MICROCONTROLLER BASED ANNUNCIATOR SYSTEM WITH PC INTERFACING

By

DIVYA CHAUHAN-041139

ANUJ KAPOOR-041144

PRIYANKA SINGH-041147

DECEMBER-2008

Submitted in partial fulfillment of the Degree of Bachelors of Technology

DEPARTMENT OF ELECTRONICS

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, **"MICROCONTROLLER BASED ANNUNCIATOR SYSTEM WITH PC INTERFACING"** submitted by Divya Chauhan, Anuj Kapoor and Priyanka Singh in partial fulfillment for the award of degree of Bachelors of Technology in Electronics of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

NSharma 04/12/08

Ms. Neeru Sharma

DIVYA CHAUHAN 041139 divya chauhan

ANUJ KAPOOR 041144 Anuj kapoor

PRIYANKA SINGH 041147. Priyanka

ACKNOWLEDGEMENT

The past 1 year that we have spent on the project has been a truly remarkable period. It has been a steep learning curve and a valuable experience, which will hold us in good stead in the future.

We are very thankful to **Ms. Neeru Sharma** who guided and helped us on day-to-day basis. We thank her for making us learn the ethics and culture of teamwork and providing us with the necessary facilities without which the accomplishment of the project would otherwise not have been possible.

We would like to take the opportunity to thank the HOD **Prof. S.V.Bhooshan** whose in-depth knowledge of the subject has inspired us to understand the fundamentals of the subject thoroughly rather than blindly adopting the rote method of learning.

TABLE OF CONTENTS

CHAPTER 6: CONCLUSION & FUTURE WORK

APPENDIX:

List of Figures:

ABSTRACT

Today all instrumentation systems pertaining to industrial process control as well as domestic applications involve some type of fault finding facility. This facility detects the faulty condition & draws the attention of the operator towards it. One such method is annunciation. Here we develop a microcontroller based **Annunciator** system to detect the faults. As prerequisites, an understanding of the microcontrollers and in particular, the familiarity with the instruction set of 80C51 is required. Our aim here is to present the entire project in a simple and step-by-step manner so as to equip the reader with a proper understanding of the functioning of the system. Annunciation is a method of activation of a visual or mechanical indicator when the remote switch is activated as a result of a fault. The device used for annunciation is called **ANNUNCIATOR.**

.

CHAPTER 1

INTRODUCTION

TO

EMBEDDED SYSTEMS

1.1 Introduction to Embedded Systems

Embedded systems are found in a variety of common electronics devices, such as Consumer electronics (cell phones, pagers, digital cameras, and calculators), home appliances (microwave ovens, answering machines thermostats), office automation (fax machines, printers, scanners) and automobiles etc.

Embedded system is a combination of hardware and software attached with certain peripherals to carry out a single task. It takes a set of input, processes it and provides the output.

An embedded system is a sub system of a large system performing only a single function. Embedded systems often require real time operation and multitasking capabilities. Real time operation refers to the fact that the embedded controller must be able to receive and process the signals from its environment as they are received. That is, the environment must not wait for the controller to become available. Similarly, the controller must perform fast enough to output control signals to its environment when they are needed. Again, the environment must not wait for the controller. In other words, the embedded controller should not be bottleneck in the operation of the systems. Multitasking is the capability to perform many functions in a simultaneous or quasi-simultaneous manner. The embedded controller is often responsible of monitoring several aspects of a system and responding according when the need arises.

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems control many of the common devices in use today.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

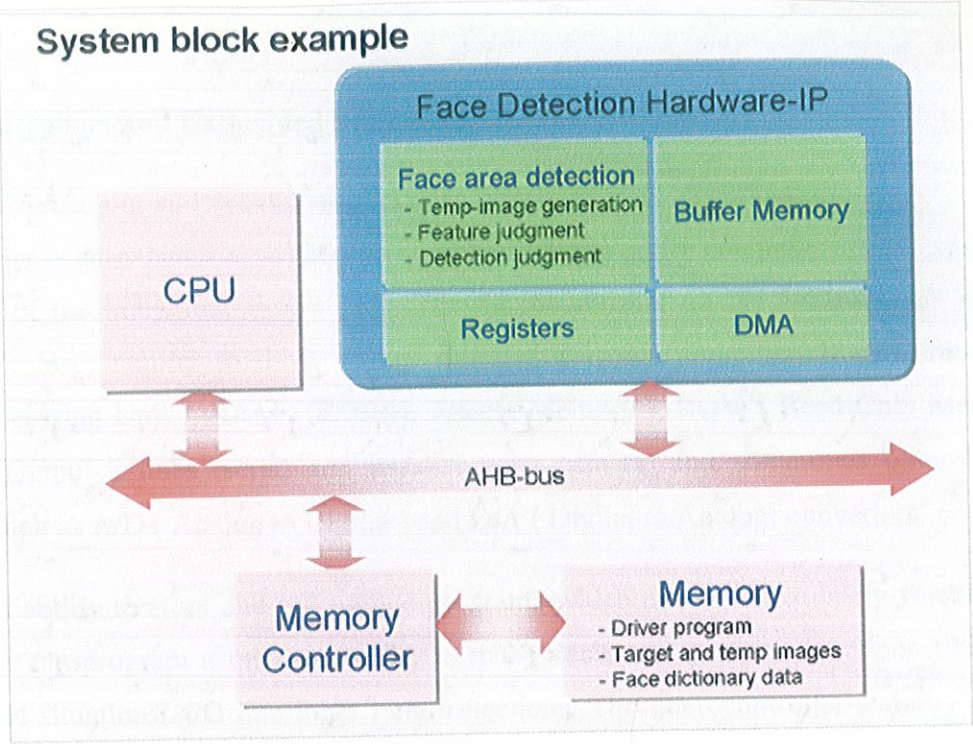


Figure 1.1- Basic module of embedded system

1.2 General Characterstics of Embedded System

- **Single functioned** - An embedded system usually executes a specific program repeatedly.
- **Tightly constrained** - Embedded systems often must cost few dollars, must be sized to lift on a single chip, must perform fast enough to process data in real time and must consume minimum power.
- **Dependable** - Depends on the output produced by the other connected system.
- **Reliable** - The reliablility of an embedded system is increased by introducing redundancy.
- **Real time** - The response of the embedded system should be real-time i.e. the output of the system should occur in specific time.
- **Efficiency** - The embedded system should be efficient, consumes less power, occupy small space, the cost should be less and speed should be more.

1.3 Microcontrollers and Embedded Systems

Microprocessors and microcontrollers are widely used in embedded system products. A microcontroller is an expensive single chip computer i.e. the entire computer system lies within the confines of the integrated circuit chip. The microcontroller on the encapsulated silver of silicon has features similar to those of our standard personal computer. It contains a CPU (Central Processing Unit), RAM (Random Access Memory), ROM (Read Only Memory), I/O (Input/Output) lines ,serial and parallel ports, timers, and sometimes other built in peripherals such as A/D (Analog to Digital) and D/A (Digital to Analog) converters.

It has ability to store and run unique programs which makes it extremely versatile. For instance, one can program a microcontroller to make decision (perform functions) based on predetermined situations(I/O line logic) and selections. The microcontroller's ability to math and logic functions can make the microcontroller behave like a neutral circuit and/or a fuzzy logic controller. Microcontroller are responsible for the "intelligence" in most smart devices on the consumer market. Because of their versatility, microcontrollers add a lot of power, control, and option, and at little cost.

1.3.1 Microprocessor and Microcontroller

A digital computer typically consists of three major components; the Central Processing Unit (CPU), program and data memory, and an Input/ Output (I/O) system.

The CPU controls the flow of information among the components of the computer. It also process the data by performing digital operations. Most of the processing is done in the Arithemetic Logic Unit (ALU) within the CPU. When the SPU of a computer is built on a single printed circuit board, the computer is called a minicomputer. A microprocessor is a CPU that is compacted into a single chip semiconductor device. Microprocessors are general-purpose devices, suitable for many applications. A computer bulit around a microprocessor is called a Microcomputer.

1.3.2 Input/output (I/O) System

In order to serve various applications, microcontrollers have a high concentration of on chip facilities such as serial port, parallel input output ports, timers, counters, interrupt control, analog to digital converters, random access memory, read only memory, etc. The I/O, memory, and on chip peripherals of a microcontroller are selected depending on the specifies of the target application. Since microcontrollers are powerful digital processors, the degree of control and programmability they provide significantly enhances the effectiveness of the applications.

8051 is the first microcontroller of the MCS 51 family introduces by Intel Corporation at the end of the 1950s. the 8051 family with its many enhanced members enjoy the largest market share, estimated to be about 40%, among the various microcontroller architectures,

The architecture of 8051 family of the microcontrollers is presented in the chapter. First, the original 8051 microcontroller is discussed, followed by the enhanced features of the 8032, and the 80C515.

1.4 The 8051 Microcontroller Architecture

The architecture of the 8051 family of microcontrollers is referred to as the MCS-51 architecture, or sometimes simply as MCS-51. The microcontrollers have an 8-bit data bus. They are capable of addressing 64K of program memory and a separate 64K of data memory. The 8051 has 4K of code memory implemented as on-chip Read Only Memory (ROM). The 8051 has 128 bytes of internal Random Access Memory (RAM). The 8051 has two timer/counters, a serial port, 4 general purpose parallel input/output ports, and interrupt control logic with five sources of interrupts besides internal RAM, 8051 has various Special Function Registers (SFR), which are the control and data registers for on- chip facilities. The SFRs also include the accumulator, the B register, and the Programs Status Word (PSW), which contains the CPU flags. Programming the various internal hardware facilities of the 8051 is achieved by placing the appropriate control words in the corresponding SFRs. The 8031 is similar to the 8051, except it lacks the on-chip ROM. As stated, the 8051 can address 64K of external data memory and 64K of external program memory. These may be separated blocks of memory, so that up to 128K of

memory can be attached to the microcontroller. Separate blocks of code and data memory are referred to as the Harvard architecture. The 8051 has two separate read signals, RD#(P3.7) and PSEN#. The first is activated when a byte is to be read from external data.

All external code is fetched from external program memory. In addition, bytes from external program memory may be read by special read instructions such as the MOVC instruction. That is, the instruction determines which block of memory is addressed, and the corresponding control signal, either RD# or PSEN# is activated during the memory read cycle. A single block may be mapped to act as both data and program memory. This is referred to as the Von Neumann architecture. In order to read from the same block using either the RD# signal or the PSEN# signal, the two signals are combined with a logic AND operation.

This way, the output of the AND gate is low when either input is low. The advantage of the Harvard architecture is not simply doubling the memory capacity of the microcontroller. Separating program and data increases the reliability of the microcontroller, since there are no instructions to write the program memory. A ROM device is ideally suited to serve as program memory. The Harvard architecture is somewhat awkward in evaluation systems, where code needs to be loaded into the program memory. By adopting the Von Neumann architecture, code may be written as data bytes, and then executed as program instructions. The 8052 has 256 bytes of internal RAM and 8K of internal code ROM. The 8051 and 8052 internal ROM cannot be programmed by the user. The user must supply the program to the manufacturers programs the microcontrollers during production. Due to setup costs, the factory masked ROM option is not essential for small quantity productions. The 8751 and 8752 are Erasable Programmable Read Only Memory (EPROM) version of the 8051 and 8052. Many manufacturers offer the EPROM versions in windowed ceramic and non-windowed plastic packages. These are user programmable. However, the non-windowed versions cannot be erased. These are usually referred to as One-Time-Programmable (OTP), which are more suitable to experimental work or for small production runs. The 8951 and 8952 contains FLASH EPROM (Electrically Erasable Programmable Read Only Memory). These chips can be programmed as EPROM versions, using a chip Programmer. Moreover, the memory may be erased. Similar to EPROM , erasing FLASH memory sets all data bits (data becomes FFh). A bit may be cleared (made 0) by programming. However, a zero bit may not be programmed to a one. This requires erasing the

chip. Some larger FLASH memories are organized in banks or sectors. Rather than erasing the entire chip, one may erase a given sector and keep the remaining sectors unchanged.

During the past decade, many manufactures introduced enhanced members of the 8051 microcontroller. The enhancements include more memory, more ports, analog-to-digital converters, more timers with compare, reload and capture facilities, more interrupt sources, higher precision multiply and divide units, idle and power down mode support, watchdog timers, and network communication subsystems. All microcontroller of the family use the same set of machine instructions, the MCS-51. The enhanced features are programmed and controlled by additional SFRs. If the program fits into the on-chip ROM and if the internal RAM is sufficient, the MCS-51 family of microcontrollers requires no additional logic to implement a complete controller system.

1.5 Motivation

The motivation for our project comes from the fact that in today's world all instrumentation systems pertaining to industrial process control as well as domestic application, like elevator control involve some type of automatic fault finding facility. This facility detects the faulty condition of the system and draws the attention of the operand towards it, enabling him to take suitable action. One such method is annunciation in which activation of a visual or a mechanical indicator (called Annunciator) takes place when a remote switch or device has been activated as a result of fault in certain parts of the system. Here we present a microcontroller based Annunciator system that detects up to ten different faulty conditions and informs the operator about them. Each annunciation results in glowing of one of the LED's and an audible sound from a buzzer to drive the attention of the operator towards the faults.

1.6 Previous Work

Annunciators have been used through decades for fault detection. These have been mechanical indicators with complex circuitry and expensive part costs. The sizes of these systems have been a great cause of concern especially for small-scale applications. The system

we came across used an 8031 microcontroller that requires an external EEPROM (Electrically Erasable Programmable Read Only Memory). We modified the system to incorporate 89C51RD2 microcontroller that has an in-built 64k programmable flash memory. Thus the overall cost was brought down to few hundreds of rupees with the development of a system that was compact in size and could be built using software that are available free of cost.

CHAPTER 2

REQUIREMENTS AND SPECIFICATIONS

2.1 Basic Requirements of Annunciator Circuit

The Annunciator system detects up to 10 faulty conditions, which leads to closing of one of the ten links through the relay contacts. Here only faults pertaining for more than 15 milliseconds are considered as critical and faults persisting for less than 15 milliseconds are rejected. When the current flows through a particular line, the corresponding pin connected to the microcontroller becomes high. The microcontroller program scans this high pin continuously and displays it at the output port connected to the LED's and the monitor screen. An error message is displayed on the monitor and the corresponding LED glows. Hence the system requires the development of both the hardware and the firmware. The specifications of the various hardware devices as the IC's used and the software used are presented in this chapter.

2.2 Hardware Requirements

An Annunciator circuit consists of the following basic components:

- A set of inputs (16 or 10 depending upon the requirements) to be scanned and controlled. The inputs include contact switches, relays etc.
- Input buffer and/or isolating devices to interface the actual inputs to the microcontroller.
- A microprocessor/microcontroller to look after the entire fault finding process such as reading the inputs, interpreting the faults, outputting the corresponding fault condition with audible alarm, and self –testing of the system itself to ensure the overall functioning of the system.
- Output indication devices such as LED's, relays, and audible alarm along with their interfacing devices (like latches).
- Various resistors, capacitors, jumpers, switches and connecting wires.

2.3 Software Requirements

The various software requirements of the system are:

- A microcontroller simulator program for simulating the target microcontroller
- In-system programmer for downloading the assembly language code into the target microcontroller.
- Firmware: A set of routines such as test, accept etc. to carry out the system functions.

2.4 System Specifications

The specifications of the hardware components and the software used are given here.

2.4.1 Hardware specifications:

The following is the list of various components used in the development of the system:-

1.*Semiconductors*

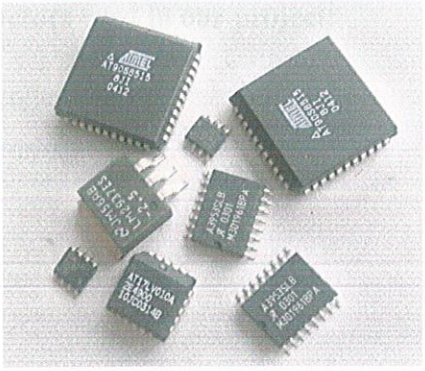


Figure 2.1- Semiconductors

A semiconductor is a substance, usually a solid chemical element or compound, that can conduct electricity under some conditions but not others, making it a good medium for the control of electrical current. Its conductance varies depending on the current or voltage applied

to a control electrode, or on the intensity of irradiation by infrared (IR), visible light, ultraviolet (UV), or X rays. The specific properties of a semiconductor depend on the impurities, or dopants, added to it. An N-type semiconductor carries current mainly in the form of negatively-charged electrons, in a manner similar to the conduction of current in a wire. A P-type semiconductor carries current predominantly as electron deficiencies called holes. A hole has a positive electric charge, equal and opposite to the charge on an electron. In a semiconductor material, the flow of holes occurs in a direction opposite to the flow of electrons.

Elemental semiconductors include antimony, arsenic, boron, carbon, germanium, selenium, silicon, sulfur, and tellurium. Silicon is the best-known of these, forming the basis of most integrated circuits (ICs). Common semiconductor compounds include gallium arsenide, indium antimonite, and the oxides of most metals. Of these, gallium arsenide (GaAs) is widely used in low-noise, high-gain, weak-signal amplifying devices

A semiconductor device can perform the function of a vacuum tube having hundreds of times its volume. A single integrated circuit (IC), such as a microprocessor chip, can do the work of a set of vacuum tubes that would fill a large building and require its own electric generating plant.

Following are the few semiconductors used in our project:

- IC1-89C51RD2 microcontroller
- IC2-ULN2803 octal buffer/driver
- T1-CL100 npn transistor
- LED1-LED10-Red LED

2.Resistors

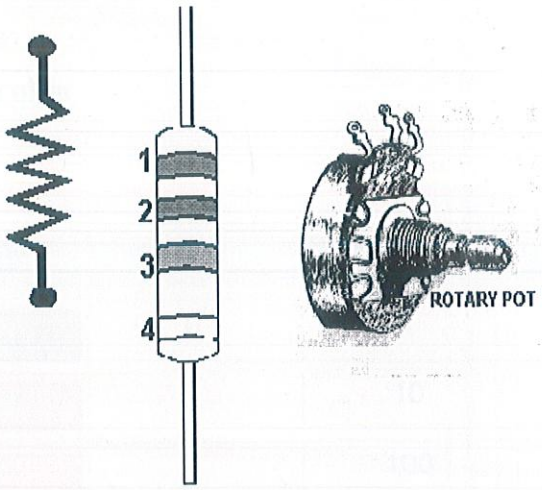


Figure2.2- Resistors

Resistance is the opposition of a material to the current. It is measured in Ohms . All conductors represent a certain amount of resistance, since no conductor is 100% efficient. To control the electron flow (current) in a predictable manner, we use resistors. Electronic circuits use calibrated lumped resistance to control the flow of current. Broadly speaking, resistor can be divided into two groups viz. fixed & adjustable (variable) resistors. In fixed resistors, the value is fixed & cannot be varied. In variable resistors, the resistance value can be varied by an adjuster knob. It can be divided into

(a) Carbon composition (b) Wire wound (c) Special type.

The most common type of resistors used in our projects is carbon type. The resistance value is normally indicated by colour bands. Each resistance has four colours, one of the band on either side will be gold or silver, this is called fourth band and indicates the tolerance, others three band will give the value of resistance (see table). For example if a resistor has the following marking on it say red, violet, gold. Comparing these coloured rings with the colour code, its value is 27000 ohms or 27 kilo ohms and its tolerance is ±5%. Resistor comes in various sizes (Power rating). The bigger, the size, the more power rating of 1/4 watts.

Following are the few resistors that we have used in our project.

- R1-R11- **4.7 kilo ohm**
- R12 - **10 kilo ohm**
- R13- **6.8 kilo ohm**
- R14-R21- **1.8 kilo ohm**

Colour	Digit	Multiplier	Tolerance
Black	0	1	
Brown	1	10	± 1%
Red	2	100	± 2%
Orange	3	1K	
Yellow	4	10K	
Green	5	100K	± 0.5%
Blue	6	1M	± 0.25%
Violet	7	10M	± 0.1%
Grey	8		
White	9		
Gold		0.1	± 5%
Silver		0.01	± 10%
None			± 20%

Figure 2.3- Resistor colour coding

4.7kilo ohm 10kilo ohm 6.8kilo ohm 1.8kilo ohm

Figure 2.4- Resistors used in the project

3. Diode:

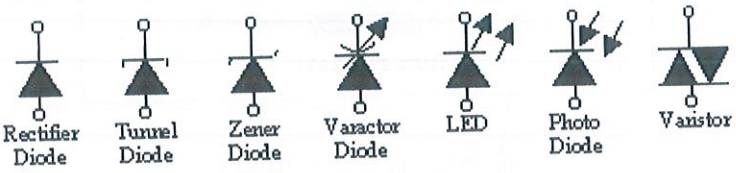


Figure 2.5- Various diodes

The simplest semiconductor device is made up of a sandwich of P-type semiconducting material, with contacts provided to connect the p-and n-type layers to an external circuit. This is a junction Diode. If the positive terminal of the battery is connected to the p-type material (cathode) and the negative terminal to the N-type material (Anode), a large current will flow. This is called forward current or forward biased. If the connections are reversed, a very little current will flow. This is because under this condition, the p-type material will accept the electrons from the negative terminal of the battery and the N-type material will give up its free electrons to the battery, resulting in the state of electrical equilibrium since the N-type material has no more electrons. Thus there will be a small current to flow and the diode is called Reverse biased.

Thus the Diode allows direct current to pass only in one direction while blocking it in the other direction. Power diodes are used in concerting AC into DC. In this, current will flow freely during the first half cycle (forward biased) and practically not at all during the other half cycle (reverse biased). This makes the diode an effective rectifier, which convert ac into pulsating dc.

Signal diodes are used in radio circuits for detection. Zener diodes are used in the circuit to control the voltage.

Some common diodes are:-

- Zener diode.
- Photo diode.
- Light Emitting diode.

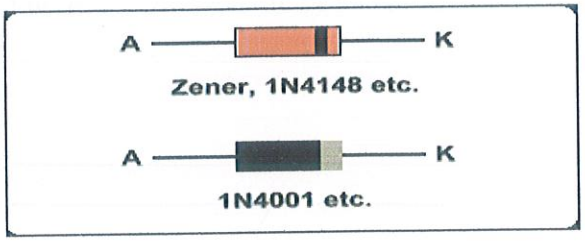


Figure 2.6- Zener diode

4. Crystal oscillators:

Crystal oscillators are oscillators where the primary frequency determining element is a quartz crystal. Because of the inherent characteristics of the quartz crystal the crystal oscillator may be held to extreme accuracy of frequency stability. Temperature compensation may be applied to crystal oscillators to improve thermal stability of the crystal oscillator.

Crystal oscillators are usually, fixed frequency oscillators where stability and accuracy are the primary considerations. For example it is almost impossible to design a stable and accurate LC oscillator for the upper HF and higher frequencies without resorting to some sort of crystal control. Hence the reason for crystal oscillators.

The frequency of older FT-243 crystals can be moved upward by crystal grinding.

I won't be discussing frequency synthesizers and direct digital synthesis (DDS) here. They are particularly interesting topics to be covered later.

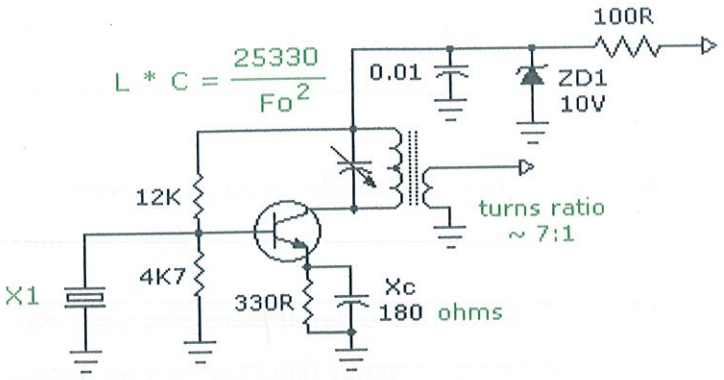


Figure2.7 - Schematic of a crystal oscillator

5. Capacitors:

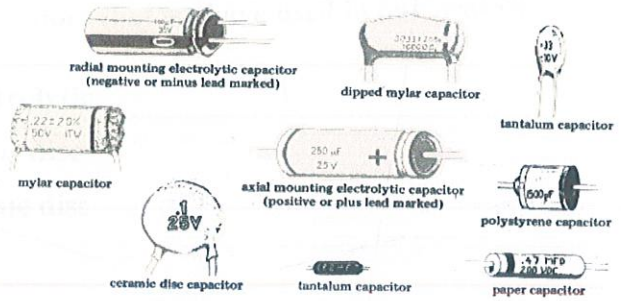


Figure 2.8- Capacitors

It is an electronic component whose function is to accumulate charges and then release it. To understand the concept of capacitance, consider a pair of metal plates which all are placed near to each other without touching. If a battery is connected to these plates the positive pole to one and the negative pole to the other, electrons from the battery will be attracted from the plate connected to the positive terminal of the battery. If the battery is then disconnected, one plate will be left with an excess of electrons, the other with a shortage, and a potential or voltage difference will exists between them. These plates will be acting as capacitors.

Capacitors are of two types: -

(1) **Fixed type** like ceramic, polyester, electrolytic capacitors-these names refer to the material they are made of aluminum foil.

(2) **Variable type** like gang condenser in radio or trimmer. In fixed type capacitors, it has two leads and its value is written over its body and variable type has three leads.

Unit of measurement of a capacitor is farad denoted by the symbol F. It is a very big unit of capacitance. Small unit capacitor are Pico-farad denoted by pf (Ipf=1/1000,000,000,000 f) Above all, in case of electrolytic capacitors, it's two terminal are marked as (-) and (+) so check it while using capacitors in the circuit in right direction. Mistake can destroy the capacitor or entire circuit in operational.

Following are the few capacitors that we have used in our project.

- C1-**10uf, 10 V electrolytic**
- C2-C3-**22pf ceramic disc**
- C4-C6-**0.1uf ceramic disc**

6. Transistors:

The name is transistor derived from 'transfer resistors' indicating a solid state Semiconductor device. In addition to conductor and insulators, there is a third class of material that exhibits proportion of both. Under some conditions, it acts as an insulator, and under other conditions it's a conductor. This phenomenon is called Semi-conducting and allows a variable control over electron flow. So, the transistor is semi conductor device used in electronics for amplitude. Transistor has three terminals, one is the collector, one is the base and other is the emitter, (each lead must be connected in the circuit correctly and only then the transistor will function). Electrons are emitted via one terminal and collected on another terminal, while the third terminal acts as a control element. Each transistor has a number marked on its body. Every number has its own specifications.

There are mainly two types of transistor (i) NPN & (ii) PNP

(i) NPN Transistors:

When a positive voltage is applied to the base, the transistor begins to conduct by allowing current to flow through the collector to emitter circuit. The relatively small current flowing through the base circuit causes a much greater current to pass through the emitter / collector circuit. The phenomenon is called current gain and it is measure in beta.

(ii) PNP Transistor:

It also does exactly same thing as above except that it has a negative voltage on its collector and a positive voltage on its emitter.

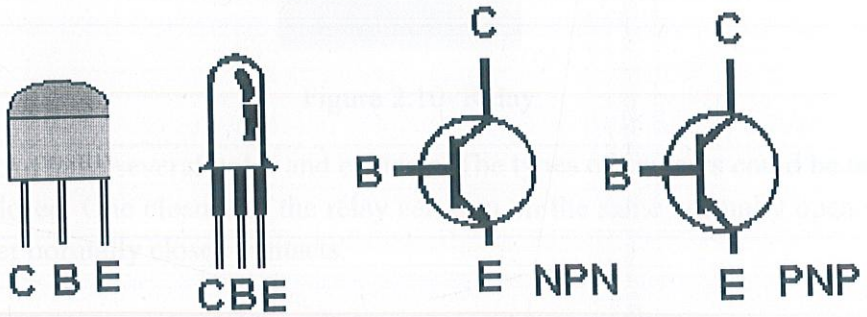


Figure 2.9- Transistors

Transistor is a combination of semi-conductor elements allowing a controlled current flow. Germanium and Silicon is the two semi-conductor elements used for making it. There are two types of transistors such as POINT CONTACT and JUNCTION TRANSISTORS. Point contact construction is defective so is now out of use. Junction triode transistors are in many respects analogous to triode electron tube.

A junction transistor can function as an amplifier or oscillator as can a triode tube, but has the additional advantage of long life, small size, ruggedness and absence of cathode heating power.

7. Relay

Relay is a common, simple application of electromagnetism. It uses an electromagnet made from an iron rod wound with hundreds of fine copper wire. When electricity is applied to the wire, the rod becomes magnetic. A movable contact arm above the rod is then pulled toward the rod until it closes a switch contact. When the electricity is removed, a small spring pulls the contract arm away from the rod until it closes a second switch contact. By means of relay, a current circuit can be broken or closed in one circuit as a result of a current in another circuit.

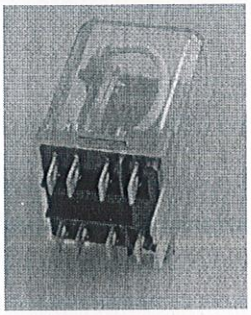


Figure 2.10- Relay

Relays can have several poles and contacts. The types of contacts could be normally open and normally closed. One closure of the relay can turn on the same normally open contacts; can turn off the other normally closed contacts.

8. Miscellaneous:

- S1-S3- Push to on switch
- XTAL-12 MHz crystal
- PZ1-piezobuzzer
- LINK.1-LINK.10-Contact jumpers

2.4.2 Key Components

1.89C51RD2 Microcontroller Evaluation Board

This is main system controller that is describes in detail in this section. The general-purpose evaluation board used is Mini51 Evaluation Board from SPJ Systems. This is a board designed as a development tool that has the facility to download hex file into the On-Chip Flash Code memory of the 89C51RD2 microcontroller without the need of removing the one from the socket.

The Mini51 board has several different modes with number of optional features.

These include:

- 89C51RD2 @ 12 MHz
- LCD interface circuit and a 16 –pin connector.
- Keyboard interface circuit and 12 pin connecting 5X5 matrix keyboard.
- RS-232 serial port with 3 or 9 pin connector.
- Regular to supply +5V to the board.
- DC Adapter (230 VAC input, 9 VDC output, 50 mA).
- 64K 12C compatible EEPROM.
- 12C compatible RTC with battery.

2. Piezoelectric Buzzer

An electromagnetic consists of a single length of wire wrapped in a coil. In a buzzer, the simplest sort of doorbell, an electromagnet is used to operate a self-interrupting circuit. One end of the electromagnet wire is connected directly to one end of the electrical circuit. The other end of the wire connects to a metal contact, which is adjacent to a moving contact arm.

Figure 2.11- Piezoelectric Buzzer

The contact arm is a thin piece of light, conductive metal, with a thin iron bar soldered onto it. The anchored end of the contact arm is wired to the electrical circuit. When the electromagnet is turned off, the free end of the arm resets again the contact point. This forms a connection between that end of the wire and the electrical circuit. In other words electricity can flow through the electromagnet when the circuit is closed. Closing the doorbell circuit (by pressing the button) puts this mechanism in motion. Initially, the electromagnet field attracts the iron bar, which pulls the contact arm off the stationery metal contact. This reestablishes the connection between the electromagnet and the circuit, and the current can flow through it again. The magnetic field draws the contact arm up, and the process repeats itself as long as you hold down the buzzer button. In this way, the electromagnet keeps shutting itself on and off.

The buzzing noise you hear is the sound of the rapidly moving arm hitting the magnet and the stationary contact dozen times a second.

3. Other Components:

The other components include LED's, resistors, capacitors, npn transistors CL100, switches and jumper contacts. The jumpers are used to indicate faults. If a fault occurs, a particular jumper is closed. The output is also shown on the LED. The LED corresponding to the particular fault is shown. Resistors and capacitors are used to complete the hardware circuit.

2.5 Overview of the System Controller 89C51RD2

2.5.1 Description:

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pin out. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

This device is Single-Chip 8-bit microcontroller manufactured in advanced CMOS process and is a derivative of the 80C51 Microcontroller family. The instruction set is 100% compatible with 80C51 instruction set. The device also has four 8-bit I/O ports, three 16-bit timer/event counters. A multi-source, four-priority-level-nested interrupt structure, and enhanced UART and on chip control and timing circuits. The added features of the AT89C51RD2 makes it a powerful microcontroller for the applications that requires pulse width modulation, high speed I/O and up/down counting capabilities such as motor control.

Block Diagram

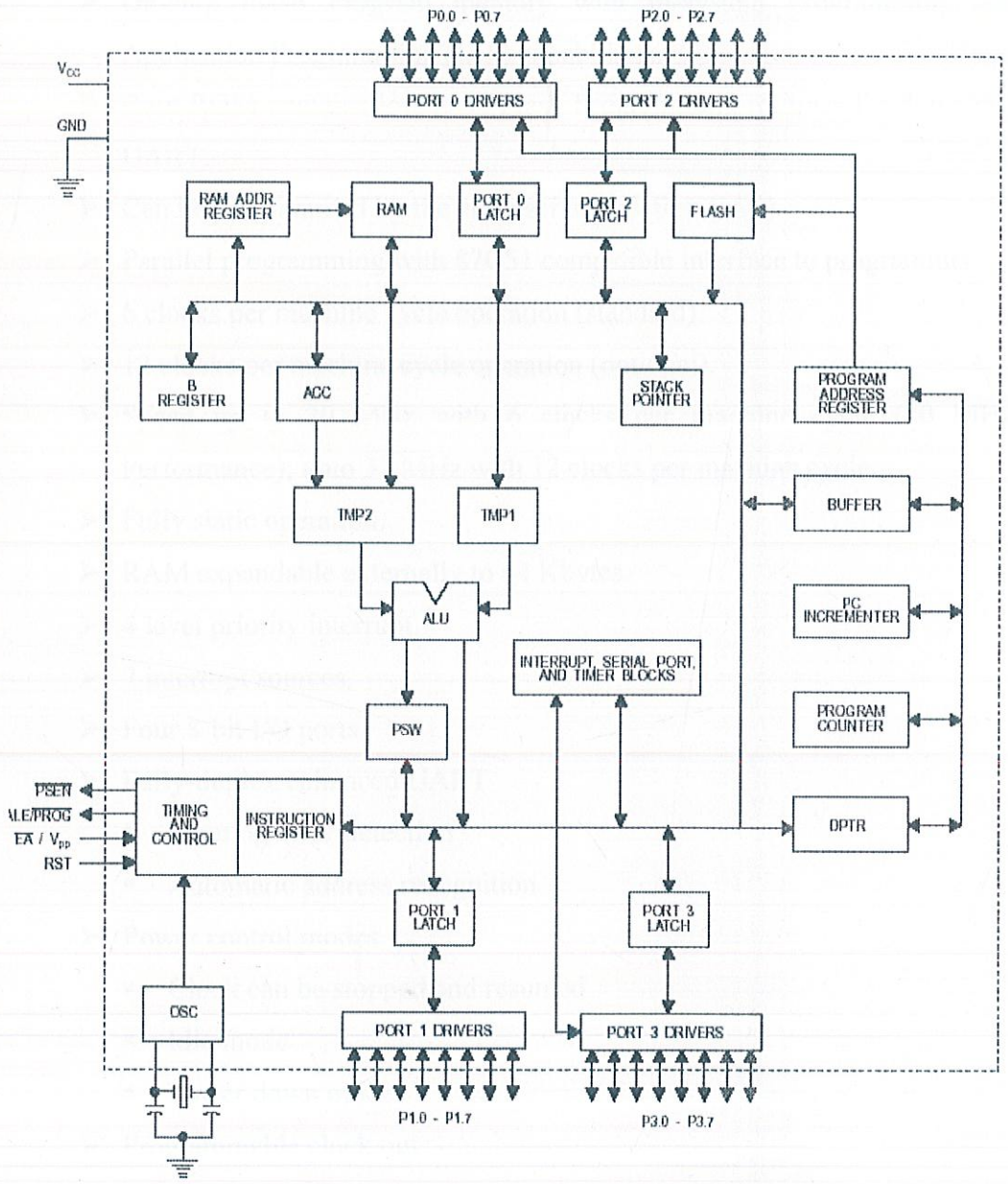


Figure 2.12- Block Diagram of 89C51RD2

2.5.2 ***Features:***

- 80C51 Central Processing Unit
- On-chip Flash Program memory with In-system Programming (ISP) and In-Application Programming (IAP) capabilities.
- Boot ROM contains low level Flash Programming routines for downloading via The UART.
- Can be programmed by the end-user application (IAP).
- Parallel programming with 87C51 compatible interface to programmer.
- 6 clocks per machine cycle operation (standard).
- 12 clocks per machine cycle operation (optional).
- Speed up to 20 MHz with 6 clocks per machine cycle (40 MHz equivalent Performance); upto 33 MHz with 12 clocks per machine cycle.
- Fully static operation.
- RAM expandable externally to 64 Kbytes.
- 4 level priority interrupt.
- 7 interrupt sources.
- Four 8-bit I/O ports.
- Fully-duplex enhanced UART
  - Framing error detection
  - Automatic address recognition
- Power control modes
  - Clock can be stopped and resumed
  - Idle mode
  - Power down mode
- Programmable clock out
- Second DPTR register
- Asynchronous port reset
- Low EMI (inhibit ALE)
- Programmable Counter Array (PCA)
  - PWM
  - Capture/Compare

Plastic Dual In-Line Package

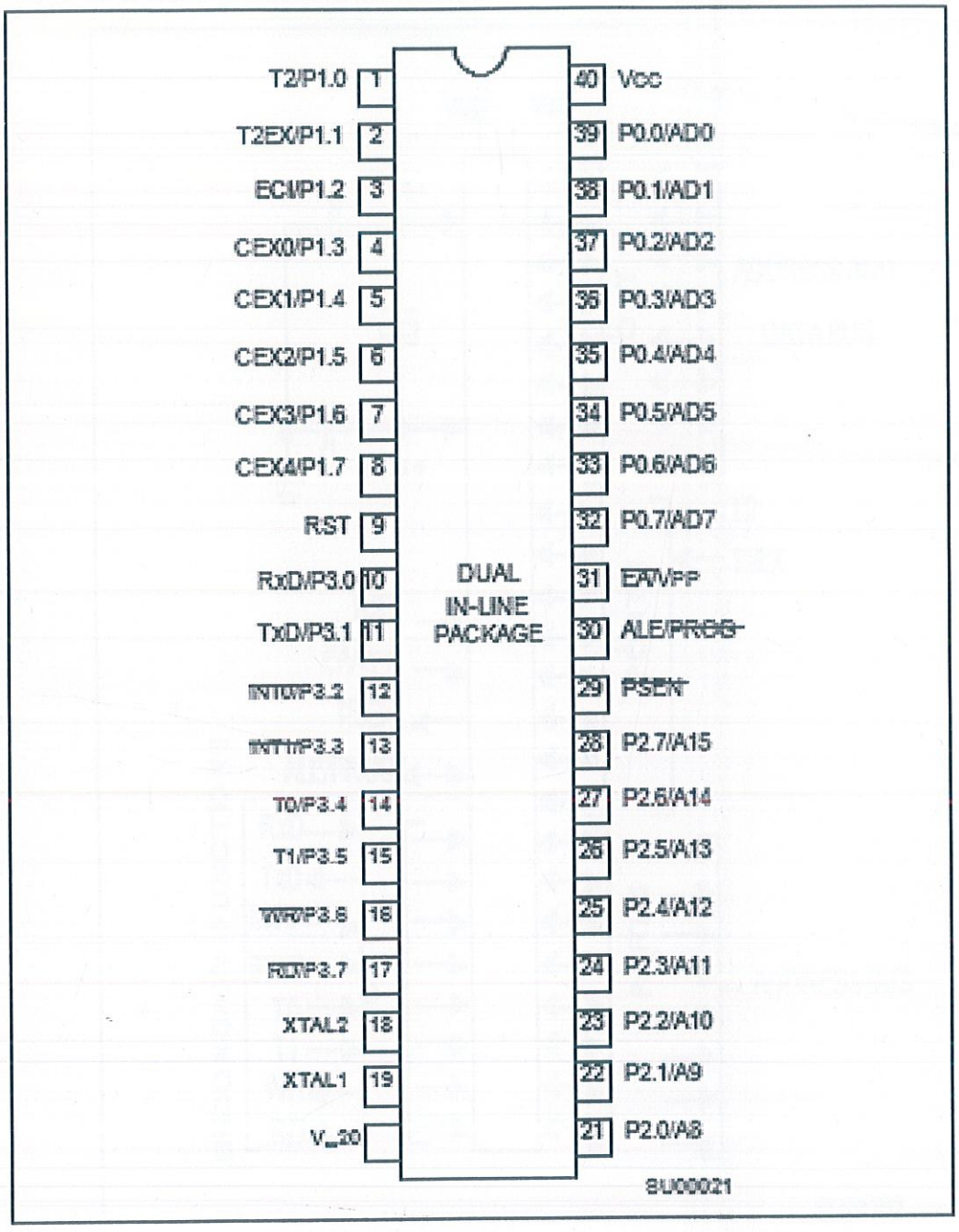


Figure 2.13- Pin Configuration of 89C51RD2

LOGIC SYMBOL

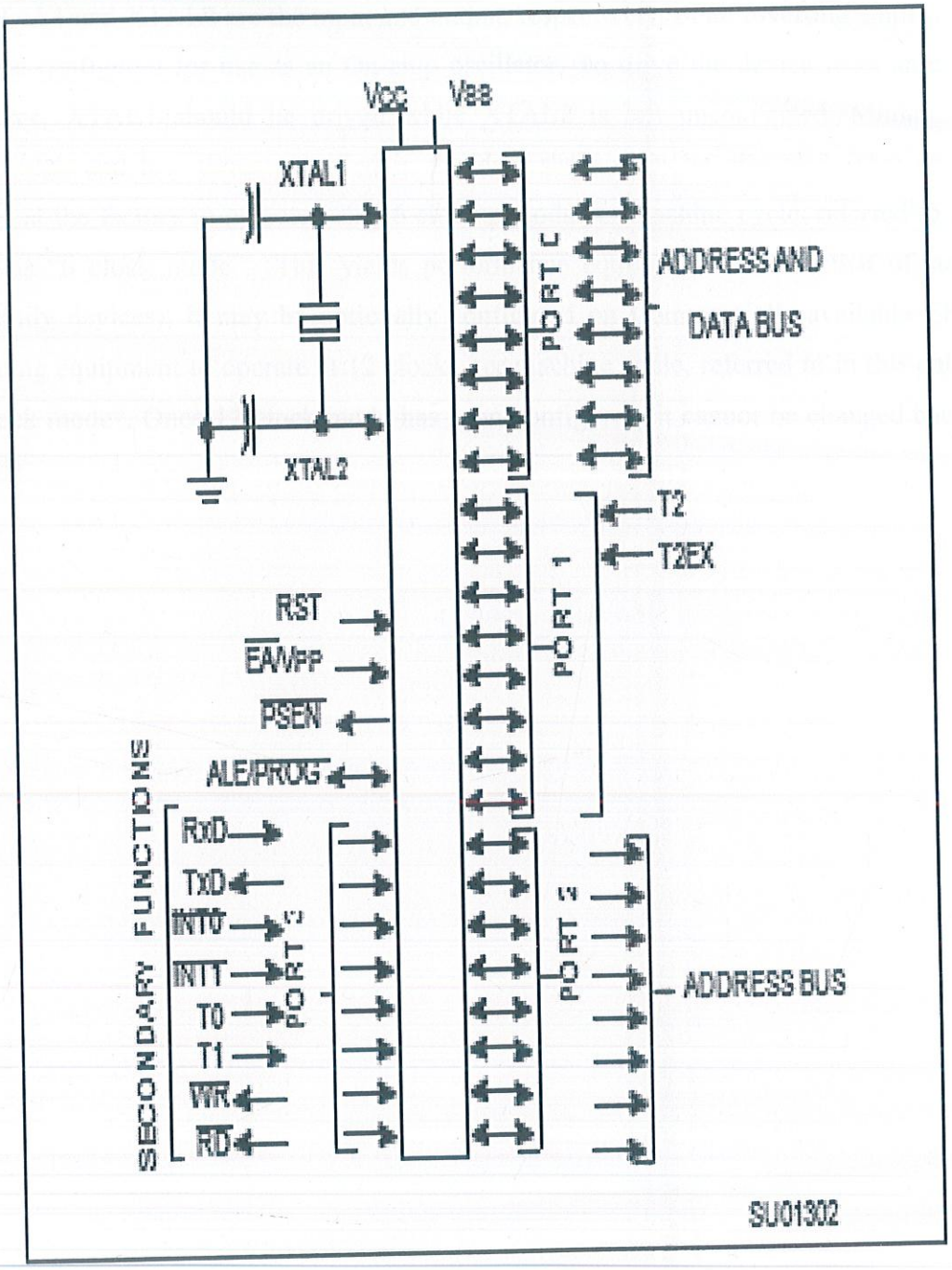


Figure 2.13- Pin Configuration of 89C51RD2

Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an Inverting amplifier. The pins can be configured for use as an On-chip oscillator. To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. Minimum and maximum high and low times specified in the data sheet must be observed. This device is configured at the factory to operate using 6 clock periods per machine cycle, referred to in this datasheet as "6 clock mode". (This yields performance equivalent to twice that of standard 80C51 family devices). It may be optionally configured on Commercially-available EPROM programming equipment to operate at 12 clocks per machine cycle, referred to in this datasheet as "12 clock mode". Once 12 clock mode has been configured, it cannot be changed back to 6 clock mode.

PIN DESCRIPTIONS

MNEMONIC	PIN NUMBER			TYPE	NAME AND FUNCTION
	PDIP	PLCC	PQFP		
V_{SS}	20	22	16	I	**Ground:** 0 V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups on all pins except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}).
					Alternate functions for 89C51RB2/RC2/RD2 Port 1 include:
	1	2	40	I/O	**T2 (P1.0):** Timer/Counter 2 external count input/Clockout (see Programmable Clock-Out)
	2	3	41	I	**T2EX (P1.1):** Timer/Counter 2 Reload/Capture/Direction Control
	3	4	42	I	**ECI (P1.2):** External Clock Input to the PCA
	4	5	43	I/O	**CEX0 (P1.3):** Capture/Compare External I/O for PCA module 0
	5	6	44	I/O	**CEX1 (P1.4):** Capture/Compare External I/O for PCA module 1
	6	7	1	I/O	**CEX2 (P1.5):** Capture/Compare External I/O for PCA module 2
	7	8	2	I/O	**CEX3 (P1.6):** Capture/Compare External I/O for PCA module 3
	8	9	3	I/O	**CEX4 (P1.7):** Capture/Compare External I/O for PCA module 4
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @RI), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 89C51RB2/RC2/RD2, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	27	O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted twice every machine cycle, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. ALE can be disabled by setting SFR auxiliary.0. With this bit set, ALE will be active only during a MOVX instruction.

Figure 2.14(a)- Pin Description

MNEMONIC	PIN NUMBER			TYPE	NAME AND FUNCTION
	PDIP	PLCC	PQFP		
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}/V_{PP}$	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations. If $\overline{EA}$ is held high, the device executes from internal program memory. The value on the $\overline{EA}$ pin is latched when RST is released and any subsequent changes have no effect. This pin also receives the programming supply voltage (V_{PP}) during Flash programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

Figure 2.14(b)- Pin Description

Reset

A reset is accomplished by holding the RST pin high for at least two machine cycles (12 oscillator periods in 6 clock mode, or 24 oscillator periods in 12 clock mode), while the oscillator is running. To ensure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on VCC and RST must come up at the same time for a proper start-up. Ports 1, 2, and 3 will asynchronously be driven to their reset condition when a voltage above VIH1 (min.) is applied to RESET. The value on the EA pin is latched when RST is reasserted and has no further effect.

Low Power Modes:

- **Stop Clock Mode**

The static design enables the clock speed to be reduced down to 0 MHz (stopped). When the oscillator is stopped, the RAM and Special Function Registers retain their values. This mode allows step-by-step utilization and permits reduced system power consumption by lowering the clock frequency down to any value. For lowest power consumption the Power Down mode is suggested.

- **Idle Mode**

In the idle mode (see Table 2), the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

- **Power-Down Mode**

To save even more power, a Power Down mode (see Table 2) can be invoked by software. In this mode, the oscillator is stopped and the instruction that invoked Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values down to 2.0 V and care must be taken to return VCC to the minimum specified operating voltages before the Power down Mode is terminated. Either a hardware reset or external interrupt can be used to exit from Power Down. Reset redefines all the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs and the on-chip RAM to retain their values.

To properly terminate Power Down, the reset or external interrupt should not be executed before VCC is restored to its normal operating level and must be held active long enough for the oscillator to restart and stabilize (normally less than 10 ms). With an external interrupt, INT0 and INT1 must be enabled and configured as level-sensitive. Holding the pin low restarts the oscillator but bringing the pin back high completes the exit. Once the interrupt is serviced, the next instruction to be executed after RET1 will be the one following the instruction that put the device into Power Down.

- **Power Off Flag**

The Power off Flag (POF) is set by on-chip circuitry when the VCC level on the 89C51RB2/RC2/RD2 rises from 0 to 5 V. The POF bit can be set or cleared by software allowing a user to determine if the reset is the result of a power-on or a warm start after power down. The VCC level must remain above 3 V for the POF to remain unaffected by the VCC level.

CHAPTER 3

DESIGN OF

THE SYSTEM

The design of the system contains both the hardware and software design. The hardware design is depicted with two levels of architecture designs.

3.1 Hardware Design

There are two levels of design. The first level of block diagram is given below:

3.1.1 First Level Block Diagram

The corresponding inputs and outputs used in the system are listed below the diagram

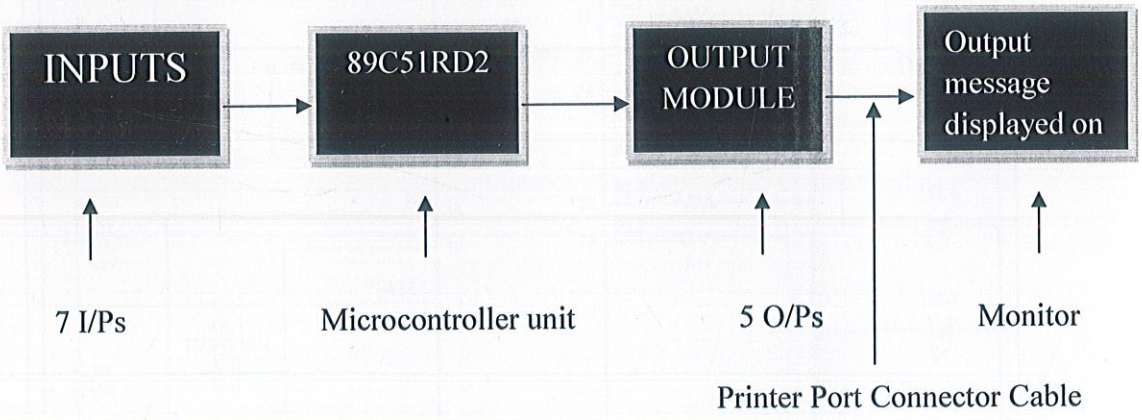


Figure 3.1- First level block diagram

3.1.2 Second Level Block Diagram

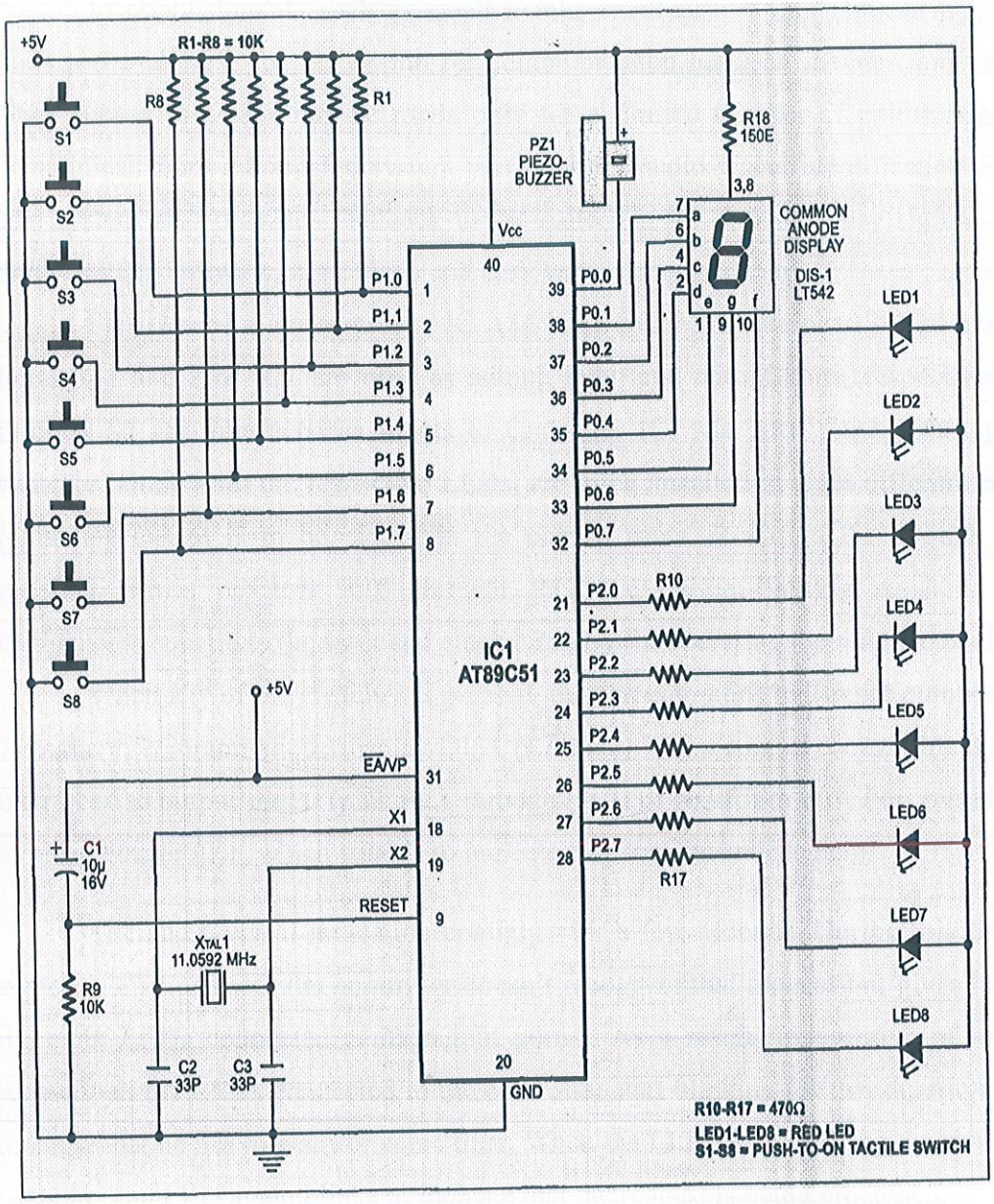


Figure 3.2-Second level circuit diagram

3.2 Circuit Description

In establishments such as small hotels, small offices and clinics, intercoms or calling bells prove to be a costlier option for communication between inmates and the assisting staff since such a provision can be made only for a limited number of points. Here's simple and economical room-monitoring system that provides audio-visual identification of the call point. The system also provides feedback to the caller (in the form of busy signal). Using minimal hardware and software, it's a clean and easy way to communicate.

Flash-based microcontroller IC AT89C51 (IC1) is at the heart of the monitoring circuit. Ports 0, 1 and 2 of IC1 are used as output, input and output ports, respectively. Switches S1 through S8 are interfaced as inputs to controller IC1 via port 1 (p1.0 through p1.7). These switches, along with the respective LEDs, are to be installed in eight different rooms, while the remaining circuit is to be placed in the control room. Resistors R1 through R8 are pull-up resistors, while resistors R10 through R17 are current-limiting resistors. Other passive components constitute the reset and clock circuitry for operating the microcontroller.

When any of the switches is pressed, the corresponding call-point number is displayed on 7-segment, common-anode display DIS1 (LTS542) in the control room. This display is directly interfaced to output port 0 (pins P0.1 through P0.7) of controller IC1. Port pin P0.0 is connected to piezo buzzer PZ1, which sounds to indicate that someone needs help.

The audio-visual indication continues for a few seconds. During this time, if any other switch is also pressed, the controller doesn't recognize that request but gives busy signal to all the eight LEDs connected at its output port 2. As a result, irrespective of any switch being pressed, all the LEDs connected to the controller start blinking for this duration, so a caller gets to know that he has to wait for some time. When the LED stops blinking, he can press the switch for help. The LED again blinks to indicate that the request is being processed.

Written in Assembly language, the program for the microcontroller (MS.ASM) is simple and easy to understand. The table provides codes for generating the 7-segment display and the RAM locations of the microcontroller where these are to be stored.

The complete procedure can be summarized as:

- Program the microcontroller with the MS.HEX file using AT89C51 programmer.
- After successfully programming the code, take the microcontroller out from the programmer and connect it to its IC base on the PCB of the circuit.
- After assembling and soldering all other components, connect a 5V DC external power supply.
- Now if you press any switch, the corresponding call-point number should display for a few seconds. At the same time, LEDs at all the call points should blink and the buzzer in the control room should sound for this duration.
- If the kit is working properly, install the main unit with 7-segment display and buzzer in the control room and all the eight switches with LEDs beside them at different call point.

3.3 Software Design

In this we have used two programs, a microcontroller based program and c ++ coding.

3.3.1 Microcontroller Coding:

Following is the coding for the microcontroller:

```
ORG   00H                ;locate reset routine at 00H
AJMP START               ;jump to START of main program
ORG   03H                ;locate interrupt 0
RETI                     ;returns from external interrupt 0
ORG   0BH                ;locate timer 0 interrupt
RETI                     ;returns from timer 0 interrupt
ORG   13H                ;locate interrupt 1
RETI                     ;returns from external interrupt 1
ORG   1BH                ;locate timer 1 interrupt
RETI                     ;returns from timer 1 interrupt
ORG   23H                ;locate serial port interrupt
RETI                     ;returns from serial port interrupt
ORG   25H                ;locate beginning of delay program

DELAYMS:
MOV   R7,#00H            ;delay of millisecond
```

```
LOOPA:
INC R7                  ;increment R7 by one
MOV  A, R7              ;store R7 value to Accumulator (A)
CJNE  A, #0FFH ,LOOPA
RET

DELAYHS:
MOV  R6,#00H            ;half second delay
MOV  R5,#008H           ;initialize R5

LOOPB:
INC R6                  ;to call milliseconds delay
ACALL  DELAYMS          ;call milliseconds delay routine above
MOV  A,R6               ;store R6 value to A
JNZ   LOOPB             ;go to LOOPB unless R6=OO
MOV  A,0A0H             ;store port 2(Address:A0 hex) value to A
CPL   A                 ;complement A and output A to port 2 to
MOV  0A0H,A             ;blinks all port LEDs
MOV  A,#00H             ;initialize A
DEC   R5                ;decrement R5
MOV  A,R5               ;move R5 value to A
```

```
JNZ    LOOPB          ;if A is not 0 then go to LOOPB
RET                   ;delay routine eight times

START:
MOV  61H,#00H         ;store 1st code at 61H
MOV  62H,#0B0H        ;store 2nd code at 62H
MOV  63H,#04H         ;store 3rd code at 63H
MOV  64H,#24H         ;store 4th code at 64H
MOV  65H,#32H         ;store 5th code at 65H
MOV  66H,#60H         ;store 6th code at 66H
MOV  67H,#48H         ;store 7th code at 67H
MOV  68H,#0F2H        ;store 8th code at 68H

LOOP:
MOV   A,90H           ;loop to continously read the inputs
CJNE  A,#0FFH,L1      ;detect any switch press
AJMP  LOOP            ;again jump to LOOP to read port 1

L1:
MOV R4,#00H           ;start from here if any switch is pressed
```

```
L2:
SETB C                  ;loop L2 unless carry=0 detected
RLC   A                 ;rotate A with carry flag
INC   R4                ;increment R4, each time loop L2 is run
JC    L2                ;jump to L2 if carry is set, otherwise come out of loop
MOV   A,R4              ;R4 value corresponds to switch number
ADD   A,#60H            ;add A
MOV   R0,A              ;move Address pointer value at A to register R0
MOV   A,@R0             ;address pointer R0 points to correct code stored
MOV   80H,A             ;code stored at A transfered to port 0
MOV   0A0H,#00H         ;initialize port 2 to start LEDs blinking
ACALL DELAYHS           ;call four seconds delay
MOV   80H,#0FFH
MOV   0A0H,#0FFH
AJMP  LOOP              ;delay returns here
END                     ;end program
```

3.3.2 Main C Programming:

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>

int gd = DETECT;
int gm;
struct time t;
char* str;

void
image(void)
{
        cleardevice();
        setcolor(15);
        outtextxy(200,11,"MICROCONTROLLER BASED ANNUNCIATOR SYSTEM ");
        circle(320,255,200);
        circle(320,255,210);
        pieslice(320,50,0,360,20);
```

```
        pieslice(145,150,0,360,20);
        pieslice(500,350,0,360,20);
        pieslice(320,457,0,360,20);
        pieslice(137,350,0,360,20);
}
void
showpos(int pos)
{
        if(pos>=80 && pos<=95)
{
                setcolor(0);
                outtextxy(290,80,"ÛÛÛÛÛÛÛÛÛ");
                gettime(&t);
                sprintf(str,"%02d:%02d:%02d",t.ti_hour,t.ti_min,t.ti_sec);
                setcolor(11);
                outtextxy(290,80,str);
         }
         else if(pos>=140 && pos<=155)
 {
                 setcolor(0);
                 outtextxy(175,150,"ÛÛÛÛÛÛÛÛÛ");
```

```
                gettime(&t);
                sprintf(str,"%02d:%02d:%02d",t.ti_hour,t.ti_min,t.ti_sec);
                setcolor(11);
                outtextxy(175,150,str);
        }
        else if(pos>=195 && pos<=210)
{
                setcolor(0);
                outtextxy(170,350,"ÛÛÛÛÛÛÛÛÛ");
                gettime(&t);
                sprintf(str,"%02d:%02d:%02d",t.ti_hour,t.ti_min,t.ti_sec);
                setcolor(11);
                outtextxy(170,350,str);
        }
        else if(pos>=260 && pos<=275)
 {
                setcolor(0);
                outtextxy(290,420,"ÛÛÛÛÛÛÛÛÛ");
                gettime(&t);
                sprintf(str,"%02d:%02d:%02d",t.ti_hour,t.ti_min,t.ti_sec);
                setcolor(11);
```

```
                outtextxy(290,420,str);
        }
        else if(pos>=320 && pos<=335)
{
                setcolor(0);
                outtextxy(400,350,"ÛÛÛÛÛÛÛÛÛ");
                gettime(&t);
                sprintf(str,"%02d:%02d:%02d",t.ti_hour,t.ti_min,t.ti_sec);
                setcolor(11);
                outtextxy(400,350,str);
        }
}

Int
main(void)
{
        unsigned char ch;
        int i;
        initgraph(&gd,&gm,"..\\bgi");
        image();
        while(true) {
```

```
        ch=inportb(0x379);
        if(ch==63) {
                showpos(80);
        }
        else if(ch==255)
{
                showpos(140);
        }
        else if(ch==95) {
                showpos(195);
        }
        else if(ch==111) {
                showpos(260);
        }
        else if(ch==119) {
                showpos(320);
        }
        if(kbhit())
                ch=getch();
        if(ch==27)
                break;
```

```
            ch=0;
            delay(25);
        }
        outportb(0x378,0);
        closegraph();
        return 0;
}
```

CHAPTER 4

PC INTERFACING

4.1 Interfacing with PC through Parallel Printer Port

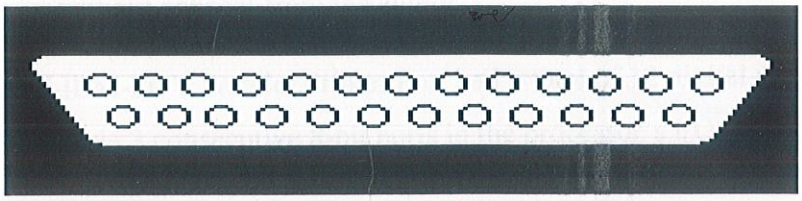


Figure 4.1 - 25 pin D-shaped female connector

PC parallel port can be very useful I/O channel for connecting your own circuits to PC. The PC's parallel port can be used to perform some very amusing hardware interfacing experiments. The port is very easy to use when you first understand some basic tricks. This document tries to show those tricks in easy to understand way.

WARNING: PC parallel port can be damaged quite easily if you make mistakes in the circuits you connect to it. If the parallel port is integrated to the motherboard (like in many new computers) repairing damaged parallel port may be expensive (in many cases it it is cheaper to replace the whole motherboard than repair that port).

Safest bet is to buy an inexpensive I/O card which has an extra parallel port and use it for your experiment. If you manage to damage the parallel port on that card, replacing it is easy and inexpensive.

NOTE: The I/O port level controlling details here has proven to work well with parallel ports on the PC motherboard and expansion cards connected to ISA bus. The programming examples might not work with PCI bus based I/O cards (they can use different hardware and/or I/O addresses, their drivers make they just look like parallel ports to "normal" applications). The programming examples do not work with USB to parallel port adapters (they use entirely different hardware , their drivers make them to look like normal parallel port to operating system "normal" applications).

DISCLAIMER: Every reasonable care has been taken in producing this information. However, the author can accept no responsibility for any effect that this information has on your equipment or any results of the use of this information. It is the responsibly of the end user to

determine fitness for use for any particular purpose. The circuits and software shown here are for non commercial use without consent from the author.

The original IBM-PC's Parallel Printer Port had a total of 12 digital outputs and 5 digital signal inputs accessed via 3 consecutive 8-bit ports in the processor's I/O space

- 8 output pins accessed via the DATA Port
- 5 input pins (one inverted) accessed via the STATUS Port
- 4 output pins (three inverted) accessed via the CONTROL Port
- The remaining 8 pins are grounded

PC parallel port can be very useful I/O channel for connecting your own circuits to PC. The port is very easy to use when you first understand some basic tricks. This document tries to show those tricks in easy to understand way.

PC parallel port is 25-pin D-shaped female connector in the back of the computer. It is normally used for connecting computer to printer, but many types of hardware for that port are available today.

Not all are needed always. Usually you can easily do with only 8 output pins (data lines) and signal ground. I have presented those pins in the table below. Those output pins adequate for many purposes.

4.1.1 Pin Functions:

Pin No.	Function	Pin No.	Function
1	Strobe	14	Auto Feed
2	Data 0	15	Error
3	Data 1	16	Init
4	Data 2	17	Select In
5	Data 3	18	Ground
6	Data 4	19	Ground
7	Data 5	20	Ground
8	Data 6	21	Ground
9	Data 7	22	Ground
10	Acknowledge	23	Ground
11	Busy	24	Ground
12	Paper Empty	25	Ground
13	Select		

Pins 18,19,20,21,22,23,24 and 25 are all grounded pins.

Those data pins are TTL level output pins. This means that they put out ideally 0V when they are low logic level (0) and +5V when they are in high logic level. In real world the output current capacity of the parallel port is limited to only few mill amperes.

4.2 Port Assignments:

The data output pins (pins 2-9) sink 24 mA, source 15 mA, and their high-level output is min. 2.4 V. The low state for both is max. 0.5 V. Pins 1, 14, 16, and 17 (the control outputs) have open collector drivers pulled to 5 V through 4.7 Kohm resistors (sink 20 mA, source 0.55 mA, high-level output 5.0 V minus pull up). Non-IBM parallel ports probably deviate from this standard.

Warning: Be careful with grounding. You can break parallel ports by connecting devices to them when PC is powered on. It is not a good idea to short the pins to ground or +5V, this can damage the port. It might be a good thing to use a parallel port not integrated on the motherboard for things like this. (You can usually get a second parallel port for your machine with a cheap standard `multi-I/O' card)

Each printer consists of three addresses; data, status and control port. These addresses are in sequential order. That if the data port is at address. 0X0378, the corresponding status port is at 0X0379 and the control port is at 0X037a.

The following is typical:

Printer	Data Port	Status	Control
LPT1	0X03bc	0X03bd	0X03be
LPT2	0X0378	0X0379	0X037a
LPT3	0X0278	0X0279	0X027a

4.3 Software Used:

The software used for interfacing the annunciator system to PC for displaying the error messages on the monitor is C++. The language supports the header file 'dos.h'. This header file has functions like 'inportb' to read an input byte from the port and 'outportb' to send a byte to the port.

CHAPTER 5

IMPLEMENTATION

TESTING &

RESULTS

5.1 Algorithm

Initialize Graphic Mode.

Draw Images (pie charts And Circles) using Image Function.

Set I/p to Port (0x379) // **I/p Device**

Read I/p byte by Byte

Display Text ("ÛÛÛÛÛÛÛÛ") and The Time in the appropriate position as per the Input.

Set Output Data Pins of port (0x378) to 1.

Close Graphic Mode.

5.2 Modules Used

Keeping in mind the functions to be performed by the system, the following modules were developed.

- **Self-test routine:** to check the reliable working of the system. This routine should be run atleast once on the start of the system. It checks that the connections of the circuit are intact and the system is ready to detect the fault and appropriately indicate the operator of it. The routine can be run even in between the working of the system whenever user wants to check the proper functioning of the system.
- **Scanning routine:** this routine continuously scans the input port of the microcontroller to check the faulty conditions.
- **Show routine:** in case of fault, the show routine is referred to. This routine gives a low output on the output port, where the LED does not glow.
- **Accept routine:** when the LED corresponding to a particular input does not glow it implies there is a fault in the input. This is noticed by the user and attended to.
- **Clear routine:** the clear routine should be run in the end when the operator wants to switch off the system. This routine is required to clear or reset the accepted fault

condition after necessary action has been taken for rectifying the faults that have already occurred.

5.3 Work Procedure

The working of the Annunciator system developed is explained through the following steps:

- Check the connection of the system whether they are intact.
- Switch on the evaluation board of 89C51RD2 by giving a +5V power supply from the adapter.
- Connect the printer cable to the parallel printer port of the computer and the other end to the output of the system on the breadboard.
- Switch on the computer and run C++ program for getting the status of the output pins of the microcontroller whether they are high or not.
- The microcontroller will scan the pins to determine whether faulty condition occurs or not.
- The program displays on the screen the time for which the input worked.
- When a fault occurs the system displays the fault at components connected to the output port.
- The output port is connected to LED, so when fault occurs the corresponding LED does not glow.
- To switch off the system firstly exit the C++ program required to display the error message.
- Switch off the computer connected to the breadboard through a parallel printer port cable.
- Finally, switch off the ac mains power supply given to the evaluation board and the breadboard through the adapter.

5.4 Testing

The testing of the code was done using the following test cases. These test cases along with the results obtained are as follows:

- **Case 1: Wires or Switches are tested**
  - ***Result:*** When the wires are tested against the copper strip or corresponding switches are pressed the LEDs glow with audible sound as long as they are being tested.
  - ***Conclusion:*** The desired result was obtained.
- **Case 2: One fault**
  - ***Result*** : The LED corresponding to that link does not glow and the time on the screen does not change
  - ***Conclusion:*** The desired result was obtained.
- **Case 3: Fault is corrected**
  - ***Result:*** The LED corresponding to the fault does not glow till the wire is tested to be correct. Once corrected the time on screen is visible.
  - ***Conclusion:*** The desired result was obtained.
- **Case 4: Wires are tested again**
  - ***Result:*** Once the fault is rectified all the wires are checked to confirm the proper working of corrected wire. The LEDs glow with audible buzzer sound.
  - ***Conclusion:*** The desired result was obtained.
- **Case 5: Multiple faults occur**
  - ***Result*** : The LEDs corresponding to the wires do not glow and present testing time cannot be seen on the screen
  - ***Conclusion:*** The desired result was obtained.

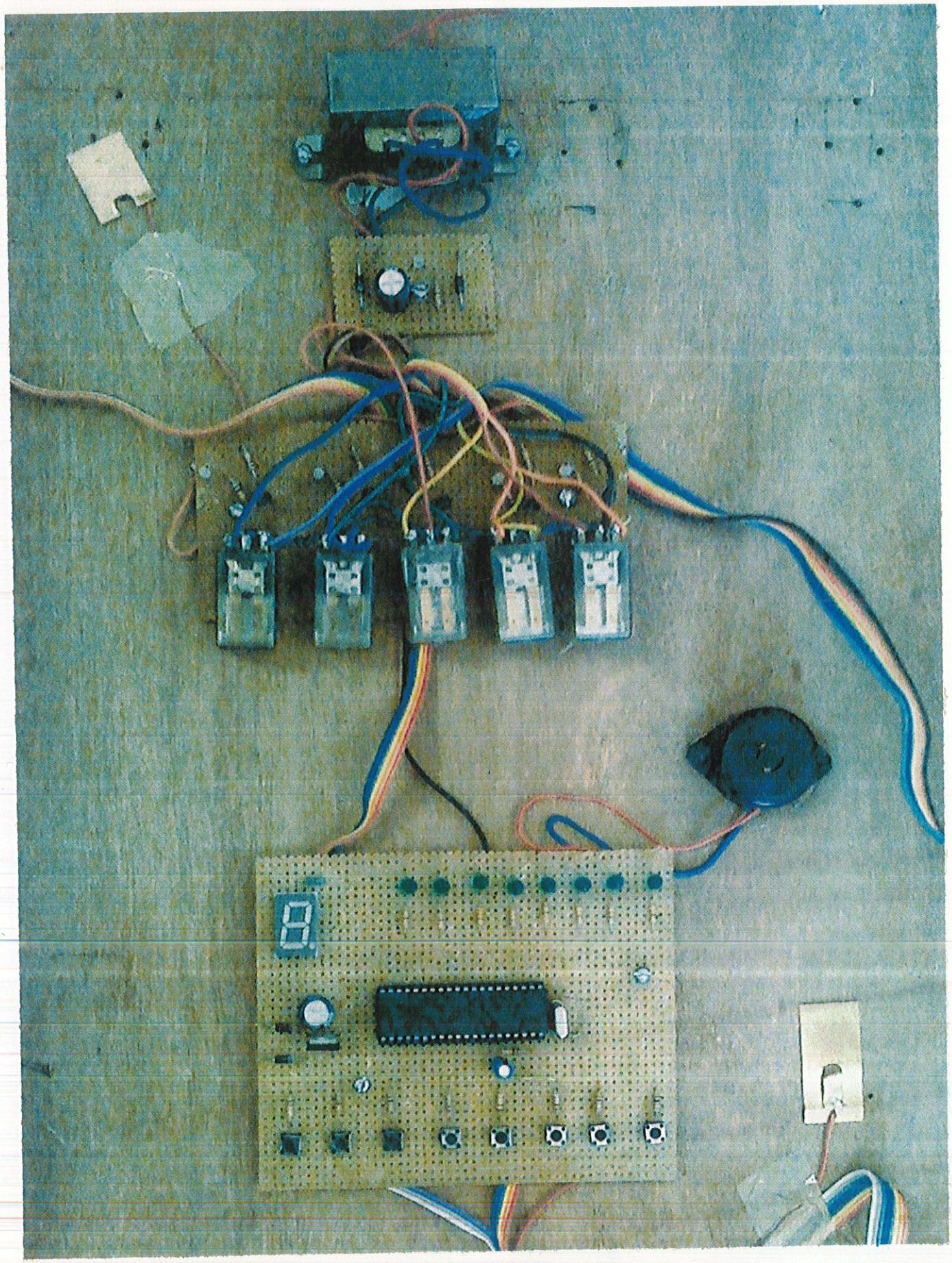
CHAPTER 6

CONCLUSION & FUTURE WORKS

6.1 Conclusion

System was designed carefully, taking all the precautions into consideration. The system was tested for the previously mentioned conditions and was found to work well. The output was displayed through LED, PC monitor and buzzer. The photograph of the hardware part of the entire annunciator system taken from a digital camera is shown on the next page.

6.2 System Designed



6.3 Future Works

- Multiple Annunciators can be effectively managed by one PC.
- They can be made to support bi-directional alarm reporting system. Hence can be used to transmit alarm information over a wide or local area channel.
- The system can be made such that inputs at the Annunciators are automatically sent to numeric or alphanumeric pagers using internal or external modems.
- The system can be improvised for error control for automatic rectification of faults.

APPENDIX- A

80C51 Instruction Sets

Arithmetic Operations:

Mnemonics	Description
ADD A,Rn	Add register to Accumulator
ADD A,direct	Add direct byte to Accumulator
ADD A,@Ri	Add indirect RAM to Accumulator
ADD A,#data	Add immediate data to Accumulator
ADDC A,Rn	Add register to Accumulator with carry
ADDC A,direct	Add direct byte to Accumulator with carry
ADDC A,@Ri	Add indirect RAM to Accumulator with carry
ADDC A,#data	Add immediate data to ACC with carry
SUBB A,Rn	Subtract Register from ACC with borrow
SUBB A,direct	Subtract direct byte from ACC with borrow
SUBB A,@Ri	Subtract indirect RAM from ACC with borrow
SUBB A,#data	Subtract immediate data from ACC with borrow
INC A	Increment Accumulator
INC Rn	Increment register
INC direct	Increment direct byte
INC @Ri	Increment indirect RAM
DEC A	Decrement Accumulator
DEC Rn	Decrement Register

DEC direct	Decrement direct byte
DEC @Ri	Decrement indirect RAM
INC DPTR	Increment Data Pointer
MUL AB	Multiply A and B
DIV AB	Divide A by B
DA A	Decimal Adjust Accumulator

Logical Operations:

Mnemonics	**Description**
ANL A,Rn	AND Register to Accumulator
ANL A,direct	AND direct byte to Accumulator
ANL A,@Ri	AND indirect RAM to Accumulator
ANL A,#data	AND immediate data to Accumulator
ANL direct,A	AND Accumulator to direct byte
ANL direct,#data	AND immediate data to direct byte
ORL A,Rn	OR register to Accumulator
ORL A,direct	OR direct byte to Accumulator
ORL A,@Ri	OR indirect RAM to Accumulator
ORL A,#data	OR immediate data to Accumulator
ORL direct,A	OR Accumulator to direct byte
ORL direct,#data	OR immediate data to direct byte

XRL A,Rn	Exclusive–OR register to Accumulator
XRL A,direct	Exclusive–OR direct byte to Accumulator
XRL A,@Ri	Exclusive–OR indirect RAM to Accumulator
XRL A,#data	Exclusive–OR immediate data to Accumulator
XRL direct,A	Exclusive–OR Accumulator to direct byte
XRL direct,#data	Exclusive–OR immediate data to direct byte
CLR A	Clear Accumulator
CPL A	Complement Accumulator
RL A	Rotate Accumulator left
RLC A	Rotate Accumulator left through the carry
RR A	Rotate Accumulator right
RRC A	Rotate Accumulator right through the carry
SWAP A	Swap nibbles within the Accumulator

Data Transfer:

Mnemonics	**Description**
MOV A,Rn	Move register to Accumulator
MOV A,direct	Move direct byte to Accumulator
MOV A,@Ri	Move indirect RAM to Accumulator
MOV A,#data	Move immediate data to Accumulator
MOV Rn,A	Move Accumulator to register

MOV Rn,direct	Move direct byte to register
MOV RN,#data	Move immediate data to register
MOV direct,A	Move Accumulator to direct byte
MOV direct,Rn	Move register to direct byte
MOV direct,direct	Move direct byte to direct
MOV direct,@Ri	Move indirect RAM to direct byte
MOV direct,#data	Move immediate data to direct byte
MOV @Ri,A	Move Accumulator to indirect RAM
MOV @Ri, direct	Move direct byte to indirect RAM
MOV @Ri, #data	Move immediate data to indirect RAM
MOV DPTR,#data16	Load Data Pointer with a 16–bit constant
MOVC A,@A+DPTR	Move Code byte relative to DPTR to ACC
MOVC A,@A+PC	Move Code byte relative to PC to ACC
MOVX A, @Ri	Move external RAM (8–bit addr) to ACC
MOVX A,@DPTR	Move external RAM (16–bit addr) to ACC
MOVX @Ri, A	Move ACC to external RAM (8–bit addr)
MOVX @DPTR, A	Move ACC to external RAM (16–bit addr)
PUSH direct	Push direct byte onto stack
POP direct	Pop direct byte from stack
XCH A,Rn	Exchange register with Accumulator
XCH A,direct	Exchange direct byte with Accumulator

XCH A,@Ri	Exchange indirect RAM with Accumulator
XCHD A,@Ri	Exchange low–order digit indirect RAM with ACC

Boolean Manipulation Transfers:

Mnemonics	Description
CLR C	Clear carry
CLR bit	Clear direct bit
SETB C	Set carry
SETB bit	Set direct bit
CPL C	Complement carry
CPL bit	Complement direct bit
ANL C,bit	AND direct bit to carry
ANL C,/bit	AND complement of direct bit to carry
ORL C,bit	OR direct bit to carry
ORL C,/bit	OR complement of direct bit to carry
MOV C,bit	Move direct bit to carry
MOV bit,C	Move carry to direct bit
JC rel	Jump if carry is set
JNC rel	Jump if carry not set
JB rel	Jump if direct bit is set
JNB rel	Jump if direct bit is not set

JBC bit,rel	Jump if direct bit is set and clear bit

Program Branching:

Mnemonics	**Description**
ACALL addr11	Absolute subroutine call
LCALL addr16	Long subroutine call
RET	Return from subroutine
RETI	Return from interrupt
AJMP addr11	Absolute jump
LJMP addr16	Long jump
SJMP rel	Short jump (relative addr)
JMP @A+DPTR	Jump indirect relative to the DPTR
JZ rel	Jump if Accumulator is zero
JNZ rel	Jump if Accumulator is not zero
CJNE A,direct,rel	Compare direct byte to ACC & jump if not equal
CJNE A,#data,rel	Compare immediate to ACC & jump if not equal
CJNE RN,#data,rel	Compare immediate to register & jump if not equal
CJNE @Ri,#data,rel	Compare immediate to indirect &jump if not equal
DJNZ Rn,rel	Decrement register and jump if not zero
DJNZ direct,rel	Decrement direct byte and jump if not zero
NOP	No operation

APPENDIX-B

References

- Predko Mike [1999]. "Programming and Customizing the 8051 Microcontroller" Tata McGraw-Hill Edition, pp.53-94.
- Houghton Bill [2001]. "AN461 In-circuit and In-application programming of the 89C51 Rx+/Rx2/66x microcontrollers" IC28 Data Handbook
- Houghton Bill [2001]. "P89C51RB2/P89C51RC2/P89C51RD2 80C51 8-bit Flash microcontroller family 16KB/32KB ISP/IAP Flash with 512B/512B/1KB RAM" IC28 Data Hand book
- Electronics For You Magazine

Websites

- http://www.allegromicro.com/sf/2801
- http://www.allegromicro.com/datafile/2801.pdf
- http://www.allegromicro.com/ic/leddriver.asp
- http://www.dataprobe.com/alarm/ann.html
- http://www.dataprobe.com/alarm/ann_demo/ann_demo.html
- http://www.dataprobe.com/alarm/index.html
- http://www.dataprobe.com/support/manuals/alm-2.pdf
- http://www.google.com
- http://www.kpsec.freeuk.com/components/resist.htm
- http://www.kpsec.freeuk.com/components/capac.htm
- http://www.kpsec.freeuk.com/components/diode.htm
- http://www.kpsec.freeuk.com/components/led.htm
- http://www.kpsec.freeuk.com/components/tran.htm
- http://www.kpsec.freeuk.com/components/connect.htm
- http://www.nohau.com/emul51pc.html
- http://www.technologystudents.com/elec1/relay1.htm