EXPERIENCE WITH THE CES STARBURST

G. Waters, G. Wait, D.A. Hutcheon
TRIUMF, Vancouver, B.C. Canada V6T 2A3

CM-P00067477

Summary

Applications of the CES 2180 auxiliary CAMAC crate controller as a data pre-processor are described. They include track analysis in a wire chamber polarimeter to produce a smart trigger, and NMR signal averaging for polarization measurements in deuterated targets. Methods of generating and down-line loading stand-alone FORTRAN programs for various host computers will also be described.

Introduction

The increased demands made on data acquisition systems at TRIUMF have made it necessary to adopt new data pre-processing methods at the CAMAC level. The use of specially designed hardware providing a "smart trigger" is expensive and of limited efficiency, while processing all events in the host computer is very slow. A CAMAC-based processor provides the precision and flexibility of digital analysis, while avoiding the huge time overhead associated with the transfer of largely useless data over a CAMAC branch. TRIUMF's 8085-based auxiliary crate controller (TRIMAC), while adequate for the control of beam lines, targets, and relatively slow closed loops, does not have the speed and memory capacity required.¹ The device chosen was the CES 2180: the so-called STARBURST.² This is a single width CAMAC auxiliary crate controller based upon the Digital Equipment corporation J11.³ This is a 16/32 bit 200 ns cycle time microprocessor with the performance of the PDP11/70. Because it executes the standard PDP11 instruction set plus floating point, it is eminently compatible with the majority of TRIUMF's data acquisition systems.

Operation

Apart from its speed perhaps the single greatest advantage of the STARBURST is its dual port memory. Fig. 1 shows its standard configuration. 128 kbytes of mixed RAM and EPROM are available. The J11 can directly address half of this using 16-bit addressing. The rest can be accessed with the memory management hardware. From the CAMAC dataway the STARBURST appears as nothing more complicated than 64 kwords of CAMAC memory. This makes possible the down-line loading of data acquisition software and parameters from a host computer plus the reading back of collected data. Note that the CAMAC address pointer (loadable using F17, A0) operates on word boundaries while the J11 operates on byte boundaries. Low memory is always configured for RAM while high memory (to the 16-bit address limit) is user selectable for RAM or EPROM. The STARBURST may be activated by a front panel signal, a CAMAC command or on power up. What action is performed is jumper selectable. For a PROM'D system start address 151000 is selected. Other options are a trap to location 24, consol ODT or a jump to the standard bootstrap address (17773000). The latter option is only useful if the front panel q-bus is connected to a disc system or other bootable device. As well as the standard PDP11 interrupt vectors the STARBURST has several other maskable interrupts. They include two front panel sources for event triggering, sum of LAMS from the crate controller and a software interrupt from the CAMAC dataway (F25, A1). These interrupts are enabled by setting the appropriate bit in the high byte of the interrupt status register. After processing the interrupt it is cleared by setting the corresponding bit in the low

byte of the interrupt status register. LAM's are treated somewhat differently. The LAM source is cleared using the standard CAMAC command and writing a "1" to bit 2 of the interrupt status register will cause the STARBURST to generate a LAM, a useful method of gaining the attention of a host computer. This assumes the host (or STARBURST) has enabled the LAM in the normal way.

Fig.1 Address map of the STARBURST

CAMAC Operation

The CAMAC IO and control registers are mapped into the IO page beginning at base address 17764000 (Fig. 1). Thus a CAMAC read or write is simple "MOV" instruction using any of the PDP11 addressing modes. Table I shows the times required for commonly used CAMAC operations. The fastest CAMAC operation are those employing general registers only as no memory fetch (other than the instruction itself) is involved. However, this in itself is not too useful as one would usually wish to read a block of data into memory for future manipulation. A common CAMAC operation would be a block transfer using autoincrement addressing in a do loop. The following code for example performs a read of say an 8 channel TDC in 33 μ s

```

MOV #8, R3          ; set up loop counter
MOV #DATA, R0       ; R0 holds data address
MOV #NIAO, R1       ; R1 holds CAMAC address
MOV 0, @#FREG       ; CAMAC Read function
READ: Mov (R1)+, (R0)+ ; read and store
SOB R3, READ        ; next channel if any more

```

Note that the next sequential subaddress is selected by

autoincrementing R1. Due to the pipe line architecture of the CAMAC IO hardware, write operations are a little faster than read operations. During a write cycle the instruction is placed in the pipe line at which time the J11 is free to execute the next instruction. Should this be another CAMAC request there is a danger of a CAMAC overrun error being generated. In practice this is not a usual occurrence since there will normally be at least one instruction between CAMAC call such as a loop control operation as in the above instruction. We have only been able to 'force' a CAMAC overrun with a direct register write (MOV R3, (R0)) immediately followed by a second. This instruction executes in 1.5 μ s which is the length of time required for the pipeline to empty. Any other sequence will not be fast enough to beat the pipeline. Should a 20 Mhz version of the J11 become available in a STARBURST this situation will change making it essential to pay close attention to CAMAC 'tuning'. We have successfully operated two STARBURST in a single crate both performing CAMAC cycles at the highest possible rate.

Table I. CAMAC Instruction Timing for a 15 Mhz J11

CAMAC ADDRESSED IN IMMEDIATE MODE			
MOV R3, @#N2AO	write immediate	1.6 μ s	
MOV (R3), @#N2AO	write indirect	2.2 μ s	
MOV (R3)+, @#N2AO	write autoincrement	2.0 μ s	
MOV @#N2AO, R3	read immediate	2.8 μ s	
MOV @#N2AO, (R3)	read indirect	2.8 μ s	
MOV @#N2AO, (R3)+	read autoincrement	2.8 μ s	
CAMAC ADDRESS HELD IN GENERAL REGISTER R0			
MOV R3, (R0)	write R3 to CAMAC	1.5 μ s	
MOV (R3), (R0)	write indirect	1.7 μ s	
MOV (R3)+, (R0)	write autoincrement	1.9 μ s	
MOV (R0), R3	read camac (data in R3)	1.7 μ s	
MOV (R0), (R3)	read indirect	2.5 μ s	
MOV (R0), (R3)+	read autoincrement	2.5 μ s	
MOV (R0)+, (R3)+	read autoincrement	2.5 μ s	

Programming with RT11

For applications demanding the very highest speed we have relied on assembler language programming. However, for more complex applications FORTRAN becomes essential. Under normal situations FORTRAN code relies heavily upon an operating system for such things as I/O and run time error handling. Since we are down-line loading to the STARBURST, it becomes necessary to produce stand alone code. For the RT11 user this presents no special difficulty. Within the RT 11 FORTRAN library (SYSLIB.OBJ) is a relocatable module referenced by the global symbol (\$SIMRT). This is a subset of the RT11 operating system containing serial I/O handlers, etc. Linking instructions for including SIMRT in your memory image file and its limitations are given in the RT11 FORTRAN users manual.

Stand alone RT11 was originally intended by DEC for loading from paper tape with a memory resident absolute loader. Consequently linking instructions specify output in absolute binary format (FILE.LDA). For down-line loading via CAMAC save image file format (.SAV) must be specified. Included in SIMRT are handlers for normal DEC interrupts and traps. We have modified the source code to include default handlers for all STARBURST traps and interrupts including the line time clock.

The RT11 FORTRAN IV compiler does not produce in-line code that uses the floating point instruction

set. Instead a call is made to a floating point subroutine in the FORTRAN library. The user can build the FORTRAN library to include the floating point instruction set. Even so the time overhead involved in the passing of parameters to the subroutine in addition to setting up the floating point registers can be excessive. In addition FORTRAN IV does not recognise integer * 4 variables an inconvenience for 24 bit CAMAC words.

Programming with RSX11M

FORTAN 77 is a more efficient compiler for STARBURST programming. Full advantage is taken of the floating point and extended instruction set in in-line code. However FORTRAN 77 is only available under the more complex RSX11M operating system making the problem of producing stand-alone code a little more difficult to solve. A stand-alone version of RSX11 known as RSX11S is available but is not included as part of the RSX11/M distribution kit. Our applications so far have not required the multi-tasking facilities of RSX11S as each STARBURST is dedicated to a single task.

We have produced stand-alone FORTRAN programs by avoiding codes that reference the operating system. This means avoiding all 'read' and 'write' routines and compiling with the 'no trace' switch. In practice this has not been a problem since the STARBURST uses the CAMAC dataway for most of its I/O. Where console I/O has been required mainly for program development and debugging we use our own none interrupt driven console IO. These include single character and string routines plus conversion routines for integer and octal I/O. FORTRAN 77 'MAIN' routines always include start-up and system initialization routines. These 'system calls' are avoided by writing all our FORTRAN codes as subroutines and linking with an assembler language 'MAIN'. A skeleton example is shown in Fig. 2. To this main may be added any useful assembler subroutines such as interrupt handlers or IO, plus undefined globals that the linker may list.

Typical applications require the STARBURST to read various CAMAC modules preprocess the data and store the results in a buffer for transfer to a host machine. Since it is essential that the host 'know' the absolute addresses of all buffers they must be included as shared common blocks at link time.

Linking is performed using the task building utility TKB. Normally the first block of the resulting task image file contains parameters used by RSX11M in a multi-tasking environment. For stand-alone configurations no such 'header block' is required therefore the /-HD (no header) switch must be employed. The following command line will work in most applications:

```
TKB>NAME/-HD,NAME/-SP=MAIN,NAME,LIBR/LB
TKB>/
TKB>OPTIONS:
TKB>STACK=0
TKB>/_
```

The underlined sequences are user typed, the rest are prompts from the task builder. 'NAME' is your compiled FORTRAN subroutine module, 'MAIN' is your assembler language main (Fig. 2) and LIBR is a user subroutine library. The options may be added or deleted subject to requirements.

We have developed a set of standard FORTRAN callable CAMAC subroutines, including CDREG and CFSA. The latter executes in 43 μ s. For dataless functions we use a none standard subroutine CGO. Also included in this STARBURST subroutine library are the FORTRAN callable console I/O routines mentioned above. For assem-

```

.TITLE MAIN

ZERO = .

. = ZERO + 24      ; power on trap, configuration 0
COLDST: .WORD START ; start-up routine
        .WORD 0     ; status register

. = ZERO + 350     ; vector external interrupt #1
        .WORD EX1   ; interrupt handler
        .WORD 200

. = ZERO + 1000

START: MOV #1000,SP ; load stack pointer
      JMP USER    ; go to user program

; interrupt handler for external interrupt 1

EX1: BISB #40,@#166004 ; clear interrupt flag
     BIC #20000,@#166004 ; disable any more
     save general registers
     USER ROUTINES
     restore general registers
     BIS #20000,@#166004 ; enable interrupt
     RTI

; define globals to keep the task builder happy

.PSECT $$AOTS
.GLOBAL $OTSVA
.GLOBAL .END START

```

Fig. 2 Assembler language 'MAIN' for stand-alone FORTRAN.

bl language applications we use a macro library that emulates the RT11 I/O macro calls such as .TTYOUT and .PRINT.

Down-line loading

For down-line loading from a host the STARBURST is regarded as a 64k word CAMAC memory. The first block of the memory image file contains information on the program length and load address. From this information the number of records to be down loaded is calculated and the address pointer loaded (F17, A0). Our down loaders are written in the most 'transportable' form of FORTRAN possible and are based upon information supplied in the STARBURST user manual with adaptations for RT11 memory image file format. Initially the host computer interfaced to our CAMAC system was a TRIMAC development system requiring a CP/M version of the down loader. We have also added the facility of down-line loading into high memory a STARBURST version of CATY and the debugging program ODT. Both of these programs relocate themselves into low memory and will therefore over-write any user programs in low memory. For debugging purposes it is therefore necessary to link a program at a base address greater than 11000 octal.

Prom'd Systems

Only high memory may be configured for eeproms (2764's). Since any program will run faster in RAM and interrupt vectors are in low memory the technique is to link for low memory and install a relocation program at 11000 octal. This is equivalent to 151000 octal when expressed as an offset from the base address of the highest eeprom Fig. 1. Problems have been encountered in trying to PROM stand-alone RT11 (SIMRT) since the initialization section tries to set system variables in PROM which results in a program crash. A solution is possible by paying careful attention to the ordering of program sections and study of the RT11 PROM users manual (which we do not possess). A custom 'PAL' has

enabled us to configure memory above the 16 bit address limit for eeproms and use the host computer to move code down to low memory. This is a useful technique for non-DEC host machines and also provides space for multiple program storage. Since the STARBURST can address itself via the CAMAC dataway high memory is directly accessible without using the memory management hardware within the J11.

Currently two applications of the STARBURST are in operation and several others in development.

Track Analysis in a Polarimeter

The focal plane polarimeter (FPP) of the medium resolution spectrometer (MRS) consists of four pairs of wire chambers and a thick carbon analyzer. Scatterings through less than about 5 degrees are not useful in determining proton polarization, but constitute about 95% of the events. The protons incident upon the polarimeter have a broad range of positions and directions, making it impossible to distinguish useful scatterings on the basis of position or angle only of an outgoing proton. The small fraction of 'useful' scatterings, coupled with a need for throughputs of several thousand events/sec, requires pre-processing before the CAMAC branch stage. A STARBURST resident in the crate of FPP CAMAC modules was chosen as the best way to handle such problems as; precise matching of gains of different TDC channels; changes in minimum useful scattering angles with proton energy; changes in the separation of wire chamber planes. Fig. 3 shows the arrangement of wire planes, position of a typical particle track and the rejection algorithms employed.

First measurements of FPP performance with a STARBURST preprocessor showed that about 1200 events/sec could be rejected in pre-processing; optimization of the rejection algorithms have increased this rate to 2000/sec. Changes in rejection criteria are readily made by altering constants which are down-loaded from the host computer at the start of a run. Some difficulty has been experienced when the dual-port memory is accessed by CAMAC at the same time as the STARBURST is processing subsequent triggers. The severity of the problem appeared to be somewhat dependent upon the type of crate controller used. New 'PALs' supplied by CES and a re-arrangement of the CAMAC request/grant 'daisy chain' have reduced this problem considerably.

Assembler language programming was used throughout as the algorithms for determining a good scatter are straight forward. Upon the detection of a good event the STARBURST (using a mail box) waits for the host to complete the collection of data from the previous event before refilling the output buffer and issuing a LAM.

Multiple buffering is expected to increase data rates considerably and does not present any serious synchronizing problems. The host does not have to 'remember' which STARBURST buffer to read as the STARBURST simply loads its own CAMAC address pointer before issuing the LAM.

NMR Measurement of Polarization

A polarized deuterated target is currently under development for experiments to study the spin dependence in (π^+ , D) scattering and absorption reactions. The target material consists of about 1 cc. of deuterated butanol which is contained in the coil of a tuned circuit mounted inside a He-3 dilution refrigerator.

In order to measure the deuteron polarization one must measure the area under the NMR peak. Additional

Dispersion correction:

$$T1 = T1 - T2' \quad \text{where } T2' = k * T2/1000$$

$$T2 = T1 + T2'$$

Repeat for all pairs (T1, T2) (T3, T4) (T5, T6)

compute X1, X2, X3, X4

$$X1 = 100 (T1 - T2' + \text{offset})/\text{gain}$$

$$X2 = 100 (T3 - T4' + \text{offset})/\text{gain}$$

$$X3 = 100 (T5 - T6' + \text{offset})/\text{gain}$$

$$X4 = 100 (T7 - T8' + \text{offset})/\text{gain}$$

Condition for straight track

$$\Delta > |(X3 - X4) - (X3 - X2)|$$

Acceptable Scatter

$$((X4 - X2) - (X2 - X1))^2 > R2$$

Fig. 3 Focal Plane Polarimeter

information can be extracted from knowledge of the detailed shape of the NMR curve. The area under the NMR curve for an enhanced polarization must be normalized to a thermal equilibrium (TE) signal which is obtained typically in magnetic field of 2.5 Tesla at a temperature of about 1 degree Kelvin. The TE polarization is determined by Boltzmann's Law:

$$N^+/N^- = (\Delta E / kT) \quad \begin{array}{l} K = \text{Boltzmann's Constant} \\ T = \text{Absolute Temperature} \\ \Delta E = \text{Transition Energy} \end{array}$$

The TE polarization is extremely small (about 0.05%) and is very difficult to extract from the background noise. The NMR tuned circuit is phase compensated to permit measurements of the real part of the signal and this also solves the dynamic range problems inherent in searching for a small signal in a large Q curve shaped background. The background is essentially flat with phase compensation.

A frequency synthesizer under the control of an 8-bit microprocessor (TRIMAC) provides the RF frequency sweep for the NMR tuned circuit. The NMR signal is recorded by a transient digitizer gated by the TRIMAC. A STARBURST reads and averages 256 K of data from the digitizer memory before signalling the host LSI/11. As soon as the STARBURST has read the transient digitizer it signals the TRIMAC to begin scanning while it averages the data. When this pre-processing is

completed a flag is set in the mailbox within the STARBURST memory which is used for inter-processor communications and the LSI/11 then reads data from the STARBURST memory via CAMAC. At this point a flag is reset by the LSI/11 to permit the STARBURST to read the next 256 K of data stored in the digitizer. This is equivalent to 1024 frequency scans at 256 frequency steps over the NMR absorption region. The LSI/11 subtracts background and processes the data to display polarization graphs on various peripheral devices. The analysis of an NMR signal is typically read in groups of 1024 scans although this can vary from 32 to 8192 scans. Typically 100 groups of 1024 scans are required to extract the weak TE signal from the background with a statistical accuracy of about 5%. The total time taken to scan, record and analyze 1024 scans is 17 seconds. Fig. 4 shows the effect of signal averaging on the raw TE signal which is just barely discernible on an oscilloscope display.

Sept 16/81 Deuteron TE 950 mK

Fig. 4 TE signal.

Acknowledgements

We gratefully acknowledge the generous service received from Creative Electronic Systems.

References

1. C. Glazier, TRIMAC Users Manual, TRIUMF A-13741
2. Creative Electronic Systems, CP 122, 1213
Petit-Lancy 1, Switzerland
3. DEC DCJ11 User's Guide, Oct. 1983