# Applications of Microprocessors

## Chapter 9

### INTRODUCTION

Before designing a microprocessor system it is necessary to examine the application in the light of certain critical parameters. These include:

(i) The number of input and output ports required
(ii) Input/output structures and interrupts
(iii) Speed of operation
(iv) Production and development costs
(v) Previous experience
(vi) Existing hardware and designs.

None of the parameters listed above can be considered in isolation and their relative importance changes from one application to another. For small production runs, the overall development cost is a significant portion of the net selling price and under such circumstances it is probably best to capitalise on previous experience and existing hardware. This often results in using a much more powerful microprocessor than is necessary but nevertheless results in a considerable reduction in development costs. By contrast, in a high-volume application, development costs can be offset against a much longer production run and the first consideration in choosing a microprocessor becomes production cost. This is usually closely related to the number of components used in the design, and as a result microprocessor systems which are to be produced in quantity are designed to keep the package count to a minimum.

The conflict between high-performance general-purpose processors and low-cost minimum package count systems has produced five basic categories of processor systems:

(i) High-performance minicomputer-type processors (e.g., 9900, CP1600)
(ii) High-performance 8-bit general-purpose processors (e.g., 8080, 6800, 2650)
(iii) Low-cost general-purpose processors (e.g., SC/MP)
(iv) High-performance dedicated processors (e.g., PPS 8/2)
(v) Low-cost dedicated processors (e.g., F8, ALPS)

Table 9-1 gives a rough indication of the normal size of these five categories. The cheaper devices tend to trade off high-speed and powerful instruction sets in return for features such as on-chip input-output ports, built-in system clock and on-chip RAM.

**Table 9-1. Microprocessor Types and System Size**

| | TYPICAL DEVICE TYPES | MOST COMFORTABLE ADDRESS CAPABILITY (KILOWORDS) | NUMBER OF PACKAGES IN A SYSTEM |
|---|---|---|---|
| Minicomputer type | 9900, CP1600 | 4 to 64 | 20 to 100 |
| High-performance 8-bit general-purpose | 8080, 6800 | 2 to 32 | 10 to 50 |
| Low-cost general-purpose | SC/MP, 4040 | up to 2 | 3 to 10 |
| High-performance dedicated processors | 9002, PPS8/2 | up to 4 | 2 to 10 |
| Low-cost dedicated processors | F8, ALPS | up to 4 | 1 to 4 |

This chapter illustrates aspects of microcomputer system design by means of five case studies. These cover the full spectrum of applications from replacement of analog and digital circuits up to dedicated minicomputer systems. Each case study gives details of the application, method of operation and factors which influenced the overall system design.

## STAR DRUM PRINTER INTERFACE

The application. This section deals with a circuit used to drive a STAR drum printer from an Intel 8080 microprocessor. In the complete application the drum printer is the output device of an intelligent instrument, but the text here is restricted to the printer drive circuit alone.

The printer consists of a rotating drum which has characters embossed on its surface. The characters are arranged as 16 columns with 13 characters in each column. To print a character a hammer briefly forces the paper against the drum so that it picks up the appropriate character from the rotating drum. There are 16 hammers, one per column, and the drive circuit is required to activate each hammer at the correct instant of time so that the desired character is printed. The drive circuit also starts the drum motor whenever a print-out is to take place, and provides a paper feed after printing each row of characters.

Two timing signals are provided by the printer for use by the interface circuit. These are a reset signal, which gives a positive going pulse each time the rotating drum begins a revolution, and a character timing signal which gives a short pulse of between 200 and 400$\mu$s at the start and finish of each character. Figure 9-1 shows the timing diagram. The timing signals are generated by electromagnetic pick-ups and need some amplification. Paper feed is achieved by activating the paper-feed electromagnet for a minimum of 15ms. When starting the motor, 200ms should be allowed for the motor to run up to full speed before printing begins.

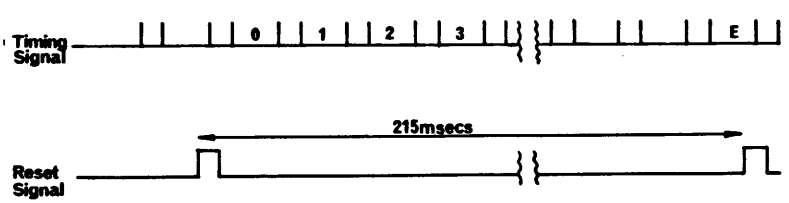


*Figure 9-1. Printer timing diagram*

The circuit. Figure 9-2 shows the block diagram of the circuit used. It consists of two 8212 8-bit latched output ports which are used to drive the 16 hammers, a 2-bit output port which drives the motor and the paper-feed mechanism, and a 2-bit input port which is used to connect the reset and character timing pulses onto the data bus. A dual 2-line to 4-line decoder is connected to the address bus so that the ports are allocated the following addresses:

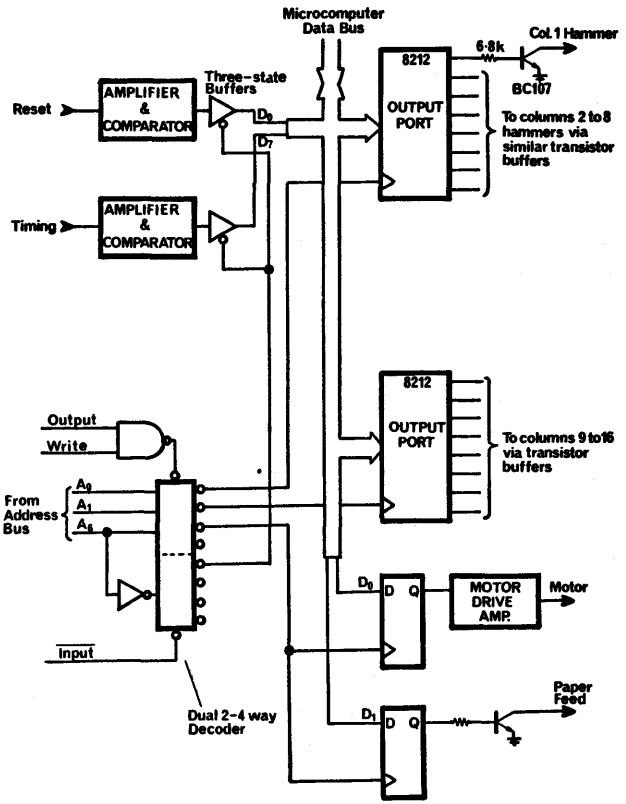


*Figure 9-2. "STAR" printer interface circuit*

Output Address x1xxxx00 Port for column hammers 1 to 8
$D_0$ corresponds to column 1
$D_7$ corresponds to column 8

| | | |
|---|---|---|
| Output Address | x1xxxx01 | Port for column hammers 9 to 16<br>$D_0$ corresponds to column 9<br>$D_7$ corresponds to column 16 |
| Output Address | x1xxxx10 | $D_0$ port switches on drum motor<br>$D_1$ port drives paper-feed mechanism |
| Input Address | x1xxxx00 | $D_0$ input port – Reset pulses<br>$D_7$ input port – Character pulses |

(x in the address indicates undefined bit)

The Intel 8080 has separate input and output instructions and when these are executed the input (or output) control line is strobed and the appropriate 8-bit port address is encoded into bits $A_0$ to $A_7$ on the address bus. The output strobe line is ANDed with the write pulse to provide an enable input to the address decoder, and the decoder output provides the clock pulse for writing information from the system data bus into the output port latches. Similarly the inputs from the two magnetic pick-up amplifiers are placed on the system data bus when the appropriate address and the input enable line are valid.

The circuit operates by putting a logic 1 on the appropriate output bit at the correct instant of time so as to print the desired character. The timing is done entirely by software. The timing pulses from the magnetic pick-up are polled using an input instruction; hardware interrupt is not used.

The software. Sixteen adjacent memory locations are used to store, in binary code, the 16 characters to be printed. When the drum has character "0" available for printing, the software makes a scan through the 16 locations to see if any of the characters is a zero; for those that are zero a logic "1" is sent to the appropriate hammer and then, after the specified period, all outputs are returned to logic "0." The process is repeated for character "1" and so on until one revolution has been completed. Then the paper feed is activated and the drum motor switched off.

The Intel 8080 has a number of internal registers as shown in Figure 9-3, and these can be used either as single 8-bit registers or two may be paired together for use as a 16-bit address pointer. In this application the internal registers are allocated as follows:

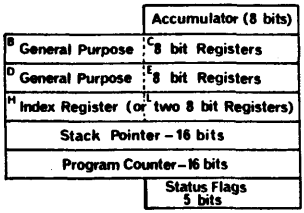


*Figure 9-3. Internal organisation of Intel 8080*

A – Accumulator.
B – Working register.
C – Hammer control bit pattern. The contents of C are eventually sent to the output port to turn on the appropriate hammers.
D – Counter to count the number of characters scanned through in the 16 memory locations.
E – Counts character pulses and thereby gives an indication of current character available for printing.
H and L – Used as a 16-bit address pointer to point to characters stored in memory.

The printer software routine is called as a subroutine and the main program arranges for the 16 characters printed to be in the locations pointed to by H and L. Upon entering the print routine, the program turns on the motor by sending a logic "1" to the motor drive output port and then waits until it has received 2 reset pulses before scanning through the first 8 characters to see if any is a zero. The software determines which characters are "O" and builds up a hammer drive pattern for the output port for columns 1 to 8 in register C.

The contents of register C are then sent to the output port and the routine is repeated for columns 9 to 16. The scan and output routine is done quite fast so that the delay between loading the two hammer ports is not significant to the electromechanical printer. The software now waits until a character pulse signifying the end of a character is received and the two ports are then loaded with all zeros. The process is repeated for the 1's character, 2's character, etc. Finally after the thirteenth character is printed,

the paper-feed mechanism is activated for two character pulses and the motor is turned off.

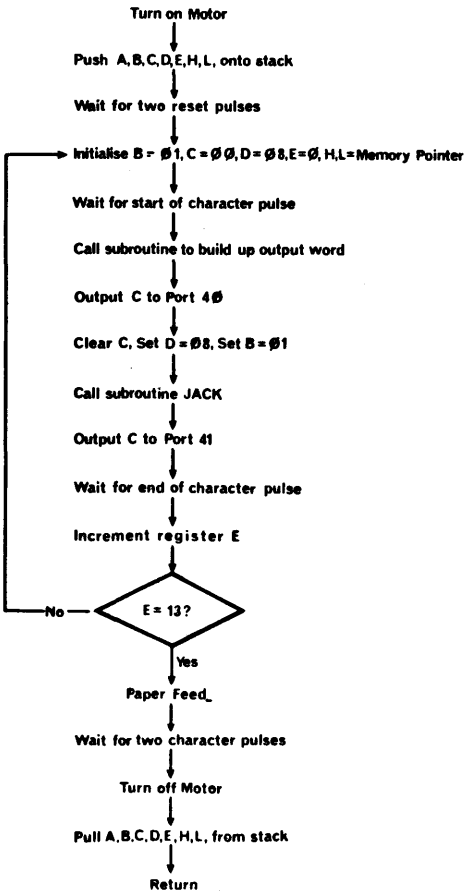


*Figure 9-4a. "STAR" printer interface program flow chart*

Figure 9-4a shows the complete program flow-chart. Subroutine "JACK" (see Figure 9-4b) performs the scan through 8 consecutive memory locations and builds up the hammer drive pattern in register C.

Software techniques. The program makes great use of the internal registers of the 8080 to store loop counts and bit patterns. Since most of the instructions which manipulate the internal registers

are single byte instructions, the total program is economical in program storage (about 110 bytes). The instructions OR, SHIFT and COMPARE are used to advantage: in a simpler processor without say the "COMPARE" instruction, the program length and execution time would be somewhat longer. In this particular application, the printer is used at the end of a long data analysis routine and there is no possibility of an interrupt occurring, but where an interrupt might occur it would be necessary to either mask-off the interrupt or rewrite the subroutine as a re-entrant one. This particular program was written in assembly language run on an Intellec 80 development system. The assembly language facility was necessary because the overall application was large and it was not feasible to program throughout in hexadecimal code, although the printer section of the program could have easily been written in hexadecimal code. The total system development time taken by an experienced 8080 user was about 40 hours. This included the time required to lay out and construct a printed circuit board.

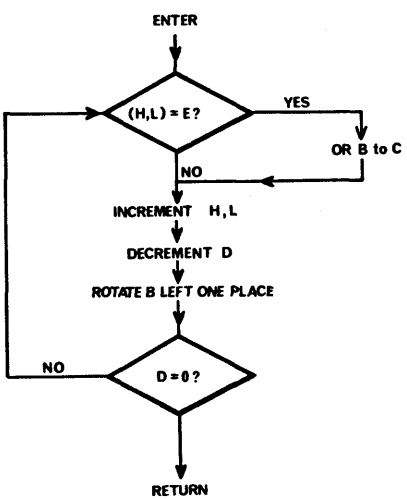


*Figure 9-4b. Subroutine to build up output hammer drive 8-bit word*

## AN INTELLIGENT LAMP-DIMMER FOR STUDIO LIGHTING

The application. This application is the control of the intensity of

individual studio lights. It is required to set the lamp intensity from a master control computer and to fade lamp intensities from one setting to another over a specified period of time. Each lamp intensity during the fade time can be expressed by the equation:

$$\text{Lamp Intensity} = \frac{[\text{new setting} - \text{old setting}].t}{\text{fade time}} + \text{old setting}$$

$$= (\text{new} - \text{old})\frac{t}{\tau} + \text{old}$$

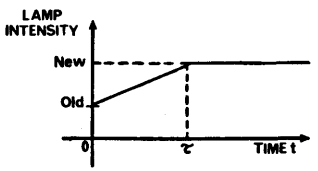


*Figure 9-5. Lamp fading method*

The lamp intensity setting and the fade time are sent from the master computer to the lamp dimmer as two 8-bit bytes: the intensity setting has a 0 as its most significant bit and the fade time has a 1 as its most significant bit. It is the function of the lamp dimmer to accept the two 8-bit words and solve the above equation for its own particular lamp. The circuit also handles phase control of the triac used to dim the lamp.

In a typical application the master computer handles between 100 and 200 lamps so that the task of continuously solving the above equation for each of the lamps is a particularly onerous one. In the approach discussed here, the equation solving is delegated to each dimmer circuit and the master computer becomes simple enough to be replaced by a general-purpose 8-bit microprocessor. The design of the complete system is outside the scope of this chapter and the text concentrates only on the intelligent lamp dimmer. The microprocessor-based dimmer replaces a D/A converter and an analog phase-control firing circuit, as well as relieving the master computer of some of its workload.

Lamp dimmer circuit and the principle of operation. Figure 9-6. shows the complete dimmer circuit. Information is sent from the master computer as two 8-bit bytes. The two bytes are transmitted

at least one full mains cycle apart so as to give the circuit time to read in each byte. The fade period begins 8 cycles after receiving the new intensity information.

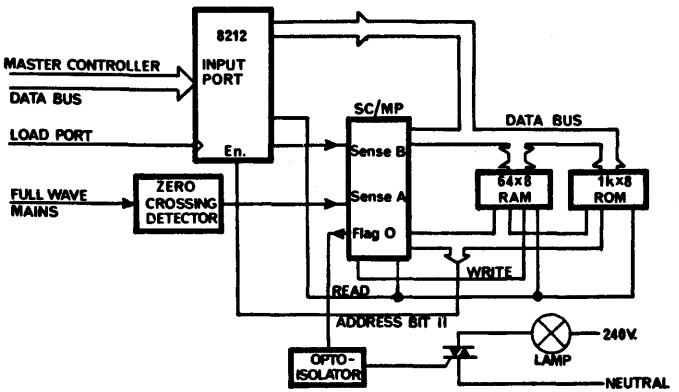


*Figure 9-6. Block diagram of fading circuit*

One of the features of the SC/MP microprocessor is that it has simple on-chip input and output ports which are used to advantage in this application. The inputs consist of an interrupt (Sense A), a single bit input (Sense B) and a serial input. There are 3 output flags ($F_0$, $F_1$ and $F_2$) and a serial output.

A mains-driven zero-crossing detector drives the interrupt line and the input request flag contained within the 8212 is connected to the Sense B input. When new information is written into the input port, the input request flag is automatically set and this can be detected by the processor. Output flag $F_0$ is connected through an opto-isolator to drive the triac. Phase-control of the triac is achieved by sensing the start of a mains cycle and waiting for a known number of clock cycles before firing the triac.

The 8-bit byte from the master computer gives scope for 128 different intensity and 128 fade time settings. One unit of a fade-time is taken as 8 full main cycles, which gives maximum fade times of 20.32 and 16.9 secs at 50 and 60Hz respectively. The relationship between lamp intensity and number of cycles delay is

not a linear one and this nonlinearity is accommodated by a 127-word look-up table. The desired lamp intensity expressed as a binary integer between 0 and 127 is used to form an address to reference the look-up table and obtain the appropriate delay time.

Techniques. The SC/MP has a limited instruction set and, in addition, an average instruction execution time of 30$\mu$s. These two factors combine to produce an 8-bit multiplication time of about 5ms and an 8-bit division time of 8ms. Clearly it is not feasible to evaluate the fading equation for each half cycle and alternative methods must be used. The technique employed is to evaluate the following expression in the eight cycles preceding the start of the actual fade time.

$$\text{Step size per unit of fade time} = \frac{(\text{new} - \text{old})}{T}$$

Thereafter the intensity is increased by one step size for every eight mains cycles until the fade time is completed. The final result is checked with the desired final value to eliminate errors resulting from round-off in the division process.

Whilst the microprocessor is counting out the delay prior to firing the triac, it is unable to make any useful calculations. In the case of the 8-cycle lead-in to a fade period, this could cause problems when the delay time is long because there would not be enough time left in the mains half-cycle to do the necessary calculations. To overcome this the microprocessor checks the delay time in the preceding half cycle, and if the firing point is in the first $\frac{\pi}{2}$ radians, it sets the interrupt vector so that, on receipt of the interrupt, the processor times out the delay and then does the calculations. If the firing point is in the range $\frac{\pi}{2}$ to $\pi$ radians, a different interrupt vector is loaded into the vector register so that the processor does the calculations, adjusts the delay time to compensate for the time taken to do the calculations and then times out the modified delay before firing the triac. It will be observed that the ability of the SC/MP to dynamically change the interrupt vector is a particularly useful feature.

The main reasons for selecting the SC/MP microprocessor for this application were low cost, 8-bit compatibility with the master computer, on-chip output for driving the triac, interrupt capability

and the ability to use UV erasable ROM's for program storage without any support circuits. A total production run of only 250 units was anticipated and mask-programmed ROM's were not economically feasible.

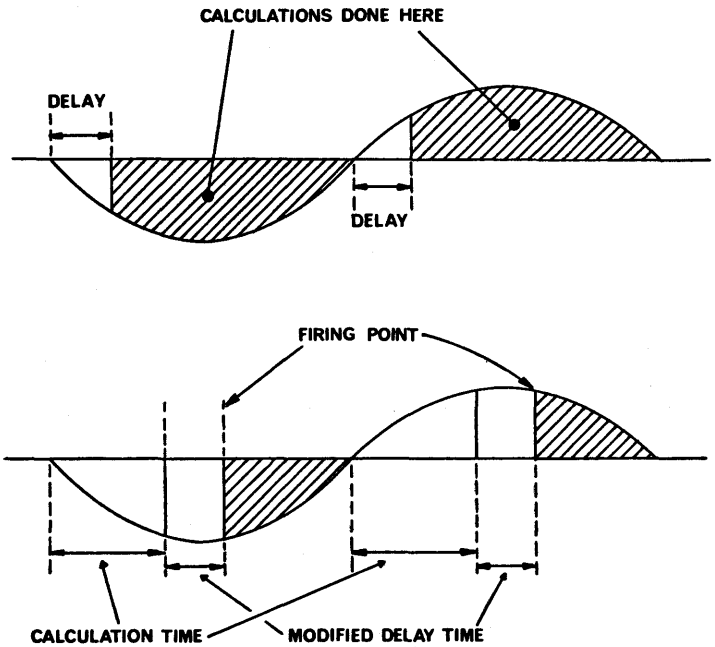


*Figure 9-7. Diagrams illustrating relationship between triac firing point and period allocated to making calculations*

System development was carried out by programming the U.V. erasable ROM's first with a simple program to check the hardware, and then with segments of the main program until the complete program was proved. Prior to designing the system, the SC/MP processor had been evaluated using a low-cost kit available from National Semiconductor. The task of writing the program and then converting to hexadecimal code for use by the programmer was done entirely by hand. This approach appears to be feasible for programs up to about 1 kbyte.

## AN OPTICALLY ISOLATED THREE-TERM CONTROLLER

The application. A three-term controller is used in control systems to improve the overall response, stability, and accuracy of the control system. The controller has an input-output relationship as follows:

$$V_{out} = A.V._{in} + B\int V_{in}.dt + C.\frac{dV_{in}}{dt}$$

The application discussed here is a self-calibrating system in which the multiplying factors A, B and C are entered from a remote computer. Figure 9-8 shows the complete system diagram. The input voltage is converted into digital format using a V/F type A/D converter. This digital value is passed to the microcomputer which evaluates the equation and gives an analog output using a D/A converter. One important feature of the system is that the analog circuits are isolated from the digital section by means of five opto-isolators. This eliminates noise problems and, since all the analog circuits are low power devices, their supply voltages are derived from a small dc-to-dc converter.

The circuit. A V/F converter is used to generate a frequency proportional to the input voltage. This frequency is fed through an opto-isolator to a 12-bit counter whose contents are automatically loaded to a 12-bit latch every 80ms (once every four mains cycles at 50Hz) by a real-time clock. The real-time clock also drives the interrupt line of the microprocessor. Each interrupt causes the microprocessor to reset the counter and read the contents of the latch, so that the counter contents are proportional to the average input analog value over the 80ms period. A time frame of 80ms was chosen because it guarantees good normal mode rejection. The voltage-to-frequency converter is run with a frequency offset at 25.6KHz so that it can accommodate both positive and negative input voltages. Zero input voltage gives a frequency of 25.6KHz, negative input voltages decrease the frequency and positive inputs increase the frequency. Full-scale input of +2 volts gives 51.2KHz.

To obtain the true numerical value of the input the microprocessor subtracts the 80ms count for zero input volts from the 80ms count for the unknown input. An analog multiplexer is used at the

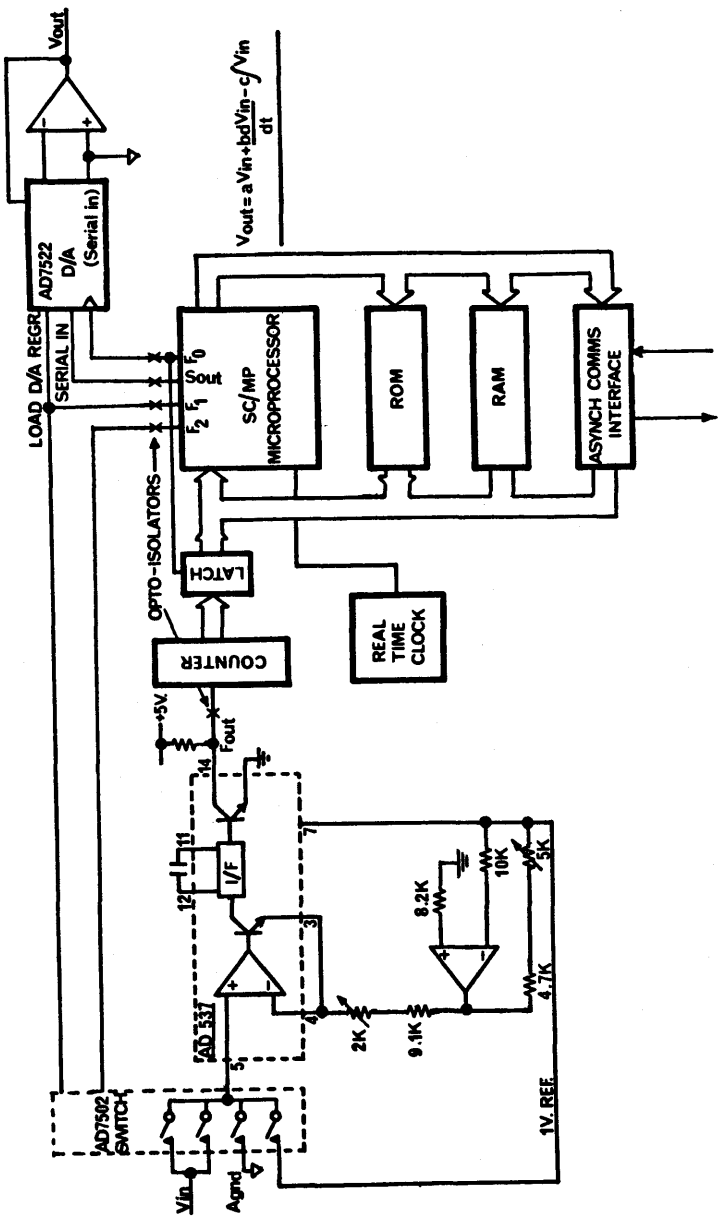


*Figure 9-8. Optically isolated three-term controller*

input to the V/F converter so that analog ground and the 1-volt internal reference of the V/F can be converted and used by the microcomputer to measure the frequency offset and check the calibration.

After the equation has been evaluated, the result is output to an AD7522 D/A converter using the serial mode of operation (see Chapter 8). Serial data transfer was used because it requires very few opto-isolators.

The circuit also includes a serial asynchronous communications interface for entering the factors A, B and C. In theory, it is possible to do the asynchronous communications using the serial input-output features of the microprocessor, but since the SC/MP is already overloaded with other software tasks, communications management was delegated to the separate device.

Software techniques. Voltage-to-frequency conversion was chosen for the conversion method not only because of the optical coupling facility but also because the true integral of the input signal can be obtained by adding successive data samples. This significantly reduces the software. The differential of the input signal is calculated using the least squares moving point polynomial*

$$\frac{dx_n}{dt} = \frac{-2x_{n-2} - x_{n-1} + x_{n+1} + 2x_{n+2}}{10}$$

where $x_n$ is the sample value at time n. Most of the software time is taken up in multiplying in the factors A, B and C. All the arithmetic is done in 16-bit fixed point two's complement notation. 16-bit precision is not necessary for the application but it does provide a fairly straightforward way of avoiding round-off and overflow errors.

Summary. One of the disadvantages of the circuit discussed here is that it is only suitable for low speed control systems, the data sample rate is only 12.5Hz. The speed limitation is imposed by two factors; first, the V/F conversion technique is inherently slow, and second, the SC/MP microprocessor is slow and has a limited

*See Savitzky and Golay, Smoothing and Differentiation of Data by Simplified Least Squares Procedures, Analytical Chemistry, Vol. 36 No. 8, July 1969, pp. 1627–1639.

instruction set. The authors have used the SC/MP microprocessor in several applications and it has been found that the on-chip input and output features are extremely powerful, but the absence of direct addressing instructions and automatic stack management make it difficult to program. Another disadvantage is that all internal register manipulations are made using the accumulator and this means that the accumulator cannot be used to temporarily store data. This in turn slows down program execution. Nevertheless, for slow, simple, low cost applications the SC/MP microprocessor has much to recommend it.

## A DIGITAL CONTROLLER FOR A DOMESTIC HEATING SYSTEM

The application. A microprocessor system is required to replace the conventional electro-mechanical logic (bimetalic thermostat, time switches, etc.) associated with the control of the start-up/shut-down of the heating system and with the regulation of the temperature within the building. The controller is also to act as a digital clock and provide a display in hours and minutes. The heating system is switched on or off via a mains contactor with a low voltage coil and the temperature inside the building is monitored using a single-point temperature measurement ($\theta$). The start-up time ($t_s$), the shut-down time ($t_f$) and the desired setting for the inside temperature ($\theta_D$) are to be defined by the user after the controller has been installed. A typical inside temperature profile is shown in Figure 9-9. Simple ON/OFF control is acceptable for regulation of the inside temperature when the building is occupied. Outside the occupancy period, the heating system is to be switched off. The complete control system is to be mass-produced and should have low material and production costs.

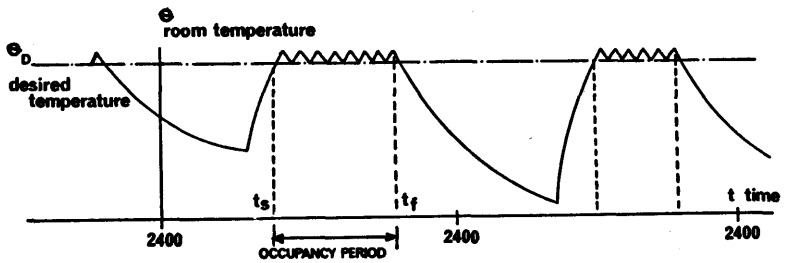


*Figure 9-9. Typical inside temperature profile*

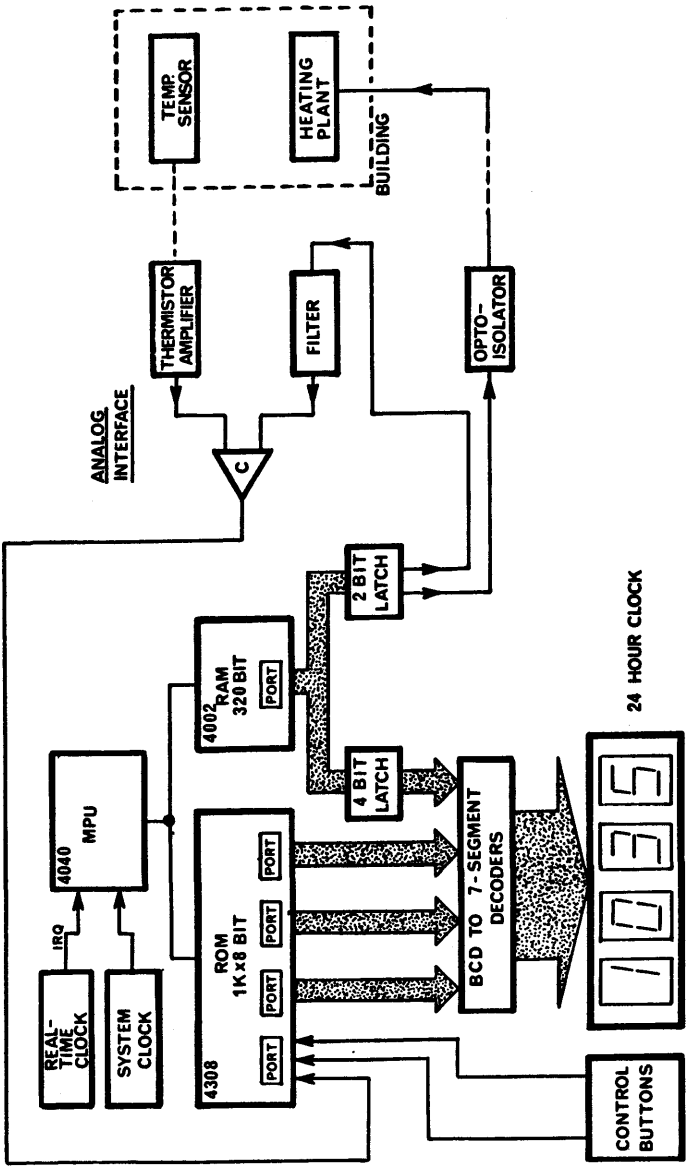


*Figure 9-10. Production design of the heating controller*

The design of the microprocessor-based controller. Since 4-bit resolution provides sufficient accuracy for domestic control applications, a cheap, but powerful, 4-bit microprocessor system is most appropriate. A design for the production version of the control system, based on the Intel 4040 chip, is shown in Figure 9-10. The main aim of the design is to minimise the number and complexity of the support circuits by using the microprocessor program to perform as many of the tasks associated with the control system as possible.

A single mask-programmed ROM chip holds the entire program and a single RAM chip provides sufficient data storage for this application. (The microprocessor chip itself includes a 96-bit scratch-pad memory.) Both ROM and RAM chips contain on-chip I/O ports which are used to drive the 24-hour clock display and to communicate with the other I/O devices.

The need for a costly analog interface is avoided by performing the logical operations associated with the analog-to-digital conversion in software and by using a minimum of external components. The techniques of the tracking A/D converter and the variable pulse-width D/A converter are most suitable for software realisation. By measuring the temperature with a thermistor sensor, the only external analog circuits are a thermistor amplifier, an analog filter and a voltage comparator. All can be constructed using low-cost integrated-circuit operational amplifiers. The desired temperature is set by adjusting the offset voltage of the thermistor amplifier.

Program execution is synchronised to real-time by interrupts generated at each zero-crossing of the mains voltage. A four-digit seven-segment display indicates the current time based on a 24-hour clock implemented in software. The clock display is also used in association with two push-button controls to allow the user to enter the desired start and stop times. Under program control one button causes the displayed time to be held or to be incremented at high speed. The other button enters the currently displayed value of time into a particular location in the data memory.

The coil of the mains contactor is driven through an opto-isolator to eliminate the effect of switching noise.

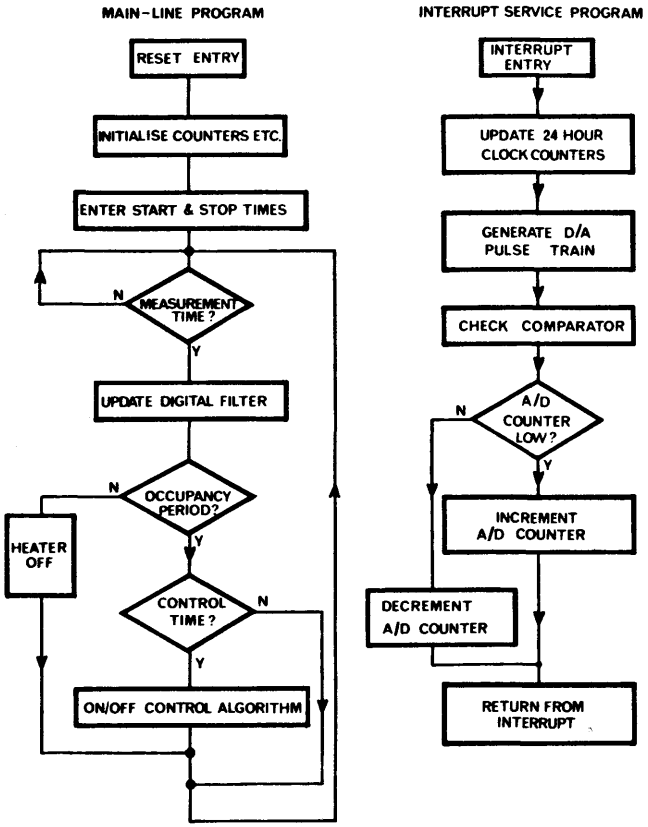


*Figure 9-11. Simplifed program flow chart for heating controller*

A simplified flow-chart for the program is shown in Figure 9-11. When power is first applied to the system, program execution starts at the reset entry point. The working registers and counters are initialised and the user enters the desired start-up and shut-down times using the control buttons. Interrupts are then enabled and real-time operation begins. The 24-hour clock counters are updated and the analog-to-digital conversion algorithms are processed in the interrupt service program which is entered every 10ms. All input-output data transfers are made under program control via the I/O ports. The output from the software A/D converter is sampled every minute and digitally filtered to reduce

signal and conversion noise. The filtering algorithm involves the most complex of the arithmetic operations – an 8-bit x 4-bit unsigned binary multiplication. During the occupancy period, control action is taken at 5-minute time intervals.
The entire program has an approximate length of some 640 bytes.

System development techniques. Since the major part of the program was concerned with special-purpose input-output operations, it was essential to develop the software on a microprocessor system which was, in design, as near as possible to the final production version. However, the development version of the controller differed from the production version in two ways:

(i) The program store had to be field-programmable to allow program modification during software development. In this application, an erasable RePROM and a Pseudo-ROM (a plug-in, ROM pin-compatible device consisting of a non-volatile read/write memory with hexadecimal keyboard input and hexadecimal display output) were used.

(ii) A control panel was included in the system to allow user control of system operation and program execution during software development. A simple panel with the basic system controls (reset, stop, etc.), binary input (bit switches) and binary output (LED display) was sufficient.
Where the program is developed in RePROM, there are several techniques which may help to reduce the number of reprogramming operations and simplify the task of program debugging:

(i) Groups of "no operation" instructions (op-code "00" hexadecimal) can be included at strategic points in the program so that jump instructions can be inserted at some future time to modify the program without erasing the RePROM.

(ii) A short execution control program loaded at the reset start address of the RePROM allows the user to enter a desired program start address from the control panel and then to transfer program control to this address. Thus, several small programs can be loaded and tested in the same RePROM in one operation.

(iii) The halt instruction and the stop/start control button allow the user to introduce temporary pauses in program execution at strategic points so that the status of I/O signals can be tested.

A program of this length and complexity can be written in machine code mnemonics and converted to hexadecimal code by hand. However, the task is tedious and detailed up-to-date program listings are essential if trivial syntax errors are to be avoided.

## AN "INTELLIGENT" INSTRUMENT FOR MEASURING PERIPHERAL BLOOD FLOWRATES

The application. A microprocessor-based instrument is required to automate the radio-isotope clearance method of measuring peripheral blood flow in a tissue. A radio-active tracer is placed on the tissue and its rate of removal, as indicated by the radio-active decay curve, can be related to the blood flowrate through the tissue. For a single blood flow path through the tissue, the decay curve is exponential and the blood flowrate is evaluated from the "half-time" of the decay curve ($T_{½}$). Where more than one blood flow path exists, the curve will have several components, each with its own $T_{½}$ value. In this application the instrument is required to calculate two $T_{½}$ values defining the blood flowrates associated with two blood flow paths from a tissue.

The radio-activity level is monitored with a scintillation detector placed in contact with the tissue. Count rates are in the range 100 to 1000 pulses/sec. The instrument has two operating modes:

(i) A single sample mode to allow the operator to take spot readings of radio-activity and to measure and enter the "background" count rate.

(ii) A run mode to allow the operator to initiate the automatic collection and storage of a predefined number of radio-activity samples at a predefined sampling rate. From the collected data, the two $T_{½}$-values are then estimated using a simple numerical procedure based on the linear least-squares method.

During sampling, the last measured value of the count rate is to be displayed in integer format, and after estimation, the two $T_{½}$ values are displayed in four digit "movable" point format.

Figure 9-12 shows a typical decay curve and illustrates the required front panel controls and displays for the instrument.

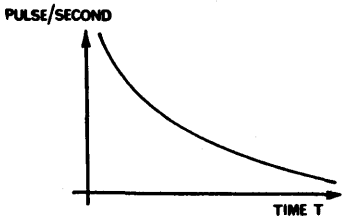


*Figure 9-12a. Typical decay curve of radioactivity*

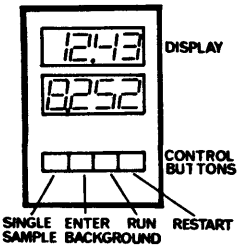


*Figure 9-12b. Blood flowmeter front panel.*

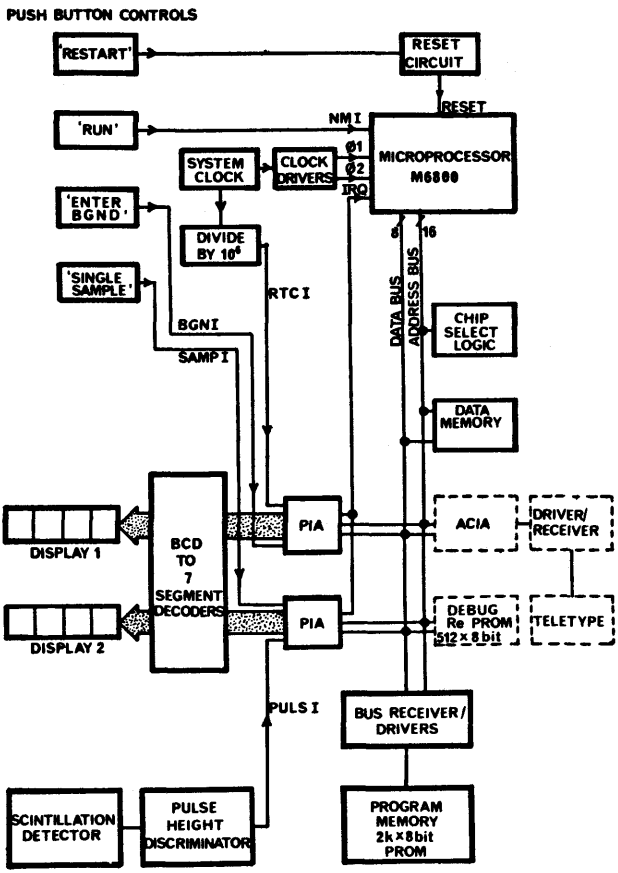


*Figure 9-13. System design of blood flowmeter*

The design of the microprocessor-based instrument. The instrument design is based on the Motorola M6800 microprocessor system. Extensive use is made of the microprocessor's powerful interrupt system and, to simplify the external circuits, I/O signals are connected into the microprocessor via general-purpose interface chips. As shown in Figure 9-13, the two 4-digit 7-segment displays are driven via decoders from two PIA (peripheral interface adapter) chips with each chip acting as a 16-bit parallel output port. The high priority "restart" and "run" front panel control buttons are connected directly to the microprocessor chip and generate the reset and non-maskable interrupt (NMI) signals respectively. The other control buttons, namely the "single sample" and "enter background" controls, are connected through a PIA to generate the SAMPI and BGNDI maskable interrupt signals respectively. The system clock, when divided down to 1Hz, generates the RTCI interrupt signal. The output from the scintillation detector, after passing through a pre-amplifier and pulse-height discriminator to reduce the background noise level, generates the PULSI interrupt signal. The PIA interrupt control logic allows each of the four types of maskable interrupt signals to be enabled or disabled individually under program control. The teletype, which is used only during program development, is connected into the system via an ACIA (asynchronous communications interface adapter) chip.

A single RAM chip provides sufficient memory capacity to satisfy the requirements of the program stack, the common working registers used by the extended-precision arithmetic routines, and the storage buffer for the count rate samples. The main program memory, which uses fusible-link PROM, is built on a separate board with signals interconnected to the microprocessor through three-state driver/receivers. During program development, a single RePROM chip, which holds the teletype paper-tape utilities and the software debug package, is also included in the system.

The application program can be divided into two main sections:

(i) Interrupt servicing and control.
(ii) Data analysis.

The interrupt control software synchronises the program execu-

tion with real-time, determines the mode of operation of the instrument and provides a measure of the count rate. For example, the count rate is determined by counting the number of PULSI interrupts occurring within a period of time defined by counting RTCI interrupts. Figure 9-14 shows the simplified flow-charts of the sections of the program concerned with interrupt service after the operator has depressed the "single sample" button and generated a SAMPI interrupt request. It is assumed that all IRQ interrupts are enabled and all but the SAMPI PIA interrupt lines are disabled. (All of the internal registers of the microprocessor are stacked away when an interrupt request is recognized. They are restored during execution of the return from interrupt instruction.)

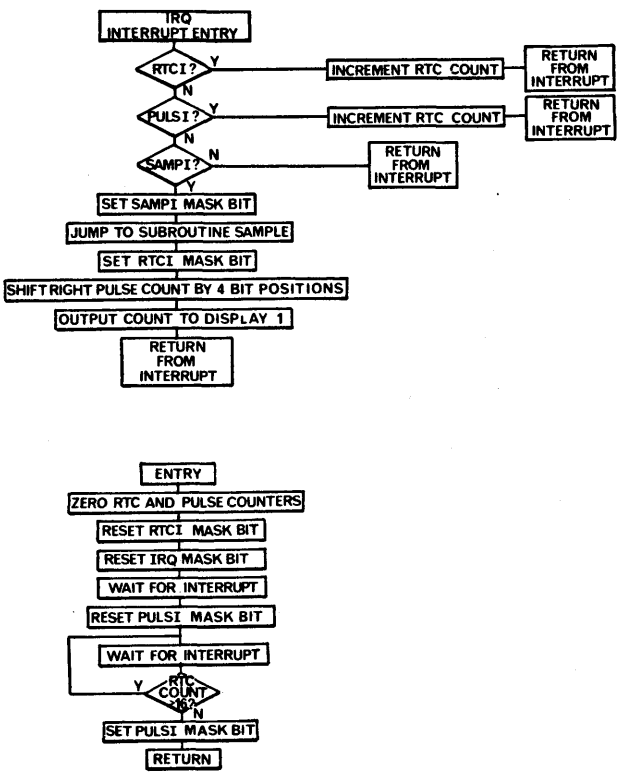


*Figure 9-14. Program flow-charts for single shot measurement of count rate*

The interrupt control program accounts for approximately 300 bytes of program memory.

In this application the data analysis is performed using two's complement binary integer arithmetic with intermediate scaling checks. The main estimation program is approximately 1 kbyte in length and requires six extended-precision arithmetic routines:

(i) A three-byte binary addition routine.
(ii) A three-byte binary subtraction routine.
(iii) A three-byte by three-byte signed binary multiplication routine.
(iv) A six-byte by three-byte unsigned binary divide routine.
(v) A five-byte square-root routine.
(vi) A two-byte natural logarithm routine.

Together with a utility routine to transfer n-byte binary numbers between different locations in the data store, the arithmetic routines require some 600 bytes of additional program memory.

System development techniques. The program was written in 6800 assembler mnemonics and translated into hexadecimal machine code using a cross-assembler executed on a large IBM 360 computer system. During program development, pin-compatible RAM chips replaced all but one of the PROM chips in the program memory of the instrument. The remaining PROM chip held the vector of interrupt trap addresses. The application program was loaded into the program memory via the teletype and under the control of the software debug program. The "reset" interrupt trap address was initially set to cause execution of the debug program whenever the system was reset during program development. After the application program had been completely tested it was dumped into paper-tape via the teletype and loaded into the fusible-link PROM's using an off-line paper-tape PROM-Programmer. The PROM's were then substituted for the RAM chips in the program memory. Finally, the ACIA, the debug PROM and the teletype were disconnected from the instrument and the "reset" trap address reset to point to the start of the application program.

To test complete microprocessor-based instruments it is frequently convenient, and sometimes essential, to write a high level language version of the data analysis section of the microprocessor program

which can be run on a large computer system. Using artificially generated input data, the output from this numerically accurate version of the program can provide comparative data to verify the basic analysis technique and to check the results and the accuracy of the microprocessor program.

## THE BUY OR MAKE DECISION

Microprocessors offer the possibility of very low cost general-purpose computers. Several manufacturers offer ready-built or half-built computers based on well-known microprocessors. The microprocessor user is faced with the decision whether to buy a ready-made system for his application or to construct one in-house. As a rough guideline it is estimated that, where the total number of units to be constructed is less than 100, then it is certainly cost-effective to buy the complete system. For between 100 and 250 units the decision is less clear-cut, and above 250 units the user should consider designing and building his own system.

In the same way as microprocessor manufacturers offer ready-built computers based on their products, the companies specialising in A/D conversion offer complete analog subsystems which plug into the popular microcomputer chassis. The make or buy decision for analog subsystems is much the same as described above, except that the design and management of analog circuits at printed circuit board level is rather more difficult than that for digital circuits and requires different expertise. Consequently it is even more desirable to buy ready-made analog interfaces for micro-computers and the break-points in the buy or make decision move towards higher numbers. This argument for buying ready-built analog subsystems is particularly applicable for systems companies which use the microcomputer as a cheap solution to a particular applications problem.

Figure 9-15 shows the system diagram for a general-purpose analog subsystem which plugs into the Intel SBC80/10 series of computers based on the 8080 microprocessor. The input circuit consists of a 16-channel analog multiplexer, which feeds through a programmable gain amplifier to a sample-and-hold amplifier and thence to a

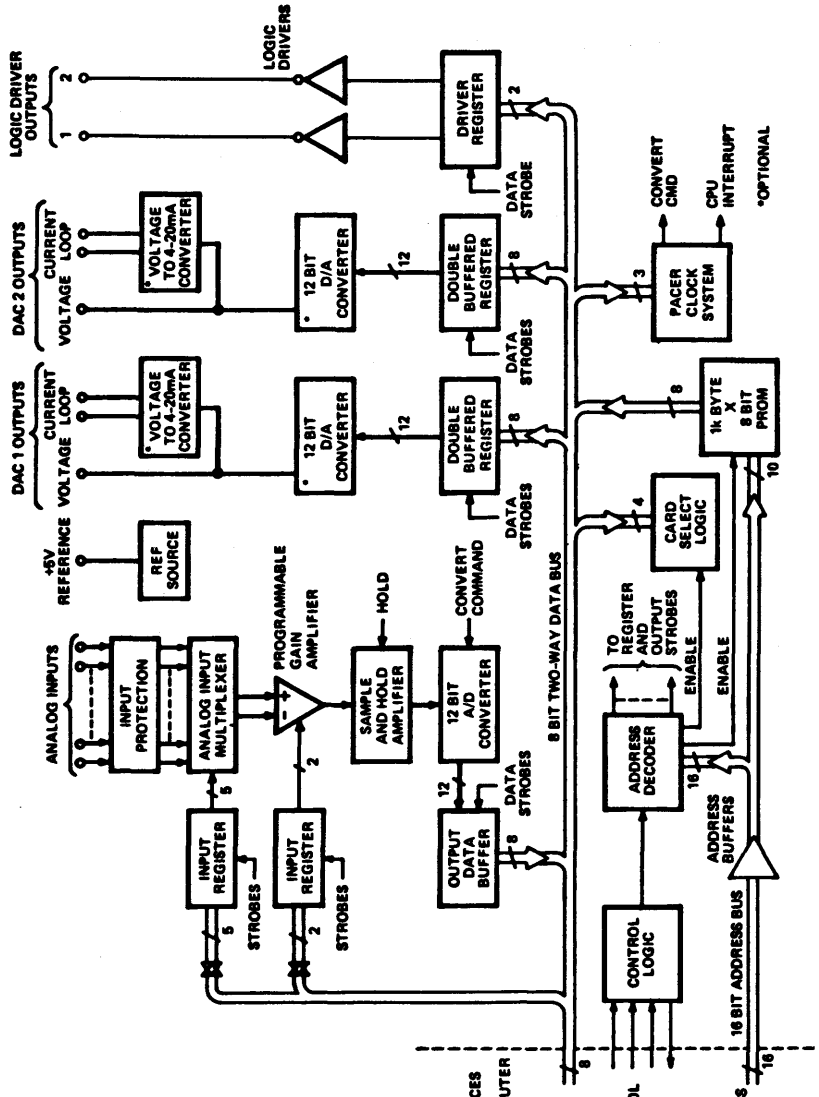


*Figure 9-15. RTI 1200 analog interface circuit for Intel SBC computer*

12-bit A/D converter. The output circuits consist of two double-buffered 12-bit D/A converters. Note that the system also includes a programmable real-time clock (called pacer clock by the manufacturer) because most microprocessors with analog interfaces require this feature. The reader is referred to the manufacturer for a full description of the system.

Finally, a word of warning. Although microprocessor-based computers offer low cost computing power, it is important to recognise that they do not offer low cost software. The Intel SBC80/10 has been used by the authors to drive automatic integrated-circuit test fixtures. The programs for driving the test fixtures use about 4K memory bytes and were written in assembly language. Software development was done on an Intel microprocessor development system consisting of a VDU, an 8080 system with 32K of RAM and a twin floppy disc drive. It is considered that this represents the minimum development hardware for this application. One of the important factors in choosing the Intel SBC80/10 was that it has extensive software support.

## SUMMARY

This final chapter has attempted to give some insight into how microprocessors are used and the engineering decisions which the user must take. Most engineers over-estimate the speed and computing power of microprocessors and, as a result, encounter severe problems in getting the software to run fast enough. The best advice which can be offered is that, all other factors being equal (which they never are), one should use the most powerful microprocessor that the system can afford. (Friends have suggested that nothing less than an IBM 370 would satisfy the authors.)