base select code offset usually from 30, 40, 50 or 60 octal.

An additional register namely the Composite Data Channel (**CDC**) corresponding to the positional select code in the computer interface bus can be used to handle direct memory access (**DMA**) transfers of 16 or 24-bit data-words. This method of data transfer is not implemented due to the lack of available machine-time to install and test the method.

See Fig 2.3 for register layout

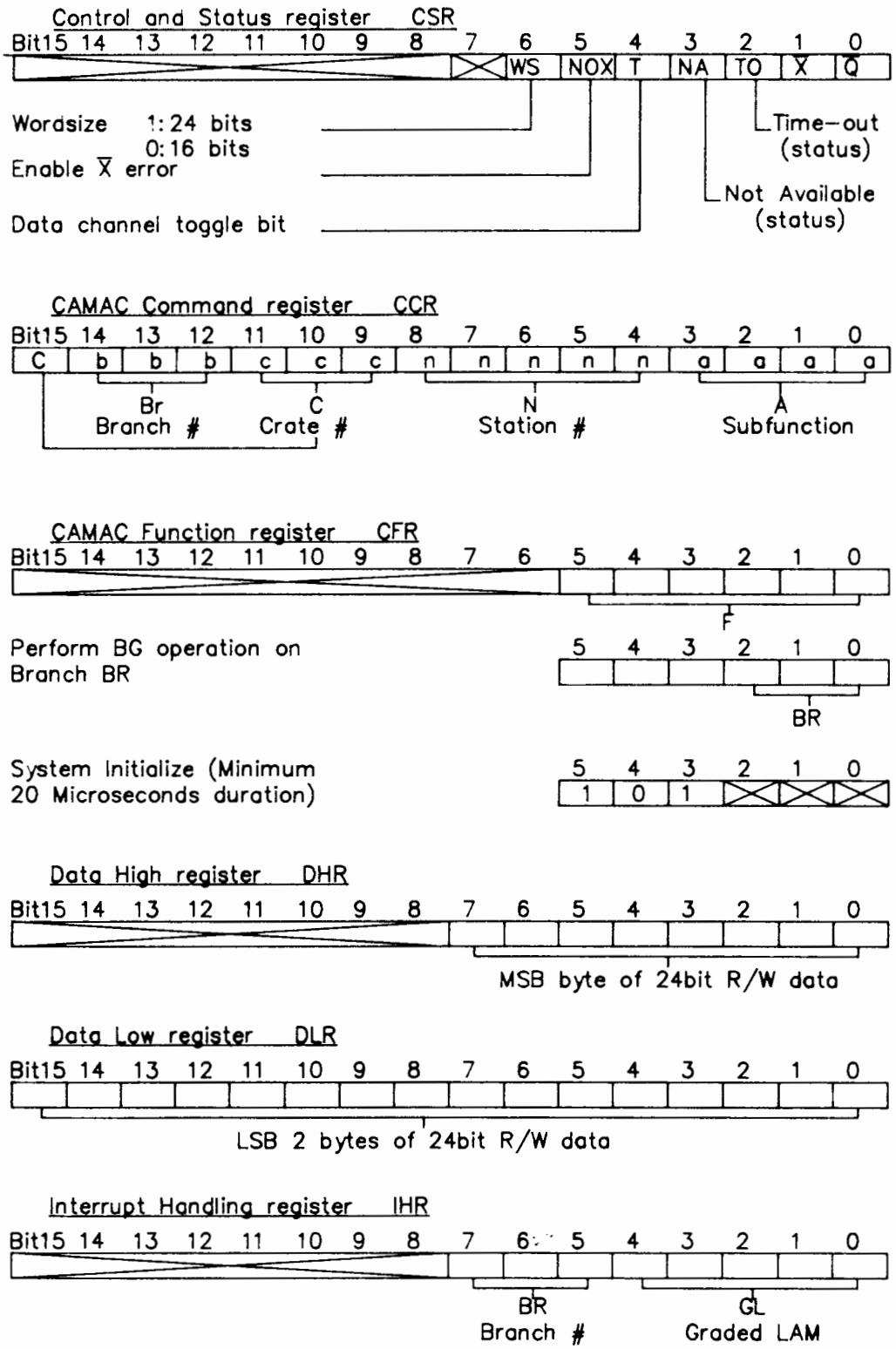


Fig 2.3 The PTI Registers

Interrupt Vector Generators (IVG) :
This module will help the computer to process CAMAC demands. The IVG autonomously accesses the source of demands to determine priority and then interrupts the computer with a pointer for the service routine.

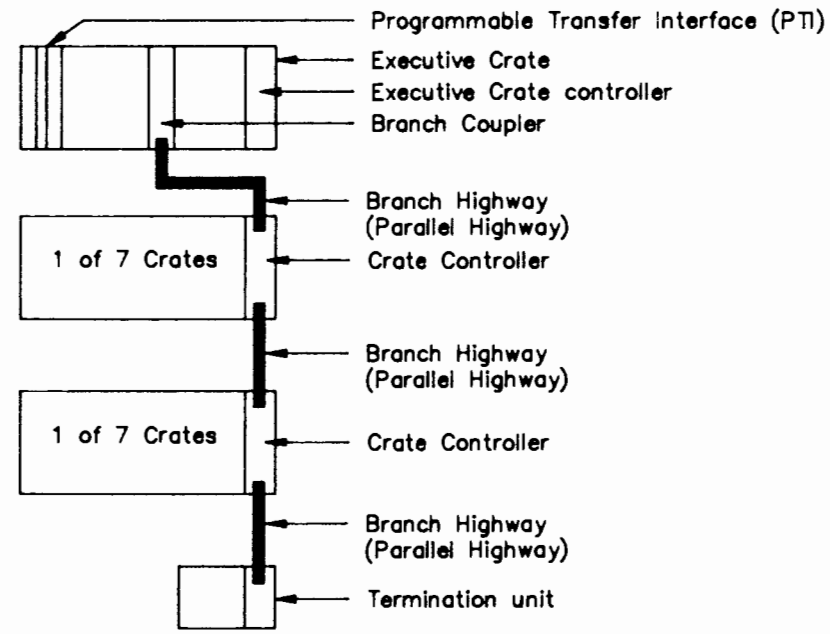
Autonomous Memory Channels (AMC):
This module uses the method of cycle stealing or Direct Memory Access (DMA) to transfer data to or from computer memory and CAMAC autonomously.

Autonomous Control Units (ACU):
This module can be used to manually control a system without a computer. It is not used in our system.

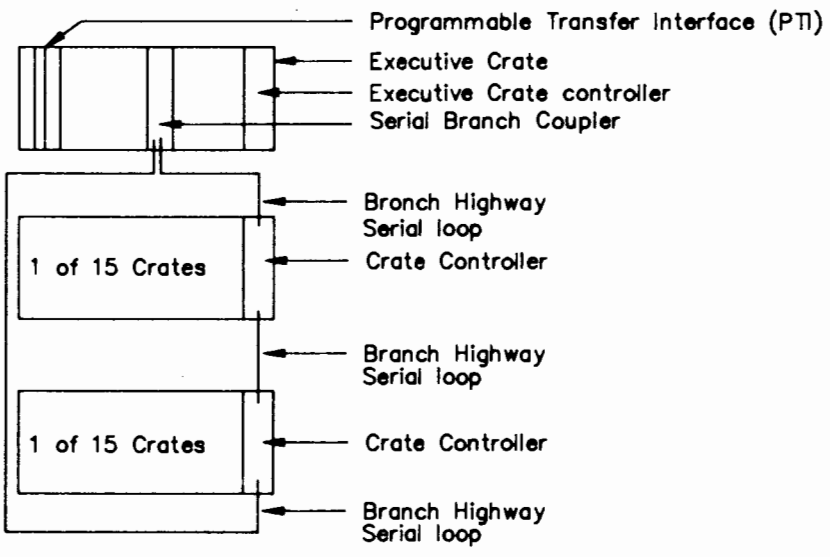
Each of the parallel branches can accommodate 7 crates while the Serial branch can accommodate 15 crates. Up to 23 stations can be addressed in each crate. In each crate a crate controller connects the crate to the branch cable and Branch Coupler residing in the Executive crate.

The parallel branch consists of parallel Branch Couplers resident in the Executive Crate and is connected to all remotely located parallel branch CAMAC crates in the system by means of Differential Branch Extenders (DBE). Each crate in the system contains a Crate Controller (CC) which receive commands from the parallel bus and transmits data and responses from modules to the Executive Crate.

The Serial branch uses a Serial branch coupler with this coupler connecting to remote crates via a limited number of lines for installations where very long highways are needed and cost plays a role. The highway is configured as a serial loop starting at the Executive crate, routed through each remote crate's serial crate controller and returning to the serial coupler in the executive crate.



The CAMAC Parallel branch



The CAMAC Serial Branch

Figure 2.4

All CAMAC instrumentation modules for instance 24 bit Input-Output modules, Analog-to-Digital converters, Stepper-motor control modules etc. adhere to strict specifications according to CAMAC standards [2] with regards to responses to CAMAC commands issued to the PTI from the minicomputer. A command to a CAMAC module contains information about the branch address, the crate address, the module's position in the crate and a function and subfunction descriptor. The function description can be one of three groups of command types namely:

a) Read data from module.
b) Write data to module.
c) Dataless transfer (usually some control command or command response read-back).

2.2 DESCRIPTION OF THE COMPUTER HARDWARE

The Computer system used to perform the control- and data-acquisition function for the Accelerator consists of a number of minicomputers connected to the CAMAC Executive crate to link the computers to a serial highway as well as several parallel branches connected to various instruments for the Accelerator. A separate mini-computer is used for the development of Control System programs. This development computer is linked to the control and acquisition system by means of a Multi-Access Disk Controller to share common disk files with the other two mini-computers.

One of the control mini-computers is used for servicing of interrupts (LAMs) generated on operator demand from the control consoles, and to display status and actual-value information on the consoles as well as programmatic access to many of the accelerator sub-systems. The other computer is used for interrupt-driven data acquisition and to graphically represent beam diagnostic information collected

from accelerator devices.

An interface card called a Multiplexer Input/Output card (provided by the mini-computer manufacturer and re-designed and improved by a colleague [3]) connects the Programmable Transfer Interface (**PTI**) to the computer interface bus via a multi-core cable. The function of this interface is to extend the computer I/O bus in order to connect to the PTI. The PTI is then treated as the equivalent of a normal computer I/O interface.

An interface card called a 'breadboard card' containing the very basic control, flag and data-buffer logic to perform the input/output actions required by a typical user peripheral device, resides in the computer interface bus structure to handle the software-generated interrupts from the CAMAC driver. This card is is positioned in the I/O bus structure so that it has the highest interrupt priority of all the peripheral interfaces. This card has no other function than to interrupt the computer when the necessary interrupt logic on the card is set as a result of a I/O instruction of the CAMAC software driver.
This concludes the description of the hardware.

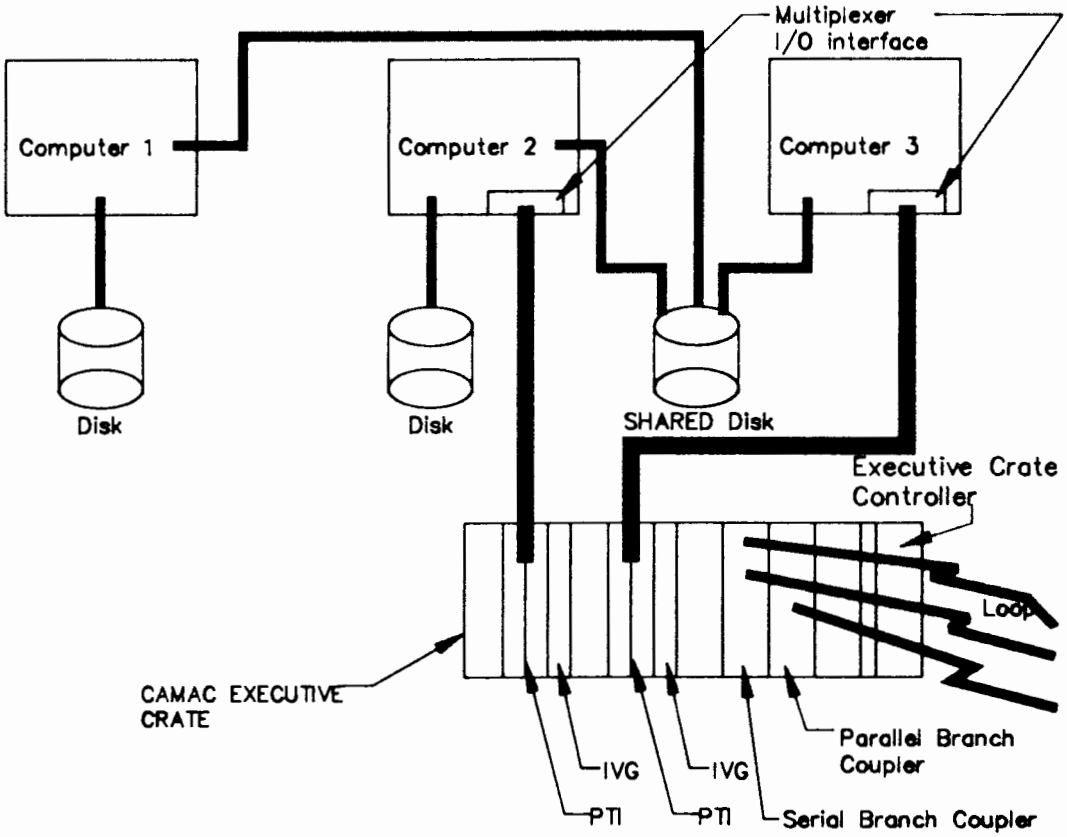


Fig 2.5 THE COMPUTER HARDWARE CONFIGURATION

CHAPTER 3

ANALYSIS OF USER REQUIREMENTS

3.1 PHILOSOPHY OF THE CONTROL SYSTEM

The Accelerator operation is controlled by operators/physicists using Control Consoles that are made to perform the required functions by programs that exist in the control computers. These programs access all the different subsystems to be controlled as well as the Control Consoles using the CAMAC crate standards. This collection of programs is called the CONTROL SYSTEM [4].

The operator consoles are used to display actual and reference value and status information regarding the Accelerator variables, to set reference values for the individual instruments on the Accelerator via the Set Point Units (SPU), to select different display pages of variables via the Page Selector, to perform certain high level control functions from the softkeys on the touch panel and to monitor actual values read back from the instruments on the analog meters in the consoles. Provision is made to support up to 5 consoles in the final system. All the 'instruments' in the console for instance the alphanumeric display, the SPU's, the analog meters, the touch panel and the page selector are connected to CAMAC modules in a crate on a branch that is connected to the Executive crate.

The Control System page display takes the form of a hierarchical tree structure with menu pages at the top of the tree containing information about the contents of the

various sub-pages. The total number of pages containing variables is 100. A typical display page contains all the information for up to 20 variables plus a section of the display used for program information messages for example any errors occurring during the control/monitoring process or additional status information regarding a particular control process. Any one of the consoles can display a page of variables including pages displayed on one of the other consoles although control of variables are mutually excluded between consoles.

The display update process will schedule programs to access the individual instruments associated with a cyclotron variable in order to read data from that instrument. The actual value read back program will then return data to the update process which in turn will pass the data, suitably formatted, to be displayed on the console and to the analog meters (4 per console). As this update process must handle 20 variables per page and all the data transfer takes place via the CAMAC bus interface connecting different crates together, the need for the fastest possible method of data transfer between the computer and CAMAC exists. Additionally the operators need to view any changes in the control parameters in Real Time without unduly long response delays introduced by the control/acquisition system.

In addition to the display update process, when an operator wishes to set a specific reference value on an instrument, it will be performed by using the SPU. This device interrupts the computer at a fixed rate of 25 per second using the CAMAC LAM mechanism. Each console contains 2 SPU's and the operators must be able to use both simultaneously. As the SPU interrupts the computer, the SPU service program must read a value from the SPU, pass it to the update process for display on the console as well as to the program

that will effect the set-up of the value in the instrument associated with the Accelerator variable. As the variable name is also written back to the SPU together with a data value reformatted to a specification contained in the database for the different variables, this will also require access to the CAMAC system. The rate of generating interrupts from the SPU's being quite high, will put a heavy load on the computer interrupt servicing mechanism.

The touch panel and page selector of each console forces demand interrupts on the Control System as the operator requests new page selection or some action from a pre-programmed softkey on the touch panel. In turn a server program on the computer will handle the demand from the operator. All of this kind of interrupts are to be handled by the interrupt service mechanism of CAMAC and the computer.

To be able to acquire data from the Beam diagnostics equipment via CAMAC, it is necessary to start a server program on an interrupt (LAM) that will command the transfer of data at the highest possible rate of transfer from the equipment using a block transfer method. As each block transfer of a small block of data will be preceded by a interrupt from the CAMAC hardware to wake up the capture program, it follows that a high rate of interrupts will also generated by this equipment. The block transfer of data must also be used to refresh the alphanumeric displays on the operators' consoles as single word data transfers on CAMAC will take a considerable time for a screen of data (64 characters by 24 lines).

The connection of all power supplies used to power the magnets for the Accelerator, will be by using the Serial Branch method. The operation of the Serial branch coupler consists of the initiation of a transfer and on successful

completion of the transfer, a completion interrupt to the computer. This method of transfer is similar to most peripheral device functioning and can be handled by a device driver. As the Serial Branch completion interrupt occurs in the Serial Branch coupler, causing a LAM, it can be identified by the IVG. A mechanism to service this specific LAM occurring in the Executive crate must tie in with the IVG mechanism to interrupt the computer. The computer must then identify that the Serial Branch has interrupted and provide a method to cause a completion interrupt for the driver as if an actual device interface has made an completion interrupt. This will be a deferred method of transaction completion.

As the user wish to access modules resident in crates on the Parallel or Serial branches without having to have prior knowledge to the different methods of access at the lowest level, the user must be provided with a software interface routine that will handle this requirement. As the access of CAMAC modules normally consists of passing a CAMAC function or command and some data value to or from CAMAC, the user will specify a CAMAC module address, the CAMAC function code, and the address of the data word(s).

To summarize all the requirements, the following types of access methods must be provided.

1. A very fast direct input/output access to CAMAC from the computer. Block transfer of data as well as single or double word CAMAC data transfers must be provided.
2. A method to service interrupts with the minimum of delay from the computer operating system. This method must also be used to service the Serial Branch completion interrupts as well as the demand interrupts from the operator consoles and

interrupts occurring asynchronously from instruments.

3. Serial branch transfers must be done via the normal operating system input/output requests.
4. A universal method to access all CAMAC modules in Parallel or Serial branch crates must be provided.

CHAPTER 4

THE IMPLEMENTATION

4.1 METHODS OF CONNECTION

In order to provide CAMAC crates containing instrumentation modules at the necessary instrumentation areas, an extensive system of cabling was installed. All Cyclotron power supplies are housed in the power supply hall. To connect the CAMAC control modules for the power supplies, the serial branch system was used. This is a daisy chain configuration which starts and ends at the CAMAC executive crate Serial Branch driver and links CAMAC crates in the power supply hall together. For the other instrumentation areas, the parallel branch system was used. All parallel branch driver modules are coupled via Differential Branch Extenders (**DBE**) to remote crates containing the complementary DBE's before feeding to the crate controllers.

4.2 IMPLEMENTATION POSSIBILITIES

The normal way to effect Input/Output to a device is via the operating system EXEC calls. These system procedures provide an interface between the device by using a device driver resident in the operating system area and using certain system tables like the Device Reference Table (**DRT**), the Equipment Table (**EQT**)(Refer to [8]) and the Interrupt Table (**INT**)(Refer to [17]). According to the Operating system specifications [5] an unbuffered 1-byte Write to a terminal takes 1,228 milliseconds from the EXEC call to the driver, Ø,735 milliseconds to exit from the driver to the system, and 77 microseconds from the interrupt to the driver on

completion. This adds up to a figure of 2,Ø4 milliseconds for a typical write EXEC call. The time spent in the driver to perform the write is not included in the figures above. The system overhead in the case of an EXEC call for I/O makes this method too slow for CAMAC transfers as the applications programs would not perform in 'Real time' as expected. A much faster and preferred method to access the CAMAC hardware from a user program is to use a 'PRIVILEGED SUBROUTINE'. This allows an applications program to do direct I/O to the specific device without causing a Memory Protect interrupt that will abort the program. Every application program must use these routines to interface to the CAMAC hardware. A small set of subroutines are created using the HP1ØØØ Macro Assembler. The method is recommended by the minicomputer manufacturer for fast access of the I/O system without crashing the operating system. In order to provide a standard procedural interface to access the Parallel and Serial Branches, routines with similar calling parameters for both types of CAMAC branches must be provided. This is necessary because the database containing the Cyclotron Control System variables information, specifies a CAMAC address for the variable specific instrumentation hardware. This hardware address is handled in the same way for the Serial or Parallel Crates by the user programs. The lowest level subroutines accessing the CAMAC Executive must distinguish whether access must be routed to the Parallel or Serial Crate Branch couplers.

To service demand interrupts from the CAMAC system, the operating system interrupt handling mechanism must be instructed to jump to and execute code contained in the special software driver loaded as part of the operating system kernel during system generation. This method of servicing interrupts bypasses the operating system overhead and is called a 'PRIVILEGED DRIVER'. It is recommended by the manufacturer in being the fastest method of interrupt

service for high speed data transfers. In order to service demand CAMAC interrupts (LAM's) by service programs existing outside of the operating system kernel, these programs must be scheduled from within the driver. A valid mechanism to implement this concept does exist as part of the operating system. However, this method might prove to be too slow due to the operating system scheduling algorithm for scanning schedule queues. Thus a faster method to get a service program running will be implemented.

The transfer of data to or from the Serial Branch will be handled by the CAMAC driver using the normal operating system method of EXEC calls for I/O. This decision is made based on the expected access rate for reading power supply actual value information back to the Cyclotron Control System. This rate is to be once per second for each power supply linked to a operator display page - a relatively slow rate as far as computer I/O cycles are concerned. The Serial Branch transfers take place via a Serial Branch driver module issuing a hardware transaction command to the remote Serial Branch Crate controller. On successful data transfer, the Serial Branch coupler receives a hardware handshake to indicate this action is completed. This in turn causes a completion interrupt to the CAMAC Executive system that must now inform the CAMAC driver of the hardware completion of the transaction. By making use of an indirect interrupt method the Serial Branch hardware interrupt is converted to signal the completion of an I/O request in the driver completion section.

Provision must be made in both the driver and privileged subroutines for error reporting to the operating system and/or the user programs. The recommended method of error reporting in the CAMAC driver's Initiation and Completion sections is via the A and B registers. On return from the driver to the Operating system, the operating system error

handling section recognizes the presence of an error condition and reports it accordingly. In the Privileged driver section of the CAMAC driver, a mechanism to schedule an error reporting service program will provide the necessary reporting mechanism. The method to be used in the privileged subroutines will be by writing an error message to the standard output device (normally the system console) as well as setting a status value for the user program to perform the necessary remedial actions.

4.3 THE IMPLEMENTATION

To be able to access CAMAC modules on the Parallel Branch, the privileged subroutine method was developed. This method did not allow any interrupt servicing. The interrupt servicing mechanism was implemented by writing a special device driver for CAMAC resident in the Operating system kernel. The Serial Branch transfer method was implemented by special processing in the CAMAC device driver. Finally the device driver was modified to handle only interrupts qualifying for service as indicated by table entries internal to the CAMAC driver. This was necessary as two computers were later connected to the Executive crate and both had to service interrupts.

4.3.1 The Parallel Branch system

In order to achieve the quickest data transfers, all parallel branch transactions will be handled by 'PRIVILEGED SUBROUTINES' (see APPENDIX E). This method is the recommended way to access Input/Output interfaces connected to the mini computer's I/O bus without the extra overhead of a device driver.

The first routine that was implemented was intended to transfer only 16-bit data to or from the CAMAC system

(Single Integer CAMac transfer - SICAM). The principle of operation of this routine is to get the parameters, switch to the privileged mode (the interrupt system is switched off so that any I/O instruction execution will not cause a Memory Protect Interrupt to abort the user program; direct user I/O instructions are illegal, system EXEC requests must be used for device I/O), issue the relevant I/O instructions, switch to non-privileged mode (interrupt system switched on), build the returned status information words and return to the calling program. In case of errors occurring, the routine will switch back to non-privileged mode before building the returned status words and also report the error via a message on a terminal using the Operating system terminal read/write routines. It is essential that the interrupt system be on before any terminal I/O can be processed.

The block structure of the body of the subroutine is:

Determine if Parallel or Serial Branch transaction

For Parallel branch:

Switch to privileged mode.
Set up Flag Flip-flops,status and CAMAC command registers.
Determine type of transfer
For Write:

1. Output CAMAC function to the Function register in the PTI
2. Output the 16-bit data word to the data channel and start the CAMAC cycle.
3. IF Function Complete Flag is set THEN
   IF Error Flag is set THEN
   ReportError
   ELSE
   Get status from Status Register,enter non-privileged mode and return to caller

```
    ELSE
      IF Error Flag is set THEN
        ReportError
      ELSE
        IF watchdog timed out THEN
          Set status to time-out and and ReportError
        ELSE
          goto 3.
```

For Read:

1. Output CAMAC function to the Function register in the PTI and start CAMAC cycle.
2. IF Function Complete Flag is set THEN

```
      IF Error Flag is set THEN
        ReportError
      ELSE
        Get data from Data register and pass to user
        buffer, get status, enter non-privileged mode
        and return to caller
    ELSE
      IF Error Flag is set THEN
        ReportError
      ELSE
        IF watchdog timed out THEN
          Set status to time-out and ReportError
        ELSE
          Goto 2.
```

For Dataless transfers:

1. Output CAMAC function to the Function register in the PTI and start the CAMAC cycle.
2. IF Function Complete Flag is set THEN

```
      IF Error Flag is set THEN
        ReportError
      ELSE
```

```
        Get status from Status Register,enter non-
        privileged mode and return to caller
    ELSE
      IF Error Flag is set THEN
        ReportError
      ELSE
        IF watchdog timed out THEN
          Set status to time-out and and ReportError
        ELSE
          goto 2.
```

For Serial Branch:

```
  IF FirstTime THEN
    InitializeParameters, Indicate Serial Br Transfer
    Set up parameters for an EXEC call to CAMAC driver
  Call EXEC
  IF errorReturn THEN
    ReportError
  ELSE
    Set status,enter non-privileged mode and return  to
    caller
  END
```

A section to report errors is included in the subroutine. As the subroutine (SICAM) was written in Assembler for the HP1000, very tight coding was used for this. It would have been possible to make the error reporting section as an external subroutine but as the method of testing for the specific status conditions is done on a basis of bit-by-bit testing of the status register contents, continual reloading of a temporary saved status value would have slowed the execution of the subroutine down. As the routine is being used very often by all processes communicating with CAMAC, it would be counterproductive to use the slower method. In the case of a transfer error or a time-out error or even the case where a CAMAC module does not accept the command, a

brief message is written to the users session terminal from where the user application is run or the system console in case the application program is run in non-session mode. Mention must be made of the short formatting routine called to format the integer values in error messages. This routine (also written in Assembler) was developed by J.A.M. De Villiers [6] and was chosen as it was specially written to execute quicker than the operating system's routine to format integers to ASCII.

According to the CAMAC instrumentations standards [1], each CAMAC module responds with a Q and X-response signal. The purpose of the Q-response is to signal to the user the status of any selected feature of the module. The purpose of the X-response is to signal to the user if a CAMAC command was accepted (by the module) for instance if X = Ø the hardware malfunctioned, a module is not present or powered or does not perform the CAMAC function requested. These signals are always returned to the calling program via the Q and X-response parameters. The actual hardware status of each is used to build the proper response. The applications program will then have a way of determining if a CAMAC transaction was successful or have to be retried, rejected or reported.

A description of the usage of the subroutines will be found in the appropriate Programmer's Reference Manual (included).

The next routine implemented was a 24-bit data word transfer routine (DoubleIntegerCAMac - DICAM) using a similar technique. The same principles as before was used with the exception that the address of a 32-bit integer containing the 24-bit CAMAC data-word in the LSB part of the 32-bit integer is passed to the routine. Internal in the routine, the 24-bit data is split into most significant 8 bits and output (or input) to the Data High Register (DHR) and the

least significant 16 bits is output (or input) to the Data Low Register (**DLR**).

```
Bits
31 ..................Ø  ===> 23......16 + 15.............Ø
     32-bit integer       ===>     DHR     +         DLR
```

A routine to transfer 16-bit data words in block mode was the next to be developed. Essentially it has the same structure as the previous two routines as far as the initial, completion and error reporting sections is concerned. Sections which reflects a change are namely write, read and data-less transfers. These sections consist of the basic I/O functions as described above at the inner level and an outer level which contains a loop for the block length of the transfer. The block transfer subroutine (**BL**ock **CAM**ac) therefore goes into the privileged non-interruptable mode and stays in this mode as long as it is looping to transfer all data words to or from the user buffer. Using this routine to transfer data is much faster than the previously discussed ones as the continual context switching from non-privileged to privileged mode and back to non-privileged for each transfer of data is avoided and is only done once for every block of data to be output.

It must be pointed out that the method of 'PRIVILEGED SUBROUTINES' also have its limitations as regarding the total time that the routine can go privileged. A recommended total time for going privileged is specified as 1 millisecond [7]. This implies that the block transfer routine can only transfer a block of prescribed length. Actual timing of the routine's inner block CAMAC transfer times (per data word) came to 25 microseconds. This allows a maximum total block length of 40 CAMAC data words per block transfer. As the current hardware configuration allows two minicomputers each to be able to transfer blocks of

data simultaneously and the one computer can be prevented from transfer completing by the other accessing the CAMAC system, it would not be unreasonable to halve the total maximum block length for each computer for example limited to 20 words per block. This will ensure that the routine will not exceed the recommended figure of 1 millisecond for being in the privileged mode for the worst case condition when both computers are executing block transfers.

A routine similar to the 24-bit transfer routine 'DICAM', but using a separate parameter for each part of the full CAMAC address (branch, crate, station number and subfunction) was next developed. This routine will internally compile a compound CAMAC address as required by the CAMAC Command Register (CCR) and then execute code in the same manner as for 'DICAM'. This routine was called 'NICAM' and was built mainly for convenience of use for certain program calls which did not have the compound CAMAC address available but rather use the Branch, crate, station number and subfunction directly in the call. A separate routine 'DECLR' is also provided which will compile and return the compound address given the separate Branch, crate, station number and subfunction. This routine was mainly called from FORTRAN programs as the PASCAL application programs allowed a datatype declaration which would allow a compound address definition without having to resort to external routines for the building of the address word.

This concludes the implementation for the Parallel Branch system as far as the non-driver software is concerned. Descriptions of all routines can be found in the Programmer's Reference Manual.

4.3.2 A General device driver description

4.3.2.1 The I/O Initiation section

When a user program makes an EXEC call to start I/O transfers, certain parameters like logical unit, control information, the location of the user buffer, the length of the buffer as well as the type of the request (Write, Read or Control) are passed to the Real Time Executive (**RTE**). This information will then be passed on to the Input/Output Control (IOC) module of RTE by the RTE request processor [8]. The IOC will check the request for validity and reject the call if errors are found. Otherwise the logical unit number will be used as an index into the system Device Reference Tables (**DRT**) to establish which I/O controller (indicated by the Equipment Table (**EQT**) entry number (see APPENDIX A)) and device are being referred to. This I/O request is linked into the request list for the particular I/O controller.

When the controller is available (no other I/O requests pending), the above mentioned parameters are placed in the associated EQT entry. The addresses of the EQT entry words are placed in the Base Page Communications Area of the Operating system. In the CAMAC driver's case the System map is enabled and the initiation section of the CAMAC driver is called. This section then initializes the CAMAC hardware (device controller), starts the CAMAC data transfer or performs the requested control function. The control function entails re-initializing the hardware, disabling the driver from servicing LAM's, entering a server program name in the interrupt service table (internal to CAMAC driver), arming or disarming the interrupt service mechanism and removing the server program name from the interrupt service table. Execution is returned to the IOC module which returns to the RTE dispatcher (process scheduler) to start execution

of highest priority program that is in the schedule list. If the initiation section has successfully started the data transfer, the hardware will concurrently perform the transfer.

4.3.2.2 The I/O continuation/completion section

When it has finished a data transfer (usually a 16-bit word), the CAMAC hardware controller in the PTI will interrupt the computer. An interrupt will force a jump to an interrupt trapcell (these trapcells are located at the lowest addresses in the base page of the operating system) which contains an instruction. This instruction will be executed and in the case of the CAMAC driver, this contains an indirect call to the system's Central Interrupt Controller (CIC) module [8]. The module CIC obtains the select code of the interrupting controller from the Central Interrupt Register in the computer hardware. It uses this select code to index into the Interrupt table of RTE. This table will contain the address of the EQT entry for the interrupting select code. CIC looks at the EQT entry and determines which driver is to be used with this interrupt. The system or user map is enabled and the particular driver's continuation/completion section is called in order to service the interrupt. For a Read operation, the data is accepted from the device, and restarts the device if more data is to be read. For a write operation, it will send more data to the device and restart it. The driver returns to the CIC with a code to indicate if more interrupts are to be expected, or in the case when the requested number of words were transferred, a code to indicate that no more interrupts are expected. CIC then transfers control to IOC to terminate the I/O process. IOC will place the program that made the original I/O request back in the scheduled list for program execution continuation. It then checks to see if any other I/O requests are pending for this

controller and if at least one request is queued, the initiation section of the driver is again called to start the next transfer. The IOC module then returns to the scheduler to dispatch the next highest priority program for execution.

4.3.3 The CAMAC device driver

This driver was developed for interfacing the CAMAC system to the operating system according to prescribed rules for Driver writing [8].

The CAMAC device driver (**DVR46**) consists of three sections namely:

1. The I/O initiation section.
2. The I/O continuation/completion section.
3. The Privileged section. This section will be dealt with in paragraph 4.4 - Servicing demand interrupts.

The main usage of the Initiation/Continuation sections of the driver is for the Serial Branch transfers. To interface the user programs to the Serial Branch hardware it was necessary to use the same software interface as for the Parallel branch except that instead of going privileged before transfer of data, an operating system I/O request to the CAMAC driver to read or write data was used.

4.3.3.1 The Initiation section

4.3.3.1.1 Initialization of the driver

On the first entry of the driver's initiation section after the computer is booted, the start address of the EQT for the CAMAC logical unit is stored for internal use. The I/O request code is picked up from the EQT and checked if initialization is required. If not, an error status code with bit 7 set is stored in the 5th word of the EQT and the error exit return is made. The operating system error handling mechanism will respond with a suitable message on the system console. If initialization is required then the position of the IVG module in the Executive crate is extracted from the 7th word of the EQT and validated. If not valid the error exit is taken. If valid and bit 15 on the EQT word 6 is set (indicating exclusive Serial Branch transfer), the driver will proceed to initialize the Serial section. If not set, then the initialization for the driver's privileged section and the IVG hardware will be done.

A check is made whether this is a first-time initialization call. If this is so, the table in the driver containing the IDSEG addresses of interrupt service programs [22] will be cleared to zeros. The trapcells in the system map for the driver's privileged entry point will then be saved at a temporary storage location in the driver. An instruction to 'jump to subroutine' to the driver's privileged section will be placed in the trapcell. This method will bypass the overhead of the central interrupt controller (CIC) to achieve a very quick service of an interrupt (see section 4.4). Certain retry counters are preset and a boolean variable to indicate that first-time processing has been done, is set. The sequence following is executed in the

first time call and all subsequent calls.

An internal routine to set the CAMAC register flags on the PTI is executed. These registers and flags are described in Chapter 2. The Dataway Inhibit signal in the Executive crate is then removed by issuing the relevant CAMAC function to the Executive crate controller [9]. The System Crate Demand (SCD) is then enabled in the same way as for the Inhibit signal. The IVG trapstore for all possible trap locations (255) will be cleared by issuing the Reset Trapstore CAMAC command to the IVG [10]. Then the IVG interrupts are enabled and IVG Graded LAM (GL) cycles will be enabled.

The time-out value for the driver is set (via the 14th word of the EQT) to 100 clock ticks (100 * 10 milliseconds = 1 second). The status word (word 4 in EQT) is loaded with a zero to indicate normal completion and the A-register loaded with value 4 to indicate immediate completion (operating system must not expect a completion interrupt) to the IOC. Return is then made to the IOC.

4.3.3.1.2 Initialization for the Serial Branch

On the first entry of the driver's initiation section after the computer is booted, the start address of the EQT for the CAMAC logical unit is stored for internal use. The I/O request code is picked from the EQT and checked if initialization is required. If not, an error status code with bit 7 set is stored in the 5th word of the EQT and the error exit return is made. The operating system error handling mechanism will respond with a suitable message on the system console. If initialization is required then the position of the IVG module in the Executive crate is extracted from the 7th word of the EQT and validated. If not valid the error exit is taken. If valid and bit 15 on the EQT word 6 is set (indicating exclusive Serial Branch

transfer), the driver will proceed to initialize the Serial section.

This section of the driver picks up the position of the Serial branch coupler receiver station from word 7 of the EQT. It then builds the necessary CAMAC commands control of the Serial Branch Coupler (**SBC**) by the driver during initialization and the normal execution of the driver. An internal routine to set the CAMAC register flags on the PTI is executed. These registers and flags are described in Chapter 2. The SBC is then programmed for split mode operation [11]. After clearing any outstanding LAM's on the SBC, it is then programmed to enable interrupts for all functions of the SBC. The serial initializing section is now complete and return is made to the main stream of the driver's initialization section. This part will set the time-out value for the CAMAC driver in word 14 of the EQT for use by the system. A boolean variable indicating completion at first-time processing, is set and the driver then returns to the IOC with the error status indicating no error (successful initialization of driver).

4.3.3.1.3 The body of the Initiation section

The driver entry for any subsequent I/O EXEC requests will isolate the I/O request code from the 6th word of the EQT. This will be a CAMAC Read/Write/Dataless function or a driver Control request. If it is a CAMAC transaction, the retry counter is preset. A test for Serial Branch operation is made and if true then on the SBC module, any outstanding LAM's are cleared and the LAM mask is enabled. The CAMAC Function code is loaded via the EQT word 1Ø. If out of range (not between Ø and 31), then the error exit is taken. The transfer method (read, write or dataless function) is determined according to the CAMAC Function code[see Appendix

B] specified in word 1Ø of the EQT. The three modes for CAMAC transfer will now be discussed.

READ:

The CAMAC BCNA module address [See APPENDIX C] will be converted to the required form for the PTI hardware. A read function from the specified CAMAC module is started using the internal CAMAC I/O routine 'RCAM'. A test for the Serial branch flag is made to determine whether received data must still be passed to the user buffer. In the case of the Serial branch this data would only become available at the receipt of the hardware completion handshake by the SBC from the Serial loop and will be processed by the driver Completion section (see 4.3.3.2). The driver will then return control to the IOC with an indication that a continuation/completion interrupt is still expected.

WRITE:

The user buffer length and address are loaded from EQT words 8 and 7. The X-register is loaded with the MSB byte of the 24-bit CAMAC data-word if a 24-bit transfer is requested by the length word, otherwise the X-register is cleared. The Y-register is loaded with the least significant 16-bit CAMAC data-word pointed to by the address of the user buffer. The BCNA word is loaded via EQT word 9 and converted to the hardware format. The CAMAC Function code is loaded via the EQT word 1Ø and a CAMAC write operation started using the internal routine 'WCAM'. On return the CAMAC status as indicated by the CSR is loaded into the EQT word 5 and the driver returns to the IOC with the immediate completion code in the A-register. This completion method indicates to the system that no interrupt for transaction completion is expected and that the dispatcher may schedule the user program to continue after the I/O suspension.

DATA-LESS:

The BCNA word is loaded via EQT word 9 and converted to the hardware format. The CAMAC Function code is loaded via the EQT word 1Ø and a CAMAC data-less operation started using

the internal routine 'DLCAM'. On return the CAMAC status as indicated by the CSR is loaded into EQT word 5 and the driver returns to the IOC with the immediate completion code in the A-register. This completion method indicates to the system that no interrupt for transaction completion is expected and that the dispatcher may schedule the user program to continue after the I/O suspension.

On return from each of the 'xCAM' routines, a check is made to see if an error has occurred during the I/O transfer. The driver will in each case repeat the I/O transfer internally for a number of times equal to the retry counter contents (preset at the start of this section). If it fails for all of the retry efforts, the driver follows the error exit path, reports the status via the EQT word 5 and passes a driver completion code to indicate the equipment malfunction to the IOC.

For a description of the internal CAMAC I/O routines, see APPENDIX D.

4.3.3.1.4 The Control section

This section of the driver will be entered when the RTE I/O request control code is specified in an EXEC call (code = 3). The driver is entered from IOC as described in the Initiation section. On checking bits Ø and 1 of word 6 of the EQT and finding the control request code, transfer is made to this section. The control function code is determined bits 6 to 9 of word 6 of the EQT table. There are 7 valid control functions in the CAMAC driver. These are :

Control code	Action
Ø	Clear the interface (RTE requirement).
1	Re-initialize the Executive crate.
2	Disable interrupt servicing.
3	Enter interrupt service program name in internal driver table.
4	Disarm interrupt service program.
5	Arm interrupt service program.
6	Remove program identification address from internal table.

Code Ø

This code will force the driver to clear the interrupt flip-flops on the Interrupt Handling Register (IHR) as well as on the Serial Branch Interrupt Handling Register (SIHR). The driver then returns to IOC with an immediate completion code in the A-register. All the other control request code exits will return in this way.

Code 1

This code will force the driver to execute the code to re-initialize the CAMAC hardware and on first time entry the internal tables of the driver will be cleared. A detailed explanation of this process is contained in 4.3.3.1.1 and 4.3.3.1.2 . The return path is the same as for the Initializing sections.

Code 2

This code will reset the first-time flag in the driver, clear the internal table containing the ID segment address (IDSEG) of the service programs (see section 4.4), and restore the patched trapcells in the user or system map trapcell area of the operating system to the values before the CAMAC driver was initialized. These values are stored in the driver temporary storage area at initialization. The driver then returns to the IOC as described above.

Code 3

This portion of the control request section takes the program name stored in a buffer whose address is stored in word 7 of the EQT and searches through the ID segments attached to the Keyword list for the ID segment address for the associated program and when found, places this IDSEG address into word 13 of the EQT. Also contained in the buffer after the program name, are three words containing the CAMAC Branch number, crate number and module station address of the module capable of generating a LAM to be serviced by the CAMAC driver. This address is used to build a pointer to an internal table (INTAB). The service program IDSEG address as found above, will be stored at this table address. Return to the IOC is made as described before.

The driver internal table (INTAB) will contain all the service programs' IDSEG addresses to enable the privileged section of the driver to schedule these programs on the occurrence of a LAM. The assumption is made here that multiple I/O control request calls with the necessary program names and CAMAC addresses are executed from a user program (see Programmer's Reference Manual - Program EXIP).

Code 4

This control request code will pick up the CAMAC module address from the 7th word of the EQT, and form a pointer to the associated entry in INTAB. The IDSEG address at that entry will then be complemented to indicate a disarmed service program when used by the privileged driver section. Return to the IOC is as described above.

Code 5

This code will function in exactly the same way as code 4 but the effect will be to re-arm the service program as indicated in the table.

Code 6

This control request code is used to remove the IDSEG address of an interrupt service program from the table INTAB. The same mechanism to calculate a pointer to the table is used as for codes 4 and 5. The table entry is then cleared to zero. This value will cause the privileged driver to reject any LAM's occurring from the specified CAMAC module. After clearing the table entry, return is made to the IOC.

Any illegal control requests will be reported to the user program making the control request via the status word (word 4) of the EQT.

4.3.3.2 The Continuation/Completion section

This section of the driver is called whenever an interrupt from the PTI is recognized. As all the EXEC I/O requests for the CAMAC device are for Serial Branch access only, a special method is used. The privileged driver section on recognition of a Serial Branch transfer completion makes a software interrupt on a dummy interface of the highest priority. The address of the CAMAC device's EQT is stored in the Operating System Interrupt Tables for the dummy interface. This will cause the system processor CIC to issue a clear flag instruction to the interrupting select code and transfers this select code into the A-register. CIC then sets up the system map needed for this call as the driver is in the System Driver Area (**SDA**). The driver is then entered by executing a 'jump to subroutine' to the continuation/completion section of the driver (address is in the system entry point C.46) [8]. The format of the driver call is :

Location	Action
P	JSB C.46 (JSB=Jump to Subroutine)
P+1	Completion return from C.46
P+2	Continuation return from C.46
P+3	Not used for this driver

When the driver is entered at the continuation/completion section, a check is made to see if entry was caused by a spurious interrupt. This is done by checking the value at EQT word 1. A zero means that no I/O request was pending for this driver and the interrupt will be ignored by doing a continuation return to the IOC.

If the interrupt is valid, the driver proceeds and will do a CAMAC read from the Serial Branch Coupler module (using the internal CAMAC I/O routines APPENDIX D). On return from the Read routine, the status of the transaction is contained in the Y-register and is temporarily stored in a local variable for return to the operating system on exiting the driver.

The path of the code to be executed next is determined by the CAMAC function code as contained in word 10 of the EQT. For a CAMAC Write or Dataless function a check is made on the buffer length so that the status word can be stored at the correct position in the user program. As only 1 or 2 word data buffers are allowed for this driver access method, the status word is stored in the next word or the next word+1 after the data buffer. As these functions have already transferred the data words during the driver initiation section, no data is returned to the user program. If the CAMAC status word does not indicate any errors, the normal completion return for drivers is taken by clearing the A-register and jumping to the completion return point of CIC. In case of errors occurring during the CAMAC transfer,

the A-register is set to 4 to indicate a time-out error and a completion return is made. The reason for using the time-out message indicator rather than a device Not Ready indicator, is to distinguish between errors occurring during the initiation section (a device Not Ready message indicated) and the completion section. The indication contained in the A-register after returning from a driver is pre-defined by the operating system requirements and can be:

A = Ø	The operation was successfully completed.
A = 1	Device or controller malfunction or not ready.
A = 2	End-of-tape or End-of-information.
A = 3	Transmission parity error.
A = 4	Device time-out.

When the CAMAC function was a read operation (in word 1Ø of the EQT), another CAMAC read is issued to pick up the data from the Serial Branch coupler. If the buffer length is 1, only the LSB 16 bits of the data contained in the Y-register is stored in the user buffer. The temporary status word read earlier is then stored at the next word in the user program. If the buffer length is 2, then the MSB 16 bits contained in the X-register is stored in the first word of the user buffer area and the LSB 16 bits in the Y-register stored in the second word. The previous status word is stored at the next word in the user buffer area. The same code path as for the CAMAC Write or Dataless functions is then followed to return to the CIC.

This concludes the driver description as far as the normal non-privileged section is concerned.

4.4 SERVICING DEMAND INTERRUPTS

This section deals with the privileged section of the CAMAC driver. The need for interrupt servicing from CAMAC devices generating interrupts at very high rates as well as the requirement of servicing the Serial Branch hardware completion interrupt, required the development of this driver section.
More detail about the principle method can be found in [8].

4.4.1 The privileged driver section

An interrupt server for a CAMAC interrupt (LAM) would consist of a sequence of CAMAC accesses as well as some program logic to handle the required service. Many different CAMAC modules are used in the Control System each of which a possibly require a different interrupt service routine to handle them. As the space allocated for each driver in the operating system driver area is limited, it is necessary to provide for a high level interrupt handler external to the CAMAC driver. This can be achieved by creating programs running in a background or real time memory partition to perform the required actions. These programs are then developed in a high-level language and can also be maintained equally easily. A mechanism is provided to do the scheduling or start execution of these programs (interrupt handlers) from the CAMAC driver.

The interrupt processing method must have the following properties :

The ability to recognize a CAMAC interrupt immediately regardless of what other RTE operation is in progress.

The system overhead associated with servicing the interrupt must be kept to a minimum.

As discussed in the driver initialization, the trapcell for the CAMAC device will contain a JSB $JP46,I instruction. The entry point $JP46 contains the address of the privileged CAMAC driver (P.46). When a privileged interrupt occurs, the current operation in progress in the computer is suspended, and the execution is transferred to the driver at the start of the privileged section via the instruction contained in the trapcell.

The privileged section must in addition to the required task of handling the CAMAC LAM, also perform several housekeeping functions that would normally be performed by the CIC. This entails the saving and restoring of the computer state on entry and exit as well as disabling of the DMA completion interrupts for the computer in order that this driver will not be interrupted while processing.

A summary of the operation of the privileged section follows:

On entry, disable the computer interrupt system.
Disable the DMA completion interrupts.
Save the computer state (various registers).
Save the status of the Memory Protect flag
Identify and store the interrupting CAMAC module address.
Disable the LAM generating source.
Pick-up the server program IDSEG address associated

with this LAM.
Establish that the program can be scheduled.
Store the program IDSEG in system Base Page Interrupt table.
Force a software interrupt on the special dummy interface in the computer I/O structure.
Reset the CAMAC interrupt handling hardware.
Restore the computer state (various registers)
Restore the status of the Memory Protect Flag.
Turn on the interrupt system.
Restore the status of the Dynamic Mapping System and return to point of interruption.

After the interrupts are disabled and the machine state has been stored, the driver identifies the location of the CAMAC module generating the LAM by obtaining the Branch and Graded LAM address from the Interrupt Handling Register (IHR) using a normal computer I/O instruction. A CAMAC function to read (via RCAM) the interrupting LAM module's station number is issued to the crate as identified from the IHR contents. The station number is validated to exist in the range 1 to 22. Any errors occurring during the driver execution are handled by an error reporting section within this section. A compound module address consisting of the Branch number, Crate number, module station address and a zero subfunction code, is built from the information gathered. This address is converted to the new format for the Executive crate as described in APPENDIX C and stored in the first word of a 5 word table at a fixed address directly before the actual start of the CAMAC driver. This 5 word table is a communications area provided for transferring of data, status and error codes to the interrupt service programs external to the operating system.

The compound module address as found above is then converted to an index to the internal table ('INTAB') containing the

IDSEG addresses of all the server programs armed for interrupt service. The complemented IDSEG address of the program associated with this LAM is stored in the interrupt table in the system Base page.

A unique technique of deferred scheduling of the high level server programs is implemented here. A 'software interrupt' is forced on a dummy interface card in the computer's I/O bus structure by setting the control and flag flip-flops on this card. As the interrupt system is disabled at present, this pending interrupt will only be serviced as soon as the interrupt system is re-enabled. The operating system will immediately service this interrupt as the dummy card occupies the highest priority interrupt slot in the I/O interface. It will examine the interrupt table in the Base page and find the negative IDSEG of the server program there. An operating system rule specifies that if a negative number occurs in the interrupt table, it contains the IDSEG of a program to be scheduled immediately.

The CAMAC interrupt mechanism in the IVG is now reset and enabled using CAMAC functions via the internal routines 'WCAM' and 'DLCAM'. The Memory Protect flag is loaded and checked to establish if it was on before entry to the driver. In case it was not, the DMA completion interrupts will not be turned on. In case it was on, the DMA completion interrupts will be turned on This status is indicated in the 15th bit of each DMA channel assignment word in the interrupt table of the operating system Base page. The various internal registers are restored to pre-entry state. In case the Memory Protect flag was on, the interrupt system and Memory Protect system will be turned on or else only the interrupt system will be turned on. The driver restores the original operating system Dynamic Mapping System (**DMS**) status and returns to the point of interruption before entry to the driver. The System Map will always be enabled when

entering the privileged driver as this section is entered directly from the trapcell entry and the Mapping system must be restored on return to pre-entry suspension point.

On reading the IHR when identifying the interrupting LAM module, a special case with the Branch number equal to zero can occur. This can indicate an interrupt from the Serial Branch Coupler completion. In this case an internal routine will process the interrupt. The function of this routine is to check if it is a Serial Branch Coupler interrupt. If it is not from the coupler, then the driver completes by resetting the CAMAC hardware interrupt mechanism and exiting; the interrupt from CAMAC is ignored.

In case it is an interrupt from the Serial Branch error station [12], a CAMAC function to clear the LAM mask in the Serial Branch coupler is issued. In case of an error, it is reported via the error handling mechanism of the driver. Otherwise the normal Serial Branch completion exit is followed.

In case the interrupt is a valid one from the Serial Branch Coupler, the LAM completion service for the Serial Branch is executed as explained below :

The address of the EQT for the CAMAC logical unit (LU) is stored in the interrupt table at the system Base Page instead of the program IDSEG address as describe before. The only difference is that the EQT address is positive. The system interprets a positive entry in the interrupt table as being an EQT entry address to be queued after all outstanding I/O requests. A 'software interrupt' is made as described above and the same return path is taken as before. When the operating system responds to the interrupt on the dummy card after exiting this driver, a completion interrupt for the CAMAC LU/EQT will be processed (by the Serial Branch

driver Completion section).

4.4.2 The Error handling section

This section of the privileged driver handles the reporting of errors to the Control System. It is specialized because error reporting cannot use any system I/O requests as the interrupt system is off during the driver execution.

The error reporting section is entered at different entry points depending on the seriousness of the various errors that may occur. The specific error parameters are passed to the system list processor '$LIST' and a high level error reporting program ('SNDER' or 'SNDAR') is put into the program scheduled queue. In case '$LIST' reports program 'SNDER' not available for immediate scheduling (will occur after exit from the driver when the interrupt system is turned back on), an attempt to use 'SNDAR' is made. See the Programmer's Reference manual for a description of these programs.

In case a LAM service program is not dormant or in a time-list (it can therefore not be scheduled immediately), the IDSEG address of this program and the CAMAC address of the LAM'ing module is passed to 'SNDER' (or 'SNDAR'). This program will in turn pass the information to a high level dispatcher program 'DUPLC' [13](considered part of the CONTROL SYSTEM [4]) so that the server program can be queued for eventual scheduling when it comes out of the time-list or goes dormant.

If certain hardware access errors occur, the offending module's CAMAC address as well as the CAMAC status is passed to '$LIST'. A certain number of retries are also done for the hardware to ensure that occasional hardware errors are overlooked except for a hard error.

More detail on the types of errors reported by the privileged driver can be found in the Programmer's Reference Manual.

CHAPTER 5

PERFORMANCE EVALUATION

At the time that that the interface method development was started, the version of the operating system in use on the mini-computer was RTE-4B. The current version in use is RTE-6/VM. The latter version of the operating system incorporated changes to allow Virtual Memory access, Shared Extended Memory access and certain significant firmware enhancements. The memory access enhancements did not affect the I/O process in any way but the firmware enhancements did.

5.1 THE PRIVILEGED SUBROUTINE METHOD

As mentioned in chapter 4, the standard way to access peripheral devices connected to the mini-computer is by using a system Input/Output request (EXEC call). Usually a device driver is provided to interface the I/O request to the device hardware. The operating system specifications [5] indicate that the system overhead for a typical transfer will be in the order of 2 milliseconds. As the user requirements indicated that transfers must be executed at the fastest possible rate, the method of privileged subroutines promised a much quicker transfer rate.

Timing measurements made to determine the access time of a CAMAC module from a user program on RTE4B compared to RTE-6/VM indicated :

RTE4B	RTE-6/VM
390 microseconds	190 microseconds

Above figures are very similar for the single and the double word transfers. The enhancement for the block transfer privileged routine was not as significant as only a small portion of the routine execution time is spent in the $LIBR (going privileged) and $LIBX (going non-privileged) system routines compared to the rest of the block transfer routine body. Typically a block transfer for one data word took 190 microseconds. This figure includes the $LIBR/$LIBX calls to go privileged as well as the actual I/O to CAMAC via the minicomputer's I/O registers and the PTI. A full data transfer is done every 25 microseconds (repetition time). As the user can specify a variable number of data words per block transfer, the total time taken to execute this routine will vary. As mentioned in Chapter 4.3.1 an operating system requirement exists that the interrupt system be switched on within 1 millisecond after it was disabled by a privileged routine and this will limit the block size to 20 words of 16 bits (refer chapter 4).

The difference in the access times above is the result of certain enhancements to the firmware used with the RTE-6/VM operating system regarding the privileged subroutine method (in particular the system routines $LIBR and $LIBX).

It was therefore possible to satisfy the user requirements regarding the transfer access speed by using the method of privileged routines instead of the standard method of an I/O request via the operating system. An additional decrease in

access time was benefit from the switch from RTE-4B to RTE-6/VM.

5.2 THE SERIAL BRANCH ACCESS METHOD

In the case of CAMAC modules residing in crates connected to the Serial Branch, the nature of the hardware transaction processing called for a device driver I/O request procedure that will be handled by using an I/O transfer start and interrupt completion mechanism. The transfer start could be done by using the privileged routines 'SICAM' or 'DICAM'. The transfer completion would then have to be a polled operation using the same routines to find the hardware completion status. As this would imply that a user program sit in a tight loop going privileged/non-privileged with the interrupt system off most of the time, it is not the most efficient way to use a multi-user multi-tasking real time operating system. No other processes would be allowed to progress unless they have a higher priority.

The CAMAC modules on the Serial Branch consists only of power supply control/monitor modules. These are Analog-to-Digital and Digital-to-Analog converters and a control module for switching the power supplies on/off or to change the polarity from positive/negative. As the response time of these modules are relatively slow compared to computer I/O, the access rate need not be so high. It is quite sufficient to use the device driver approach to effect an I/O transfer with the resultant minimum transfer time of 2 milliseconds per transfer.

5.3 INTERRUPT SERVICING

The method as described in Chapter 4 using the privileged driver approach was used as it promised to be much faster in processing an interrupt than the standard method using a system I/O request. The main reason for the speedier service is that all of the system overhead associated with processing of an interrupt is eliminated by bypassing the Central Interrupt Controller (CIC).

During the development of the privileged driver, the standard method to schedule a high level interrupt service program (external to the driver) was to use the system list processor '$LIST' [14]. When implemented in this way, it was found that a service program (although being dormant and available to be scheduled for execution) would only start executing at the second 10 millisecond clock tick after the CAMAC privileged driver has completed. This is as a result of the dispatcher mechanism used by the operating system. As all interrupt service programs have priorities set to a value of less than 50, it signifies that a linear scheduling method is used by the operating system [15]. Programs of this priority are given processor control until the program is either completed, terminated or suspended to await the availability of a required resource. The dispatcher will only enter the program information in the schedule queue at the next clock tick after the driver has completed. The program will start at the following clock tick or when another program completes, or is suspended. It meant that a service program could take between 10 and 20 milliseconds to start service after an interrupt. This was not acceptable as the interrupt service was now clock driven instead of event driven.

The method used in the final implementation was to force a software interrupt on a higher priority interface from

within the CAMAC driver. The result is that on driver completion the pending high priority interrupt is serviced by the operating system and the service program is scheduled immediately. This has changed the service mechanism from being clock driven to being event driven.

Measurements done on the RTE-6/VM operating system indicated that the service program started repeatedly within 4,5 milliseconds after the interrupt occurred. This overhead is due to the context switching of the operating system [14]. As the internal operation of the RTE-6/VM operating system is not generally available in source form or for modification, it was not possible to improve the performance from the interrupt servicing mechanism for CAMAC.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

The development of this interface method to allow the Control System software to access CAMAC modules connected to various Accelerator subsystems was required as the available computer and CAMAC hardware system did not incorporate any software interface method.

As the Accelerator subsystems and the hardware to be controlled became available over a lengthy period of time, the requirement for the availability of the individual components of the driver software were also spread out. The first requirement was for access to modules on the Parallel Branches. The next requirement was to provide a method to service interrupts originating from the CAMAC modules. This tied in with the availability of the Serial Branch Accelerator hardware (used to control and monitor the power supplies). This software connection to the Serial Branch was next provided. During the implementation of the different access methods, the methods were optimized for throughput. All of the different software methods are incorporated in the operating system. The Control System performs satisfactorily and is in continuous use at present.

The need to have two computers individually accessing the CAMAC system caused certain complications as far as the interrupt servicing is concerned. Each computer had to identify if a specific LAM occurring on CAMAC was for it,

and if so, service it; otherwise reject the interrupt. This method was incorporated in the driver section servicing the interrupts and is performing satisfactorily. The method to service Serial Branch transfers consists of an I/O request to the operating system. The computer setting up the Initiation request for a transfer to the Serial Branch also had to service the Completion interrupt. As the completion interrupt from the Serial Branch hardware was serviced by both computers in the privileged driver section, only one computer had to acknowledge the hardware completion interrupt. This in turn would provide the Completion interrupt for the I/O request as was set up during the Initiation section (see Chapter 4). A spurious interrupt (no transfer Initiation) would be generated on the other computer if both computers were allowed to service the Serial branch. It was therefore necessary to limit the access of the Serial Branch modules to a specific computer.

It must be pointed out that as a result of the relatively long time taken by the operating system to perform context switching when an interrupt service program is scheduled (using the special technique described in Chapter 4.4), the situation arises when there is a burst of interrupts from CAMAC causing service programs to be non-dormant when they are required to be dormant. This in turn causes a queue build up of interrupts not yet serviced. This is taken care of by the program 'DUPLC' that acts as a high level dispatcher [13]. If this dispatcher does not empty the queue of non-serviced interrupts in good time, the situation arises that the operating system (and therefore the Control system) becomes very sluggish in operation. This is an unfortunate result of deferring the interrupt service to a program external to the driver. The interrupt service programs external to the driver must therefore also execute as fast as possible (very short programs) in order to prevent an abnormal long queue of interrupts.

It would be possible to improve the throughput of data transfer blocks if the Autonomous Memory Controller (AMC) hardware and supporting software driver could be implemented. This method would make use of the Direct Memory Access (DMA) method to transfer data directly to or from the computers' memory and the CAMAC hardware via the Executive crate. The implementation could not be performed as the Control System and the associated hardware have been in use continuously since the three other methods of CAMAC transfers were implemented.

The method of connection for the CAMAC system forms a typical star topology. This topology has the shortcoming that all CAMAC I/O transfers have to proceed via the computers. This makes it very awkward to pass control or data acquisition process information between different instruments residing on the CAMAC branches. Another problem that exists is to transfer information between programs (processes) residing on the different computers. A commercial product that will allow inter-computer communication is available but this method would have a detrimental effect on the CAMAC driver software. It would slow down the servicing of interrupts as well as any direct CAMAC I/O. The reason being that the inter-communication hardware and firmware takes the highest priority in the I/O processing system of the computers. This would violate the design objective of the CAMAC/computer link requiring the highest priority I/O service.

A method to implement a suitable decentralized control system could be investigated for a future project. The current system as implemented will however have to be utilized in parallel to the networked solution until such time that all of the current control functions are replaced by the necessary network servers.

6.2 CONCLUSION

To conclude the success of this research it is necessary to take a look at the objectives.

A given set of hardware without any interfacing software was available at the start. The need to provide an efficient method to interface the CAMAC system with the computers in use was required. This requirement had to be as efficient possible to prevent any bottlenecks in the process to access Accelerator instrumentation. Special techniques to service interrupts from the CAMAC system as well as special methods to effect the transfer of data to or from CAMAC was implemented. All of the goals are successfully reached as the Particle Accelerator Control System is currently using the described software interfaces on the CAMAC hardware from the minicomputers.

REFERENCES/BIBLIOGRAPHY

1. CAMAC Instrumentation and Interface Standards : IEEE Standard 583-1975, Modular Instrumentation and Digital Interface System p.7.

2. Clout, Peter, "A CAMAC PRIMER", Los Alamos National Laboratory in-house document LA-82-2718, Table 1.1a p.3.

3. Weehuizen, H.F., Circuit diagram for modified type HP12894A Multiplexer I/O interface card, NAC Project number ECS28, March 1985.

4. Cloete, I.,"A CONTROL CONSOLE SYSTEM FOR A CYCLOTRON FACILITY",M.Sc. thesis, University of Natal (1983).

5. Hewlett Packard Technical publication for HP1000 Systems: RTE-6/VM Performance Brief, 7/1982, Section 2 : Logic Analyzer Performance test, Table "I/O Profiles", p.4.

6. De Villiers, J.A.M., "Conversion to ASCII for Output and format free Input", Private communication, 1982.

7. Penkler, D., System Engineer, Hewlett Packard South Africa, Private communication.

8. Hewlett Packard Manual Part no. 92200-93005, Dec 1983 : "RTE Operating System Driver Writing Manual".

9. GEC-Elliot Process Automation Limited : "System Crate Catalog", Section 2,"Executive Controller type MX-CTR-3".

10. GEC-Elliot Process Automation Limited : "System Crate Catalog", Section 2, "Universal Interrupt Vector Generator IVG 2404".

11. Alexander, J., Daresbury Laboratory, England : Daresbury Serial Coupler EC 505, "Programming spec and operational Description", April 1978, Section 3.1, p.2.

12. Alexander, J., Daresbury Laboratory, England : Daresbury Serial Coupler EC 505, "Programming spec and operational Description", April 1978, Section 5, p.3.

13. Cloete, I., Program listing of 'DUPLC', Program forms part of programs for [4], 1984.

14. Hewlett Packard Manual Part no. 92068-90013, Jan 1980 : "RTE-IVB Technical Specifications Reference Manual", Chapter 4, Scheduler, p4-1.

15. Hewlett Packard Manual Part no. 92068-90013, Jan 1980 : "RTE-IVB Technical Specifications Reference Manual", Chapter 1, RTE-IVB Dispatcher, p1-1.

16. Hewlett Packard Manual Part no. 92068-90013, Jan 1980 : "RTE-IVB Technical Specifications Reference Manual", Chapter 2, "Interrupt processing", p2-12.

17. Hewlett Packard Manual Part no. 92068-90013, Jan 1980 : "RTE-IVB Technical Specifications Reference Manual", Chapter 2, "RTIOC overview", p2-38 to p2-41.

18. GEC-Elliot Process Automation Limited : "System Crate Catalog", Section 3 - Computer interfaces,Functional specification, Specification No. CSE 2020, "Programmed Transfer Interface for HP2100 series of computers type PTI-21".

19. GEC-Elliot Process Automation Limited : "System Crate Catalog", Section 3 - Computer interfaces,Functional specification, Specification No. CSE 2017, "Autonomous memory channel type PTI-21".

20. GEC-Elliot Process Automation Limited : "System Crate Catalog", Section 1 - Introduction, "GEC-Elliot System Crate Philosophy".

21. GEC-Elliot Process Automation Limited : "System Crate Catalog", Section 2 - Standard System Crate Units, "Branch Coupler type BR-CPR-3".

22. Hewlett Packard Manual Part no. 92084-90004, Jan 1983 : "RTE-6/VM Terminal User's Reference Manual", Appendix C, pC-1.

NOMENCLATURE

CAMAC	-	Computer Automated Measurement and Control
ESONE	-	European Standards On Nuclear Electronics
PTI	-	Program(med) Transfer Interface
SC	-	Select Code
CSR	-	Control and Status Register
CCR	-	CAMAC Command Register
CFR	-	CAMAC Function Register
DHR	-	Data High Register
DLR	-	Data Low Register
IHR	-	Interrupt Handling Register
CDC	-	Composite Data Channel
DMA	-	Direct Memory Access
IVG	-	Interrupt Vector Generator
AMC	-	Autonomous Memory Channel
ACU	-	Autonomous Control Units
DBE	-	Differential Branch Extender
CC	-	Crate Controller
LAM	-	Look At Me (interrupt from CAMAC module)
SPU	-	Set Point Unit
DRT	-	Device Reference Table
EQT	-	Equipment Table
INT	-	Interrupt Table
I/O	-	Input/Output
RTE	-	Real Time Executive (operating system kernel)
IOC	-	Input/Output Control
CIC	-	Central Interrupt Controller
DVR46	-	CAMAC device driver
IDSEG	-	36-word block defining a program's properties
SCD	-	System Crate Demand
GL	-	Graded LAM
SBC	-	Serial Branch Coupler
BCNA	-	Branch,Crate,Station,subfunction (CAMAC module address)
LSB	-	Least Significant Byte
MSB	-	Most Significant Byte
SIHR	-	Serial Interrupt Handling Register

EXEC - Operating System function call

SDA - System Driver Area

JSB - Jump to Subroutine

DMS - Dynamic Mapping System

LU - Logical Unit

APPENDIX A

The Operating system Equipment Table (EQT)

Word	Contents															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O Request list pointer														
2	R	Driver "Initiation" section address														
3	A	Driver "Continuation/Completion" section address														
4	D	B	P	S	T	Subch 5 bits					I/O select code					
5	Av	Equip type code							Status							
6	Current I/O Request word															
7	Request buffer address															
8	Request buffer length															
9	Temporary storage or optional parameter															
10	Temporary storage or optional parameter															
11	Temporary storage for driver															
12	Temporary storage for driver or EQT extension size															
13	Temp storage for driver or EQT ext start address															
14	Device time-out reset value															
15	Device time-out clock															

THE EQUIPMENT TABLE (EQT)

THE EQT TABLE FOR DVR46 :

EQT1 - SUSPENDED LIST LINKAGE. LINKED TO THE ID SEGMENT OF THE CALLING PROGRAM. IF REIO IS USED IN THE CALL THE LINKAGE IS TO AN ID SEGMENT CREATED BY EXEC.

EQT2 - INITIATION SECTION ENTRY POINT

EQT3 - CONTINUATION SECTION ENTRY POINT.

EQT4 - FORMAT D BPS TUU UUU CCC CCC

- D = DMA (NOT USED IN THIS DRIVER)
- B = BUFFERING ON YES = 1 OR NO =Ø (NOT USED)
- P = POWER FAIL NO = Ø
- S = TIME OUT SERVICED BY DVR YES=1 (NOT USED IN THIS DRIVER)
- T = TIME OUT OCCURENCE
- U = UNIT OR SUBCHANNEL
- C = I/O CHANNEL(SLOT)

EQT5 - FORMAT A ATT TTT TSS SSS SSS

- A = AVAILABILITY
- T = DEVICE TYPE 46
- S = STATUS BYTE (for Serial branch) WHERE bit # :
  - 1 = Transaction completed ok
  - 2 = Transaction completed with error
  - 3 = Transaction terminated by timer
  - 4 = ERR bit from reply message
  - 5 = SX bit from reply message
  - 6 = SQ bit from reply message.
  - 7 = DERR bit from reply message
  - 8 = Bit sync lost during transaction
  - 9 = Command(s) rejected due to bit 2 being set

EQT6 - CONWRD

EQT7 - REQUEST BUFFER ADDRESS

EQT8 - REQUEST BUFFER LENGTH

EQT9 - IVG or Serial Branch Coupler station # or BCNA for CNAF If Serial Branch coupler, bit 15 will be set

EQT10- Camac function code (Ø -> 31)

EQT11- Not used

EQT12- Not used..

EQT13- STORAGE FOR ID SEGMENT ADDRESS FOR INTERRUPT PROCESSING ROUTINES. THIS LOCATIONS IS ZERO IF NO INTERRUPT PROGRAM IS AVAILABLE AND A NEGATIVE IDSEGMENT ADDRESS IF THE INTERRUPT PROGRAM HAS BEEN DISARMED.

EQT14- TIME OUT VALUE.

EQT15- TIME OUT COUNTER.

APPENDIX B

Description of CAMAC Function codes

Also refer to: CAMAC Primer by Peter Clout LA-UR-82-2718
Page 37 Table 2.9 [CAMAC primer]

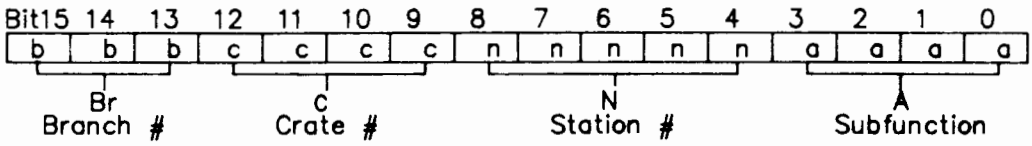
There are 31 CAMAC function codes and are usually addressed by a 5 bit code. Function codes Ø to 7 uses the Read lines in the CAMAC crate system and can be termed as being READ functions.
Function codes 16 to 23 uses the Write lines in the CAMAC crate system and can be termed as being WRITE functions.

Function codes 8 to 15 and 24 to 31 does not use the Read and Write lines of the CAMAC crate system and can be termed as data-less functions.

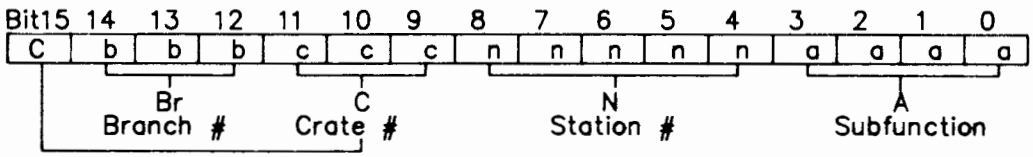
APPENDIX C

A brief description of the CAMAC module address

The compound CAMAC module address (Branch (B), Crate (C), station number (N) and subfunction code (A) sometimes called the BCNA address; see figure) is loaded from the EQT word 9. This format as passed by the user programs differ from the actual hardware requirements and is converted. The reason for this conversion stemmed from the requirement to provide a total of 15 crates for the Serial Branch system. The CAMAC Command register in the hardware used 15 of a possible 16 bits for this address. The 16th bit was used for an indicator to use the data in the CCR to address the actual module at address BCNA or the CAMAC Address Register (CAR). As all our addressing modes was only using the BCNA address mode(bit 15 of CCR = Ø), 1 bit was available for extending the maximum crate address from 7 to 15 with suitably modified hardware in the PTI.



User program BCNA format for CAMAC module Address



PTI hardware BCNA format for CAMAC module Address

APPENDIX C

APPENDIX D

Description of the internal CAMAC I/O routines of the driver

These routines are very similar to parts of the PRIVILEGED SUBROUTINES. The major difference is that when the driver code is executed, the interrupt structure of RTE is already off. There are 3 different routines namely :

RCAM, WCAM and DLCAM for READ, WRITE and DATA-LESS transfers.

RCAM:

The following computer registers are used:

Calling sequence: A-register contains BCNA
B-register contains the CAMAC Function code

Error Return: A-reg : contents of the CSR (PTI Status)
B-reg : destroyed

Normal Return: X-reg : Top 8 bits of 24-bit CAMAC word
Y-reg : Bottom 16 bits of 24-bit CAMAC word
A-reg : Contents of CSR (PTI Status)
B-reg : destroyed.

Output the BCNA word to the Command register CCR
Output the Function code to the Function register CFR and start CAMAC cycle in PTI.

```
READ: IF Data Flag is set THEN
        IF Error Flag is set THEN
          GoTo ErrorReturn
        ELSE
          Get data from Data registers ( DLR and DHR ) and
          copy it to the X-and Y-registers. Get the PTI
```

status from the status register CSR and return to calling section.
IF Error Flag is set THEN
  GoTo ErrorReturn
ELSE
  IF watchdog timed-out THEN
    GoTo ErrorReturn
  ELSE
    GoTo READ.
ErrorReturn: Set Function Complete Flag, Clear Error Flag
Get status from Status register and return error

WCAM:
The following computer registers are used:

Calling sequence: A-register contains BCNA
B-register contains the CAMAC Function code
X-register contains Top 8 bits(24-bit write)
Ø (16-bit write)
Y-register contains Bottom 16 bits of data
Error Return: A-reg : contents of the CSR (PTI Status)
B-reg : destroyed
Normal Return: A-reg : Contents of CSR (PTI Status)
B-reg : destroyed.

Output the BCNA word to the Command register CCR
Output the Function code to the Function register CFR and start CAMAC cycle in PTI.
Copy the X-register to the A-register and output to DHR.
Copy the Y-register to the B-register and output to DLR.
WRITE: IF Function Complete flag is set THEN
IF Error Flag is set THEN

```
            GoTo ErrorReturn
          ELSE
            Get status from Status register and return normal
        ELSE
          IF Error Flag is set THEN
            GoTo ErrorReturn
          ELSE
            IF watchdog timed-out THEN
              GoTo ErrorReturn
            ELSE
              GoTo WRITE.
ErrorReturn:   Set Function Complete Flag, Clear Error Flag
               Get status from Status register and return
               error
```

DLCAM:

The following computer registers are used:

```
Calling sequence:   A-register contains BCNA
                    B-register contains the CAMAC Function
                    code
Error Return:  A-reg : contents of the CSR ( PTI Status )
               B-reg : destroyed
Normal Return: A-reg : Contents of CSR ( PTI Status )
               B-reg : not destroyed.
```

Output the BCNA word to the Command register CCR
Output the Function code to the Function register CFR and start CAMAC cycle in PTI.

```
DATALESS: IF Function Complete flag is set THEN
            IF Error Flag is set THEN
              GoTo ErrorReturn
            ELSE
               Get status from Status register and return
               normal
          ELSE
            IF Error Flag is set THEN
```

```
            GoTo ErrorReturn
          ELSE
            IF watchdog timed-out THEN
              GoTo ErrorReturn
            ELSE
              GoTo DATALESS.
ErrorReturn:  Set Function Complete Flag, Clear Error Flag
              Get status from Status register and return
              error
```

APPENDIX E

Sample privileged routine Assembler code

```
     NAM  sample *name of routine
     ENT  sample *entry point
     EXT  $LIBR,$LIBX *external routines called
sample    nop          *No-operation
*
* Go privileged ( switch interrupts off )
*
     JSB  $LIBR  *jump to subroutine $LIBR
     nop
     .
     .
     .
     LIA  selectCode *(normally illegal (I/O instr))
     .
     .
     .
*
* Go non-privileged ( switch interrupts on )
* and return to calling program
*
     JSB  $LIBX
     DEF  sample  *return to caller.
     END
```

PROGRAMMERS REFERENCE MANUAL

Table of contents

INTRODUCTION

This reference guide is written for experienced programmers; it provides implementation-specific details about the CAMAC driver and routines. In addition, it provides definitions for each of the CAMAC procedures, listed in alphabetical order. These are the chapters and appendixes in the programmer's reference guide:

Chapter 1: Using the CAMAC library routines summarizes the CAMAC input/output (I/O) support.

Chapter 2: The CAMAC library is an alphabetical reference of all the CAMAC library routines. Each definition gives syntax, related routines (if any) an operative description, return values and possible portability information for the routine.

Chapter 3: Error messages lists and explains each of the error messages and summarizes the possible or probable

causes of the problem that generated that message.

Appendix A: CAMAC utilities describes various other utilities for driver initialization and presets, status display and interactive CAMAC access from a user terminal for hardware debugging/manipulation.

Appendix B: CAMAC Status definition describes the meaning of the various bits of the status register of the PTI.

CHAPTER 1

USING THE CAMAC LIBRARY ROUTINES

The library routines are contained in the library file CAMLIB.LIB on directory \LIBRARIES. These libraries will automatically be searched during linking of user programs.

In order to access instrumentation modules contained in the CAMAC crates, the library routines provide an interface to the CAMAC Executive crate. The Executive crate hardware modules will initiate the required hardware actions to address and control the instrumentation modules that the user wishes to access.

The basic routines (SICAM, DICAM, NICAM) accepts the CAMAC module address, the CAMAC function (Read, Write or Dataless (CAMAC control functions) transfer) and the data-word(s) to be transferred as variables via procedure parameter passing. Internal to the routines these parameters are then (after disabling of the mini-computer interrupt structure to effect I/O) passed to the CONTROL module of the Executive Crate in order to start the transaction. The standard method for computer I/O namely Skip-On-Flag is used to ascertain that the actual command is transferred to the hardware. In case of any hardware errors occurring, hardware as well as complementary software flags are set. Only if no errors occurred will the data-word(s) be written or read from the DATA module of the Executive crate. Proceeding to the completion section of these privileged routines, in case errors occurred certain bits indicating the error type will be set in the status word returned to the user program (See Chapter 3 Error messages). The interrupt system will at this point be switched on again so as to keep the interrupt service dead time to a minimum. In case the transfer was

successful, the status word is cleared and returned. Another status word (Q-response) is an indicator to the user whether the CAMAC module addressed in the program has accepted or rejected the CAMAC command. This is very helpful in the case of certain modules that need to be polled to accept the CAMAC command. The user is however not encouraged to use this method for CAMAC access as it burdens the mini-computer interrupt system for every access via these privileged routines causing the rest of the system to slow down dramatically. The method of operation of the privileged routine is to switch the interrupt system off for I/O and back on after I/O is done. Repeating this very often causes the operating system to do only this (interrupts off/on). Other real-time processes that need attention via the interrupt system will be delayed for service. The 3 different routines differ only as far as the number of data-words is concerned. The routine 'SICAM' (**S**ingle Integer **CAM**ac) will read or write one 16-bit word to or from CAMAC. The routine 'DICAM' (**D**ouble Integer **CAM**ac) will read or write 2 16-bit words to or from CAMAC. The third routine 'NICAM' will accept a full branch, crate and station number address as separate parameters and internally compound it to the full BCNA address for the Executive system.

A special routine to allow the block-transfer of data (16-bit words only) namely 'BLCAM' (**BL**ock **CAM**ac) exists. The operation of this routine is similar to the previous routines except that when all address information was set up, the routine does transfer of data-words for the required number of words specified by the user before exiting the routine. This will cause the interrupt structure to be switched off for longer times than single/double word transfers. In order not to violate the operating system prescribed suggested value of interrupts off-time namely 1 millisecond, it is necessary to limit the maximum number of words to transfer. By timing measurements a suggested safe

number of words to transfer per call was found to equal 20.

Library routines by category

Conversion Routine

This routine converts a CAMAC station address to a compound address contained in a 16-bit word as required by the hardware in the Executive crate.

declr (camlib.lib)

Input/Output routines

These routines provide the specific functions as described above for data transfer to CAMAC.

sicam	(camlib.lib)	**blcam**	(camlib.lib)
dicam	(camlib.lib)	**declr**	(camlib.lib)
nicam	(camlib.lib)		

Interface routines

These routines provide machine-specific information and program sources reside in directory /USERS/JNT/DELIVER/SOURCES .

camlu	(camlu.ftn)	**nschd**	(nschd.mac)
getgl	(getgl.mac)	**iupit**	(iupit.ftn)
drflex	(drflex.ftn)		

CHAPTER 2

THE CAMAC LIBRARY

This sample library look-up entry explains how to use this section of the manual.

Using library routine entries

Name **routine** - summary of what the library routine does.

Usage decleration syntax

Related
Function usage Routine2 (parameters)
Description This describes what **routine** does, the parameters it takes, and any details you need to use **routine** and related routines listed.

Return value The value that **routine** returns (if any), is described here.

See also List of other routines concerned with same subject

camlu

Name **camlu** - Get LU of CAMAC device.

Usage

```
camlu( VAR CamacLU,
           EQTAddress    : integer16bit);
with
  TYPE integer16bit = -32768..32767;
```

Description **camlu** Return the LU number and EQT address of the first LU associated with DVR46 (CAMAC driver).

Return value The LU of the CAMAC device is returned in 'CamacLU'. The address of the Equipment Table (EQT) in the Base Page of the operating system for this device is returned in 'EQTAddress'. In case no DVR46 reference is found, a value of zero is returned in 'CamacLU'.

See also

blcam

Name **blcam** - Block transfer of CAMAC data.

Usage

```
blcam(CamacFunction,
      CompoundAddress: integer16bit;
      VAR DataBuffer : ARRAY[1..20] OF
      integer16bit;
      Howmany        : integer16bit;
      VAR QResponse,Status: integer16bit);
      with
        TYPE integer16bit = -32768..32767;
```

Description **blcam** transfers a block of 16-bit integer data-words (address 'DataBuffer') from/to CAMAC according to the CAMAC function in 'CamacFunction' at CAMAC station at compound CAMAC address in 'CompoundAddress'. The number of words to transfer is passed in 'Howmany'.

Return value The requested number of data-words are placed in the buffer at address as specified in 'DataBuffer' in case the CAMAC function is a Read function. The response from the addressed CAMAC module is returned in 'QResponse'. The status of the CAMAC hardware after the transfer is returned in 'Status' (see appendix for description of Q-response and status word).

See also **sicam, dicam, nicam** and **declr**

declr

Name **declr** - Build compound CAMAC module address

Usage

```
declr( VAR CompoundAddress : integer16bit;
           BranchNumber,
           CrateNumber,
           StationNumber,
           SubAddress     : integer16bit);
with
  TYPE integer16bit = -32768..32767;
```

Description **declr** builds a compound CAMAC module address given the branch number, crate number, station number and subadress (for internal subfunction for each module).

Returnvalue The compound address is returned to caller. See also the Appendix for description of the compound address.

See also **sicam**, **dicam**, **nicam** and **blcam**

dicam

Name **dicam** - 24-bit transfer of CAMAC data.

Usage

```
dicam(      CamacFunction,
            CompoundAddress : integer16bit;
            VAR Dataword : integer32bit;
            VAR QResponse,
                Status      : integer16bit);
with
  TYPE
    integer16bit = -32768..32767;
    integer32bit = -2147483648..+2147483647;
```

Description **dicam** transfers transfers one 32-bit integer data-word (address 'Dataword') from/to CAMAC according to the CAMAC function in 'CamacFunction' at CAMAC station at compound CAMAC address in 'CompoundAddress'. The data-word to be transferred will occupy the least-significant 24 bits of a 32-bit word describing the data-word.

Return value The data-word is placed in the buffer at address as specified in 'DataBuffer' in case the CAMAC function is a Read function. The response from the addressed CAMAC module is returned in 'QResponse'. The status of the CAMAC hardware after the transfer is returned in 'Status' (see appendix for description of Q-response and status word).

See also **sicam, nicam, blcam** and **declr**

drflex

Name **drflex** - Send control request to DVR46.

Usage

```
drflex( DriverCommand,
        OptionalParameter[,
        BranchNumber,
        CrateNumber,
        StationNumber]  : integer16bit);
with
  TYPE integer16bit = -32768..32767;
  [...] is optional for 'DriverCommand'= 3..6
```

Description **drflex** is used to send control requests to the CAMAC driver DVR46. The control requests (in 'DriverCommand') can be :

Code = Ø
This code is used by the operating system and is a system requirement. It will force the driver to clear the interrupt flip-flops on the Interrupt Handling Register (IHR) as well as on the Serial Branch Interrupt Handling Register (SIHR).

Code = 1
This code will force the driver to execute the code to re-initialize the CAMAC hardware and if the first time that the driver was entered, also the internal tables of the driver.The parameter 'OptionalParameter' contains the module station number in the Executive Crate of the Interrupt Vector Generator(IVG).

Code = 2

This code will reset the first-time flag in the driver, clear the internal table containing the ID segment address (IDSEG) of the service programs, and restore the patched trapcells in the user or system map trapcell area of the operating system to the values before the CAMAC driver was initialized. These values are stored in the driver temporary storage area at initialization.

Code = 3

This portion of the driver control request section takes the program name at address 'OptionalParameter' and searches through the ID segments attached to the Keyword list for the ID segment address for the associated program and when found, places this IDSEG address into word 13 of the EQT. The 'BranchNumber', 'CrateNumber' and 'StationNumber' parameters of this procedure call contains the CAMAC Branch number, crate number and module station address of the module that can generate a LAM to be serviced by the CAMAC driver. This address is used to build a pointer to an internal table (INTAB) of the driver. The service program IDSEG address as found above, will be stored at this table address.

The driver internal table (INTAB) will contain all the service programs' IDSEG addresses in order that the privileged section of the driver can schedule these programs on the occurrence of a LAM.

Code = 4

This code is used to pass the CAMAC module address from 'BranchNumber', 'CrateNumber' and 'StationNumber', and form a pointer to the associated entry in INTAB internal to DVR46. The IDSEG address at that entry will then be complemented to indicate a disarmed service program.

Code = 5

This code will function in exactly the same way as code 4 but the effect will be to re-arm the service program as indicated in the table.

Code = 6

This control request code is used to remove the IDSEG address of an interrupt service program from the table INTAB. The same mechanism to calculate a pointer to the table is used as for codes 4 and 5. The table entry is then cleared to zero. This value will cause the privileged driver to reject any LAM's occurring from the specified CAMAC module.

Return value No values returned to caller.

See also **exip,nschd**

getgl

Name **getgl** - Get Graded LAM value from IHR

Usage

```
getgl( VAR GradedLAM  : integer16bit);

with
  TYPE integer16bit = -32768..32767;
```

Description **getgl** returns the Graded LAM as found in the Interrupt Handling Register (IHR). As the user program cannot directly access the IHR (low level I/O will cause program to abort with Memory Protect error), it was necessary to provide the user with this procedure.

Return value The returned value is contained in 'GradedLAM'. The LSB byte contains the contents of the IHR. Bits Ø to 4 contains the Graded LAM value and bits 5 to 7 contains the Branch number of the CAMAC LAM Grader that signalled the LAM.

See also **vinig, CAMAC STATUS DEFINITION**

iupit

Name **iupit** - Issue "UP" device command

Usage

```
iupit(LogicalUnit:integer16bit):integer16bit;

i := iupit( LogicalUnit );

with
  TYPE integer16bit = -32768..32767;
```

Description **iupit** will calculate the EQT for the given LU (Logical Unit) in the system and then issue a call to the message processor to "UP" the EQT in case it was down (not available, not ready).

Return value If the call was successful in the execution of its function, a value of zero is returned. Otherwise a non-zero value is returned.
word.

See also

nicam

Name **nicam** - 24-bit transfer of CAMAC data.

Usage

```
nicam( CamacFunction,
       Crate,
       Station,
       SubFunction  : integer16bit;
       VAR Dataword : integer32bit;
       VAR QResponse: integer16bit;
       Branch       : integer16bit;
       VAR Status   : integer16bit);
with
  TYPE
    integer16bit = -32768..32767;
    integer32bit = -2147483648..+2147483647;
```

Description **nicam** transfers one 32-bit integer data-word (address 'Dataword') from/to CAMAC according to the CAMAC function in 'CamacFunction' at CAMAC station at CAMAC address in 'Branch','Crate', 'Station' and 'Subfunction'. The data-word to be transferred will occupy the least-significant 24 bits of a 32-bit word describing the data-word.

Return value The data-word is placed in the buffer at address as specified in 'DataBuffer' in case the CAMAC function is a Read function. The response from the addressed CAMAC module is returned in 'QResponse'. The status of the

CAMAC hardware after the transfer is returned in 'Status' (see appendix for description of Q-response and status word).

See also **sicam, dicam, blcam** and **declr**

---

nschd

Name **nschd** - Set-up interrupt service program name in DVR46 (CAMAC driver)

Usage

```
nschd(CamacLu,ControlWord,
      ProgramNameAddress : integer16bit)
with
  TYPE integer16bit = -32768..32767;
```

Description **nschd** is an Assembler routine used internal in 'DRFLEX'. It is not intended to be used by programmers and is included here for completeness.

Return value None

See also **drflex**

sicam

Name **sicam** - 16-bit transfer of CAMAC data.

Usage

```
sicam(     CamacFunction,
           CompoundAddress : integer16bit;
           VAR Dataword : integer16bit;
           VAR QResponse,
               Status     : integer16bit);
with
  TYPE integer16bit = -32768..32767;
```

Description **sicam** transfers one 16-bit integer data-word (address 'Dataword') from/to CAMAC according to the CAMAC function in 'CamacFunction' at CAMAC station at compound CAMAC address in 'CompoundAddress'.

Return value The data-word is placed in the buffer at address as specified in 'DataBuffer' in case the CAMAC function is a Read function. The response from the addressed CAMAC module is returned in 'QResponse'. The status of the CAMAC hardware after the transfer is returned in 'Status' (see appendix for description of Q-response and status word).

See also **dicam**, **nicam**, **blcam** and **declr**

CHAPTER 3

ERROR MESSAGES

CAMAC DRIVER DVR46 ERROR MESSAGES

Error handling for the Initiation/Completion sections

Errors for these sections are handled by the RTE operating system. The status of the device is normally stored in EQT word 5 status byte (bits Ø-7) for the CAMAC device. The return code in the A-register signals the type of error to RTE. The indication contained in the A-register after returning from a driver is pre-defined by the operating system requirements and can be :

A = Ø	The operation was successfully completed.
A = 1	Device or controller malfunction or not ready.
A = 2	End-of-tape or End-of-information.
A = 3	Transmission parity error.
A = 4	Device time-out.

The messages reported by the operating system will take one of the following message formats :

```
IONR L xxx E yyy S zz qqq
IOET L xxx E yyy S zz qqq
IOPE L xxx E yyy S zz qqq
IOTO L xxx E yyy S zz qqq
```

with xxx = device's logical unit number.
with yyy = device's Equipment Table number
with zz = device's subchannel
with qqq = device status returned by the driver (also in word 5 of the EQT)

If the device is already down at the I/O request, the status will be = ****.

For the CAMAC driver all errors occurring during the Initiation section are flagged as 'Parity' errors or 'End-Of-Tape' errors and Bit 7 of the Status word is set. These messages are used for the lack of being able to modify the operating system internal error reporting messages to indicate a relevant error message for the CAMAC driver. Possible errors during the Initiation section can be :

1. IVG station number or Serial Branch Coupler station number not in range (3 to 23). An End-Of-Tape message is used to indicate error.
2. Any error occcurring trying to execute a CAMAC Read/Write/Dataless function (using the internal driver routines)
3. CAMAC function code out of range will be indicated by an End-Of-Tape message and bit 7 of the status word set.

The error reported during execution of the Completion section of the driver is :

1. Device time-out with bit 7 of the status word set and the actual status of the CAMAC transaction in the rest of the status word.

For the detail about the CAMAC status word see the section titled 'CAMAC STATUS DEFINITION'

Error handling for privileged section (interrupt servicing)

Errors to be passed to program 'SNDER' or 'SNDAR'in parameter list (5 words maintained and passed to scheduled programs by the operating system list processor '$LIST' --

P1, P2, P3, P4, P5).

Parameter P1 : Pointer = 9 for table index in error-reporting program

P2 : Error type :

= Ø : Everything is OK, program will be scheduled
= 1 : Service program not enabled / or RP'ed
= 2 : Service program not dormant
= 3 : CAMAC H/W error
= 4 : CAMAC H/W Error (in IVG Trapstore resetting)
= 5 : Empty entry in LAM table for LAM grading.
= 6 : Program to be scheduled is in time list.
= 7 : Error accessing LAM grader.
= 8 : LAM station out of range (1..22).
= 9 : Bad Access of Serial Coupler.

Error messages for the privileged subroutines

The privileged subroutines 'SICAM', 'DICAM', 'NICAM', and 'BLCAM' will all report errors durring a CAMAC transfer by displaying a message on the user terminal (if user is using a program from session) or on the system console of the computer if non-session. These messages are :

S/W WATCHDOG TIME-OUT - This message reports that the CAMAC hardware did not respond within a specified retry repeat count with the Flag flip-flop set. This usually happens if the CAMAC crate is not mastered by the Executive crate but by an auxiliary crate controller in the reported crate. Check crates containing micros and mailboxes.

NO X RESPONSE - This message give the indication that the particular module as addressed by the CAMAC routine did not accept the CAMAC command.

NO SYS CRT S2, BR/CR OFF LINE - This message indicate that the CAMAC hardware lost the return handshake from accessing a module. The time-out bit in the status word is set. A possible reason is that the crate addressed, is off-line.

PTI FAILED TO SEIZE SYS CRATE (5 US) - This message indicate that a problem exists in the hardware of the System crate. It is possible that another PTI grabbed the system crate and did not release it.

FALSE ERROR FLAG - The error flag in the PTI was set when not expected to be set. This would indicate a hardware malfunction in the PTI.

SERIAL BRANCH ERROR - This message indicate that during a transfer to a module in the Serial Branch, an error occurred. This usually indicates that the Serial branch hardware did not respond because it is powered-down (possibly one of the crates prevented the Serial loop completion).

SERIAL BRANCH BLOCK TRANSFERS NOT ALLOWED - This message will only be flagged by the block transfer routine 'BLCAM'. The method to access the Serial Branch is by using I/O requests via the system EXEC calls. The EXEC call method does not support block transfers.

The CAMAC address of the module as well as the CAMAC function code and subfunction code are displayed on the next line after the error message.

A typical error message : NO X RESPONSE
B=Ø2 C=Ø3 N=17 A=ØØ F=16

means that the module residing at Branch 2, crate 3, station

17 did not accept the CAMAC write function (F = 16).

APPENDIX A

CAMAC UTILITIES

All the sources of the following utility programs are in directory /USERS/JNT/DELIVER/SOURCES/ .

cnaf	(cnaf.ftn)	**exip**	(exip.ftn)
inis	(inis.ftn)	**maklm**	(maklm.ftn)
paklm	(paklm.ftn)	**iprt**	(iprt.pas)
padvr	(padvr.ftn)	**snder**	(snder.ftn)
serfix	(serial_fix.pas)	**vinig**	(vinig.ftn)

---

cnaf

Name **cnaf** - Program to access CAMAC modules via command-line parameters.

Usage RU,CNAF,branch,crate,station,funct,subfunct,data
or for CI (Command Interpreter):
CNAF,branch,crate,station,function,subfunct,data

Description **cnaf** allows users to issue a CAMAC command to a CAMAC module from a procedure file or interactively from a terminal. The module address (branch number, crate number, station number and subfunction) as well as the CAMAC function code and the data-word (maximum 24-bits) are passed via the command-line to the program. Commands of this nature can be read,

write or data-less (CAMAC module control) commands. Any data passed back to the program is displayed on the terminal.

Help mode In order to assist user in using the program, if the user types RU,CNAF without any parameters in the command-line string, a message regarding the usage will be displayed:

```
Program CNAF runstring error
U      s      a      g      e      :
CNAF,branch,crate,station,function,a,data
for example  CNAF,2,7,22,16,Ø,32767
```

---

exip

Name **exip** - Program to preset CAMAC driver.

Usage Command-line mode :

```
RU,EXIP,command,param1,param2,param3,param4
```

Interactive mode :

```
   RU,EXIP
```

Description **exip** allows users to issue control requests to CAMAC driver either from a procedure file or interactively from a terminal. These control requests can also be issued programatically from a user's program (see NSCHED and operating system EXEC calls). A total of 6 control requests can be executed

by this program namely :

1. Initialize driver
2. Disable driver from servicing CAMAC LAM's
3. Enter LAM service program name to be scheduled
4. Arm driver LAM service program to be scheduled.
5. Disarm driver LAM service program from being scheduled.
6. Remove LAM service program name from list.

The format for each command (1..6) is as follows:

Command

1. RU,EXIP,1,CamacLu,IvgStationNumber with 'CamacLu' the system LU associated with the CAMAC driver. 'IvgStationNumber' is the station number of the InterruptVectorGenerator (IVG) in the Executive crate. The current present system values for above is :

   LU = 7 and IVG = 6.

   for example RU,EXIP,1,7,6

2. RU,EXIP,2

   This command will disable the driver and all further LAM's from CAMAC will be disregarded.

3. RU,EXIP,3,ProgName,Brnch,Crate,station

   The program 'ProgName' will be scheduled when an LAM from CAMAC address from 'brnch', 'crate' and 'station' occurs.

'ProgName' adheres to the RTE-6/VM specifications for program names namely 1 to 6 characters starting with a Alpahabetic character (refer to [<namr>] in HP1000 Programmer's Reference Manual).
e.g. RU,EXIP,3,PAKLM,2,3,7

4. RU,EXIP,4,Branch#,Crate#,Station#
The program as entered by the command=3 mode will be armed for LAM servicing. This is the default mode after command=3 but this mode can be used to toggle LAM servicing on/off.
e.g. RU,EXIP,4,2,3,5

5. RU,EXIP,5,Branch#,Crate#,Station#
This command can be used to toggle LAM servicing on/off. This command is provided in order to distinguish between arming and disarming as seperate commands to the driver.

6. RU,EXIP,6,Branch#,Crate#,Station#
This command can be used to remove the LAM service program from the table internal to the CAMAC driver . All LAM's from the specified branch,crate and station will, after execution of this command, be ignored.

See also **iprt**

inis

Name **inis** - Program to initialize Serial branch and couplers.

Usage RU,INIS,CamacLu#,Station#
with 'CamacLu#' the system LU associated with the CAMAC driver and 'Station#' is the station number of the Serial Branch driver in the Executive crate. The current present system values for above is :
LU = 7 and Serial branch driver = 6.
e.g. RU,EXIP,1,7,6 or CN,7,1,6

Description This program will "up" the CAMAC LU in case it was down, if successful then it will attempt to clear any CAMAC related errors on the serial crate controllers. If no errors exists then terminate. If not successful repeat for at least 1Ø times else abort program.

maklm

Name **maklm** - Program to enable the CAMAC interrupt hardware for a module in a crate on a CAMAC branch.

Usage RU,MAKLM,Branch#,Crate#,Station#
with 'Branch#' branch address of CAMAC module, 'Crate#' the crate address and

'Station#' is the station number.

Description This program will clear the LAM request on the specified module. Then it will enable the System Crate demands in the Executive crate, enable the branch demand for the addressed branch, enable the crate controller to respond to LAM's, and initialize the Interrupt Vector Generator (IVG) trapstore. It will then enable the IVG to handle Graded Lam (GL) cycles. This means that the IVG will keep on scanning branches with the Demand flag set while the computer is servicing the IVG via the Programmable Transfer Interface (PTI). Next the interrupt mechanism on the IVG is enabled.

The commands for above functions in more detail:
f2.aØ for given B C and N (with B - Branch C - Crate and N - station).
f26.aØ for BCN
Enable System Crate demands :
f26.alØ for Br = Ø, Cr = Ø, N=3Ø (Executive controller station #)
Enable Branch demand on Branch coupler:
f26.alØ for B=Ø, C=Ø, N=station number of Branch coupler
Enable A2 crate controller :
f26.alØ for B as given, Crate as given, N=3Ø
Init the IVG:
Reset the trapstore : f17.alØ for IVG module
Enable GL cycles : f26.all for IVG module
Enable interrupt : f26.alØ for IVG module

The LAM servicing mechanism for a particular

module as well as the entire mechanism from the crate controller, to the branch coupler and the IVG will be enabled after this program has executed. Any LAM occurring in the module will trigger the service mechanism and eventually interrupt the computer in order to run a LAM handler so that the necessary action can be taken to service the module needing attention.

See also paklm

---

paklm

Name **paklm** - Program to service a LAM generating module in a crate on a CAMAC branch.

Usage RU,PAKLM,BCNAWord
with 'BCNAWord the compound branch, crate and station number address of CAMAC module.

Description This program will clear the LAM request on the specified module and reset the Interrupt Vector Generator (IVG) trapstore only for the particular interrupting module trap bit.
The commands for above functions in more detail:
f2.aØ for given B C and N (with B - Branch C - Crate and N - station).
Init the IVG:
Reset the trapstore : f17.alØ for IVG module
This program functions as a very limited LAM server. It does not read any data or perform

any other functions but clearing the LAM in the originating module. It usefulness lies in the fact that it can be used to test if the LAM service mechanism is fuctional.

See also maklm

---

iprt

Name **iprt** - Program to list the internal table of the driver to a terminal

Usage RU,IPRT

Description This program will list the names of LAM service programs with associated CAMAC addresses and the current status regarding the LAM servicing capabilities to a terminal.

---

padvr

Name **padvr** - Program to patch driver into the System Driver Area .

Usage RU,PADVR,absoluteDriverModuleName
e.g. RU,PADVR,!DV6ØC

Description This program read a file with the absolute version of the CAMAC driver (name in

'absoluteDriverModuleName') and store the code at the address in the System Driver Area as indicated by the Assembler. This program obsoletes the necessity to regenerate the Operating System for every new version of the CAMAC driver. This proved to be a tremendous help during the development of the driver. The phases that a driver will go through would be to edit the source, assemble to absolutized code (fixed addresses in the system) and then use this program to load and store the latest version in the operating system area.

---

snder

Name **snder** - Program to service error processing from the CAMAC driver.

Usage Program is scheduled only from the CAMAC driver. It is passed 5 parameters in 5 16-bit words :

Word 1: Contains the LAM'ing module address BCNA

Word 2: Contains the error code (See Chapter 3) "Error handling for privileged section"

Word 3: Contains IDSEG address of service program or CAMAC hardware error code number or CAMAC hardware status depending on the error type in Word 1

Word 4: Not used

Word 5: Not used

Description This program in the CONTROL SYSTEM version, passes the BCNA and IDSEG information to program 'DUPLC' for building up a queue of programs to schedule (for error codes 2 and 6). In the other cases, specific messages are passed to the message processor of the CONTROL SYSTEM in order to inform the operators at the Control Consoles. The version used on the Data Acquisition computer reports messages on the System console for that computer and re-enables the LAM service mechanism for the CAMAC module at address as received in 'BCNA'.

---

serfix

Name **serfix** - Program to remove the bypass on Serial Branch crates.

Usage RU,SERFIX

Description This program will remove the bypass on crates connected to the Serial Branch. As the bypass status on the Serial branch cause an operating system message indicating that the CAMAC device has timed out, no further transactions to the Serial Branch can be performed. In order to enable the Serial branch for further access, it is necessary to clear this bypass condition on all crates.
The program issues a fixed number of repeated

calls to the routine 'IUPIT' (see chapter 2) until the CAMAC device driver is in the ready state. In case of failure, no action is taken, but the program continues. In case it is ready, a CAMAC command f17.aØ to Branch 1, all crates, station 3Ø (Serial Branch controller) is issued. This is the command to clear the bypass on any crates that may have it on. In case of failure, a message is printed on the user's terminal.

This facility was needed after power failures to automatically set the Serial Branch in a healthy state. Otherwise an operator had to access each Serial Branch crate using 'VINIG' to clear the bypass. He would also have to 'UP' the driver manually after each access with 'VINIG'.

---

vinig

Name **vinig** - Program to allow users to send CAMAC functions to modules.

Usage RU,VINIG

Description This program will allow a user to exercise a CAMAC module in an interactive way from a user terminal. The user can issue all CAMAC commands to a specific module including CAMAC dataless commands. Data read back from a module is displayed after the 'X' command. This is also done for data transferred to a

module.

The commands available to the user are :

A,a - Select CAMAC module subaddress.

B,b - Select CAMAC module branch address.

C,c - Select CAMAC module crate address.

D - Specify the data value to pass to module.

F,f - Select CAMAC function code.

L,l - Read the LAM register in the IVG

N,n - Select CAMAC module station number.

Q,q - Quit the program.

R,r - Repeat the present CAMAC command until interrupted by user. This can be done by typing any key and at the prompt 'S = 1 COMMAND? ' type BR . This will cause the program to abort the repeat action and display its prompt.

S - Send ASCII string to CAMAC module.

X - eXecute the CAMAC command with the previously preset addresses and data value.

Y - Repeat mode without terminal display of CAMAC address and data information.

Z - Display the CAMAC Branch#, Crate#, Station# and subaddress in compound address mode as well as seperated. This mode is useful when a compound address is know and the actual Branch#, Crate#, Station# and subaddress is required.

? - Lists the commands available

On starting the program, the program prompt '>' is displayed. The user then enters the required command to set the CAMAC address of the module to be exercised. This will be

'B','C','N' or 'A' (in any order). The data display mode can also be specified by entering the command 'DØ' followed by:

D for decimal mode

B for binary mode

H for hexadecimal mode.

The 'X' command requires a carriage return (<cr>) to complete.

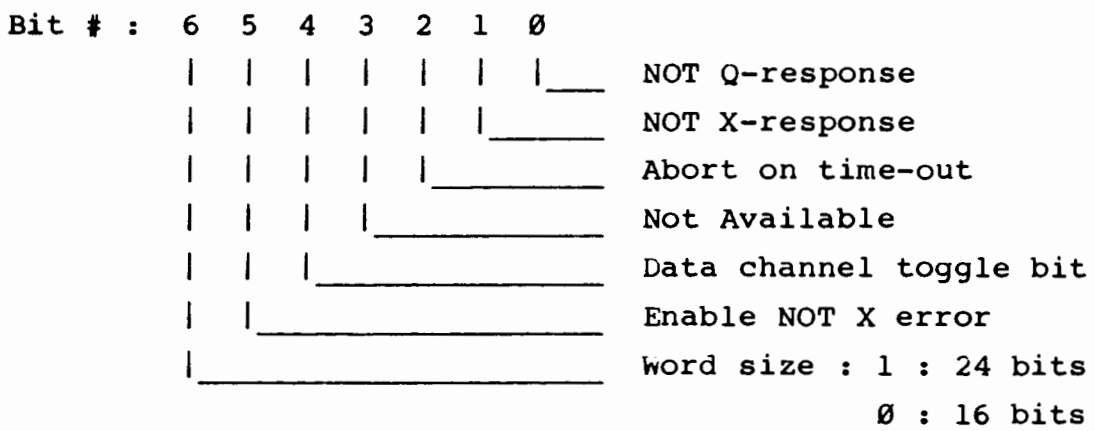
for example :

```
>B2
>C5
>N6
>DØH
>F16   {write to CAMAC}
>X<cr>
CAMAC   B2 CØ5   NØ6 AØØ F16 Q1 X1
DØØØØØØH
>Q
:     { : is RTE6/VM FMGR prompt }
```

APPENDIX B

CAMAC STATUS DEFINITION

The status word described in this appendix is returned from the CAMAC Status Register (CSR) in the PTI. A returned status word is passed to the calling program using the 'PRIVILEGED SUBROUTINES'. The format of this word built internally in these routines follows :

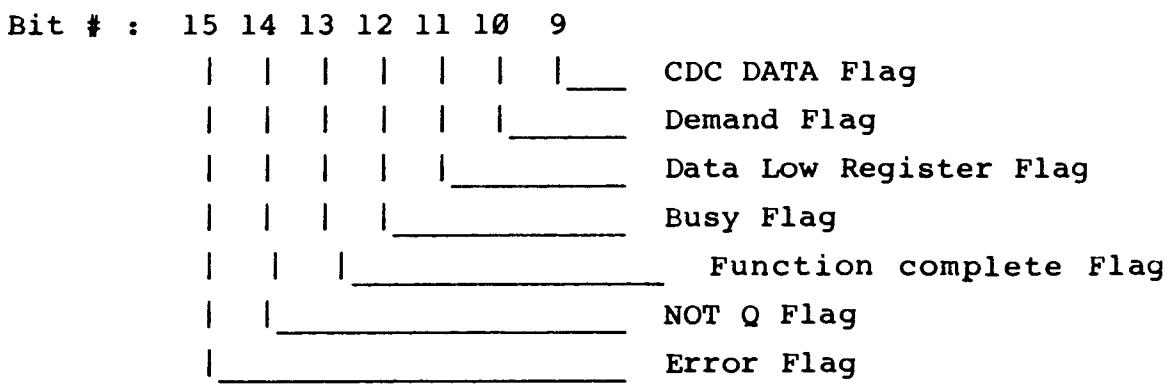


Note :

NOT X can be disabled by software to prevent the hardware setting the Error Flag when a module returns a NOT X status.

Bits 4,5 and 6 are command bits.

The MSB of the status word returned to the calling program, is used to represent the associated flag settings for the various PTI operational registers. These flags are returned to the user in order to get the maximum information about the CAMAC hardware registers in case of any hardware errors.



Also Refer to:

GEC-Elliot Process Automation Limited : "System Crate Catalog", Section 3 - Computer interfaces,Functional specification, Specification No. CSE 2020, "Programmed Transfer Interface for HP2100 series of computers type PTI-21".