PROJECT REPORT

ON

**“ANTI RIGGING VOTING MACHINE FOR FREE GOVERNING USING FINGER PRINT SENSOR”**

BY

**A VINEETH 2014/B/03**

**N JAI SAI GOUTHAM 2014/B/25   
R MANIKANTA 2014/B/33**

Under Guidance of

**Mr.T SUNIL KUMAR SINGH**

NIELIT CENTER, Aurangabad.

In fulfillment of

**B.Tech in Electronics system engineering**

(2014-2018)



**NIELIT CENTRE, AURANGABAD**

(An Autonomous body under Ministry of Communication & Information Technology, MCIT. Govt. of India)

**NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY OF INDIA(NIELIT),**



**NIELIT CENTER, AURANGABAD.**

**CERTIFICATE**

This is to certify that

**A VINEETH 2014/B/03**

**N JAI SAI GOUTHAM 2014/B/25   
R MANIKANTA 2014/B/33**

Have satisfactorily completed the project titled,

**“ANTI RIGGING VOTING MACHINE FOR FREE GOVERNING USING FINGER PRINT SENSOR”**

In fulfillment of B.Tech

**2014-2018**

**Mr.T Sunil Kumar Singh Mr.Yashpalgogia**

PROJECT GUIDE PROJECT CO-ORDINATOR

NIELIT CENTRE, NIELIT CENTRE,

AURANGABAD AURANGABAD

**Dr.Sanjeev Kumar Gupta**

Executive Director of

NIELIT CENTRE, AURANGABAD



**NIELIT CENTRE, AURANGABAD.**

(An Autonomous body under Ministry of Communication & Information Technology, MCIT. Govt. of India)

Aurangabad 431004, Maharashtra, India.

**CERTIFICATE**

This is to certify that

**A VINEETH 2014/B/03**

**N JAI SAI GOUTHAM 2014/B/25   
R MANIKANTA 2014/B/33**

Have satisfactorily completed the project titled,

**“ANTI RIGGING VOTING MACHINE FOR FREE GOVERNING USING FINGER PRINT SENSOR”**

In fulfillment of

(B.Tech in Electronics system engineering)

**2014-2018**

**EXAMINER**

**ACKNOWLEDGEMENT**

It gives us intense pleasure as we have privilege to submit our project on **“ANTI RIGGING VOTING MACHINE FOR FREE GOVERNING USING FINGER PRINT SENSOR”** as a part of our final semester study of B.Tech.

We are related to express our deep sense of gratitude to our director **Dr.Sanjeev Kumar Gupta,**for giving us opportunity, necessary facilities and wonderful working environment.

We are greatly indebted to our project guide **Mr. T.Sunil Kumar,** without whose vision motivation, invaluable discussion, interaction, encouragement and supervision this project with never have taken off.

Also we pay our gratitude to **Mr.T.Sunil Kumar** for valuable help in interfacing of components.

We are truly indebt to **Mr. S. G. Wankhede** for helping us in all kinds of problems.

We express a deep sense of gratitude to our project coordinator **Mr.Yashpalgogia** for his help right from beginning to make this project successful.

We are also thankful to all teaching and non-teaching staff of NIELIT CENTRE, Aurangabad.

**A VINEETH 2014/B/03**

**N JAI SAI GOUTHAM 2014/B/25   
R MANIKANTA 2014/B/33**

**ABSTRACT**

Now a day’s voting process is exercised by using EVM(Electronic voting machine). In this paper we present and use implementation is to implement the development of anti rigging voting system using finger print .The purpose of the project and implementation is to provide a secured and reliable environment to the customers is to electing the candidates by using the intelligent electronic voting machine by providing a unique identity to every user using the FINGER PRINT identification technology. Here in this project and implementation we are going provide the at most security since it is taking the FINGER PRINTS as the authentication for EVM. Intelligent EVM is an Embedded based project and implementation. It involves microcontroller and interfaces. Intelligent EVM has been specially designed to collect, record, store, count and display cent percent accurately. It has got two units control unit and ballot unit. It has “DISPLAY” section that will display the number of votes to respective candidate at the end of the poll. Keywords- Finger Print Module, EEPROM ,Max232

**TABALE OF CONTENT**

**ABSTRACT**

**CHAPTER-1: INTRODUCTION**

Introduction of Project

Introduction of Embedded system

Introduction of Microcontroller

**CHAPTER-2: BLOCK DIAGRAM AND CIRCUIT DESCRIPTION**

Block Diagram

Block Diagram Description

Circuit Diagram

Circuit Description

**CHAPTER-3: COMPONENT DESCRIPTION**

Power supply

Microcontroller

LCD

Buzzer

Fingerprint sensor (R-305)

EEPROM(AT 24C08)

**CHAPTER-4: ADVANTAGES, DISADVANTAGES AND APPLICATIONS**

Advantages

Disadvantages

**CHAPTER-5: SOFTWARE DESCRIPTION AND CODING**

Introduction of keil

Code

**CONCLUSION**

**REFERENCE**

**CHAPTER-1**

**INTRODUCTION**

**1.1 INTRODUCTION OF PROJECT**

In present days, computer becomes a main part of human beings for storing information. This information is up to some extent is a secured one. For example the details of employees and student etc... The authority person may only change the details. For this protection we are going to provide a PASSWORD for the PCs. This is secure up to some extent only because there may be a chance of revealing the password or some times the authorized person may forgot the password. So we have to provide security for PCs with a unique and simple to remember identification. One of such identification is the FINGER PRINT. Finger print Scanner is a device for computer Security featuring superior performance, accuracy, durability based on unique NITGEN Fingerprint Biometric Technology. Fingerprint Scanner can be plugged into a computer separately with your mouse. Fingerprint Scanner is very safe and convenient device for security instead of password that is vulnerable to fraud and is hard to remember.

**1.2 INTRODUCTION TO EMBEDDED SYSTEMS**

Embedded systems are electronic devices that incorporate microprocessors with in their implementations. The main purposes of the microprocessors are to simplify the system design and provide flexibility. Having a microprocessor in the device helps in removing the bugs, making modifications, or adding new features are only matter of rewriting the software that controls the device. Or in other words embedded computer systems are electronic systems that include a microcomputer to perform a specific dedicated application. The computer is hidden inside these products. Embedded systems are ubiquitous. Every week millions of tiny computer chips come pouring out of factories finding their way into our everyday products.

Embedded systems are self-contained programs that are embedded within a piece of hardware. Whereas a regular computer has many different applications and software that can be applied to various tasks, embedded systems are usually set to a specific task that cannot be altered without physically manipulating the circuitry. Another way to think of an embedded system is as a computer system that is created with optimal efficiency, thereby allowing it to complete specific functions as quickly as possible.

**CHARACTERISTICS:**

Two major areas of differences are cost and power consumption. Since many embedded systems are produced in tens of thousands to millions of units range, reducing cost is a major concern. Embedded systems often use a (relatively) slow processor and small memory size to minimize costs.

**PLATFORM:**

There are many different CPU architectures used in embedded designs. This in contrast to the desktop computer market which is limited to just a few competing architectures mainly the Intel/AMD x86 and the Apple/Motorola/IBM Power PC’s which are used in the Apple Macintosh. One common configuration for embedded systems is the system on a chip, an application-specific integrated circuit, for which the CPU was purchased as intellectual property to add to the IC's design.

**TOOLS:**

Like a typical computer programmer, embedded system designers use compilers, assemblers and debuggers to develop an embedded system. Those software tools can come from several sources:

Software companies that specialize in the embedded market Ported from the GNU software development tools. Sometimes, development tools for a personal computer can be used if the embedded processor is a close relative to a common PC processor. Embedded system designers also use a few software tools rarely used by typical computer programmers..

**OPERATING SYSTEM:**

They often have no operating system, or a specialized embedded operating system (often a real-time operating system), or the programmer is assigned to port one of these to the new system.

**DEBUGGING:**

Debugging is usually performed with an in-circuit emulator, or some type of debugger that can interrupt the micro controller’s internal microcode. The microcode interrupt lets the debugger operate in hardware in which only the CPU works. The CPU-based debugger can be used to test and debug the electronics of the computer from the viewpoint of the CPU.

**DESIGN OF EMBEDDED SYSTEMS:**

The electronics usually uses either a microprocessor or a microcontroller. Some large or old systems use general-purpose mainframes computers or minicomputers.

**START-UP:**

All embedded systems have start-up code. Usually it disables interrupts, sets up the electronics, tests the computer (RAM, CPU and software), and then starts the application code. Many embedded systems recover from short-term power failures by restarting (without recent self-tests). Restart times under a tenth of a second are common.

**THE CONTROL LOOP:**

In this design, the software has a loop. The loop calls subroutines. Each subroutine manages a part of the hardware or software. Interrupts generally set flags, or update counters that are read by the rest of the software. A simple API disables and enables interrupts. Done right, it handles nested calls in nested subroutines, and restores the preceding interrupt state in the outermost enable. This is one of the simplest methods of creating an exocrine.

**USER INTERFACES:**

Interface designers at PARC, Apple Computer, Boeing and HP minimize the number of types of user actions. For example, use two buttons (the absolute minimum) to control a menu system (just to be clear, one button should be "next menu entry" the other button should be "select this menu entry"). A touch-screen or screen-edge buttons also minimize the types of user actions.

For example, Boeing's standard test interface is a button and some lights. When you press the button, all the lights turn on. When you release the button, the lights with failures stay on. The labels are in Basic English.

* 1. **INTRODUCTION TO MICROCONTROLLER**

Microcontrollers as the name suggests are small controllers. They are like single chip computers that are often embedded into other systems to function as processing/controlling unit. For example the remote control you are using probably has microcontrollers inside that do decoding and other controlling functions. They are also used in automobiles, washing machines, microwave ovens, toys ... etc, where automation is needed.

Micro-controllers are useful to the extent that they communicate with other devices, such as sensors, motors, switches, keypads, displays, memory and even other micro-controllers. Many interface methods have been developed over the years to solve the complex problem of balancing circuit design criteria such as features, cost, size, weight, power consumption, reliability, availability, manufacturability. Many microcontroller designs typically mix multiple interfacing methods. In a very simplistic form, a micro-controller system can be viewed as a system that reads from (monitors) inputs, performs processing and writes to (controls) outputs.

Microprocessor - A single chip that contains the CPU or most of the computer

Microcontroller - A single chip used to control other devices

Microcontroller differs from a microprocessor in many ways. First and the most important is its functionality. In order for a microprocessor to be used, other components such as memory, or components for receiving and sending data must be added to it. In short that means that microprocessor is the very heart of the computer. On the other hand, microcontroller is designed to be all of that in one. No other external components are needed for its application because all necessary peripherals are already built into it. Thus, we save the time and space needed to construct devices.

**MICROPROCESSOR VS MICROCONTROLLER:**

**Microprocessor:**

**•** CPU is stand-alone, RAM, ROM, I/O, timer are separate

• Designer can decide on the amount of ROM, RAM and I/O ports.

• expensive

• versatility general-purpose

**Microcontroller:**

**•** CPU, RAM, ROM, I/O and timer are all on a single chip

• fix amount of on-chip ROM, RAM, I/O ports

• for applications in which cost, power and space are critical

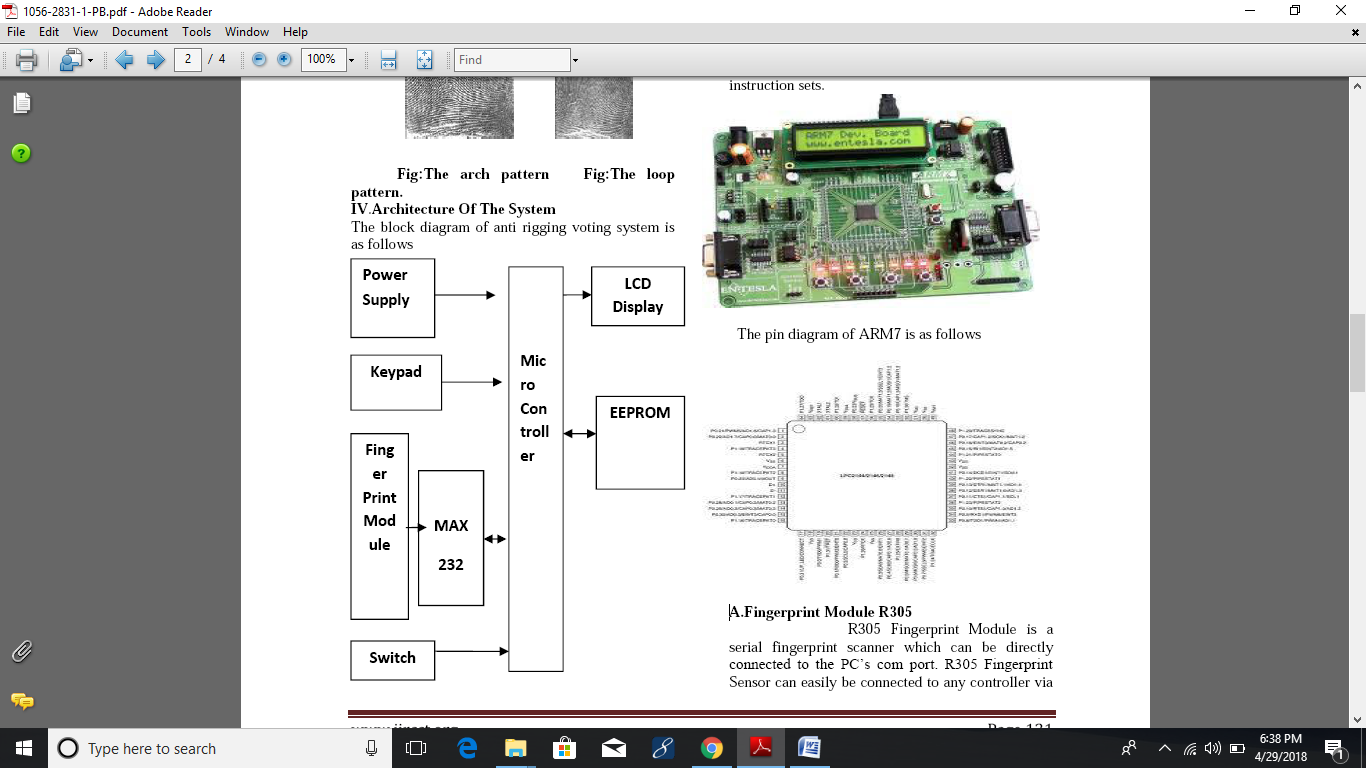
• single-purpose

**CHAPTER-2**

**BLOCK DIAGRAM AND CIRCUIT**

**DIAGRAM EXPLANATION**

**2.1 BLOCK DIAGRAM**

****

**2.2 BLOCK DIAGRAM DESCRIPTION**

In this project total we have 7 blocks which are

1. Power supply

2. Microcontroller

3. LCD

4. Buzzer

5. Fingerprint sensor (R-305)

6. EEPROM(AT 24C08)

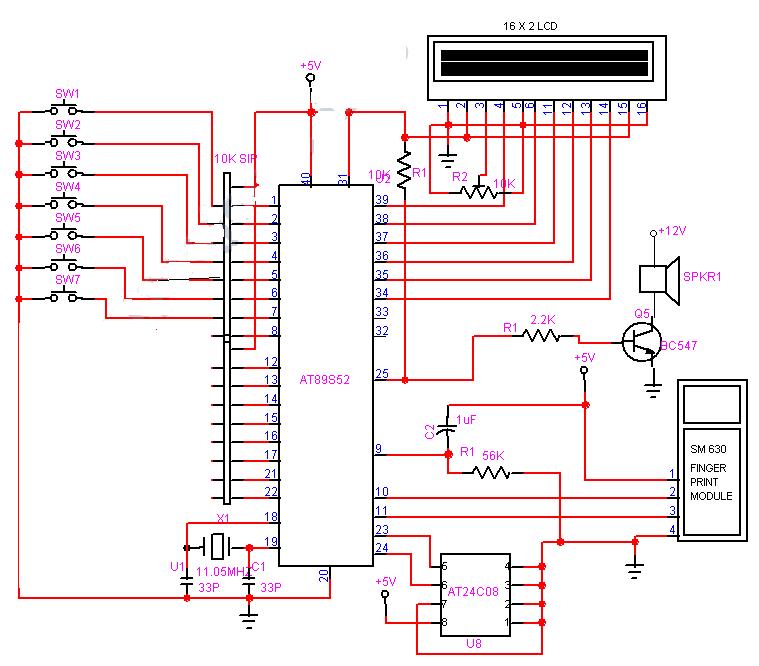
**Power supply**

Power supply block is providing sufficient power for all block from 230v AC. in this project we required 5v DC regulated power.

**Buzzer**

This block is used for converting electrical signal to sound signal. And alerting the neighbour people.

**2.3 CIRCUIT DIAGRAM**

****

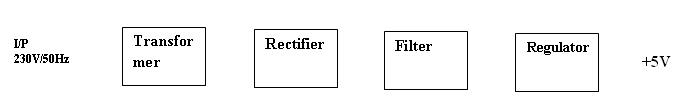
**2.4 CIRCUIT DESCRIPTION**

**CHAPTER-3**

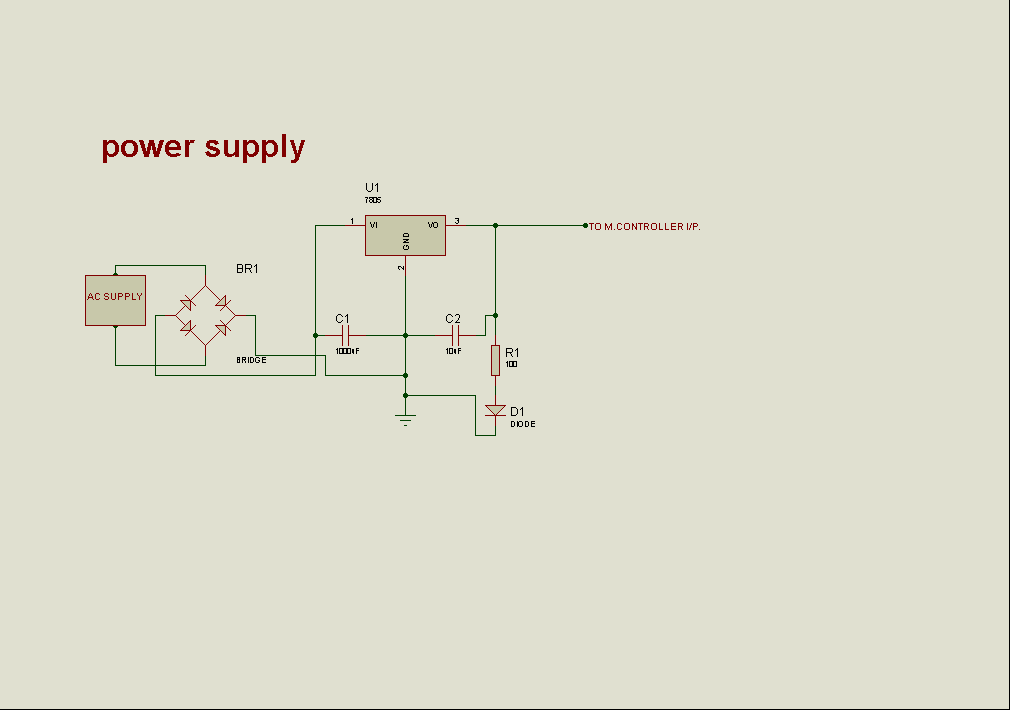
**COMPONENT DESCRIPTION**

**3.1 POWER SUPPLY:**

**Block diagram:**

****

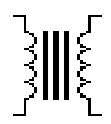
**Figure: Power Supply**

**Circuit diagram:**

**Description:**

**Transformer:**

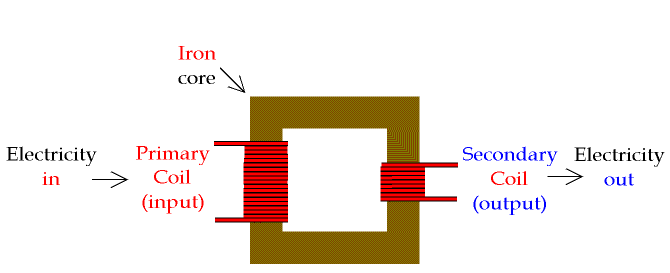
A **transformer** is a device that transfers [electrical energy](http://en.wikipedia.org/wiki/Electrical_energy) from one [circuit](http://en.wikipedia.org/wiki/Electrical_network) to another through [inductively coupled](http://en.wikipedia.org/wiki/Inductive_coupling) conductors—the transformer's coils. A varying [current](http://en.wikipedia.org/wiki/Electric_current) in the first or *primary* winding creates a varying [magnetic flux](http://en.wikipedia.org/wiki/Magnetic_flux) in the transformer's core, and thus a varying [magnetic field](http://en.wikipedia.org/wiki/Magnetic_field) through the *secondary* winding. This varying magnetic field [induces](http://en.wikipedia.org/wiki/Electromagnetic_induction) a varying [electromotive force (EMF)](http://en.wikipedia.org/wiki/Electromotive_force) or "[voltage](http://en.wikipedia.org/wiki/Volt)" in the secondary winding. This effect is called [mutual induction](http://en.wikipedia.org/wiki/Mutual_induction).



**Figure: Transformer Symbol**

**(or)**

Transformer is a device that converts the one form energy to another form of energy like a transducer.

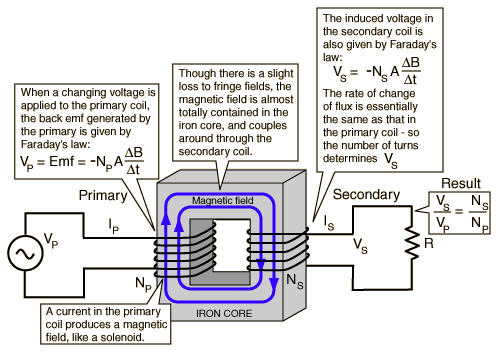


**Figure: Transformer**

**Basic Principle :**

A transformer makes use of [Faraday's law](http://hyperphysics.phy-astr.gsu.edu/hbase/electric/farlaw.html#c1) and the [ferromagnetic](http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html#c4) properties of an [iron core](http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/elemag.html#c4) to efficiently raise or lower AC voltages. It of course cannot increase [power](http://hyperphysics.phy-astr.gsu.edu/hbase/electric/powerac.html#c1) so that if the voltage is raised, the current is proportionally lowered and vice versa.

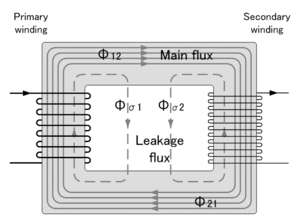
|  |
| --- |
| tran |



**Figure: Basic Principle**

**Transformer Working:**

A transformer consists of two coils (often called 'windings') linked by an iron core, as shown in figure below. There is no electrical connection between the coils, instead they are linked by a magnetic field created in the core.



**Figure: Basic Transformer**

Transformers are used to convert electricity from one voltage to another with minimal loss of power. They only work with AC (alternating current) because they require a changing magnetic field to be created in their core. Transformers can increase voltage (step-up) as well as reduce voltage (step-down).

Transformers have two great advantages over other methods of changing voltage:

1. They provide total electrical isolation between the input and output, so they can be safely used to reduce the high voltage of the mains supply.
2. Almost no power is wasted in a transformer. They have a high efficiency (power out / power in) of 95% or more.

**Classification of Transformer:**

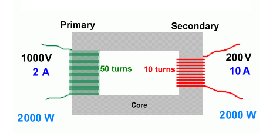
* Step-Up Transformer
* Step-Down Transformer

**Step-Down Transformer:**

Step down transformers are designed to reduce electrical voltage. Their primary voltage is greater than their secondary voltage. This kind of transformer "steps down" the voltage applied to it. For instance, a step down transformer is needed to use 110v product in a country with a 220v supply.

Step down transformers convert electrical voltage from one level or phase configuration usually down to a lower level. They can include features for electrical isolation, power distribution, and control and instrumentation applications. Step down transformers typically rely on the principle of magnetic induction between coils to convert voltage and/or current levels.

Step down transformers are made from two or more coils of insulated wire wound around a core made of iron. When voltage is applied to one coil (frequently called the primary or input) it magnetizes the iron core, which induces a voltage in the other coil, (frequently called the secondary or output). The turn’s ratio of the two sets of windings determines the amount of voltage transformation.

****

**Figure: Step-Down Transformer**

An example of this would be: 100 turns on the primary and 50 turns on the secondary, a ratio of 2 to 1.

**Step down transformers can be considered nothing more than a voltage ratio device.**

With step down transformers the voltage ratio between primary and secondary will mirror the "turn’s ratio" (except for single phase smaller than 1 kva which have compensated secondary). A practical application of this 2 to 1 turn’s ratio would be a 480 to 240 voltage step down. Note that if the input were 440 volts then the output would be 220 volts. The ratio between input and output voltage will stay constant. Transformers should not be operated at voltages higher than the nameplate rating, but may be operated at lower voltages than rated. Because of this it is possible to do some non-standard applications using standard transformers.

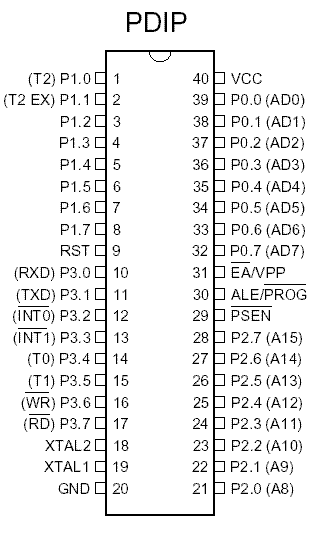
**3.2 DESCRIPTION OF MICROCONTROLLER 89S52**

**Features:**

* 8K Bytes of In-System Reprogrammable Flash Memory
* Endurance: 1,000 Write/Erase Cycles
* Fully Static Operation: 0 Hz to 24 MHz
* 256 x 8-bit Internal RAM
* 32 Programmable I/O Lines
* Three 16-bit Timer/Counters
* Eight Interrupt Sources
* Programmable Serial Channel
* Low-power Idle and Power-down Modes

The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8Kbytes of Flash programmable and erasable read only memory (PEROM). 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 AT89C52 is a powerful microcomputer, which provides a highly flexible and cost-effective solution to many embedded control applications.

**PIN DIAGRAM - AT89S52:**



**Pin Description:**

**VCC -** Supply voltage.

**GND -** Ground.

**Port 0:**

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs. Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pull-ups are required during program verification.

**Port 1:**

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively.

**Port Pin Alternate Functions:**

**P1.0 T2 (external count input to Timer/Counter 2), clock-out**

**P1.1 T2EX (Timer/Counter 2 capture/reload trigger and direction control**

**Port 2:**

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they 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 (I IL) because of the internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that uses 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that uses 8-bit addresses (MOVX @ RI); Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

**Port 3:**

Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they 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 (I IL) because of the pull-ups. Port 3 also serves the functions of various special features of the AT89C51. Port 3 also receives some control signals for Flash programming and verification.

**Port Pin Alternate Functions:**

P3.0 RXD (serial input port)

P3.1 TXD (serial output port)

P3.2 INT0 (external interrupt 0)

P3.3 INT1 (external interrupt 1)

P3.4 T0 (timer 0 external input)

P3.5 T1 (timer 1 external input)

P3.6 WR (external data memory write strobe)

P3.7 RD (external data memory read strobe).

**Rst:**

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

**Ale/Prog:**

Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during flash programming. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. However, that one ALE pulse is skipped during each access to external data memory. If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

**Psen:**

Program Store Enable is the read strobe to external program memory. When the AT89C52 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

**Ea/Vpp:**

External Access Enable (EA) must be strapped to GND in order to enable the device to fetch code from external pro-gram memory locations starting at 0000H up to FFFFH. However, if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12V programming enable voltage (VPP) during Flash programming when 12V programming is selected.

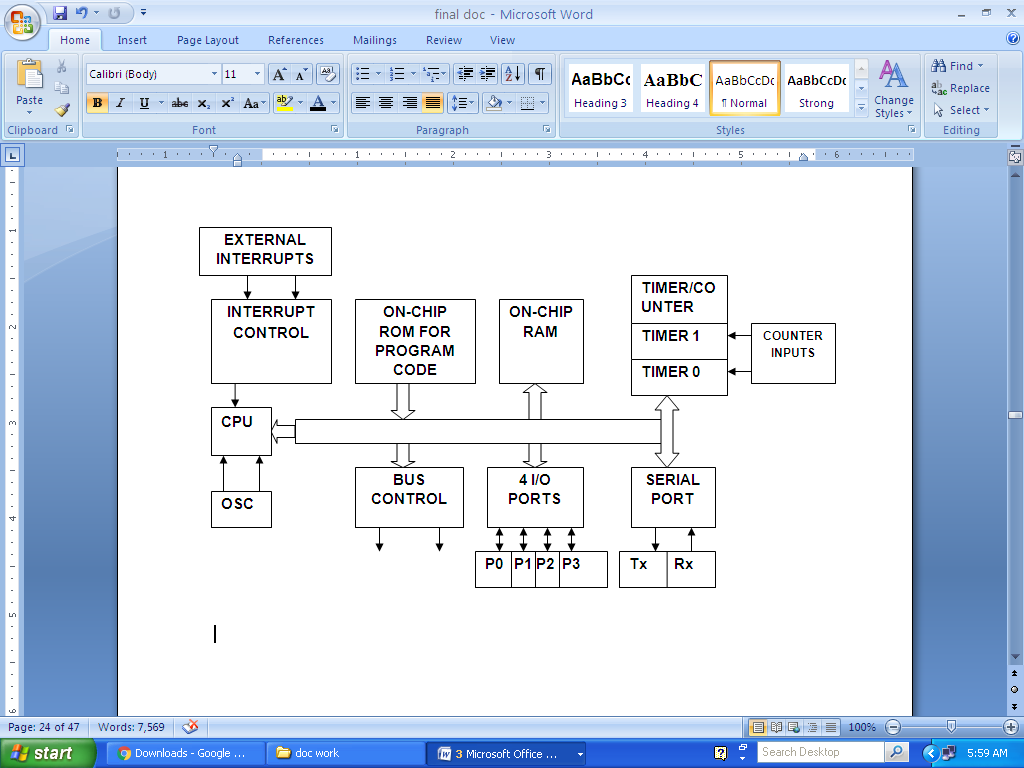
**Xtal1:**

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

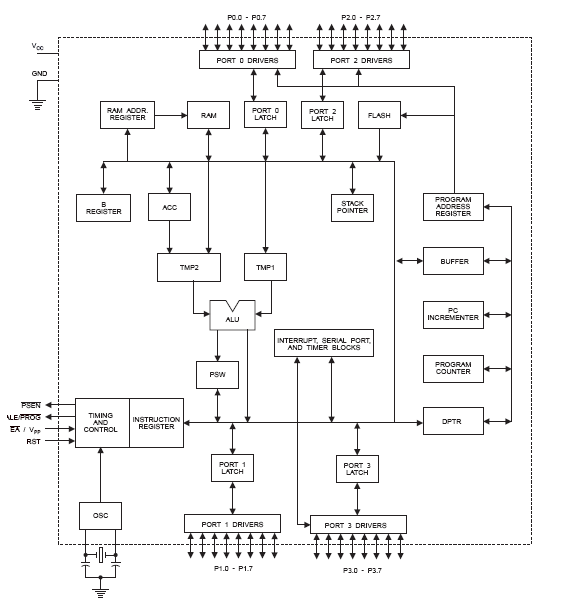
**Xtal2:**

It is an output from the inverting oscillator amplifier.

**Block Diagram of 89S52:**

****

**Architecture of 8052 Microcontroller:**



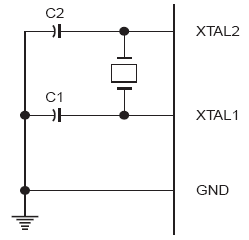
**Oscillator Characteristics:**

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier, which can be configured for use as an on-chip oscillator. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

**Idle Mode:**

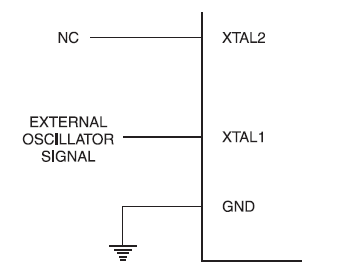
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset. It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

**Oscillator Connections:**



**Note:** C1, C2 = 30 pF ± 10 pF for Crystals

= 40 pF ± 10 pF for Ceramic Resonators



**External Clock drives Configuration.**

**3.3 LCD (Liquid Cristal Display):**

A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. Each pixel consists of a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other. Without the liquid crystals between them, light passing through one would be blocked by the other. The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

Line lengths of 8, 16, 20, 24, 32 and 40 characters are all standard, in one, two many microcontroller devices use 'smart LCD' displays to output visual information. LCD displays designed around LCD NT-C1611 module, are inexpensive, easy to use, and it is even possible to produce a readout using the 5X7 dots plus cursor of the display.

They have a standard ASCII set of characters and mathematical symbols. For an 8-bit data bus, the display requires a +5V supply plus 10 I/O lines (RS RW D7 D6 D5 D4 D3 D2 D1 D0). For a 4-bit data bus it only requires the supply lines plus 6 extra lines (RS RW D7 D6 D5 D4). When the LCD display is not enabled, data lines are tri-state and they do not interfere with the operation of the microcontroller.

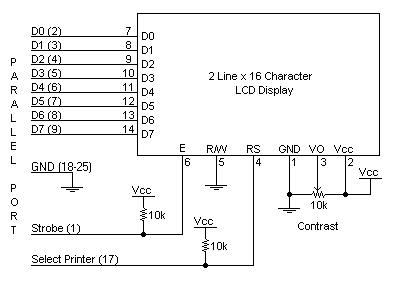
**Features:**

* Interface with either 4-bit or 8-bit microprocessor.
* Display data RAM
* Character generator ROM
* 160 different 5 x7 dot-matrix character patterns.
* Display data RAM and character generator RAM may be Accessed by the microprocessor.
* Numerous instructions
* Clear Display, Cursor Home, Display ON/OFF, Cursor ON/OFF,
* Blink Character, Cursor Shift, Display Shift.
* Built-in reset circuit is triggered at power ON.
* Built-in oscillator.

**Description Of 16x2:**

This is the first interfacing example for the Parallel Port. We will start with something simple. This example doesn't use the Bi-directional feature found on newer ports, thus it should work with most, if no all Parallel Ports. It however doesn't show the use of the Status Port as an input. So what are we interfacing? A 16 Character x 2 Line LCD Module to the Parallel Port. These LCD Modules are very common these days, and are quite simple to work with, as all the logic required running them is on board.

**Schematic Diagram:**



## Above is the quite simple schematic. The LCD panel's Enable and Register Select is connected to the Control Port. The Control Port is an open collector / open drain output. While most Parallel Ports have internal pull-up resistors, there are a few which don't. Therefore by incorporating the two 10K external pull up resistors, the circuit is more portable for a wider range of computers, some of which may have no internal pull up resistors.

## The 10k Potentiometer controls the contrast of the LCD panel. Nothing fancy here. As with all the examples, I've left the power supply out. You can use a bench power supply set to 5v or use a onboard +5 regulator. Remember a few de-coupling capacitors, especially if you have trouble with the circuit working properly.

## The 2 line x 16 character LCD modules are available from a wide range of manufacturers and should all be compatible with the HD44780. The one I used to test this circuit was a Power tip PC-1602F and an old Philips LTN211F-10 which was extracted from a Poker Machine! The diagram to the right, shows the pin numbers for these devices. When viewed from the front, the left pin is pin 14 and the right pin is pin 1

## 16 x 2 Alphanumeric LCD Module Features:

## Intelligent, with built-in Hitachi HD44780 compatible LCD controller and RAM providing simple interfacing

## 61 x 15.8 mm viewing area

## 5 x 7 dot matrix format for 2.96 x 5.56 mm characters, plus cursor line

## Can display 224 different symbols

## Low power consumption (1 mA typical)

## Powerful command set and user-produced characters

## TTL and CMOS compatible

## Connector for standard 0.1-pitch pin headers

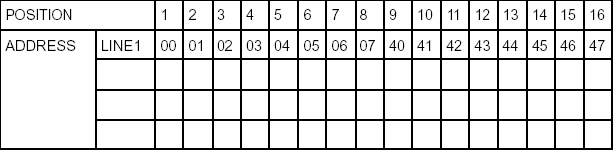
## 16 x 2 Alphanumeric LCD Module Specifications:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin** | **Symbol** | **Level** | **Function** |
| 1 | VSS | - | Power, GND |
| 2 | VDD | - | Power, 5V |
| 3 | Vo | - | Power, for LCD Drive |
| 4 | RS | H/L | Register Select Signal H: Data Input L: Instruction Input |
| 5 | R/W | H/L | H: Data Read (LCD->MPU) L: Data Write (MPU->LCD) |
| 6 | E | H,H->L | Enable |
| 7-14 | DB0-DB7 | H/L | Data Bus; Software selectable 4- or 8-bit mode |
| **15** | NC | - | **NOT CONNECTED** |
| 16 | NC | - | **NOT CONNECTED** |

**FEATURES:**

* 5 x 8 dots with cursor
* Built-in controller (KS 0066 or Equivalent)
* + 5V power supply (Also available for + 3V)
* 1/16 duty cycle
* B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A.K (LED)
* N.V. optional for + 3V power supply

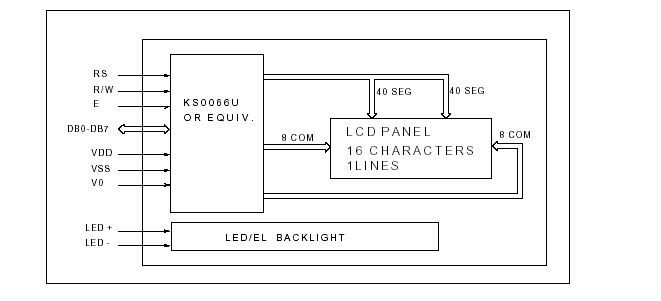
**Data can be placed at any location on the LCD. For 16×1 LCD, the address locations are:**



**Fig: 15: Address locations for a 1x16 line LCD**

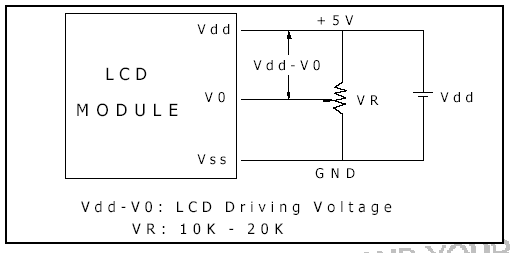
Even limited to character based modules, there is still a wide variety of shapes and sizes available. Line lengths of 8, 16,20,24,32 and 40 characters are all standard, in one, two and four line versions.

Several different LC technologies exist. “Supertwist” types, for example, offer Improved contrast and viewing angle over the older “twisted nematic” types. Some modules are available with back lighting, so that they can be viewed in dimly-lit conditions. The back lighting may be either “electro-luminescent”, requiring a high voltage inverter circuit, or simple LED illumination.



Electrical Block Diagrm

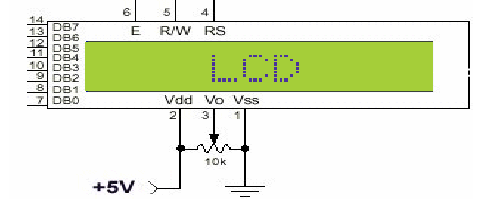
**Power supply for LCD driving:**



**Fig: 18: power supply for LCD**

**PIN DESCRIPTION:**

Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections).



**Fig 19: pin diagram of 1x16 lines LCD**

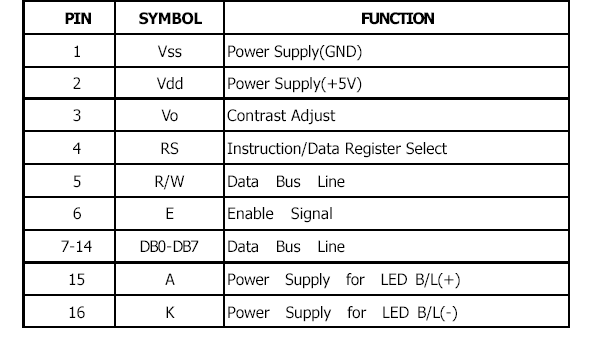


Fig 17: Pin specifications

**CONTROL LINES:**

**EN:**

Line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

**RS:**

Line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which sould be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

**RW:**

Line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands, so RW will almost always be low.

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

**Logic status on control lines:**

* E - 0 Access to LCD disabled- 1 Access to LCD enabled
* R/W - 0 Writing data to LCD- 1 Reading data from LCD
* RS - 0 Instructions- 1 Character

**Writing data to the LCD:**

1. Set R/W bit to low
2. Set RS bit to logic 0 or 1 (instruction or character)
3. Set data to data lines (if it is writing)
4. Set E line to high
5. Set E line to low

**Read data from data lines (if it is reading) on LCD:**

1) Set R/W bit to high

2) Set RS bit to logic 0 or 1 (instruction or character)

3) Set data to data lines (if it is writing)

4) Set E line to high

5) Set E line to low

**Entering Text:**

First, a little tip: it is manually a lot easier to enter characters and commands in hexadecimal rather than binary (although, of course, you will need to translate commands from binary couple of sub-miniature hexadecimal rotary switches is a simple matter, although a little bit into hex so that you know which bits you are setting). Replacing the d.i.l. switch pack with a of re-wiring is necessary.

**Switches:**

The switches must be the type where On = 0, so that when they are turned to the zero position, all four outputs are shorted to the common pin, and in position “F”, all four outputs are open circuit.

The first 16 codes in Table 3, 00000000 to 00001111, ($00 to $0F) refer to the CGRAM. This is the Character Generator RAM (random access memory), which can be used to hold user-defined graphics characters.

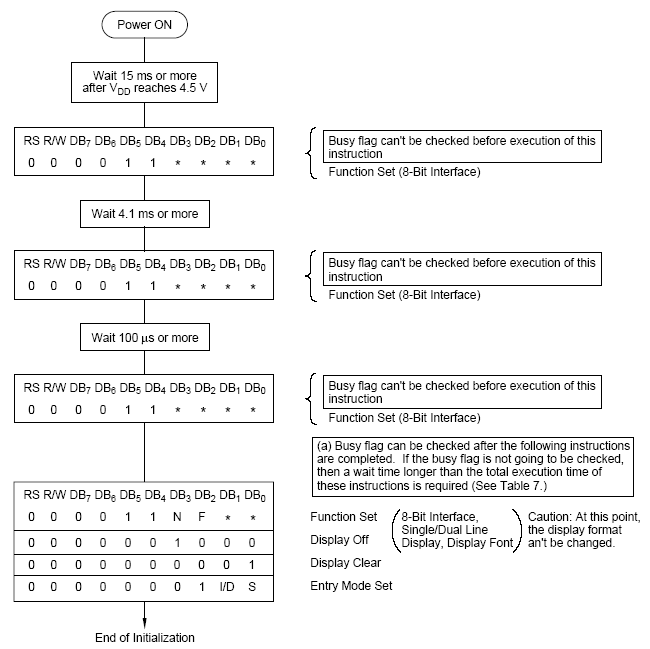
This is where these modules really start to show their potential, offering such capabilities as bar graphs, flashing symbols, even animated characters. Before the user-defined characters are set up, these codes will just bring up strange looking symbols.

Codes 00010000 to 00011111 ($10 to $1F) are not used and just display blank characters. ASCII codes “proper” start at 00100000 ($20) and end with 01111111 ($7F). Codes 10000000 to 10011111 ($80 to $9F) are not used, and 10100000 to 11011111 ($A0 to $DF) are the Japanese characters.

****

**Fig 20: character details in LCD**

**Initialization by Instructions:**

****

**Fig 21: flow chart of lcd**

If the power conditions for the normal operation of the internal reset circuit are not satisfied, then executing a series of instructions must initialize LCD unit. The procedure for this initialization process is as above show.

**FLOWCHART:**





**3.4 BUZZER**

A buzzer or beeper is an [audio](http://en.wikipedia.org/wiki/Sound) signaling device, which may be [mechanical](http://en.wikipedia.org/wiki/Machine), [electromechanical](http://en.wikipedia.org/wiki/Electromechanics), or [piezoelectric](http://en.wikipedia.org/wiki/Piezoelectric). Typical uses of buzzers and beepers include [alarms](http://en.wikipedia.org/wiki/Alarm), [timers](http://en.wikipedia.org/wiki/Timer) and confirmation of user input such as a mouse click or keystroke. A piezoelectric element may be driven by an [oscillating](http://en.wikipedia.org/wiki/Oscillation) electronic circuit or other [audio signal](http://en.wikipedia.org/wiki/Audio_signal) source, driven with a [piezoelectric audio amplifier](http://en.wikipedia.org/wiki/Piezoelectric_audio_amplifier). Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

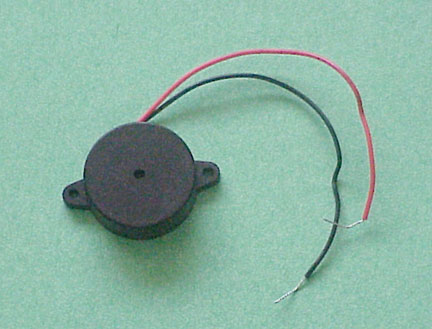


Fig.- piezo buzzer

The word "buzzer" comes from the rasping noise that buzzers made when they were electromechanical devices, operated from stepped-down AC line voltage at 50 or 60 cycles. Other sounds commonly used to indicate that a button has been pressed are a ring or a beep. Some systems, such as the one used on Jeopardy!, make no noise at all, instead using light.

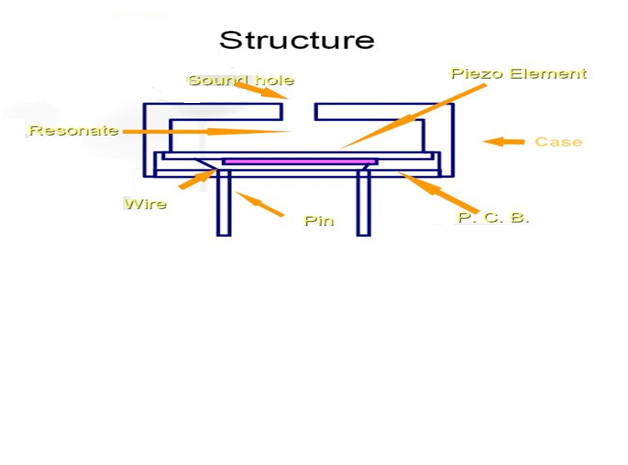


Fig.16- buzzer structure

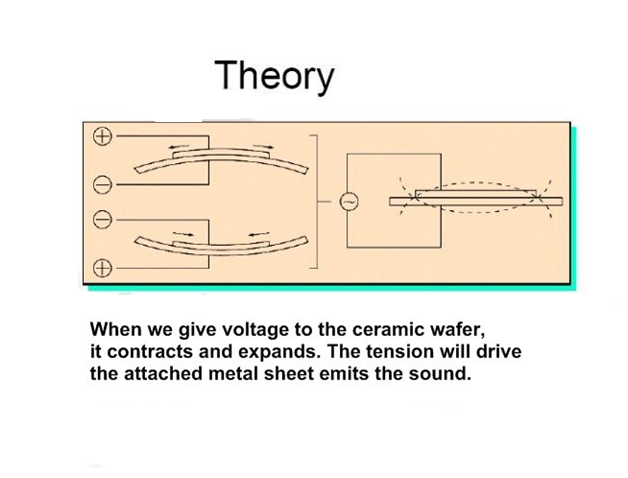


Fig.17- buzzer theory

**Specifications:**

**Rated Voltage:**

A piezo buzzer is driven by square waves (V p-p).**Operating Voltage:** For normal operating. But it is not guaranteed to make the minimum SPL under the rated voltage.

**Consumption Current:**

The current is stably consumed under the regular operation. However, it normally takes three times of current at the moment of starting to work.

**Capacitance:**

A piezo buzzer can make higher SPL with higher capacitance, but it consumes more electricity.

**Sound Output:**

The sound output is measured by decibel meter. Applying rated voltage and square waves, and the distance of 10 cm.

**Rated Frequency:**

A buzzer can make sound on any frequencies, but I suggest that the highest and the most stable SPL come from the rated frequency.

**Operating Temperature:**

Keep working well between -30℃ and +70℃.

**3.5 FINGERPRINT SENSOR (R-305)**

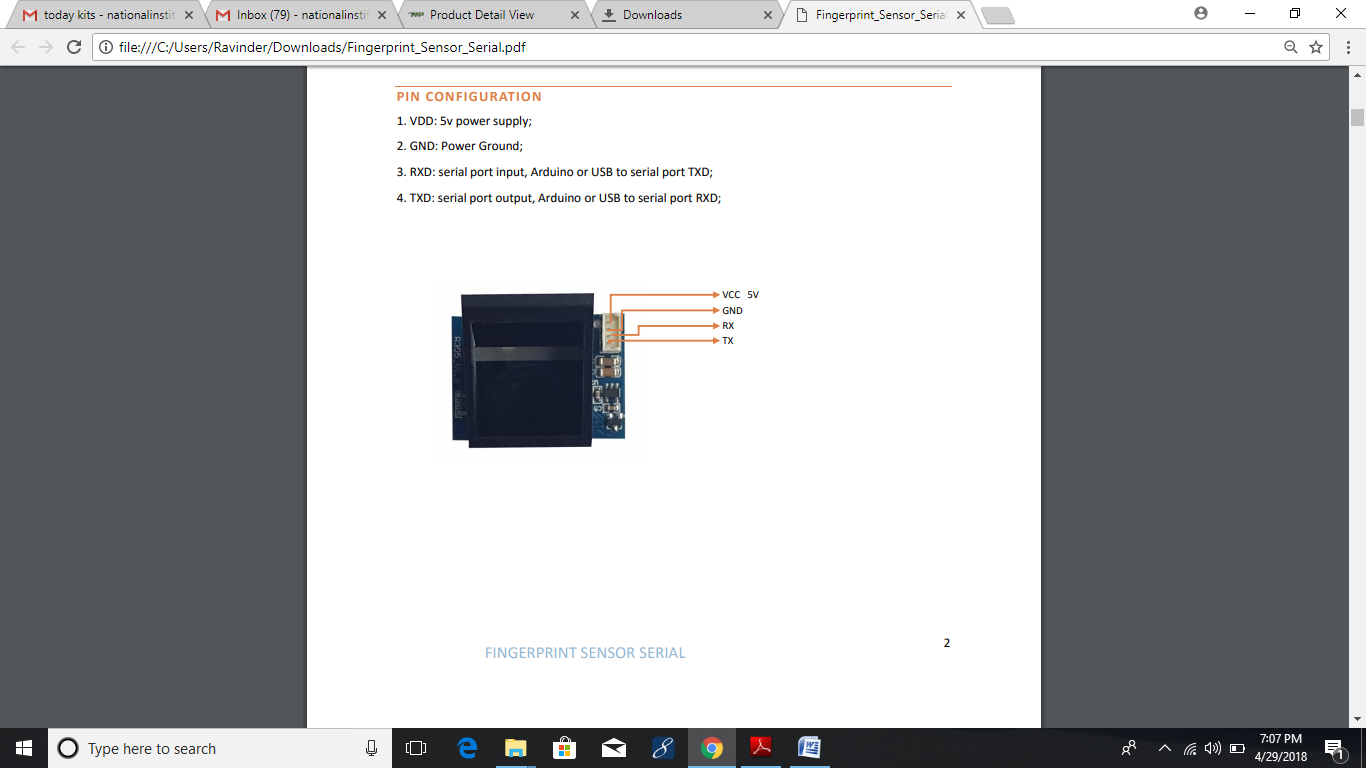
****

**INTRODUCTION**

This optical fingerprint sensor will make adding fingerprint detection and verification super simple. These modules are typically used in safes – there’s a high powered DSP chip that does the image rendering, calculation, feature-finding and searching. Connect to any microcontroller or system with TTL serial, and send packets of data to take photos, detect prints, hash and search. You can also enroll new fingers directly – up to 162 finger prints can be stored in the onboard FLASH memory.

**PIN CONFIGURATION**

* 1. VDD: 5v power supply;
  2. GND: Power Ground;
  3. RXD: serial port input, Arduino or USB to serial port TXD;
  4. TXD: serial port output, Arduino or USB to serial port RXD;

****

**FEATURES**

* Integrated image collecting and algorithm chip together, ALL-in-One
* Fingerprint reader can conduct secondary development, can be embedded into a variety of end
* products
* Low power consumption, low cost, small size, excellent performance
* Professional optical technology, precise module manufacturing techniques
* Good image processing capabilities, can successfully capture image up to resolution 500 dpi

**SPECIFICATIONS:**

* + - Fingerprint sensor type: Optical
    - Sensor Life: 100 million times
    - Static indicators: 15KVBacklight: bright green
    - Interface: USB1.1/UART(TTL logical level)
    - RS232 communication baud rate: 4800BPS~115200BPS changeable
    - Dimension: 55\*32\*21.5mm
    - Image Capture Surface 15—18(mm)
    - Verification Speed: 0.3 sec
    - Scanning Speed: 0.5 sec
    - Character file size: 256 bytes
    - Template size: 512 bytes
    - Storage capacity: 250
    - Security level: 5 (1,2,3,4,5(highest))
    - False Acceptance Rate (FAR) :0.0001%
    - False Rejection Rate (FRR): 0.1%
    - Resolution 500 DPI
    - Voltage :3.6-6.0 VDC
    - Working current: Typical 90 mA, Peak 150mA
    - Matching Method: 1: N
    - Operating Environment Temperature: -20 to 45° centigrade

**OPERATION PRINCIPLE**

Fingerprint processing includes two parts: fingerprint enrolment and fingerprint matching (the matching can be 1:1 or 1: N). When enrolling, user needs to enter the finger two times. The system will process the two time finger images, generate a template of the finger based on processing results and store the template. When matching, user enters the finger through optical sensor and system will generate a template of the finger and compare it with templates of the finger library. For 1:1 matching, system will compare the live finger with specific template designated in the Module; for 1: N matching, or searching, system will search the whole finger library for the matching finger. In both circumstances, system will return the matching result, success or failure.

**ADVANTAGES:**

* Eliminates paper card and al cost associated with paper card
* Reduces time needed to verify attendance data
* Reduces time needed to calculate and process payroll

**APPLICATION**

* Access control
* Attendance control
* Door Lock
* Safe
* POS system
* ATM

**DEVICE IMAGE**

****

**3.6 EEPROM (AT 24C08)**

**Features**

* Low-voltage and Standard-voltage Operation
  + - 2.7 (VCC = 2.7V to 5.5V)
    - 1.8 (VCC = 1.8V to 5.5V)
* Internally Organized 128 x 8 (1K), 256 x 8 (2K), 512 x 8 (4K), 1024 x 8 (8K) or 2048 x 8 (16K)
* Two-wire Serial Interface
* Schmitt Trigger, Filtered Inputs for Noise Suppression
* Bidirectional Data Transfer Protocol
* 100 kHz (1.8V) and 400 kHz (2.7V, 5V) Compatibility
* Write Protect Pin for Hardware Data Protection
* 8-byte Page (1K, 2K), 16-byte Page (4K, 8K, 16K) Write Modes
* Partial Page Writes Allowed
* Self-timed Write Cycle (5 ms max)
* High-reliability
* Endurance: 1 Million Write Cycles
* Data Retention: 100 Years
* Automotive Devices Available
* 8-lead JEDEC PDIP, 8-lead JEDEC SOIC, 8-lead Ultra Thin Mini-MAP (MLP 2x3), 5-lead
* SOT23, 8-lead TSSOP and 8-ball dBGA2 Packages
* Die Sales: Wafer Form, Waffle Pack and Bumped Wafers

**Description**

The AT24C01A/02/04/08A/16A provides 1024/2048/4096/8192/16384 bits of serial

Electrically erasable and programmable read-only memory (EEPROM) organized as

128/256/512/1024/2048 words of 8 bits each. The device is optimized for use in many

Industrial and commercial applications where low-power and low-voltage operation

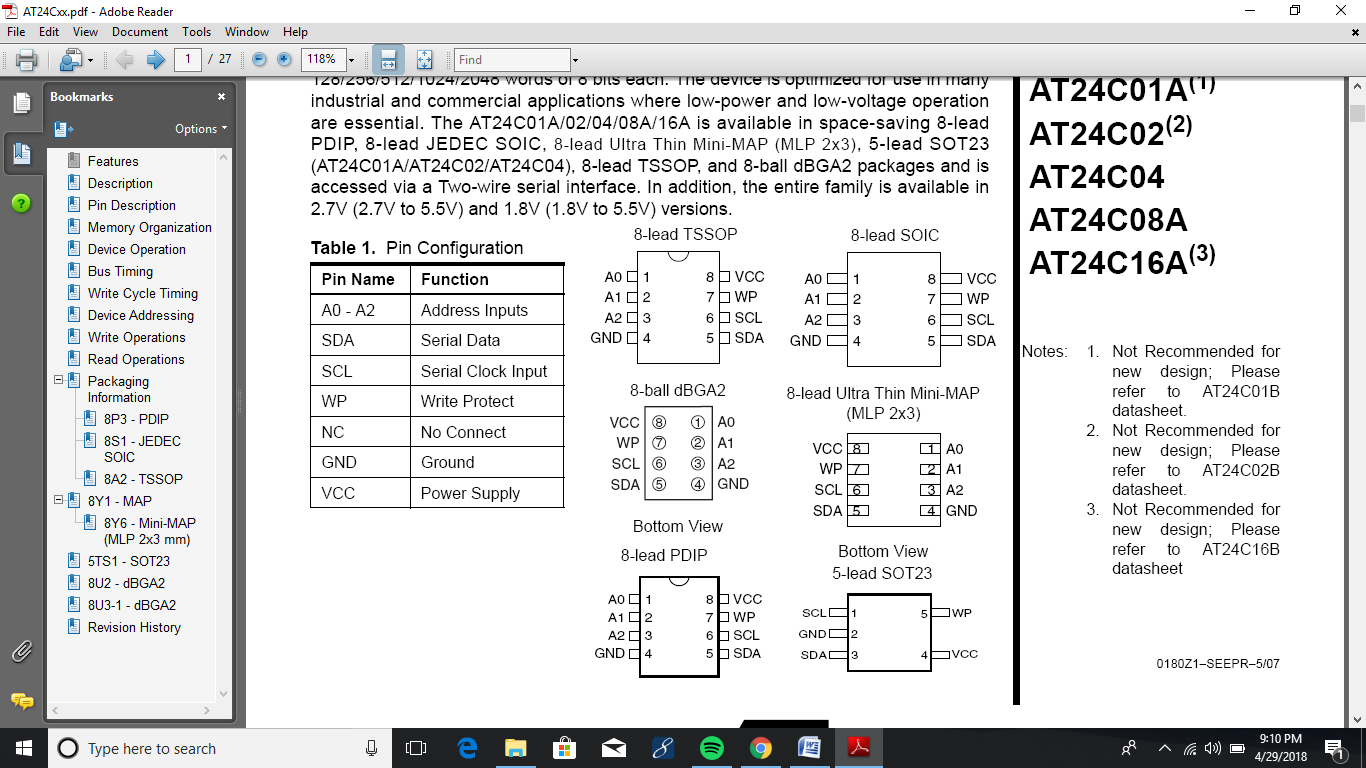
are essential. The AT24C01A/02/04/08A/16A is available in space-saving 8-lead

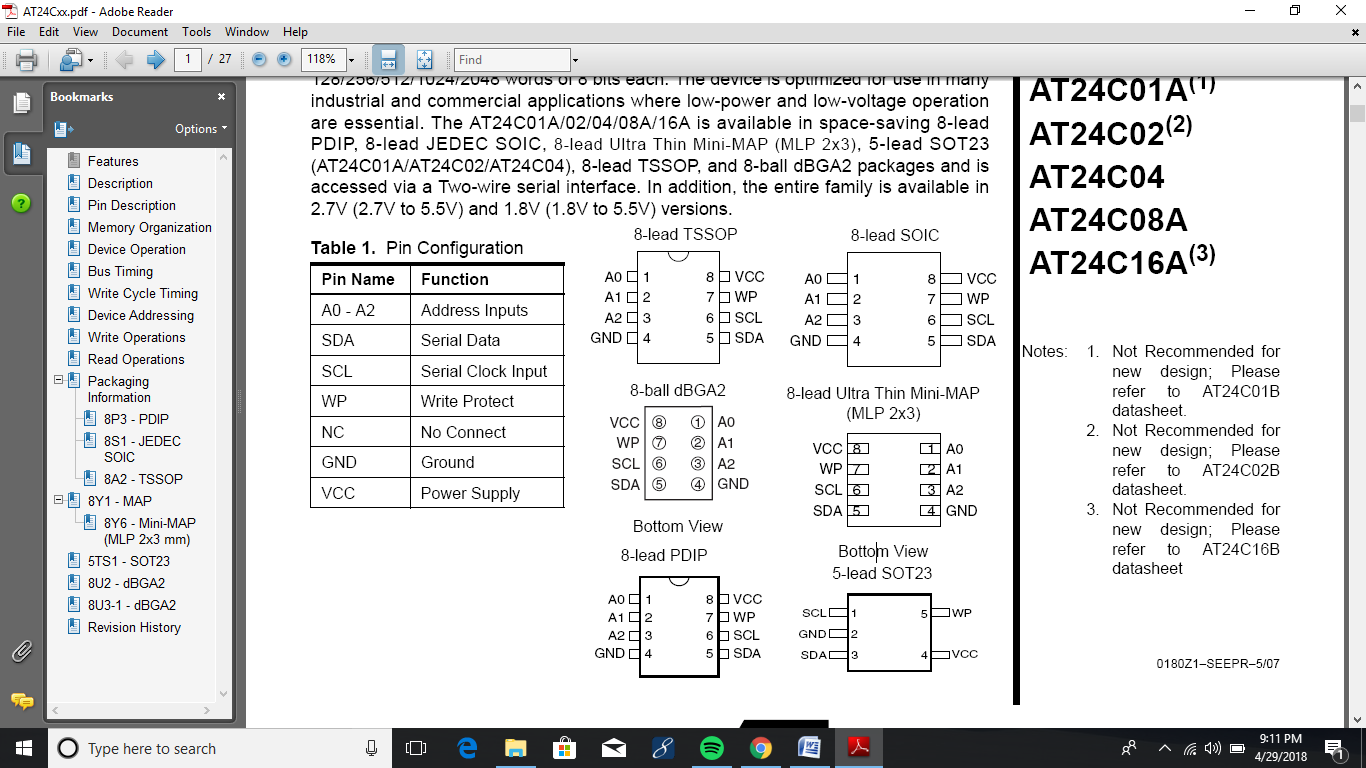
PDIP, 8-lead JEDEC SOIC, 8-lead Ultra Thin Mini-MAP (MLP 2x3), 5-lead SOT23

(AT24C01A/AT24C02/AT24C04), 8-lead TSSOP, and 8-ball dBGA2 packages and is

accessed via a Two-wire serial interface. In addition, the entire family is available in

2.7V (2.7V to 5.5V) and 1.8V (1.8V to 5.5V) versions.





**Pin Description**

**SERIAL CLOCK (SCL):** The SCL input is used to positive edge clock data into each EEPROM device and negative edge clock data out of each device.

**SERIAL DATA (SDA):** The SDA pin is bidirectional for serial data transfer. This pin is open-drain driven and may be wire-ORed with any number of other open-drain or opencollector devices.

**DEVICE/PAGE ADDRESSES (A2, A1, A0):** The A2, A1 and A0 pins are device

address inputs that are hard wired for the AT24C01A and the AT24C02. As many as

eight 1K/2K devices may be addressed on a single bus system (device addressing is

discussed in detail under the Device Addressing section).

The AT24C04uses the A2 and A1 inputs for hard wire addressing and a total of four 4K devices may be addressed on a single bus system. The A0 pin is a no connect and can be connected to ground.

The AT24C08A only uses the A2 input for hardwire addressing and a total of two 8K

devices may be addressed on a single bus system. The A0 and A1 pins are no connects and can be connected to ground.

The AT24C16A does not use the device address pins, which limits the number of

devices on a single bus to one. The A0, A1 and A2 pins are no connects and can be

connected to ground.

**WRITE PROTECT (WP):** The AT24C01A/02/04/08A/16A has a Write Protect pin that provides hardware data protection. The Write Protect pin allows normal Read/Write operations when connected to ground (GND). When the Write Protect pin is connected to VCC, the write protection feature is enabled and operates as shown in Table 2

**Memory Organization**

**AT24C01A, 1K SERIAL EEPROM:** Internally organized with 16 pages of 8 bytes each, the 1K requires a 7-bit data word address for random word addressing.

**AT24C02, 2K SERIAL EEPROM:** Internally organized with 32 pages of 8 bytes each, the 2K requires an 8-bit data word address for random word addressing.

**AT24C04, 4K SERIAL EEPROM:** Internally organized with 32 pages of 16 bytes each, the 4K requires a 9-bit data word address for random word addressing.

**AT24C08A, 8K SERIAL EEPROM**: Internally organized with 64 pages of 16 bytes

each, the 8K requires a 10-bit data word address for random word addressing.

**AT24C16A, 16K SERIAL EEPROM:** Internally organized with 128 pages of 16 bytes each, the 16K requires an 11-bit data word address for random word addressing.

**Device Operation**

**CLOCK and DATA TRANSITIONS:** The SDA pin is normally pulled high with an external device. Data on the SDA pin may change only during SCL low time periods (see Figure 4 on page 7). Data changes during SCL high periods will indicate a start or stop condition as defined below.

**START CONDITION:** A high-to-low transition of SDA with SCL high is a start condition which must precede any other command (see Figure 5 on page 8).

**STOP CONDITION:** A low-to-high transition of SDA with SCL high is a stop condition. After a read sequence, the stop command will place the EEPROM in a standby power mode (see Figure 5 on page 8).

**ACKNOWLEDGE:** All addresses and data words are serially transmitted to and from the EEPROM in 8-bit words. The EEPROM sends a zero to acknowledge that it has received each word. This happens during the ninth clock cycle.

**STANDBY MODE:** The AT24C01A/02/04/08A/16A features a low-power standby mode which is enabled: (a) upon power-up and (b) after the receipt of the STOP bit and the completion of any internal operations.

**MEMORY RESET:** After an interruption in protocol, power loss or system reset, any 2- wire part can be reset by following these steps:

1. Clock up to 9 cycles.
2. Look for SDA high in each cycle while SCL is high.
3. Create a start condition.

**Device Addressing**

The 1K, 2K, 4K, 8K and 16K EEPROM devices all require an 8-bit device address word following a start condition to enable the chip for a read or write operation (refer to Figure 7).

The device address word consists of a mandatory one, zero sequence for the first four

Most significant bits as shown. This is common to all the EEPROM devices.

The next 3 bits are the A2, A1 and A0 device address bits for the 1K/2K EEPROM.

These 3 bits must compare to their corresponding hard-wired input pins.

The 4K EEPROM only uses the A2 and A1 device address bits with the third bit being a memory page address bit. The two device address bits must compare to their corresponding hard-wired input pins. The A0 pin is no connect.

The 8K EEPROM only uses the A2 device address bit with the next 2 bits being for

memory page addressing. The A2 bit must compare to its corresponding hard-wired

input pin. The A1 and A0 pins are no connect.

The 16K does not use any device address bits but instead the 3 bits are used for memory page addressing. These page addressing bits on the 4K, 8K and 16K devices

should be considered the most significant bits of the data word address which follows.

The A0, A1 and A2 pins are no connect.

The eighth bit of the device address is the read/write operation select bit. A read operation is initiated if this bit is high and a write operation is initiated if this bit is low.

Upon a compare of the device address, the EEPROM will output a zero. If a compare is not made, the chip will return to a standby state.

**Write Operations**

**BYTE WRITE**: A write operation requires an 8-bit data word address following the

device address word and acknowledgment. Upon receipt of this address, the EEPROM will again respond with a zero and then clock in the first 8-bit data word. Following receipt of the 8-bit data word, the EEPROM will output a zero and the addressing device, such as a microcontroller, must terminate the write sequence with a stop condition. At this time the EEPROM enters an internally timed write cycle, tWR, to the nonvolatile memory. All inputs are disabled during this write cycle and the EEPROM will not respond until the write is complete

**PAGE WRITE:** The 1K/2K EEPROM is capable of an 8-byte page write, and the 4K, 8K and 16K devices are capable of 16-byte page writes.

A page write is initiated the same as a byte write, but the microcontroller does not send a stop condition after the first data word is clocked in. Instead, after the EEPROM acknowledges receipt of the first data word, the microcontroller can transmit up to seven (1K/2K) or fifteen (4K, 8K, 16K) more data words. The EEPROM will respond with a zero after each data word received. The microcontroller must terminate the page write sequence with a stop condition

The data word address lower three (1K/2K) or four (4K, 8K, 16K) bits are internally

incremented following the receipt of each data word. The higher data word address bits are not incremented, retaining the memory page row location. When the word address, internally generated, reaches the page boundary, the following byte is placed at the beginning of the same page. If more than eight (1K/2K) or sixteen (4K, 8K, 16K) data words are transmitted to the EEPROM, the data word address will “roll over” and previous data will be overwritten.;

**ACKNOWLEDGE POLLING:**

Once the internally timed write cycle has started and the EEPROM inputs are disabled, acknowledge polling can be initiated. This involves sending a start condition followed by the device address word. The read/write bit is representative of the operation desired. Only if the internal write cycle has completed will the EEPROM respond with a zero allowing the read or write sequence to continue.

**Read Operations**

Read operations are initiated the same way as write operations with the exception that the read/write select bit in the device address word is set to one. There are three read operations: current address read, random address read and sequential read.

**CURRENT ADDRESS READ:** The internal data word address counter maintains the last address accessed during the last read or write operation, incremented by one. This address stays valid between operations as long as the chip power is maintained. The address “roll over” during read is from the last byte of the last memory page to the first byte of the first page. The address “roll over” during write is from the last byte of the current page to the first byte of the same page.

Once the device address with the read/write select bit set to one is clocked in and acknowledged by the EEPROM, the current address data word is serially clocked out. The microcontroller does not respond with an input zero but does generate a following stop condition.

**RANDOM READ:** A random read requires a “dummy” byte write sequence to load in the data word address. Once the device address word and data word address are clocked in and acknowledged by the EEPROM, the microcontroller must generate another start condition. The microcontroller now initiates a current address read by sending a device address with the read/write select bit high. The EEPROM acknowledges the device address and serially clocks out the data word. The microcontroller does not respond with a zero but does generate a following stop condition.

**CHAPTER-4:**

**ADVANTAGES, DISADVANTAGES**

**4.1 ADVANTAGES:**

The main advantages of the systems are,

* Providing the preventive measures system for voting.
* It completely rules out the chance of invalid votes.
* Its use results in reduction of polling time.
* Results in fewer problems in electoral preparations, law and order candidates' expenditure.
* Provide easy and accurate counting without any mischief at the counting centre.
* It is capable of saving considerable printing stationery and transport of large volumes of electoral material.

**4.2 DISADVANTAGES:**

**Costly**

Biometric systems can be costly to implement, which might exclude many companies or organizations from implementing it.

**False Readings**

Biometric systems—especially lower cost systems—are prone to errors, including failing to identify an authorized person and incorrectly identifying unauthorized

**CHAPTER-5**

**SOFTWARE DESCRIPTION AND CODING**

#### 5.1 INTRODUCTION TO KIEL SOFTWARE

Many companies provide the 8051 assembler, some of them provide shareware version of their product on the Web, Kiel is one of them. We can download them from their Websites. However, the size of code for these shareware versions is limited and we have to consider which assembler is suitable for our application.

**KIEL U VISION2:**

This is an IDE (Integrated Development Environment) that helps you write, compile, and debug embedded programs. It encapsulates the following components:

. A project manager

. A make facility

. Tool configuration

. Editor

. A powerful debugger

To get start here are some several example programs

**BUILDING AN APPLICATION IN UVISION2:**

To build (compile, assemble, and link) an application in uVision2, you must:

. Select Project–Open Project

(For example, \C166\EXAMPLES\HELLO\HELLO.UV2)

. Select Project - Rebuild all target files or Build target. UVision2 compiles, assembles, and links the files in your project.

**CREATING YOUR OWN APPLICATION IN UVISION2:**

To create a new project in uVision2, you must:

. Select Project - New Project.

. Select a directory and enter the name of the project file.

. Select Project - Select Device and select an 8051, 251, or C16x/ST10 device from the Device

. Database

. Create source files to add to the project.

. Select Project - Targets, Groups, and Files. Add/Files, select Source Group1, and add the source files to the project.

. Select Project - Options and set the tool options. Note when you select the target device from the Device Database all-special options are set automatically. You only need to configure the memory map of your target hardware. Default memory model settings are optimal for most.

**APPLICATIONS:**

**.** Select Project - Rebuild all target files or Build target.

**DEBUGGING AN APPLICATION IN UVISION2:**

To debug an application created using uVision2, you must:

. Select Debug - Start/Stop Debug Session.

. Use the Step toolbar buttons to single-step through your program. You may enter G, main in the Output Window to execute to the main C function.

. Open the Serial Window using the Serial #1 button on the toolbar.

. Debug your program using standard options like Step, Go, Break, and so on.

**LIMITATIONS OF EVALUATION SOFTWARE:**

The following limitations apply to the evaluation versions of the C51, C251, or C166 tool chains. C51 Evaluation Software Limitations:

. The compiler, assembler, linker, and debugger are limited to 2 Kbytes of object code but source Code may be any size. Programs that generate more than 2 Kbytes of object code will not compile, assemble, or link the startup code generated includes LJMP's and cannot be used in single-chip devices supporting Less than 2 Kbytes of program space like the Philips 750/751/752.

. The debugger supports files that are 2 Kbytes and smaller.

. Programs begin at offset 0x0800 and cannot be programmed into single-chip devices.

. No hardware support is available for multiple DPTR registers.

. No support is available for user libraries or floating-point arithmetic.

**EVALUATION SOFTWARE:**

**.**  Code-Banking Linker/Locator

. Library Manager.

. RTX-51 Tiny Real-Time Operating System

**PERIPHERAL SIMULATION:**

The u vision2 debugger provides complete simulation for the CPU and on chip peripherals of most embedded devices. To discover which peripherals of a device are supported, in u vision2. Select the Simulated Peripherals item from the Help menu. You may also use the web-based device database. We are constantly adding new devices and simulation support for on-chip peripherals so be sure to check Device Database often.

**5.2 CODING:**

#include<reg52.h>

#include<lcdp2.h>

#include<i2c.h>

sbit sw1=P1^0;

sbit sw2=P1^1;

sbit sw3=P1^2;

sbit sw4=P1^3;

sbit sw5=P1^4;

sbit sw6=P1^5;

sbit sw7=P1^6; //S1

sbit sw8=P1^7; //52

sbit buz=P0^0;

int ref,rfe0id,rfe1id,rfe12id,rfe3id,rfe4id,

unsigned char key;

unsigned int j=0;

unsigned char idata buff[20];

code unsigned char get\_im\_cmd[12]={0xEF,0x01,0xFF,0xFF,0xFF,0xFF,0x01,0x00,0x03,0x01,0x00,0x05};

code unsigned char generate\_ch\_cmd[13]={0xEF,0x01,0xFF,0xFF,0xFF,0xFF,0x01,0x00,0x04,0x02,0x01,0x00,0x08};

code unsigned char identify\_im\_cmd[17]={0xef,0x01,0xff,0xff,0xff,0xff,0x01,0x00,0x08,0x1b,0x01,0x00,0x00,0x01,0x01,0x00,0x27};

unsigned char a1=10,a2=20,a3=30,a4=40,a5=50,a6=60,ch=6;

unsigned char student1,student2,student3,student4,student5,student6;

unsigned int check=0;

void SEND\_CHR(unsigned char c);

void get\_im(void);

void generate\_ch(void);

void identify\_im(void);

void FLUSH\_BUFF(void)

{

unsigned int i=0;

while(i < 20)

buff[i++] = 0x00;

}

void main(void)

{

buz=0;

SCON = 0x50;

TMOD = 0x20;

TH1 = 0xFD;

TR1 = 1;

init\_lcd();

cmd\_lcd(0x01);

display\_lcd(" FINGER PRINT ");

cmd\_lcd(0xC0);

display\_lcd("VOTING MACHINE");

delay\_ms(500);

cmd\_lcd(0x01);

cmd\_lcd(0x01);

display\_lcd("PRESS S1 TO ");

cmd\_lcd(0xC0);

display\_lcd("RESET E.V.M");

do{

if(sw7==0)

{

write\_i2c(0xa0,a1,0);

write\_i2c(0xa0,a2,0);

write\_i2c(0xa0,a3,0);

write\_i2c(0xa0,a4,0);

write\_i2c(0xa0,a5,0);

write\_i2c(0xa0,a6,0);

cmd\_lcd(0x01);

display\_lcd("E.V.M RESET ");

cmd\_lcd(0xC0);

display\_lcd("SUCESSFULLY");

delay\_ms(500);

}

check=check+1;

delay\_ms(3);

}while(check!=500);

delay\_ms(50);

check=0;

student1=(read\_i2c(0xa0,a1));

student2=(read\_i2c(0xa0,a2));

student3=(read\_i2c(0xa0,a3));

student4=(read\_i2c(0xa0,a4));

student5=(read\_i2c(0xa0,a5));

student6=(read\_i2c(0xa0,a6));

cmd\_lcd(0x01);

display\_lcd("PRESS S2 F0R ");

cmd\_lcd(0xC0);

display\_lcd("TOTAL COUNT");

do{

if(sw8==0)

{

cmd\_lcd(0x01);

display\_lcd("CONGRESS = ");

integer\_lcd(read\_i2c(0xa0,a1));

cmd\_lcd(0xc0);

display\_lcd("shiv = ");

integer\_lcd(read\_i2c(0xa0,a2));

delay\_ms(1000);

cmd\_lcd(0x01);

display\_lcd("B.J.P = ");

integer\_lcd(read\_i2c(0xa0,a3));

cmd\_lcd(0xc0);

display\_lcd("S.S.N = ");

integer\_lcd(read\_i2c(0xa0,a4));

delay\_ms(1000);

cmd\_lcd(0x01);

display\_lcd("N.C.P = ");

integer\_lcd(read\_i2c(0xa0,a5));

cmd\_lcd(0xc0);

display\_lcd("NOTA = ");

integer\_lcd(read\_i2c(0xa0,a6));

delay\_ms(1000);

}

check=check+1;

delay\_ms(3);

}while(check!=500);

while (1)

{

Y: cmd\_lcd(0x01);

cmd\_lcd(0x80);

display\_lcd(" PLACE FINGER ");

cmd\_lcd(0xC0);

display\_lcd(" PRESS switch-1 ");

while(sw7==1);

cmd\_lcd(0x01);

display\_lcd("Identifying...");

delay\_ms(100);

delay\_ms(25);

get\_im();

delay\_ms(750);

delay\_ms(250);

generate\_ch();

delay\_ms(250);

identify\_im();

if(buff[9]==0x00)

{

cmd\_lcd(0x01);

display\_lcd("Successfully ");

cmd\_lcd(0xc0);

display\_lcd("Identified ");

delay\_ms(500);

cmd\_lcd(0x01);

display\_lcd("User ID: ");

key=buff[11];

write\_lcd(buff[11]/10+48);

write\_lcd(buff[11]%10+48);

delay\_ms(1000);

ref=buff[11]%10+48;

if(ref==0)

{

if(rfe0id==0)

{

goto X;

}

else

{

cmd\_lcd(0x01);

display\_lcd(" HELLO USER U R ");

cmd\_lcd(0xc0);

display\_lcd(" VOTE completed ");

delay\_ms(200);

goto Y;

}

rfe0id=1;

}

if(ref==1)

{

if(rfe1id==0)

{

goto X;

}

else

{

cmd\_lcd(0x01);

display\_lcd(" HELLO USER U R ");

cmd\_lcd(0xc0);

display\_lcd(" VOTE completed ");

delay\_ms(200);

goto Y;

}

rfe1id=1;

}

if(ref==2)

{

if(rfe2id==0)

{

goto X;

}

else

{

cmd\_lcd(0x01);

display\_lcd(" HELLO USER U R ");

cmd\_lcd(0xc0);

display\_lcd(" VOTE completed ");

delay\_ms(200);

goto Y;

}

rfe2id=1;

}

if(ref==3)

{

if(rfe3id==0)

{

goto X;

}

else

{

cmd\_lcd(0x01);

display\_lcd(" HELLO USER U R ");

cmd\_lcd(0xc0);

display\_lcd(" VOTE completed ");

delay\_ms(200);

goto Y;

}

rfe3id=1;

}

if(ref==4)

{

if(rfe14d==0)

{

goto X;

}

else

{

cmd\_lcd(0x01);

display\_lcd(" HELLO USER U R ");

cmd\_lcd(0xc0);

display\_lcd(" VOTE completed ");

delay\_ms(200);

goto Y;

}

rfe14d=1;

}

FLUSH\_BUFF();

X: cmd\_lcd(0x01);

display\_lcd(" HELLO USER ");

cmd\_lcd(0xc0);

display\_lcd(" GIVE U R VOTE ");

delay\_ms(200);

while((sw1==1)&&(sw2==1)&&(sw3==1)&&(sw4==1)&&(sw5==1)&&(sw6==1));

if(sw1==0)

{

cmd\_lcd(0x01);

display\_lcd("YOU HAVE ELECTED");

cmd\_lcd(0xC0);

display\_lcd("CONGRESS");

buz=1;

student1=student1+1;

write\_i2c(0xa0,a1,student1);

delay\_ms(1000);

buz=0;

while(sw1==0);

}

if(sw2==0)

{

cmd\_lcd(0x01);

display\_lcd("YOU HAVE ELECTED");

cmd\_lcd(0xC0);

display\_lcd("Shiv");

buz=1;

student2=student2+1;

write\_i2c(0xa0,a2,student2);

delay\_ms(1000);

buz=0;

while(sw2==0);

}

if(sw3==0)

{

cmd\_lcd(0x01);

display\_lcd("YOU HAVE ELECTED");

cmd\_lcd(0xC0);

display\_lcd("B.J.P");

buz=1;

student3=student3+1;

write\_i2c(0xa0,a3,student3);

delay\_ms(1000);

buz=0;

while(sw3==0);

}

if(sw4==0)

{

cmd\_lcd(0x01);

display\_lcd("YOU HAVE ELECTED");

cmd\_lcd(0xC0);

display\_lcd("S.S.N");

buz=1;

student4=student4+1;

write\_i2c(0xa0,a4,student4);

delay\_ms(1000);

buz=0;

while(sw4==0);

}

if(sw5==0)

{

cmd\_lcd(0x01);

display\_lcd("YOU HAVE ELECTED");

cmd\_lcd(0xC0);

display\_lcd("N.C.P");

buz=1;

student5=student5+1;

write\_i2c(0xa0,a5,student5);

delay\_ms(1000);

buz=0;

while(sw5==0);

}

if(sw6==0)

{

cmd\_lcd(0x01);

display\_lcd("YOU HAVE ELECTED");

cmd\_lcd(0xC0);

display\_lcd("NOTA");

buz=1;

student6=student6+1;

write\_i2c(0xa0,a6,student6);

delay\_ms(1000);

buz=0;

while(sw6==0);

}

}

else

{

cmd\_lcd(0x01);

display\_lcd(" Not Identify ");

cmd\_lcd(0xC0);

display\_lcd(" ");

buz=1;

delay\_ms(1000);

buz=0;

}

}

}

void identify\_im(void)

{

unsigned char i;

FLUSH\_BUFF();

RI=0;TI=0;

delay\_ms(5);

for(i=0;i<17;i++)

{

SEND\_CHR(identify\_im\_cmd[i]);

}

for(j=0;j<12;j++)

{

while(RI==0);

buff[j]=SBUF;

RI=0;

}

}

void generate\_ch(void)

{

for(j=0;j<13;j++)

{

SEND\_CHR(generate\_ch\_cmd[j]);

}

delay\_ms(250);

}

void get\_im(void)

{

for(j=0;j<12;j++)

{

SEND\_CHR(get\_im\_cmd[j]);

}

delay\_ms(250);

}

void SEND\_CHR(unsigned char c)

{

SBUF = c;

while(TI==0);

TI=0;

}

**CONCLUSION**

Thus the advent of this biometric voting system would enable hosting of fair elections in India. This will preclude the illegal practices like rigging. The citizens can be sure that they alone can choose their leaders, thus exercising their right in the democracy.

**FUTURE SCOPE OF THE PROJECT**

* This system can be used for elections since it provide complete security and will provide accurate results and save time and expenditure.
* This can be modify by interfacing it with a PC through a serial port in order to provide additional security.

**REFERENCE**

[1.] Schurmann, C.; IT Univ. of Copenhagen, Copenhagen, Denmark . ―Electronic Elections: Trust Through Engineering‖, First international workshop Requirements Engineering for e-Voting Systems (RE-VOTE), 2009.

[2.] .Introducing Electoral voting, IDEAinternational institute for democracy and Electoral Assistance, policy paper, Dec 2011.

[3]. Firas Hazzaa, Seifedine Kadry, New System of E-voting Using Fingerprint, International Journal of Emerging Technology and Advanced Engineering, 2 (2012) 355-363.

[4.] K. Memon, Dileep Kumar, S. Usman, Next Generation A secure E-Voting System Based On Biometric Fingerprint Method, International Conference on Information and Intelligent Computing (IPCSIT). (2011).

[5.] Emin Martinian, Sergey Yekhanin, Jonathan S. Yedidia, Secure biometrics via syndromes, 43rd Annual Allerton Conference on Communications, Control, and Computing, Monticello, IL. (2005).