A CAMAC-BASED NUCLEAR DATA ACQUISITION SYSTEM

by

DAVID ANDREW LE PATOUREL

B.Sc., University of British Columbia, 1971

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

Physics

We accept this thesis as conforming to the

required standard

THE UNIVERSITY OF BRITISH COLUMBIA

September, 1972

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Physics

The University of British Columbia
Vancouver 8, Canada

Date 25 September, 1972.

Abstract

This paper describes a data acquisition system developed for an intermediate energy nuclear scattering experiment. Equipment standards of CAMAC and NIM were used together with a 12K minicomputer and an industry compatible magnetic tape driver.

The higher level languages of BASIC and FORTRAN were equipped with subroutines that allow input/output communication with the CAMAC data acquisition system. The BASIC-CAMAC system proved to be most useful to the experimenter only taking second place to the FORTRAN-CAMAC system when data handling speed was of paramount importance.

Contents

List of Figures

Acknowledgment

For much of the preliminary work on both software and hardware for the existing system, I wish to thank the Controls Group at TRIUMF, University of British Columbia. I am also indebted to Dr. Quentin Ingram, U.B.C., for his continued interest in further software development; to Professor Garth Jones for the opportunity to build and run the system on his experiment at the Lawrence Berkeley Laboratory, and in particular to my advisor, Dr. Richard Johnson, for continued help and encouragement during the course of my studies.

1. Introduction

This paper describes the data acquisition system (DAS) developed for a 50 MeV π^+d scattering experiment utilizing a NOVA 1200 computer. Section 2 describes the equipment used, and anticipates the problems involved in designing a DAS compatible with the hardware. Section 3 describes the necessary system interfaces, and the software developed for the experiment. The data acquisition program was initially developed using the BASIC interpreter. This commonly known interactive language allowed the experimenters involved to edit and expand the DAS easily. When a final configuration was reached for production running, a change to FORTRAN as the system language was initiated to speed up the data logging process. Generally applicable features of the DAS are described in the main portion of the text; detailed aspects specific to this experiment are discussed in the appendices.

2. Requirements of a Data Acquisition System and the CAMAC Concept

2.1 Scope of Current Experiment

The purpose of the present experiment is to measure $d\sigma/d\Omega$ for elastic scattering of 50 MeV positive pions from deuterium over an angular range of 10 deg. to 170 deg. testing the predications of impulse approximation calculations. This low energy range has not previously been theoretically considered, and good data is scarce. A beam of known intensity strikes a liquid deuterium target and scattered particles are detected in a stopping plastic scintillator. Fig. 1 illustrates the beam line configuration, and Fig. 2 the electronic equipment used. The experiment embodies most of the nuclear detection techniques used today. The data acquisition system is generally applicable to these techniques.

2.2 General Requirements of a Nuclear Data Acquisition System (DAS)

2.2.1 Requirements of the current system are typical of most nuclear scattering experiments:

i) beam characteristics such as flux and energy must be monitored. The beam current is integrated or each incident particle is counted to obtain the estimate of flux. A counter (scaler) is required to record the measurement. A sample of the incident beam particles stopped in a scintillator are analysed with an analogue to digital converter (ADC). The beam energy distribution corresponds to the pulse height distribution obtained from the ADC.

ii) Scattered particles of experimental interest must be identified by a fast logic coincidence network that defines an event. This task is handled by fast nuclear electronics (NIM equipment). The basic speed of the electronics is several nano seconds. Figure 1 presents a logic diagram for this experiment. The event of interest is defined as

$$\text{EVENT} = C_1 C_2 C_3 C_4 C_5$$

Figure 1 Beam Line Configuration

The pion beam is produced by the primary proton beam of the Berkeley 184" cyclotron. It is taken off at a backward angle of 59 deg. and focussed achromatically onto a liquid deuterium target. Knowledge of an incident particle's position on the target and which hodoscope element it passed through is sufficient for momentum determination.

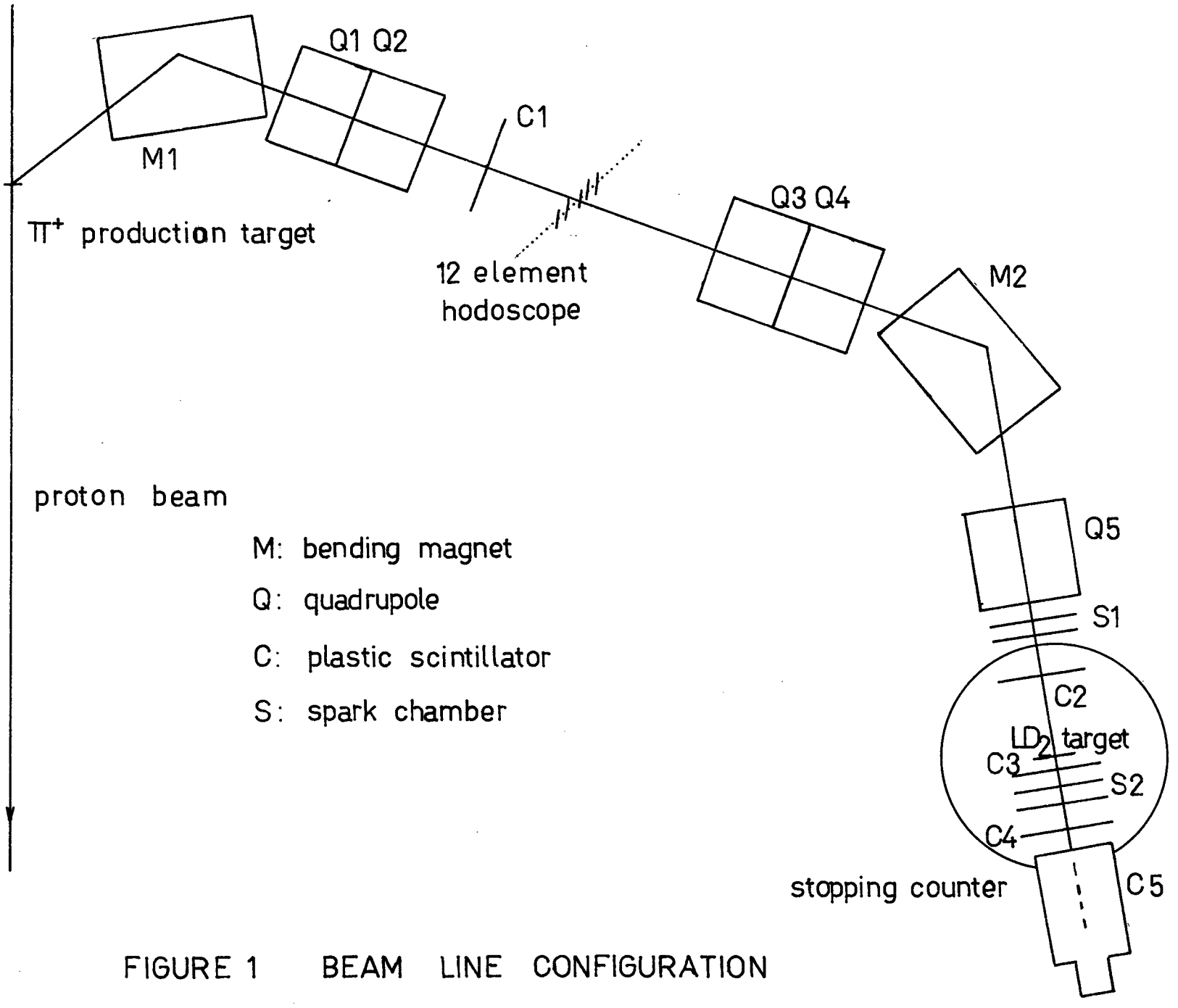

FIGURE 1 BEAM LINE CONFIGURATION

Figure 2 Data Acquisition Hardware

Ehough instrumentation for the system exists to fill several NIM bins; only the one powering the interrupt unit is diagrammed. The BHD is shown in a CAMAC crate. This is not necessary as the unit can function with just a +6v power supply. No back plane connections are needed unless more than seven CAMAC crates are used.

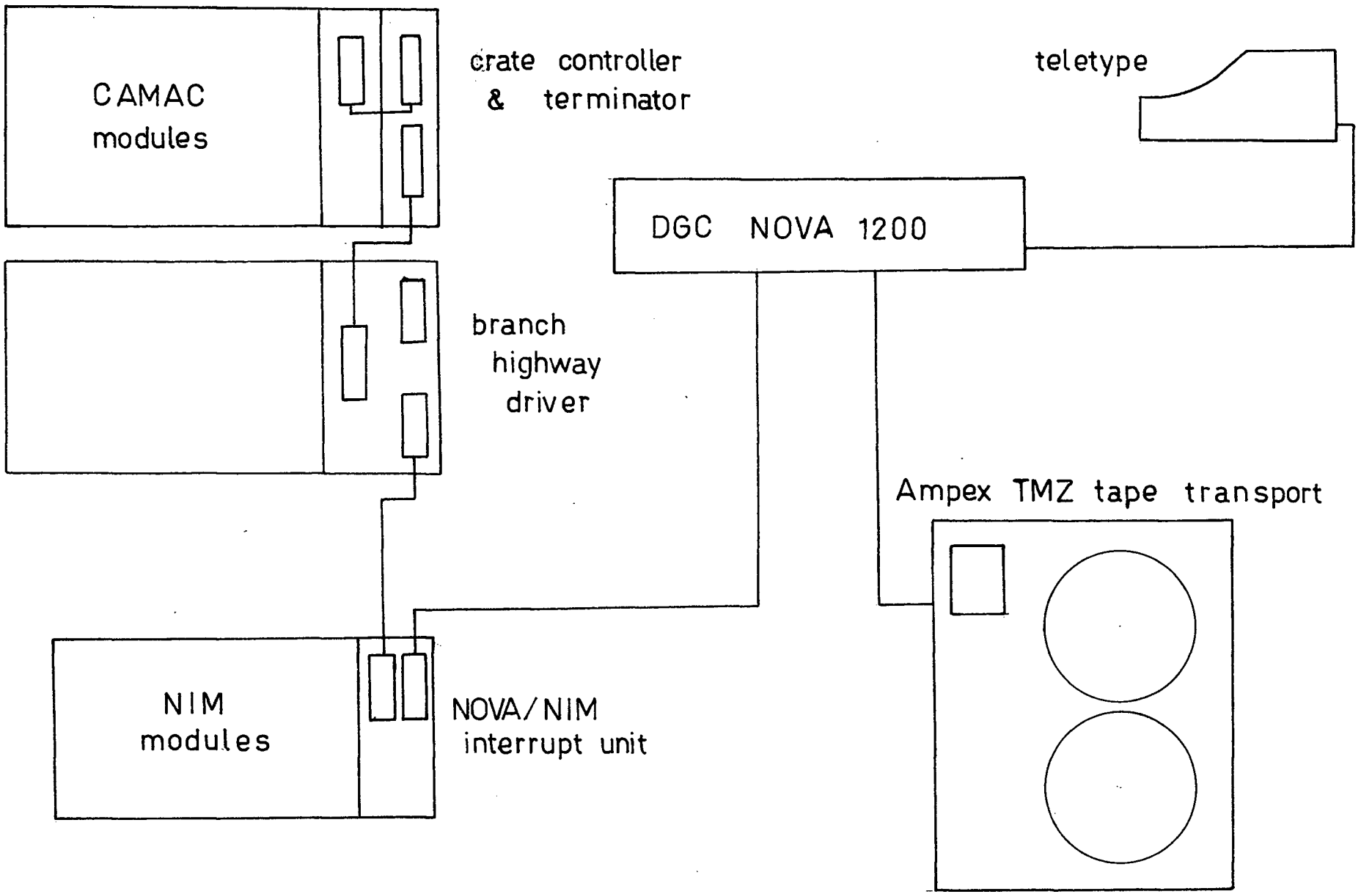

FIGURE 2 DATA ACQUISITION HARDWARE

The NOVA/NIM unit presents an interrupt to the computer when such an event has occurred.

iii) the scattered particles must be tagged with their trajectories, their energy and momentum. Particle trajectories are tagged by spark locations in several chambers in the beam path, as well as an array of finger scintillator counters (hodoscope). The energy is obtained from analogue to digital conversions of stopped particle scintillator pulse heights. Momentum is obtained through computation based on the trajectory or pulse height information.

iv) Data associated with each event must be stored for later retrieval and analysis.

v) Some display capability is required for diagnostic and monitoring purposes. A teletype is standard equipment. A CRT display is convenient and in this experiment is driven from two digital-to-analogue converters that are capable of driving an X-Y plotter.

Use of small computers to aid data handling has resulted in a variety of interfaces designed to allow on-line control of experimental equipment, and it is for this purpose the CAMAC specification has been developed. This hardware standard allows experimenters to build a generally applicable DAS, requiring most specialized changes for specific experiments to be made in the controlling software and not the hardware. Thus, consideration of a particular system illustrates a solution to a much wider problem.

2.2.2 CAMAC

The CAMAC interface, called a BRANCH HIGHWAY DRIVER (BHD), allows the computer to perform all the tasks outlined in 2.2.1. Two-way communication with experimental equipment is achieved through it. Fast NIM equipment signals an event. The fast NIM event pulse is separately interfaced

to the NOVA as an interrupt (Sec. 3.2.1). Sixteen CAMAC scalers in the spark chamber system are read, for each event, along with a pattern recogniser for the hodoscope and an ADC for the stopping counter on the exit arm. Fig. 2 presents a schematic sketch of this equipment. This is all the necessary information for trajectory, energy and momentum analysis, and is stored immediately on magnetic tape. Since multiply-dimensioned arrays are possible with the high level language used in the DAS multiple data binning can be done with incoming data that is not possible with a conventional hardwired multi-channel analyser. Display of these is accomplished with two CAMAC DAC's used to drive X and Y traces on a storage oscilloscope; data from core is simply read out to the units sequentially, so a continuous plot appears on the scope. Any two parameters from the data array can be used as coordinates. On-line calculations are undertaken for low event rates while the system waits between events. For example, spark chamber raw data is converted to an absolute geometric position and individual scattering angles are calculated.

Sophisticated analysis depends only on available time and core.

2.2.3 Hardware Structure of CAMAC

Detailed specifications for CAMAC are found in references 1 and 2. A general description of CAMAC is available[3] and a journal is devoted to its applications[4]. A brief presentation of the CAMAC structural hardware is included here for completeness.

Hardware necessary for a CAMAC system is indicated in Fig. 2; up to seven CAMAC crates can be daisy chained to one BHD. Each crate has 24 stations for modules and the 25th, or control station, for a crate controller type A (CCA) which interfaces modules in the crate to the BHD. Communication between computer and BHD is effected with the standard I/O cable for the machine used.

To address a module, and cause it to perform some function, one must specify to the BHD, the module's crate number C, station number N, sub address A within the module, and some allowed function code F denoting the operation that is required of that section of the module.

3. System Description

3.1 The system was built from the idea of incorporating an easy to use, flexible programming language with the CAMAC specification. In particular, high level languages that are familiar to most scientists were used. Greater utility of the system follows from this standardization since both electronics and programming are specified. The system speed is limited by the CAMAC cycle time and more importantly the program language speed. Operating system details also limit the DAS speed. For example, computer cycle time affects the speed with which the CAMAC system can handle incoming data and hence limits count rates. The two distinct periods during an experiment, debugging and full scale data acquisition, have different program requirements. Initially, ease of program editing is the most important consideration; later, speed of event handling replaces it. Such a change has been met by using two different programming languages, BASIC and FORTRAN.

3.2 System Hardware Interfacing

Two system interfaces are required; one, to translate a fast NIM coincidence signal to a computer interrupt signifying an event, the other the standard CAMAC interface. In principle, the standard CAMAC interface having an interrupt feature is capable of undertaking the event interrupt task.[5]

3.2.1 NOVA/NIM Interrupt Unit

This unit, daisy-chained onto the NOVA I/O cable before the BHD is recognized by the computer as a separate peripheral. The "event definition" pulse (indicating a $C_1 \ldots C_5$ coincidence has occurred) is input to it, and causes this device to request servicing (see Appendix A); the service routine initiates transfer of experimental data from CAMAC to the NOVA. During the transfer, this unit outputs a NIM pulse, which can be used to inhibit functioning of any NIM unit in response to a new event.

This unit was designed as economically as possible; however, use of the Look-At-Me (LAM) CAMAC feature provides an alternate way of signalling an event. Either the external LAM point on a crate controller or LAM point on a coincidence register can be used. This approach is equally acceptable, and has been used in more complicated systems.[6]

3.2.2 Branch Highway Driver

The Master Branch Highway Driver is designed to interface a NOVA computer to a CAMAC system. It can drive up to 7 crates alone. However, it can drive three Slave BHD's which in turn drive seven crates, thus bringing twenty-eight CAMAC crates under the control of the Master BHD. This is a non-standard CAMAC feature and necessitates a Branch Highway code in each command as well as the standard CNA.

3.2.2.1 Loading the BHD

In order to initiate a command to the CAMAC system the three sixteen-bit buffers labelled A,B,C of the BHD must be loaded from core as indicated in Fig. 4. A command requires specification of a three-bit crate number C (1 to 7), a five-bit station number N (usually numbered 1 to 24) corresponding to the addressed module's position in the crate and a four-bit sub-address A to specify a device within a module (e.g. scaler 1 within a quad scaler). A five-bit function code F is also necessary to make the addressed unit perform some task, such as read or clear. A four-bit branch code must be specified if the Master/Slave configuration is used; however, for this purpose, just a master BHD is used so the branch code is always 1110. Fig. 4 also illustrates the position of the twenty-four data bits which can be transferred either way.

3.2.2.2 A CAMAC Cycle

With the buffers loaded in the above manner a start pulse from the computer will initiate a CAMAC cycle. Fig. 3 presents a

Fig. 3 Lines Used During A CAMAC Cycle

i) C,N,A,F lines are activated by BHD and the BTA timing reference pulse is generated.

ii) the crate controller activates a single N Line and the A and F lines.

iii) the addressed module puts its data on the read (R) lines and generates a Q response to indicate its successful completion of the operation.

iv) the strobe pulse S1 puts the data into the controller which then generates the BTB timing reference pulse to indicate proper operation.

v) the BHD terminates BTA and the crate controller generates strobe S2 to clear the R lines.

vi) the BTB pulse is removed, causing the BHD to clear the C,N,A,F lines and the cycle is complete.

Receipt of the Q and BTB pulses is checked under program control in each CAMAC cycle since lack of either indicates a program error or a hardware failure.

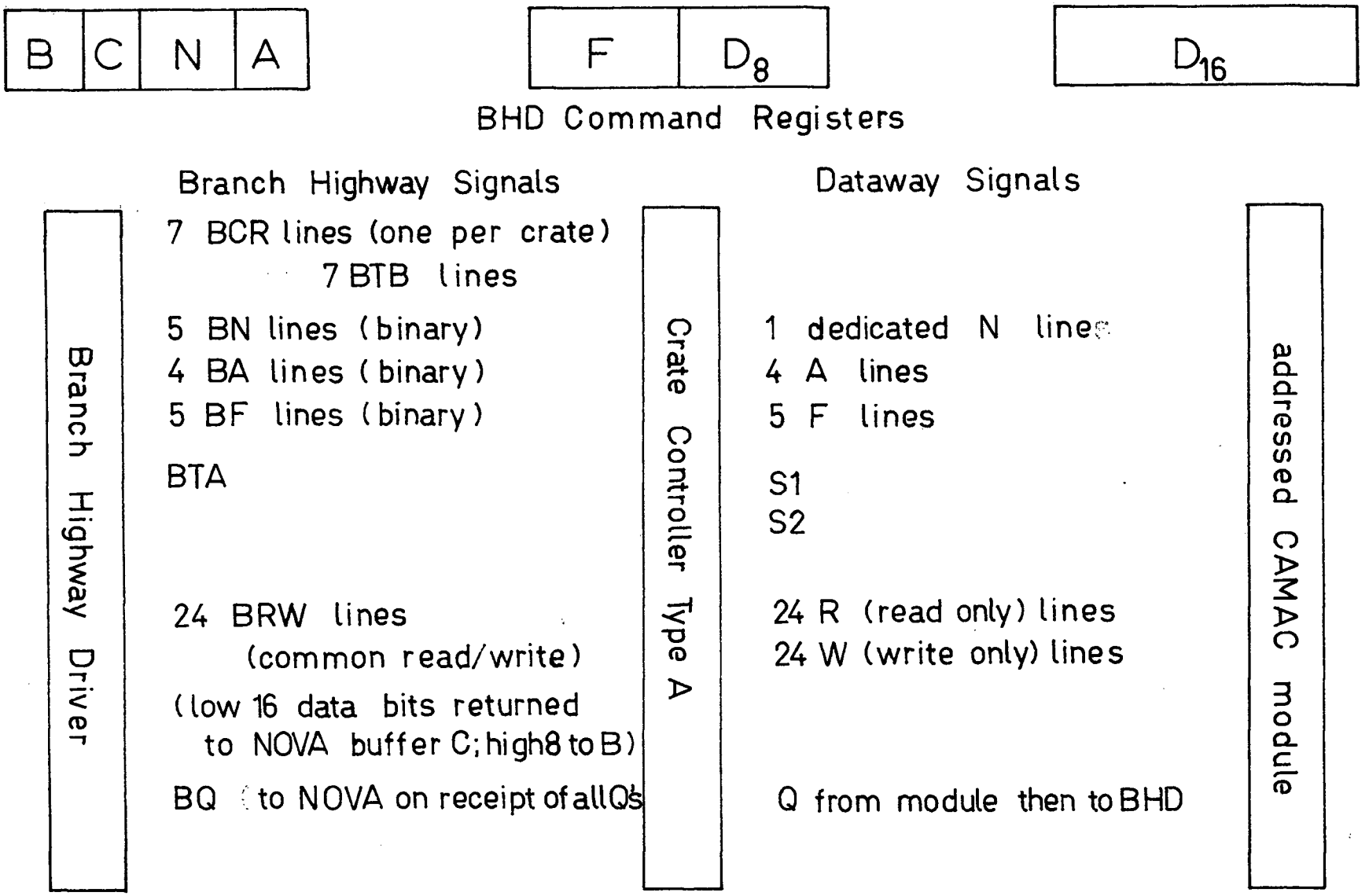

FIGURE 3 LINES USED DURING A CAMAC CYCLE

Figure 4 CAMAC Command Bit Configuration

This illustrates the standard designation of the 48 bits involved in a CAMAC cycle and their positions when loaded into the command registers. When a CAMAC sequence is built in core, three 16-bit words can be used for each entry, with the same configuration as drawn. (See 3.3.2.2).

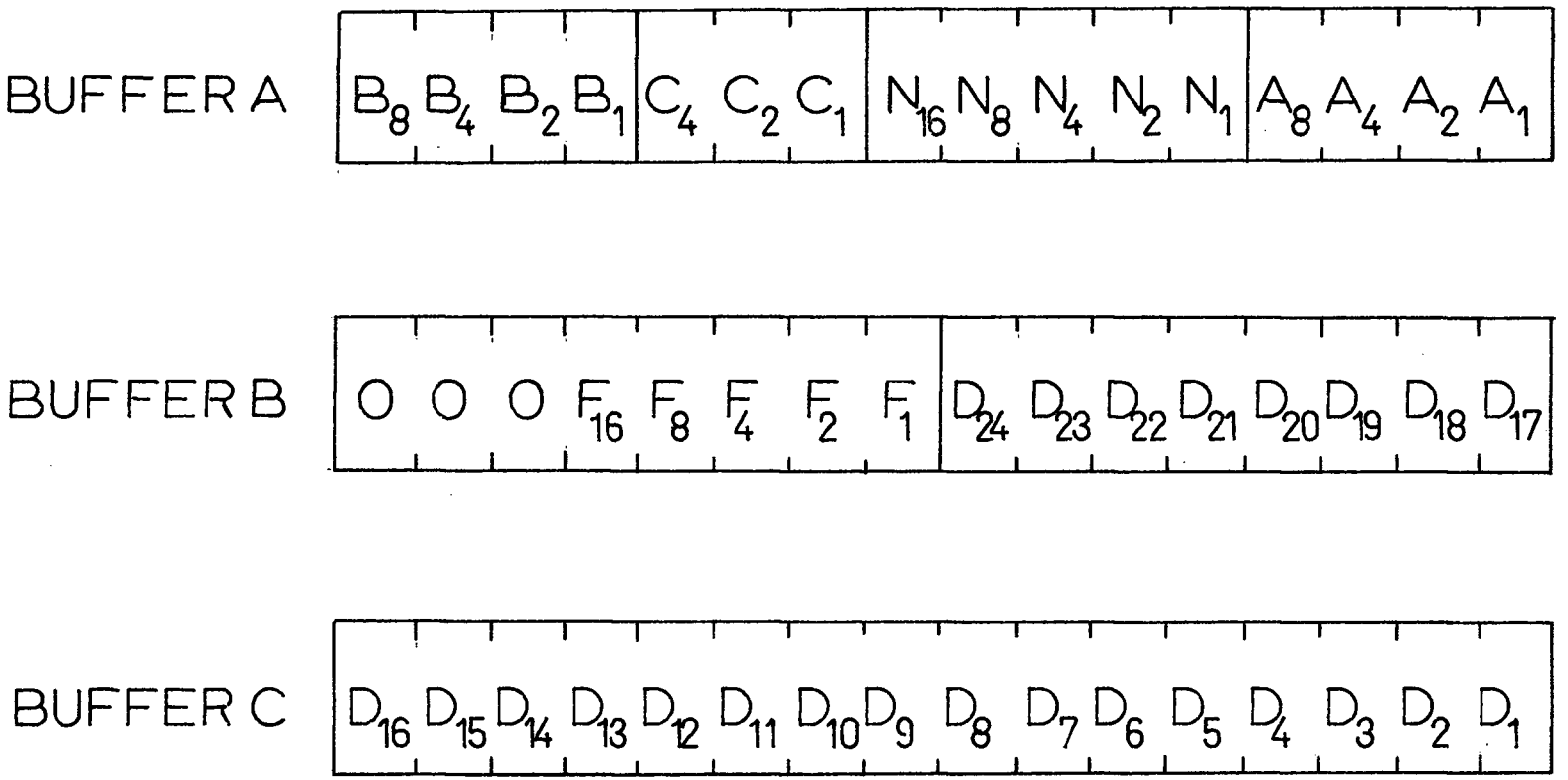

FIGURE 4 CAMAC COMMAND BIT CONFIGURATION

description of the operation. Data is sent to or from CAMAC and some CAMAC function is performed. An error flag is returned indicating the success or failure of the CAMAC cycle.

3.3 Programming

3.3.1 BASIC

BASIC was chosen as the programming language to use during the system installation and commissioning phase of the experiment. BASIC is interactive. Programs can be halted, edited and restarted from a teletype at any time during execution without otherwise affecting them. Since BASIC is an interpretive system storing the teletype code program line-by-line it is slow. Assembler language subroutines doing the CAMAC operations were added to the BASIC system. This was done using the CALL feature of DATAGEN BASIC.* The CALL command is interpreted as a jump-to-subroutine r (JSR) command followed sequentially by the adresses in core of each of the subroutine parameters classified as output (input) to (from) BASIC. The assembler routines are headed by a table in core specifying subroutine number, core entry point, and parameter list.

3.3.1.1 CAMAC Subroutines

CALL 1, (C), (N), (A), (F), D, E executes a CAMAC read operation. The user enters decimal numbers for C N A and F to read or read and clear a module. Data is returned as variable D. A zero is returned in E for a successful CAMAC operation or a CAMAC error code is returned in E if the operation is unsuccessful. CALL 2 (C), (N), (A), (F), (D), E is used analagously to write data into a module. Either operation takes about 10 msec although only 30 usec of that is the actual CAMAC cycle.

Single CAMAC read or write instructions are used to check the

---

*DATAGEN of CANADA, LTD. Hull, Quebec. Copywrite 1970.

operation of modules. However, even during commissioning the execution times of single CAMAC instructions have proven to be excessive. Sequences of CAMAC instructions can be defined by CALL 4711, J, I, C, N, A, F, D where J is one of 5 allowable sequences and I is one of 10 allowed sequence elements. CNAFD bear the usual CAMAC significance. CALL 3, J, Z(J, 1), E allows the execution of the CAMAC sequence J with the resulting data returned to row J of array Z. A zero returned for E indicates a successful sequence execution while other E values denote CAMAC response errors during the sequence execution. The execution speed is much shorter (about 1 msec per CAMAC instruction) than single CAMAC instructions since an assembled program is used when the sequence is being executed. The time difference between this and the CAMAC cycle time is due to data conversion to the BASIC floating point format.

3.3.1.2 CALL 4

BASIC responds to only teletype interrupts. CALL 4 is used to allow program access when event interrupts occur. CALL 4 has no parameters. It takes program comtrol from BASIC and puts the system into a loop awaiting the arrival of an event interrupt from the NOVA/NIM unit. When an event interrupt occurs program control is returned to BASIC and CAMAC sequence reads are then executed to gather data associated with that event.

One additional feature is associated with CALL 4. Data acquisition may be stopped at a known point in the program. The CALL 4 loop checks front panel switches expecting to find only Switch 14 up. If the user puts it down, a crate inhibit command is sent to CAMAC suspending CAMAC operation until the switch is again returned to an up position.

3.3.1.3 CALL 9

A magnetic tape handler was incorporated into a separate

subroutine, Call 9. It is used primarily to write raw data from the Z array mentioned onto tape for later analysis, but is capable of any tape handling operation. Free format for recorded data was used so analysis of the tapes produced at a large computing center usually requires some specialized software to get data from the tapes. Writing onto tape considerably increases time to analyze an event. From start up to completion a tape write operation requires about 50 msec. Consequently data from 12 events are stored in the computer before the data is logged as a single record on magnetic tape. This reduces the net writing time to about 5 msec per event. Appendix C contains a list of magnetic tape options available in Call 9.

3.3.1.4 Physics Subroutines

In order to speed up event analysis, assembler subroutines (CALL 21 and CALL 22) were developed to interpret the hodoscope reading and the spark chamber data for each event. The hodoscope reading is a 12 bit word, one bit for each of the scintillator fingers. CALL 22 is a bit-examining routine to see if just one element, or possibly two adjacent elements fired. Any other result is interpreted as an error. Either multiple passage of particles through the hodoscope, or a failure of the equipment is possible. CALL 21 checks proper operation of all 8 spark planes. Each plane has associated with it two scalers; after a successful firing, a fiducial count should be held in the second scaler. An intermediate count will reside in the first scaler, corresponding to the particles trajectory position in the spark plane. If this condition is not met, an error message results. Fig. 5 illustrates the spark chamber configuration and readout.

The execution speed achieved by assembler coding these two program segments is considerable. CALL 21 requires 130_8 instructions and

Figure 5 Spark Chamber Readout

The pulse trains are from a 20 MHz clock. The fiducial pulse separation for each spark chamber is characteristic of the wand. The spark chamber system is typically 85 per cent efficient in prolonged operation.

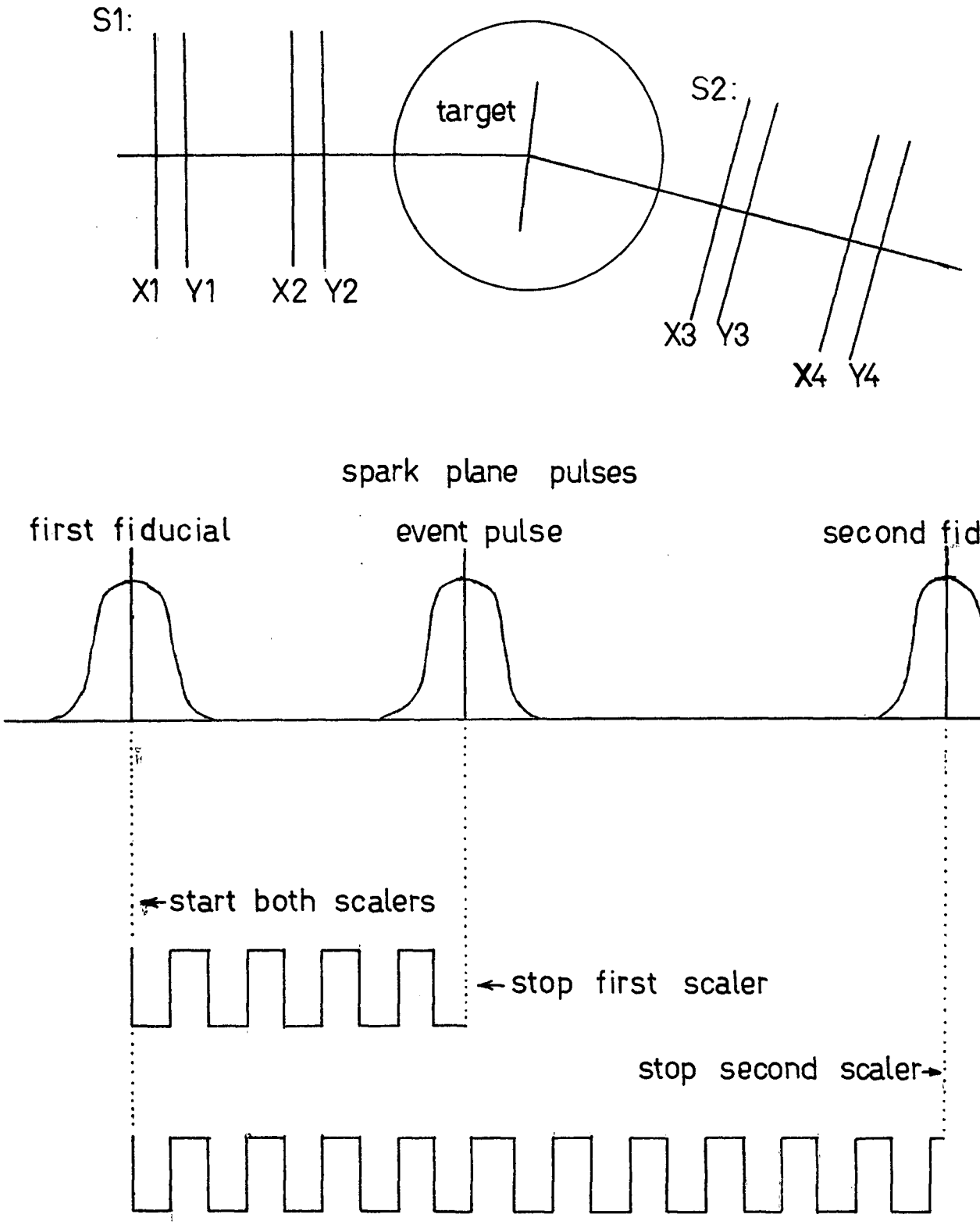

FIGURE 5 SPARK CHAMBER READOUT

is executed in about 100 usecs. CALL 22 requires 53_8 and executes in about 50 usecs. Ten lines of BASIC are required for the BASIC equivalent of CALL 22 for an execution time of 100 msec. About 25 BASIC lines are required for a CALL 21 BASIC equivalent for an execution time of 250 msec. The net speed increase is over three orders of magnitude.

The most rapid data acquisition program used with the BASIC/CAMAC system to date accepts data from CAMAC with sequence reads, performs CALL 21 and CALL 22 operations and records the data on magnetic tape, requiring about 200 msec per event. This is about the speed limit on this form of BASIC/CAMAC data acquisition.

Appendix C contains the BASIC/CAMAC subroutine operating instructions, and Appendix D, a simple BASIC data acquisition program.

3.3.2 FORTRAN

3.3.2.1 Several advantages are gained by a system change to FORTRAN as the DAS language. Primarily, the increase in speed of event handling justifies the change. FORTRAN-CAMAC sequence reads offer a time saving of up to an order of magnitude over BASIC-CAMAC sequence reads. Since single precision (16 bit) integer format can be specified for all variables in the main FORTRAN program, no time is used for conversion between fixed and floating formats. This has the added advantage that magnetic tapes written on with free-formatted binary integers are easy to read, so translation software necessary for data analysis is minimal.

Assembler language subroutines are simple to incorporate, since the DATAGEN FORTRAN compiler accepts machine language instructions as part of a FORTRAN program. Although FORTRAN is not interactive, this feature is hardly missed after a system has reached its final configuration.

The major drawback associated with FORTRAN is the sophisticated hardware necessary for quick compilation and execution. To prepare a

program for execution requires the loading of the compiler itself, assembler to produce a relocatable binary tape from the compiled output, and the library routines to supply the assembled program with the necessary mathematical subroutines (trigonometric, logarithmic, etc.) This represents about 40K words worth of paper tape, which even with a high speed reader, is rather tedious to load.

A disc is the best fast storage and retrieval system to use for FORTRAN programming with a NOVA, but is expensive. Attempts to use the tape unit as a file storage system by means of DATAGEN's "MAGNETIC TAPE SIMPLE MONITOR" (SIMON) system have been disappointing since the tape unit simply has neither the speed nor the reliability of a disc. Two changes to SIMON to make it more compatible with magnetic tape may allow a complete conversion to a FORTRAN system. The first change is a miminization of the necessary writing done by the system, particularly at start-up. The fewer opportunities the system has to destroy itself by overwriting, the fewer system crashes there will be. The second change involves increasing to 4192 from 256 the number of words written per block when the system is storing a file. This reduces by a factor of 16 the number of times directory access is made, thus ensuring less wear on the tape, and less chance of overwriting the directory accidently.

3.3.2.2 FORTRAN Subroutines

Subroutines written for CAMAC and tape unit handling are quite similar to their BASIC counterparts. Thus, CALL CYCLE (C,N,A,F,D(1),$n) does a single CAMAC operation, returning the low order sixteen bits in D(2) and the high order eight bits to D(1). A hardware error causes return to line n in the main program.

Similarly, a sequence operation is effected with CALL EXEC (X(1,I),N,$n). X is a 3 x N integer array. CNA is encoded into X(1,I) as

the I^{th} command in the Sequence. F and the eight high order data bits are in X(2,I), and the sixteen low order bits are returned in X(3,I), just as in Fig. 4's representation of the BHD buffers. Provision for just one long sequence of as many CAMAC operations as necessary is made; N is the number of commands in the sequence.

The routine to build a sequence performs the same function as BASIC's CALL 4711.* The hodoscope and magnetic tape routines are unchanged, but spark chambers are handled just with a FORTRAN subroutine. A brief data acquisition system program in FORTRAN which will do the same analysis as the BASIC program in Appendix D takes no more than 50 msec per event, most of the time being required for the tape unit to write the data record. The immediately obvious improvement to this is one which was incorporated to the BASIC program as well; that is, multiple event logging. Data for 20 events is accumulated in a large array, then written onto tape so that the time required to record each event averages out to less than 2.5 msec.

---

*An example of a FORTRAN sequence build is found in Appendix E.

4. Conclusion

The approach adopted towards DAS development has been reasonably successful. A BASIC-CAMAC system certainly offers experimenters a simple way in which to reach a final configuration without much re-programming necessary. The system fails to be satisfactory, however, when events rates rise much above five per sec. A change to FORTRAN is the logical next step to preserve the flexibility and ease of use offered by a high level language and increase the system speed to a point where the spark chambers will limit the event rate to about twenty per sec. Since the FORTRAN data logging program described in section 3.3.2 occupies only 5 msec (200 per second), the FORTRAN DAS is not computer limited. Additional calculation can be done during the nuclear electronics dead time.

It has been demonstrated that a CAMAC system with a high level controlling language does offer wide applicability once the initial system has been established.

References

1) "CAMAC": A Modular Instrumentation System, Description and Specification", EURATOM Report EUR 4100e, Luxembourg, 1969.

2) "CAMAC: Organisation of Multi-Crate Systems. Specification of the Branch Highway and CAMAC Crate Controller Type A-1". EURATOM Report EUR 4600e, Luxembourg, 1972.

3) IEEE Transactions on Nuclear Science NS-18. 2, (1971).

4) CAMAC Bulletin, a journal of the ESONE Committee.

5) "TRIUMF/CAMAC Branch Highway Driver", W.K.Lacey and D.P.Gurd, TRIUMF Report No. TRI-1-71-2, May 1971 (unpublished).

6) Dollard, Marquardt, Gurd, Johnson, "Proceedings of the 6th Annual Accelerator Conference", July, 1972. (to be published).

7) How to Use the Nova Computers, Data General Corporation, Southboro, Massachusetts, April 1971.

Figure 6 Circuit Logic for the NOVA/NIM Interrupt Unit

The event input, labelled EVENT, is a negative NIM pulse converted to a positive-going TTL pulse. The CPU BUSY output is a TTL level which is converted to a NIM level before being used outside the unit. The unit has device code 40_8.

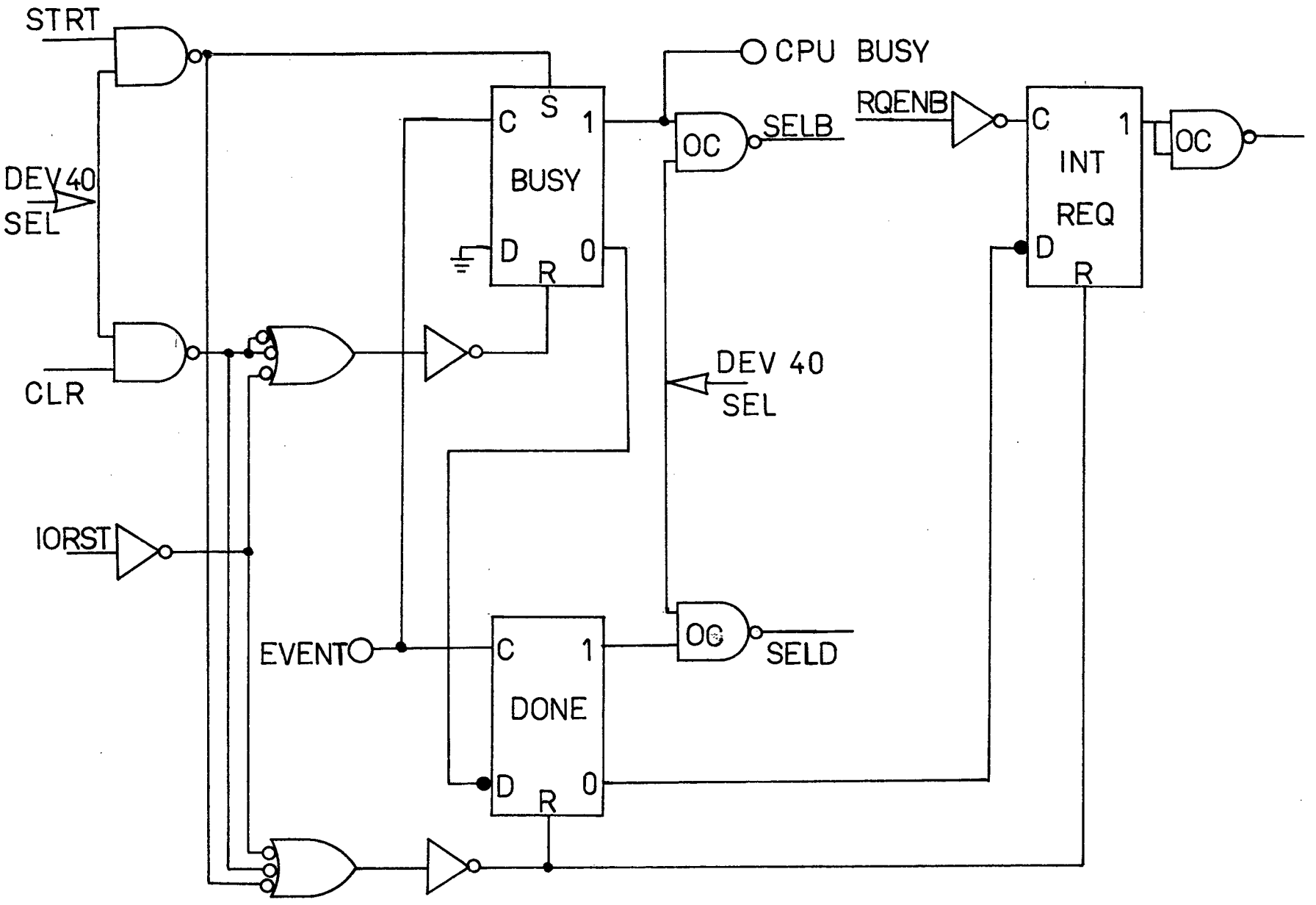

FIGURE 6 CIRCUIT LOGIC FOR THE NOVA/NIM INTERRUPT UNIT

APPENDIX A

Design of the NOVA/NIM Interrupt Unit

Figure 6 illustrates the logic associated with the interrupt unit. It is a simple device, designed to request a CPU interrupt when a NIM event pulse is input to it.

When the data acquisition program enters the CALL 4 subroutine a Start (STRT) pulse sets the BUSY flip-flop (ff) and clears the DONE ff. An event NIM pulse in then clears BUSY and sets DONE. The leading edge of the next Request Enable (RQENB) pulse from the CPU sets the Interrupt Request (INT REQ) ff. The response of the interrupt service routine within the assembler subroutines (recall BASIC ignores all but teletype interrupts) to this request is simply to clear the DONE ff, and transfer program control back to BASIC. The next RQENB senses that DONE has been cleared, and so clears INT REQ. The BUSY ff is still clear and is not reset until the next CALL 4 is executed; the unit is then ready to accept the next event pulse.

For the length of time that BUSY is clear, that is, from receipt of event pulse to next execution of CALL 4, a NIM signal is output from this unit to indicate that the computer is busy analysing an event. The signal is used to veto possible incoming events.

A more detailed analysis of interface operation in general will be found in reference 7.

APPENDIX B

Operation of the CALL Statement

The CALL statement implemented to BASIC has the general format CALL r,a,b,c,....h where r is the number of the assembler subroutine to be executed, and a,b,c,....(maximum of 8) are either input parameters passed from the BASIC program to the assembler routine, or output parameters, passed from the assembler routine to BASIC. The BASIC interpreter occupies the first 4K of core, and at address 10_8 resides the starting address of the assembler subroutines. They are preceded by a three word table for each, specifying subroutine number, subroutine entry point, and a variable control word which classifies parameters as either input (coded 10_2) or output (coded 11_2). (Hence the eight variable restriction - 2 bits for each in a 16 bit word.)

Input parameters are in BASIC's floating point format, and must be encoded as binary integers before being used in the assembler routines. Output parameters are binary integers, which must be encoded as BASIC floating point numbers when passed back to BASIC. These processes are accomplished with the FIX and FLOAT routines within the BASIC Interpreter and probably require of the order of 100 usec each. (Use of FORTRAN's integer format for data handling will negate the need for such wasteful bit shuffling.)

Preceding these tables is the address of the interrupt servicing routine which handles CAMAC, magnetic tape, interface and NOVA/NIM unit interrupts. This device service routine capability exists in addition to BASIC's own teletype servicing routine. An attempt to incorporate both together would require reasonably major editing to the BASIC interpreter, which is uneconomical when system requirements are in a state of flux.

When the statement CALL r,a,b,c,... is executed, the following sequence "appears" to the assembler package:

```
JUMP TO SUBROUTINE (JSR) r
ADDRESS of a
ADDRESS of b
ADDRESS of c
  ...
ADDRESS OF LAST PARAMETER
RETURN TO BASIC HERE
```

When the JSR instruction is executed, its location in core is stored in an accumulator, so the method of access to the parameters and to the return address is obvious.

APPENDIX C

CALL 9 The Magnetic Tape Subroutine

Interfaces for NOVA-compatible magnetic tape units such as the AMPEX Type TMZ transport are three buffer devices. The A buffer can hold one of seven possible tape command codes F: 0 (Read), 1 (Rewind), 3 (Space Forward), 4 (Space Reverse), 5 (Write), 6 (Write End of File), 7 (Erase). Buffer B contains an address A which is the start of a block in core, and buffer C contains a word count, N.

For a Read or Write command, the user must specify both an N and an A as well as F. If the command is Write, the unit will write on tape N 16 bit words from core, beginning at location A. If the command is Read, the unit will read a record off tape, and deposit N words from it onto core beginning at location A.

For Space Forward or Reverse commands, the tape will advance over or rewind over N records, positioning itself at the start of a new record when complete.

Subroutine Call 9, N,F,A,E,X is capable of all six possible tape functions. F is simply the function code, as described. X is a flag (either 0 or 1) to indicate whether the tape unit is going to read or write BASIC variables (32 bits) or an area of core (16 bit words). In the former case, A is interpreted as the beginning address of a BASIC array such as Z (0,0), and N is doubled to take into account the double word length of the parameters. In the latter case, A is interpreted as an absolute address in core. Thus, for example, the statement CALL 9, 50, 1, Z(0,0), E, 1 would write 50 BASIC variables (100 16 bit words) onto tape, beginning with Z(0,0) and ending with Z(4,5).

E is a hardware error flag returned to BASIC from the subroutine. Any non-zero return indicates a fault in a particular record (parity or bad tape) or in the system itself.

Any time data acquisition is halted, the user can inspect the data for any particular event by reversing over sufficient records and reading the desired record back into core. A simple way to organize data on tape is to write one record for each event, tagging it with the run number, and separating runs with end of file (EOF) marks.

CALL 21 The Spark Chamber Subroutine

After a sequence data read of the sixteen scalers associated with the eight spark planes, the counts from each scaler reside in absolute locations in core. The statement CALL 21,W3,F,1,L,U,W4 will test that the count contained in the first scaler (event pulse on Fig.5) on the first spark plane (X1) is non-zero, and less than the fiducial count F for that wand. The count from the second scaler (second fiducial on Fig. 5) must lie within five of F to be accepted as the fiducial count. If it is less, two event pulses presumed to have occurred. If greater, the scaler is assumed to have overrun. Each of these possibilities has a code number; the outcome of each test is returned in variable W3.

A further restriction on the event pulse can be made. The user can test that it lay within lower and upper bounds L and U. Variable W4 is returned as 1 if this test is passed, and 0 if failed. Any number of spark planes can be tested in this way with CALL 21; the essential operation of the spark chamber test in Appendix D's program should be clear.

CALL 22 The Hodoscope Subroutine

Immediately after an event, the hodoscope pattern recognizer has bits set corresponding to the elements of the hodoscope which fired. The statement CALL 22,H,I,E causes the data word stored as BASIC variable H to be examined for set bits. A valid event is characterized by a single bit or two adjacent bits being set. I is a variable returned from the routine indicating which bit or bits were high. (For example, I=20 indicates the tenth hodoscope element fired; I=21 indicates both tenth and eleventh fired). E, the hodoscope quality flag is returned as 0 or 1 for these cases, respectively.

Possible errors are hardware malfunctions, in which case I=0, or multiple passage of particles through the channel, resulting in several bits being set. In that case I is returned as 1. Both errors cause an E return of 2. A practical example of the use of this subroutine is found in Appendix D.

APPENDIX D

A Brief Data Acquisition Program in BASIC

As a specific example of a data acquisition program in BASIC, the following one is included, with a brief description of its operation:

```
1000  CALL 4
1020  LET Z(0,10) = Z(0,10)+1
1040  CALL 3,1,Z(1,1),E1
1041  CALL 3,2,Z(2,1),E2
1042  CALL 3,3,Z(3,1),E0
1043  CALL 3,4,Z(4,1),E3
1060  CALL 9,50,1,Z(0,9),W1,1
1100  CALL 22,Z(2,10),H1,W4
1101  LET H(H1) = H(H1)+1
1102  IF W4< 2 GOTO 1120
1103  LET E(8) = E(8)+1
1104  GOTO 1000
1120  LET W1 = 0
1121  FOR I = 0 TO 7
1123  CALL 21,W3,F(I),I+1,L(I),U(I),W4
1124  IF W4 >0 GOTO 1128
1125  LET S(I,W3) = S(I,W3)+1
1127  LET W1 = 1
1128  NEXT I
1129  LET F8 = F8+W1
1130  IF W1 = 1 GOTO 1000
1140  LET S2 = Z(2,8)/25+25
1141  IF S2 >0 GOTO 1143
1142  LET S2 = 0
1143  IF S2< 100 GOTO 1145
1144  LET S2 = 100
1145  LET Q(0,S2) = Q(0,S2)+1
1150  GOTO 1000
```

Line 1000 is the CALL 4 execute, which sets the system in a condition to accept an event.

Control is returned to BASIC when the event interrupt occurs, and the next four lines (1041 to 1043) are four CAMAC sequence reads of ten commands each. These read in all necessary data associated with a single event, and store it in 5 x 10 array Z. Line 1060 is the write on tape instruction, writing out the useful segment of the Z array.

Lines 1100 to 1103 do the hodoscope check with CALL 22; errors are binned in E(8) and analysis is aborted; or, if the hodoscope functioned properly, proceeds to the Spark Chamber test, CALL 21. Lines 1121 to 1128 check that all eight positional sparks were non-zero, fell within their respective fiducial limits, F(I), and within any assigned lower or upper limits L(I) and U(I). Any error causes a halt to further analysis and a return to line 1000. Otherwise, the ADC reading from the stopping counter on the scattering arm is binned (lines 1140 to 1145). Line 1150 is the normal return to line 1000, to await the next event.

APPENDIX E

An example of a ten element CAMAC sequence build using DGC FORTRAN is included to demonstrate the use of some of the FORTRAN subroutines.

```
      INTEGER D(2)
      INTEGER X(3,10)
      DO 20 I=1,10
      ACCEPT C,N,A,F,D,(1),D(2)
20    CALL SEQNC (X(1,I),C,N,A,F,D(1))
```

Each time the program encountered the ACCEPT statement, it would pause while the user entered C,N,A,F and data necessary for any write commands. Data can be up to 24 bits with the low order 16 bits in D(2) and the high order 8 bits in D(1).

The core image of this sequence is the array X(3,10), each column of which has the configuration of Fig. 4.

1st element	2nd element	...	10th element
(X(1,1))=CNA	.		(X(1,10))=CNA
(X(2,1))=F plus D_8	.		(X(2,10))=F plus D_8
(X(3,1))=D_{16}	.		(X(3,10))=D_{16}

The following segment would await the arrival of an event pulse (i.e. an interrupt from device 40) and then execute the entire sequence.

```
40    CALL IWAIT (DEV)
      IF (DEV.EQ.40) GOTO 50
45    TYPE "ERROR"
      PAUSE
50    CALL EXEC (X(1,1),10,$45)
      GOTO 40
```

After the sequence execution, data from the Ith entry would reside in X(2,I) and X(3,I) (high and low order respectively). BTB or Q response error would cause transfer to line 45.