SLAC-PUB-2633
October 1980
(I)

THE VAX CAMAC CHANNEL*

CONF-801163--34

D. J. Nelson, M. Breidenbach, C. D. Granieri,
J. E. Grund, J. F. Patrick, and L. J. Weaver**

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

ABSTRACT

A new generation CAMAC System has been developed for the Mark II Detector at SLAC's PEP storage ring. This flexible system can efficiently transfer data between a host computer and a very large set of CAMAC data acquisition and control modules. A bipolar microprocessor operates as a "Channel" interface by supervising the CAMAC system and minimizing the host computer's work. This programmable channel couples the host to a set of "System Crates"; each System Crate houses "Branch Drivers" that can directly control a set of crates or communicate over differential parallel highways to "Branch Receivers" for control of distant crates. A coherent software package integrates the high level programs, system driver level programs, and microcode control of the system.

BACKGROUND

The Mark II CAMAC system is basically an evolved version of the CAMAC system designed for the Mark I Detector at SPEAR that coupled a Xerox Sigma V Multiplexor Input/Output Processor (MIOP) to Branch Drivers. The VAX CAMAC Channel (VCC) is roughly analogous to the Xerox MIOP and its CAMAC interface, although designed to operate with the Digital Equipment Corp. (DEC) 32 bit VAX 11/780 computer through its PDP 11 style UNIBUS. A CAMAC system was needed that could be invoked simply and cleanly from high level software; that could control a large number of CAMAC branches, possibly at separated locations; that could rapidly transfer large amounts of data using various block transfer modes with little CPU overhead; and that had separated modular functions for increased reliability and ease of maintenance. These requirements have been met by a bipolar microprocessor based programmable Channel coupling the host computer to a tree structured set of System Crates, Branch Drivers, and CAMAC Branches. The channel, or VCC, fetches commands from VAX main memory, sets up the Branch Drivers and transfers data between the CAMAC modules and VAX main memory. Address scanning for block transfers, as well as long distance signal transmission, is handled by the Branch Drivers. Data acceptance conditioned by CAMAC X and Q responses, and data packing, are performed by VCC. Communication with non-CAMAC peripherals or data links can be accomplished through special modules in the system crates.

SYSTEM OVERVIEW

The connecting hardware between the VAX (host computer) and the many CAMAC modules that comprise the input/output system of the Mark II detector is diagrammed in Figure 1. The host connection is made by a UNIBUS Interface circuit board that plugs into the UNIBUS of the VAX. Data and control signals are transmitted from the UNIBUS board to the VCC chassis through flat cable.

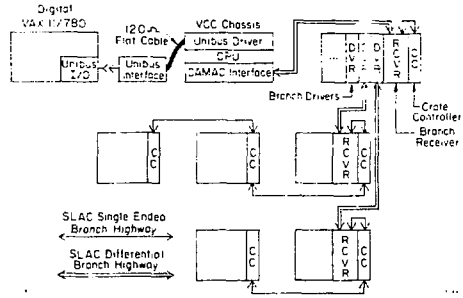

Figure 1: System Overview

The VCC chassis, which is physically located in a rack of the VAX, is pictured in Figure 2. The VCC takes directives from the VAX and controls the CAMAC system through System Crates.

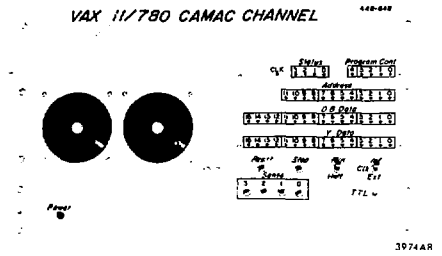

Figure 2: The VCC Front Panel

*Work supported by the Department of Energy under contract DE-AC03-76SF00515

**Presently with Digital Equipment Corp.
9 North East Blvd., Salem, NH 03079

Contributed to the 1980 IEEE Nuclear Science and Nuclear Power Systems Symposium, Orlando, Florida, November 5-7, 1980.

The VCC talks to the System Crates via a "SLAC Differential Branch Highway" cable. This differential TTL cabling system allows System Crates to be distant from the VCC if desired. A Branch Receiver converts the Differential Branch to a "Single Ended Branch Highway" compatible with the SLAC standard Crate Controllers[1]. Along with the Branch Receiver and Crate Controller a System Crate has a number of Branch Drivers. Each Branch Driver can drive a Highway of up to 7 crates. Since Single Ended Branches become problematic after 15 meters of cable, the Branch Driver provides a Differential Branch that has been used with over 1 km of cable. Crates on the Highways use Branch Receivers and Crate Controllers that are identical to the System Crate's modules. A system of 56 Branches and 392 crates could be assembled with this hierarchy.

The VCC

The heart of the system is the VAX CAMAC Channel or VCC. It consists of a high speed 16 bit CPU, an interface to a DEC UNIBUS, and a CAMAC interface that runs the SLAC CAMAC Protocol.

Figure 3: The Top View of the VCC

VCC's CAMAC and UNIBUS Interfaces are peripherals to the CPU. Communication between them and the CPU is via 4 buses:

1. A DB Bus for data to the CPU,
2. A Y Bus for data from the CPU,
3. A STATUS Bus from the Interfaces, and
4. An I/O µCODE Bus to control the Interfaces.

These buses are ribbon cables connecting the boards of the VCC, as shown in Figure 3. To explain the operation of the VCC we first describe its Interfaces.

UNIBUS Interface. The UNIBUS Interface consists of 2 circuit boards: one that plugs into the VAX UNIBUS and one that resides in the VCC chassis. Figure 4 shows the main components of these boards. The VAX can perform programmed I/O transfers to send "Start I/O" commands and read a "Test Device" or TDV status register. The VCC can become UNIBUS Master to DMA data to and from VAX memory. A VAX vectored interrupt can also be performed.

The UNIBUS Driver board in the VCC chassis has the required data line drivers and receivers and a peripheral instruction register. VCC peripheral control is implemented as part of the CPU microcode. This part of the microcode is made available to the peripherals which decode and latch it for their instructions.

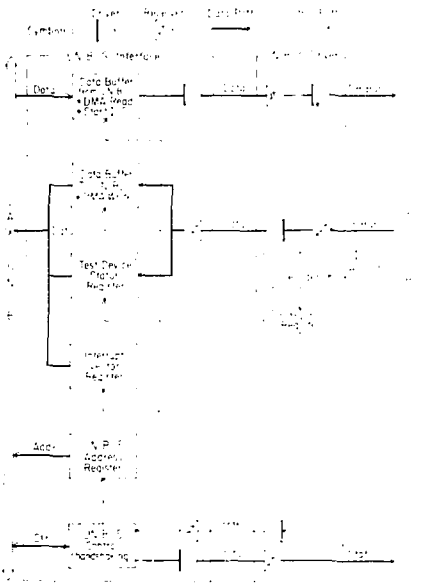

Figure 4: The UNIBUS Connection

CAMAC Interface. The CAMAC Interface is similar to the UNIBUS Interface. Again the I/O µCODE controls the operation. Control registers are loaded to select a Branch Driver in a System Crate and to read or write to it.

"Byte Shifters" in the data path allow three modes of mapping the 24 bit wide CAMAC data words into the 32 bit words of the VAX. These packing modes are defined in Figure 5.

The timing of the CAMAC cycle is generated at the host computer interface in the SLAC protocol[1]. The CAMAC cycle generated by this board has a programmable cycle speed, in steps of 1.6 µs, so that the system can

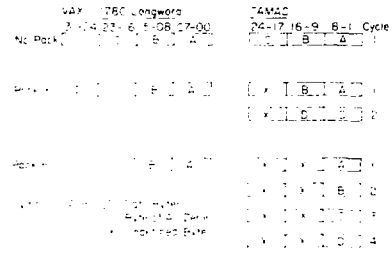

Figure 5: Packing Modes

...ast data transfer rates, thereby allowing adequate propagation time for very long cables while not penalizing transfers that can be accomplished quickly.

Figure 6: The CAMAC Connection

The CPU. The CPU is designed around AMD 2900 Bit Slice bipolar microprocessor components which are assembled into a 16 bit wide processor. It incorporates the 2903 ALU and Registers, 2904 Status and Shift Control Unit, 2902 Carry Look-Ahead Unit, 2910 Microprogram Controller, a 256 word RAM and a Microprogram ROM. The processor is organized with one level of pipelining by overlapping fetch and execute cycles.

The detailed operation of VCC is microprogrammed in fusible link PROM's forming a 64 bit 256 word memory that is expandable to 4096 words. A 256 word fast RAM, implemented as an I/O device, is available as a scratch pad in addition to the 17 registers of the microprocessor. Stand alone microcoded test routines allow VCC to evaluate the CAMAC system status and exercise the interface to the VAX to verify proper UNIBUS operation.

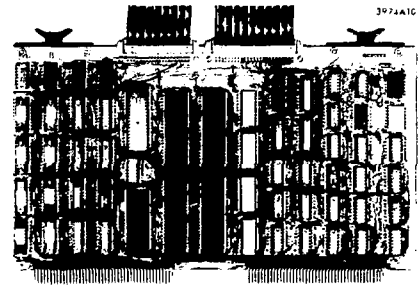

Figure 7: The Wire-Wrapped CPU Circuit Board

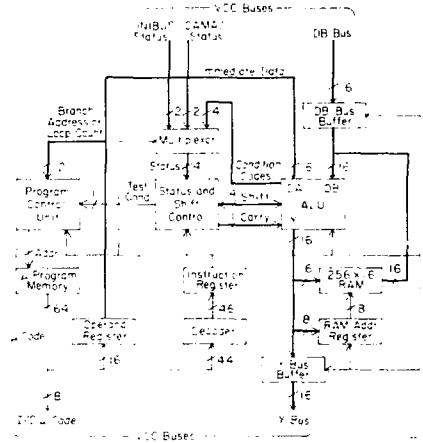

Figure 8: The VCC CPU

The CTLW. The CAMAC functions, the CAMAC address and type of block transfer, if used, are defined for the VCC and Branch Driver by a Control Word (CTLW). The 32 bit longword CTLW, defined in Table 1, is passed from the VAX to VCC. The most significant byte (bits 31 to 24) is exclusively for VCC while the rest of the CTLW (bits 23 to 00) is sent to the Branch Driver.

The Branch Driver

The concept of System Crates with Branch Drivers is to add a level of indirection to CAMAC which increases the effective address space. The initial Destination Address (Crate, Module and Sub-Address), Function Code, and Scan Mode is sent to the Branch Driver by VCC writing 24 bits of the CTLW via an F(17). Subsequent data transfers are accomplished with F(0) for read and F(16) for write functions The overhead introduced by

TABLE 1

The Control Word - CTLW

	X	X	Q	Q	P	P	C	C	C	M	M	M	M	M
-	S	M	M	M	M	8	1	4	2	1	1	8	4	2	1
		1	2	1	2	6					6				
31						24	23								16

A	A	A	A	I	C	I	I	S	S	S	F	F	F	F	F
8	4	2	1			N	L	C	M	A	1	8	4	2	1
						Q					6				
15						08	07								00

S=1	Synchronize with R response from CC
XM1,QM1	Transfer data iff X or Q=1, resp.
XM2,QM2	Terminate transmission if X or Q=0
P8,P16	Packing Modes
Cx,Mx,Ax	Initial Crate, Module & Subaddress
I, C	CAMAC Inhibit and Clear lines
SC,SM,SA	Enable Crate Module & Sub. scanning
ILQ	Increment least sig. cntr iff Q=0
IN	Reset least & incr next cntr if X=0

writing the CTLW to the Branch Driver is a factor of two for single CAMAC cycle transfers, but the vast majority of transfers are large blocks, often utilizing the scanning and data acceptance capabilities of the system. After each CAMAC cycle the effective CAMAC address can be updated conditioned on the X and Q responses of the addressed module. 24 scanning modes and the VCC's ability to condition data transmission on X and Q responses allow flexible block transfers to be implemented. A block transfer is terminated by satisfying the byte count, the XM2 or QM2 requirement, or by the Branch Driver signaling "End of Scan" via the System Crate L lines. A non-dataway LAM system is independently implemented in the Branch Driver for LAM's generated on the branch. An F(1) instruction to the Branch Driver returns the current value of the CTLW register, permitting the ending scan address to be part of the VCC status block.

A handshaking technique has been defined to eliminate system contention when autonomous crate controllers are used. A new signal line is provided for status information from the Crate Controller of the addressed crate on the Branch. This new status signal, called R, has the meaning that the receiving Crate Controller is Ready to Execute the dataway cycle as instructed. In future systems with updated Crate Controllers, the R signal will be generated by the Controller in response to the Busy signal after any cycle in progress. When the S bit is set in the CTLW, VCC will wait for the R signal before generating S1 and S2 of the CAMAC cycle.

SOFTWARE

An operation of the VCC is initiated by a software device driver passing VCC the addresses of control and data buffers via the Start I/O register. No further VAX CPU activity is required until VCC has completed all the data transfers implied by the control "channel program" buffer. The high level language programmer effectively defines a set of "packets" which constitute the channel

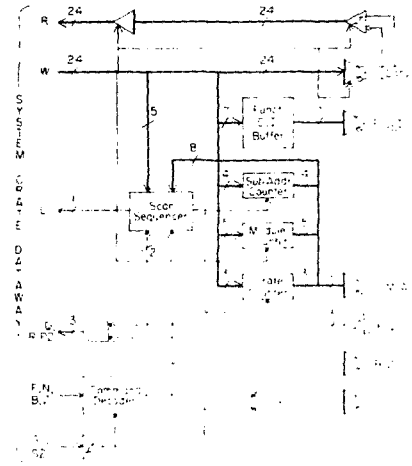

Figure 9: The Branch Driver

program. Each packet defines a set of CAMAC operations by specifying the CTLW, a maximum byte count, and buffer positions for the data. Thus a packet can describe a single control cycle or a block transfer involving many crates. VCC processes each packet and interrupts the UNIBUS when all have been completed.

The operation of VCC is, of course, defined by its software, and is thus expected to evolve as new applications arise. The basic organization starts from a small set of Fortran callable routines that format a "channel program" and define data and status buffers. The parameters of these buffers are passed to a device driver by the VMS QIO system service. The driver is responsible for forcing and locking these buffers into physical memory, assigning UNIBUS addresses to the buffers, and allocating buffered datapaths through the UNIBUS Adapter. A VCC operation is initiated when the driver passes the UNIBUS addresses of the three buffers and two other words of information to VCC by programmed data writes. VCC then takes over, accessing the buffers by three separate DMA datapaths and executing the channel program. The process can abort if a time-out occurs or the data buffer is too small. The capability for VCC to perform conditional braches in the channel program has been designed but not yet implemented.

The basic Fortran subroutine is

CAMIO (BRANCH,CTLW,DATA,DATALEN,STATUS)

where BRANCH defines the Branch Driver to be addressed, and CTLW was defined in Table 1. The other arguments are optional. DATA is the address of a data buffer; if it is not specified, data coming towards the VAX is

flushed and data written to CAMAC has a value of 0. DATALEN is the length of the data buffer in bytes. If it is not specified, it defaults to the VAX normal word length of 4. The status buffer is 4 words long containing the final status of the Branch Driver control register, the actual number of bytes transferred, and the TDV status from VCC. If the STATUS buffer is not specified, the status information generated by VCC is ignored.

The second Fortran subroutine is

CAMIOS (NPAK,BRANCH,CTLW,DATA,DATALEN,STATUS)

and is logically equivalent to NPAK separate calls to CAMIO. CAMIOS constructs a channel program having NPAK packets; all the other arguments of CAMIOS are analogous to CAMIO except that they are extended to arrays of length NPAK. The DATA buffer is usually treated as a linear buffer, however. The purpose of CAMIOS is to avoid the overhead of the multiple calls to QIO that would result from the repeated use of the more primitive CAMIO.

Another set of high level routines has been developed for the PEP storage ring instrumentation and control project. In this system, serial Branch Drivers communicate with microprocessor crate controllers using an SDLC protocol. The basic protocol is that the driver transmits a message block to a crate controller and that crate controller replies, so that there is no interlaced activity among the crates on a serial Branch. The subroutine PEPCAM generates a channel program which can launch messages down each serial Branch, then poll the Branch Drivers for the returned data and scatter write it into VAX memory. This basic cycle can be repeated many times, so that a database reflecting the status of the PEP machine can be maintained with almost no CPU supervision.

The VCC microcode is written in MIMIC[2], a general purpose microcode assembler. A simplified flow chart of the microcode is shown in Figure 10. The basic cycle, transferring 2 bytes per CAMAC cycle, requires 8 microinstructions.

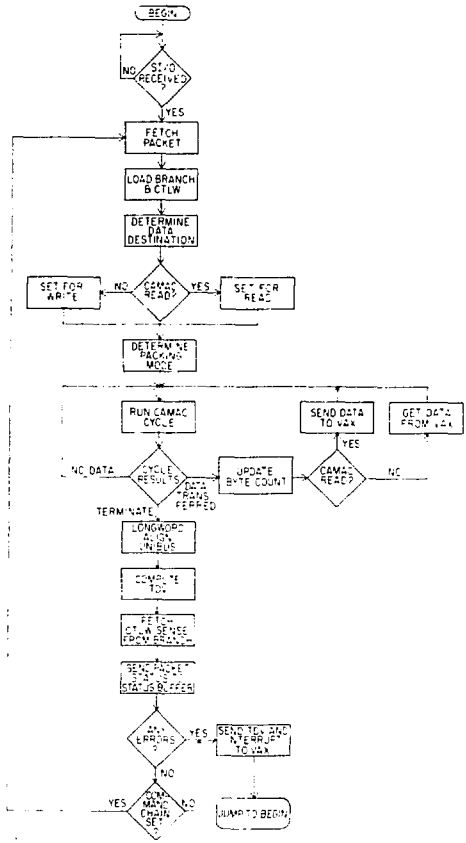

Figure 10: VCC Microcode Flow Chart

SUMMARY

The VAX CAMAC Channel answers the problem of interfacing an input/output system to a computer when their word sizes, address space structure and handshaking protocols are incompatible. A general purpose microcoded processor was coupled with hardware mated to the systems to be interfaced. This architecture is flexible enough to allow strong optimization of the communication between the two systems. While the System Crates normally house modules which are CAMAC Branch Drivers, they may be interfaces to other I/O organizations.

REFERENCES

(1) Dale Horelick, IEEE Trans. on Nucl. Sci. NS-22, No. 1, 517 (1975).

(2) M. Breidenbach, E. Frank, J. Hall and D. Nelson, IEEE Trans. on Nucl. Sci. NS-25, No. 1, 706 (1978).