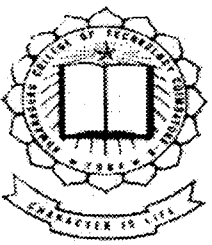


MICRO CONTROLLER BASED

REMOTE MONITORING USING PC

PROJECT WORK DONE AT SALEM STEEL PLANT, SALEM.
PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF
BACHELOR OF ENGINEERING IN INFORMATION TECHNOLOGY
OF BHARATHIAR UNIVERSITY, COIMBATORE

SUBMITTED BY

Ms.Malathi.R	**9827S0014**
Mr.Paulson Samuel Vinod Kumar.J	**9827S0018**
Mr.Rahul Joshi	**9827S0020**

GUIDED BY

Dr.S.Thangasamy, B.E, Ph.D.,

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE-641 006

This is to certify that the project work entitled "MICRO CONTROLLER BASED REMOTE MONITORING SYSTEM USING PC" done by

Ms.Malathi.R	**9827S0014**
Mr.Paulson Samuel Vinod Kumar.J	**9827S0018**
Mr.Rahul Joshi	**9827S0020**

in partial fulfillment for the award of the degree of Bachelor of Engineering in **Information Technology** of Bharathiar University, Coimbatore during the Academic year 2001-2002.

S. Jhanjay 18/3/02
Head of the Department

S. Jhanjay 18/3/02
Guide

Certified that the candidate was examined by us in the Project Work Viva Voce Examination held on 18/3/2002 and the University number was 9827S0018.

Rajan
Internal Examiner

18/3/2002
External Examiner

ACKNOWLEDGEMENT

First of all I would like to thank God for his abundant grace and mercies without which this project would not have been taken up and successfully completed.

I am grateful to our Principal Dr.K.K.Padmanabhan B.sc (Engg), M.Tech, Ph.D, Kumaraguru College of technology, Coimbatore for the overall support that he provided during the project work.

I express my heartfelt thanks to my internal guide, Dr.S.Thangasamy, Ph.D., Professor and Head of the Computer Science Department, Kumaraguru College of technology for his sound counsel and timely guidance for the successful completion of this project.

I take this opportunity to express my heartfelt gratitude to Ms.S.Rajini, Lecturer in Computer Science Department, Kumaraguru College of Technology for the overall support that she provided during the project work.

I would fail in my duty, if I do not thank Steel Authority of India Ltd, Salem Steel Plant, Salem and the following executives for their immense help in designing, assembling and testing this project at their plant.

1, SRI. MURUGESAN, senior manager (HRD)

2, SRI. J.P. JEYASEKARAN senior manager (electrical)

3, SRI. THIAGARAJAN senior technician

My sincere word of thanks for all the staff members of Kumaraguru College of technology, Coimbatore for the help rendered in the successful completion of this task.

SYNOPSIS

Nowadays, PC is used for the following.

(1) Storing data and files.

(2) Sending E-mails to any part of the globe.

(3) For sharing information from any part of the globe using Internet.

(4) For E-commerce through Internet.

It can also be used as a remote monitor. For example, to monitor continuously the weather conditions and also to monitor the position of doors and windows (as a security product). Our project is to demonstrate how a PC can be used as a remote monitor to display continuously the ambient temperature and also the position of the safety locker, door etc of the bank by a security person.

Our project consists of a temperature transducer, placed in the atmospheric conditions or at any temperature. It gives millivolts output, corresponding to the ambient temperature or the temperature, sensed by the transducer. The value of Rt changes with the value of temperature (t). Since, t varies the current (I) varies which results in the variation of the output voltage (V) from the transducer. This mV is amplified by the operational amplifier and given to the ADC. ADC converts the analog input to 8-bit binary and gives it to the micro controller on demand. The controller output is given to the RS232 converter, which gives RS232 output and connected to the serial port of the PC. This is the hardware part of it. We use micro controller instructions for controlling the operations of the micro controller. We use VC++ as our front-end.

A limit switch is attached to the safety locker room door of the bank. The moment the door is open, a digital input is given to the micro controller, which again gives output to the PC through a RS232 converter. The PC, placed in the house of the bank manager can continuously display the door position.

INDEX

INDEX

SYSTEM REQUIREMENTS

1. SYSTEM REQUIREMENTS

1.1 PRODUCT DEFINITION:

1.1.1 Problem Statement:

Remote Monitoring using PC:

* Monitoring of temperature.
* Security System.

1.1.2 Processing Environment:

The Processing Environment of our Project involves both the hardware and software together.

Hardware Used:

- Opamp (LM324)
- ADC (0808)
- Temperature Sensors (AD590)
- Micro Controller (89C51)
- Serial RTC
- Power Supply Cord
- Limit Switch
- Buzzers (1.5 to 15V)
- Opto Couplers
- Various resistors & Capacitors

Software Used:

- Micro Controller Assembly Language Programming
- VC++ (for front end)

1.1.3 User Characteristics:

The user can continuously record the various temperatures , store it in a file and observe for suitable required measures.

The security person can continuously watch the position of the door, gate or safety locker etc through the PC in a remote location .

1.1.4 Product Features:

The various Product Features of our project are

- Buzzer sounds for security mechanism.
- Display of various temperatures in degree Celsius.

PROJECT PLAN

2. PROJECT PLAN

2.1 ABOUT VISUAL C++:

Programs written for traditional operating environments use a procedural programming model in which programs execute from top to bottom in an orderly fashion. The path taken from start to finish may vary with each invocation of the program depending on the input it receives. In a C program, execution begins with a first line in the first line named main and ends when the main returns.

Windows programs operate differently. They use the event driven programming model in which the applications respond to the events by processing the messages sent by the operating system. An event could be a keystroke, a mouse click, or the command for the window to repaint itself, among others things. The entry point for the windows program is the Win Main function, but most of the function takes place in the function called the window procedure. The window procedure process messages sent to the window .Win Main creates that the window and then enters the message loop, alternately retrieving and dispatching them to Windows procedure. Messages wait in a message queue until they are retrieved. Atypical windows application performs the bulk of its processing in response to message it receive, and in between messages it does little except wait for the next message to arrive.

The message loop ends when a WM_QUIT message is retrieved from the message queue signaling its time for the application to end. The message usually appears because the user selected Exit from the file menu. When the message loop ends Win Main returns and the application terminates.

The window procedure typically calls other functions to help process the message it receives. It can call functions local to the application or can call API functions provided by windows. API functions are contained in special modules known as Dynamic Linking Libraries. (DLL). The code provided to process a particular message is known as Message Handler.

Win Main begins by calling the API function Register Class to register a window class. The window class defines important characteristics of a window such as its window procedure address, its default background color and so on. These and other properties are defined by filling in the fields of a WNDCLASS structure, which is subsequently passed to Register class. An application must specify a window class when it creates a window. Once the window class is registered, Win Main calls the all-important Create Window function to create the application's window. The first parameter to Create Window is the name of the WNDCLASS from which the window will be created. The second parameter is the text that will appear in the windows title bar. The third specifies the window style. The next four parameters specify the window's initial position and size .CW_USEDDEFAULT tells window to use default values for both. The final four parameters specify, in order, the handle of the window's parent window.

Next comes the message loop. In order to retrieve and dispatch messages, Win Main executes a simple while loop that calls the GetMessage, TranslateMessage, and DispatchMessage API functions repeatedly. GetMessage checks the message queue. If a message is available, it is removed from the queue and copied to msg; otherwise, GetMessage blocks on the empty message queue until a message is available. msg is an

instance of the structure MSG. TranslateMessage converts a keyboard message denoting a character key to an easier-to-use WM_CHAR message, and DispatchMessage dispatches the message to the window procedure. The message loop executes until GetMessage returns 0, which happens only when a WM_QUIT message is retrieved from the message queue. Messages dispatched with DispatchMessage generate call to the window procedure WndProc. Finally, the WM_DESTROY handler calls the PostQuitMessage API function to post a WM_QUIT message and ultimately cause the program to terminate.

2.2 Programming the 8051:

Lines of Code

Assembly language programs are written as a sequence of text lines. A text line is nothing more than a line of all alphanumeric characters that are stored in a disk file. Each line ends when a carriage return character (or carriage return line feed) is stored by the text editor on the disk. Using a word processor or text editor creates the file. The only condition placed on the file is that it must be stored on the disk in pure ASSCII (American Standard Code for Information Interchange) format. Using ASCII format for the file ensures that unusual characters, used by most word processors for special functions (such as underlining or tabs), do not appear in the final file.

Each line of text is an instruction to the CPU, a directive to the assembler, or a combination of the two. Many names have been used for a single text line in a source file, such as a statement, or a line of code, or an instruction.

8051 Instruction Syntax.

Almost all computer instructions involve taking one piece of data from a location somewhere inside the computer and "operating" on that data together with another piece of data located somewhere else inside the computer. Data originates at the source location and ends up at the destination location. (We will use the terms location and address interchangeably when discussing the physical place that binary data is stored).

Instructions commonly start with an action mnemonic that reminds us of what the instruction does. The instruction mnemonic 6 is the first word of the instruction and indicates in what manner, or how, the data are to be combined or otherwise manipulated by the CPU.

Each code line may also contain comments and a name for the instruction address in code memory, called the label of the code address. A label is a name, or symbol, given to the address number where the first byte of the labeled instruction is stored in code memory. We have seen that instructions are assembled into code memory with each byte of code located at a unique address number. Labels let us convert address numbers in code memory into names of our choosing and let the assembler worry about exactly which address number has each name. If the address number of a line of code should happen to change as a result of adding a new instruction, for instance, the name remains the same, only the address number for that name will change. The assembler can keep track of address numbers for each name, and we can keep track of the names.

The overall syntax of a line of 8051 program code is shown in Figure.

An assembly Language Instruction Line

Label:	instruction	;comment(s)

Labels:

A label can be any combination of up to 8 letters (A-Z), numbers (0-9), and period (.).

Labels must begin with a letter fro A(a) to Z(z) or the period. We do not begin a label with any decimal number from 0 to 9, or the assembler may assume it is a number. Examples of code address labels are the following.

fred.2:

square:

tabletwo:

curses:

setmode:

transmit:

receive:

ad1234h:

Although we may legally use labels up to 8 characters in length, experienced programmers recommend names of 7 or fewer characters. Try to keep label names short and related to the program. Do not use label names that could be considered offensive to a reasonable person.

Labels should be placed in the first text column, on the far left of the source file page, so that we can see, quickly, that it is label and not an instruction. Labels may also be used alone on a line, with no following

instruction. The assembler will associate the label with the first instruction line after the label.

Labels, when used to assign a name to allocation in the program, always end with a colon(:) to indicate they are labels and not instructions. Any other use of a label, such as part of an instruction operand, always omits the colon.

Labels cannot be the name of any register or instruction used by the 8051 such as DPH, mov, or djnz. Such names are called reserved words, or keywords, because they already been defined and reserved for assembler use by the A51 assembler author. Instructions, without a label preceding them on a line, can begin in any text column after Column 1. Programs are more readable, however, if instructions are indented 10 spaces from the left margin.

INSTRUCTIONS

An 8051 instruction is any of the coded set that has been defined by the manufacturer of the 8051. Every instruction can be converted into a unique machine-language binary code that can be acted on by the 8051 internal circuitry. Undefined binary codes will cause the CPU to perform erratically.

Each instruction is generally made up to three distinct parts, as shown below.

An 8051 Instruction

Instruction Part	1	2	3
	mov	destination,	source

Part 1, the instruction mnemonic, is intended to jog our memories by sounding like the operation to be performed by the CPU. For instance, the instruction shown in Figure above will move data from on location in the computer (the source) to another location (the destination).

Part 2 is called an operand or data address, because it specifies the destination for the data that is being copied from the source.

Part 3 is also an operand and contains the address of the source location in the computer that is providing data to be copied to the destination.

The names of the destination and source address are separated by a comma (,) so that the assembler will know when one operand name ends and the other begin. Do not leave out the comma, for it is as important as any other part of the instruction.

Note that I have used the term copied rather than moved in the description of a data mov instruction. This is usually the case for most computer data transfer operations. Data is copied from the source to the destination, not physically moved and nothing left behind.

There is no reason that the destination address should be listed before the source address except that doing so is a convention established for the 8051 assembler. A programming convention unlike a law of physics is purely a human invention.

COMMENTS

The programmer includes comments as simple, terse explanations of exactly what the instruction is doing to make the program function. Beginning programmers often omit comments, a very dangerous practice that can lead to sloppy and erroneous programs. Commenting each program line is encouraged, even when the meaning of the line is obvious. Comments begin with a semicolon (;) to indicate that they are not one of the operands. Anything can be typed after the semicolon with impunity, even reserved words. Examples of comments include the following:

```
;this is a comment
;and so is this
;mov a, b
;hello mom!
;and so on
```

One rather handy way to use the semicolon when debugging a program is to place a semicolon in front of a line of code that you may think (but are not absolutely sure) needs to be deleted. Placing a semicolon in front of the line of suspect code will ensure it is not part of the assembled program. You can easily restore the line in your program by removing the semicolon, should your suspicions prove wrong.

HARDWARE REQUIREMENTS

3. HARDWARE DESCRIPTION:

3.1 Temperature Transducer

The AD590 is a three-terminal integrated circuit temperature transducer that produces an output current proportional to absolute temperature. For supply voltages between +4v and +30v the device acts as high impedance, constant current regulator passing 1 microA/K.Laser trimming of the chip's thin-film resistors is used to calibrate the device to 298.2 microA output at 298.2K(+25'C).

The AD590 should be used in any temperature sensing application below +150'C in which conventional electrical temperature sensors are currently employed. The inherent low cost of a monolithic integrated circuit combined with the elimination of support circuitry makes the AD590 an attractive alternative for many temperature measurement situations. Linearization circuitry, precision voltage amplifiers, resistance
Measuring circuitry and cold junction compensation is not needed in applying the AD590.

In addition to temperature measurement, applications include temperature compensation or correction of discrete components; biasing proportional to absolute temperature, flow rate measurement, level detection of fluids and anemometry. The AD590 is available in chip form making, is suitable for hybrid circuits and fast temperature measurements in protected environments.

The AD590 is particularly useful in remote sensing applications. The device is insensitive to voltage drops over long lines due to its high impedance current output. Any well-insulated twisted pair is sufficient for operation hundreds of feet from the receiving circuitry. The output

characteristics also make the AD590 easy to multiplex; the current can be switched by a CMOS multiplexer or the supply voltage can be switched by a logic gate output. The size and responsiveness of the AD590 make it perfect for uses where size is a consideration, such as on PC boards or heat sinks.

AD 590 WORKING PRINCIPLE:

The transducer has three terminals ; positive, negative and ground . 5volts dc is

Applied between positive and ground and a 1000ohms resistor is connected between the negative and ground .That means 5 V0lt dc is applied to the transducer and a resistor of 1000ohms, which is in series with the the transducer.

Transducer's resistance=Rt

Rt=Ro (1+mt)

m=temperature coefficient

t=temperature

Now current= 5v/ (Rt+1000)

Volt across 1000ohms resistor= 5/(Rt+1000)*1000

This voltage is applied to the op-amp, which depends on the temperature.

Temperature(t-Deg Centigrade)	Thermal Resistance(Rt in ohms)	Current(I- amps)	Voltage(V-in mV)
32 Deg	-937.5	0.08	80
100 Deg	-980	0.25	250
0 Deg	1000	0.0025	2.5

Features:

1. Linear Current output
2. Broad Range –55 to 150'C
3. No Linearization circuitry required
4. Versatile and Economical
5. Fast response

AD590 – Two Styles available:

1. Low cost metal TO-52 case
2. Miniature Ceramic Flat pack

Two Accuracies Available:

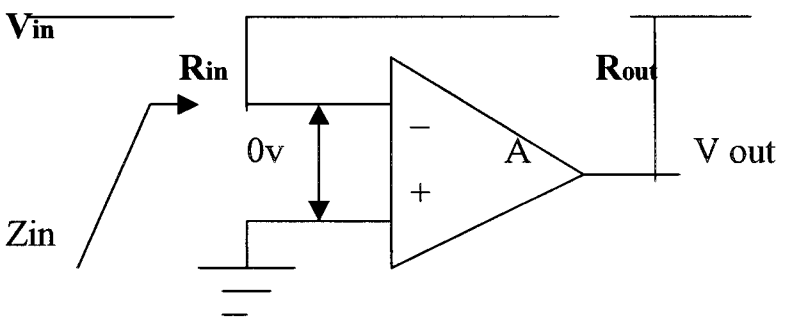
1. J for Low cost
2. K for increased accuracy

AD590 Applications:

1. Use in Custom Made Probes
2. Use on PC Boards for Accurate Measurements

3.2 **Operational Amplifiers**

The Operational amplifier (op amp) is used with A to D and D to A converters. We will zero in on the key features that make the op amp useful at the analog interface. The following figure shows the symbol for an op amp.



Virtual Ground:

In this figure, V_{out} is the output voltage with respect to ground. A is the open-loop voltage gain of the op amp, often more than 100,000. When connected as an inverter, the noninverting input (+ input) is grounded. The inverting input (-input) receives the signal voltage.

Because the voltage gain of an op amp is so large, the input voltage is in microvolts. To a first approximation, the input voltage may be treated as 0V. Furthermore, the input impedance of the inverting input approaches infinity (sometimes FETs are used for the input stage, as in BIFET op amps). These key features, zero input voltage and infinite input impedance, make the inverting input a virtual ground point.

Virtual ground differs from an ordinary ground since the ordinary ground zero voltage while sinking any amount of current. A virtual ground, however, is a ground for voltage but not for ground; it has zero voltage but can sink no current. In the discussion that follows, we will approximate the inverting input of an op amp as a virtual ground point: this means zero voltage and zero current.

Output Voltage and Current

We consider an inverting op amp with input and output resistors. V_{in} is the input voltage with respect to ground, and V_{out} is the output voltage with respect to ground. Because of the high and input impedance, we can approximate the inverting input as a virtual ground point. Therefore, all the input voltage appears across the input resistor, which means that the input current is

$$I=V_{in} / R_{in}$$

Since none of the input current can enter the virtual ground point, it must pass through the output resistor. In other words, the output current equals the input current. And the output voltage is

$$V_{out}= -I_{Rout}$$

The minus sign indicates phase inversion, if the input voltage is positive, the output voltage is negative.

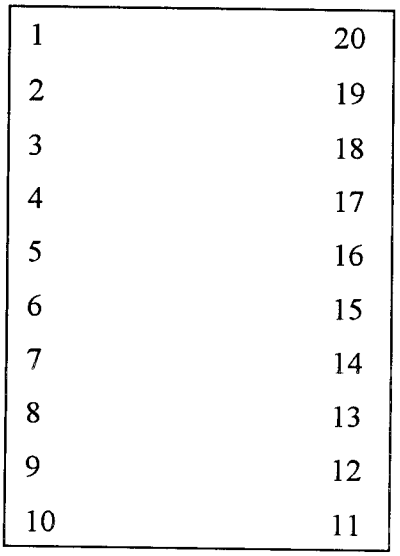
3.3 The ADC0808

A/D converters are commercially available as integrated circuits with 8- to 16-bit resolution. This section introduces the ADC0808, an 8-bit A/D converter that is easily interfaced to an 8080 or 8085. The device uses successive approximation an analog input (0 to 5V) to an 8-bit digital equivalent. The ADC0808 has an on-chip clock generator, needs a supply of only +5V, and has an optimum conversion time of approximately 100 μs.

20 pin outs

9) Vref

10) GND



1) CS/ *11 to 18) D7 to D0

19) CLK R 20) Vcc

2) RD/ *

3) WR/ *

4) CLK IN

5) INTR/ *

6) Vin (+)

7) Vin (-)

* The / implies the "Bar" which gives the negative value

Pinout Diagram

Figure shows the pinout. Pins 11 to 18(digital output) are a three state output; this allows a direct connection to the address-data bus if desired. If CS/ or RD/ pin is high, pins 11 to 18 float; when CS/ and RD/ are both low, the digital output appears on the output lines.

The start-of-conversion signal has been labeled WR/ (pin 3). To start a conversion, CS/ must be low. When WR/ goes low, the converter is reset; when WR/ returns to the high state, the conversion begins.

The converter clock frequency must be in the range of 100 to 800 kHz. The CLK IN (pin 4) may be derived from the CPU clock. If the system clock is running at a frequency greater than 800 kHz, we can divide it down with one or more flip-flops. Alternatively, we can use an on-chip clock generator by connecting an external RC circuit between CLK IN (pin 4) and CLK R (pin 19).

Pin 5 is INTR/, the end-of-conversion signal. INTR/ goes high at the start of a conversion; it is asserted (made active low) when the conversion is finished. The falling edge of INTR/ can be used to interrupt a microprocessor, which then branches to a service subroutine processing the converter output.

Pins 6 and 7 are a differential input for the analog signal; this type of input allows us to ground pin 7 for single ended positive input, to ground

pin6 for single ended negative input, or to drive both pins for differential input.

The device has two grounds, A GND (pin 8) and D GND (pin 10). Both must be grounded. Pin 20 is the supply voltage, +5 V in a microprocessor- based system.

Pin 9:

In the ADC0808, Vref is the maximum analog input voltage; this is the voltage that produces a maximum output of FFH. If pin 9 is unconnected, Vref equals the supply voltage Vcc.This means that a supply of +5V allows an analog input range of 0 to +5V for single-ended positive input (input on pin 6 with pin 7 grounded).

In some applications we may prefer a different analog range. This is where pin 9 comes in. The voltage we connect to pin 9 overrides the supply voltage and controls the maximum analog input voltage. If we want a maximum analog input of +4 V, for instance, we must apply +2 V to pin 9.

3.4 The 8051 Micro Controller

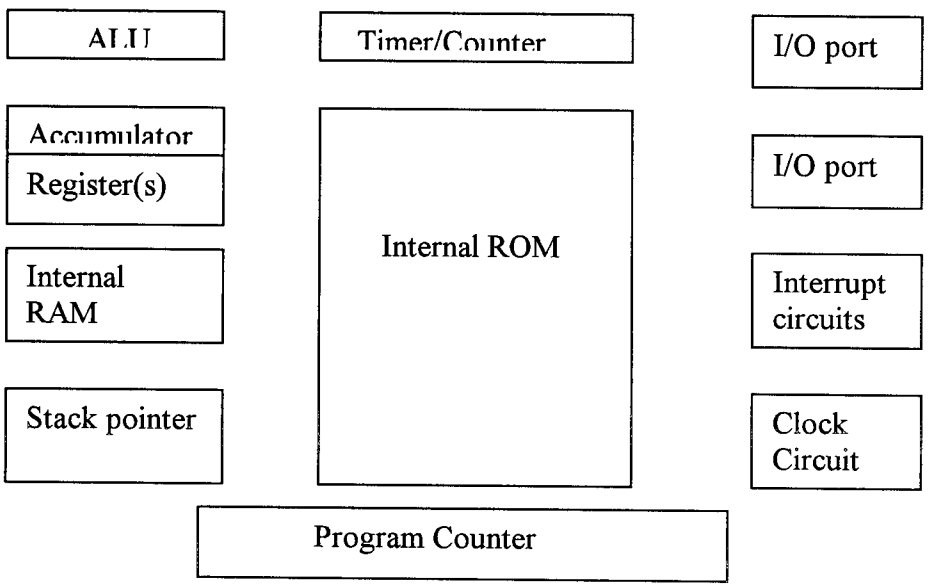


Figure shows the block diagram of a typical micro controller (MC), which is a true computer on a chip. The design incorporates all of the features found in a microprocessor (MP) CPU: ALU, PC, SP and registers. It also has added the other features needed to make a complete computer: ROM, RAM, parallel I/O, serial I/O, counters, and a clock circuit.

Like the microprocessor, a MC is a general-purpose device, but one that is meant to read data, perform limited calculations on that data, and control its environment based on those calculations. The prime use of a MC is to control the operation of a machine using a fixed program that is stored in ROM and that does not change over the lifetime of the system.

The design approach of the MC mirrors that of the MP: make a single design that can be used in as many applications as possible in order to sell as many as possible. The MP design accomplishes this goal by having a very flexible and extensive repertoire of multi- byte instructions. These Instructions work in a hardware configuration that enables large amounts of memory and I/O to be connected to address and data bus pins on the integrated circuit package. Much of the activity in the MP has to do with moving code and data to and from external memory to the CPU. The architecture features working registers that can be programmed to take part in the memory access process, and the instruction set is aimed at expediting this activity in order to improve throughput. The pins that connect the MP to external memory are unique, each having a single function. Data is handled in byte, or larger, sizes.

The MC design uses a much more limited set of single and double-byte instructions that are used to move code and data from internal memory

to the ALU. Many instructions are coupled with pins on the integrated circuit package; the pins are "programmable" – that is, capable of having several different functions depending on the wishes of the programmer.

The MC is concerned with getting data from and to its own pins; the architecture and instruction set are optimized to handle data in bit and byte size.

Micro controller Hardware:

The 8051 MC generic part number actually includes a whole family of MCs that have numbers ranging from 8031 to 8751 and is available in N-channel Metal Oxide Silicon (NMOS) and Complementary Metal Oxide Silicon (CMOS) construction in a variety of package types. An enhanced version of the 8051, the 8052, also exists with its own family of variations and even includes one member that can be programmed in BASIC. Some unique features of a micro controller are,

Internal ROM and RAM

I/O ports with programmable pins

Timers and counters

Serial data communication

Program counter, ALU, working registers, and Clock circuits.

The 8051 architecture consists of these specific features:

1. 8-bit CPU with registers A (the accumulator) and B
2. 16-bit program counter and data pointer
3. 8-bit program status word
4. 8-bit stack pointer
5. Internal ROM or EPROM of 0 to 4K
6. Internal RAM of 128 bytes:

a) 4 register banks, each containing 8 registers

b) 16 bytes, which may be addressed at the bit level

c) 80 bytes of general purpose data memory

7. 32 I/O pins arranged as four 8-bit ports: P0-P3
8. Two 16-bit timer/counters: T0 and T1
9. Full duplex serial data receiver/transmitter: SBUF
10. Control registers: TCON, TMOD, SCON, PCON, IP, and IE
11. Two external and three internal interrupt sources
12. Oscillator and clock circuits

All the 8-bit and 16-bit registers and 8-bit memory locations can be made to operate using the software instructions that are incorporated as part of the design. The program instructions have to do with the control of the registers and digital data paths that are physically contained inside the 8051, as well as memory locations that are physically located outside the 8051.

The number of special purpose registers complicates the model that must be present to make a microcomputer a MC. Most of the registers have a specific function; those that do occupy an individual block with a symbolic name, such as A or TH0 or PC. Others, which are generally indistinguishable from each other, are grouped in a larger block, such as internal ROM or RAM memory.

Each register, with the exception of the program counter, has an internal 1-byte address assigned to it. Some registers are both byte and bit addressable. That is, the entire byte of data at such register addresses may be read or altered, or individual bits may be read or altered. Software

instructions are generally able to specify a register by its address, its symbolic name, or both.

Input/Output Pins, Ports, and Circuits:

One major feature of a MC is the versatility built into the I/O circuits that connect the 8051 to the outside world. The MP designs must add additional chips to interface with external circuitry; this ability is built into the MC.

To be commercially viable, the 8051 had to incorporate as many functions as were technically and economically feasible. The main constraint that limits numerous functions is the number of pins available to the 8051 circuit designers. The DIP has 40 pins, and the success of the design in the marketplace was determined by the flexibility built into the use of these pins.

For this reason, 24 of the pins may each be used for one of two entirely different functions, yielding a total pin configuration of 64. The function a pin performs at any given instant depends, first, on what is physically connected to it and, then on what s/w commands are used to "program" the pin. Both of these factors are under the complete control of the 8051 programmers and circuit designer.

Given this pin flexibility, the 8051 may be applied simply as a single component with I/O only, or it may be expanded to include additional memory, parallel ports, and serial data communication by using the alternate pin assignments.

There are two data paths in the pin port circuitry that read the latch or pin data using two entirely separate buffers. The upper buffer is enabled when latch data is read, and the lower buffer, when the pin state is read. The status of each latch may be read from a latch buffer, while an input buffer is

connected directly to each pin so that the pin status may be read independently of the latch state.

Different opcodes access the latch or pin states as appropriate. Port operations are determined by the manner in which the 8051 are connected to external circuitry.

Programmable port pins have completely different alternate functions. The configuration of the control circuitry between the output latch and the port pin determines the nature of any particular port pin function. The ports are not capable of driving loads that require currents in the tens of milliamperes.

Port 0:

Port 0 pins may serve as inputs, outputs, or when used together, as a bi-directional low order address and data bus for external memory. For example, when a pin is to be used as an input, a 1 must be written to the corresponding port 0 latch by the program, thus turning both of the output transistors off, which in turn causes the pin to "float" in a high impedance state, and the pin is essentially connected to the input buffer.

When used as an output, the pin latches that are programmed to a 0 will turn on the lower FET, grounding the pin. All latches that are programmed to a 1 still float; thus, external pull-up resistors will be needed to supply logic high when using port 0 as an output.

When port 0 is used as an address bus to external memory, internal control signals switch the address lines to the gates of the Field Effect Transistories (FETs). Logic 1 on an address bit will turn the upper FET on and the lower FET off to provide logic high at the pin. When the address bit is zero, the lower FET is on and the upper FET off to provide a logic low at

the pin. After the address has been formed and latched into external circuits by the Address Latch Enable (ALE) pulse, the bus is turned around to become a data bus. Port 0 now reads data from the external memory and must be configured as an input, so logic 1 is automatically written by internal control logic to all port 0 latches.

Port 1:

Port 1 pins have no dual functions. Therefore, the output latch is connected directly to the gate of the lower FET, which has an FET circuit labeled Internal FET Pull-up as an active pull-up load.

Used as an input, a 1 is written to the latch, turning the lower FET off; the pin and the input to the pin buffer are pulled high by the FET load. An external circuit can overcome the high impedance pull-up and drive the pin low to input a 0 or leave the input high for a 1.

If used as an output, the latches containing a 1can drive the input of an external circuit high through the pull-up. If a 0 is written to the latch, the lower FET in on, the pull-up is off, and the pin can drive the input of the external circuit low.

To aid in speeding up switching times when the pin is used as an output, the internal FET pull-up has another FET in parallel with it. The second FET is turned on for two oscillator time periods during a low-to-high transition on the pin. This arrangement provides a low impedance path to the positive voltage supply to help reduce rise times in charging any parasitic capacitances in the external circuitry.

Port 2:

Port 2 may be used as an input/output port similar in operation to port 1. The alternate use of port 2 is to supply a high order address byte in conjunction with the port 0 low order byte to address external memory.

Port 2 pins are momentarily changed by the address control signals when supplying the high byte of a 16-bit address. Port 2 latches remain stable when external memory is addressed, as they do not have to be turned around (set to 1) for data input as in the case for port 0.

Port 3:

Port 3 is an I/O port similar to port1. The input and output functions can be programmed under the control of the P3 latches or under the control of various other special function registers.

Unlike ports 0 and 2, which can have external addressing functions and change all eight-port bits when in alternate use, each pin of port 3 may be individually programmed to be used either as I/O or as one of the alternate functions.

Serial Data Input/Output:

Computers must be able to communicate with other computers in modern multiprocessor distributed systems. One cost effective way to communicate is to send and receive data bits serially. The 8051 have a serial data communication circuit that uses register SBUF to hold data. Register SCON controls data communication, register PCON controls data rates, and pins RXD and TXD connect to serial data network.

* Serial Data Interrupts

Serial data communication is a relatively slow process, occupying many milliseconds per data byte to accomplish. In order not to tie up valuable processor time, Serial Data flags are included in SCON to aid in efficient data transmission and reception. The data transmission is under the complete control of the program, but reception of data is unpredictable and at random times that are beyond the control of the program.

The serial data flags in SCON, TI, and RI are set whenever a data byte is transmitted or received. These flags are ORed together to produce an interrupt to the program. The program must read these flags to determine which caused the interrupt and then clear the flag. This is unlike the timer flags that are cleared automatically; it is the responsibility of the programmer to write routines that handle the serial data flags.

* Data Transmission

Transmission of serial data bits begins anytime data is written to SBUF. TI is set to a 1 when the data has been transmitted and signifies that SBUF is empty and that another data byte can be sent. If the program fails to wait for the TI flag and overwrites SBUF while a previous data byte is in the process of being transmitted, the results will be unpredictable.

* Data Reception

Reception of serial data will begin if the receive enable bit (REN) in SCON is set to 1 for all modes. In addition, for mode 0 only, RI must be cleared to zero. Receiver Interrupt flag RI is set after data has been received in all modes. Setting REN is the only direct program control that limits the

reception of unexpected data; the requirement that RI also be 0 for mode 0 prevents the reception of new data until the program has dealt with the old data and reset RI.

Reception can begin in modes 1, 2, and 3 if RI is set when the serial stream of bits begins. The program must have reset RI before the last bit is received or the incoming data will be lost. Incoming data is not transferred to SBUF until the last data bit has been received so that the previous transmission can be read from SBUF while new data is being received.

* There are four Serial Data Transmission Modes

1.Shift register mode

2.Standard UART

3.Multiprocessor mode

4.Serial data mode 3

Addressing Modes

There are four types of addressing modes: immediate, register, direct, and indirect.

Immediate Addressing Mode:

The simplest way to get data to a destination is to make the source of the data part of the opcode. The data source is then immediately available as part of the instruction itself.

When the 8051 execute an immediate data move, the program counter is automatically incremented to point to the bytes following the opcode byte in the program memory. Whatever data is found there is copied to the destination address.

The mnemonic for immediate data is the pound sign (#). Occasionally, in the rush to meet a deadline, we might forget to use the # for

immediate data. The resulting opcode is often a legal command that is assembled with no objections by the assembler. This omission guarantees that the rush will continue.

Register Addressing Mode:

Certain register names may be used as part of the opcode mnemonic as sources or destinations of data. Registers A, DPTR, and R0 to R7 may be named as part of the opcode mnemonic. Other registers in the 8051 may be addressed using the direct addressing mode.

Some opcode mnemonics:

MOV A, #n ➔Copy the immediate data byte n to the A register

MOV A, Rr ➔Copy data from register Rr to register A

MOV Rr, A ➔ Copy data from register A to register Rr

Cautions:

- Impossible to have immediate data as a destination
- All numbers must start with a decimal number or the assembler assumes the number is a label
- Register to register moves using the register addressing mode occur between registers A and R0 to R7

Direct Addressing Mode:

All 128 bytes of internal RAM and the SFRs may be addressed directly using the single byte address assigned to each RAM location and each special function register. RAM addresses 00 to 1Fh are also the locations assigned to the four banks of 8 working registers, R0 to R7. This assignment means that R2 of register bank 0 can be addressed in the register mode as R2 or in the direct mode as 02h. The

direct addresses of the working registers are ranging from 00 to 1F. Only one bank of working registers is active at any given time. The PSW special function register holds the bank-select bits, RS0 and RS1, which determine which register bank is in use.

The moves made possible using direct mode are as follows:

MOV A, add ➔Copy data from direct address add to register A

MOV add, Rr ➔Copy data from Rr register to direct address

MOV add, #n ➔Copy immediate data n to direct address add

Indirect Addressing Mode:

The indirect mode uses a register to hold the actual address that will finally be used in the data move; the register itself is not the address, but the number in the register. Indirect addressing for MOV opcodes uses register R0 or R1, often called a data pointer, to hold the address of one of the data locations in RAM from address 00h to 7Fh. The number that is in the pointing register cannot be known unless the history of the register is known. The mnemonic symbol used for indirect addressing is the "at" sign, which is printed as @.

The moves made possible using indirect mode are as follows:

MOV A, @Rp ➔ Copy the contents of the address in Rp to A

MOV @Rp, #n➔ Copy immediate byte n to the address in Rp

MOV add, @Rp➔Copy the content of the address in Rp to add

Caution:

1. The number in register Rp must be a RAM address
2. Only registers R0 and R1 may be used for indirect mode

3.5 RS-232 MAX Convertor

RS-232 was introduced in 1960, and is currently the most widely used communication protocol. It is simple, inexpensive to implement, and though relatively slow,it is more than adequate for more simple serial communication devices such as keyboards and mice. Some of the unique features of RS-232 are,

- Single ended data transmission system
- One driver per line
- One receiver per line
- Maximum cable length is 50 feet
- Maximum data rate is 20 kbps
- Driver output maximum voltage is –25 to +25v
- Driver output signal level is –5 to +5v(loaded)
- Driver output signal level is –15 to +15v(unloaded)
- Driver load impedance is 3kOhm to 7 kOhm
- Max. Driver output current is Vmax/300 ohm (power off)
- Slew rate is 30 V/micro sec (max)
- Receiver input voltage range is –15v to +15v
- Receiver input sensitivity is –3 to +3v
- Receiver input resistance is 3kOhm to 7kOhm

DTE and DCE: serial communication partners

A typical system is made up of 2 types of device, data communication equipment (DCE) and data terminal equipment (DTE). DTE is defined as the communication source and DCE is defined as the device that provides a communication channel between two DTE-type devices. In order for DCE and DTE to communicate with each other, two wires must be

used –one for transmission and another for reception – and both devices must not use the same wire for the same purpose.

Connectors for RS-232 devices are always constructed using standard assignments for the wires in a RS-232 cable in order to maintain the DTE-DCE relationship. These connectors can be modular (phone jack) or male D-shell.

RS-232 Signal Descriptions:

Abbreviations used in the following definitions of RS-232 wire functions are

DTR: Data Terminal Ready--Used by a DTE to signal that it is plugged in and available to begin communication.

DSR: Data Set Ready--Sister signal to DTR, it is used by the DCE to indicate it is ready to begin communication.

CTS: Clear To Send--Used by DCE to signal it is available to send data, and used in response to RTS request for data.

RTS: Request To Send--used by DTE that it wants to send data. Also, in a multi-drop network, used to turn carrier on the modem on and off.

DCD: Data Carrier Detect--Used by a DCE to indicate to the DTE that it has received a carrier signal from the modem and that real data is being transmitted.

RI: Ring Indicator--Used by DCE modem to tell the DTE that the phone is ringing and that will be forthcoming.

TxD: Transmit Data--This wire is used for sending data.

RxD: Receive Data--This line is used for receiving data.

GND: Signal Group--This pin is the same for DTE and DCE devices, and it provides the return path for both data and handshake signals.

Synchronous Communication only: -

TxCLK: Transmit Signal Element Timing—Used by DTE to provide DCE with timing information for data transfer

RxCLK: Receiver Signal Element Timing—Used by DCE to provide DTE with timing information for data transfer

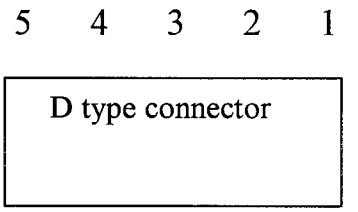
LLBK: Local Loop back—Used by DTE to make sure the local transmit and receive interface is functioning properly

RLBK: Remote Loop back—Used by DTE to make sure a remote transmit and receive interface is functioning properly

TEST MODE: Used by DCE to indicate that it is testing itself in response to a local or remote loop back signal from a DTE.

3.6 D-Type CONNECTOR

A D-type connector can either have 9 pins or 25 pins. There are 2 types of connectors that are needed for transmitting data to the serial port of PC namely male and female connectors.



Pin 2->sending data
Pin 5->ground
Pin 3->receive data
Pin 6->clear to send

3.7 Trimmers

Trimmer is a variable potentiometer that controls the output in accordance with the variation in the input. Suitable trimmers along with some resistors are used to tune the input voltages given, so that we get appropriate outputs at the receiver end.

3.8 Opto Couplers

Opto Coupler is mainly used in the second part of the application that is security system. Since any device if directly given a digital input might result in failure of the device, we use opto coupler so that the input signal is properly controlled and any distortion in the signal is prevented and the device is thus secured.

3.9 Serial RTC

Real Time Clock – This is used to control the timing intervals between two successive outputs.

3.10 Regulated Power Supply

"5V RPS of IC7805" – Since ADC can accept a maximum of +5V, we have to convert the given input of 230V to 5V. First, we use a step down transformer to achieve 12 volts. This 12V RPS is an alternate current. So we use IN4002 diode to convert AC to direct current of 12V. DC is similar to a digital signal and it might have distortions. To eliminate the distortions, two capacitors are connected in parallel, thus we get a pure DC signal.

This 12V signal is to be converted to a 5V DC signal using another diode IN4002. The resulting 5V DC signal is given to the heat sink 7805, which absorbs the excessive heat produced, when the supply is given for a long period of time.

3.11 Limit Switch

Limit switch is similar to an ordinary toggle switch. When the switch is in on state, a signal of 5V goes to the micro controller. Otherwise it will be in an ideal state.

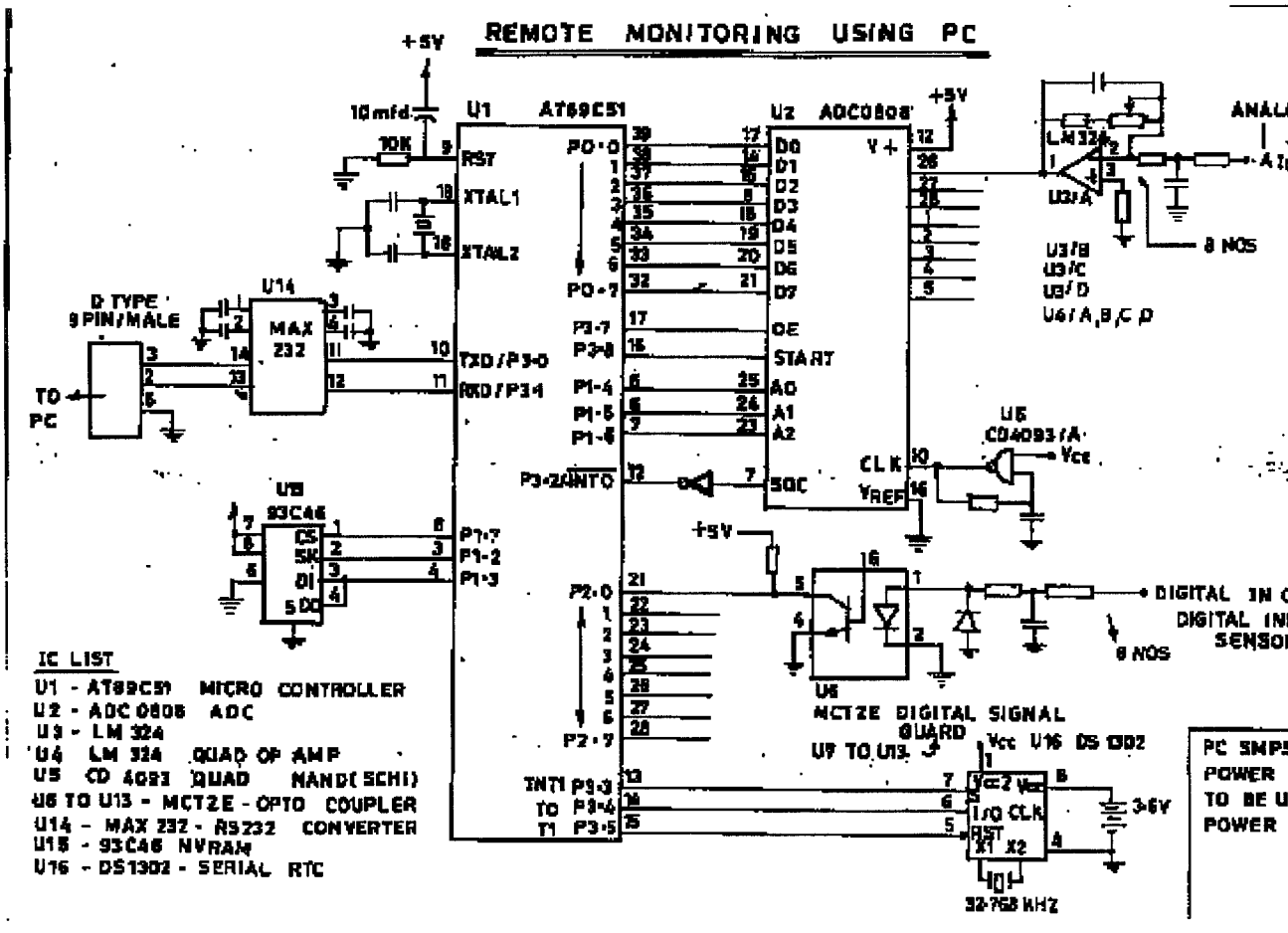
3.12 Other Devices

To complete the circuit, various types of resistors and capacitors are included in the appropriate places. The entire circuit configuration is soldered on a PCB, and connected to a PC for getting the output temperature and security alarm.

DESIGN DOCUMENT

4. DESIGN DOCUMENT

CIRCUIT DIAGRAM:



TEST PLAN

5. TEST PLAN

5.1 Objectives of Testing:

The Objectives of testing are as follows:

a. Testing is a process of executing a program with the intent of finding an error.

b. A good test is one that has a high probability of finding an undiscovered error.

A successful test is one that uncovers an as-yet undiscovered

5.2 SYSTEM TESTING:

The following errors were detected during testing

(1) A pull-up resistor was put up at the input end so that we got maximum voltage all the time.

(2) The transmit and receive pins of D-type connector was given wrongly.

IMPLEMENTATION

6. IMPLEMENTATION

After testing the modules successfully, the necessary privileges are given to users. All the users are requested to handle the system properly. The real time problems that occur are successfully solved.

The objective is to put the tested system into operation. It consists of

a. Testing the developed product with sample inputs

b. Detection and correction of errors

c. Making necessary changes in the system

d. Checking of reports with that of existing system

e. Testing the user personnel

f. Creating computer compatible files

c. Installation of Hardware and Software utilities

CONCLUSIONS & FUTURE ENHANCEMENTS

7. CONCLUSIONS AND FUTURE ENHANCEMENTS

Broad conclusions:

With the development of the proposed system, the purpose of the system is achieved. The goals that have been achieved by the developed system are,

- The temperature has been monitored and recorded regularly.
- Security mechanism for various applications.

Future Enhancements:

This project can be further enhanced to incorporate the following applications,

* Measurement of humidity
* Intensity of rainfall
* Measurement of velocity of wind
* The data which are acquired can be used intelligently in any knowledge based systems and other information systems.

REFERENCES

8. REFERENCES:

(1) www.8051.com (For micro controller).

(2) www.national.com (For ADC).

(3) Micro controller interfacing by Douglas Hall.

(4) 8051 Micro controller programming by Thomas Adaya.

(5) Micro Controller Programming & Interfacing by MervHughes.