SLAC-PUB-1918
March 1977

A MICROPROCESSOR CONTROLLER FOR PHASING THE ACCELERATOR*

S. K. Howry and A. R. Wilmunder
Stanford Linear Accelerator Center
Stanford, California

Abstract

A microprocessor controller is being developed to perform automatic phasing of the SLAC accelerator. It will replace the existing relay/analog boxes which are ten years old. The new system is all solid state except for the stepping motors that drive the phase shifters. The paper describes the components of the system, the control algorithm, microprocessor hardware and software design and development, and interaction with SLAC's computer control system.

Introduction

A microprocessor controller is being developed to perform automatic phasing of the SLAC accelerator. The design objective is to reduce experimental time lost during phasing of the 240 klystrons and also to eliminate gradual degradation of beam quality between rephasings. These objectives are best met by continuously rephasing, stealing a pulse or two every ten seconds rather than the 60 pulses per second required by the old analog electronics and relay logic. By the end of the year, a controller will be installed in each of the 30 sectors at which time it will be possible to phase each of the klystrons once every three hours.

The controllers have high-speed electronics that measure the phase difference between a reference and either a beam-induced RF signal or any of eight klystron output signals. All signal frequencies are 2856 MHz and the measurement sampling time is .2 μs. Stepping motors rotate phase shifters connected to reference and klystron signals. The microprocessor sequences these resources to make the phase of all klystron signals equal to that of the beam.

The system was partitioned into Power Supplies, CPU, High-Speed Interface and Low-Speed (or Power) Interface. NIM bins and modules were chosen because of their flexibility of width and integral shielding. The specialized nature of the system allowed a bus structure with full decoding within each module instead of the more expensive crate controller concept used in CAMAC and other general purpose systems. Four-phase stepping motors were used for all mechanical drives because of their match with digital systems and high torque regardless of displacement required. An additional bonus was the non-energized locking torque which eliminated additional brakes.

Mechanics

All circuitry is done with standard wire wrap boards with modified I/O terminals to accomodate 40 pin arrays for our bus. Internal and external bussing uses a mass-terminated ribbon cable carrying 16 addresses, 8 data, and 8 control lines. The bus ribbon cable runs across the backs of the NIM modules and connects to each board through a short link. Servicing of any unit can be accomplished by disengaging the connector, putting the unit on an extender for power, and reconnecting the unit to the bus.

High-Speed Interface

This unit processes all the RF signals, deriving phase and amplitude information and converting these to digital data and validity flags respectively. The sample and hold circuit has a gain of unity. The digitizing gate is .2 μs wide and has a delay set by the CPU to one of 16 positions from .1 to 1.8 μs. The valid data detector generates a 'DATA OK' flag only if the low level signal (10 mV) can be separately detected from a 1 V signal on the same line.

Central Processing Unit (CPU)

Factors affecting microprocessor selection were accuracy, versatility, software support available, additional hardware chip variety now and in the future, and cost. Input phase measurements are digitized to 10 bits and this implies some 16-bit programming. The Intel 8080 seemed to be the most attractive micro in view of these considerations, and the extensive 8080 program development package[1] already running on SLAC's central computer was the feature that decided it. The three-chip Intel 8080 CPU, 1024 bytes EPROM, 256 bytes RAM, serial link chip (Intel 8251) is laid out on a single panel. Additional wire wrapping provides for memory expansion to 4K ROM and 1K RAM. All I/O is connected to the bus as high memory addresses. The serial link will connect to SLAC's control computer.[2]

Power System

One of the unusual features of the phasing system is its large number of I/O ports. Each of nine stepping motors has four ports requiring 1/4 A. The RF selection relays need four ports 1/2 A capacity. Miscellaneous I/O requires 16 more ports, giving a total of 56 ports. Although most of the wires from these ports must pass adjacent to the high power klystrons along the machine, the use of isolated ground returns and a separate 'rough' power supply to operate the motors and relays permitted direct drive without use of optoisolators. The only ports requiring these are associated with accelerator control, a minus 24 volt system. Quad optoisolators driving Darlington devices drive the 100 mA relays without additional amplification. Eventually this system will be replaced with the serial data interface at great reduction in complexity.

Phase Shifters

The phase shifters used are those originally purchased when the accelerator was first designed. The only change needed was the removal of the brake and replacement of motors.

RF Detector Panel

This panel selects the desired klystron from the eight in the sector. It also has a phase detector, amplitude detector, and video amplifiers. This unit was modified only to the extent needed to reverse the power supply polarity to accomodate the plus 24 V system of the microprocessor.

Programming the Controller

From the programmer's viewpoint, the controller measures phase differences between the reference signal and the beam or the output of any of eight klystrons. These values are read by a ten-bit ADC. Motor Commands can be issued to change the phase of the reference signal or any of the klystron output signals. The controller can measure either the normal or the 'wobbled'

* Work supported by the Energy Research and Development Administration

(Contributed to the 1977 Particle Accelerator Conference, Chicago, Illinois, March 16 - 18, 1977)

(changed by 180°) phase difference. Each digitized input has the following relationship to the measured phase difference Q.

$Y0 = A.\sin(Q) + B$ normal

$Y1 = A.\sin(Q + 180^\circ) + B$ wobbled

Where A, B are constants related to gain and DC bias respectively. The scaling is arranged so that: 1 ADC unit = 1 motor step = 0.25° phase. This fixes the variation of the digitized inputs Y0, Y1 to plus or minus 230 units. The bias B is nominally 512 units so values read for Y0, Y1 will vary well within range of the ten-bit ADC (0 to 1023 units). The control algorithm is briefly described below:

1) Read the digitized inputs Y0, Y1.

2) Calculate the difference Y0-Y1 using 16 bit arithmetic.

3) Clip (Y0-Y1) to an 8 bit signed number U (i.e. force the value of U to be within ± 127). Now 8 bit arithmetic can be used in later calculations. When the phase is far from the null position, a fixed correction of $\pm 34^\circ$ ($= \pm 127$ steps) will be made. Elsewhere, nonlinearity is bounded by $34.\pi/180-\sin(34)$ or 6%.

4) If this is the first reading of U, and it is already near a null, then replace U by (sign U) $.2^\circ$. This permits backphased signals (at the unstable null position) signals to be corrected.

5) Cause the motor to move N = -U steps. Note this will always move the phase away from the unstable null (at phase difference 180°) since the sign of N is opposite that of U.

6) Convergence is achieved only if the last read U is at a null and the previous U was not.

Gain Correction

It is unreasonable to assume that the scaling of all 30 phasing systems will remain as given above over long periods of time. The program can be modified to compensate for this by keeping a constant G initially = 1 in memory. If at any point the sign (previous U) is not the same as the sign (current U) then overshoot has occurred and G is halved. If at any point, we have more than three iterations in the linear region with U having the same sign, then G is doubled (undershoot). The number of motor steps is computed as N = -U/G rather than as step 5 above.

Program Structure

The software was structured into levels of hierarchy:

0 - real time delay
1 - acquire digitized point from the ADC (Y0 or Y1)
1 - move a specified motor by N steps
2 - calculate one clipped 8 bit signed value U
3 - calculate a smoothed data point U
4 - adjust phase difference to a stable null
4 - view a specified phase difference without changing it
5 - sequence the 8-klystron phasing operation
6 - accept commands (from computer or manual) and interpret

To each subprogram of a given level, every lower level subprogram functions like a single instruction that accepts inputs from and generates outputs to specified CPU registers and RAM locations. Each can be executed 'manually' with the appropriate call from the executive program (level 6).

Program Checkout

All software was entered, edited, saved, assembled, and simulated from a terminal connected to SLAC's central computing facility. Every correction is done at the source level and the entire collection of programs is assembled as one absolute 1K ROM image. For simulation, certain programs must be altered; for example, the real time delay subroutine is replaced by a RETurn instruction. Simulation output consists of a printout of all microprocessor registers and flags after each instruction 'execution' in selected regions of ROM. A single canned command gets the source program file, makes the fixed edits necessary for the simulation, then assembles and 'runs' it. Another command takes the same source file and creates an unedited ROM image which is burned into the phasing system EPROM through a special device connected to the central computer.

System Checkout Using the Test Unit

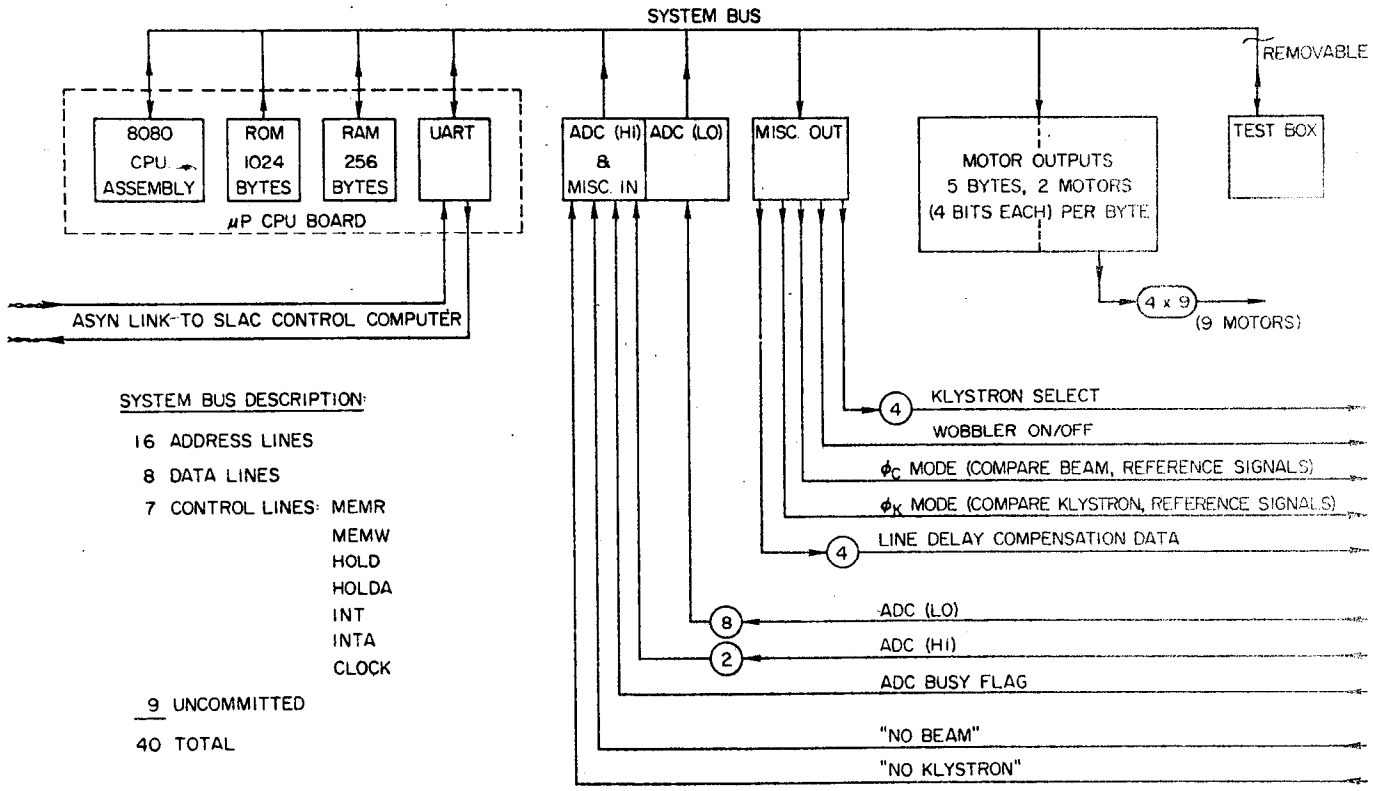
Included in the design is a single removable test box used for system development and trouble shooting. When needed, it plugs directly onto the bus of any of the 30 systems and performs either dynamic tests, with the CPU running, or static tests, with the CPU board disconnected. Functions include:

1. Read from any ROM, RAM, or readable I/O register.
2. Write into any RAM or writable I/O register.
3. Trap (halt) on any ROM, RAM or I/O register reference.
4. View any RAM or readable I/O register while the program runs.
5. Continue program execution, signle stop or high-speed, from any ROM location.

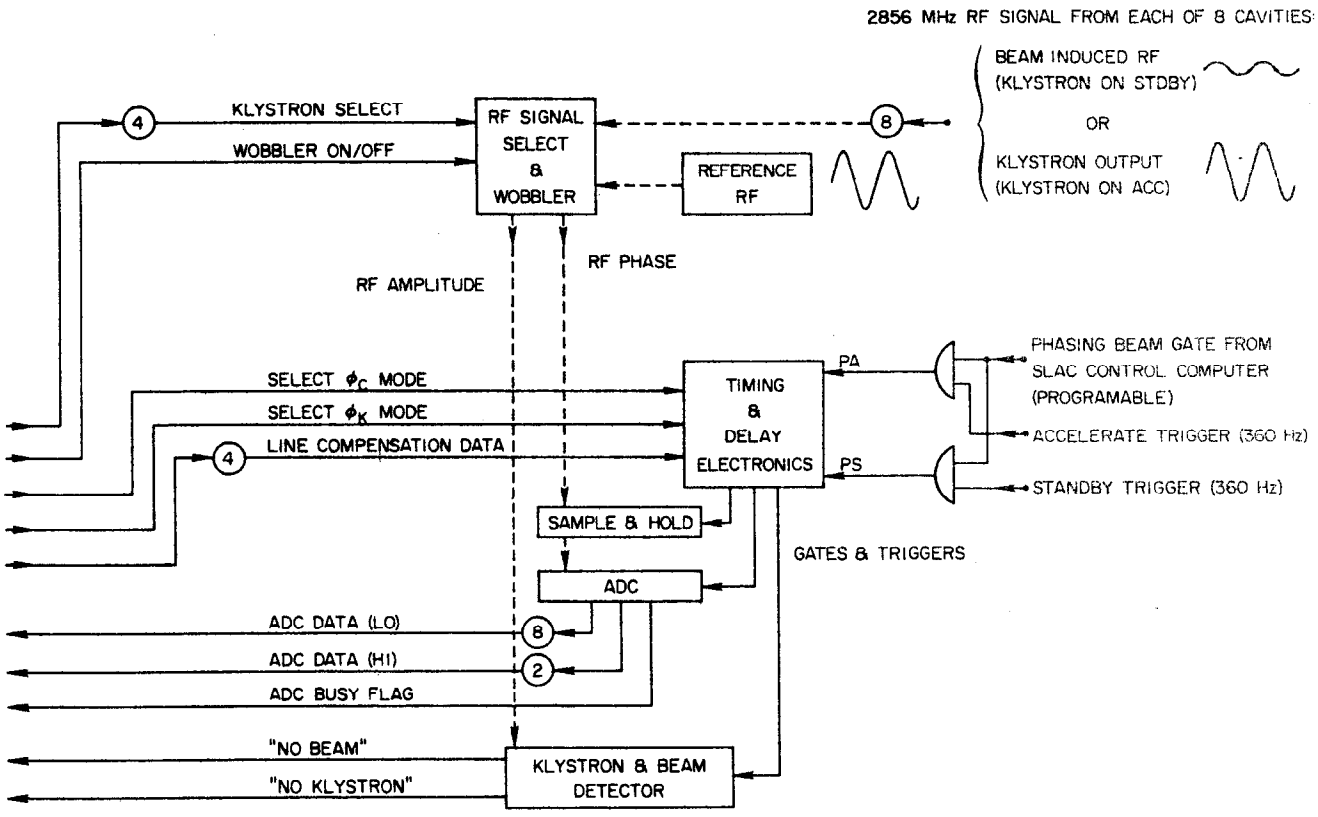
Register selections are made from a set of four hex thumbwheels and values are read from a two hex digit display. Design of the box was simplified by the fact that our system never uses the Intel in/out bus; all I/O registers are connected to the address bus just like memory. Also, it was decided not to build a facility to read/write the CPU registers or flags. This is not necessary because (1) computational parts of the code are debugged by simulation, and (2) desired checkout parameters are stored in a scratch RAM array so they can be viewed. System checkout can be made on the accelerator, or in the lab with a generator supplying the RF signals. In a typical static test, the stepping motors were replaced by lamps and each lamp was lighted independently of all others, as in a continuity check. A typical dynamic checkout procedure is to view one of the variables (klystron #), U, N, or beam/klystron mode flags, as the sector phasing system executes.

References

1. "Microcomputer Assemblers for the Intel 8080 and Motorola 6800", L. J. Shustek; SLAC CGTM 174 (May 1976)
2. "Applications of Microprocessors in Upgrading of Accelerator Controls", K. B. Mallory; Paper L-3 of this Conference



MICROPROCESSOR PANEL



RF DETECTOR PANEL & HI SPEED INTERFACE