AN ABSTRACT OF THE THESIS OF

CHI-MING YU for the degree of Master of Science in Electrical and Computer Engineering presented on February 27, 1987.

Title: Implementation of a Digital PID Controller in a Hierarchical Distributed Control System

Redacted for privacy

Abstract approved: James H. Herzog

In this is paper, an implementation of a microprocessor based distributed control system (DCS) is investigated. This system consists of a digital PID controller and a high level operator interface. A temperature control system was designed to experimentally verify the system operation. Experimental results compare closely with theoretical prediction. The device performed satisfactorily as a mode of a distributed control system.

# Implementation of a Digital PID Controller in a Hierarchical Distributed Control System

by

Chi-Ming Yu

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed February 27, 1987

Commencement June, 1987

APPROVED:

Redacted for privacy

Associate Professor of Electrical and Computer Engineering
of Major

Redacted for privacy

Head of department of Electrical and Computer Engineering

Redacted for privacy

Dean of Graduate School

Date thesis is presented February 27, 1987

Typed by researcher for Chi-Ming Yu

TABLE OF CONTENTS

LIST OF FIGURES

# Implementation of a Digital PID Controller in a Hierarchical Distributed Control System

## CHAPTER 1

## INTRODUCTION

Because of the rapid advances in microprocessor technology, the microprocessor based distributed control system (DCS) has grown from a technical novelty to a dominant force in industrial automation [1]. In such a system, the control of a plant is distributed to many controllers (see Fig-1.1). Each controller is microprocessor based, and can control a specified process unit independently. A system controller is used to coordinate actions among the distributed controllers. One or more operator interfaces can communicate with these controllers through an appropriate data link. There are several good reasons to account for using a DCS, the most important being [2,3,4] the following:

(1) Lower Cost: A digital controller using microprocessor components is usually less expensive than an analog controller

(2) Increased Capability: A digital controller offers advanced control strategies.

(3) Greater Flexibility: The operator can change the control algorithm and the control parameters at any time as desired.

(4) Data Storage: All the information can be saved in the memory. The information may be helpful for fault diagnosis and analysis of the process.

(5) Friendly Human Interface: It is fairly easy to implement a wide variety of human interfaces.

## 1.1 The Controller

The controller, one node of the DCS, takes inputs from the measuring element and commands from the operator (Fig-1.2). It computes the outputs needed to make the process follow the operator's commands. These control outputs are used by actuators, drives, valves and other devices that regulate variables to be controlled in the plant. There are several commonly use controller modes. A PID ( Proportional, Integral, and Derivative) controller was selected for this investigation because it is most likely to be capable of satisfactory control [5]. The characteristics of the PID algorithm will be developed in chapter 3.

Fig-1.3 shows the component configuration chosen to implement the controller. The PID control algorithm was coded in assembly language and stored in the ROM (read only memory). The RAM (random access memory) is used to store the PID parameters. When the controller is operating, it reads input data from the digital port and A/D converter, and

generates control output. This output is sent to the actuators via the D/A converter or the digital output port.

## 1.2 The Operator Interface

The operator interface often is divided into two types. A low level operator interface (LLOI) allows the operator to interact with the controllers using a direct connection (e.g., through a attached knob). A high level operator interface (HLOI) is a collection of devices that perform functions similar to the LLOI but with increased user friendliness [1]. For example, the HLOI may use a CRT to show the operating condition of the plant.

The operator communicates to the controller via the communication facilities. The general configuration of the HLOI is shown in Fig-1.4. It can be seen that the HLOI has the capability of displaying, printing, saving messages from controller, and receiving commands from a keyboard. Thus the HLOI will let the operator manage and analyze the process.

## 1.3 Objective and Design Approach

Even though a DCS system has many advantages, its implementation may be quite difficult to achieve. The

objective of this project is to design a control node for a digital PID controller in a hierarchical distributed control system. An Intel 8031 microprocessor and accessory chips are used to implement the PID controller and manage the communication interface. The high level operator interface and master controller is an IBM-PC with standard devices. The PID controller acts as a slave device which controls the process in an on-line mode. The IPM-PC acts as a master device which can monitor the process in an off-line mode.

In the next chapter, the overview of the hierarchical DCS is described. The digital PID controller and HLOI are also presented. The PID control algorithm and its considerations in implementing in discrete form are discussed in chapter 3. Chapter 4 presents a temperature monitor and control system to verify the control concept. The ON-OFF and PID control are experimentally verified. The last chapter will give suggestions about the future expansion of this system and a brief conclusion.

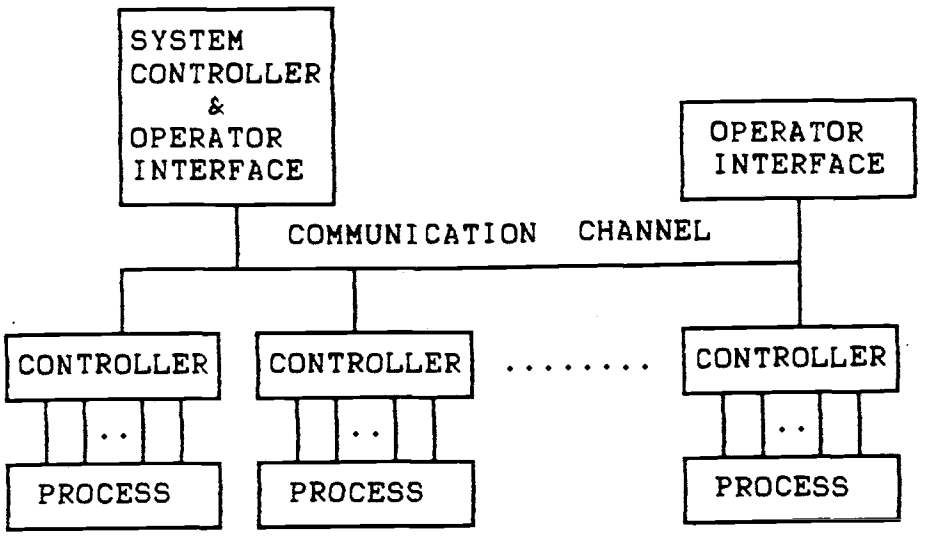



Fig-1.1 The Concept of Distributed Control System

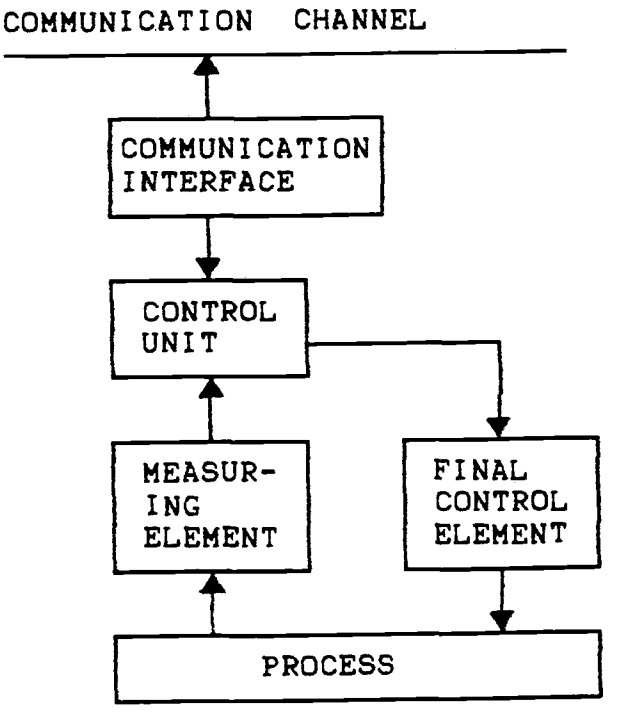



Fig-1.2 A Control Node in a DCS

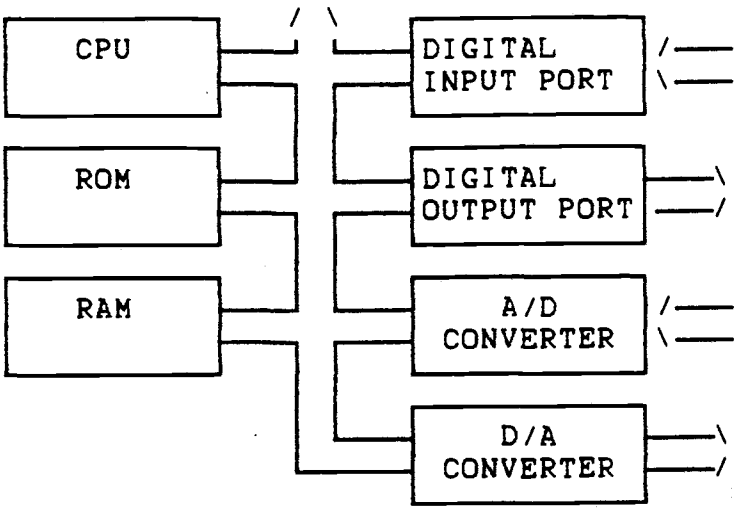



Fig-1.3 The PID Controller Configuration

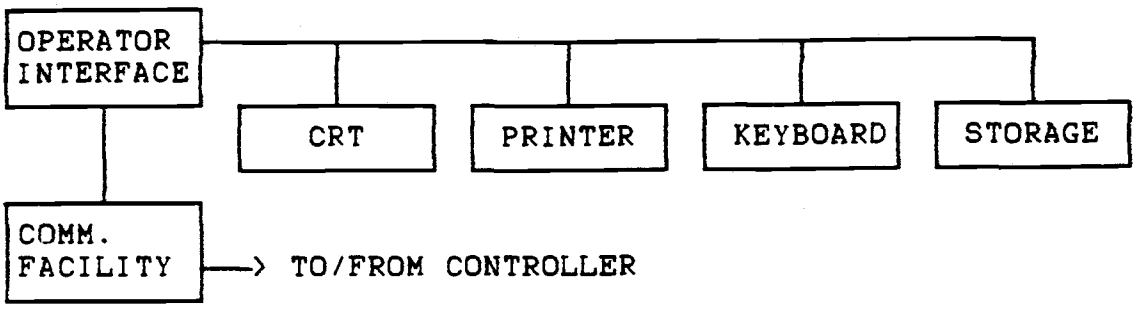



Fig-1.4 The HLOI Configuration

# CHAPTER 2

# SYSTEM DESCRIPTION

In this chapter, the Intel 8031 microprocessor is reviewed. The hardware and software of the HLOI and the controller are discussed. Some design issues of the data transfer between devices, and the interrupt handling are also described.

## 2.1 Intel 8031 microprocessor

The Intel 8031 is a high performance single-chip microprocessor intended for use in sophisticated real time applications [10]. It is a control oriented CPU which can address 64K-bytes of external program memory in addition to 64K-bytes of external data memory. Its internal memory address consists of 128-bytes data RAM. Its instruction set includes integer multiplication/division and provision for table look-up.

The 8031 has 4 I/O ports. Port 0 provides the multiplexed low-order address and data bus. Port 2 provides the high-order address bus. Port 3 provides the external interrupt request inputs, counter inputs, a full duplex serial port and the control signals for the external data memory. Port 1 can be used as 8 bit parallel I/O. The block

diagram of the 8031 is shown in Fig-2.1.

The 8031 services interrupt requests from five sources; two from external interrupt requests , two from internal counter/timers, and one from the serial I/O port. Each of the five sources can be assigned to either of two priority levels and can be independently enabled and disabled.

The 8031 uses two 16-bit counter/timers for measuring time intervals, counting events and generating periodic interrupt requests. Each can be programmed independently to operate as a 16 bit counter/timers or an 8 bit counter/timers with automatic reload upon overflow.

## 2.2 System Overview

### 2.2.1 System Architecture

Fig-2.2 shows how the system is arranged. The IBM-PC and peripheral devices function as the high level operator interface. It is responsible for process monitoring, data storage, alarm reporting and receiving commands from the keyboard.

An asynchronous communication conforming to the EIA RS232-C standard is used for interface between the IBM-PC

and the controller. This standard defines the DC voltage level that are either greater than +3 volts or less than -3 volts. The IBM-PC provides a asynchronous communication adapter (COM 1) and a BASIC statement (OPEN COM) to support the serial data transfer. The 8031 on chip microprocessor's serial port provides a full duplex communication channel with the DC voltages level +5 volts and 0 volts. A transmitter and a receiver were used in the controller to convert the signal level to match the standard RS232-C (Fig-2.7).

The controller provides the hardware interfacing to the process and software program to control the process. A 2K ROM (2716) is used for the external program memory. It contains the operating system and the control algorithms. The external 256 byte RAM/IO (8155) is used to handle digital inputs and outputs. For analog inputs and outputs, an 8 bit A/D (analog to digital) converter (ADC0804) and an 8 bit D/A ( digital to analog ) converter (AD558) are used. The A/D converter converts 0-5 Volt (input range) to 00-FFH (output range) [12]. The D/A converter converts 00-FFH (input range) to 0-10 Volt (output range)[13]. Several LEDs are used for the local display and system diagnosis.

### 2.2.2 System Software

System software operation is shown in the flow chart in Fig-2.3. The controller alternates between executing the assigned control action and checking for the new command. After executing the control action, the controller sends the result back to the HLOI. When a new command from the HLOI is received, the controller will interpret it then modify the control action.

The HLOI alternates between receiving and displaying the responses from the controller and checking for new operator input commands. If a new command is keyed, the HLOI will transmit appropriate information to the controller.

The HLOI and the controller act in a master-slave relationship, but are functionally independent. The software in the HLOI is written in the BASIC language. The operating system and application program in the controller is written in assembly language.

## 2.3 High Level Operator Interface

### 2.3.1 User Response and Command Structure

Fig-2.4 shows that the BASIC program is written in a loop structure so the keyboard is checked repeatedly. If the operator gives a new command, the program will execute a

corresponding subroutine and generate an appropriate action. There are three command groups provided by the HLOI:

(1) The mode control command.

(2) The direct control command.

(3) The device control command.

The mode control command allows the user to assign a control strategy to the controller (e.g., on-off control or PID control). The direct control command is used for directly controlling the controller for actions, such as alarm reset. The third type is used for managing peripheral devices for data storage or printing.

All commands are composed of ASCII characters, and the "<" and ">" symbols are used as a delimiters. The form of the command packet is

<K A1 A2 A3...>

< & >: Delimiters. "<" stands for the beginning of the packet and ">" stands for the end of the packet.

K: Key word of the packet. It is a single ASCII character that specifies the control function.

Ai: Arguments of the packet. Each argument is a parameter for the specified control function. It is either an integer or a real number. More than one argument can be assigned.

The function of the space is to delimite the key word and the arguments. But spaces are ignored when they are received by the controller. Different control actions have different command packet since they may have different numbers of parameters. A maximum of 16 characters can be assigned to the packet excluding the delimiters. It should be pointed out that the command packet is constructed by the BASIC program in the HLOI. The user need not worry about the format of the command packet. The user need only respond to instructions provided by the program.

### 2.3.2 Controller Response

The response subroutine performs four functions; data receiving, data storage, data displaying and alarm report printing (see Fig-2.5). After the keyboard is checked for user input, the serial port buffer (IBM-PC) is then checked for response data. If data has been received from the controller, the HLOI will convert it to the appropriate format for displaying and storage. If necessary, an alarm report will be displayed or printed.

## 2.4 PID Controller

The controller architecture is arranged as in Fig-2.6. A 3-8 decoder is used for generating the specific chip

select to select the ROM, RAM, A/D or D/A chips. The controller has two possible sources of data inputs; the A/D converter and the serial port. Fig-2.7 shows the main loop of the software in the controller. The data sampling routine samples the A/D converter data. The command service routine executes the command packet received from the serial port.

### 2.4.1 Data sampling and Data Output

Operation of the A/D converter depends on the sample time period. The 8031 has two internal timers. Timer 1 is used for the baud generator of the serial port. Timer 0 is used to generate the sample time period. Since timer 0 can count at the rate of 1/12 of the crystal frequency and the 16 bit timer register can be assigned any value between 0-65534, a wide range of the sample time periods can be generated. An interval of 2 seconds was selected as appropriate for the temperature control system.

When it is time to sample, a data sampling subroutine starts (see Fig-2.8), and the 8031 microprocessor instructs the A/D converter to begin the data conversion. The conversion time of the A/D converter is about 110 micro seconds [12], a short time in relation to the sample time period. After completing the conversion, the A/D converter will generate an interrupt to inform the 8031. The 8031 then

reads the A/D converter value and saves it for future use.

After a data sampling, the controller executes the control algorithm to compute the control output. This output is then sent to the D/A converter to drive the final control element.

### 2.4.2 Serial Port Interrupt and Command Packet

The command service routine shown in Fig-2.7 includes three major functions:

(1) Serial port interrupt handling

(2) Command packet conversion

(3) Command execution.

The serial port interrupt service routine handles the receiving of the command packet. The packet conversion routine converts the command packet to a suitable format. The command execution routine modifies the control action and associated parameters.

The 8031 uses a common serial port interrupt to handle both the data receiving and transmitting. Data transmitting is handled easily under program control. Data receiving can not be predicted and thus makes effective use of the serial port interrupt. The serial port handling procedure is shown in Fig-2.9. Command packet reception begins when the symbol

"<" is encountered. Each Character is temporarily stored in internal memory in sequential order. When the symbol ">" is encountered, a flag is set to indicate that a new command packet has been received. The command packet conversion routine then starts at next loop cycle.

The received characters are in the ACSII code format. The command packet must be converted from ASCII code to the hexadecimal (HEX) format. As soon as the conversion is completed, the command execution starts. Then, a new control action is assigned and the associated parameters are revised.

## 2.5 Interrupt Handling Consideration

In this design, three interrupts are used in the controller. The serial port interrupt is used for data transfer between the controller and the HLOI. The Timer 0 interrupt (TIMER 0) is used for generating the data sampling intervals. The External interrupt 0 (EXT 0) is used for checking the completion of the A/D conversion. Problems may exist between the interrupts. For example, a command packet character may be lost if the serial port interrupt is ignored. The sample rate may not be a constant if the TIMER 0 interrupt is ignored. Data being processed may be changed by another interrupting process. To minimize these problems;

(1) The serial port is assigned the highest priority.

(2) Computational routines are as short as possible.

(3) Information flags are extensively used.

These flags are set and reset corresponding to a specified condition. Flags are checked immediately upon completion of a subroutine. Then, the delayed (but unaffected) subroutine can be executed as soon as possible according to the condition of the flag. The detailed flag assignment is given in Appendix B.

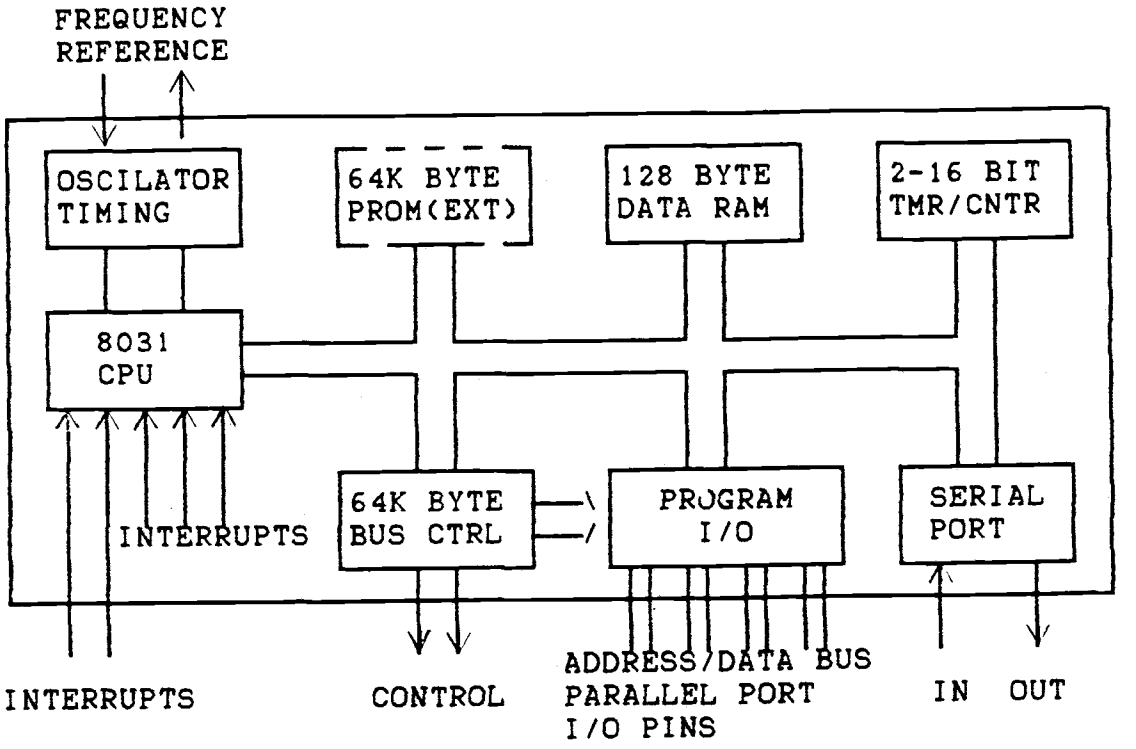



Fig-2.1 Block-Diagram of the 8031
(Courtesy of Intel)

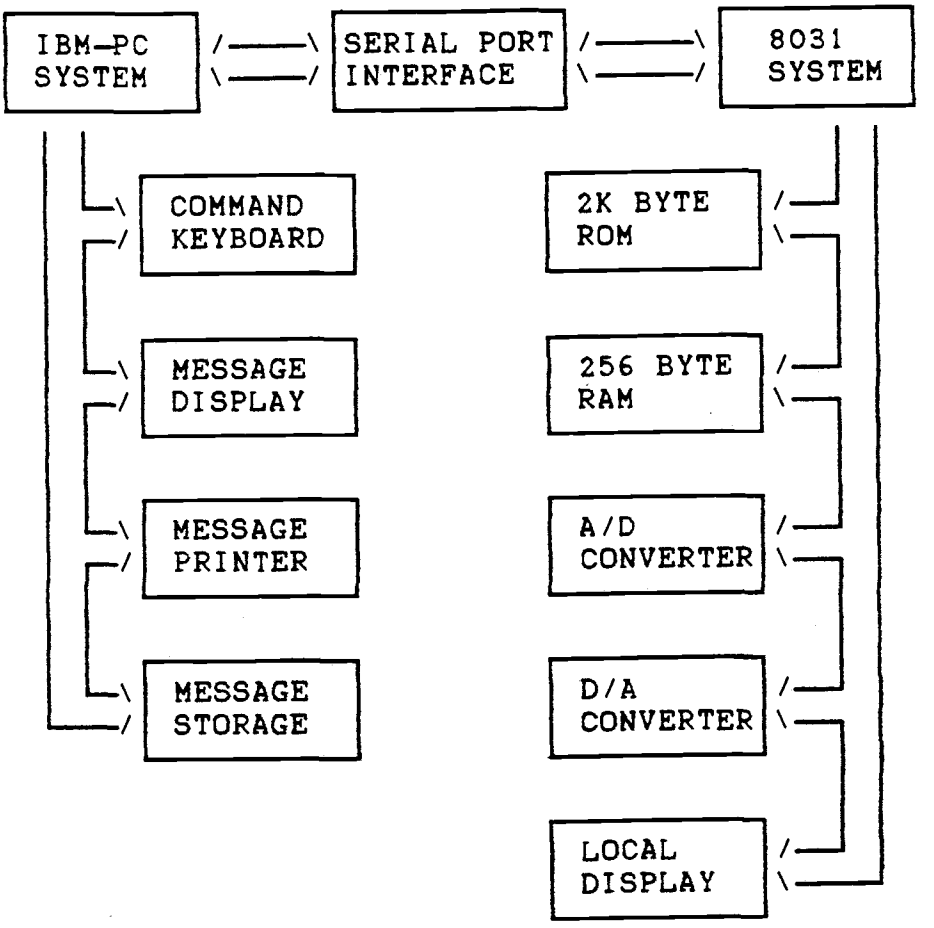



Fig-2.2 Hadware Structure of the Hierarchical DCS

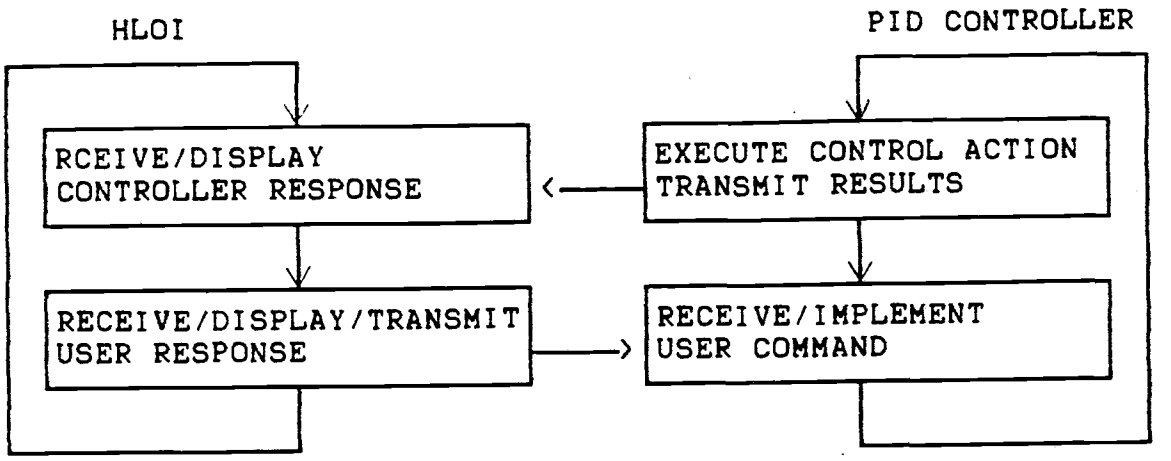



Fig-2.3 Software Structure of the Hierarchical DCS

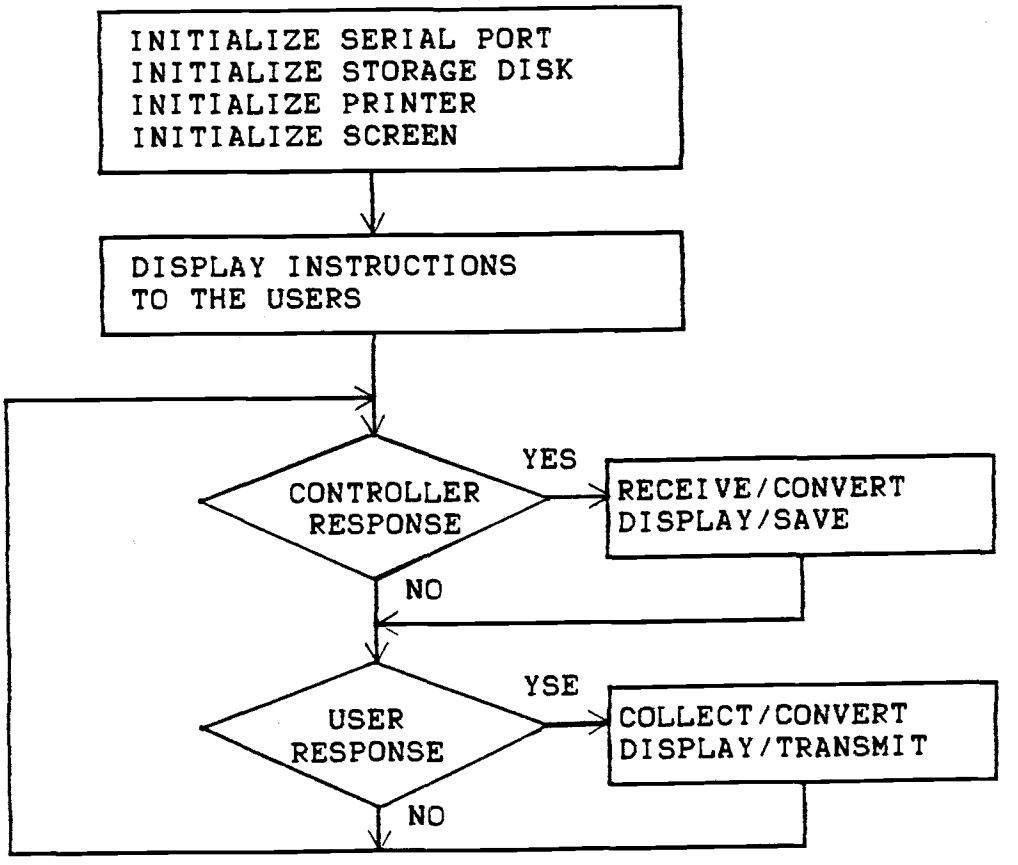



Fig-2.4 Main Loop of HLOI

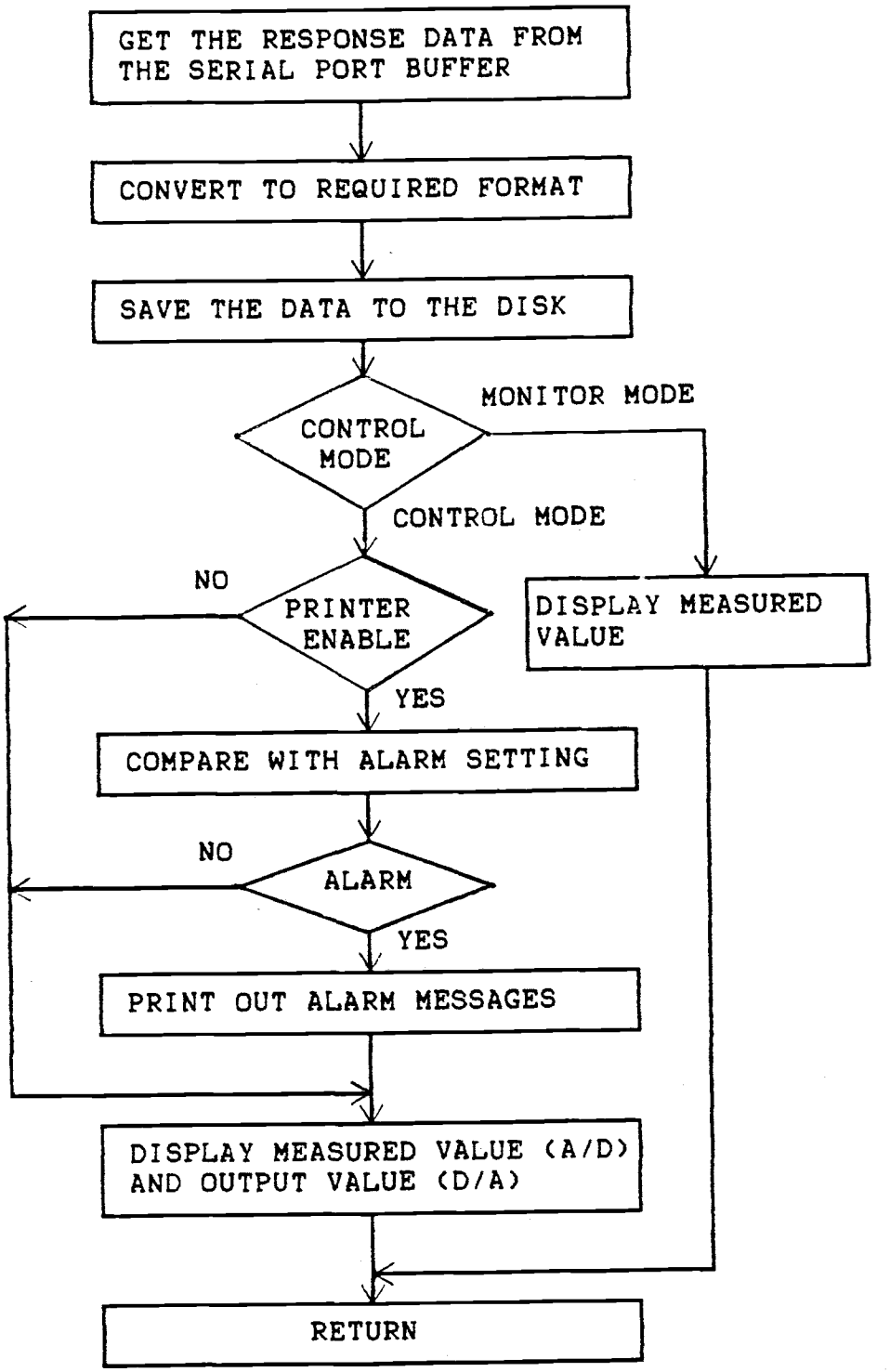



Fig-2.5 Response Handling in HLOI

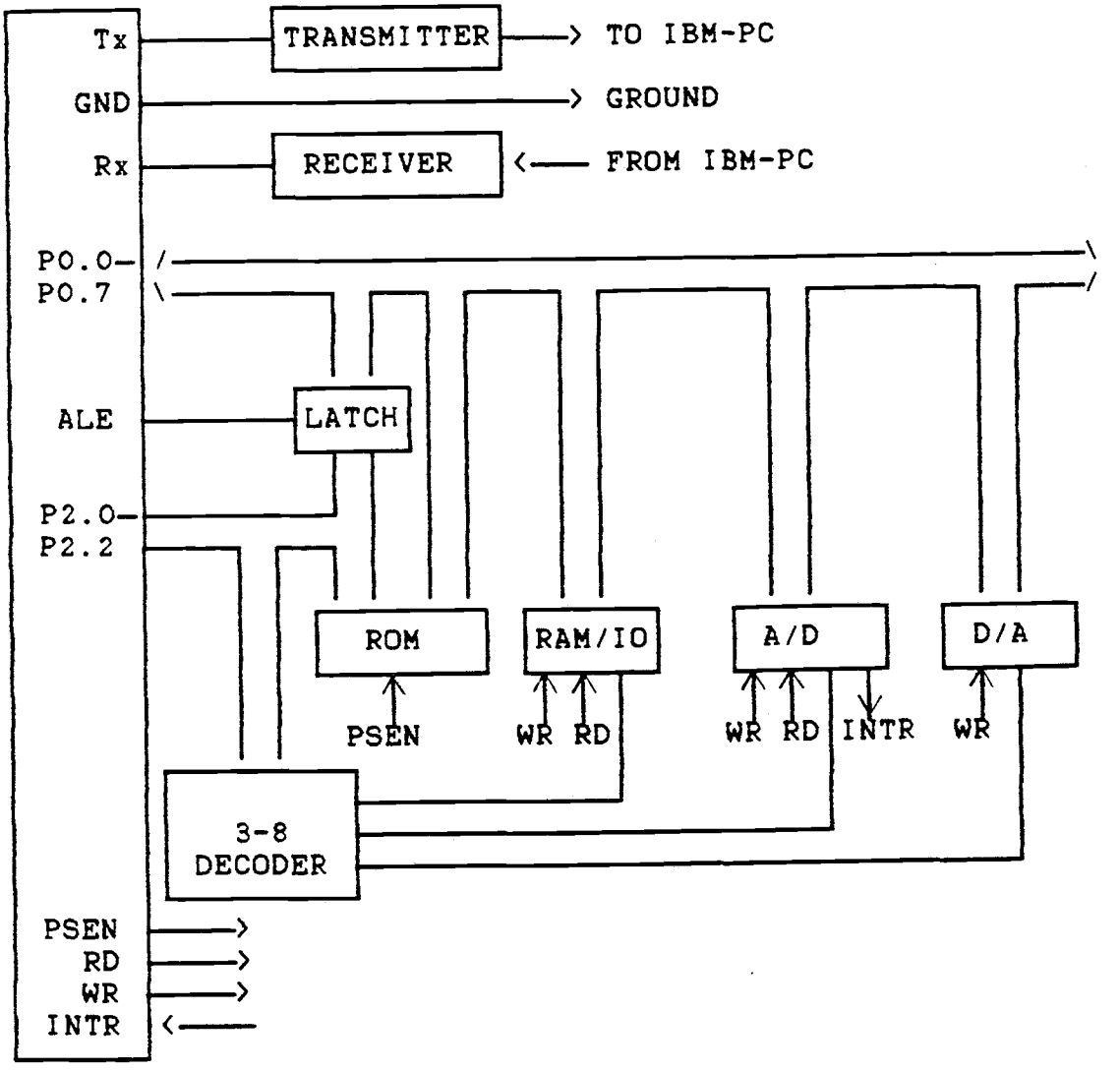



Fig-2.6 Hardware Structure of PID Controller

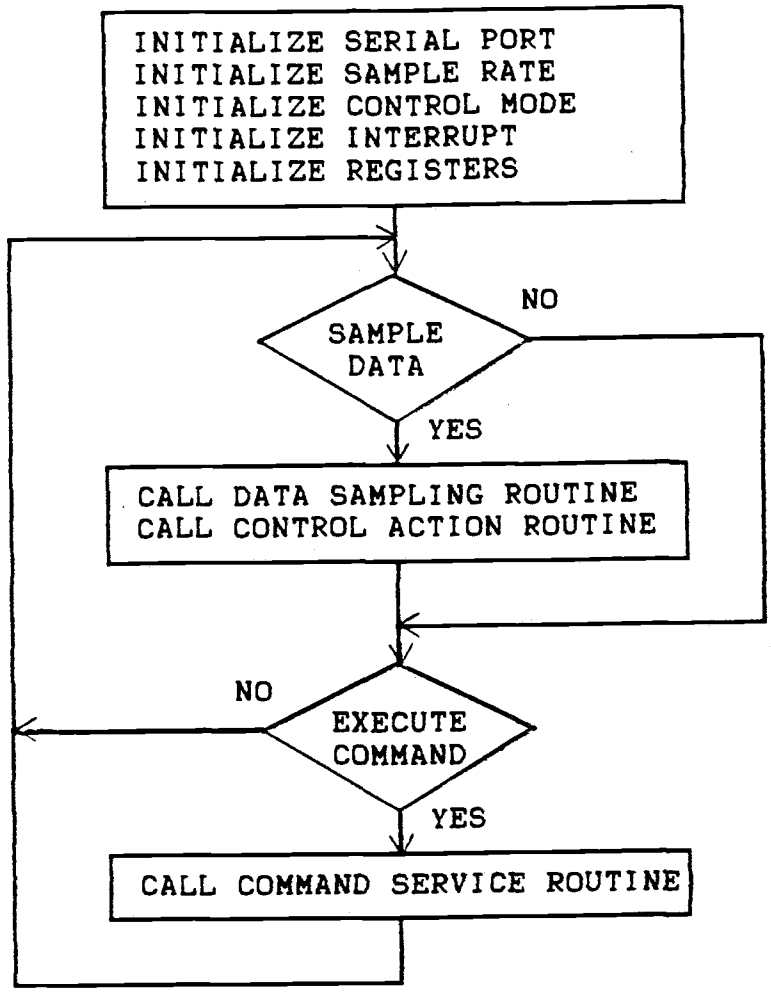



Fig-2.7 Main Loop of PID Controller

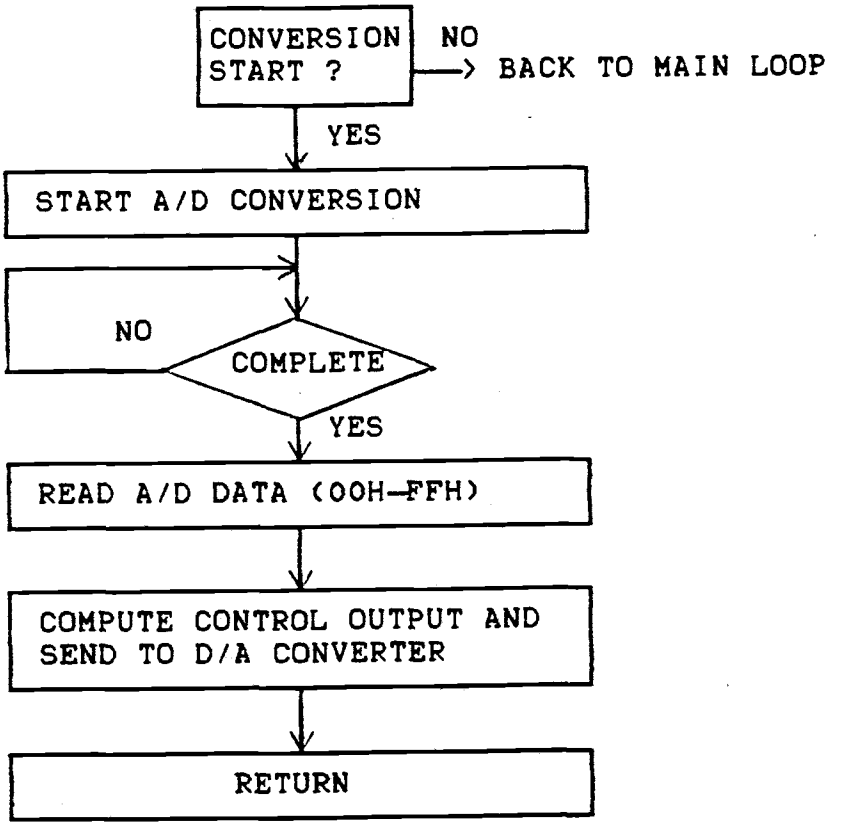



Fig-2.8 Data Sampling and Data Output

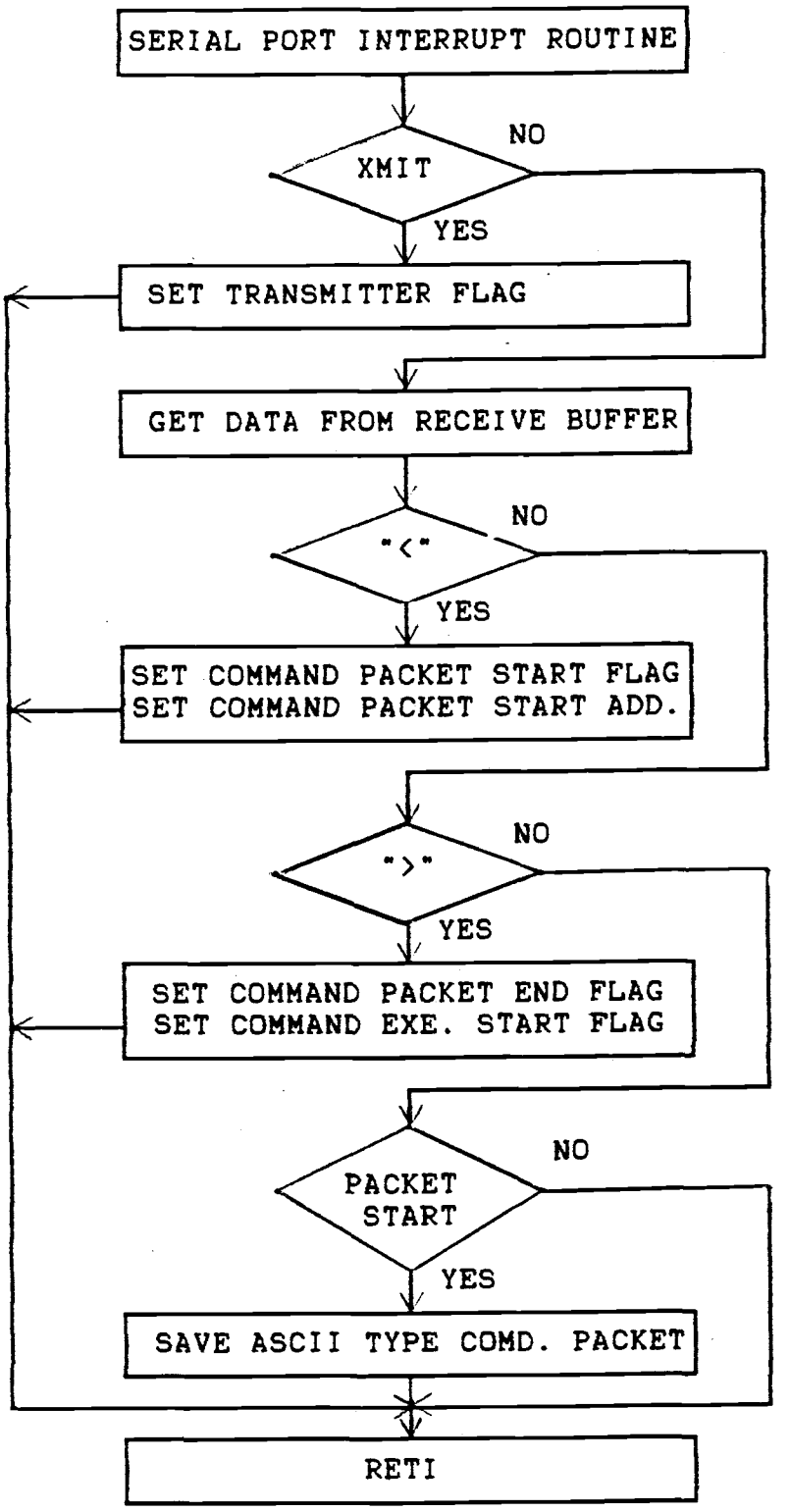



Fig-2.9 Serial Port Handling in PID Controller

# CHAPTER 3

# DIGITAL CONTROL ALGORITHM

In this chapter, some concepts and definitions of the control algorithm are reviewed. The discrete form of the ON-OFF and the PID control algorithms are then investigated. There are some problems in attempting to perform continuous control using a discrete representation. These problems will be considered and overcome.

## 3.1 Review of Analog Control Algorithm

The task of a PID controller [5] is to adjust the state of a process, the process variable (PV), to conform to a particular value called the set point (SP). The difference between PV and SP is termed the error (ER). It can be expressed as

$$ER = PV - SP$$

The PV should be measured first by a measuring element then transmitted to the controller. The SP value of the controller is assigned by the operator. The controller calculates the difference (ER) to produce the output signal (OV) to the final control element which acts on the process. This concept is shown in the block diagram in Fig-3.1.

### 3.1.1 ON-OFF control mode

If the controller can only generate two states, the maximum and the minimum output, the system is called an ON-OFF or BANG BANG control system. This is the simplest action of the controller. The control law of this can be expressed as

$$ER = PV - SP \qquad 3.1.a$$

$$OV = F( \text{ sign of } ER ) \qquad 3.1.b$$

A positive sign of the error means the output value is the maximum. The negative means the minimum. The ER can be any suitable unit, but the units of SP, PV and ER must all be the same.

An example of the ON-OFF control is the home furnace. When a lower temperature is sensed at the thermostat, the power switch is closed, the furnace operates and the temperature rises. When the temperature at the thermostat goes high enough, the power switch opens, shutting down the furnace. Continuous cycling is the characteristic of the ON-OFF control.

### 3.1.2 Proportional Control Mode

Proportional (P) control avoids the cycling associated with the ON-OFF control mode. It produces an output signal which is proportional to the error signal. It can be

expressed as

$$OV = KP * ER + OV0 \qquad 3.2$$

where

KP: constant of proportional gain.

OV0: constant value at error = 0

The value of KP can be changed by the operator. Small values of KP result in a small correcting signals, while large values result in large correcting signals.

The OV0 is the value of output when the error is zero. In most conditions, this value can be adjusted to obtain a required output value when the control system is at steady state. The correcting signal of the proportional function is carried out around this particular value. If a load changes, the initial value must be revised. Otherwise, an off-set error will exist. This is a disadvantage of the P control mode.

### 3.1.3 PI Control Mode

The P control mode may lead to a steady state error. One way to overcome this automatically is by using the proportional-integral (PI) control action [5]. In this mode, the controller will continuously take control action as long as an error exists. Thus, the offset error is eliminated. This mode of control is described by the equation

$$OV = KP * ER + KI * \int_0^t ER\, dt \qquad 3.3$$

where

KI: constant of integral gain

In this case, the integral term is added to the proportional term and replaces the initial value VOO. A positive error results in an increasing correcting signal.

### 3.1.4 PID control mode

With a P or PI mode control action, a substantial amount of time may elapse before the process returns to the zero error condition. The derivative (D) mode control action is added to the P or PI mode to improve this condition [5]. The expression of the PID mode mathematically is

$$OV = KP * ER + KI * \int_0^t ER\, dt + KD * \frac{dER}{dt} \qquad 3.4$$

where

KD: constant of derivative gain

The effect of the derivative term is to anticipate the linear change in error by adding an additional output. The correcting action is proportional to the rate of change of error. Thus the derivative term has the effect of damping the error and stabilizing the system performance.

## 3.2 Digital Control Algorithm

### 3.2.1 ON-OFF Control Mode

The ON-OFF control system uses one process variable (PV), one setting value (SP) and one output variable (OV). However, in a real system, two set points and two alarm setting values are needed. These two alarm setting value are required to generate an alarm when the process exceeds normal bounds. Using the magnitude constraints of the selected components, the equation of the ON-OFF control mode including the alarm signal in digital format can be represented as

$$\begin{aligned} OV = {} & FFH \;;\; PV < SL \text{ (low set-point)} \\ & 00H \;;\; PV > SH \text{ (high set-point)} \\ alarm : {} & ON \;;\; PV > AH \text{ (high alarm setting)} \\ & \quad ;\; PV < AL \text{ (low alarm setting)} \\ & OFF \;;\; \text{otherwise} \end{aligned} \qquad 3.5$$

The general flow chart of the ON-OFF control in discrete form is shown in Fig-3.2. Note that the process variable may actually be continuous but the output value can only be in two states of maximum or minimum.

### 3.2.2 PID Control Mode

PID control action uses a summation of three terms: one is directly proportional to the error, one is dependent on

the error accumulation, and one is dependent on the present rate of change of the error. The equation is

$$OV = KP*ER + KI * \int_0^t ER\ dt + KD * \frac{dER}{dt} \qquad 3.6$$

For a digital controller, methods must be found to compute the integral and derivative terms. From the viewpoint of simplicity, the integral term is approximated by rectangular integration and the derivative term by the difference operation [20].

A derivative is defined as the rate at which a quantity (ER) is changing at instant in time. An approximation can be achieved by calculating only the rate at which it is changing over a sample period [19]. In terms of an equation, this is

$$\frac{dE}{dt} \approx \frac{ERi - ERi-1}{TP} \qquad 3.7.a$$

where

TP: Sample Time Period

i: The present time

For the rectangular approximation [15], the height of the rectangle is the error (ER) and the width is the sample time period (TP). In each time period, if the ER is positive, then the product ER * TP , is also positive. For the same reason, the negative error will produce a negative

product. The integration is the summation of all the products. The discrete representation of integration then can be expressed as

$$ARi = \sum_{j=0}^{i} ERj * TP \qquad 3.7.b$$

Eq-3.7.a and Eq-3.7.b, Eq-3.6 becomes

$$OVi = KP*ERi + KI \sum_{j=0}^{i} ERj *TP + KD/TP*(ERi-ERi-1) \qquad 3.8$$

however, we know that ER = PV - SP, so this equation can be converted to

$$OVi = KP*ERi + KI \sum_{j=0}^{i} ERj *TP + KD/TP*(PVi-PVi-1) \qquad 3.9$$

KI, KD and TP are constants. Let us define

$$PIT = KI * TP$$

$$PDT = KD * TP$$

and

$$ARi = \sum_{j=0}^{i} ERj$$

$$SLi = (PVi-PVi-1)$$

then Eq-3.9 becomes

$$OV = (KP*ER) + (PIT*AR) + (PDT*SL) \qquad 3.10$$

The flow chart of this control equation is shown in Fig-3.3.

## 3.3 Software Consideration

In this section, some general conditions for writing

the software program are considered. These conditions include the parameter format and computation polarity.

### 3.3.1 Parameter Format

For the practical point of view, most controllers have a proportional gain with a scale of approximately 0.1 to 10. And the tuning coefficient for the integral and derivative mode has the range varied from less than one minute to a maximum of about 30 minutes [21]. A number format (XX.XXH) is selected for the KP, PIT, and PDT as appropriate for the real number requirement. The process variables OV, ER, and SL are assigned with an integer format XXH (H means hexadecimal) because of component constraint.

Considering a condition of a small PIT number (e.g., 00.10H) and a 8 bit value AR (XXH), the product of PIT*AR will never reach the maximum value (FFH) even though AR equals FFH. This means that no matter how large and how long the error accumulation is, the output still is maintained at a small value in the integration. This is not a practical result for the control strategy. To solve this problem, a 16 bit integer value (XXXXH) is assigned to the AR.

### 3.3.2 Computation Polarity

As mentioned in section 3.1.3, the integral term PIT*AR is the initial value in the PID control algorithm. Since the PIT is a positive constant, the AR should be always a positive value. The ER and SL, however, may have the negative value. This means the ER (SL) will be the 2's complement format if it is negative. It is inconvenient to use this representation in subsequent computations required to produce the output. Therefore it should be converted to a unsigned value and the polarity remembered by a flag. An example of negative ER conversion is shown in Fig-3.4.

According to Eq-3.10, if ER (SL) is positive, the KP*ER (PDT*SL) is added to the PIT*AR. But, this computation shows a positive feedback. This should be avoided in a process control. So, the reverse action is introduced. This means that a positive ER decreases the output.

To compute the PID control output, the sign of ER combined with flag R/D (reverse or direct) determines whether KP*ER is added to or subtract from PIT*AR. The result of this computation will be saved as a temporary control output. The sign of SL combined with flag R/D determine whether the SL*PDT is added to or subtracted from the temporary control output. The result of this computation is the final control output.

The software program in Appendix D shows all algorithms stored in the ROM in the form of subroutines. A specified control action is accomplished by executing group of subroutines. The user can change the control action to any mode at any time without shuting down the system which would be very difficult with a traditional controller.

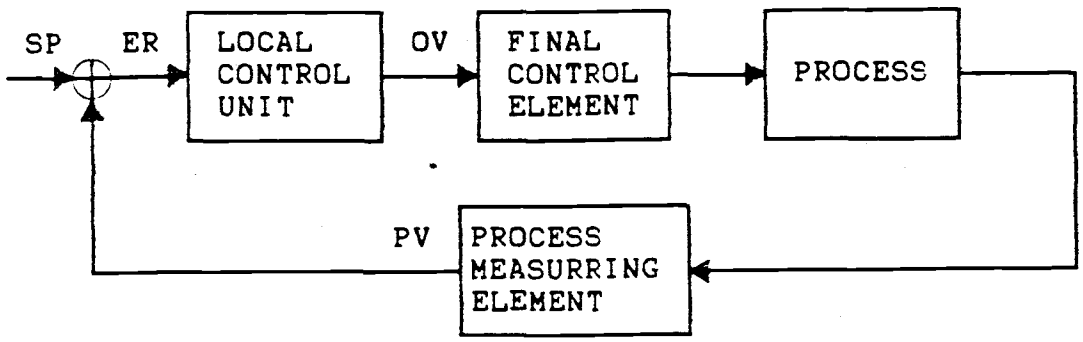



Fig-3.1 Concept of the Process Control

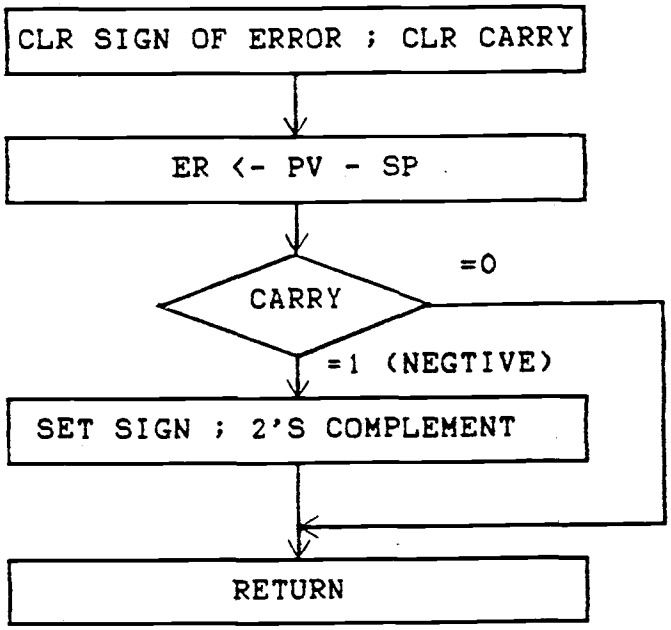



Fig-3.2 The Polarity Consideration of Error

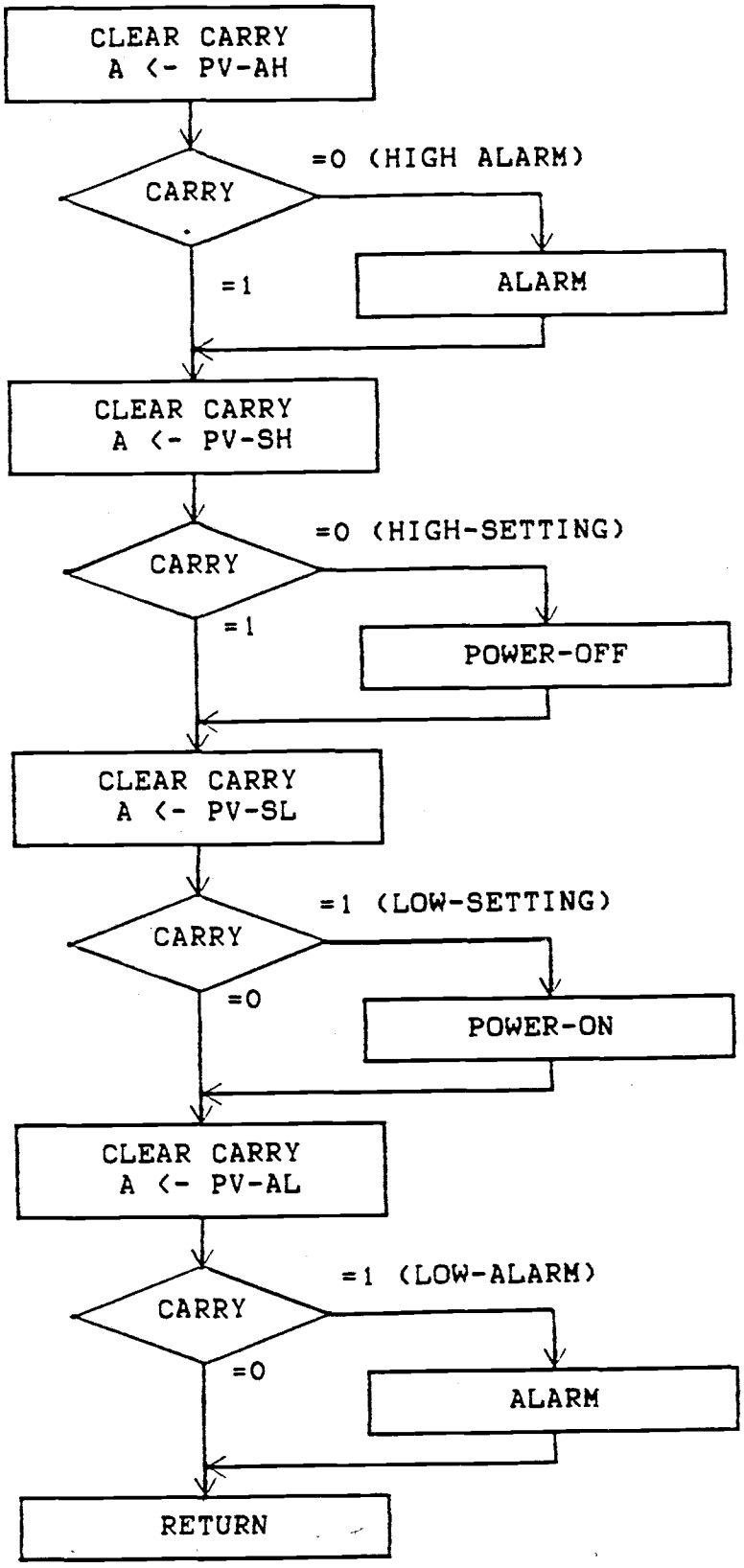



Fig-3.3 The Flow Chart of ON-OFF Control Algorithm

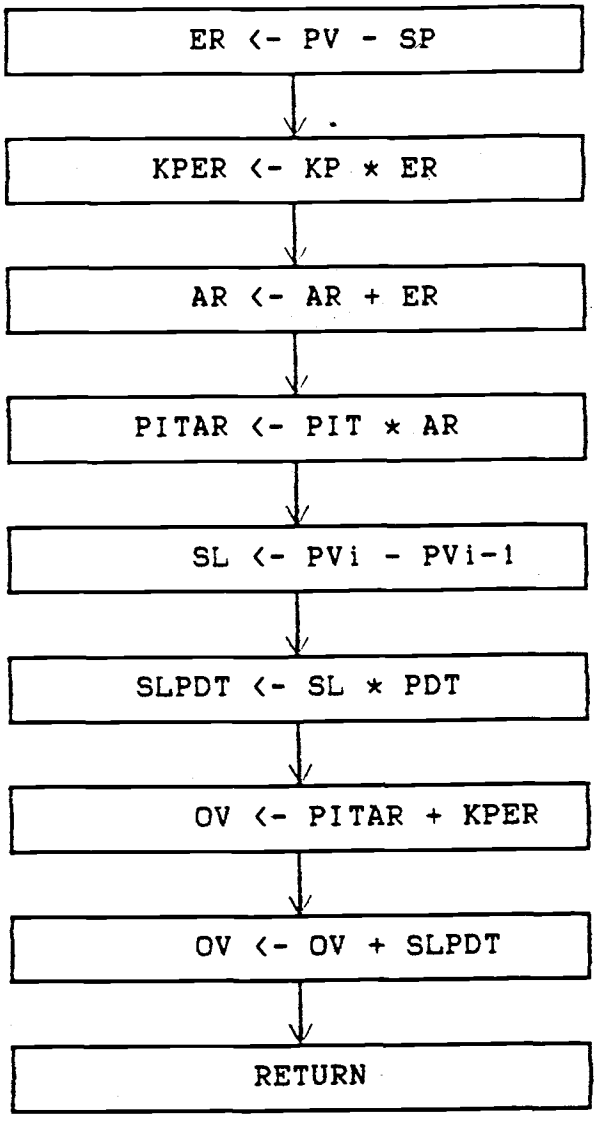



Fig-3.4 The Flow Chart of the PID Control Algorithm

# CHAPTER 4

# TEMPERATURE CONTROL SYSTEM

A hierarchical distributed control system has been designed. A temperature control system was implemented to verify system operation. The block diagram in Fig-4.1 shows that two elements are needed to implement the temperature control, a temperature measurement element and the final control element.

## 4.1 Temperature Measurement

To sense the temperature, an Analog Devices' AD590 temperature sensor was used. This integrated circuit temperature transducer produces an output current proportional to the absolute temperature. The 1μA/°K output current flows through a current-voltage converter to develop an output voltage with the units of 1 mV/°K (see Fig-4.2). This voltage value is sent to the negative input of the difference amplifier. The output of an AD580 2.5 volt reference is divided down by resistors to provide a 273.2 mV offset. It is then sent to the positive input of the difference amplifier. Thus, the output of the difference amplifier generates a voltage with the units of mV/°C.

Because the AD590 is limited to applications below +150

degree Celsius [12], a range of 0-100 degree was selected. This means that a range of 0-100 mV exists at the output of the difference amplifier. The input range of the A/D converter is 0-5 Volt. In order to match these two ranges, the output signal of the difference amplifier was amplified by a voltage amplifier with the gain of 50. With the voltage amplifier and the difference amplifier, the temperature measurement had the units of 100mV/°C with the base 0 degree Celsius. The detailed circuit is shown in Appendix C.

After a sensed temperature voltage is converted to a HEX format by the A/D converter, the controller uses a look up table to convert the HEX format data to the value with the unit of Fahrenheit. This temperature value is then used for local display. The look up table conversion method is used because it is easily implemented and fast.

## 4.2 Final Control Element

In this design, a small electric light bulb was used to heat the region around the temperature transducer. The D/A converter (AD558) has an output drive capacity of 5 mA [13]. This output current capability is not enough to turn on the light bulb, so a current driving circuit diagram shown in Fig-4.3 was used. The voltage to current converter produces an output current proportional to the input voltage from D/A

converter. The current amplifier provides the needed current for the lamp. The detailed driving circuit is shown in the Appendix C.

## 4.3 System Verification

This temperature control system is tested with various control situations. The test results are shown in Fig-4.4 to Fig-4.8. To compare these figures with the characteristics of the control algorithm discussed in the chapter 3, it can be seen that:

(1) The ON-OFF control mode has the cycling phenomenon (Fig-4.4).

(2) The P control mode can eliminate the cycling problem but it has the off-set phenomenon (Fig-4.5). This error can be eliminated by adjusting the initial value manually (Fig-4.6).

(3) The PI control can overcome the off-set problem automatically (Fig-4.7) but overshoot and oscilation are induced.

(4) The response curve is improved when the PID control mode is used (Fig-4.8). The offset is eliminated by the integral action while the derivative action serves to eliminate some of the oscilation.

It should be pointed out that the sensed temperature

fluctuates randomly. This is because the A/D converter being used only has an accuracy of plus-minus one bit. Each time the sensed temperature voltage is converted to the HEX format, the temperature value may have about plus-minus one degree Fahrenheit error.

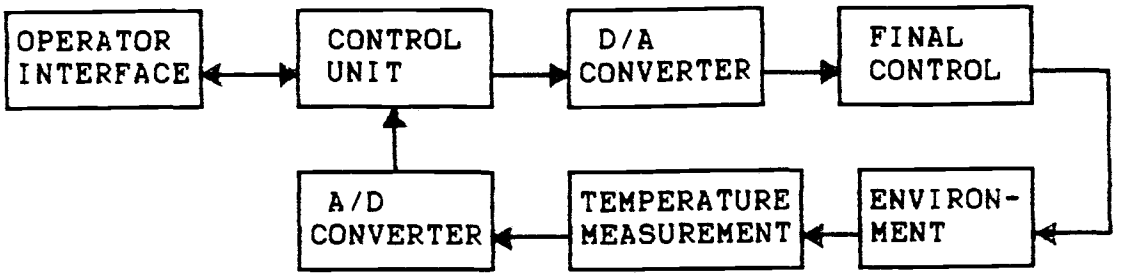



Fig-4.1 The Temperature Control System

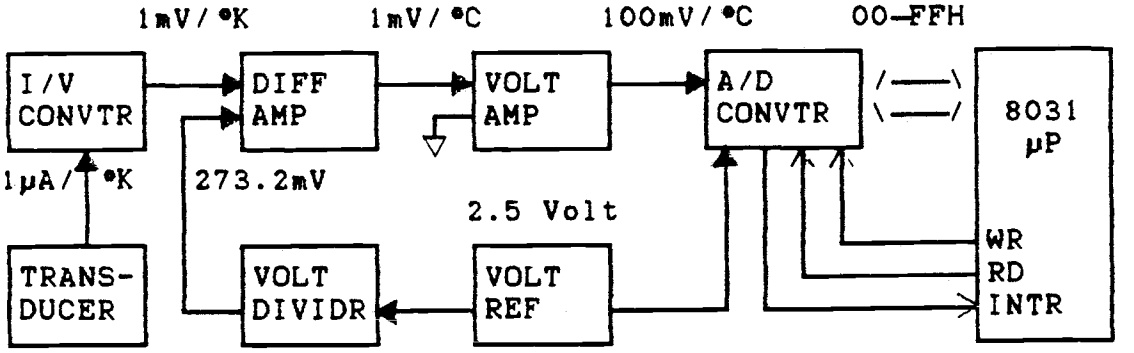



Fig-4.2 The Temperature Measurement

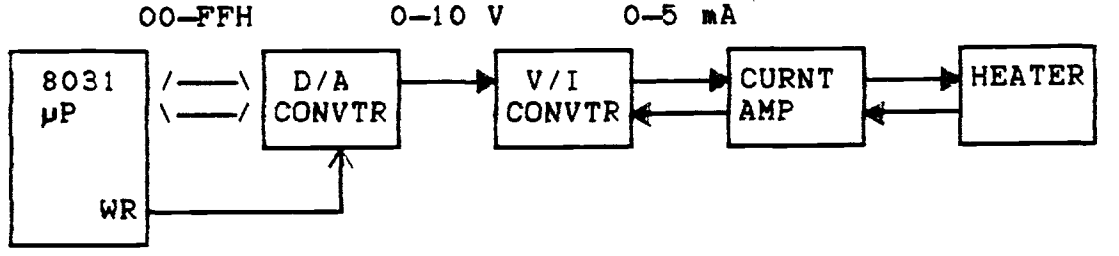



Fig-4.3 The Final Control Element

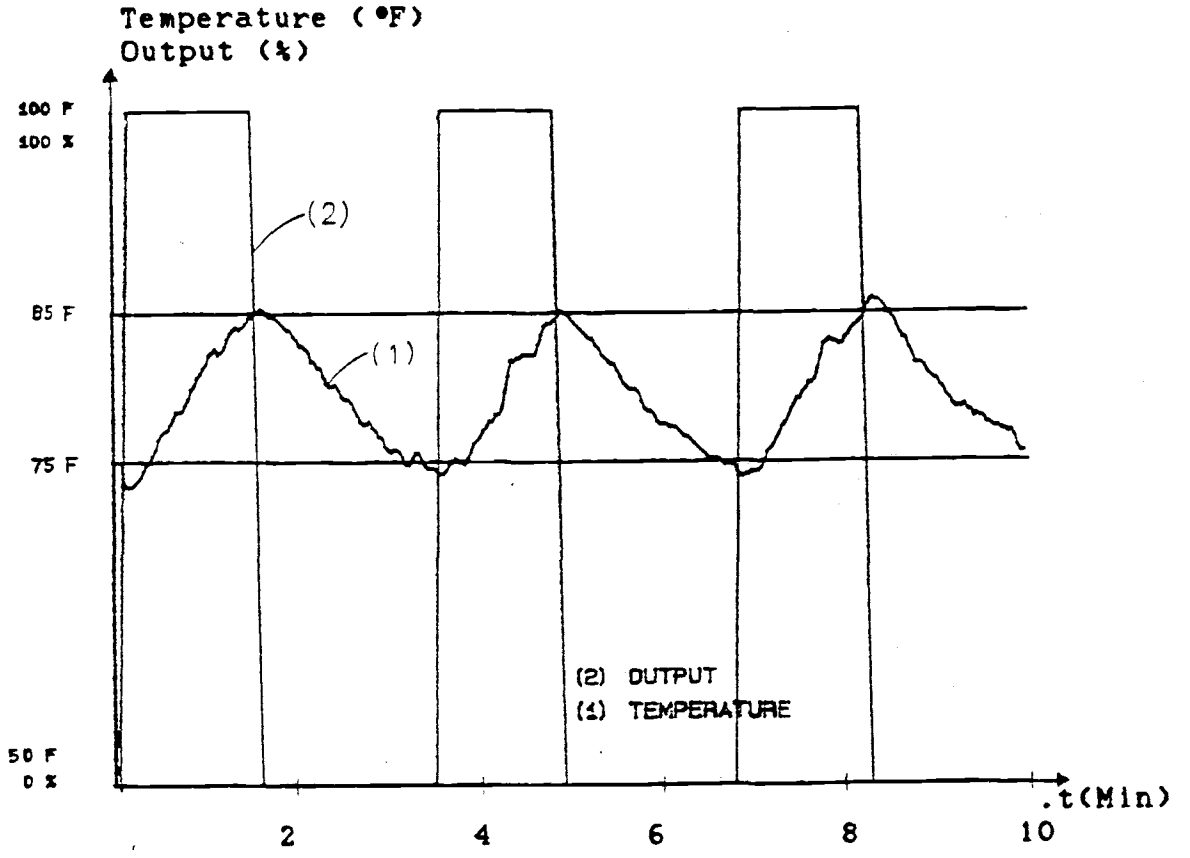



Fig-4.4 Temperature and Output Response in ON-OFF Control Mode

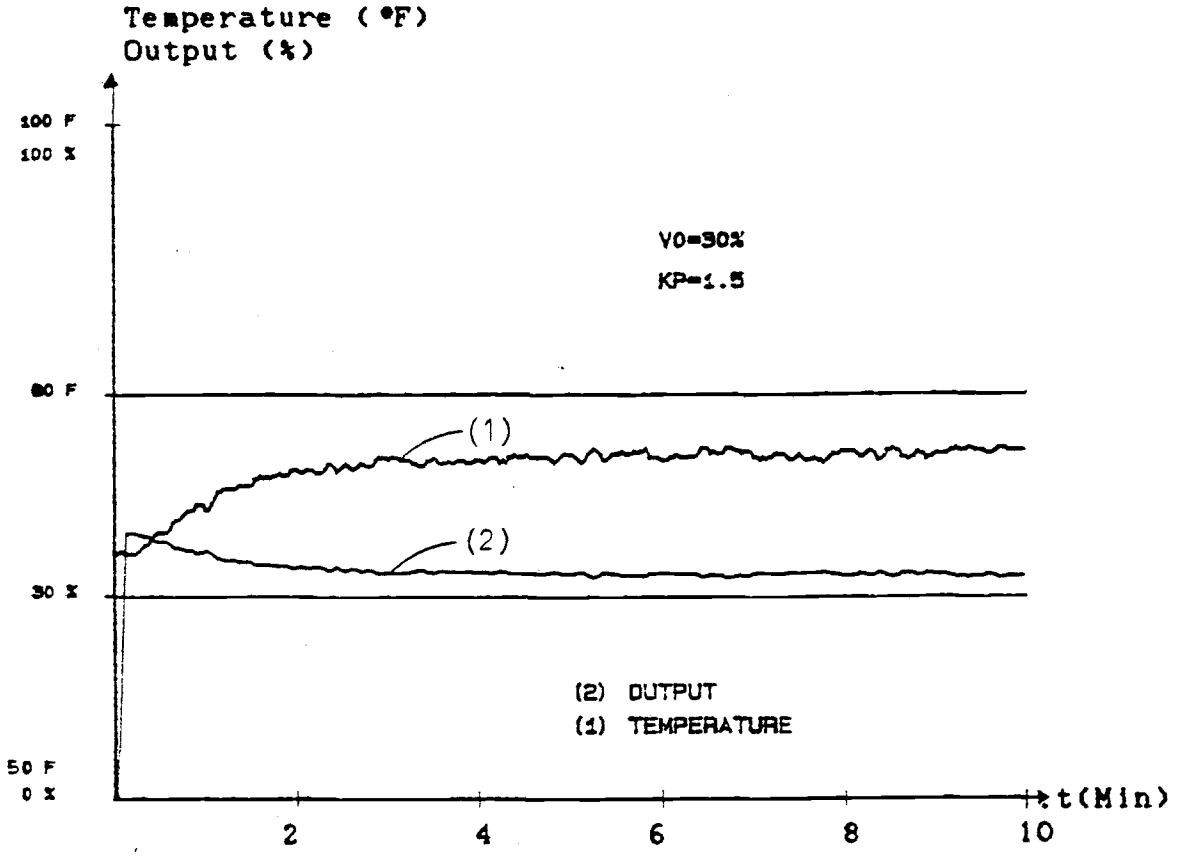



Fig-4.5 Temperature and Output Response in P Control Mode with 30% Initial Output

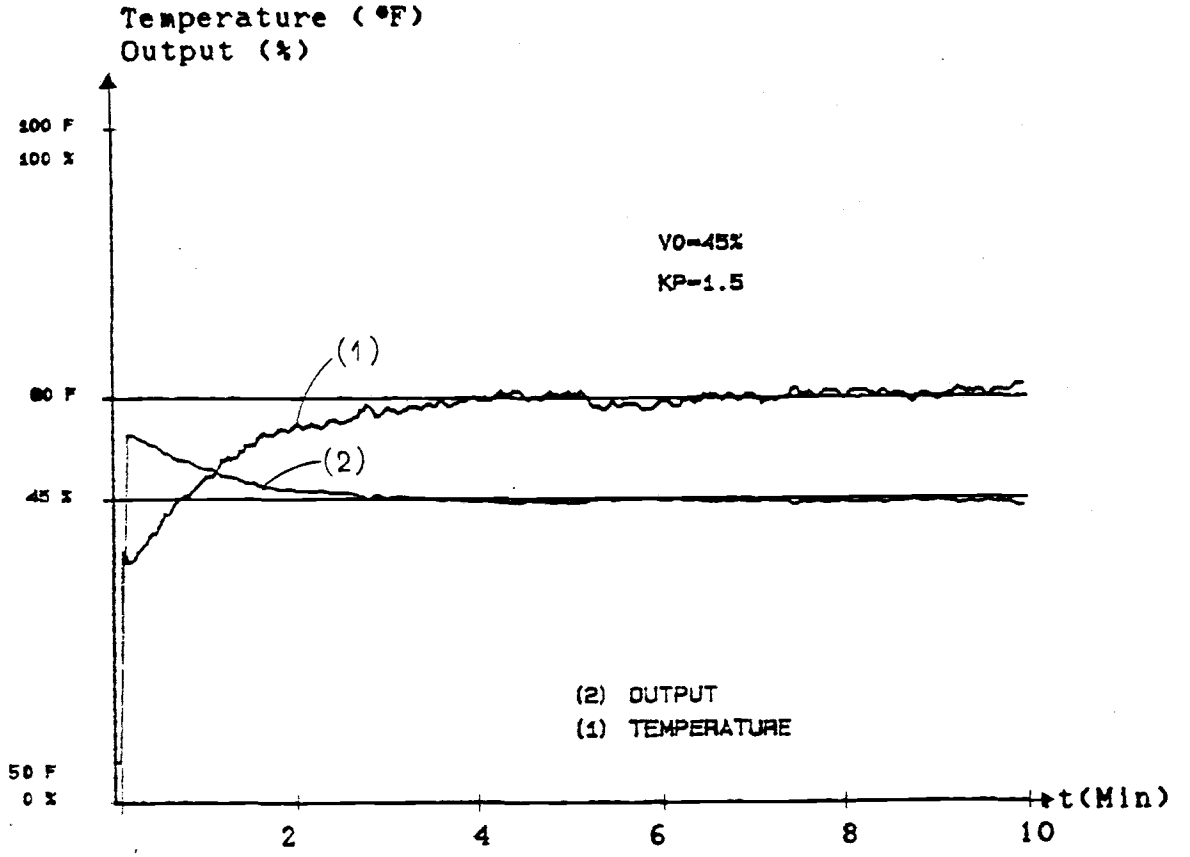



Fig-4.6 Temperature and Output Response in P Control Mode with 45% Initial Output

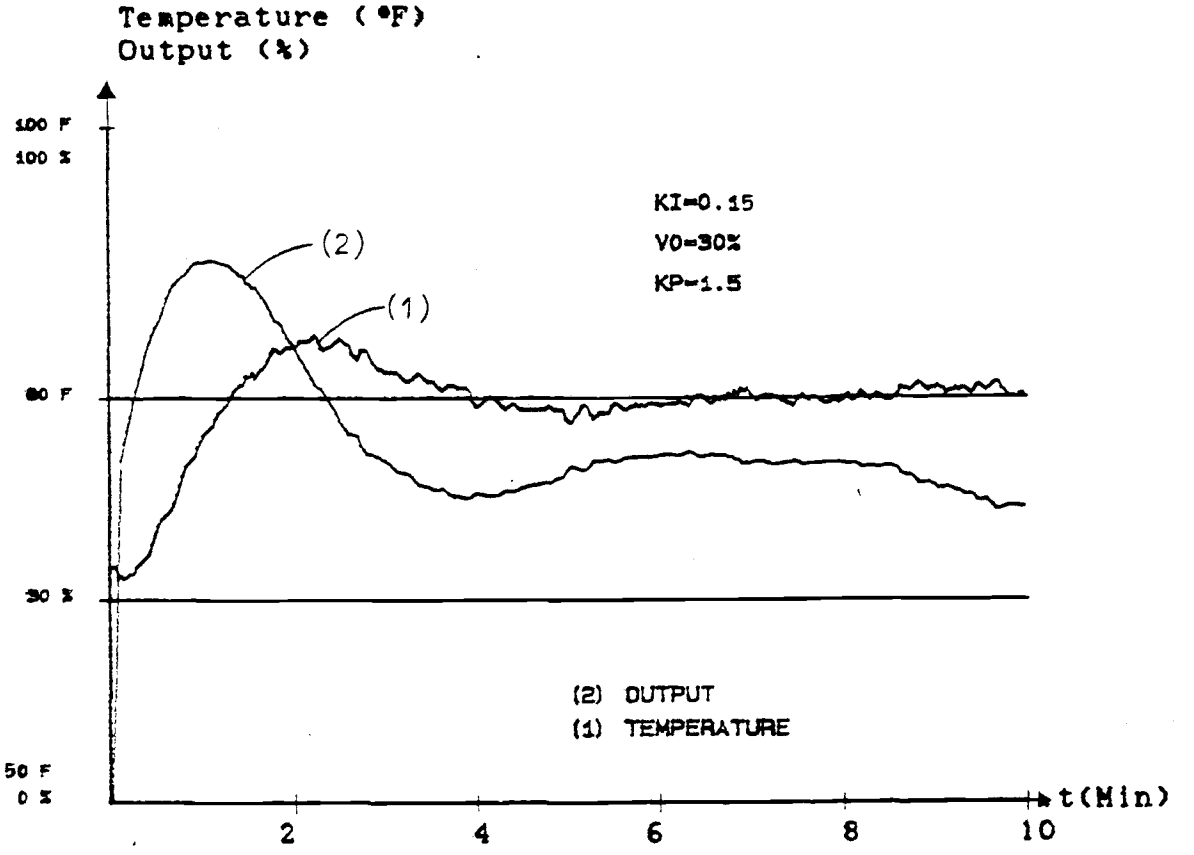



Fig-4.7 Temperature and Output Response in PI Control Mode

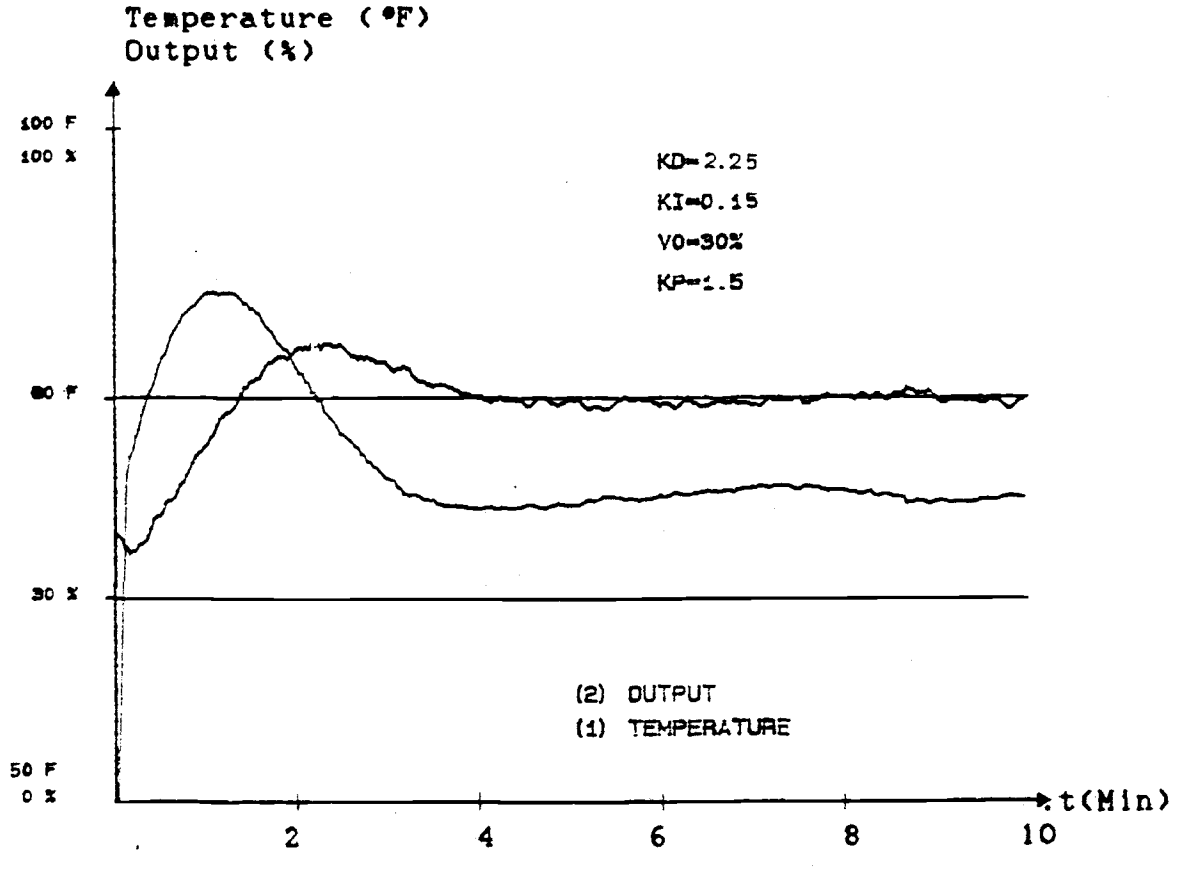



Fig-4.8 Temperature and Output Response in PID Control Mode

# CHAPTER 5

# FUTURE EXPANSION AND CONCLUSION

## 5.1 Future Expansion

One of the advantages of a microprocessor based distributed control system is the capability for developing new software offline using development tools. After debugging and simulation, the program can be transferred to the controller. The same controller hardware may be perform several control functions. For example, with this flexibility, the control algorithm of the PID controller can be replaced with a more accurate control algorithm. A round-off algorithm can replace the truncation algorithm to reduce the computation error. For the hardware, a higher accuracy or higher bit A/D (and D/A) converter can be used. The operational amplifier can be replaced with an instrument amplifier to get the improved results.

To improve the functional capabilities, more control algorithms can be added to the controller so that a higher level control strategy can be implemented. If a multiple controller system is considered, (see Fig-5.1.), the communication protocol will be an important point. It includes the control loop addressing and the data transfer between these loops. A multiple operator interface (see Fig-5.2) is necessary to allow information from many sources to

be shown to the operator. In such a case, a graphic oriented display is preferred [16].

## 5.2 Conclusion

In this paper, The hierarchical distributed control system is described. The discrete form of the PID control algorithm and the hardware structure of the PID controller were implemented with an the Intel 8031 microprocessor. The IBM-PC implemented the high level operator interface. By combining the PID controller and the man-machine interface, the operator can manage the PID controller and supervise the process. This system provides enough capability and flexibility to control the process loop, and the expected result is achieved.

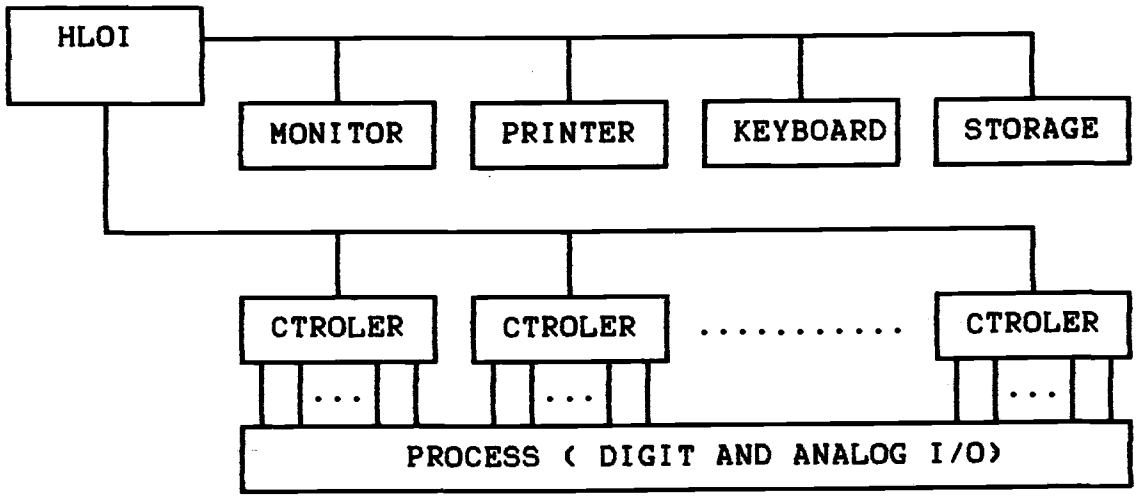



Fig-5.1 Multiple Controller Configuration

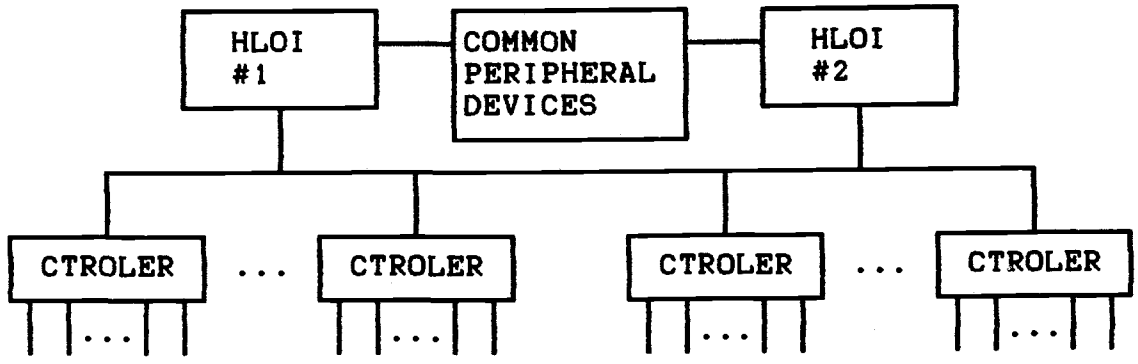



Fig-5.2 Multiple HLOI Configuration

# BIBLIOGRAPHY

[1] M.P. Lukas, Distributed Control System, Their Evaluation and Design, Van Nostrand Reinhold, New York, 1986.

[2] N. J. Krikelis and S. D. Fassois, "Microprocessor Implementation of PID Controller and Lead-Lag Compensator," *IEEE Transactions on Industrial Electronics*, pp79-85, Vol. IE-31 No 1, Feb., 1984.

[3] M. Reed and H.W. Nergler, "A Microprocessor Based Control System," *IEEE Transactions on Industrial Electronics and Control Instrument*, pp253-257, Vol. IECI-24 No 3, Aug., 1977.

[4] C.M. Yu and C.T. Wang and W.N. Chung and C.C. Weng and R.S. Yeng and C.L. Hwang, "The Application of The Distributed Control System in Taoyung Refinery," Symposium on Computer Process Control, Taiwan University, 1982.

[5] D.R. Coughanowr and L.B. Koppel, *Process Systems Analysis and Control*, McGraw Hill, New York, 1965.

[6] R.A. Taylor, "A Distributed Microprocessor Control Phylosophy," *Advances in Instrumentation* , pp131-133 Vol. 32 Part 1, Proceedings of The ISA Conference and Exhibit Oct. 17-20, 1977.

[7] J. H. Herzog, *TASK*MASTER Operating System Manual*,1985.

[8] J. H. Herzog and J. T. Ebner, *X-TENDER Operating System*, Version 1.0, 1984

[9] *TASK-TENDER 1 Application Program*, Agile Systems, 1985.

[10] *Microcontroller Handbook*, Intel, 1986.

[11] G.A. Tendulkar, "Microprocessor Based Industrial Controller," pp207-230 , *Microprocessor in Signal Processing, Measuring and Control*, D. Reidel, 1983.

[12] *Data Conversion / Acquisition Data Book*, National Semiconductor, 1984.

[13] *Integrated Circuits Data Book*, Analog Devices, Vol.1, 1984.

# APPENDICES

# APPENDIX A

## INTERNAL MEMORY MAP OF THE CONTROLLER

R0-R1: GENERAL INDIRECT REGISTER
R3: TEMPORARY REGISTER FOR LOCATION 40H
R4-R5: GENERAL COUNTER IN COMMAND SERVICE ROUTINE
R6: TEMPORARY ACCUMULATOR
R7: SAMPLE RATE TIMER

08-2DH: STACK
2E-2FH: FLAGS (SEE APPENDIX B)
30-3FH: NO USE

40H: PRESENT TEMPERATURE (PVi)
41H: LAST TEMPERATURE (PVi-1)
42-43H: CONVERTED TEMPERATURE (DECIMAL VALUE)
44H: OUTPUT VALUE (OV)
45H: ERROR (ER)
46H: INTEGER PART OF KPER (I(KPER))
47H: FRACTION PART OF KPER (F(KPER))
48H: INTEGER PART OF PITAR (I(PITAR))
49H: FRACTION PART OF PITAR (F(PITAR))
4AH: DIFFERENCE OF (40H)-(41H) (SL)
4BH: INTEGER PART OF SLPDT (I(SLPDT))
4CH: FRACTION PART OF SLPDT (F(SLPDT))
50H: COMMAND KEY WORD (CONTROL ACTION)
51H: HIGH ALARM SETTING (AH)
52H: SETTING POINT (SP) (HIGH-SET FOR ON-OFF)
53H: LOW ALARM SETTING (AL) (LOW-SET FOR ON-OFF)
54H: INTEGER PART OF KP (I(KP)) (AL FOR ON-OFF)
55H: FRACTION PART OF KP (F(KP))
56H: HIGH BYTE OF INITIAL OUTPUT (H(OVO)
57H: LOW BYTE OF INITIAL OUTPUT (L(OVO))
58H: DIRECT/REVERSE (0/1)
59H: INTEGER PART OF PIT (I(PIT))
5AH: FRACTION PART OF PIT (F(PIT))
5BH: SAMPLE PERIOD (TP)
5CH: INTEGER PART OF PDT (I(PDT))
5DH: FRACTION PART OF PDT (F(PDT))

60-7FH: TEMPORARY LOCATIONS FOR COMMAND PACKET

# APPENDIX B

## FLAG ASSIGNMENT

F0: (In status word) instead of EX0
Set when A/D conversion complete.
Cleared when data is read.

2EH.0: DIRECT/REVERSE Action
Set when reverse action is required.
Clear when direct action is required.

2EH.1: Sign of ER (For OUTPUT)
Set when ER is negative
Cleared when ER is positive

2EH.2: Sign of ER (for INTEGRATION)
Set when ER is negative.
Cleared when ER is positive.

2EH.3: Sign of SL
Set when SL is negative.
Cleared when SL is positive.

2FH.0: Instead of TI
Set when byte transmitted
Cleared by user

2FH.1: Indicate the beginning and the end of the command packet.
Set when "<" is encountered.
Cleared when ">" is encountered.

2FH.2: Using for sending out "." back to host in temperature value. For example: the temperature value 123.4 will be place 12 in high byte 34 in low byte.
Set when high byte has been sent.
Cleared when low byte has been sent.

2FH.3: Indicate time to start A/D conversion.
Set when sampling time reached.
Cleared when it is processed.

2FH.4: Indicate the beginning and end of command packet service routine.
Set when ">" is encountered.
Cleared when command packet is processed.

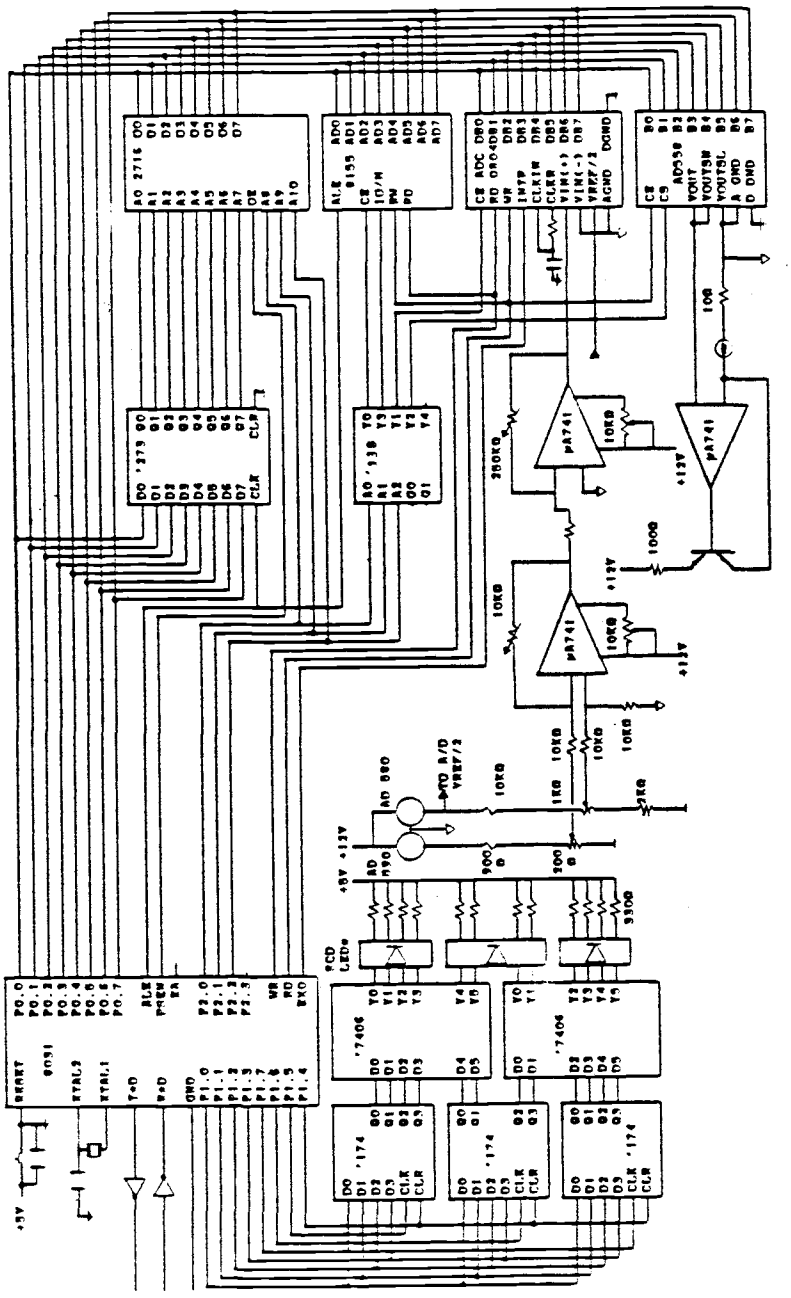
2FH.5: Control function enable/disable
Set when command <M> is received
Cleared when command <C> is received

2FH.6: Alarm signal enable/disable
Set when alarm happened
Cleared when command <R> is received.

2FH.7: Indicate the first time executing ON-OFF control
Set when <O> command received.
Cleared when it is processed.

# APPENDIX C:

## CIRCUIT DIAGRAM



# APPENDIX D

## PROGRAM LIST

```
;
;*************************************************************
;                      THESIS TOPIC:
;          IPLEMENTATION OF A DIGITAL PID CONTROLLER
;         IN A HIERARCHICAL DISTRIBUTED CONTROL SYSTEM
;*************************************************************
;
;
;*************************************************************
;          INTERRUPT VECTOR LOCATION
;*************************************************************
;
;
          ORG       00H
          AJMP      RSTVEC
;
          ORG       03H
          SETB      F0              ;F0 A/D CONVERSION DONE
          RETI
;
          ORG       0BH
          AJMP      TIMEINT
;
          ORG       23H
          AJMP      SPINT
;
;
;*************************************************************
;          INITIALIZATION
;*************************************************************
;
;
          ORG       40H
RSTVEC:   MOV       TMOD,#21H       ;TIMER0 MD1,TIMER1 MD 2
          MOV       TL1,#0E5H
          MOV       TH1,#0E5H       ;BAUD=300
          MOV       TL0,#00H
          MOV       TH0,#06H        ;4 COUNTS/SEC.
          MOV       TCON,#51H       ;START TIMER 0,1
          MOV       SCON,#52H       ;8 BIT SERIAL PORT
          MOV       IE,#93H         ;ENABLE EXT0,ET0,ES
          MOV       IP,#10H         ;S.P HIGHER PRIORITY
          MOV       R7,#08H         ;TIMER COUNTER
          MOV       5BH,#08H        ;2 SECOND SAMPLE RATE
          MOV       2FH,#01H        ;FLAG REGISTER
          MOV       2EH,#00H        ;
          MOV       44H,#00H        ;INITIAL OUTPUT
```

```
;
;
;*************************************************************
;          MAIN LOOP
;*************************************************************
;
;
MAIN:       JNB         2FH.3,CHKCOMSR ;1 SECOND ?
            ACALL       ADSERVE        ;A/D CONVERSION START
            ACALL       ADSAVE         ;SAVE TEMP. AT 8155
            JB          2FH.5,ADSREND
            ACALL       CONTROL
ADSREND:    CLR         2FH.3          ;JOB DONE
CHKCOMSR:   JNB         2FH.1,MAIN     ;RECEIVE COMMAND ?
            ACALL       COMSERVE
            CLR         2FH.1
            AJMP        MAIN
;
;
;*************************************************************
;          TIMER 0 INTERRUPT SERVICE ROUTINE
;*************************************************************
;
;
TIMEINT:    CLR         TF0
            MOV         TH0,#06H
            CJNE        R7,#04H,CHKTIME
            JNB         2FH.6,CHKTIME
            CLR         P1.4
CHKTIME:    DJNZ        R7,TIMEGO
            MOV         R7,#08H
            SETB        2FH.3          ;2FH.3 FOR AD START
TIMEGO:     RETI
;
;
;*************************************************************
;          SERIAL PORT INTERRUPT SERVICE ROUTINE
;*************************************************************
;
;
SPINT:      JNB         TI,RINT
            CLR         TI
            SETB        2FH.0          ;2FH.0 = TI
            RETI
;
RINT:       CLR         RI
            PUSH        ACC
            PUSH        PSW
            MOV         A,SBUF
;
CHKLT:      CJNE        A,#"<",CHKGT
            SETB        2FH.4
            MOV         R0,#60H
```

```
          AJMP      GOBACK
;
CHKGT:    CJNE      A,#">",CHKCOM
          CLR       2FH.4
          SETB      2FH.1
          AJMP      GOBACK
;
CHKCOM:   JNB       2FH.4,GOBACK
          CJNE      A,#" ",SAVCHA
          AJMP      GOBACK
;
SAVCHA:   MOV       @R0,A
          INC       R0

;
GOBACK:   POP       PSW
          POP       ACC
          RETI
;
;
;*************************************************************
;         A/D CONVERSION SERVICE SUBROUTINE
;*************************************************************
;
;
ADSERVE:  MOV       DPTR,#0100H
          MOVX      @DPTR,A         ;START CONVERSION
          JNB       F0,$            ;WAIT EXO INT.
          CLR       F0
          MOVX      A,@DPTR         ;GET AD DATA
          MOV       41H,40H         ;SAVE LAST TEMP
          MOV       40H,A
          ACALL     TEMPVALU        ;CONVERT TO DEGREE
;
          MOV       A,42H
          SWAP      A
          ANL       A,#0FH
          ORL       A,#10H
          MOV       P1,A            ;DISPLAY 1'ST DIGIT
          SETB      P1.7
;
          MOV       A,43H
          ANL       A,#0FH
          ORL       A,#10H
          MOV       P1,A            ;DISPLAY 2'ND DIGIT
          SETB      P1.6
;
          MOV       A,43H
          SWAP      A
          ANL       A,#0FH
          ORL       A,#10H
          MOV       P1,A            ;DISPLAY 3'RD DIGIT
          SETB      P1.5
```

```
        MOV     A,40H
        ACALL   ECHO1           ;ECHO TM EVERYTIME
        RET
;
;
;*************************************************************
;       A/D DATA SAVE TO 8155 RAM SUBROUTINE
;*************************************************************
;
;
ADSAVE: RET
        MOV     R0,#00H
        MOV     R1,#01H
FRESH:  MOVX    A,@R1           ;OLDEST DATA
        MOVX    @R0,A
        INC     R0
        INC     R1
        CJNE    R0,#0FH,FRESH
        MOV     A,40H           ;NEWEST DATA
        MOVX    @R0,A
        RET
;
;
;*************************************************************
;       CONTROL FUNCTION SUBROUTINE
;*************************************************************
;
;
CONTROL: MOV    A,50H
;
CHKM2:  CJNE    A,#"M",CHKO2
        ACALL   MCONTROL
        RET
;
CHKO2:  CJNE    A,#"O",CHKP2
        ACALL   OCONTROL
        RET
;
CHKP2:  CJNE    A,#"P",CHKI2
        ACALL   PCONTROL
        RET
;
CHKI2:  CJNE    A,#"I",CHKD2
        ACALL   ICONTROL
        RET
;
CHKD2:  CJNE    A,#"D",CHKS2
        ACALL   DCONTROL
        RET
;
CHKS2:  CJNE    A,#"S",CHKNO2
        ACALL   SCONTROL
CHKNO2: RET
```

```
;
;
;*************************************************************
;          MANUAL CONTROL ROUTINE
;*************************************************************
;
;
MCONTROL: MOV       44H,51H
          ACALL     POWEROUT
          MOV       A,44H
          ACALL     ECHO1
          RET
;
;
;*************************************************************
;          ON-OFF CONTROL ROUTINE
;*************************************************************
;
;
OCONTROL: JNB       2FH.7,OCTRLBEG ;NEW COMMAND ?
          ACALL     POWEROFF       ;YES
          CLR       2FH.7
OCTRLBEG: MOV       A,40H          ;NO
OCHKTAH:  CLR       C
          SUBB      A,51H
          JC        OCHKTSH
          ACALL     ALARM
;
OCHKTSH:  MOV       A,40H
          CLR       C
          SUBB      A,52H
          JC        OCHKTSL
          ACALL     POWEROFF
;
OCHKTSL:  MOV       A,40H
          CLR       C
          SUBB      A,53H
          JNC       OCHKTAL
          ACALL     POWERON
;
OCHKTAL:  MOV       A,40H
          CLR       C
          SUBB      A,54H
          JNC       ONORMAL
          ACALL     ALARM
ONORMAL:  MOV       A,44H
          ACALL     ECHO1
          RET
;
;
;*************************************************************
;          PROPOTIONAL CONTROL ROUTINE
;*************************************************************
```

```
;
;
PCONTROL: ACALL     GETERROR
          ACALL     GETKPERR
          ACALL     POUTPUT
          ACALL     CHKALARM
          MOV       A,44H
          ACALL     ECHO1           ;ECHO BACK OUTPUT
          RET
;
;
;*************************************************************
;         P & I CONTROL ROUTINE
;*************************************************************
;
;
ICONTROL: ACALL     GETERROR        ;MESSURED-SP
          ACALL     GETKPERR        ;KP*ERROR
          ACALL     GETARERR        ;SUM OF AREA+ERROR
          ACALL     GETPITAR        ;KP*KI*TP*AREA
          ACALL     IOUTPUT
          ACALL     CHKALARM
          MOV       A,44H
          ACALL     ECHO1
          RET
;
;
;*************************************************************
;         P & D CONTROL ROUTINE
;*************************************************************
;
;
DCONTROL: ACALL     GETERROR
          ACALL     GETKPERR
          ACALL     GETSLOPE
          ACALL     GETSLPDT
          ACALL     DOUTPUT
          ACALL     CHKALARM
          MOV       A,44H
          ACALL     ECHO1
          RET
;
;
;*************************************************************
;         P I D CONTROL ROUTINE
;*************************************************************
;
;
SCONTROL: ACALL     GETERROR
          ACALL     GETKPERR        ;P MODE RESULT
          ACALL     GETARERR
          ACALL     GETPITAR        ;I MODE RESULT
          ACALL     GETSLOPE
```

```
          ACALL     GETSLPDT        ;D MODE RESULT
          ACALL     SOUTPUT
          ACALL     CHKALARM
          MOV       A,44H
          ACALL     ECHO1
          RET

;
;
;*************************************************************
;         GET ERROR SUBROUTINE (TM-SP)
;*************************************************************
;
;
GETERROR: CLR       2EH.1           ;CLEAR SIGN OF ERROR
          CLR       2EH.2
          CLR       C
          MOV       A,40H
          SUBB      A,52H           ;SP IN 52H
          JNC       ERPOSIT         ;POSITIVE
          SETB      2EH.1           ;NEGTIVE
          SETB      2EH.2           ;FOR AREA USE
          CPL       A
          ADD       A,#01H
ERPOSIT:  MOV       45H,A           ;SAVE ERROR IN 45H
          RET
;
;
;*************************************************************
;        GET KP * ERROR SUBROUTINE
;*************************************************************
;
;
GETKPERR: MOV       A,45H           ;ERROR IN 45H
          MOV       B,54H           ;INTEGER PART OF KP
          MUL       AB
          MOV       46H,A           ;SAVE INT. PART OF KP*ERR
          MOV       A,B
          CJNE      A,#00H,KPERROV ;OVERFLOW
;
          MOV       A,45H
          MOV       B,55H           ;FRACTION PART OF KP
          MUL       AB              ;
          MOV       47H,A           ;SAVE FRACT. OF KP*ERR
          MOV       A,B             ;USE INTEGER PART ONLY
          CLR       C
          ADD       A,46H
          JC        KPERROV         ;OVERFLOW
          MOV       46H,A           ;SAVE INT. PART OF KP*ERR
          AJMP      KPERREND
KPERROV:  MOV       46H,#0FFH       ;SAVE INT. PART OF KP*ERR
KPERREND: RET
;
```

```
;
;*************************************************************
;         GET INTEGRATE AREA SUBROUTINE
;*************************************************************
;
;
GETARERR: JNB       2EH.0,ARERRBEG ;
          CPL       2EH.2          ;REVERSE ACTION
ARERRBEG: JB        2EH.2,ARERMIN
          CLR       C              ;AREA+ERROR
          MOV       A,57H          ;LOW BYTE OF AREA
          ADD       A,45H
          MOV       57H,A          ;SAVE BACK
          JNC       ARENEND
          CLR       C
          MOV       A,56H          ;HIGH BYTE OF AREA
          ADD       A,#01H
          MOV       56H,A          ;SAVE BACK
          JNC       ARENEND
          MOV       56H,#0FFH      ;OVER FLOW
          MOV       57H,#0FFH
          AJMP      ARENEND
;
ARERMIN:  CLR       C               ;AREA-ERROR
          MOV       A,57H           ;LOW BYTE OF AREA
          SUBB      A,45H
          MOV       57H,A           ;SAVE BACK
          JNC       ARENEND
          CLR       C
          MOV       A,56H           ;HIGH BYTE OF AREA
          SUBB      A,#01H
          MOV       56H,A           ;SAVE BACK
          JNC       ARENEND
          MOV       56H,#00H        ;DOWN FLOW
          MOV       57H,#00H
ARENEND:  RET
;
;
;*************************************************************
;         GET PITAR SUBROUTINE (KP*KI*TP*AREA)
;*************************************************************
;
;
GETPITAR: MOV       A,59H           ;INT(KP*KI*TP)
          CJNE      A,#00H,PITAROF  ;IF <>0 THEN OVERFLOW
          MOV       A,5AH           ;FRAC(KP*KI*TP)
          MOV       B,56H           ;HIGH BYTE OF AREA
          MUL       AB
          MOV       48H,A           ;LOW BYTE OF RESULT
          MOV       A,B
          CJNE      A,#00H,PITAROF  ;OVERFLOW
;
          MOV       A,5AH           ;FRAC(KP*KI*TP)
```

```
          MOV       B,57H           ;LOW BYTE OF AREA
          MUL       AB
          MOV       A,B             ;INT. (HIGH BYTE RESULT)
          CLR       C
          ADD       A,48H
          JC        PITAROF         ;OVER FLOW
          MOV       48H,A           ;SAVE THE RESULT
          AJMP      PITAREND
PITAROF:  MOV       48H,#0FFH
PITAREND: RET
;
;
;*************************************************************
;         GET TEMPERATURE DIFFERENCE SUBROUTINE (40H)-(41H)
;*************************************************************
;
;
GETSLOPE: CLR       2EH.3
          CLR       C
          MOV       A,40H           ;PRESENT TEMP
          SUBB      A,41H           ;LAST TEMP
          JNC       SLOPOSIT
          SETB      2EH.3           ;NEGATIVE SIGN
          CPL       A
          ADD       A,#01H          ;2'S COMPLEMENT
SLOPOSIT: MOV       4AH,A           ;RESULT OF SLOPE IN 4AH
          RET
;
;
;*************************************************************
;         GET SLOPE*KP*KD/TP SUBROUTIN
;*************************************************************
;
;
GETSLPDT: MOV       A,4AH           ;GET SLOPE
          MOV       B,5CH           ;INT(KP*KD/TP)
          MUL       AB
          MOV       4BH,A           ;SAVE INT RESULT
          MOV       A,B
          CJNE      A,#00H,SLPDTOV  ;OVER FLOW
;
          MOV       A,4AH
          MOV       B,5DH           ;FRAC(KP*KD/TP)
          MUL       AB
          MOV       4CH,A           ;SAVE FRAC RESULT
          MOV       A,B
          CLR       C
          ADD       A,4BH
          JC        SLPDTOV         ;OVERFLOW
          MOV       4BH,A           ;GET RESULT
          AJMP      SLPDTEND
SLPDTOV:  MOV       4BH,#0FFH
SLPDTEND: RET
```

```
;
;
;*************************************************************
;          OUTPUT OF P CONTROL SUBROUTINE
;*************************************************************
;
;
POUTPUT:   JNB        2EH.0,POUTDIR   ;DIRECT ACTION
           CPL        2EH.1           ;REVERSE ACTION
POUTDIR:   JB         2EH.1,POUTMIN   ;
           CLR        C               ;OUTPUTO+(KP*ERR)
           MOV        A,57H           ;OUTPUT AT ERROR=0
           ADD        A,46H           ;INT(KP*ERR)
           JC         POUTOF          ;OVER FLOW
           MOV        44H,A
           AJMP       POUTEND
POUTOF:    MOV        44H,#0FFH
           AJMP       POUTEND
;
POUTMIN:   CLR        C               ;OUTPUTO-(KP*ERR)
           MOV        A,57H
           SUBB       A,46H
           JC         POUTDF          ;DOWN FLOW
           MOV        44H,A
           AJMP       POUTEND
POUTDF:    MOV        44H,#00H
POUTEND:   MOV        A,50H
           CJNE       A,#"P",POUTPD   ;FOR PD CONTROL
           ACALL      POWEROUT
POUTPD:    RET
;
;
;*************************************************************
;          OUTPUT OF P&I CONTROL SUBROUTINE
;*************************************************************
;
;
IOUTPUT:   JNB        2EH.0,IOUTDIR   ;DIRECT ACTION
           CPL        2EH.1           ;REVERSE ACTION
IOUTDIR:   JB         2EH.1,IOUTMIN   ;
           CLR        C               ;(KP*KI*TP*AR)+(KP*ERR)
           MOV        A,48H           ;(KP*KI*TP*AR)
           ADD        A,46H           ;INT(KP*ERROR)
           JC         IOUTOF          ;OVER FLOW
           MOV        44H,A           ;SAVE OUTPUT
           AJMP       IOUTEND
IOUTOF:    MOV        44H,#0FFH
           AJMP       IOUTEND
;
IOUTMIN:   CLR        C               ;(KP*KI*TP*AR)-(KP*ERR)
           MOV        A,48H
           SUBB       A,46H
           JC         IOUTDF          ;DOWN FLOW
```

```
          MOV       44H,A
          AJMP      IOUTEND
IOUTDF:   MOV       44H,#00H
IOUTEND:  MOV       A,50H
          CJNE      A,#"I",IOUTPID ;FOR PID CONTROL
          ACALL     POWEROUT
IOUTPID:  RET
;
;
;*************************************************************
;         OUTPUT OF P&D CONTROL SUBROUTINE
;*************************************************************
;
;
DOUTPUT:  ACALL     POUTPUT
DOUTPUTS: JNB       2EH.0,DOUTDIR
          CPL       2EH.3           ;REVERSE ACTION
DOUTDIR:  JB        2EH.3,DOUTMIN
          CLR       C               ;OUT+SLPDT
          MOV       A,44H           ;OUTPUT FROM P MODE
          ADD       A,4BH           ;ADD OUT WITH D MODE
          JC        DOUTOV          ;OVERFLOW
          MOV       44H,A           ;GET FINAL OUTPUT
          AJMP      DOUTEND
DOUTOV:   MOV       44H,#0FFH
          AJMP      DOUTEND
;
DOUTMIN:  CLR       C               ;OUT-SLPDT
          MOV       A,44H
          SUBB      A,4BH
          JC        DOUTDF          ;DOWN FLOW
          MOV       44H,A
          AJMP      DOUTEND
DOUTDF:   MOV       44H,#00H
DOUTEND:  MOV       A,50H
          CJNE      A,#"D",DOUTPID
          ACALL     POWEROUT
DOUTPID:  RET
;
;
;*************************************************************
;         OUTPUT OF PID CONTROL
;*************************************************************
;
;
SOUTPUT:  ACALL     IOUTPUT
          ACALL     DOUTPUTS
          ACALL     POWEROUT
          RET
;
;
;*************************************************************
;         POWER ON SUBROUTINE FOR ON-OFF CONTROL
```

```
;*************************************************************
;
;
POWERON:  MOV       DPTR,#0200H
          MOV       A,#0FFH
          MOVX      @DPTR,A
          MOV       44H,A
          RET
;
;
;*************************************************************
;         POWER OFF SUBROUTINE FOR ON-OFF CONTROL
;*************************************************************
;
;
POWEROFF: MOV       DPTR,#0200H
          MOV       A,#00H
          MOVX      @DPTR,A
          MOV       44H,A
          RET
;
;
;*************************************************************
;         POWER OUT SUBROUTINE FOR PID CONTROL
;*************************************************************
;
;
POWEROUT: MOV       DPTR,#0200H
          MOV       A,44H
          MOVX      @DPTR,A
          RET
;
;
;*************************************************************
;        CHECK HIGH AND LOW ALARM SUBROUTINE
;*************************************************************
;
;
CHKALARM: MOV       A,40H
          CLR       C
          SUBB      A,51H
          JC        CHKTAL
          ACALL     ALARM
;
CHKTAL:   MOV       A,40H
          CLR       C
          SUBB      A,53H
          JNC       NORMAL
          ACALL     ALARM
;
NORMAL:   RET
;
;
```

```
;*************************************************************
;          ALARM SUBROUTINE
;*************************************************************
;
;
ALARM:     SETB       2FH.6
           RET
;
;
;*************************************************************
;          COMMAND SERVICE SUBROUTINE
;*************************************************************
;
;
COMSERVE:  MOV        A,60H
;
CHKT1:     CJNE       A,#"T",CHKR1    ;ECHO 256 TEMP. BACK
           ACALL      ECHO256
           RET
;
CHKR1:     CJNE       A,#"R",CHKC1
           CLR        2FH.6           ;RESET ALARM
           SETB       2FH.5           ;NO CONTROL
           RET
;
CHKC1:     CJNE       A,#"C",CHKM1
           CLR        2FH.5
           RET
;
CHKM1:     CJNE       A,#"M",CHKS1
           MOV        R4,#04H         ;MANUAL CONTROL
           MOV        R5,#02H
           ACALL      ASCHEXB
           RET
;
CHKS1:     CJNE       A,#"S",CHKP1
           MOV        R4,#1CH
           MOV        R5,#0EH
           ACALL      ASCHEXB
           RET
;
CHKP1:     CJNE       A,#"P",CHKI1
           MOV        R4,#12H
           MOV        R5,#09H
           ACALL      ASCHEXB
           RET
;
CHKI1:     CJNE       A,#"I",CHKO1
           MOV        R4,#18H
           MOV        R5,#0CH
           ACALL      ASCHEXB
           RET ; CHKO1:    CJNE       A,#"O",CHKD1
           SETB       2FH.7           ;NEW COMMAND
```

```
          MOV       R4,#0CH
          MOV       R5,#06H
          ACALL     ASCHEXB
;
CHKD1:    CJNE      A,#"D",CHKNO1
          MOV       R4,#1CH
          MOV       R5,#0EH
          ACALL     ASCHEXB
CHKNO1:   RET
;
;
;*************************************************************
;         CONVERT COMMAND PACKET FROM ASCII TO HEX
;*************************************************************
;
;
ASCHEXB:  MOV       R0,#61H
ASCHEX:   MOV       A,@R0
          JNB       ACC.6,NOREVIS
          ADD       A,#0F9H
NOREVIS:  ANL       A,#0FH
          MOV       @R0,A
          INC       R0
          DJNZ      R4,ASCHEX
;
          MOV       R0,#61H
          MOV       R1,#51H
HEXHEX:   MOV       A,@R0
          SWAP      A
          INC       R0
          XRL       A,@R0
          MOV       @R1,A
          INC       R0
          INC       R1
          DJNZ      R5,HEXHEX
          MOV       R5,58H
          CJNE      R5,#01H,ASCHEXDR  ;0 OR 1 IN  57H
          SETB      2EH.0             ;REVERSE ACTION
          AJMP      ASCHEXEN
ASCHEXDR: CLR       2EH.0             ;DIRECT ACTION
ASCHEXEN: MOV       50H,60H           ;SAVE HEY COMMAND WORD
          RET
;
;
;*************************************************************
;         ECHO NEWEST TEMPERATURE BACK SUBROUTINE
;*************************************************************
;
;
ECHO1:    ACALL     TEMPVALU
          MOV       A,43H
          ACALL     SENDTEMP
          MOV       A,42H
```

```
          SETB      2FH.2           ;2FH.2 FOR PRINTINT "."
          ACALL     SENDTEMP
          CLR       2FH.2
          RET
;
;
;*************************************************************
;         ECHO 256 ELLAPSED TEMPERATURE BACK
;*************************************************************
;
;
ECHO256:  RET
          MOV       R0,#00H
ECHOCONT: MOVX      A,@R0
          ACALL     ECHO1
          INC       R0
          CJNE      R0,#10H,ECHOCONT
          RET
;
;
;*************************************************************
;         SEND TEMPERATURE BACK TO TERMINAL SUBROUTINE
;*************************************************************
;
;
SENDTEMP: MOV       R6,A
          JB        2FH.2,SEND0
          MOV       A,#" "
          ACALL     OUT
;
SEND0:    MOV       A,R6
          SWAP      A
          ANL       A,#0FH
          ORL       A,#30H
          JB        ACC.3,SC1
          AJMP      SC3
SC1:      JB        ACC.2,SC2
          JB        ACC.1,SC2
          AJMP      SC3
SC2:      ADD       A,#07H
SC3:      ACALL     OUT
;
          JNB       2FH.2,SEND1
          MOV       A,#"."
          ACALL     OUT
;
SEND1:    MOV       A,R6
          ANL       A,#0FH
          ORL       A,#30H
          JB        ACC.3,SC4
          AJMP      SC6
SC4:      JB        ACC.2,SC5
          JB        ACC.1,SC5
```

```
          AJMP      SC6
SC5:      ADD       A,#07H
SC6:      ACALL     OUT
          RET
;
;
;*************************************************************
;         OUTPUT CHARACTER SUBROUTINE
;*************************************************************
;
;
OUT:      JNB       2FH.0,$
          CLR       2FH.0
          MOV       SBUF,A
          RET
;
;
;*************************************************************
;         GET THE VALUE OF TEMPERATURE (F)
;*************************************************************
;
;
TEMPVALU: MOV       R3,A
          MOV       DPTR,#ADTABLL
          MOVC      A,@A+DPTR
          MOV       42H,A
          MOV       A,R3
          MOV       DPTR,#ADTABLH
          MOVC      A,@A+DPTR
          MOV       43H,A
          RET
;
;
;*************************************************************
;         TEMPERATURE LOOK UP TABLE
;*************************************************************
;
;
ADTABLH:  DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          03H
          DB          04H
          DB          04H
          DB          04H
```