SLAC-PUB-1897
March 1977

APPLICATIONS OF MICROPROCESSORS IN UPGRADING OF ACCELERATOR CONTROLS*

K. B. Mallory
Stanford Linear Accelerator Center
Stanford, California

Abstract

Experience at SLAC demonstrates that the criteria for selection and use of microprocessors in modifying an existing control system may differ from the criteria that apply during installation of the control system of a new accelerator. Considerations such as cost of individual projects, progressive installation without disruption of operations and training of on-board personnel can outweigh "obvious" goals such as standardization of hardware, uniformity of software, or even a rigid specification of link protocols with the main computer system.

What is a Microprocessor?

Microprocessors are developing so rapidly that it would be rash to predict what their properties and capabilities might be in a few years. But at the moment they typically consist of one to three chip-types that contain all of the addressing, instruction-decoding and data-processing logic that comprise the central processing unit (CPU) of a computer. In the simplest configuration, they can be used as inexpensive special-purpose controllers by using limited amounts of read-only memory (ROM) for program storage and of read-write random-access memory (RAM) for data, scratch work and variable control parameters. At the most complex end of the scale, a group at SLAC has developed a type 2901 bit-slice microprocessor, with additional pipeline hardware and fast memory, into a unit that can process data as fast as our largest IBM computer.[1] In between, there are models of standard minicomputers available that boast of using a microprocessor chip as CPU.

The microprocessor chips can thus be used in configurations ranging from fixed-purpose controller through minicomputer to a unit that competes in some respects with a maxi-computer. For this paper, I wish to limit the definition of a microprocessor.

In a real-time control system, a computer either must execute a fixed control program or must have provisions for loading or overlaying programs dynamically. The latter type, which must have core or other read-write memory for storing various programs, I call a minicomputer no matter what type of CPU it may have. It is the fixed-purpose device, whose program could be stored in ROM, that I shall discuss here.

Microprocessors are showing up in control systems in a number of applications. Typical applications include (1) graphic display controllers and other operator interfaces to the computer system, (2) general purpose data multiplexers and controllers for computer system interface, (3) link message switchers and buffers with provision for translating external link protocols to standard internal system protocols, (4) autonomous controllers that have no direct computer system interface, and (5) any number of microprocessors embedded in instruments bought from others. The last will generally have a link designed by the manufacturer, which may need translation as in (3) above.

They are also found as CAMAC crate controllers and as elements within a CAMAC crate, where they are used for some combination of the applications described above.

The control system designer will normally have direct control only of the link message processors and of the microprocessors that interface directly to the signal sources. The latter processors will be used primarily at the lowest level of the system as the interface between a minicomputer and the equipment that is to be controlled and monitored (power supplies, vacuum system components, signal multiplexers, etc.). One should beware of applications that are expected to require frequent program modifications, that require growth or that use expensive amounts of memory. A minicomputer may be more reliable or more economical than the microprocessor configuration for these applications.

Standardization in a New System

In the design of a large, new control system, it is common to establish a definite set of design standards, to prescribe certain circuit design and fabrication techniques and to recommend lists of components and even subsystems to be used in design of equipment. Such a procedure can lead to a system with minimum duplication of design effort, with maximum utilization of similar components, and with a uniformity that will make it easy to maintain. These design criteria should be maintained as long as the standards and technology involved remain reasonably up-to-date. Generally this will last at least three to five years into the useful life of the control system.

Standardization is especially convenient within a computer control system. If all computers within the system use the same (or at least subsets of the same) instruction set, then (1) code that has been tested in one processor can immediately be used in another; (2) the same executive program may be used in all, so it need be debugged only once; (3) applications programmers need learn only one environment for writing programs, whether in the central or in a peripheral processor; (4) linking and routing protocol can be identical at all levels of the system; and (5) program and data-file management can be handled in the same fashion at all levels of the system.

The microprocessor will not have an executive program, since it will be executing a fixed permanent program whose requirements are determined by the equipment engineer. I suggest that the logical boundary between the "computer control system" and the equipment is not at the wiring connections to the microprocessor but is within the microprocessor's program, between the link-handler and the specific control code for the equipment.[2] To the minicomputer, the microprocessor should appear to be just like any other computer. The same link protocol should be used for messages (tasks) and for data blocks (files) as is used in the rest of the control system. This allows the routing of a task to the microprocessor to be handled in the same fashion as the routing of a task to any computer in the system. The road is also left for later replacement of the microprocessor by a minicomputer if the job becomes too big or requires variable programs. None of the higher-level code would need to be changed when the replacement is made.

Relaxation of Standards

I do not practice what I have just been preaching. In particular, of course, we did not design our entire computer-control system at once. Computer control at SLAC started fifteen years ago with a single processor,

* Work supported by the Energy Research and Development Administration

(Presented at the 1977 Particle Accelerator Conference, Chicago, Illinois, March 16 - 18, 1977)

the SDS-925, devoted to the task of magnet control in the beam switchyard. It was installed in what is now the Main Control Center (MCC). Six years later, we procured a second processor, a PDP-9, which was installed in the former Central Control Room for the accelerator (CCR); two more years elapsed before the two were linked together to allow moving all operations to MCC. It was not until this time that a "computer system" was started. We now have twelve computers on-line, plus two spares, of four different types -- not merely with different instruction sets but with different word sizes.

We have introduced a standardization in that the executive program provides essentially the same services to (and imposes the same restrictions on) the applications programmer in all of the computers. But minor differences exist because of differing instruction sets and word sizes. Link protocols are similar throughout the system, but differences again exist because of differing word sizes.

Even in a system that started out consistently, however, it is not clear that rigid adherance to an old standard should be continued after a few years of system life. The arguments against change are that engineers and technicians may have to be retrained, that maintenance must become much more complicated as more technologies become mixed in the system, and, perhaps, that the system has remained clean and uniform to date. The last argument, if valid, is a powerful one which I would not dispute.

The arguments for change are that new people would otherwise have to be trained in obsolete methods, that replacement parts will become difficult to obtain, that more-efficient devices will become available, and that designers will become more and more convinced that they cannot do a good job without using newer techniques.

One must use the engineers one has; they have continued their own education, and it is to the manager's advantage to use their current knowledge. Unless only one engineer is allowed to work on microprocessors, they are bound to want to work on different types. Each engineer has his own head-start due to his specialized knowledge. Allowing him to choose his own direction can save time and encourages him to do his best.

A specialized project may be much easier to implement with one processor type than with another; expediency or cost may thus dictate the use of a particular processor type.

Although uniformity of equipment makes for easier maintenance, differences between microprocessor controllers will be no more traumatic in the long run than differences between separate hardware controllers. Uniformity of software is also a fine goal but the coding for simple controllers is not difficult in any of the standard processors.

Applications of Microprocessors

Microprocessor controllers may be installed with new equipment for which no controllers exist. They also may be used to replace existing controllers. In this case they must provide new features such as local closed-loop control, control of modified equipement, new or enlarged local parameter memory, or digital save-restore of sets of values. The fact that the old controller is difficult to maintain is not in itself a sufficient justification for replacement.

We have found that the microprocessor projects currently under development or consideration fall roughly into four types: (a) A large program is required to handle closed-loop controls or a large and varied set of I/O interfaces. (b) The processor emulates, with a few improvements, a hardwired controller. (c) The processor provides logical or computational features not readily available by other means. (d) The project is not yet defined well enough to classify. Of the eighteen projects now in queue, five are of type (a), nine are of type (b), two are of type (c), and two are of type (d).

A brief description of our first four microprocessor projects in development and of two that have been under consideration will illustrate these types.

Type (a): Large Control Program

Typically, the equipment is being replaced; a new controller is required. Often all control and monitor functions are changed. Considerable local logic is added; the controller may even monitor its own performance. The controller depends heavily on a link to the computer system. This type will have a complicated control program. Since everything is new, there is a risk that this type of controller will run into problems with indefinite requirements and continuing development.

Phasing Programmer.[3] The phasing system is being modernized, with stepping motors replacing DC motors for the phase shifters, new phase detection electronics and a new controller based on the Intel 8080 microprocessor. A closed-loop control program drives the motors for eight klystrons to reduce phase errors to zero. It provides data filtering, sequences the procedure, makes allowances for differing gains of the measurements and outputs raw driving pulses to the appropriate motors. A link to a PDP-8 will provide a number of new control and monitoring features.

An Intel 8080 system was selected because the 8080 was well established and has some 16-bit arithmetic capability, and a good cross-assembler and simulator were available in the Triplex.[4]

Type (b): Enhancement/Replacement of Hardware Controller

The controller provides new features such as direct computer link, increased local memory for parameters, new monitoring and control functions, lower cost when installing additional copies of same equipment, etc. The equipment may remain unchanged or may be new equipment designed to be compatible with the same controller. The old control circuits generally remain operational. Typically, a small program selects values from a table, with minimal logical decisions, and sends sets of values to the equipment on a pulse-to-pulse basis.

Beam Guidance Controller.[5] When additional pulsed steering and pulsed quadrupoles were installed along the accelerator during the past year, we found it cheaper to build microprocessor controllers than to duplicate the hardware controllers used on the older pulsed beam-guidance supplies. The Motorola 6800 microprocessor was selected because of the availability of an evaluation kit, on which our present controllers are based, as well as its modest advantage over the 8080 in speed and program size. It took the author of SLAC's 8080 cross-assembler but one week to write a cross-assembler for the 6800. The controllers provide enough new features that we plan to phase out the old controllers to provide greater multiple-beam capacity and to provide direct digital save-restore of sets of values for beam setup.

The controller reads patterns (gating signals which define which experimental beam will be accelerated on the next pulse), selects appropriate values from a table, and writes them on an output register. It reads bits from the local command decoder and counts up or down on a particular selected value in the table. It also has a link to a PDP-8 with software to patch any value in the table ("set") or to report any value ("read") and to read or restore the entire table as a block of data.

In this processor, the register outputs are connected to DAC's which drive the horizontal, vertical and quadrupole pulsed power supplies in one sector.

Trigger controller.[5] This is virtually the same controller, with a similar program used for setting a DAC output to the pulsed beam loading delay and with additional logic for folding pattern, rate, modulator interlock and phasing request data into output registers for standby and accelerate pulses for eight klystrons.

An option has been provided for generating the "rate" signal in the microprocessor, so that it can be set directly by link command from MCC instead of from the rate generator in CCR.

This signal defines the basic pulse repetition rate for the klystron modulators and determines the power consumption of the accelerator. No other provision for MCC control of the rate is now planned.

Type (c): Special Features

Beam Monitor Processor. We have analog signals from 36 beam position monitors which have been multiplexed, at eight microsecond intervals, into a single pulse-train for oscilloscope display. Each position monitor has a zero-error which is, in general, a function of beam current and therefore is different for each beam. A DC zero-set control at each monitor, which affects all beams, allows correction of the signals for one beam at any time. The job of the processor is to store a table for each beam of zero-error values for each monitor. Then it is to correct all beam position signals "on the fly". After it has selected the correct table, it must execute a fast sequence of operations: (a) digitize a signal from one beam position monitor, (b) subtract the error for that monitor, (c) correct for possible arithmetic overflow, and (d) convert the result back to analog. This must be repeated every eight microseconds.

Of all the processors we have used in the control system, the PDP-8 is the fastest, but it still would require eleven microseconds to process each signal internally. We investigated the 2901 bit-slice device mentioned above: It can do the calculations in four microseconds, but it has been implemented for computation, not for real-time control. Several months would be required to develop a model with the necessary I/O and synchronization interfaces.

We finally decided to use a 6800 in a hybrid configuration. Only a small part of the processing is done in the microprocessor; the rest is done in an external arithmetic logic unit (ALU) with additional hardware for detecting and correcting overflow conditions. Except for mode-select commands, the processor has no connection with the rest of the computer system.

In the normal mode of operation, the microprocessor reads a pattern, checks the sector 27 toroid to determine if there was any beam, and then selects successive error values from a table and puts them in an output register. The ALU computes the difference between an ADC input signal from the beam monitor multiplexer and the error and delivers the result to an output DAC which sends a corrected signal to MCC. Additional logic circuits detect arithmetic overflow and substitute the appropriate zero or full-scale value for the ALU output to the DAC.

In zeroing mode, the microprocessor checks pattern and sector 27, and then reads successive error values from the ADC and stores them in the appropriate correction table. With no output from the microprocessor, the ALU and output DAC send the zero errors on to MCC unmodified, so that the operator can operate the DC zero-set controls if he desires.

Although we used our familiar 6800 for this particular processor, I expect we will find applications for bit-slice devices once we understand better how to implement them.

Type (d): The As-Yet Undefined Controller

Beam Interlock System. We now have a hardwired interlock chassis for the beam switchyard which combines status signals into a permissive signal for the beam. A different combination is used for each beam, depending on the destination target area and the selected mode of operation of that area. It is very difficult to modify the logic when equipment or mode-select requirements change.

A microprocessor controller to handle these interlocks could be designed so that the program and the signal connections must be changed each time the interlock requirements or equipment configuration is changed. It could be designed so that all the possible signals are connected to the processor and only the program is changed. Or it could be designed so that all the signals are connected, and the program executes from a variable list of the current interlock requirements. In the last case, only the list need be changed when a different interlock configuration is required. As operational requirements change, any of various lists may be loaded from a file in the computer system.

The first case would be no improvement over the present hardwired controller. The second leads to an unbounded job that should not be implemented in a microprocessor. The third represents a well-bounded job that should not require significant hardware or software modification for years.

Data-Disk Buffer. The PDP-11 that was put on-line this winter is currently serving as a fast buffer between the 925 and the Data-Disk display unit. This job could have been handled by a microprocessor controller; but as part of our program to improve system reliability by moving the 925 off-line, the display-generating code will be moved from the 925 into the PDP-11. The job will then require a minicomputer configuration.

Conclusions

A microprocessor controller should be considered a dedicated device. It has the advantage that its programming can be worked on easily during debugging and that if major changes must be made later, they can be made primarily in the software. But building a microprocessor for the purpose of changing its structure frequently is not making best use of its properties.

It is "obvious" that microprocessors have their place in a control system for any of a large number of well-specified interface applications. The advantage of the microprocessor controller is realized when enough options are built into the controller before it is put into service. Changes within this range of options can be implemented easily at any later time. The use of a standard configuration whenever feasible will save time and effort when building a new controller, but should not preclude use of other microprocessors when special features are required.

One may distinguish between a discrete change reserved for future development and changes that have no obvious immediate tying-off point. Minicomputers may be more suitable than microprocessors in interface situations where the requirements are indefinite or are likely to be significantly changed within a year of design, or in a situation in which the engineer and programmer rely on the flexibility of the processor to take care of all unforeseen requirements. It is probable that a microprocessor system that is built with continuous development in mind will become a maintenance nightmare, it will be operationally out of service with great frequency, and it will remain a drain of designer's and programmer's time for years. This design problem is, of course, not new with microprocessors. The fact that the microprocessor's I/O devices and program can be varied ad nauseam is not sufficient to make a good controller; it can not do the job correctly until it has been told exactly what to do.

The discussion of our controller projects shows that the suitability of a job for a microprocessor controller depends not so much on the job itself as on the way in which the job is defined. There still is no substitute for good engineering.

References

1. "The LASS Hardware Processor", Paul F. Kunz; Nuclear Instruments and Methods 135 (1976) 435-440
2. "A New Approach to Control System Interfaces", K. B. Mallory, Isabelle Summer Study, Vol. II (1975) p. 712
3. "A Microprocessor Controller for Phasing the Accelerator", S. K. Howry and A. Wilmunder; Paper K-23 of this Conference
4. "Microcomputer Assemblers for the Intel 8080 and Motorola 6800", L. J. Shustek; SLAC CGTM 174 (May 1976)
5. "Two Microcomputer Controller Applications at SLAC", W. C. Struven and K. B. Mallory; Paper L-22 of this Conference