IEEE Transactions on Nuclear Science, Vol. NS-26, No. 3, June 1979

APPLICATIONS OF DISTRIBUTED MICROPROCESSORS IN THE CESR CONTROL SYSTEM

Donald B. Reaves[1], Frank W. Dain[2], and Ray Helmke[1]

Abstract

Intelligent microprocessor based interfaces are used to relieve the main control computers of time consuming tasks requiring real-time response. The basic system consists of a Zilog Z80 CPU, CESR control system interface, read-only and read-write memory, and input/output address decode circuitry. One application employs inexpensive potentiometers to provide a cost effective operator-machine interface with excellent response. It uses a 64 channel analog-to-digital converter to scan 30 two-gang potentiometers, calculating the change in position of each knob 60 times a second. A second application uses multiprogramming techniques to achieve separate position setting of ten motor-controlled devices with adaptive feedback. The controller can accept high-level commands to let the microprocessor guide the device to its destination, or low-level commands to let the main computer retain complete control of the device. Applications include RF phase-shifters and attenuators, and motor-driven variacs.

Introduction

The overall relationship between parts of the CESR control system is illustrated in Figure 1. The control computer (a PDP-11/34) communicates with the microcomputers via a data link called the X-Bus[3]. Like all other interface cards, the microcomputers are mounted in one of 16 crates which attach directly to the X-Bus.

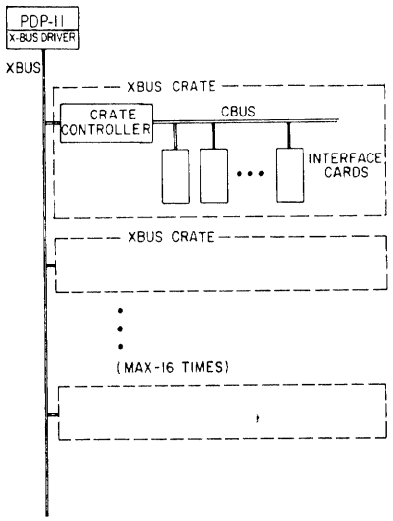

Figure 1

The common part of all applications which are based on microcomputers is called the microprocessor foundation logic. It consists of a simple microcomputer and the circuitry necessary to interface it to the bus in the crate (C-Bus). The computer comprises the central processing unit, or CPU, its program and data memory, the input/output devices, and the bus structure which connects them.

1. Laboratory of Nuclear Studies, Cornell University, Ithaca, New York 14853
2. Department of Geological Sciences, Kimball Hall, Cornell University, Ithaca, New York 14853

The CPU used in the microprocessor foundation logic is the Zilog Z80, a typical general purpose 8 bit microprocessing unit. It was chosen because it is compatible with Intel's 8080 processor and in addition offers many enhancements. An input called Nonmaskable Interrupt can be used to interrupt a program, cause execution of a special routine, and then return to the original program. It is used in the Motor Controller application to allow the CPU to measure time intervals; NMI is asserted at regular intervals and the interrupt routine increments a counter.

The C-Bus interface allows the control computer to perform direct memory access operations in the microcomputer's memory. Use of DMA can greatly simplify many communication problems. For instance, the CPU may keep an up-to-date table of information in its memory for use by the control computer. When the control computer requires the data, it can get them directly without having to wait for the CPU to perform transfer functions. The CPU need only acknowledge the bus request asserted by the C-Bus interface, an action which is performed automatically in a manner which is transparent to the program which the CPU is executing.

Input/output devices in this system are referenced in the same manner as memory. This reduces the complexity of the C-Bus interface and allows the Z80 to use data from input devices directly in instructions such as add or subtract. (For devices which are implemented as standard Z80 inputs, using input addresses instead of memory addresses, only load and store instructions are allowed.)

The microcomputers are programmed in assembly language using a commercial cross assembler which resides on the lab's main computer, a Decsystem-10. Object code is then transferred to a PDP-11 where it is either programmed into an EPROM or loaded directly into the microcomputer's memory via the X-Bus. The direct loading process is used during hardware and firmware testing to reduce the time required for changing programs.

The foundation logic, which required about 3 man months to implement, uses 50 TTL and MOS LSI chips and is mounted on a standard 9 by 12 inch C-Bus board. The Knob Scanner and Motor Controller applications require an additional 15 and 50 chips respectively.

Microprocessor Knob Scanner

The purpose of the knob scanning system is to provide the operator with computer inputs in the form of control panel knob positions. Because these data are to be used to effect changes in program variables, the information is most useful in incremental form. Ideally, the incremental change reported should be proportional to the amount of rotation of the knob. While this is not beyond the ability of commercially available hardware (e.g. shaft encoded knobs), it is not absolutely necessary. A smooth, monotonic response has proven to be sufficient.

Knob Hardware

Each knob consists of two 1K potentiometers on a single shaft which are used as voltage dividers to provide inputs to an analog to digital converter. The two pots are mounted 180 degrees out of phase, and have had their stops removed to allow unlimited rotation. Since the approximately linear region of each pot extends over more than half a revolution, at least one

0018-9499/79/0600-3445$00.75 © 1979 IEEE

of the pots is always in this "linear" region. The voltage curves for the two pots in a sample knob, plotted as a function of angle, are shown in Figure 2.

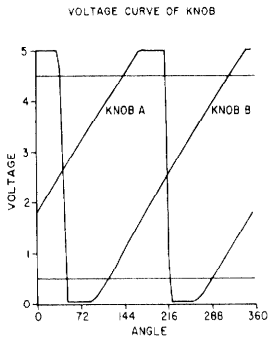

Figure 2

Analog to Digital Interface

The Microprocessor Knob Scanner consists of an analog input interface built upon the Microprocessor Foundation Logic. This interface is made up of three basic parts: an input multiplexer, a 12 bit analog to digital converter, and control registers necessary for the CPU to manipulate the interface. The time required for a conversion is 25 microseconds, so that the maximum rate is 40,000 conversions per second.

Knob Scanning Algorithm

The knob scanning algorithm is an infinite loop which first determines if the control computer has requested a particular action, such as halting or clearing the incremental data. It then updates the incremental data for each knob.

The increment of a knob is determined as follows. By considering Figure 2, one can see that if both voltages are within the bounds indicated by the horizontal lines, the one with the smaller slope represents the linear region for the associated pot. Thus, if both pots are now and were on the last pass within these bounds, the increment is chosen to be the smaller of the changes in the pots. If only one pot is within these bounds, the increment is chosen to be the change in that pot. Otherwise, the change in pot B is arbitrarly chosen.

Communication between the control computer and the Knob Scanner is accomplished by reserving arrays of information in the microcomputer memory. In addition to the raw analog values read from the pots and the incremental changes in the knobs, a Control/Status byte is available which indicates fluctuations detected in the power supply and is used by the control computer to initiate certain actions.

Performance

Knob Scanners have been used in the control system since November 1977, and have proven to meet their design requirements reliably.

When the calculated position of a knob undergoing constant rotation is plotted as a function of time, the response is shown to be quite linear, as long as the knob is turned at a reasonable rate. The only characteristic of the knobs in which the small nonlinearity is noticeable is that if a knob is returned to its original position after some use, the associated variable does not exactly resume its original value. This phenomenon generally does not cause any problems, since the operator observes the variable on an adjacent numeric display.

In an earlier implementation in which the control computer itself scanned the knobs 20 times a second, increments were often calculated incorrectly because of the low sample rate. In the Knob Scanner, the time interval between samples of a given knob varies with the amount of calculation being performed by the CPU. This time varies from 1/60 of a second, when no knobs are being turned, to 1/40 of a second, when all 30 knobs are being turned. This represents a significant improvement over the earlier implementation.

In order to achieve the same performance using a PDP-11/34 approximately 40% of the available processing time is required, as measured by a program written by S. Peck[4]. In a system where control panel servicing requires 50% of the processor, using the control computer to perform the Knob Scanning algorithm would leave virtually no time for application programs. If the equivalent of two Microprocessor Knob Scanners (i.e. 60 knobs) were required, the control computer would simply be unable to read each knob often enough to achieve acceptable results.

Microprocessor Motor Controller

Several machine variables in the lab are remotely controlled via motors with simple directional controls. These include RF phase shifters and attenuators, and motor driven variacs. Unlike other machine variables, there is no straightforward way to set these variables to predetermined values. Instead, only simple increase or decrease commands are easy to implement.

The purpose of the Motor Controller is to transform the desired destinations into directional commands for each of ten motors. In addition, the Motor Controller allows the operator to issue directional commands in order to fine-tune the associated machine variables.

Motor Hardware

To indicate the precise location of the device, an analog signal is sent to the controller. The operator specifies the value to be assigned to the associated machine variable in terms of the digitized values of this signal. The device being moved by the motor can travel only a limited distance in either direction. To prevent damage to the device, two limit switches turn the motor off when the end of travel is reached. A single 24 volt signal is sent to the Motor Controller when either of the switches is actuated. Control of the motor is accomplished via two 24 volt signals. Each turns the motor on in a particular direction. If neither is asserted, the motor is off.

Interface Hardware

The Motor Controller consists of three interfaces built upon the Microprocessor Foundation Logic. Each interface has ten channels, so that ten motor controlled devices can be serviced. The analog input interface is similar to that used in the Knob Scanner. A high level digital input interface uses opto-isolators to isolate and translate the limit switch input signals to TTL levels. These inputs are fed into latches which can be read by the CPU or C-Bus interface. A high level digital output interface performs a similar function for output.

An interrupt circuit, which supplies a time base for scheduling, is a simple oscillator that provides a short pulse to the NMI input of the Z80 every 5 ms.

In order to prevent a failure on the part of the Motor Controller from issuing invalid commands to its output devices, a failsafe output enable circuit has been implemented. It consists of two one-shots, triggered by the CPU, which are used to enable the tri-state outputs of the latches containing the motor direction data. If the latches are not enabled, all motor direction outputs are turned off. To keep the outputs enabled, the enable one-shot must be retriggered at time intervals no greater than 20 ms. The trigger one-shot prevents the enable one-shot from being triggered during a failure by allowing it to be triggerred for only a very short time. When adjusted properly, there are only six possible combinations of two instructions which can trigger the enable one-shot, virtually eliminating the possibility of enabling the hardware during a failure.

Motor Controller Algorithm

The Motor Controller algorithm consists of three basic parts: the TECO (Timed Electrical Contact Output) algorithm, the interrupt handler, and the reset algorithm.

The TECO algorithm allows the operator to issue directional commands for fine-tuning machine variables by accepting an instruction specifying a direction and a length of time for which a motor is to be run. This is implemented by keeping an array, TICKS, which contains for each motor the number of time units until the motor is to be turned off. The main program loop examines these values and, using information stored in a direction array, turns the motors on or off accordingly.

The interrupt handler, which is executed every 5 ms, decrements any nonzero entries in the TICKS array. It also increments a variable which is used by the reset algorithm to determine time intervals.

The reset algorithm operates by first starting the motor in the required direction, and then turning it off at the proper time so that the device will coast to a stop at the requested destination. This process is repeated until the position of the device is within a small value of the destination.

The Motor Controller learns the characteristics of the devices it controls, so that after some practice it will be able to perform a reset in a single try. This is accomplished by keeping a table of estimated values to determine how far each motor will coast when turned off. The possible destinations are partitioned into segments. For each of these segments, the table contains the expected amount the motor will coast when turned off from either direction. After each successful reset operation, the appropriate entry in the table is set to the observed coast value.

The TECO and reset algorithms can be combined into a single program by enclosing them in a loop which determines at the beginning of each interation which algorithm to execute. The resulting program will control only a single device, and therefore needs modification to control multiple devices. By a strightforward procedure, the program can be generalized to service any number of devices in a simple multitask environment[5]. In the present implementation, each device is serviced about 100 times a second. This seems to be more than adequate.

The Motor Controller provides the control computer with information about the general status of the controller and the status of each device. This includes an indication of which algorithm (TECO or reset) is being executed, the location of each device, whether the device is stuck or is at an extreme of travel, and how many trys were necessary to perform the last reset operation.

Performance

At the present time, a microcomputer has been controlling the RF phase shifters for only a few months. Fairly thorough testing has proven it quite capable of meeting its design specifications. However, the real test of the reset algorithm will be made when the Linac and Synchrotron are routinely switched between e^+ and e^- settings.

Error Detection and Reliability

Methods used for detecting and correcting errors were intended to cover all cases which could be handled in the firmware. These include hardware errors external to the microcomputer such as knob power supply fluctuations or a motor controlled device being stuck, and operator errors such as attempting to set a motor controlled device past its extreme. Algorithm errors, such as being unable to move a device to a particular position within a reasonable number of tries, are also reported.

Errors in the microcomputer itself are handled either by standard X-Bus mechanisms or by a handshaking mechanism involving the control computer. All microprocessor based applications include a Control/Status byte. One of its functions is to allow the control computer to determine whether the microcomputer is executing its program. At regular intervals, the control computer clears one bit of this byte. During normal operation, the microcomputer sets the same bit. If the control computer detects this bit in the zero state after an appropriate delay, it is assumed that the microcomputer has failed, and appropriate action is taken.

Errors which are fatal to the microcomputer's program integrity are prevented from affecting output devices in the Motor Controller by use of the failsafe output enable circuit.

These measures seem to be adequate for the problems which have arisen to date. It is not presently known how the large scale integration components used in the microcomputer, the CPU and memory, will stand up to the environment in which they must operate. Data concerning radiation effects on these components are not available. It will be interesting to see if the error mechanisms listed above will be adequate to handle any problems which arise.

Cost Effectiveness

It might be argued that, including total development time, the cost of these two projects exceeds the cost which would be incurred by using alternate solutions. It is certainly true that using different hardware (shaft encoded knobs instead of potentiometers, and stepping motors instead of DC motors) would have simplified the algorithms and enhanced the results of these projects. However, the development of a simple yet flexible microcomputer interface will continue to pay for itself as other applications arise.

3. R. Helmke, S. Ball, and D. Rice, Interface Hardware for the CESR Control System, E-14 this conference.
4. Personal communication.
5. D.B. Reaves, Applications of Distributed Microprocessors in the CESR Control System, Master's thesis, Cornell University, 1979.