SLAC – PUB – 3781
September 1985
(I)

SLAC SCANNER PROCESSOR APPLICATIONS IN THE DATA ACQUISITION SYSTEM FOR THE UPGRADED MARK II DETECTOR*

T. Barklow, T. Glanzman, A. J. Lankford and K. Riles
Stanford Linear Accelerator Center, Stanford University, Stanford, California 94305

ABSTRACT

The SLAC Scanner Processor is a general purpose, programmable FASTBUS crate/cable master/slave module. This device plays a central role in the readout, buffering and preprocessing of data from the upgraded Mark II detector's new central drift chamber. In addition to data readout, the SSPs assist in a variety of other services, such as detector calibration, FASTBUS system management, FASTBUS system initialization and verification, and FASTBUS module testing.

1. INTRODUCTION

The SLAC Scanner Processor (SSP) is a general-purpose FASTBUS module combining the functions of a FASTBUS master and slave on both crate and cable segments, a large buffer memory, and a programmable central processing unit.[1] FASTBUS I/O is accomplished by dedicated hardware, and many of the standard protocols are supported. Processing power is provided by a 32-bit bit-slice CPU implementing a subset of the IBM System/370 integer instruction set supplemented by a set of FASTBUS I/O instructions. The main buffer memory consists of 128K bytes, but can be expanded to 512K bytes through the use of 64K static RAM chips.

Although this device was specifically designed to support crate level readout and processing of TDC and FADC modules within the data acquisition for the upgraded Mark II detector,[2] its capabilities are sufficiently general for other applications within the FASTBUS system. The Mark II, which will begin operation at the SLAC Linear Collider[3] in early 1987, provides an example of a FASTBUS system utilizing multiple SSP's with different responsibilities. SSP's occupy three distinct positions and many more roles within the present Mark II data acquisition system[4] at PEP (See Fig. 1). Each FASTBUS crate of data acquisition modules is directly controlled by its own 'remote' SSP. All of these crates are grouped together on a small number of FASTBUS cable segments. Each cable segment is managed by a 'system' SSP which communicates directly with the VAX host. Finally, a cluster of 3081/E processors[5] are controlled by a pair of SSP's which manage the inward and outward data flow and monitor processor status. In addition, SSPs are used offline in the FASTBUS test bench for the checkout of new FASTBUS modules, the diagnosis and testing of faulty modules, and software development.

The SSP supports programs coded in either IBM assembly language or FORTRAN 77. Special functions such as FASTBUS I/O are provided by assembler macros and a library of FORTRAN-callable subroutines. Object files are then translated, a process which modifies the code to accomodate the special features of the SSP, and linked to construct an executable image. Images are downloaded first to the VAX host, and then to the SSP itself through the VAX-FASTBUS interface and FASTBUS system. SSP operation can be verified by a complete set of diagnostic routines which test both the internal (arithmetic and logical) and FASTBUS I/O instructions, and the memory.

This paper will present a summary of some SSP operational details, and will then discuss the specific applications of the SSP to the Mark II system.

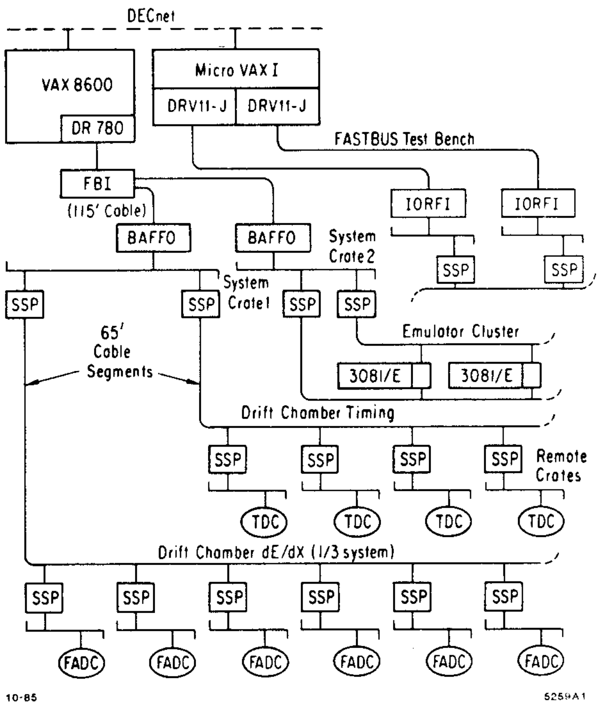

Fig. 1. Mark II FASTBUS System at PEP.

2. SSP I/O Operation

A partial operational description of the SSP has already been given in Ref. 1, which discusses the hardware instruction set, memory configuration, processing speed, and other features. The utility of the SSP is largely due, however, to its FASTBUS I/O characteristics. Control of the SSP is exercised through two front panel LEMO connectors, and via FASTBUS through a set of control registers and memory locations.

In addition to an assortment of informative LEDs, the SSP's front panel contains START input and DONE output connections. The START signal is ORed with bit 11 in CSR0, and serves as a means for starting the SSP with a simple NIM level pulse, such as an event trigger. The DONE output is a reflection of bit 5 in CSR0, and is a convenient means for the SSP to signal some condition to an external logic module.

There are six special registers and memory locations within the SSP used to control its operation. With the exception of the PSW, all are directly accessable via the connected FASTBUS segments.

CSR0: This register contains the primary control bits governing SSP I/O operation. Each bit is described in Table 1.

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

Presented at the Nuclear Science Symposium, San Francisco, CA, October 23-25, 1985.

TABLE 1: CSR0

SSP Bit	Power-on default	Meaning on READ if bit is set	Meaning on WRITE if bit is set
0	0		STOP
1	0	SR (crate)	Enable RB (crate)
2	0	RUN enabled	Enable RUN state
3	0	SR (cable)	Issue RB pulse
4	0	SR enabled (crate)	Enable SR (crate)
5	0	DONE (SR) status	Assert DONE (SR)
6	0	Single step enabled	Enable single step mode
7	0	GKUP (maintain GK)	Enable GKUP
8	0	Crate segment selected (0=> cable selected)	Select crate segment
9	0	SR enabled (cable)	Enable SR (cable)
10	0	Timeout (TO) inhibited	Enable TO inhibit
11	0		START processor
12	-	Device Class: 4-bit	Device Class: 4-bits
13	-	value used in	(these bits are set
14	-	broadcasts (case 1)	only when bit 16
15	-		is also set)
16	0	Mfg. ID (0106 hex)	Enable Device Class
17	1	Mfg. ID	Enable RB (cable)
18	1	Mfg. ID	Disable RUN (hard STOP)
19	0	Mfg. ID	
20	0	Mfg. ID	Disable SR (crate)
21	0	Mfg. ID	Lower DONE (SR)
22	0	Mfg. ID	Disable single step
23	0	Mfg. ID	Disable GKUP
24	1	Mfg. ID	Select cable segment
25	0	Mfg. ID	Disable SR (cable)
26	0	Mfg. ID	Disable TO inhibit
27	0	Mfg. ID	
28	0	Mfg. ID	
29	0	Mfg. ID	
30	0	Mfg. ID	
31	0	Mfg. ID	

Note: SR is asserted on either or both FASTBUS segments when both DONE is asserted and SR is enabled. RB is issued as a one microsecond pulse on only one segment at a time. Bit 3 in addition to either bits 1 or 17 are necessary to assert RB.

CSR8: This register contains the arbitration vector used for both crate and cable operation, and is write only.

DM0: Data memory location 0 is used to contain the initial program status word (PSW). When an idle SSP receives a START, the PSW is loaded from DM0 and execution commences. Any action resulting in an error condition causes the current PSW to be dumped to DM0.

DM4: The second full-word in data memory is used to dump the contents of the memory address register (MAR) whenever an error condition occurs. The contents of this register are used, for example, when an SS=2 response is encountered during a block read in order to compute the word count transferred.

PM0: The first location in program memory is the hardware trap location upon detection of any error. This location generally contains a branch instruction to an error handling routine.

PSW: The Program Status Word contains the program counter (PC) and 20 status bits used to describe processor status. These bits are described in Table 2. Note that the PSW is generally not available for reading except after an error occurs at which time it is dumped into DM0.

TABLE 2: THE PROGRAM STATUS WORD

INTR (MSB)	EXCP	TOUT	SS CODE	BREQ	EG	ASAK	MS	RD	WAIT	EX	NPA	SMSK	OMSK	CC	PC (LSB)
31	30	29	28 27 26	25	24	23	22 21	20	19	18 17	16	15	14	13 12	11 0

INTR	-	External interrupt (attached as FASTBUS slave)
EXCP	-	Exception (divide by 0, overflow, unsupported opcode, or block transfer word count overflow)
TOUT	-	Timeout during FASTBUS operation
SS	-	Slave status at time of I/O error
BREQ	-	Arbitration cycle in progress at time of error
EG	-	Address cycle in progress at time of error
ASAK	-	ASAK lock present at time of error
MS	-	Mode select bits (MS0, and MS1) at time of error
RD	-	Read in progress at time of error (else, write in progress)
WAIT	-	WT was being asserted at time of error
EX	-	Exception code: 00 [0] => divide by zero; 01 [1] => overflow; 10 [2] => unsupported opcode; 11 [3] => block transfer count overflow
NPA	-	Next PROM Address. This bit is set during execution of a multiple PROM cycle instruction. if set, the PC must be decremented by one upon an error in order to return to the next instruction in sequence.
SMSK	-	IBM system mask (enables external interrupt)
OMSK	-	IBM overflow mask (enables overflow interrupt)
CC	-	IBM encoded condition code
PC	-	Program counter (or instruction address in Program Memory)

Initial PSW: PC = starting address of program
Error PSW: PC = one beyond instruction causing error

Special attention is directed to PSW bit 15 (SMSK) which governs how the SSP responds as a FASTBUS slave while it is running. If *enabled* (SMSK=1), once an AS/AK lock is established, DS up results in the SSP returning WT (wait). This condition persists while the current instruction completes, and the error handler is called upon to halt the CPU (the PSW dumped to DM0 has bit 31 set). Once halted, WT is dropped, DK raised and the transaction resumes. If SMSK is *disabled*, DS up results in an SS=1 (busy) response. Note that any successfully interrupted SSP must be explicitly restarted.

3. SYSTEM SSPs

System SSPs reside in a FASTBUS crate directly accessible to the VAX host. Within the Mark II architecture, system SSPs mediate all transactions between the VAX and remote SSPs, hence replace segment interconnects. In addition, system SSPs participate in system initialization and verification and are used as event builders during event acquisition. Each system SSP is attached, via a FASTBUS cable segment, to some number of remote SSPs.

At system initialization and verification time, the system SSP is used as a convenient memory buffer with which to test the VAX-FASTBUS interface, crate segment operation, and to some extent the SSP itself. (A set of integer diagnostics verify internal SSP operation.) Software is then loaded into the system SSP which maps the FASTBUS occupancy of the attached cable segment. It then proceeds with a series of data transfers to each cable device, usually SSPs, verifying the integrity of the data path, cable segment operation, and memories of the remote modules.

During all other operations, the system SSPs all execute a general purpose service program called SYSCAN. The SYSCAN program contains two main algorithms: a command list processor and an interrupt handler. Commands serviced by the command list processor include requests for FASTBUS I/O operations and system SSP memory allocation and deallocation requests. The interrupt handling algorithm checks for interrupt messages from remote SSPs. If an interrupt message is found, the interrupt handler can perform any or all of the following tasks: read remote SSP memory; pack together data read from other remote SSPs; and, send an interrupt message to the VAX.

To illustrate a typical transaction, consider a request to load a COMMON block into a remote SSP's data memory. A single call is made to the routine SSP_WR_COMMON on the VAX. This routine causes a command list and data buffer be sent to the appropriate system SSP. The system SSP is then started by a random write to CSR0. The command list causes the system SSP to perform the requested block write to the appropriate remote SSP. A FASTBUS interrupt message is then sent from the system SSP to the VAX indicating completion of this request.

The SYSCAN program is used during event acquisition, calibration, and all other applications requiring communication with remote SSPs.

4. REMOTE SSPs

Remote SSPs reside, one each, in FASTBUS crates filled with data acquisition modules, either TDC or FADC modules which provide timing and pulse height information from the central drift chamber.

Once the internal operation of a remote SSP has been verified via the attached cable segment, it then completes cable, crate and module verification. This is accomplished by software which addresses each module on the cable and crate segments in turn and performs comprehensive memory tests. These tests are customized for the various types of modules in the system reflecting their unique memory features. All modules present in the system are tested. A package of interactive routines may be invoked at any time which allows additional testing of any part of the system.

The final phase of system initialization consists of data acquisition module initialization. This step includes downloading registers and tables necessary for proper operation during calibration and event acquisition. Verification and initialization occur once at the beginning of each data run or about every two to three hours, and requires about one minute.

Event acquisition is the primary task for remote SSPs. This program for TDC crates, for example, consists of six steps: direct FASTBUS readout of TDC modules; data sorting and translation; calculation of pulse widths; time offset correction; construction of the final data format; and, notifying the system SSP that processing is complete. The data translation step involves changing FASTBUS geographical addresses and channels into layer, cell, and wire numbers. Data reduction may also be applied, such as a minimum pulse width requirement. In addition, a simple error log is maintained for I/O and processing errors encountered. Event acquisition processing completes within 20 ms, even for large events.

Control of event acquisition is achieved through a series of VAX-CAMAC-SSP interactions. An event trigger (NIM pulse) is used to start each remote SSP via its front panel. Upon completion of event processing a short interrupt message is sent to the system SSP, followed by a write to CSR0 to start the system SSP. This action causes the system SSP to read the remote SSP's data buffer, and then re-enable it for the next event. A master system SSP builds the event with data from remote SSPs on its cable segment, and from other system SSPs on its crate segment. After all remote and system SSPs have responded, the master system SSP sends a FASTBUS interrupt message to the VAX. The VAX reads the event buffer from master system SSP memory with a single FASTBUS block read, and resets the trigger logic allowing the cycle to repeat.

Corrections for channel-to-channel variations in electronic response are applied by the remote SSPs during event acquisition. The remote SSPs also participate in the determination of these corrections by computing for each electronics channel the sum and sum of squares of the of the responses to a set of identical calibration pulses. As sets of pulses are applied to various portions of the electronics, the remote SSPs read out the data acquisition modules and compile statistics for each set of pulses. In addition, an error log is maintained which is used to identify faulty electronics channels. After all channels have been pulsed, the results are read from the remote SSP memories. Thus, remote SSPs perform parallel, local processing. The VAX host sees data only from a complete set of pulses to all channels. Complete sets of pulses of several different values enable the VAX to parameterize the response of each channel. Detailed descriptions of the drift chamber and calibration systems can be found in Refs. 6 and 7.

Calibration control is similar to that for event acquisition. Calibration signals and triggers are issued by special hardware under CAMAC control. Trigger pulses are fed to each remote SSP's front panel START. The system SSP is notified by each of the four remote SSPs as they finish. The system SSP then re-enables the remote SSPs and, after waiting for all to complete, issues a DONE signal through its front panel to a CAMAC module at which point the cycle repeats. At the end of each set of calibration pulses, remote SSP memory is read by the VAX for final processing. The entire TDC calibration program requires between 30 and 40 seconds to complete, is performed once about every eight hours and represents less than a 0.1% contribution to the total detector dead time. An analogous program is being developed for the FADC system.

5. 3081/E MANAGERS

On-line 3081/E processors will be used for event reconstruction, and analysis. Data from each event will be channeled to an available processor. After processing is complete, the results must be read from 3081/E memory and transferred to the VAX for display and logging. The SSPs managing the 3081/E cluster will run a modified system SSP program. One important difference between these programs is the added task of SR scanning. Since the 3081/E's are FASTBUS slaves[8] they must indicate their readiness for event readout via the service request (SR) line. Software for 3081/E management is currently being designed and expected to be running concurrent with delivery of the first 3081/E. The first 3081/E with a dual-ported FASTBUS slave interface is expected to be delivered by the end of 1985.

6. FASTBUS TEST BENCH

The FASTBUS test bench is based upon a MicroVAX I computer with dual DRV11-J/IORFI[9] interfaces serving two independent FASTBUS crates (see Fig. 1). Software running on the VAX 8600 need only be linked with one alternate library to run on the MicroVAX: all VAX-FASTBUS subroutine calls are nearly identical on the two machines. The MicroVAX is also able to field FASTBUS interrupt messages in the same manner as the VAX 8600. While the IORFI provides a relatively slow access to the FASTBUS system (arbitrarily slow to $\sim$ 1300 32-bit words/s during block transfers), the addition of an SSP pushes this rate into the five to ten million words/s range.

The SSP can provide, for example, a series of repetitive FASTBUS operations in a debugging environment using an oscilloscope to search for intermittant problems. Simple programs of this type may be hand-loaded into SSP memory using the SSP debugger program, SNOOPY. More sophistocated test software containing more complex operations and options is generally coded on the IBM, following the normal sequence of processing steps. The SNOOPY program is also the primary tool for debugging new SSP software, as it provides simple access to all SSP registers and memory locations, de-assembles hardware instructions, and recognizes symbolic variable names.

REFERENCES

1. "The SLAC Scanner Processor: A FASTBUS Module for Data Collection and Processing," H. Brafman, T. Glanzman, A.J. Lankford, J. Olsen, and L. Paffrath, IEEE Transactions on Nuclear Science (NS-32), No.1, February 1985, p. 336.
2. "Proposal for the Mark II at SLC," CalTech document CALT-68-1015, April 1983.
3. "SLAC Linear Collider Conceptual Design Report,", SLAC-Report-229, June 1980.
4. "Data Acquisition and FASTBUS for the Mark II Detector", A.J. Lankford, and T. Glanzman, IEEE Transactions on Nuclear Science (NS-31), No. 1, February 1984, p. 225.
5. "The 3081/E Processor and its On-line Use," P. Rankin, *et. al.*, IEEE Transaction on Nuclear Science (NS-32), No. 4, August 1985, p. 1321.
6. "The New Drift Chamber for the Mark II Detector at the SLAC Linear Collider," Patricia R. Burchat, *et. al.*, IEEE Transactions on Nuclear Science (NS-32), No. 1, February 1985, p. 600.
7. "The SLAC Mark II Upgrade Drift Chamber Front End Electronics," D. Briggs, *et. al.*, IEEE Transactions on Nuclear Science (NS-32), No. 1, February 1985, p. 653.
8. "A FASTBUS Interface for the 3081/E," L. Barker, *et. al.*, these proceedings.
9. "An I/O Register to FASTBUS Interface," C.A.Logg, and L. Paffrath, IEEE Transactions on Nuclear Science (NS-30), No. 1, February 1983, p. 228.