DESIGN AND CONSTRUCTION OF A 5 ACCOUNT CONDITIONAL ACCESS DIGITAL DOOR LOCK

ELUI OBIORA ANTHONY
2004/21143EE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING,
SCHOOL OF ENGINEERING AND ENGINEERING TECHNOLOGY

FEDRAL UNIVERSITY OF TECHNOLOGY,
MINNA, NIGERIA.

NOVEMBER, 2008.

DESIGN AND CONSTRUCTION OF 5 ACCOUNT CONDITIONAL ACCESS DIGITAL DOOR LOCK

BY

ELUI OBIORA ANTHONY
2004/21143EE

A Thesis Submitted to the department of electrical/computer engineering, Federal University of Technology, Minna.

NOVEMBER,2008

DEDICATION

This project is dedicated to my Lord and personal savior Jesus Christ, my parents, siblings, my friends and all who have in one way or the other contributed to my academic pursuit may the good lord bless you all abundantly.

CERTIFICATION

This is to certify that the project titled "Design and Construction of a 5 account conditional access digital door lock" was done by Elui Obiora Anthony under the supervision of Engr. E. Eronu and submitted to the Electrical and Computer Engineering Department, Federal University of Technology Minna, Niger State. In partial fulfillment of the requirement of the Award of Bachelor of Engineering (B.Eng.) degree in Electrical and Computer Engineering.

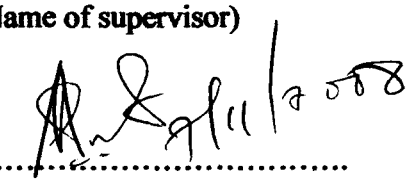
Elui Obiora Anthony

....

(Name of student) Date

DECLARATION

I Elui Obiora Anthony, declare that this work was done by me and has never been presented elsewhere for the award of a degree. I also hereby relinquish the copyright to the Federal University of Technology Minna.

Elui Obiora Anthony	Engr. E. Eronu
.....	.....
(Name of student)	(Name of supervisor)
	
.....	.....
(Signature and date)	(Signature and date)
Prof. Yunusa Adediran	.....
.....	
(Name of H.O.D)	(Name of External Examiner)
.....	.....
(Signature and date)	(Signature and date)

ACKNOWLEDGEMENT

To my father in heaven, the creator of Heaven and Earth, Jesus Christ I thank you. My parents; H.R.H A.C. Elui and Lolo G.N Elui, words are definitely not enough to appreciate and thank you for your prayers, support and the trust you both have given to me. What can I do without my siblings; Chinelo, Chidi, and Chisom, three of you are my source of inspiration, my joy and my hope.

To my principled and incorruptible head of Department, Prof. Yunusa Adediran, you are one in a million, I thank you sir for the discipline and sense of responsibility you have instilled in my colleagues and I. My project supervisor, Engr. E. Eronu. I thank you for your assistance, support and encouragement throughout the course of executing this project. God bless you sir.

To the class of 2008 graduates of Electrical and Computer Engineering; the top is where all of us belong; I will definitely miss you. Yemi, Chris, Ugo, Ugo (UG), Tala, Blessing, Esther, and Ebele, we remain friends and My Boss Engr Bamidele Ajanaku you are the best.

ABSTRACT

The goal of this project was to design and develop an electronic door lock, which is controlled by a 5-digit password, and holds five different access codes to be entered by the user. The password is entered using a standard 16-button keypad, which is mounted on the door. Basically, the functionality of this device can be seen once the correct password is entered. The user then has a choice between unlocking the door and changing the current password. By unlocking the door, the locking mechanism remains open for about 20 seconds. Once the 20-second time interval elapses, the door lock automatically returns to the locked position. Upon changing the current password, the new password is valid immediately. The main features of this electronic door lock are its high reliability, low operating power consumption, and ease of use. Thus, this electronic locking mechanism definitely possesses marketable potential.

TABLE OF CONTENT

Chapter One

1.1 Introduction

Over the years, crime rate has experienced an alarming increase, from 10.4million in the United States of America alone as at 1980 to about 14million in 2005 [1]. 11% of these cases were reported to be property-crime related incidences, which occurred due to the apparent insecurity in what people call 'secure doors'. This has hereby become a major cause for concern as various research works have gone into the construction of a mechanism that can best secure a building, with special reference to the door. Nigeria was ranked the top most insecure nation by the United States Department of State on February 17th, 2006 [3], it also advised all intending visitors to the country to reconsider their intentions, till this day security of homes is still a serious issue in Nigeria. Property theft, amongst others, was one of the warnings cited by the United States. Implementing a more secure device, like the one discussed in this project will help, to a large extent, alleviate this problem.

1.2 Historical background

From the use of crossbeams to bolts and nuts, door locks have undergone changes leading to the advent of Roman Padlocks, then came Beddington Locks [2]. These changes did not stop there, door locks, as conditional access systems, have metamorphosized into complex and more sophisticated devices, having electronics as their backbone. One of the driving forces behind this project work dates back to the lapse discovered in one of the electronic controlled locks (conditional access systems), 'the punch card access system'. This system uses a transmitter, a receiver and a punch card as its major operating mechanisms. However, this system experienced easy duplication of cards (especially by

unauthorized users), it also lacked the authenticity and integrity possessed by more 'intelligent' systems. Most of these more intelligent systems applied the use of microcontrollers as their control device.

Microcontrollers are introduced as small, low powered devices, dedicated to perform a specific duty as decided by the program stored in its ROM (Read-Only Memory). An example of a conditional access system that implements a microcontroller is seen in the use of smart cards alone as the access system, and access system without smart cards (4-6 PIN based access code). In other cases, we have also seen the use of voice recognition as the access system. These three systems are used along with a microcontroller to aid perform dedicated tasks for the purpose of security. However, this project work is based on a 5-digit access code non-smart card security system. It is designed to suit small residence, or a family house, where not users are expected.

1.3 Aims and Objectives

This project aims to achieve a high level security in at homes.

It helps reduce incidences of unauthorized access by individuals.

Since access codes can be over written easily, security of key is guaranteed.

1.4 Methodology

Using a program written in Assembly language, the microcontroller controls the flow of all activities in the device. At power up the micro controller sends a "terminal ready" text to display at the L.C.D (Liquid Crystal display) screen. The microcontroller then prompts the user for the access code, before this a list of possible access codes are already stored in the non-volatile memory. If any 5-digit pin is inputted, the controller

checks the validity of the access code in the non-volatile memory. If the access code is inputted is located, meaning that a correct password has been entered, it grants the user access, by signaling the controller to supply voltage to the relay, which in turn energizes the solenoid, making it to pull back its plunger, hence opening the door. On the door frame is situated a switch that when open indicates to the controller that the door is still open, hence no other password can be entered. When the door is closed the switch also is closed this sends a message to the controller that the door is closed hence the power supply to the relay could be cut-off. This will cause the solenoid to drop the plunger back into the notch in the door, also allowing another user to access the security device; for this project the plunger of the solenoid serves as the latch for the door. This process can also be achieved using a motor in place of the solenoid, owing to the similarity in working principles. The major difference is that in the case of a motor the plunger which is driven by the motor is kept in a horizontal position, when powered the motor rotates in an anti-clockwise direction hence pulling in the latch, but the locking principle is different as it still needs power from the microcontroller to rotate in the clockwise direction, there by pushing the latch that in turn locks the door.

Therefore the microcontroller is responsible for controlling the process of opening and closing the door. All these steps are described in detail in the Chapter 3 of this project report. It is interesting to note at this point that though the microcontroller chips are designed to perform a specific task, during design, the program can be altered to suite the end use's requirement, for example, varying the length of the access code, changing the welcome message and so on. This helps in creating variations of this project and at the same time it improves its versatility.

1.5 Scope of work

This project involves the use of a solenoid as the main door latch, a relay to drive the solenoid and a microcontroller. These devices can be used for widely extensive applications as discussed in the Chapter 2 of this report. However, in this project, they are used to provide limited access to premises and other secure areas. At an inexpensive rate, most average homes or offices can own one of these, varied to their taste, thereby aiding improved security. Later sections detail the methodology involved in building this device, making room for reconstruction and improvement of this work. The structure of the rest of this report is as follows: Chapter 2, Literature Review/Theoretical Background; Chapter 3, Design and Implementation; Chapter 4, Tests, Results and discussion; Chapter 5, Conclusions and recommendation.

Chapter Two

2.1 Literature Review/Theoretical background

2.1.1 History of Locks

Locks can be said to be 'omnipresent'. They control physical access to premises or spaces. All residences, businesses, hotels, governmental offices, vehicles and storage spaces, cabinets and safes utilize them. Although locks are most commonly employed on doors, they also control access to windows, lockers, storage cabinets, safes and so on. Locks have long been traditionally operated using a mechanical key. In the history of mechanical locks, it is interesting and important to trace the means adopted in making the lock secure, as age succeeded age. There are and have been throughout the centuries, only two mechanical principles by which security in key operated locks is obtained. One is by means of fixed obstructions to prevent wrong keys from entering or turning in the locks.

The other, which is superior, employs one or more movable detainers which must be arranged in pre-selected positions by the key before the lock will move [2]. A roman padlock in metal form is an example of the mechanical locks. It possesses a shackle, which is separate from its body. On its lower side is a pair of spreading springs. When the shackle is pressed in, the springs which during the operation had been gradually closed up, spread out inside the body and so hold the two pieces together. To take them apart, the springs, which were simply flexible barbs, have to be compressed. This is done in some locks by a key which is turned after being pushed through an engraved entry in the body similar to the shape of the key [2]. After the advent of the Roman padlocks, other mechanical locks were introduced. Research work by Linus Yale in 1848 developed the Egyptian Lock, using the pin tumbler mechanism. This mechanism decreased the dimensions of common locks as at that time [2].

Here the bolt is mounted on the inside surface of the door. In this case, if fastened by the tumbler pin, it would not be so necessary to conceal it as when bolt and pin were outside the door. There are some curious sickle-shaped pieces of iron found now and again which look as if they were made for the purpose of putting through a hole in the door and pulling up or pushing up the pin. Perhaps they simply engaged the bolt in a direct fashion, and, being turned from the outside, move it to and fro, but they vary in their outlines too much for this supposition to be probable, some being full sickle-shackle and other only slightly cranked or bent, and in some well- preserved specimens their ends have been carefully shaped, as it to fit a hole exactly. Several other mechanical locks came to be as the years passed by. Some of which are: Beddington locks, Chubb locks, Thomas Parson's lock amongst others. Although simple and relatively reliable, most of these mechanical keys could easily be duplicated and the locks easily picked, leading to electronic locks becoming more prevalent.

The electronic locks are electrically activated to a locked or an unlocked condition using a myriad of key technologies. One of the earliest inclusions into the family of electronic locks is the "keyless combination lock". It was developed by Linus Yale Jr. in 1862 [4]. Additional work was done on this lock by inventors like Miller, James C., Harvey and Michael P. in 1991. This lock consists of a rotary dial which is connected to a stepper motor/generator providing the electrical power to a capacitor for powering the system. The stepper motor/generator also provides input signals in the form of a code sequence to a microprocessor that processes the signals to initiate the operation of a drive motor to release a lock bolt once the proper combination is dialed. A Read-Only Memory (ROM) determines the proper combination from a combination storage means and feeds the combination to the microprocessor for comparison to the inputted signals from the dial. As each combination dialing sequence is begun, a random code initiator provides a

different starting position in the sequence so that electronic or visual surveillance equipment cannot be used to surreptitiously obtain the combination. In order to thwart computerized input dialing to open the lock assembly, a dial speed sensitive lockout device also controls the microprocessor. During the combination dialing, a display unit presents the code and direction of movement of the sequence for observation by the person dialing the combination.

2.2 Previous efforts

Several researchers investigated into the long access period associated with the combinational lock and developed plastic punch cards which allowed for much better control of keys. Here, a particular card designed with coded punched holes on it is inserted inside its mechanism, depending upon the position of punched holes on the card; a particular door open or close. The circuit uses one photo transistor. When there is no card in the lock, there will be a transmission in the circuit with a switch putting it off (not connected), as soon as a card is inserted into the lock the transmission beam will be break which puts the circuit ON, a switch will be pressed to toggle a mechanical device which rolls the door open and stops when it reaches its end and a second switch is pressed to do the opposite when the card is removed. Two digital comparators are interfaced with two timers (MONOSTABLES) which compare the two digital input and output [10].

However, due to the easy duplication of access cards also experienced with this invention, it was soon replaced by more advanced inventions. As the quest for knowledge increased, crime rate and technological know how also increased, it therefore became necessary to improve the potency of available locking systems. As such, more research work gave way to the introduction of a memory device fussed with a locking

system. The device could either be a microcontroller, a microcomputer or both. This aided in monitoring the inflow and outflow of traffic in a place. It also helped improve security, as two-way authentication was introduced between the user and the computer before access was granted. Extra memory chips were also used to store required information of a user. An example of this kind of locking system is seen in the keyless locking system. Here, the locks are programmed to allow particular individual's access to particular rooms using digital codes either encrypted on a card key or entered via a digital keypad interface. These locking systems have an embedded microcontroller which checks the validity of an inserted card or code. They have the advantage of being stand-alone systems and as such, they reduce the physical size of most 'intelligent' locking systems [6, 7]. They are mostly used in hotels. Improvements have also been made on the access cards used. They are popularly called 'smart cards'. These smart cards were first invented in France in the late seventies and millions have been used over the past few years as payphone cards, picture Identification cards, banking debit and credit cards and GSM mobile phone identifiers. The smart cards that are highlighted in this report are, however, much more advanced than their predecessors from the seventies and eighties.

Nevertheless, the concept remains simple: the microchip is embedded inside the device so as to inhibit access to it. The microchip in use here is the I^2C chip with an input and an output channel; It can be used to store and/or manage the identity of its user. The chip interacts with the microcontroller using a program written in Assembly language. Improvements have been made on the power, speed and capacity of the chip [8]. Additional research has gone into the development of systems that verify each user that accesses a particular facility at any point in time. This in turn, increased the complexity of access systems. The project in view is an example of a simple home security device. It consists of an access code, a non-volatile memory, a microcontroller, and a solenoid as its

four main operating mechanisms. Here, each user has a 5-digit pin written on the non-volatile secure memory for exchanging of data with the microcontroller. The microcontroller now verifies the validity of the data in the card's memory; it then compares this data against that stored in the non volatile memory. After this information is authenticated, upon final validation, access is then granted to the user.

2.3 Theoretical Background

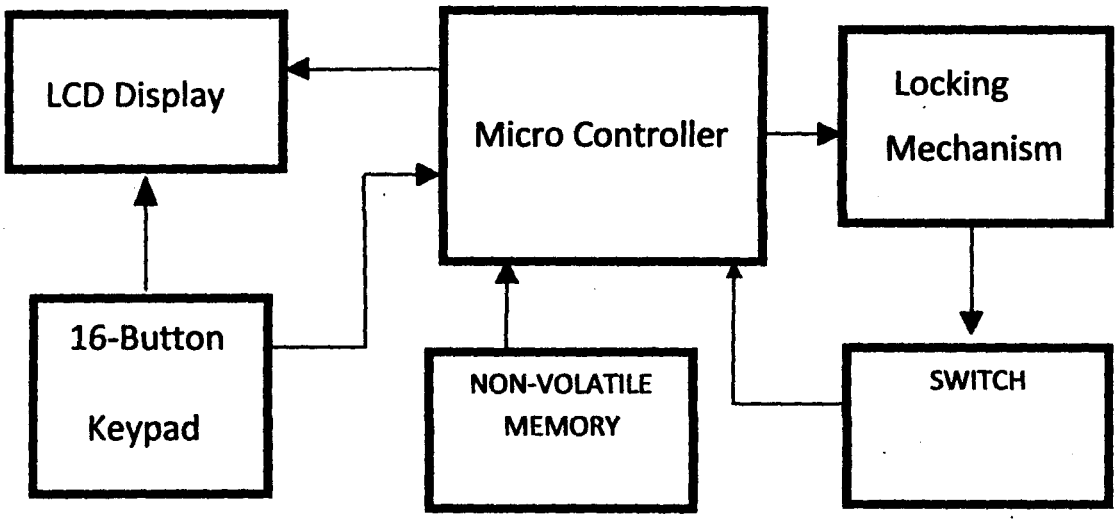


Fig. 2.1 Block Diagram of the Conditional Access System

From the Block diagram, we see that signal flow revolves around the microcontroller. It sends out signals to the LCD screen, when a 5-digit key is entered, it analyses the pin inputted by the user in the keypad, it sends out signals to the relay drivers, instructing them to either open or close the door.

2.4 Microcontrollers

A microcontroller can be simply identified as a computer on a chip [11]; which can be used to perform a number of desired task as defined or instructed by the programmer, using either assembly language (ASM) or complier (C- language). For this project the 8051 family was used, specifically the 8951.

2.4.1 The At89c51 Microcontroller.

The AT89C51 also known as the 8051 microcontroller was introduced in the early 1980s by Intel Cooperation. It currently has many different versions (members of the family) and many of these types include an on-chip analogue-to-digital converter, a considerable large size and data memory, pulse width modulation outputs and flash memories that can be erased and reprogrammed by electrical signals. The version used for this project is the AT89C51, which is a low power CMOS version of the family and it is a 40-pin microcontroller. Other members of the family are shown in the table below

Table 1

DEVICE	PROGRAM MEMORY	DATA MEMORY	TIMER / COUNTER	INPUT/OUTPUT PINS	NUMBER OF PINS
AT89C1051	1K Flash	64K RAM	1	15	20
AT89C2051	2K Flash	128K RAM	2	15	20
AT89C51	4K Flash	128K RAM	2	32	40
AT89C52	8K Flash	256K RAM	3	32	40
8051AH	4K Flash	128K RAM	2	32	40
87C51AH	4K Flash	128 K RAM	2	32	40

2.4.2 The At89c51 Microcontroller Architecture

The AT 89C51 microchip has 40-pins and the pin configuration is illustrated in the figure shown

One major advantage of the microcontroller is that it uses software programmed instructions which replace the use of several gates to implement.

Microcontrollers are used in many appliances today, for example, alarm clock, coffee maker, refrigerator, remote control, smoke detector, telephone and so on [9]. The picture below shows the physical structure of the 8951 micro controller.

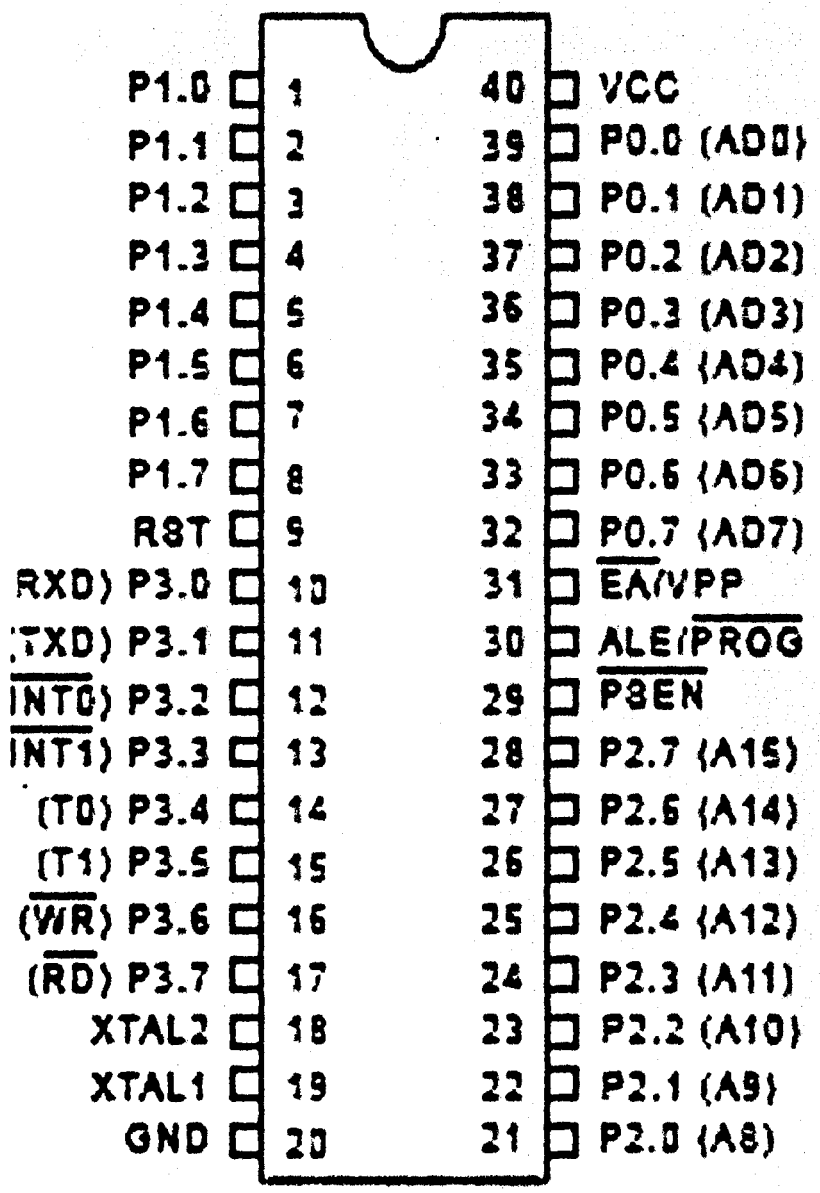


Fig 2.2: showing an 8951 microcontroller.

PINS	DESCRIPTION
PORT P0s (P0.0 –P0.7)	This is a dual function port which is used to access external devices. In other designees, it may be used to address external memory. Theses ports access low addresses A0 – A7 of the external devices. They have no pull up resistors so it is necessary to connect pull-up resistors to these pins depending on the characteristics of the parts to be driven by the pins
PORT P1s (P1.0 – P1.7)	These ports are used to interface the microcontroller with external devices or hard wares such as LEDs, LCDs, keypads and other such devices.
PORTS P2s (P2.0 – P2.7)	Similar to port P0, it is used to provide the high address byte (A8 –A15) during the fetches from external program memory and during access to external memory. Pull-up resistors have to be connected to the as well.
PORT P3s (P3.0 – 3.7)	These ports consist of dual function input/output lines. Each pin has a pre-defined function as specified by the software codes developed. The microcontroller handles automatically the reading and writing to the P3s' special function registers (SFR) which specify each pin functions.
RST	This is the reset input. A reset is accomplished by holding the RST pin high for at least two machine cycles. A reset is performed by connecting an external capacitor and resistor to this pin.
VCC	This is the power supply input pin of the microcontroller.
GND	This is the ground output of the microcontroller.
PSEN	This is the program store enable pin. It is activated when the device is accessing codes from external memory.
XTALS	These are oscillator inputs XTAL 1 and XTAL 2. External crystals are connected to these pins to avoid interrupt in the operation of the microcontroller
EA	This is an external access line used to determine whether the microcontroller is executing the codes from an external memory or an internal memory. If the EA is high, then it is executing from an internal memory else it is from external

2.4.3 Locking Mechanism

The locking mechanism is an electric DC motor, powered by $6V_{DC}$, Controlling whether the locking device is in the locked or unlocked state is accomplished using a solid state relay, which will be discussed in more detail in chapter 3. Figure 4 shows the locking device in the locked position. In the locked state; the motor which is used to drive the motor remains stationary. The motor is not powered, the door is kept locked, the end of the door slides into the door frame, hence preventing even an intending burglar the opportunity of priming the door open. Thus, it will resist any force exerted on the door, such as someone trying to force open the lock. When voltage appears across the inputs of the motor as a result of the input of any of the correct digital keys, the motor rotates in an anticlockwise manner; unlocking the door.

Chapter Three

3.1 Design and Implementation

The electronic security keypad lock system was desired among the following subsystems:

1. 5 volt system supply voltage.
2. 4x4 matrix keypad for data input.
3. 8 bit system controller.
4. 256-byte 24C02 I.C non-volatile memory
5. The relay for output switching
6. 2x16 characteristics LCD

3.2 Power supply

The system power input was retrieved from local mains via a 12 V, 0.5 amp step down transformer. The transformer was connected to a 4 diode full wave rectifier as depicted in fig. 3.1

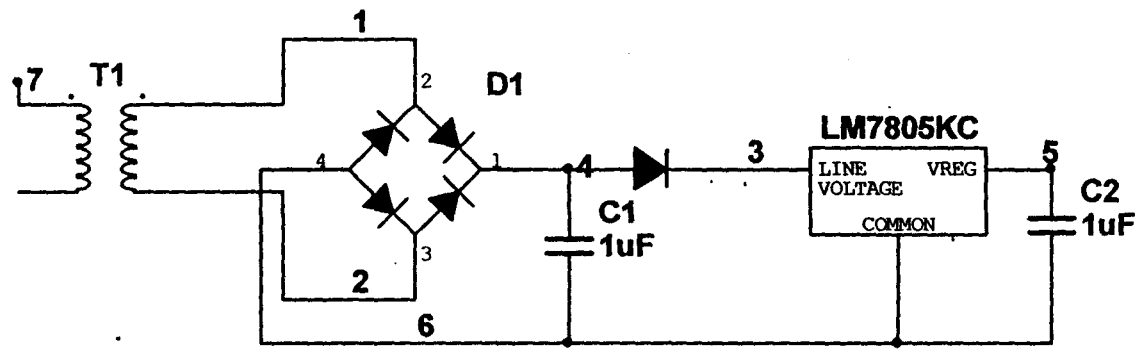


Fig. 3.1 The Power Supply Stage

The peak rectified DC voltage is expressed by the relation:

The 12V ac voltage (240 line level) is connected to pulsating dc amplitude;

$V_{AC}= (V_{RMS}\sqrt{2} - 1.4)V$.....(1)

V_{DC}=peak rate of the dc voltage.

V_{RMS}=rms voltage of the transformer secondary winding.

Using a $12V_{rms}$ input voltage, the peak dc voltage was $12\sqrt{2} - 1.4 \equiv 15.5v$

The $15.5v_{dc}$ supply was smoothed by an electrolytic capacitor of value derived from the expression:

$$C = \frac{Idt}{V}$$

Where I=maximum load current.

T= 1/f=1/2*f(FWBR)

V=maximum AC ripple voltage on the DC voltage

The load current was calculated by summing together the system current drain as follows;

AT 8951 controller: 15mA

24c02 memory: 3mA

LCD -10mA peak

Relay (when on): 15mA

Total current: 43mA

A 7805 5-volt regulator was connected across the rectifier. The 7805 device has a minimum input voltage of about 7v (from the manufacturer; micro controller specification). On a 15.5v input;

The maximum allowable ripple is then (15.5 -7)=8.5V.

The capacitance was calculated thus:

$$C = (43 * 10^{-3} * 2 * 50)/8.5$$

$$C = \frac{0.043 * 0.01}{8.5}$$

$$\frac{0.00043}{8.5} = 0.0000506 \; or \; 50.6\mu f$$

This value of capacitance is the minimum required to keep the 5 volt system supply regulated. A value of 2200uf capacitance was used for improved system performance.

3.3 keypad

The keypad itself consumes no power. It works by simply making connections between wires. The keypad has eight inputs: one for each of the four rows of buttons and one for each of the four columns of buttons. When a button is pressed, a connection is made between the input for the row and the input for the column in which the button is located. For example, suppose the input for the first column is grounded and a voltage is applied at the input for the first row. If no button is pressed no current will flow through any input. Yet, if button "1," which is located in the first row and first column, is pressed, a

connection is made between the two inputs mentioned above and current flows through each input. This will only happen if button "1" is pressed.

This is the input device, the keypad comprises of 16 buttons on the panel, which is interfaced to the controller over P3 as shown below.

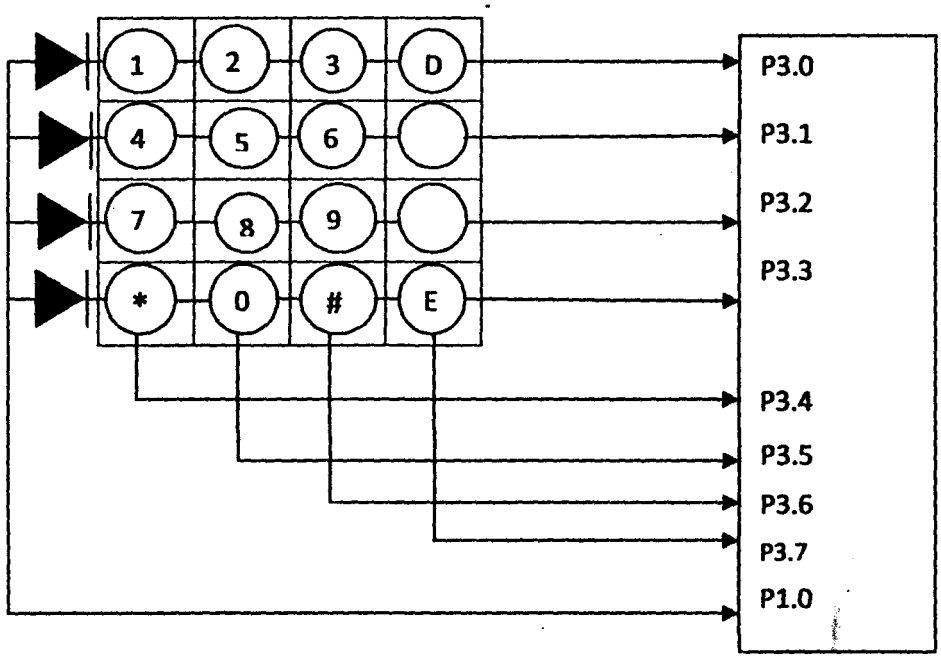


Fig 3.2. 4X4 keypad matrix

The matrix is a modified version of commonly implemented matrix type, in that a single bit input was provided along with the eight bits needed for the interfacing.

For the keypad port (port 3) was initialized with the binary pattern 00001111_B at system power up, with no key pressed, the logic input into P1.0 is high. The software loops around this flag to detect key presses.

However, if a key is pressed, sat 1, P3.0 is low, and so also P1.0 the software enters into a keypad scanning routine to determine the key pressed.

The software stores a unique key for every low row (i.e. P3.0 - P3.3). If P3.0 is low, a 1 is stored in a temporary register, if P3.1 is low, a 4 is stored. If P3.2 is low, then a 7 is stored. For P3.3 is a 10 that is stored.

The software there after flips the logic level on port 3. The value 11110000_B is written to the port. P1.0 is again checked for a low. 16 a low is detected on P3.4, a zero is stored in a second temporary register, if P3.8 is low, a 1 is stored, if P3.6 is low, a 2 is stored. For P3.7, a 3 is stored.

The values of the temporary register 1 and temporary register 2 are added together and used as an index to a look-up table in memory. The rate corresponding to the pressed key is thus read into a "RAM-valuable" key code from flash memory.

D_1-D_4 enable keystroke detection without an actual keypad scan, they serve as 4 AND gate.

3.4 8Bit SYSTEM CONTROLLER

An 8951 micro controller was utilized for system for system control. The device was loaded with software that defined the system functionality. The AT89C51 has the following characteristics: It is a low power, high performance 40-pin DIP, 8 bit microcontroller with 4Kilobytes of in- system reprogrammable flash memory, 128 bytes of SRAM and 32 programmable I/O lines. It also has a fully static operation of 0Hz to 24MHz. The diagram and pin layout was shown in chapter two above. The micro controller was interfaced with the following components:

It was;

i. Interfaced with the keypad.

ii. Interfaced with the non-volatile memory

iii. Interfaced with a 16x2 LCD.

iv. Interfaced with a DC relay for external control interfacing.

The device has the specifications listed below;

- Insert 8951 datasheet level
- The device was run off a 12MHz clock source. Since the micro controller characteristics frequency was internally divided down by 12, the effective system operating frequency is 1 MHz yielding an instruction execution time of 1microsecond (1μs).

The software was used to address the keypad and perform the computations needed based on the input parameters.

256-byte I^2C EEPROM

A 256-byte inter-integrated circuit electrically erasable read only memory was interfaced to the controller over P2.0 and P2.1 the device was used to hold the 5 digit user password and other system signature bytes.

The device has a property interface not supported by the micro controller. A bit banded software emulation of the interface was implemented for micro controller memory interface.

Talking to the non-volatile memory involves the following sequences of actions;

i. Start condition.

ii. Data read and write.

iii. Stop data.

The micro controller implemented the above three conditions via software as included in the appendix.

Error indication was incorporated in the software routines to detect non-existence of an inputted password, or faulty memory device. Should the controller detects a faulty I.C bus transaction, the LCD displays an "FAULTY "message, or a "ACCESS DENIED" message if a wrong password is typed, and this occurs during memory read or write cycle.

3.5 SYSTEM STATUS INDICATOR

A liquid crystal display was used for interaction between man and machine, as it indicates to the user the system status conditions, using various predefined text, already stored in the memory of the micro controller, such as;

i. "TERMINAL READY" displayed after power up.

ii. "ENTER PASSWORD" prompting the user to input a password.

iii. "ACCESS GRANTED" indicating that the correct password has been inputted.

iv. "FAULTY" indicates the possibility of a major component failure.

v. "NO-NVM" indicates the absence of the non volatile memory, e.t.c.

also displays the system status; at power up the LCD post the system start up messages, which are;

i. Project title.

ii. Student's name and matric number.

iii. Student's supervisor's name.

iv. Enter password.

During the keypad read of the user password, the inputted character are replaced by "*" to hide the key values. Low and the diode connected to it are forward biased, damping the voltage on P1.0 at a diode voltage drop above ground (0). This logic level change is detected in software as a key press and the matrix is immediately scanned to recover the key stroke, LCD is connected to P2.6, and P2.7, to reflect the state of the system.

Resistors of values calculated from the expression

$R_S = (V_S - V_{LCD})/I_{LCD}$

V_S = +5V

V_{LCD} = typically 1.7V

I_{LCD} = 5mA – 10mA

A continous LCD current of 10 mA was choosen.

$$R_S = \frac{5 - 1.7}{0.01} = 330\Omega$$

DC relay for output switching

To interface with a controlled device, a 6volt dc relay of 10mA contact voltage was driven by a controller pin. A PNP transistor driver was used to switch current into or out of the relay winding as shown in Fig 3.3

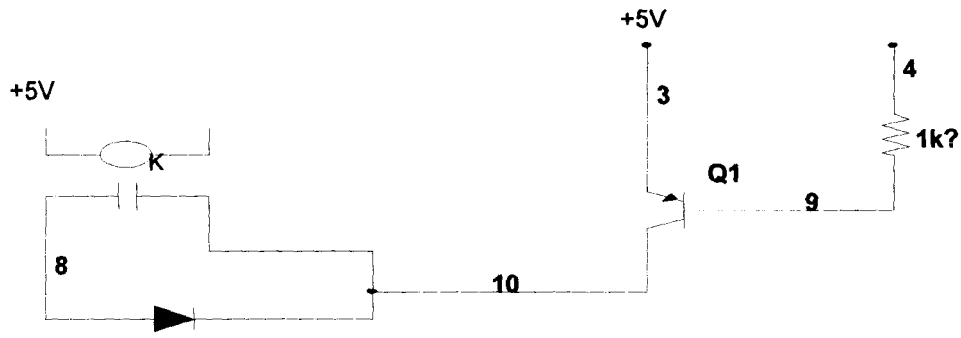


Fig 3.4 output relay driver

A 25A-1015GR PNP transistor of gain 220(Hfe) was used to effect switching. A 6v400Ω relay was used.

The relay current was calculated from;

$I_R = V_R / R_R$

$\frac{6}{400} = 15mA$

This relay current was made the collector current of

$I_C = 15mA$

A value of R_B calculated from the expression below was used;

$R_B = (V_C - V_{BE})/I_B$

$I_B = I_C/H_{FE} = 0.015/220 = 6.8$ mA

$R_B = \left(\frac{5-0.7}{68*10^{-6}}\right)\Omega = 63235$ or $63K\Omega$

A value of was used to ensure the transistor switching at all times.

P5 was connected across the relay winding to absorb inductive kickback. An LED was also connected across the relay winding to show when the relay is energized.

3.6 SOFTWARE OVERVIEW

The software was maintained for easy design; the control software exerts an endless loop waiting for a key to be pressed. When a key press is detected, the keypad is read. The software expects only two kinds of data from the keypad; a password change command or a password input.

If a password change command (*123#) is read over the keypad, the system demands for the previous password, if the password is inputted, a search is made in the non-volatile memory for the password string. If a match is found, the user is prompted to enter the new password. The user's previous password is then over written with the new one.

The inputted passwords are checked to ensure that they lie between (0-9),otherwise an "INVALID PASSWORD" text is posted on the LCD.

During the keypad read of the user password, the inputted character are replaced by "*" to hide the key values. Low and the diode connected to it is forward biased, damping the voltage on P1.0 at a diode voltage drop above ground (0). This logic level change is detected in software as a key press and the matrix is immediately scanned to recover the key stroke.

D_1-D_4 evable keystroke detection without an actual keypad scan. The system power input was retrieved from local mains via a 12 V, 0.5 amp step down transformer. The transformer

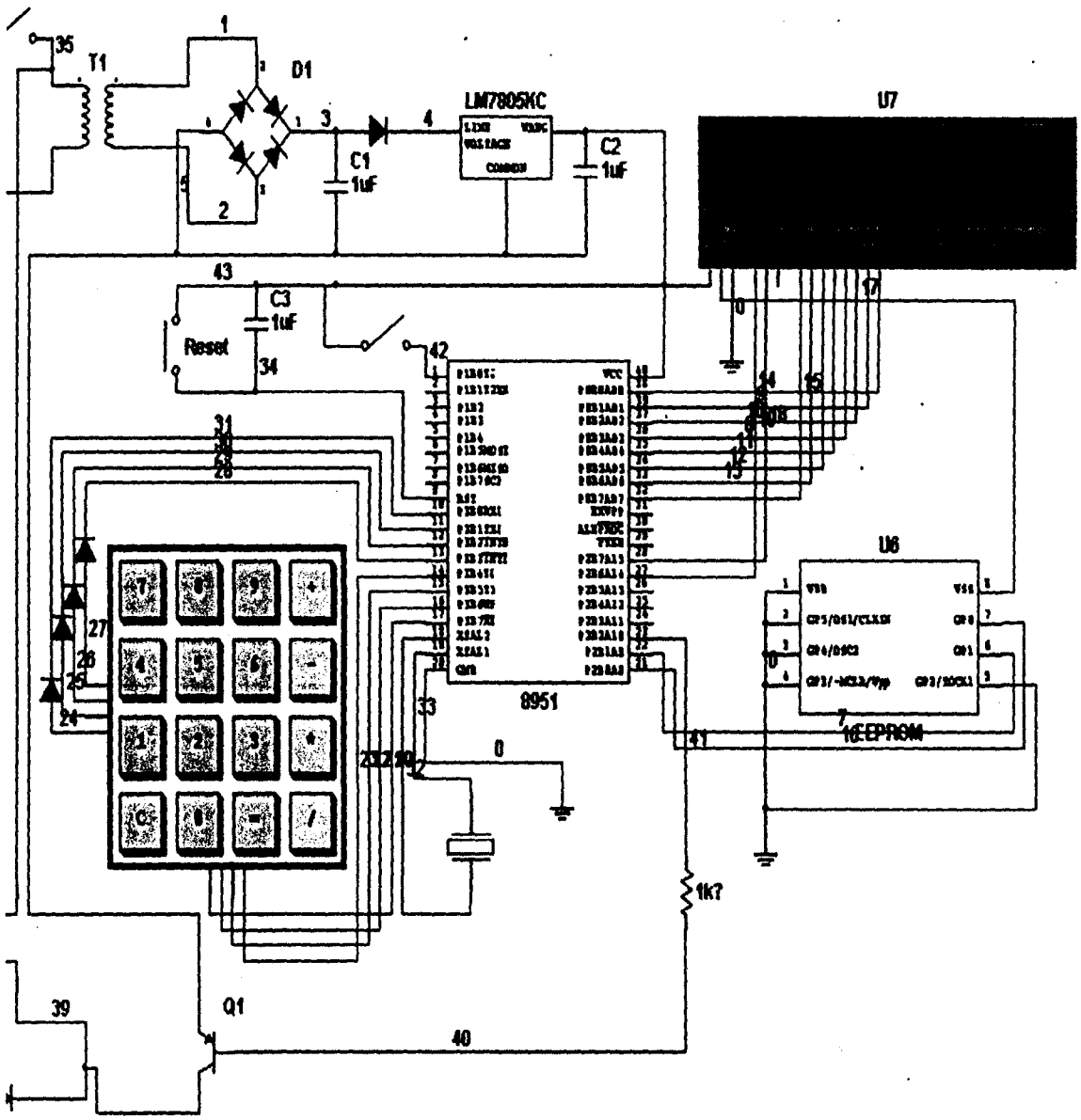


Fig 3.5 complete circuit diagram

3.6 SOFTWARE OVERVIEW

The software was maintained for easy design; the control software exerts an endless loop waiting for a key to be pressed. When a key press is detected, the keypad is read. The software expects only two kinds of data from the keypad; a password change command or a password input.

If a password change command (*123#) is read over the keypad, the system demands for the previous password, if the password is inputted, a search is made in the non-volatile memory for the password string. If a match is found, the user is prompted to enter the new password. The user's previous password is then over written with the new one.

The inputted passwords are checked to ensure that they lie between (0-9), otherwise an "INVALID PASSWORD" text is posted on the LCD.

During the keypad read of the user password, the inputted character are replaced by "*" to hide the key values, and make it low. The diode connected to it is forward biased, damping the voltage on P1.0 at a diode voltage drop above ground (0).

This logic level change is detected in software as a key press and the matrix is immediately scanned to recover the key stroke.

D_1-D_4 evable keystroke detection without an actual keypad scan. The system power input was retrieved from local mains via a 12 V, 0.5 amp step down transformer.

Chapter Four

Results, Test and Discussion

4.1 Result

The device functioned as expected, with the stipulated voltage of 6-12 volts DC. An alternative source of power was incorporated into the main design, enabling the device to be powered either from the battery or using a rectified DC voltage from a step down transformer connected to the mains.

4.2 Test

4.2.1 Micro controller Test

Much of the testing with this project involved debugging the micro controller code and interfacing the micro controller with the wired hardware. The debugging entailed positioning various print statements throughout the code to see exactly which opcodes were executed. Oftentimes, it was difficult to determine if a problem was rooted in the hardware wiring or in the micro code. We used the computer monitor to explicitly output the code executed and the oscilloscope to diagnose any potential hardware problems.

The biggest hurdle we had to overcome was making sure that each input was sent to the right port of the micro controller. This was ensured with numerous compare statements within the micro code. Therefore, if the micro controller read signals different than those expected, this could be easily detected when the program was stuck in an endless loop outputting error statements. Only when the micro controller recognized a valid input did it jump to the next loop.

Making sure that the micro controller output the correct signals was also important. This was verified using a number of test LEDs placed in appropriate locations within in our circuit. If an LED was erroneously triggered, we first traced it back to the micro code to see if any of the instructions interpreted the inputs incorrectly. Next, we looked at how the particular component was wired and determined if it was behaving correctly based on the signal sent to it by the micro controller. Ultimately, this was how many of our debugging problems were rectified. As a result, interfacing the micro controller with the rest of our circuit proved to be a long and iterative trial and error process.

4.2.2 Solid State Relay Tests

One of the tests performed on the relay was to find the actual control voltage range of the input. The control voltage range to was rated to be 3-32 V_{DC}. When tested, the minimum voltage was found to be 1.2 V_{DC}, which is even below the rated 3 V_{DC}. No tests were performed to find the actual upper limit to avoid ruining the relay. However, voltages as high as 15 V_{DC} were tested and the device still performed as expected. Since the only voltage that should appear at the input is 5 V, a range of 1.2-15 V_{DC} is a sufficient control voltage range.

4.2.3 Motor test

The $6V_{DC}$ motor was tested for efficiency and effective driving force, when powered by with a 9volt battery and it was observed to be in good

working condition. It was not overheating, or making any noise, and when tested with the desired circuit voltage (5V), it still performed well.

4.3 Discussion

The electronic door lock performed as expected, we were able to implement all of the functions specified in our proposal. The biggest hurdle i had to overcome with this project was interfacing the micro controller with the hardware components, another was getting a solenoid with a spring, where by the spring in it's relax state would push out the door latch, hence locking the door until the solenoid is energized.

Chapter Five

Conclusion and recommendation

5.1 Conclusion

I feel that this electronic door lock is very marketable because it is easy to use, comparatively inexpensive due to low power consumption, and highly reliable. This door lock is therefore particularly useful in applications such as small stores, residential housing, and even office buildings.

There are always changes one can make to potentially improve a design's performance or marketability.

5.2 Recommendation

Some of the modifications that might have been useful for our electronic door lock include adding an automatic shut-off switch, utilizing a ROM chip instead of a RAM chip, and also adding a manual override. The automatic shut-off switch could have been implemented as a cautionary measure in the event of a micro controller malfunction. In this instance, the automatic shut-off switch would disconnect any power source to the solenoid and subsequently protect against any potential fire hazard. This automatic shut-off switch could also be connected to a temperature sensor and turn off the power if the temperature of the system rose above a particular threshold temperature.

Also the inclusion of a timer and a micro processor to help keep track of the user's time of entrance, and who's pin number was used to access the lock last. This will improve the security feature a great deal.

Also, the addition of a manual override would have added marketability to our electronic door lock. For instance, if there is a total system failure, the door would

remain in its locked position. Therefore, if one needed to unlock the door, especially in an emergency, the manual override would account for this contingency.

One of the main issues we considered in determining the design of the solenoid locking mechanism was safety. Ultimately, our locking mechanism unlocked only when power was applied to it and otherwise remained locked. This was important in the event of power outages. If the power went out in the middle of the night or while nobody was home, for example, we wanted to ensure that the door remained locked. If we had designed the locking mechanism to lock only when power was applied, inevitable power outages would leave the door unlocked and the house or building would then be vulnerable to burglars.

This electronic door lock is evidently appropriate for those who feel they have had enough of the hassles of dealing with losing or forgetting keys, duplicating keys, or even changing door locks. The electronic door lock definitely eliminates the need for conventional keys or key cards and is indeed the perfect solution to many homes, businesses, and hotels that lack a reliable and easy to use electronic security door lock.

References

[1] Help for the 555 Timer Chip,

http://www2.ebtech.net/~pais/555_Timer_Help.html.

[2] MM74C922, MM74C923 16-Key Encoder, General Manual, 1999.

[3] Motorola 68HC12 Micro Controller Reference Manual,

http://ebus.mot-sps.com/mcu/documentation/pdf/cpu12rmr1.pdf.

[4] R. L. Theraja and A.K. Theraja, a Textbook of Electrical Technology

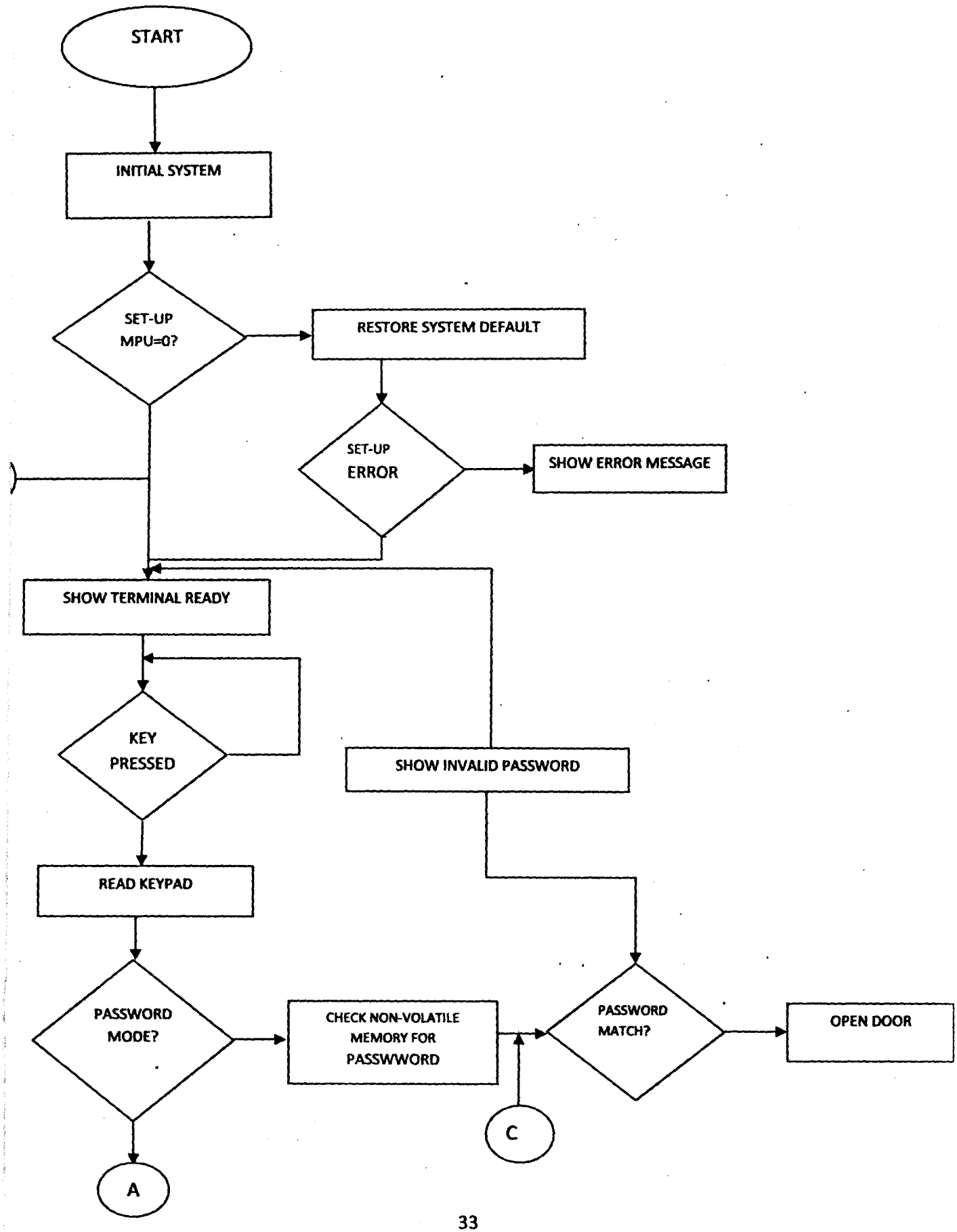
[5] Electronic Door Lock By; Joseph Petrovic, Robert Sanford, and TA: Jon Benson (Dec, 1999)

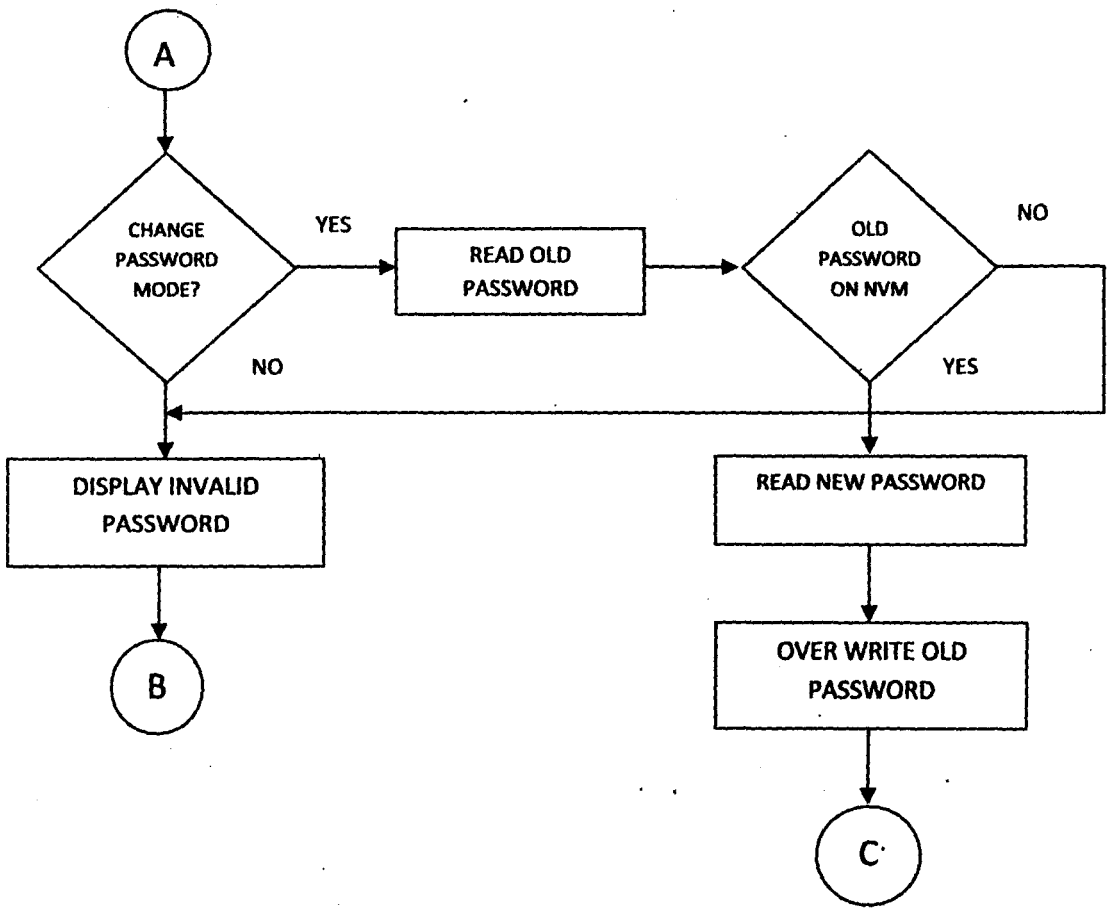
[6] www.rapidelectronics.com/schmitttriggeraction

[7] www.uiciedu/projects/org.

[8] www.8051projects.org

APPENDIX


START
INITIAL SYSTEM
SET-UP MPU=0?
RESTORE SYSTEM DEFAULT
SET-UP ERROR
SHOW ERROR MESSAGE
SHOW TERMINAL READY
KEY PRESSED
SHOW INVALID PASSWORD
READ KEYPAD
PASSWORD MODE?
CHECK NON-VOLATILE MEMORY FOR PASSWWORD
PASSWORD MATCH?
OPEN DOOR
C
A



FLOW CHART OF THE CONDITIONAL ACCESS DIGITAL DOOR LOCK

SOURCE CODE FOR THE AT8951

```
INCLUDE 89c51.mc
;*************************************
lcd_rs BIT p2.7
lcd_port EQU p0
lcd_en BIT p2.6
MODE_sW BIT P1.1
;*************************************
STACK EQU 60
;*************************************
clock BIT p2.1
data_out BIT p2.0
;*************************************
NVM_BUFFER DATA 8
KEYPAD_BUFFER DATA 13
ADDRESS DATA 18
DATA_READ DATA 19
DATA_2_wRITE DATA 20
PASSWORD_RETRY DATA 21
KEY_CODE DATA 22
TEMP DATA 23
TEMP_X DATA 24
TEMP_Y DATA 25
ACCOUNT_NO DATA 26
;*************************************************
prog_mode_header_byte EQU 10
prog_mode_exit_byte EQU 11
;*****************************
error BIT 127
TIMEOUT BIT 126
password_valid BIT 125
SETUP_ERROR BIT 124
IO_ERROR BIT 123
;*****************************************
key_in BIT p1.0
relay_dx BIT p2.4
;*************************************
row_1 BIT p3.3
row_2 BIT p3.2
row_3 BIT p3.1
row_4 BIT p3.0
col_1 BIT p3.7
col_2 BIT p3.6
col_3 BIT p3.5
col_4 BIT p3.4
;*********************************

NVM_ADDRESS EQU 10100000B
read_flag EQU 01h
write_flag EQU 00h

;*************************************************
delete_flag EQU 15
enter_flag EQU 14
;*************************************************
PIN_CODE_0 EQU 1
PIN_CODE_1 EQU 2
PIN_CODE_2 EQU 3
```

```
PIN_CODE_3 EQU 4
;************************************************
LOCK_MASK EQU 0FFH
UNLOCK_MASK EQU 00H
;************************************************
PASSWORD_LENGHT EQU 5

;************************************************
                org 0000h
;************************************************
START:          CLR EA
                MOV sp, #stack
                CALL init_system
                JB mode_Sw, maINLOOP
                ACALL init_NVM
                JNB SETUP_ERROR,MAINLOOP
                CALL SHOW_SETUP_eRROR
                SJMP $
;*********************************************************************

mainloop:       MOV password_Retry,#3
                CALL terminal_ready
                JB key_in,$
re_Read:        ACALL get_user_password
                JNB error, skip_0
                call invalid_password_handler
                JMP dec_retry

skip_0:         ACALL get_entry_mode
                JB TIMEOUT, MAINLOOP
                JB io_error, nvm_error
;*********************************************************************

                JB password_valid, SKIP_5
                ACALL invalid_password_handler
dec_retry:      DJNZ password_retry,re_read
                JB TIMEOUT, MAINLOOP
                ACALL invalid_password_handler
                CALL LONG_dELAY
                JMP MAINLOOP
;*********************************************************************

SKIP_5:         ACALL access_granted_Handler
                ACALL GRANT_ACCESS_HANDLER
                JMP mainloop
;************************************************************
WRITE_2_LINE:   ACALL WRITE_sTRING
write_1_line:   ACALL WRITE_sTRING
                ACALL LONG_DELAY
                RET
;************************************************************

SHOW_SETUP_ERROR: MOV DPTR,#SETUP_ERROR_MSG
                ACALL WRITE_1_LINE
                ACALL LONG_DELAY
                RET
;*********************************************************************

SHOW_io_error:      MOV DPTR,#NVM_io_Error_message
                    JMP write_1_line
```

```
;*****************************************************************
terminal_ready:          MOV DPTR,#terminal_ready_message
                         JMP write_1_line

keypad_timeout_handler: MOV DPTR,#keypad_TIMEOUT_message
                         JMP write_1_line

access_granted_handler: MOV DPTR,#access_granted_message
                         ACALL write_String
                         ACALL long_delay
                         RET

;*****************************************************************

grant_access_handler:    MOV DPTR,#opening_msg
                         ACALL write_String
                         CLR RELAY_DX
                         ACALL LONG_DELAY
                         ACALL LONG_DELAY
                         ACALL LONG_DELAY
                         ACALL LONG_dELAY
                         ACALL long_delay
                         SETB RELAY_DX
                         RET
opening_msg:       DB 01h, 80h, "   WELCOME...     ", 0
;*************************************************************

invalid_media_handler:          MOV DPTR,#invalid_media_message
                                JMP write_2_line

invalid_password_handler:       MOV DPTR,#invalid_password_message
                                call write_String
                                call long_delay
                                JMP mainloop
;*************************************************************
nvm_error:               MOV DPTR,#nvm_io_error_message
                         call write_String
                         call long_delay
                         JMP mainloop

;*********************************************
SETUP_ERROR_MSG:  DB 01H, 80H, "  SETUP ERROR   ",0,0C0H, 0C0H, 0
terminal_ready_message:DB 01H,80h," TERMINAL READY ",0
access_granted_message:DB 01h,80h," ACCESS GRANTED  ",0
INVALID_MEDIA_MESSAGE: DB 01H, 80H, "  UNRECOGNIZED  ", 0, 0C0H, 0C0H, "
MEDIA      ", 0
new_password_message:DB 01H,80h,"NEW PASSWORD: ",0,0C0H,0C0H,0
old_password_message:DB 01H,80h,"OLD PASSWORD: ",0,0C0H,0C0H,0
invalid_password_message: DB 01H,80h,"INVALID PASSWORD",0
access_Denied_message: DB 01H,80h," ACCESS  DENIED ",0
enter_password_message: DB 01H,80h,"ENTER PASSWORD: ",0,0C0H,0C0H,0
NVM_io_error_message: DB 01h, 80h," NVM I/O ERROR  ",0
keypad_TIMEOUT_message: DB 01h,80h," TIME-OUT ERROR ",0

;*************************************************************
write_password:          MOV R0,#KEYPAD_BUFFER
                         MOV A, ADDRESS
                         CLR C
                         SUBB A,#PASSWORD_LENGHT
                         MOV address,A
```

```
write_passworD_loop:    MOV A, @R0
                        MOV data_2_write, A
                        CALL writE
                        JC WRITE_IO_eRROR
                        CALL read
                        JC write_io_Error
                        MOV A, data_Read
                        XRL A, @R0
                        JNZ write_io_error
                        INC R0
                        INC address
                        CJNE R0,#KEYPAD_BUFFER+PASSWORD_LENGHT,
write_password_loop
                        CLR C
write_io_error:         RET

;*****************************************************************
filter_user_password:   MOV R0,#keypad_buffer
                        CLR error
loop:                   MOV A,@R0
                        CJNE A,#9, chk_2
skip_back_filter_user_password:INC R0
                        CJNE R0,#keypad_Buffer+PASSWORD_LENGHT,loop
                        CLR C
                        RET

chk_2:                  JC skip_Back_filter_user_password
                        SETB error
                        CLR C
                        RET
;*****************************************************************
get_entry_mode:         ACALL parse_user_password
                        JNB error, CHK_PROGRAM
exit_chk_pgm:           CLR passworD_Valid
                        RET
;*****************************************************************************
chk_program:            MOV R0,#KEYPAD_BUFFER
                        MOV A,@R0
                        CJNE A,#prog_mode_header_byte, CHK_PASSWORD
                        MOV A,#1
                        INC R0
chk_pgm_loop:           XRL A, @R0
                        JNZ exit_chk_pgm
                        MOV A,@R0
                        INC R0
                        INC A
                        CJNE R0,#KEYPAD_BUFFER+4,chk_pgm_loop
                        MOV A,#prog_mode_exit_byte
                        XRL A,@R0
                        JNZ exit_chk_pgm

pgm_mode_proper:        ACALL get_old_password
                        JB error, exit_chk_pgm
                        ACALL filter_user_password
                        JB error, exit_chk_pgm
                        ACALL find_Account
                        JNB password_Valid, exit_chk_pgm
                        ACALL get_new_password
                        JB error, exit_chk_pgm
                        ACALL filter_user_password
```

```
                        JB error, exit_chk_pgm
                        ACALL write_password
                        JC exit_here_3
                        CLR C
                        SETB password_valid
exit_here_3:            RET
;*****************************************************************

chk_password:           CLR password_valid
                        ACALL filter_user_password
                        CLR C
                        JB error, exit_chk_password
                        ACALL find_Account
exit_chk_password:      RET
;*************************************************************
GET_USER_PASSWORD:              MOV DPTR,#ENTER_PASSWORD_MSG
                                CALL WRITE_sTRING
                                CALL WRITE_sTRING
                                CALL READ_KEYPAD
                                RET
ENTER_PASSWORD_MSG:     DB 01H, 80H, "ENTER PASSWORD: ", 0, 0C0H, 0C0H,0
;*****************************************************************
get_OLD_passworD:       MOV DPTR,#OLD_password_message
we_dig:                         CALL write_string
                                CALL write_String
                                CALL read_keypad
                                RET
;*****************************************************
parse_user_password:    CLR error
                        MOV A,R0
                        CLR C
                        SUBB A,#KEYPAD_BUFFER
                        JZ EXIT_parse_user_password_1
                        XRL A,#PASSWORD_LENGHT-1
                        JZ exit_parse_user_password_2
exit_parse_user_password_1: SETB error
EXIT_PARSE_USER_PASSWORD_2:RET
;*****************************************************
get_new_password:MOV DPTR,#new_passworD_message
                                SJMP we_dig
;*************************************************************************
*

read_keypad:            CLR error
                        CLR TIMEOUT
                        MOV R0,#KEYPAD_BUFFER


t_out_3:                MOV R7,#10          ; 40 HERE BEFORE
t_out_2:                MOV R6,#0
t_out_1:                MOV R5,#0
wait_key1:              JNB key_in, wait_key
                        DJNZ R5, wait_key1
                        DJNZ R6, t_out_1
                        DJNZ R7, t_out_2
                        SETB TIMEOUT
                        RET

reload_r7:              MOV R7,#7
reload_r6:              MOV R6,#0
```

```
reload_r5:              MOV R5,#0
wait_key:               JNB key_in, go_get_key
                        DJNZ R5, wait_key
                        DJNZ R6, reload_r5
                        DJNZ R7, reload_r6
                        SETB error
                        RET

go_get_key:             CALL read_key_code
                        MOV temp, A
chk_delete:             XRL A, #delete_Flag
                        JZ delete_pressed
                        MOV @R0, TEMP
                        INC R0
                        MOV A, #"*"
                        'MOV A, TEMP
                        'CALL CONVERT_2_ASCII
                        CALL write_lcd_data
                        MOV R2,#31
again_x:                CALL dly_2ms
                        CALL dly_2ms
                        CALL dly_2ms
                        CALL dly_2ms
                        CALL DLY_2MS
                        DJNZ R2, again_X
                        CJNE R0,#KEYPAD_BUFFER+PASSWORD_LENGHT,
reload_r7
                        DEC R0
                        CLR error
                        RET

delete_pressed:         MOV DPTR,#enter_password_message
                        CALL write_string
                        CALL write_String
                        JNB key_in,$
                        SJMP read_keypad
;*****************************************************************

read_key_code:    MOV key_code,#10h
                  JB row_1, sCAN1
                  MOV key_code,#0
SCAN1:            JB row_2, sCAN2
                  MOV key_code,#4
sCAN2:            JB row_3, sCAN3
                  MOV key_code,#8
sCAN3:            JB row_4, flip_bits
                  MOV key_code,#12
flip_bits:        MOV p3,#11110000b
                  CALL settle_Delay
                  JB col_1, sCAN5
                  MOV temp,#0
sCAN5:            JB col_2, sCAN6
                  MOV temp,#1
sCAN6:            JB col_3,sCAN7
                  MOV temp,#2
sCAN7:            JB col_4,compute_key
                  MOV temp,#3
compute_key:      MOV p3,#00001111b
                  MOV A, key_code
                  XRL A,#10h
```

```
                JZ no_key
                MOV A, key_code
                ADD A,temp
                MOV DPTR,#xlate_table
                MOVC A,@a+dptr
no_key:         RET

xlate_table:    DB 1,2,3,15,4,5,6,14,7,8,9,13,10,0,11,12
;*****************************************************************
settle_delay:   MOV R7,#0
                DJNZ R7,$
                MOV R7,#0
                DJNZ R7,$
                RET
;*****************************************************************
init_lcd:       MOV A, #38h
                CALL write_lcd_cmd
                MOV R2,#10
delay_Again:    CALL dly_2ms
                DJNZ R2, delay_again
                MOV A, #38h
                CALL write_lcd_cmd
                CALL dly_100us
                MOV A,#38h
                CALL write_lcd_cmd
                CALL dly_100us
                MOV A,#38h
                CALL write_lcd_cmd
                CALL dly_100us
                MOV A,#0fh
                CALL write_lcd_cmd
                CALL dly_100us
                MOV A,#01h
                CALL write_lcd_cmd
                CALL dly_2ms
                MOV A,#06h
                CALL write_lcd_cmd
                CALL dly_2ms
                RET
 ;*************************************************************
 init_system:           CALL LONG_DELAY
                        CLR lcd_rs
                        SETB data_out
                        SETB RELAY_DX
                        MOV p3,#00001111b
                        SETB key_in
                        CALL INIT_LCD
                        CALL SHOW_wELCOME
                        CALL SHOW_WAIT
                        CALL SHOW_ID
                        RET
;*************************************************************
write_String:           CLR A
                        MOVC A,@a+dptr
                        CALL write_lcd_cmd
                        CALL dly_2ms
                        CLR A
                        INC DPTR
                        MOVC A,@a+dptr
                        CALL write_lcd_cmd
```

```
                        CALL dly_2ms
write_String_lp:        CLR A
                        INC DPTR
                        MOVC A,@a+dptr
                        JZ exit_Write_string
                        CALL write_lcd_Data
                        CALL dly_100us
                        SJMP write_String_lp
exit_write_string:      INC DPTR
                        RET
;************************************************************
Dly_2ms:            MOV R7,#0
dly_2ms_lp:         NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    NOP
                    DJNZ R7,dly_2ms_lp
                    RET
;************************************************************
dly_100us:  MOV R7,#100
            DJNZ R7,$
            RET
;************************************************************

long_Delay:         MOV R6,#200
long_delay_lp:      CALL dly_2ms
                    DJNZ R6, long_delay_lp
                    RET
;************************************************************
write_lcd_cmd:      MOV lcd_port, A
                    CLR lcd_rs
                    CLR lcd_en
                    SETB lcd_en
                    CLR lcd_En
                    CALL dly_2ms
                    RET
;************************************************************
write_lcd_data:     MOV lcd_port, A
                    SETB lcd_rs
                    CLR lcd_en
                    SETB lcd_en
                    CLR lcd_en
                    CALL dly_100us
                    RET
;************************************************************
convert_2_Ascii:    CJNE A, #9, chk_10
go_1:               ADD A, #30h
                    SJMP exit_1
chk_10:             JC go_1
                    ADD A, #37h
exit_1:             RET
;************************************************************
```

```
write:                  CLR io_error
                        CALL i2c_Start
                        MOV A, #NVM_Address
                        ORL A, #write_flag
                        CALL write_byte
                        JC write_Abort
                        MOV A, address
                        CALL write_byte
                        JC write_Abort
                        MOV A, data_2_Write
                        CALL write_byte
                        JC write_Abort
                        CLR C
                        CALL i2c_Stop
                        call write_time_out
                        RET

write_abort:            SETB io_Error
                        CALL i2c_Stop
                        CALL write_time_out
                        ret
;*************************************************************

read:                   CLR io_error
                        CALL i2c_Start
                        MOV A, #NVM_Address
                        ORL A, #write_flag
                        CALL write_byte
                        JC read_Abort
                        MOV A, address
                        CALL write_byte
                        JC read_abort
                        CALL i2c_Start
                        MOV A, #NVM_Address
                        ORL A, #read_flag
                        CALL write_byte
                        JC read_Abort
                        CALL read_byte
                        MOV data_Read, A
                        CALL no_Ack
                        CLR C
                        CALL I2C_STOP
                        RET


read_Abort:             SETB io_Error
                        CALL i2c_Stop
                        RET
;*************************************************************


i2c_start:              SETB data_out
                        SETB clock
                        CALL dly_7us
                        CLR data_out
                        LCALL dly_5us
                        CLR clock
                        CALL dly_7us
                        CLR C
                        RET
```

```
;*************************************************************

i2c_stop:               CLR data_out
                        CALL dly_5us
                        SETB clock
                        CALL dly_7us
                        SETB data_out
                        RET
;*************************************************************

write_byte:             MOV R7,#8
write_loop:             RLC A
            MOV data_out, C
            NOP
            NOP
            SETB clock
            CALL dly_7us
            CLR clock
            CALL dly_7us
            DJNZ R7, write_loop
            SETB data_out
            NOP
            NOP
            NOP
            SETB clock
            NOP
            NOP
            MOV C, data_out
            CLR clock
            RET

;*****************************************************
read_byte:  MOV R7,#8
            SETB data_out
read_loop:  NOP
            SETB clock
            CALL dly_7us
            SETB data_out
            NOP
            NOP
            MOV C, data_out
            RLC A
            NOP
            NOP
            CLR clock
            CALL dly_5us
            DJNZ R7, read_loop
            MOV data_Read, A
            RET
;*****************************************************
write_time_out:   MOV R7,#30
reload:           MOV R6,#250
                  DJNZ R6,$
                  DJNZ R7, reload
                  RET
;*****************************************************
no_Ack:           SETB data_out
                  NOP
```

```
                        CALL LONG_DELAY
                        CALL LONG_DELAY
                        CALL LONG_DELAY
                        CALL LONG_DELAY
                        RET
;*************************************************************************

show_id:                MOV DPTR,#id_message
                        CALL WRITE_STRING
                        CALL WRITE_sTRING
                        CALL LONG_DELAY
                        CALL LONG_dELAY
                        CALL WRITE_sTRING
                        CALL WRITE_sTRING
                        CALL LONG_DELAY
                        CALL LONG_DELAY
                        CALL WRITE_sTRING
                        CALL WRITE_STRING
                        CALL LONG_DELAY
                        CALL LONG_dELAY
                        RET
```