SLAC–PUB–6642
October 1994
(A)

Code for the New MultiBus Camac Driver Compatible with the Old MultiBus Camac Driver

TAMARA J. RICHARDSON

for

The National Consortium for

Graduate Degrees for Minorities In Engineering and Science, Inc

at

Stanford Linear Accelerator Center

Stanford University, Stanford CA 94309

and

Candidate for

Master of Science Degree in Electrical Engineering

at

Georgia Institute of Technology,

Atlanta, Georgia

Submitted for publication

*Work supported by Department of Energy contract DE–AC03–76SF00515.

ABSTRACT

Stanford Linear Accelerator Center (SLAC) uses a Computer Automated Measurement and Control (CAMAC) device for accelerator hardware control and data acquisition. A MultiBus CAMAC Driver (MBCD) module is used to transfer data to and from the multibus memory. A New MBCD is being designed to execute packet instructions and to transfer data faster than the existing Multibus CAMAC Driver. An ADSP-2101 microcomputer is one of the new components that will be used to accomplish this goal. The software for the New MBCD board was developed in two stages:~ the first stage insures compatibility with the existing version and the second stage implements the new features. This paper addresses the software developed for this New MBCD board

Introduction

The MBCD transfers data to and from the multibus memory connecting the 80×86 Single Board Computer and the CAMAC System. Computer Automated Measurement and Control operations are performed through the SLAC Linear Collider (SLC) standard serial highway using the SLC Serial Crate Controller (SCC). Transmission is carried out at a 5 Mbit/sec serial rate. Today, this board is operating at its maximum capacity. The programmable chips and parts mounted on it have become obsolete in the electronic industry.

The Stanford Linear Accelerator Center is now designing a New MBCD that will execute packet instructions eight times faster than the old one. It is designed to have four CAMAC Ports to which up to fifteen crates will be randomly assigned to maximize throughput. An ID Register containing revision level and serial number will be implemented, and two Start Input/Output (SIO) registers and Test DeVice (TDV) registers will be implemented. The first of the SIO and TDV registers are identical to the ones on the existing MBCD, and are discussed in detail in this paper. The second SIO and TDV registers will be high priority registers that will be able to suspend CAMAC operations on the first SIO and TDV registers. These two registers will be implemented in the future.

Five Xilinx Programmable Logic Devices and an ADSP-2101 microcomputer are new components on the board that accomplish the overall goal of the New MBCD Board:~ to process more information and to decrease the time for transmitting information. The New MBCD Board will replace all the old boards in the SLC System, and will be the standard for the B-Factory and other systems.

MultiBus Camac Driver Description

The MBCD contains two registers accessible through Multibus: the SIO and TDV registers. Computer Automated Measurement and Control operations are initiated by writing the multibus segment address of a CAMAC package to the SIO register. Once the SIO register has been loaded, the MBCD fetches the first six words (16 bits) of the CAMAC package. These six words are referred to as a CAMAC packet. A CAMAC package consists of an arbitrary number of CAMAC packets. The first word of the packet (Control Word Low or CTLWLO), defines the crate address (crate 1–15). Bits 12–15 contain the crate number where data is to be read from or written to. The second word (Control Word High, or CTLWHI) defines the CAMAC command. Bits 0–4 are checked to determine whether the CAMAC operation is a read, write, or control command. For the read command, data is read from a port and written to the multibus memory. For the write command, data is read from multibus memory and written to a port. For the control command, the packet status is written to the multibus memory. The packet status register contains information about the hardware errors, remaining word count, last crate and module address, and terminating status.The sum of the third and fourth words of the packet, DATa buffer address OFFset (DATOFF) an DATa buffer address SEGment (DATSEG), is the address where the data is located. The fifth word, MAXimum Word Count (WCMAX), is a 14–bit number representing the maximum number of words to be transferred for each individual packet. The last word is the Completion Interrupt Code (CIC) of which only bits 0-3 are used to specify an interrupt number (0–7) and whether or not to generate an interrupt at the end of a package. Figure 1 shows a schematic diagram of the MBCD’s interaction between an 80×86 microprocessor and the CAMAC Crate Controllers. Figure 2 shows the layout for the New MBCD where four CAMAC Ports are implemented to speed up the transaction between the microprocessor and CAMAC Crates.

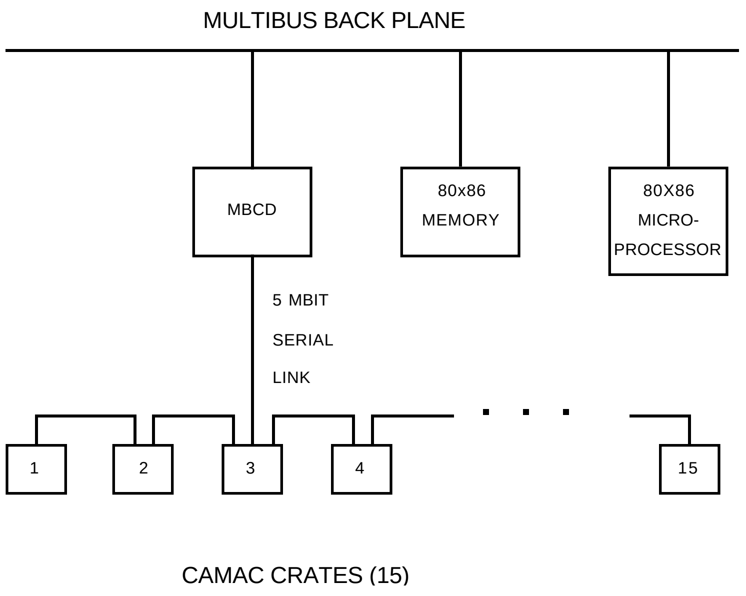

Figure 1. MBCD interface through which CAMAC operations are carried out by an 80x86 microprocessor.

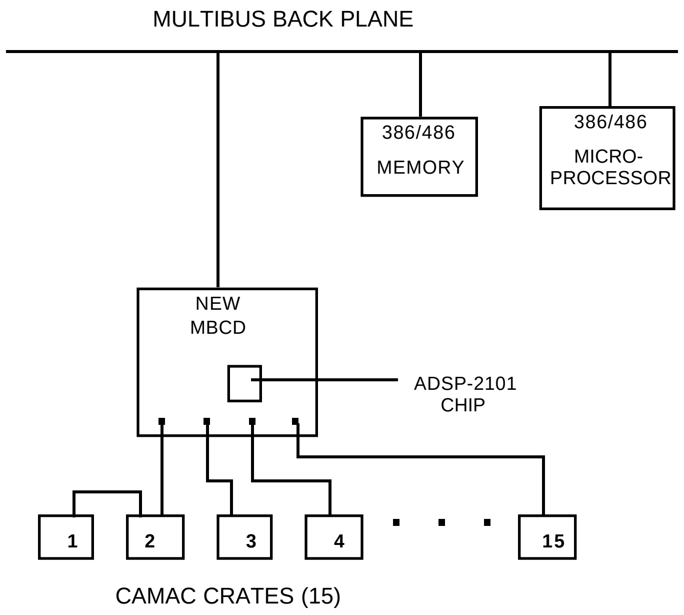

Figure 2. New Multibus CAMAC Driver Module.

ADSP-2101 PERFORMANCE

The ADSP–2101 is a single chip microcomputer optimized for Digital Signal Processing (DSP) and other high-speed numeric processing applications. It operates with an 80 ns instruction cycle time. In one cycle, the ADSP–2101 can generate the next program address, fetch the next instruction, perform one or two data moves, update one or two data address pointers, perform a computational operation, and receive and/or transmit data via the two serial ports. The processor contains three independent computational units: the Arithmetic/Logic Unit (ALU), the Multiplier/Accumulator (MAC), and the shifter. A powerful program sequencer and two dedicated data address generators ensure efficient use of these computational units. The program sequencer supports conditional jumps, subroutines, and returns from a subroutine or an interrupt in a single cycle; it executes looped code with zero overhead. Each data address generator provides addresses for simultaneous dual operand fetches and maintains and updates four address pointers. Hardware circular buffers are used to handle address pointer wraparound, simplifying the implementation of circular buffers both on– and off–chip.

NEW MBCD SOFTWARE

My contribution to the design of the New MBCD is to write code for the new module that is compatible with the old one. As shown in Figure 3, this program is divided into six subroutines:~ the initialization, multibus slave process, multibus master process, fetch packet, CAMAC Write (CAMACWR), and CAMAC Read (CAMACRD) subroutines.

The initialization subroutine enables the integer mode and secondary computational registers, disables Serial Port 0 and Serial Port 1, sets the data memory wait states to zero, and waits to receive an interrupt from a multibus input/output operation.

Once the multibus generates an interrupt, the multibus slave process subroutine first reads the slave control register. It then determines what type of input or output operation is being requested, and responds to it by either reading or writing to the read/write slave data register. For example, if the operation is a write to Port Map Register 0, the ADSP–2101 will read data from the read/write slave data register and write it to a multibus register. If it is a read, then data is read from a multibus register and written to the read/write slave data register.

After the multibus slave process receives an SIO write, the multibus master process fetches the first packet of the CAMAC package, loads the write address counter low with the packet address, and clears the write address counter high to initiate a multibus read. When the data is ready for the ADSP–2101 to read, the Flag In (FI) bit is set in the ADSP–2101. The ADSP-2101 reads the data from the multibus register and increments the address counter to read the next word.

After the multibus fetches the first packet, the fetch packet subroutine decodes the packet format to determine the crate address, CAMAC function code, maximum word count, and data location.

The second word of the packet defines the CAMAC operation:~ CAMAC Write, CAMAC Read, or Control Command. The CAMACWR subroutine checks the CAMAC status register to see if the write buffer is full. If the buffer is full, the processor continues to check the status register until it is not full. Whenever the buffer is not full, the ADSP–2101 reads the data from the multibus and writes it to the specified port; this routine does not complete until the maximum word count is zero.

The CAMACRD subroutine checks the same status register to see if the read buffer is empty. If the read buffer is empty, the processor continues to check the status register until it is not empty. Whenever the buffer is not empty, the ADSP–2101 reads the data from the port and writes data to the multibus.

Both subroutines also check the status register to see if the packet has ended. For a CAMAC Write Command, the processor continues to check the status register until a bit 1 indicates end of packet. For a CAMAC Read Command, if the packet is not finished, an error has occurred and the data buffer offset address overflowed (difficulties with this part of the subroutine have not yet been resolved).

The Control Command is part of the fetch packet subroutine; it writes the packet status to the multibus and checks to see if more packets are left in the CAMAC package. If there are more packets coming, the multibus master subroutine fetches the next packet, but if there are no more packets the program returns back to the initialization subroutine and waits for another interrupt from the multibus.

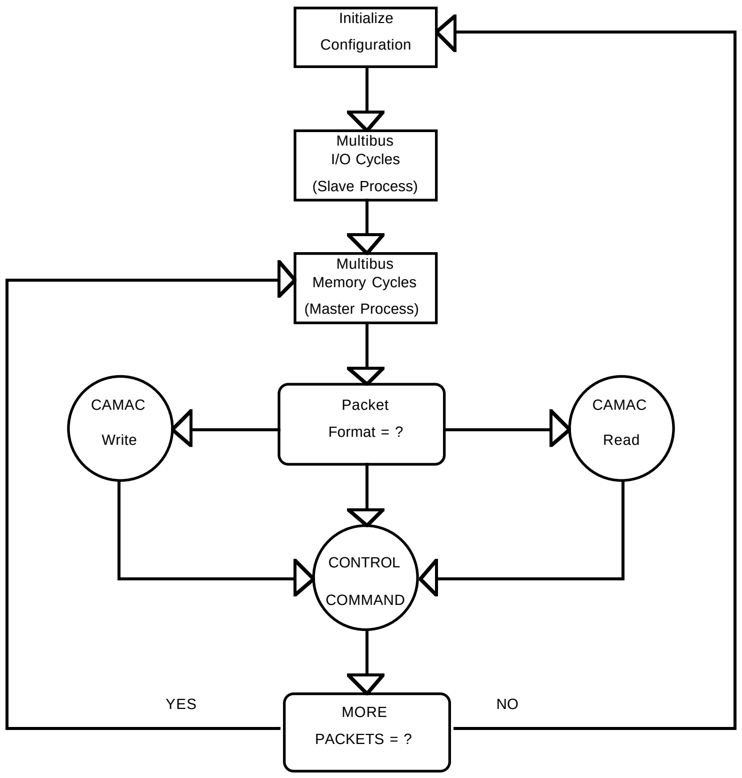

Figure 3. Flowchart of program

SIMULATIONS AND RESULTS

An ADSP–2101 Simulator was used to simulate and debug my code. The simulator provided the following functions: instruction-level simulation of program booting and execution, simulation of external interrupts, break points and single-step execution, simulation of various data transfers through I/O ports, and full view of all processor registers and memory with the ability to directly modify any register’s or memory location’s contents. Using this simulation tool, I was able to simulate I/O ports by using data files to represent data coming from and going to the multibus memory and CAMAC crates. Also, I inspected simulated memory locations and registers to verify that the program was executing correctly.

CONCLUSION

The software developed for the New MBCD is compatible with the Old MBCD.

After the hardware design for the new board is finished, the code will be installed in the ADSP–2101 programmable chip. In the near future, the code will be revised to implement new features to the module.