ON-LINE EXPERIENCE WITH THE 168/E*

J. T. Carroll, J. E. Brau, T. Maruyama and D. B. Parker
Stanford Linear Accelerator Center, Stanford, California 94305

J. S. Chima, D. R. Price and P. Rankin
Imperial College, London, England

R. W. Hatley
Rutherford Laboratory, Chilton, Didcot, England

ABSTRACT

Current 20 GeV/c photoproduction experiments at the SLAC Hybrid Facility require a decision to take a picture within 150-300 μs after the beam pulse. A charged track trigger is provided by a 168/E processor which finds tracks in a downstream PWC system. To meet trigger time requirements the 168/E SNOOP module CAMAC interface is augmented by a CAMAC Auxiliary Controller and dedicated I/O cards in the 168/E chassis. Between beam pulses a floating point Fortran program executing on a 168/E monitors data acquisition. Experience with software development and application are reviewed.

1. INTRODUCTION

The current SLAC Hybrid Facility (SHF) experiment studies photoproduction at 20 GeV/c using a backscattered laser beam.[1)] Electronic detectors downstream of the 40" bubble chamber include eleven proportional wire chambers, two multi-cell atmospheric Čerenkov counters and a large neutral detector using lead glass blocks. To make this experiment practical as regards the quantity of film involved, data from these external detectors is used on-line to determine if a picture should be taken and enhance the ratio of pictures with hadronic production. Previous SHF experiments used assembly language algorithms executing on the host computer for picture selection. For this experiment a system of 168/E processors[2)] was developed to provide more efficient and flexible support for on-line picture algorithms.

The main components of the SHF 168/E system are shown in Fig. 1. The host computer is a Data General NOVA 4/X running under RDOS with 256K bytes of memory and 20M bytes of disk space. The NC023C controllers in each CAMAC crate are daisy chained together on the NOVA I/O bus. The 168/E processors perform as slaves to the host computer which controls them via a SNOOP module CAMAC interface.[3)] A 50 wire flat cable connects an interface card in each 168/E crate to SNOOP modules in the CAMAC crates. Processors #0 and #1 can be accessed by SNOOP modules in either crate 25_8 or 26_8 while 168/E CPU #2 is only connected to CAMAC crate 27_8. Using CAMAC programmed I/O to a SNOOP module, a program on the host computer can download a 168/E program, set the program counter, start/stop the processor, read results in 168/E memory, etc. SNOOP module CAMAC functions are described in Ref. 3.

The SNOOP also provides a direct transfer from CAMAC to 168/E data memory by eavesdropping on input from other CAMAC modules in the same crate. For example, a transfer from BADC[4)] to 168/E #0 requires the following sequence of operations:

i) set interface address register(s) on target 168/E(s);

ii) set interface device select register for target processor(s);

* Work supported in part by the Department of Energy, contract DE-AC03-76SF00515 and by the U. K. Science Research Council.

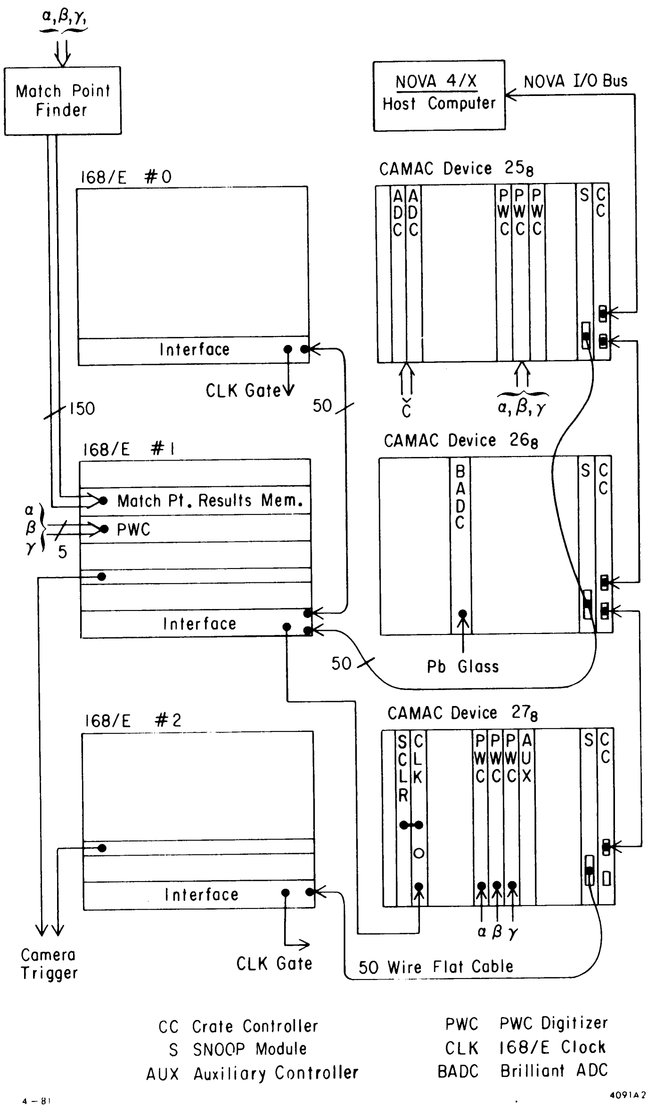


Fig. 1. Organization of the 168/E and CAMAC system at the SLAC Hybrid Facility.

iii) enable Listen Mode on SNOOP in crate 26_8;

iv) read BADC in single instruction or DMA mode;

v) disable Listen Mode, clear device select register and proceed with data transfer complete.

"Listen Mode" eliminates the overhead of first transferring data to the NOVA and then writing it back to 168/E memory. For SHF real-time applications the 168/E programs are stable (no overlays) and fast data transfer is only required for trigger algorithms. With CAMAC the hardware components of the interface are independent of the type of host computer and as evident in Fig. 1, there is considerable flexibility in the arrangement of processors and CAMAC crates.

Several key data acquisition modules in the 168/E - CAMAC system are shown in Fig. 1 and the current hardware configuration on each processor is listed in Table I. We assume the reader is familiar with the basic concepts and capabilities of the 168/E processor.

Following descriptions will focus on SHF on-line applications and supplemental hardware developed to meet the severe time constraint of the camera trigger.

TABLE I. 168/E Configuration

CPU #	Function/Contents
0	Full Algorithm and Fortran Monitor Program Interface to CAMAC Crates 25_8 and 26_8 Integer and Floating Point CPU Three Memory Boards (48K Bytes Data Memory)
1	Hardware and Algorithm Development Interface to CAMAC crates 25_8 and 26_8 Integer CPU One Memory Board (16K Bytes Data Memory) Eight Channel PWC Digitizer Card Match Point Results Memory Camera Trigger Card
2	Fast PWC Algorithm Interface to CAMAC Crate 27_8 Integer CPU One Memory Board (16K Bytes Data Memory) Camera Trigger Card

2. ON-LINE SOFTWARE SUPPORT

Programs for execution on the 168/E are written in IBM Fortran and/or assembly language and normally debugged and tested off-line on an IBM 370/168. Compiler and assembler object modules are processed by a 168/E Translator which produces microcode for the processor. Card image program and data memory files produced by the Translator are transferred to the NOVA 4/X and saved on disk using a 9600 bpi WYLBUR link. A Fortran callable subroutine (DPDOWN), written in NOVA assembly language, reads these disk files and loads 168/E program and data memory. After loading memory, DPDOWN reads 168/E memory, compares each halfword with the corresponding datum from disk and sets an error flag if any discrepancy is detected — any difference is a fatal error. This readback and compare is a vital step in loading 168/E memory. Errors can be produced by WYLBUR — NOVA transfer errors (each line includes a checksum), corrupt NOVA disk files, SNOOP I/O errors and 168/E memory failures. However, as discussed below, most errors result from legitimate contention for the CAMAC bus. Large blocks of constants, e.g., algorithm look-up tables, are maintained on NOVA disk files independent of Translator output and loaded into COMMON blocks in 168/E data memory using a separate utility (DPXFER).

The key program in development of this system has been the 168/E Editor (DPEDIT). With this interactive program executing on the NOVA 4/X, users can examine and control 168/E processors from the main NOVA console. DPEDIT can display and modify 168/E memory with a word size of half, full or double and a base of decimal, octal or hex. The ability to examine processor memory in fullword format is particularly convenient since the NOVA has a 16 bit word size and the SNOOP transfers data in halfwords. DPEDIT can also write/read the 168/E program counter, start/stop processor execution, etc. With the exception of "Listen Mode", DPEDIT exercises all SNOOP interface functions required for data acquisition. Interface problems have usually been isolated using DPEDIT together with an oscilloscope on the interface flat cable or 168/E backplane. However, some problems have required placing 168/E interface, CPU and memory cards on an open test bus rather than the closed 168/E crate to allow easy access for test probes.

Listen Mode data transfers are tested using a memory module in the CAMAC crate. The SNOOP module Listen Mode in Crate 26_8 (see Fig. 1) can be tested as follows:

i) write a pattern into BADC memory;
ii) set interface device select register for CPU #0, #1 or both;
iii) enable Listen Mode and read BADC;
iv) disable Listen Mode and read 168/E data memory to compare with original pattern.

This type of test was important for system development but during the last nine months no errors have been observed.

To test the 168/E processor itself we have developed the Integer, Floating Point and Memory diagnostics listed in Table II. These diagnostic programs are written in assembly language and test all IBM 370 instructions supported by the 168/E. The programs have a common structure with definition of each test specified by a few registers. If a diagnostic program executing on the 168/E detects an error, it follows a standard procedure which includes dumping all registers to data memory for user examination. Checking for successful execution of these diagnostics is essential whenever the processor configuration or hardware are modified, but during scheduled data acquisition we usually rely on infrequent off-line

TABLE II. 168/E Diagnostics

Name	Instructions Tested
INTTEST1	Branch, Integer Load and Store
INTTEST2	Integer Arithmetic and Logical
INTTEST3	Logical Shift
INTTEST4	Arithmetic Shift
FLTTEST1	Non-Arithmetic Floating Point
FLTTEST2	Floating Point Arithmetic - Simple tests
FLTTEST3	Floating Point Shift Matrix
FLTTEST4 *	R*4 Floating Point Arithmetic
FLTTEST5 *	R*6 Floating Point Multiply/Divide
FLTTEST6 *	R*6 Floating Point Add/Subtract
MEMTEST *	Test any Window in Data Memory

* $\Longrightarrow$ Not executable on IBM 370/168.

checks to confirm normal processor operation. Perhaps the most serious flaw in the entire diagnostic procedure is the 168/E processor's inability to test its own program memory in the normal crate configuration. However, we have only experienced one program memory failure in the last two years.

3. NOVA FOREGROUND/BACKGROUND OPERATION

For SHF data acquisition the NOVA 4/X is run in a Foreground/Background mode. The Foreground program is written entirely in Data General-RDOS assembly language and performs all time critical functions while a run is in progress. Table III lists the sequence of 168/E related events which occur after the operating system responds to a CAMAC pre-beam interrupt and transfers control to the Foreground. Background tasks written mostly in Fortran perform functions which are not time critical and usually interactive, e.g., select a 168/E diagnostic. Background programs communicate with the Foreground via RDOS system calls which read/write to a communications array in the Foreground. The F/B communication array contains a set of control words for each 168/E processor, e.g., on/off flag, initial value for program counter, data memory address for algorithm results, etc. These control words allow 168/E hardware or algorithm development and real-time tests while a run is in progress. About 2.5 ms after each beam pulse the Foreground reads a forty word summary of algorithm results from each active processor and saves this summary in the communication array where it can be examined by the user.

TABLE III. Foreground 168/E Control

Time (μs)	Activity
- 2025	Pre-beam interrupt $\Longrightarrow$ RDOS suspends Background task and executes Foreground
- 1550*	Start match point processor and execute algorithm on 168/E CPU #1
- 1425*	Initialize algorithm on 168/E CPU #2 and Auxiliary Controller
0	Beam Pulse (100 ns)
+ 15	Z_α digitizer finishes in Crate 27_8 and sets trigger level to start Auxiliary Controller
+ 20	Z_β digitizer finished $\Longrightarrow$ trigger Auxiliary Controller
+ 33	Z_γ digitizer finished $\Longrightarrow$ trigger Auxiliary Controller
54	Auxiliary Controller starts algorithm on CPU #2
120-300	High Resolution camera trigger
620	DMA readout of Crate 25_8 complete $\Longrightarrow$ start algorithm on 168/E CPU #0
2500	Read results from all 168/E algorithms
3000	Main camera trigger
9500	DMA readout of BADC complete with data transferred to CPU #0
17000	Start monitor program on CPU #0
38000	Foreground suspends itself and RDOS restores Background execution
97975	Next Pre-beam interrupt

* These activities may be delayed by ~500 μs depending on the status of the background program at the time of the interrupt.

The Background start-run task reads 168/E control words and program/data filenames from a card image disk file and loads processor memory. Another task provides menus of diagnostic programs and algorithms with each menu read from a card image file. By editing these card image files, 168/E programs can usually be tested and added to production run procedures without modification to NOVA Foreground or Background programs. Since Background SNOOP I/O can always be interrupted by the Foreground, data transfer errors are inevitable. This happens infrequently for the size of programs used in this experiment and when a 168/E program load error is detected the procedure is simply restarted.

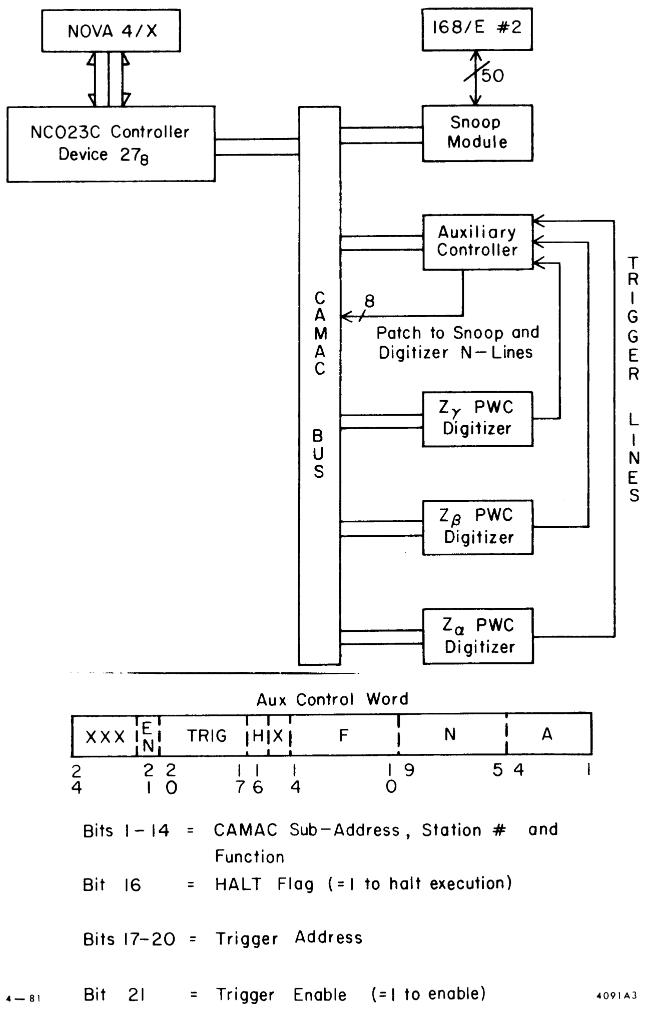
Algorithm execution time is measured in micro-seconds using a 168/E CAMAC clock module (see crate 27_8 in Fig. 1). Each interface board provides a TTL level which is true (high) when the processor is running. This level opens or closes a gate for clock pulses to a CAMAC scaler. On each beam pulse the Foreground reads the corresponding scaler sub-address for each active processor and saves this execution time in the F/B communication array. This clock module also has three LED's to indicate when each processor is running. Observation of the frequency and intensity of these LED's provides a convenient check for abnormal processor behavior.

4. TRIGGER ALGORITHM

The standard SHF charged track trigger uses PWC's in the nonbend plane (Z) to search for straight-line trajectories originating from the bubble chamber fiducial volume. When a good trajectory is found, the corresponding coordinates in the bend plane (Y) of the BC field can be used to calculate the momentum. Experiment BC72/73 uses High Resolution Optics (HRO) to measure charm particle decay lengths and requires a decision to take a HRO picture within 150-300 μs after the beam pulse — the main 40" BC camera still has a flash delay of 3 ms. Digitization and NOVA readout of the three primary PWC stations (α,β,γ with Z,Y,U at each station) requires a minimum of 150 μs and space point calculation on the 168/E takes a few hundred micro-seconds for normal multiplicities. So these procedures are too slow for the HRO trigger which was proposed after the 168/E system and SNOOP interface had been developed.

To meet the HRO time constraint the current algorithm uses only Z-planes and an Auxiliary Controller provides a fast data transfer to the 168/E. The Auxiliary Controller executes CAMAC functions sequentially from its control memory with a cycle time of 1 μs. It has a 1024 word control memory and a corresponding 1024 word data memory which contains input/output data for each control memory read/write. Control word format and the logical relationship of 168/E, controller and digitizer modules are shown in Fig. 2. CAMAC functions for NOVA I/O to the controller itself are listed in Table IV.

At the start of each run the Auxiliary Controller is loaded with the sequence of functions required to read Z_α, Z_β, Z_γ (8 hits and multiplicity for each plane) and start 168/E CPU #2. Before each beam pulse the Foreground arms the Auxiliary Controller and then avoids I/O to crate 27_8 until the HRO trigger sequence is complete. The first control word to read each PWC plane has the corresponding trigger bit(s) set so the controller waits for the digitizer module to finish and set a TTL trigger level (True = Low). Since the SNOOP module is in Listen Mode, data read by the Auxiliary Controller is transferred to CPU #2. When the Controller has finished reading Z_γ, it disables Listen Mode and starts the processor using SNOOP module CAMAC functions (see Ref. 3). As shown in Table III, this entire sequence is completed in 54 μs. The NOVA 4/X would require ~15 μs just to disable Listen Mode and start the 168/E while the Auxiliary Controller executes this sequence in 3 μs.



Bits 1-14 = CAMAC Sub-Address, Station # and Function

Bit 16 = HALT Flag (=1 to halt execution)

Bits 17-20 = Trigger Address

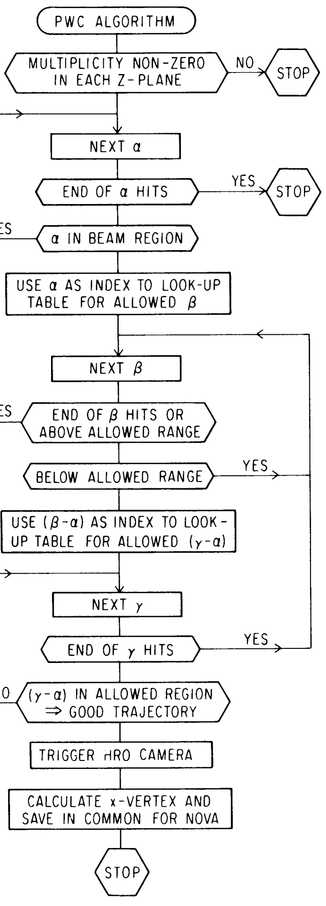
Bit 21 = Trigger Enable (=1 to enable)

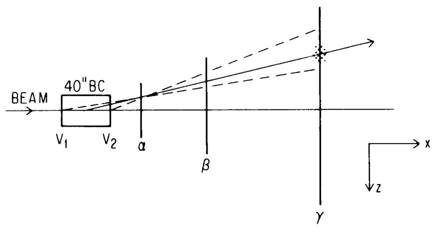
Fig. 2. Relationship of CAMAC Auxiliary Controller, PWC and SNOOP modules and control word format.

TABLE IV. Auxiliary Controller Functions

F	A	Description
0	0	Read Data Memory at S1 and Increment Address Register at S2
0	1	Read Control Memory and Increment Address
0	2	Read Address Register
1	0	Read Data Memory
1	1	Read Control Memory
8	0	Test Done Flag (LAM if Enabled)
16	0	Write Data Memory and Increment Address
16	1	Write Control Memory and Increment Address
17	0	Write Data Memory
17	1	Write Control Memory
17	2	Write Address Register
24	0	Disable LAM
25	0	STOP Execution and Clear LAM
26	0	Enable LAM
27	0	Start Control Memory Execution

A flowchart of the algorithm on CPU #2 and profile of the bubble chamber and Z-planes are shown in Fig. 3. The algorithm is written in IBM assembly language and uses look-up tables rather than a calculation of allowed regions for maximum speed. A single 168/E memory board is quite sufficient for this program. PWC data and algorithm results are saved in a COMMON Block located at a fixed address by the 168/E Translator. Consequently algorithm modifications seldom require changes in on-line constants which provide 168/E data addresses for the NOVA Foreground.


PWC ALGORITHM
MULTIPLICITY NON-ZERO IN EACH Z-PLANE
NO
STOP
NEXT α
END OF α HITS
YES
STOP
YES
α IN BEAM REGION
USE α AS INDEX TO LOOK-UP TABLE FOR ALLOWED β
NEXT β
YES
END OF β HITS OR ABOVE ALLOWED RANGE
BELOW ALLOWED RANGE
YES
USE (β-α) AS INDEX TO LOOK-UP TABLE FOR ALLOWED (γ-α)
NEXT γ
END OF γ HITS
YES
NO
(γ-α) IN ALLOWED REGION ⇒ GOOD TRAJECTORY
TRIGGER HRO CAMERA
CALCULATE x-VERTEX AND SAVE IN COMMON FOR NOVA
STOP



--- LOOK-UP TABLE BOUNDARY FOR β

····· ALLOWED REGION IN γ (SET BY MULTIPLE SCATTERING WHICH IS ±8 mm AT γ AND ±4 mm AT β)

$V_1 = -38$ cm, $V_2 = +38$ cm: BC VERTEX LIMITS

PWC	x (cm)	MAX WIRE*	BEAM REGION*	⟨HITS⟩†
α	80	240	± 9	1.7
β	121	312	±15	1.8
γ	203	600	± 27	1.3

*ONE WIRE # = 1mm (xz PROFILE IS NOT DRAWN TO SCALE)
†AVERAGE MULTIPLICITIES FOR BEAM FLUX OF 20 PHOTONS/PULSE

Fig. 3. Flowchart and BC/PWC XZ-profile for algorithm to find straight-line trajectories.

When a good trajectory is found the algorithm generates the HRO trigger by writing to a camera trigger card in the 168/E crate. This is a simple card with a few gate circuits to define two high data memory addresses. An access to these addresses sets or clears an external TTL level which provides the camera trigger. The algorithm can access this pseudo-memory card in 300 ns while the NOVA 4/X would need ~17 μs to poll the processor and set a CAMAC register when a trigger is detected.

CPU #2 algorithm execution time is shown in Fig. 4 — this distribution does not include Auxiliary Controller data transfer time. This algorithm usually executes in less than 140 μs and the fraction of triggers arriving too late for the HRO camera is insignificant. The PWC algorithm triggers on ~66% of σ_t (γp) and 42% of pictures taken with this algorithm have hadronic production in the bubble chamber. (A logical OR of the PWC algorithm and a

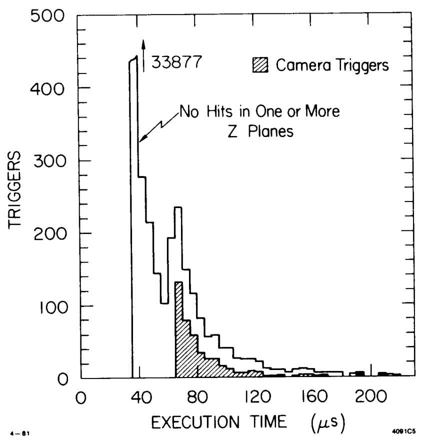


Fig. 4. PWC algorithm execution time on the 168/E.

dynode sum of energy deposited in the lead glass detector provides a trigger on ~90% of σ_t.) On the 168/E the algorithm runs ~2.4 times slower than on the IBM 370/168. Performance checks made by running the same algorithm and data off-line show no on-line processor errors since the start of production operation in July, 1980.

5. 168/E MONITOR PROGRAM

A Fortran program executing on 168/E CPU #0 monitors SHF data acquisition. Although NOVA Foreground and Background provide many monitor facilities, they each have limitations which are overcome with a dedicated processor. One is normally reluctant to modify the Foreground since it is vital for data acquisition and performance checks added to the Foreground decrease time available for Background execution. A Background program can only sample beam pulses and must be designed to be interrupted and replaced by another task while a run is in progress. The 168/E can monitor every beam pulse and the noninteractive Fortran program is easy to modify. Development/debugging are done on an IBM 370/168 and the same program can be used for both on-line and off-line applications without modification.

Currently the 168/E monitors data for three detectors — PWC, Čerenkov counters and Lead Glass. Statistics collected for each detector are listed in Table V. A subroutine which finds all charged tracks in the PWC system is used to calculate PWC efficiencies and Čerenkov counter performance. Monitor program execution time is 30-40 ms for normal PWC multiplicities so there is ample time available to support new routines and check other detectors (current pulse rate is 10 Hz).

TABLE V. Monitor Program Summary Data

A. PWC
1. Illegal wire data, e.g., zero or negative wire #'s, wire # disorder, etc.
2. Chamber multiplicities.
3. Chamber efficiencies.
4. Momentum and BC vertex distribution for PWC tracks.

B. Čerenkov
1. Cell # and multiplicity.
2. Average pulse height and pedestal for each cell.
3. Momentum distribution for PWC tracks with signals above theshold in $\check{C}_1$ and $\check{C}_2$.

C. Lead Glass Columns
1. Pulse height distribution in active converter and absorber.
2. Pedestals for lead glass blocks, reference counters and hodoscopes.
3. Average LED pulse heights for lead glass blocks and reference counters.

The program on CPU #0 and related Foreground control logic include support for a trigger algorithm. The 168/E MAIN program calls an assembly language algorithm and the monitor in the following sequence:

i) Foreground starts MAIN which calls the algorithm;

ii) algorithm finishes and executes a HALT instruction to stop the processor;

iii) Foreground reads algorithm results;

iv) lead glass data (not required by the algorithm) is transferred to CPU #0;

v) Foreground restarts the 168/E at instruction following HALT and algorithm returns to MAIN which calls monitor.

As indicated in Table III, the monitor program is executed after all trigger related Foreground functions and CAMAC data transfers are complete. All monitor input/output data and control variables are saved in COMMON blocks. The Foreground sets a single control flag which directs the monitor to initialize itself, collect data or produce a summary. At the end of each run a Background program reads this summary, saves it on disk and prints the results.

6. 168/E DIGITIZER CARD

Although the Auxiliary Controller provides satisfactory support for the algorithm using three Z-planes, data transfer time would become excessive for a trigger using all nine primary chambers. There is also a fourth station (δ) with Y- and Z-planes downstream of γ. Data transfer overhead is eliminated completely by placing the PWC digitizer in the 168/E crate where results are available as read only data memory.

The 168/E digitizer[5)] with logic for one channel is shown in Fig. 5 — each card has eight identical channels. Before a digitizing sequence can begin a "Clear" pulse, usually generated externally but optionally from the 168/E, resets the Wire Address and Word Counters. Signals from PWC wires set corresponding bits in a shift register which, when shifted by "Clock" pulses, are presented as "Data" signals to the digitizer together with "Clock" pulses. The "Clock" signal increments a Wire Address Counter which contains the wire #. If a "Data" pulse also arrives indicating a wire has been hit, the contents of the Wire Address Counter are loaded into memory for that channel. "Clear", "Clock" and "Data" signals are input through BNC connectors added to the front panel of the 168/E crate.

The loading sequence increments a Word Count register which also acts as an address register for channel memory. Each channel can hold sixteen 12-bit wire numbers in sequential locations and input which exceeds this limit is lost. The eight channels operate in parallel and the digitization sequence ends when the Wire Address Counter reaches its (selectable) terminal count and sets the "Ready" flag. The 168/E algorithm is started before the beam pulse so it can interrogate the "Ready" flag and proceed as soon as data is available.

To read the digitizer, Data Address signals from the 168/E bus are multiplexed onto channel memory address inputs as shown in Fig. 5. Channel memory data are read through a block of 128 (8×16) addresses followed by address space for eight word counts. The base address is selectable by switch settings — it is currently memory card #3 with address

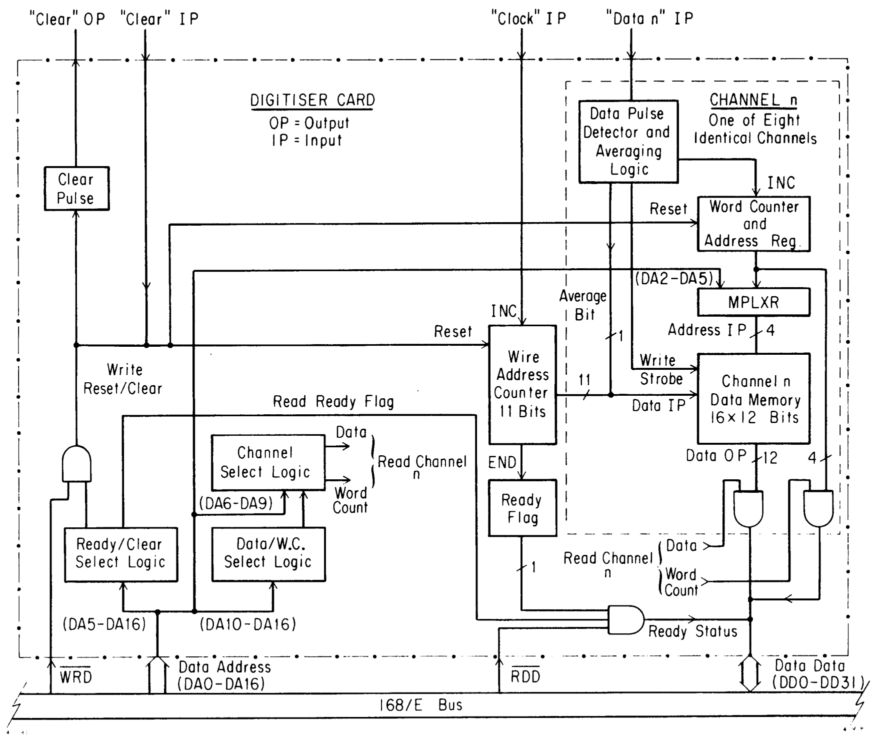


Fig. 5. Block diagram of 168/E PWC digitizer card.

$DC00_{16}$ for channel wire numbers and $DE00_{16}$ for multiplicities. Wire numbers are read onto Data Data lines 0-11 and multiplicities normally go to 168/E Data Data lines 0-3. In addition to this normal access mode, a switch option allows word counts to be included with wire numbers on Data Data lines 12-15. "Ready" flag and "Clear" pulse share a common address ($FC00_{16}$) which can also be selected by switches. A read to this address returns the "Ready" flag on Data Data line 15 (1 = ready). A write to this address generates "Clear" and reset pulses.

Development of the Digitizer card was completed recently, and it is currently operational on 168/E CPU #1. The production algorithm from CPU #2 has been modified to read the digitizer and is normally executed on every beam pulse. CPU #1 algorithm results are read by NOVA Foreground and saved on SHF data tapes for off-line analysis.

7. MATCH POINT RESULTS MEMORY

The Z,Y,U planes at each of the three PWC stations (α,β,γ) form a 3-4-5 right triangle and "Space" or "Match Points" are defined by the relation:

$$|3Z + 4Y - 5U - \text{constant}| \leq \text{tolerance} \quad .$$

A hardware match point-finder (one for each PWC station) has been developed to make match point results available within the time constraint of the HRO trigger.[6] After finding a straight-line trajectory in the Z-planes, the 168/E algorithm could use match point results to check for one or more confirmation hits in the Y-planes and calculate momentum.

The interface between the point-finder and 168/E is a Results Memory card in the 168/E crate. The interface was designed to meet the following requirements:

i) compatibility with existing 168/E systems;

ii) results from each point-finder should be accessible as soon as it has finished;

iii) the 168/E program should be able to check match point-finder status;

iv) the 168/E should be able to initialize each result's memory location.

We considered the option of a multi-port memory permitting DMA transfers from several external sources. This would require logic to arbitrate contention between two or more ports including the 168/E CPU itself. This option fails requirement i) since 168/E data memory control logic would have to be modified to allow a temporary delay in execution of a memory access.

A block diagram of Results Memory designed to meet the above requirements is shown in Fig. 6. Three separate memory units are write-only from each point-finder and read-only from the 168/E. Match points for a PWC station are transferred to the corresponding memory unit over a 50-wire flat cable (one cable from each point-finder). Each memory unit has two blocks, P and R, and each block has sixteen halfwords, permitting a maximum of sixteen matched (P,R) coordinate pairs per station. If coordinate pairs are transferred they would be loaded into sequential locations in memory with 3*Z in P and 4*Y in the R block. The α point-finder currently transfers 4*Y to a location in P with address set by the corresponding Z hit.

The memory chip used, 74S289, has the advantage for this application of separate data-in and data-out lines. However, to avoid the possibility of putting spurious signals on the 168/E data bus when the point-finder writes to memory, the output lines are buffered by 25LS240 bus drivers. These 3-state outputs are only enabled when a 168/E access is

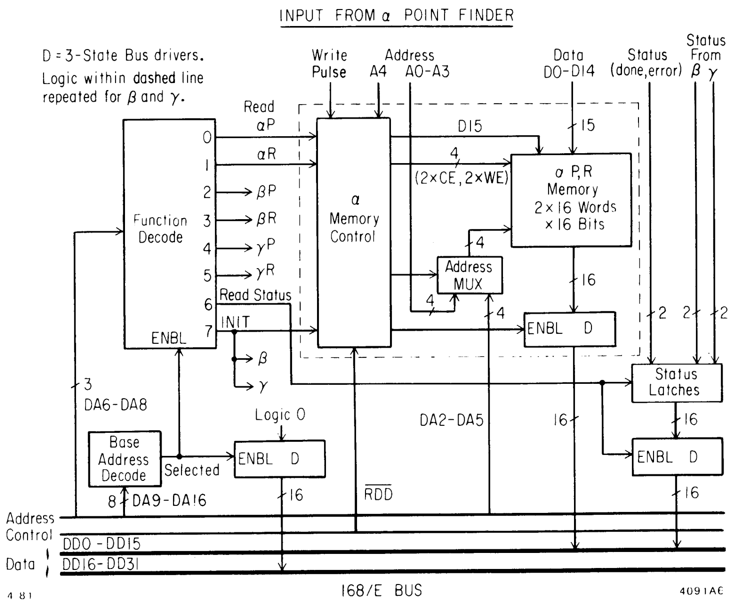


Fig. 6. Block diagram of 168/E Results Memory card.

decoded. To make this pseudo-memory compatible with 168/E logic the memory access time plus propagation delay of the bus drivers must be less than or equal to that of normal 168/E memory which is ~55 ns. The 168/E PWC digitizer uses the same type of memory and bus drivers.

Results memory is currently memory card #2 on CPU #1 with a base address of $9C00_{16}$. Memory-to-Register load instructions are used for both control and data input with eight functions decoded from data address lines 6-8 as shown in Fig. 6. Memory unit addresses are multiplexed with the address from each point-finder normally selected. For read functions 0-5 the multiplexor is switched to select P,R memory locations from 168/E data address lines 2-5. Data from a selected memory unit is placed on the least significant half of the 168/E data memory bus (DD0-DD15) and the upper halfword is logic zero.

To avoid possible interference with point-finder input, a 168/E program should not attempt to read α,β or γ memory units until the corresponding data transfer is complete. The status word with ERROR and DONE bits for each point-finder can be read at any time (function = 6):

	γ	γ	β	β	α	α
Not Used	D	E	D	E	D	E
31		4		2		0

Results Memory Status Word

ERROR bits are set to one and DONE bits to zero when the point-finders are initialized. Each point-finder clears its ERROR bit when the first match point is transferred and sets the DONE bit when processing is finished.

The seventh function, initialization, sets a logic one in bit 15 of the addressed word (DA2-DA5) in all six memory blocks. The algorithm on CPU #1 is started before the beam pulse and initializes all memory blocks by reading the sixteen words with data address bits 6-8 set to one for function seven. The algorithm polls the status word to determine when match point data is available and could use bit 15 in each results memory location as a flag for valid data. The α match point-finder is operational and Results Memory has been read by a 168/E algorithm, saved on NOVA run tapes and verified off-line.

8. SUMMARY

The 168/E processor has been versatile and stable in the SHF on-line operating environment. Program development and hardware tests can be done with minimal dedicated time on the NOVA host computer. Most algorithm development and testing is done off-line on an IBM 370 and most Results Memory tests were made during production runs. The PWC Digitizer and Results Memory cards provide fast 168/E data input, however, they also require algorithms which cannot be completely tested off-line. Diagnostic procedures for these cards would be more robust if their design allowed the 168/E to write any valid pattern to Results Memory and execute production algorithms using this test data. Results Memory is a convenient approach to special 168/E I/O since design only requires familiarity with the processor bus and timing.

ACKNOWLEDGEMENTS

We thank D. Bernstein and L. Paffrath for design and development of the SNOOP module interface and Auxiliary Controller. Consultation with P. F. Kunz has been most helpful in development of this system and A.V. Bevan provided software development for the point-finder.

REFERENCES

1. K. C. Moffeit et al., SLAC Proposal BC72 (1979); G. Kalmus et al., SLAC Proposal BC73 (1979).

2. P. F. Kunz et al., "The LASS Hardware Processor," Proc. of the 11th Annual Microprogramming Workshop, SIGMICRO Newsletter 9, 25 (1978).

3. D. Bernstein et al., "SNOOP Module CAMAC Interface to the 168/E Microprocessor," Proc. of the 1979 Nuclear Science Symposium, IEEE Trans. on Nucl. Sci. NS-27, 587 (1980).

4. M. Breidenbach et al., "Semi-Autonomous Controller for Data Acquisition the Brilliant ADC," Proc. of the 1977 Nuclear Science Symposium, IEEE Trans. on Nucl. Sci. NS-25, 706 (1978).

5. This digitizer was adapted from a design by D. Freytag, "A Compact Time Digitizer in CAMAC Format," Nucl. Instrum. Methods 138, 685 (1976).

6. BC72/73 Collaboration Letters from Dave Price, Imperial College (1979-1980).