DESIGN AND CONSTRUCTION OF A THREE DIGIT ALPHA-NUMERIC DISPLAY ELECTRONIC LEARNING KIT

BY

JATTO MOHAMMED

2004/18829EE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, FEDERAL UNIVERSITY OF TECHNOLOGY MINNA, NIGER STATE, NIGERIA.

DECEMBER, 2009.

DESIGN AND CONSTRUCTION OF A THREE DIGIT ALPHA-NUMERIC DISPLAY ELECTRONIC LEARNING KIT

BY

JATTO MOHAMMED

2004/18829EE

A THESIS SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELOR OF ENGINEERING (B.ENG) SCHOOL OF ENGINEERING AND ENGINEERING TECHNOLOGY, FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA. NIGER STATE

DECEMBER, 2009.

DECLARATION

I JATTO MOHAMMED 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.

JATTO MOHAMMED

....

(NAME OF STUDENT)

18/12/09

....

(SIGNATURE AND DATE)

MR. PAUL O. ABRAHAM ATTAH

....

(NAME OF SUPERVISOR)

18/12/09

....

(SIGNATURE AND DATE)

ENGR. DR. Y.A. ADEDIRAN

....

(NAME OF H.O.D)

(May 6, 2010)

....

(SIGNATURE AND DATE)

....

(NAME OF EXTERNAL EXAMINER)

....

(SIGNATURE AND DATE)

ACKNOWLEDGEMENT

My profound gratitude goes to the Almighty God for seeing me through the completion of this work. In a special way, I wish to recognize the contribution of Mr. Paul O.A. Attah, who in no small measure guided me during the course of this project work. My sincere gratitude goes to my parents Mr. and Mrs. Shittu Usman, Engr. Philip Saba N.(Dmitry), Mr. Stephen Oyewobi, and all that have imparted my life positively.

I also want to appreciate my brothers and sisters for their immerse support namely Muriana, Ja'afaru, Sherifat, Sumbi and Aisha Usman Jatto. I want to appreciate the entire effort of my friends namely Abdul, Emmanuel, Anne-Marie, Kingsley, Rufai, Jacob to mention a few. I love you all.

May God continue to bless you all, in Jesus name. Amen.

ABSTRACT

The project undertaking emphasizes the usefulness of utilization of embedded computer technology for the learning process of children and adults. It also establishes a form of self-motivating educational involvement. A computerized tool that teaches alphabets, numbers, mathematical symbols etc is presented as the end result of the design undertaking. The enhancement tool was design out of the need to make learning more interesting an interactive to the target audience that is children. It demonstrates the use of computer hardware and software as agents of computer –assisted learning

TABLE OF CONTENTS

CHAPTER ONE

GENERAL INTRODUCTION

1.1 INTRODUCTION

Nowadays, children find it easier to perform better in school when they are already familiar with the basic and fundamental aspect of their work which serves as the foundation of every learning.

It is well known that learning starts with recognition and pronunciation of alphabets, numbers and various shapes. Without the knowledge of alphabets, no word can be constructed since the knowledge of alphabets is required before thinking of constructing any word since every word is made up of alphabets.

Also, numbers and shapes are very important because just like words are formed everyday, we do one counting or the other everyday and numbers together, multiply them with each other, subtract from each other as well as divide with each other. In this case, without the knowledge of the objects you are dealing with, no addition, subtraction, division, multiplication or any other mathematical operation can be performed. The knowledge of this fundamental requirements of learning before a child is enrolled in school often help them to perform well after enrolment since they are already familiar with this things from home and since in every school, every teacher starts with pre-scholars by teaching them how to name and identify numbers, alphabets and various shapes.

Often, parents get too busy to tutor their children. The children are left with what they learn in school only without revision and most children need continuous revision to get things stock to their brain.

It's on this note, I came up with the design of a three digits and alphabetical display electronic learning kit which can display numbers, alphabets, and word spelling [basically three letter].

Previously, notebooks and slates are been used to teach children at home. This requires full attention by the parent or any body acting as the tutor at all time. Inspection is required since books could be torn due to its texture and work written on slates could be wiped out by the child who is just a pre-scholar and does not know how to handle things with care.

The above can be said to be inadequate since the aim might be achieved without proper attention. In this design the display shall consist of three [3] separate 5x7 LED array. The system shall consist of 15 columns and 7 rows while software multiplexing is used to provide a display of the required symbol.

1.2 OBJECTIVES OF THE PROJECT

The electronic learning aid (kit) was design to meet the following objectives:

i. To help pre-scholars and the children derive greater sense of involvement in the learning process.
ii. To develop an affordable and cost-effective replacement for the traditional wooden slate.

iii. To demonstrate the utilization of a ready to-go replacement for non-educational toys.

iv. Provide an electronic means of teaching simple three-lettered words, shapes, symbols, alphabets etc.

v. To provide a means of storing user –defined characters /words besides the fixed vocabulary set in the controller memory at assembly time.

vi. This design will help parents to impact the knowledge of the fundamental requirements of learning to their children before they are enrolled in school and even act as a form of revision to the child after school hours.

vii. This design is to provide an electronic replacement for the slates and notebook teaching system for pre-scholars and kindergartens and also to make teaching easy for the tutors since is made easy for the children.

1.3 ADVANTAGES OF THE DESIGN

This kit shall incorporate a non-volatile memory of the symbols intended for display. These symbols shall be stored in the form of bitmaps and for connectivity with the users; a visual display shall be employed. With these at the press of a particular button, which shall be provided, the desired numbers, alphabets or words are been displayed. Another advantage of this design is that it can display three (3) digits and words i.e. 000-999 and word like car, boy, cat, A-Z, a-z etc. compare to others which might display 0-9 and single alphabet.

1.4 DISADVANTAGES OF DOT MATRIX OVER SEVEN SEGMENT DISPLAY.

Since the kit/pack is required to display any desired symbol without restriction, the dot matrix tends itself to applications where standard display system [seven segment displays] do not fit in.

An obvious example is the display of alphabetical characters. A seven segment display will not display almost eight characters which are not in the desired format. A dot matrix on the other hand allows full customization of the display format. For example a seven segment can not display "B" only "b" which is a small letter, can be displayed.

1.5 SCOPE OF THE PROJECT

The project aims at designing and constructing a device that will visually teach the elemental components of pre-scholar learning framework.

A reduced complexity microprocessor [microcontroller] was wired to a 5 x7 alpha metric dot matrix display to form a unit that displays several characters at the touch of buttons designed for the system control. The target user can select any mode [numeric, alphabet, symbols etc] desired and master such since the system is software driven, it is easily recognizable by simply downloading form wave updates via a suitable programming pod to redefine the functionality of the entire system.

1.6 METHODOLOGY

The realization of the electronic teaching aid/kit was effected to the system layout with the block diagram in fig.1

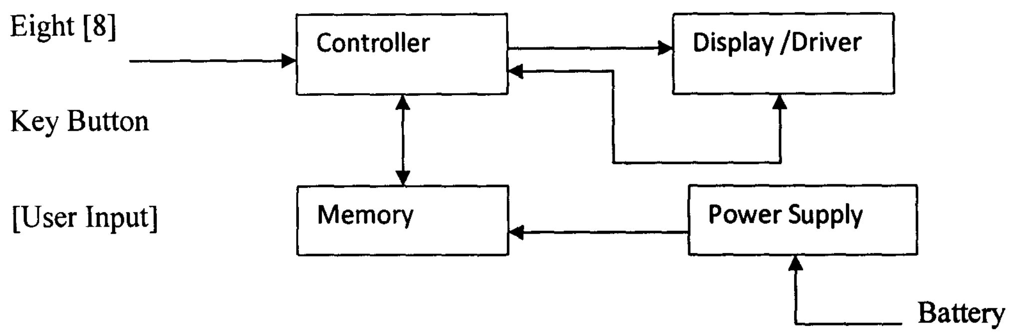


Fig. 1 Block diagram of three digit alpha – numeric display electronic learning unit.

Portable system was intended and hence a light weigh power supply option was needed. A 9V DC battery cell readily achieved this:

A system controller loaded with the firm wave affected the task of responding to user key process and displaying requested symbols on the dot matrix display.

A dot matrix display constituted the visual output device. It was chosen since it allows the display of any desired symbols; this is done by appropriate bit patterning.

1.7 LIMITATIONS

The design is only visually oriented. No audio feedback was incorporated to introduce the spoken equivalent of the display symbol.

This is a setback as it demands that a teacher be on hand to fill that need. It is also a three [3] digit unit, and hence can only teach alphabets, spellings [three [3] lettered words] and numbers. It cannot, for example teach counting beyond 999 and words beyond three letters.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

Computer assisted learning, CAL, is not a new phenomenon. This field of education has been growing since the introduction of computers into the learning environment. Starting with the humble digital textbook, CAL has moved into the way of creating a learning environment, where the participant has the ability and necessary tools to produce their own learning paths and outcomes.

2.2 HISTORY OF COMPUTER ASSISTED LEARNING [CAL]

In [1] CAL is referred to a computer program or file developed specifically for educational purposes. The techniques used throughout the world in a variety of contexts, from pre and primary school aged children to train adults in medicine, law, accounting etc.

However its use has not been so widespread until recently. In 1980s, the first computer assisted learning became available to university students looking for an alternative to the traditional text book. These programs uses only digital files transferred into a CDROM to give students highly portable and accessible learning materials.

According to [2], the popularization of this form of learning and the increase of personal computers led to the development of widely distributed educational

CDROM such as Encarta. As technology continued to grow and with the advent of the internets information on CAL programs become more interactive, reflecting a social need for flexible learning outcomes. There is now a multitude of CAL platforms designed for every market from school based entertainment packages to computer based management learning environment [1].

In retrospect, CALs history begins in the early 60s when the third generation of digital computers were built and introduced. These systems were cheaper and more reliable than the earlier models. So digital computers became typical facilities in universities and research entries. Consequently, researchers started to find raw fields of applications for the computers and CAL was one of those certainties of the beginning, like other technological productions, CAL systems which are a combination of computer hardware, added special purpose peripheral, and CAL software has only scientific and academic applications and is experimental. At that time [3], before other specialist psychologist used the computer as an ideal tool for converging programmed instruction, this branch of CAL was called computer.

In 1969, [4] listed about 20 various CAL systems that had been developed in universities and research centers. Obviously there were the first steps and were concentrated on academic goals and had a wise range from a computer based science testing system [5] to one of the earliest and remote educating systems that was developed in Harvard University [6]. During the 70s CAL systems were developed but the development rate was still low, because of technical and also economical limitations especially concerning hardware that was still expensive, massive and mainly without adequate sound and graphic facilities.

In the UK from 1973, The National Development Program in computer assisted learning was founded. This research center had a 5 –year plan and a considerable two million pounds budget for developing CAL systems in the UK.

In the middle 70s CAL system were used in Interventional Computers Limited [ICL] computer for operator's training. The project started in 1963 but became operational in 1971 [8] one of the earliest computer-produced AV materials (or as they were later called 'multi-media') was reported by A.H Francis in 1975. These materials were some parts of long-term project for using computers in the production of educational films.

Around the beginning of the 70s, A + Kinson and Wilson [9] suggested that the main problems of CAL systems were:

i. Low quality of graphic displays (e.g monochrome monitors).
ii. Handling problems of random access audio tapes.
iii. Cost of terminal per hour, it was higher than an adequate level for general use.

During the 80's and 90's these and other CAL problems were eliminated by the Digital Revolution. Digital technology development, upgraded hardware facilities and performances and lower prices solved, these problems of low quality, mono chrome graphic displays. Nowadays computing and processing speed of a typical computer are tens of times higher than the most advanced computer of the 70's. Compact dices [CDS and DVDs] and real sound video peripherals brought multi-media facilities in the PC world. Therefore as time went by, CAL system designers could develop their ideas and implement them. The results are current CAL system.

2.3 CURRENT COMPUTER ASSISTED (CAL) SYSTEMS

Currently used CAL systems have the following features.

i. Easy to access-nowadays everybody can get an educational package, which is normally a CD or uses educational web sites. The total cost of PC and an educational software are cheap enough for most average wage earners and more importantly, to educational centre all around the world. The world can bear.

ii. Quality –sound and graphics are really in high quality

iii. Storage –a normal CD can store up to about 650million characters

Definitely, CAL systems have made a lot of progress over recent years, although is not the end of the line and systems still have some disadvantages and weaknesses, that must be reduced in future. Besides this, future CAL systems must satisfy more expectations.

2.4 FUTURE DEVELOPMENTS.

CAL system's development in future years will be concentrated on the following areas:

i. Increasing system performance – this include higher speed of processing and stronger software. This is a normal performance to current system and their development policies.

ii. In [10] it was postulated that more distributed systems –nowadays have modern world wide networks, high speed digital and mobile communications create a strong context for design and implementation of more distributed CAL systems. Now teleconferencing and non-centralized classrooms will be usual as postulated by mayadas [10].

iii. Simulation and visual reality –new technology [11] allows us to create 3D near to real environments for simulators systems. These will have important roles especially for blind and rarely blind people education.

iv. Special purpose CAL systems for the disabled –whereas typical teaching methods are suitable for normal learners, each sort of disabled has their special needs, feature and appropriate teaching method requirements. For instance, the teaching process of people with severe learning difficulties is mainly based on repeating some simple materials again and again, because the understanding speed of thus category is unfortunately low. A multi-media work stations, could hold the teacher by repeating the words, and decreases teachers duty load sharply.

v. Intelligent CAL systems –these systems [12] shall have many sensitive advantages compare to conventional ones. They are direct results of artificial intelligence methods.

2.5 CHARACTERISTICS OF AN INTELLIGENT COMPUTER ASSISTED LEARNING SYSTEM.

A brief of review of the characteristics of some implemented ICAL systems will be helpful for understanding their features and abilities.

[13] Introduced CALAT, which is an intelligent tutoring system on the World Wide Web [www]. Users via a typical web browser on access CALAT systems server. CALAT is organized by a goal –sub goal scheme and had three types of course wave pages' explanations exercise and simulation.

[14] Have discussed a system for learning Japanese sign language with three dimensional placements of hand motions. This system had been designed for translations from Japanese to sign language and gets Japanese sentences as inputs and generates and shows sign language motion pictures on the screen. Generalizing of such a system could be very useful in the educating of having impaired peoples.

In recent years many attempts have focused on the distributed systems and because of internet standardization, many designers have selected this as a base platform.

2.6 ADVANTAGES OF COMPUTER ASSISTED LEARNING (CAL)

The use and advantages for this system are endless. For example, teachers are now encouraged to use entertainment computer based learning to introduce students to new or difficult concepts. Time tables or grammar are incorporated into platform games that use positive reinforcement to encourage children more onto new difficulty level.

Children are able to pick which times tables or grammatical problems they wish to tackle in the session. This interactive element [2] is essential to ensure children feel they have some control over their education.

2.7 DISADVANTAGES OF COMPUTER ASSISTED LEARNING (CAL)

Despite it successes, CAL has encountered some problems as well within the business sector, while manager, generally understand the benefits of CAL , employees often do not, in a study completed by John Henry, it was revealed that 85% of workers thought that CAL is too difficult to use and they cannot see the benefits of implementing such a program [16]. While CAL may encounter some negativity from people resistant to change, there is no doubt that this educational tool is extremely valuable for children and adults, there is much to be saved from CAL's interactive and self motivating format for learning .

2.8. INTRODUCTION TO MICROCONTROLLER

Circumstances that we find ourselves in today in the field of microcontrollers has their beginnings in the development of technology of integrated circuits. This development has made it possible to store hundreds of thousands of transistors into one chip. That was a prerequisite for production of microprocessors, and the first computers were made by adding external peripherals such as memory, input-output lines, timers and other. Further increasing of the volume of the package resulted in creation of integrated circuits. These integrated circuits contained both processor and peripherals. That is how the first chip containing a microcomputer, or what could later be known as a microcontroller came about.

2.8.1 COMPONENTS OF A MICROCONTROLLER

Microcontroller is a chip that contains components such as memory, or components for receiving and sending data etc. 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.

2.8.2 BUS

Bus is a way through which data goes from one block to another especially in the case where we wish to add contents of two memory locations and return the result back to memory. It's a connection between memory and CPU. Physically, it represents a group of 8, 16, or more wires.

There are two types of buses: address and data bus. The first one consists of as many lines as the amount of memory we wish to address, and the other one is as wide as data first one serves to transmit address from CPU memory, and the second to connect all blocks inside the microcontroller.

2.8.3 INPUT –OUTPUT

This are known as ports which contains several memory locations whose one end is connected to the data bus, and the other connected with the output lines on the microcontroller which can be seen as pins on the electronic component. It is necessary to choose which port to work with, and then send data, or take it from port when working with ports.

2.8.4 TIMER UNIT

This is a timer block which gives us information about time, duration, protocol etc. the basic unit of the timer is a free-run counter which is in fact a register whose numeric value increments by one in even intervals, so that by taking its value during periods and on the basis of their difference we can determine how much time has elapsed.

2.9. PHYSICAL CONFIGURATION OF THE INTERIOR OF A MICROCONTROLLER

Thin lines which lead from the centre towards the sides of the microcontroller represent wires connecting inner blocks with the pins on the housing of the microcontroller so called bonding lines.

2.9.1 PROGRAM

Beside a microcontroller, there is need for a program that would be executed, and a few more elements which make up interface logic towards the element of regulation. Programming can be done in several languages such as assembly language, c-language etc. In this design, assembly language is used due to fast execution speed.

CHAPTER THREE

3.1 DESIGN AND IMPLEMENTATION

The three (3) digits [redacted] alphabetical display electronic learning kit system comprises the following sub systems:-

1. Power Supply
2. System Controller
3. 5 X 7 Dot Matrix Display
4. Row/Column Drives
5. System Memory

3.2 POWER SUPPLY

Due to the requirement for portability a battery driver realization was adopted. The system was driven from a 9v battery source the system's operational voltage of 5 volts was derived from the 9 volts source via a 7805 5 volts step-down regulator connected to the battery as shown in fig. 3.1

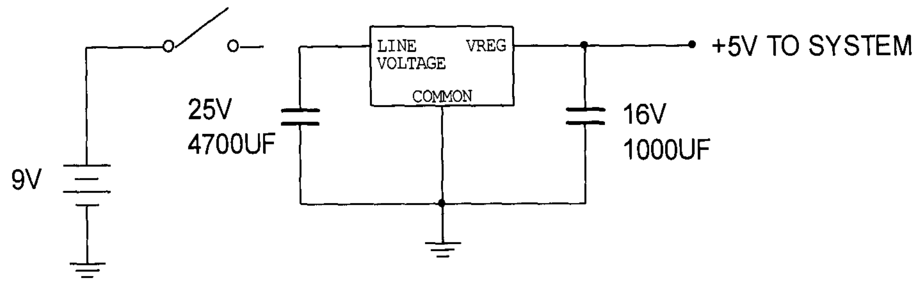


Fig 3.1 System Power Supply.

The 9 v DC is regulated down to 5volt DC by the three-terminal device the 5 volts being the system supply voltage. The battery voltage was buffered by 4700µf capacitance connected across the input supply rails. The 5-volt output was stabilized by 1000µf and fed into the circuit.

3.3 SYSTEM CONTROLLER

An 8-bit microcontroller was incorporated in the design for maximum flexibility and versatility. An AT89C51 microcontroller device was selected due to its simple instruction set and hardware.

The micro controller was coded to perform the following.

1. Respond to the button keys, decode and execute the relevant instructions.
2. Address (read/write) the 24C02 EPROM memory to load user-defined words.
3. Address the 3 –digit 5x 7 dot matrix display to light the relevant led(s).

The controller interfaced with other system components as shown in fig 3.2

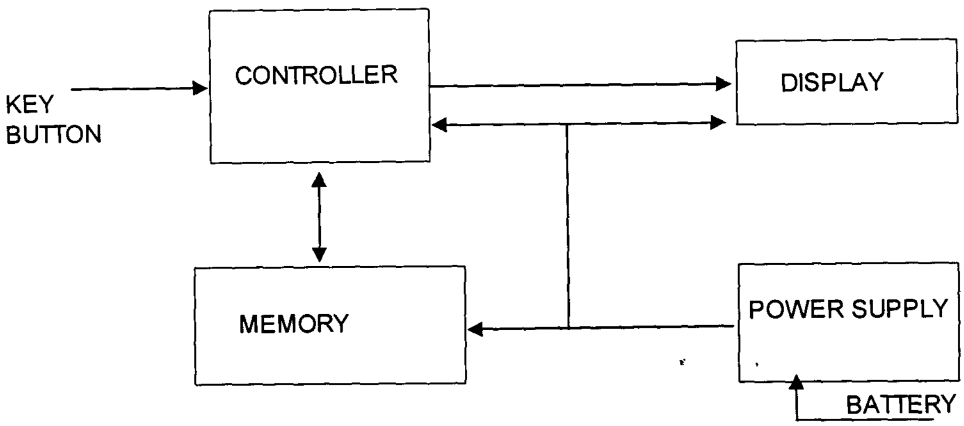


Fig 3.2 Controller –System Interface.

The microcontroller was coded in assembly language for optimal performance since a portion of the control software required bit-barged implementation of the Philips I^2C bus protocol, a software emulation of which required bit-level processing.

The controller also refreshed the display at about 60Hz via a 74LS245 row – driver and two 74LS138 column drivers. An 8-button key input was provided over P3 of the device to respond to user inputted commands. The key buttons were designated the following functions;

Button 1	:	Digit 1
Button 2	:	Digit 2
Button 3	:	Digit 3
Button 4	:	Scan up
Button 5	:	Scan Down
Button 6	:	Mode Select

Button 7 : Store

Button 9 : Delete

Button 1, 2 and 3 enable cycling through the possible symbols depending on the selecting of the mode selection key. In the digit mode, Button 1,2, and 3 can be manipulated to yield a count of 000 to 999.

In the alphabet mode, the keys can be cycled through A-Z, a-z, thus, any desired three –lettered words can be generated e.g. Dad, Mom, Car, Cat, Dog, Egg, Mug, e.t.c. when the non-volatile memory (NVM) mode is selected, button 4 and 5 can be used to cycle through the 85 possible 3-lettered vocabulary stored on the memory device. Button 7 is used for storing the displayed symbols to memory. This only occurs when either the numeral mode or alphabet mode is selected.

Button 8 enables clearing the embedded memory devices at initialization the non-volatile memory NVM holds only one word, "cat".

This was used to prevent a blank display when the mode selection is set to NVM (non-volatile memory).

A look-up table in flash memory held the 5 x7 font bytes. The table was referenced based on the user –selected character in number. The 5-byte matrix code was then transferred to the display for visualization. The system was run on a 12 MHz crystal connected between pins 18 and 19 of the system controller.

3.4 THE LED (LIGHT EMITTING DIODE)

A light emitting diode (LED) is a type of component that uses electricity to emit in coherent narrow spectrum light when electrically based in the forward direction of the p-n junction, the effect is a form of electroluminescence, unlike the fluorescent or incandescent lamp requires a very little voltage to regulate the flow of power through the lamp. In LED lamps, the arrangement of LED in the housing unit or LED array promoted perfect illumination intensity with low power consumption.

3.5 5 x 7 DOT MATRIX

A dot matrix display provides a convenient means of displaying almost any desired symbol since it offers the designer choice of specifying the bitmaps required to generate the displayed symbols. A dot matrix display uses a combination of Logic I and Os to produce any desired character.

A dot matrix display consists of LEDs wired in a grid as shown in Fig 3:3

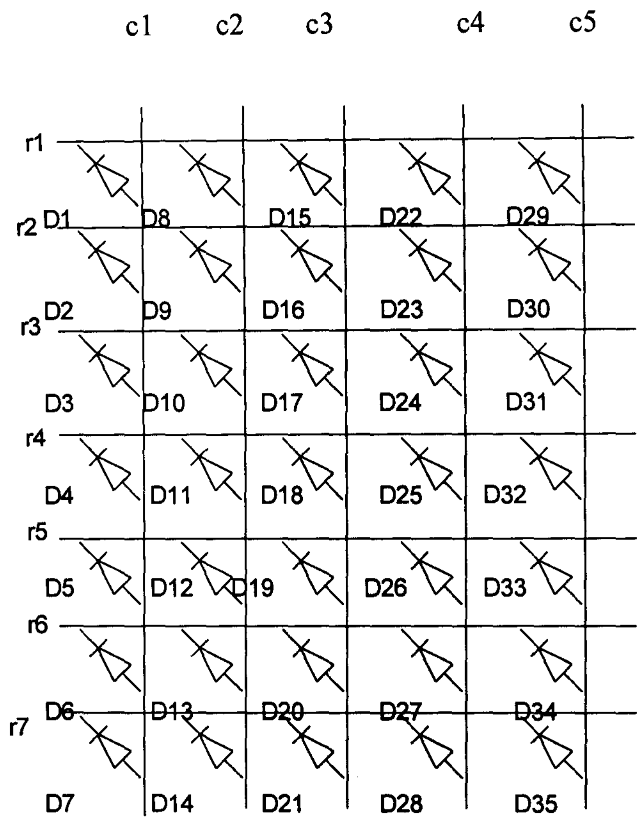


Fig 3.3 5×7 Dot Matrix

3.7 SYSTEM MEMORY

Electrically Erasable Programmable read only memory (EEPROM). For storage of user-defined symbols, a 256 byte EPROM device (24C02) was incorporated in the system. The memory size thus allowed a maximum of 85 words (3 letter each) to be stored on a clip. The 24C02 has a proprietary interface that was emulated in software as the generic 8051 core does not posses an I^2C hardware interface.

The 24C02 device has the following specifications the device was configured at address ooh, and interfaced over P1.0 and P1.1 being SCL and SDA respectively.

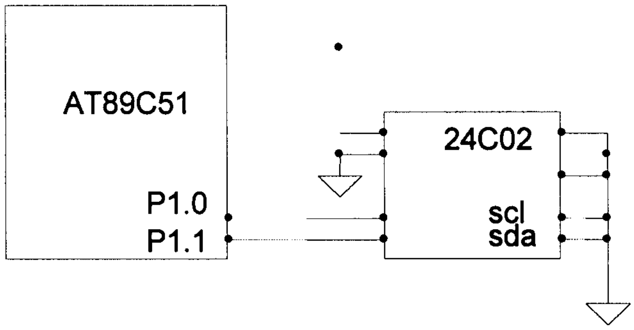


Fig 3.4 Memory Controller Interface

3.8 DISPLAY

A 3-digit dot matrix display was used. An anode row configuration was employed. A 74LS 245 bus driver was used as the anode driver while the current through the LEDs were sink to ground via two 74LS138 decoders connected to the cathodes of the LEDs as indicated below:-

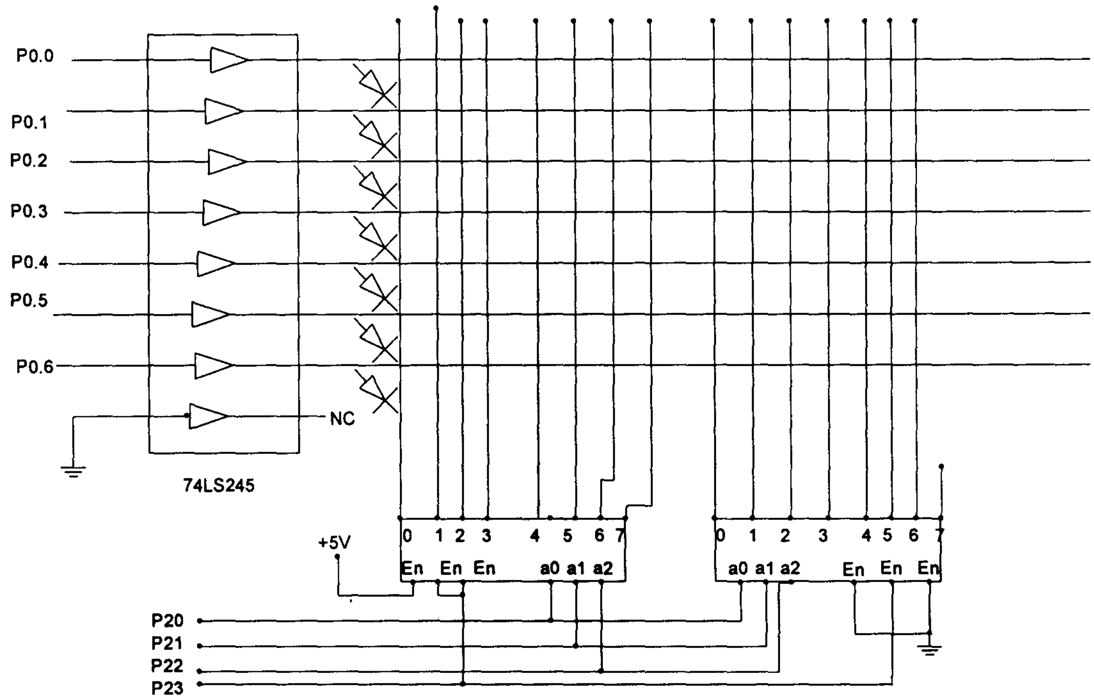


Fig 3.5 Display Driver Circuitry

The displayed data was multiplexed onto port (PO) of the controller, a byte a time. The byte position on the display was then determined by the logic which on the address bus on p2.0 – p2.3

A 3 – bit addressing mode was used successive data bytes are displayed on different display positions as the address bus is incremented from O through 14, giving a possible 15 display positions corresponding to 15 columns making up the three digits. A 74LS245 device was used for driving the rows as shown in fig 3.5. The 74LS245 is a bidirectional bus driver.

The device was configured for unidirectional data transfers by typing the direction control to 1 (+5v), yielding a data transfer only from input A to output B.

Seven of the eight drivers were used for the 5 x 7 display.

3.9 USER MODE

Eight buttons were provided for user interactivity. The buttons allow the user to cycle through the displayed symbols. The first three buttons are the digit mode (i.e. digit 1, digit 2, and digit 3).

The buttons 4 and 5 provides up/down controls, allowing the displayed character. The button six in the mode select button (the Digit mode, Alphabet mode and memory mode) allows the selection of different modes datasets.

The seventh button allows storing of displayed symbols and the eighth button is the delete button which allows deletion of stored words, alphabets or numbers. The incorporated datasets supports:

i. Display of alphabets A-Z, a-z
ii. Display of numbers 000-999
iii. Display of words e.g. Cat, Car, Bag, Two etc. (Basically three (3) lettered word).

The eight buttons were interfaced with the controller over P3.0, P3.1, P3.2, P3.3, P3.4, P3.5, P3.6 and P.3.7 shown in Fig 3.6

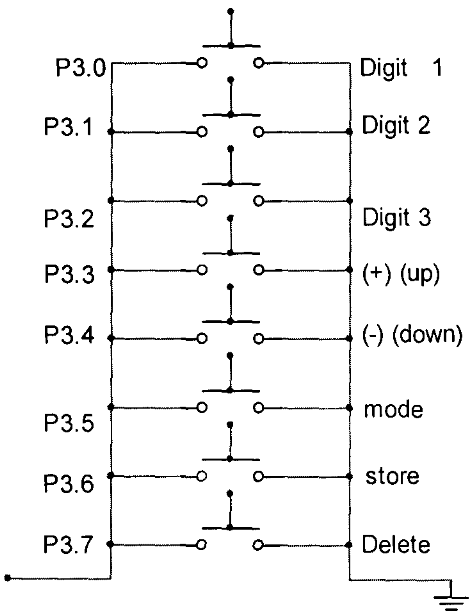


Fig. 3.6 User Input Buttons

CHAPTER FOUR

TESTS, RESULTS AND DISCUSSION.

4.1 TEST

The system was designed according to the projected circuit layout. The various components were soldered in place on a Vero board. The LEDs were arranged in 3 (5*7) and also soldered in place. The AT89C51 controller was docked for code download; an In-system programmer was used.

In system programming, it allows code update with the controller mounted on the target board without necessarily removing ABC. The mode button was pressed to select the required display.

The software was assembled using Batronix prog. Studio 6.09 and a parallel port based ISP Loader application in order to transfer the Intel ascitlex files to the controller. Various modifications were made to control the software during the programming decision.

The Input keys (switches) were tested to ascertain their intended control functionalities.

The eight (8) keys were all observed to function.

It was first tested using a 9v battery. Due to the software driven system realization, minimal problems were encountered since the hardware had already been proven to work perfectly based on the intended system function.

4.2 RESULTS

After the above steps were taken, alphabets were first displayed with the capital letters displayed along side the small letters in order of arrangement of English alphabets that is A,B,C,D.........Z. Also when the mode button for the numerals was pressed, numbers were displayed in order of arrangement of numerals. Also words spelt could be stored by selecting the store button. It stores a total of 85 words and numbers.

4.3 DISCUSSION OF RESULT

On the alphabet displayed, the codes were written into the chip in such away that the first 5x7 LEDs displayed both capital and small letters which is also applicable to the second and third display. In the display of numerals from 000-999, zero is displayed on the left hand side and the middle constantly while the right hand side display starts from 0-9 and since the display of three digits is also required, the decimal codes were combined during debugging. So the first 5x7 display the first digit on the left hand side, the middle 5x7 display the second digit and the third 5x7 display the third digit on the right hand side. An illustration on the display of three digits is shown below.

The above shows the display 101 with 1 displayed on the left hand side, 0 on the middle and 1 displayed on the right hand side. All this can be done via the first three keys/buttons. In the case of alphabets and words, the mode button is selected before spelling takes place.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSIONS

The objective of designing an interactive three digits alphabetical display electronic learning kit was duly met by the construction of a device that visually teach the elemental components of pre-scholars learning framework. The display of alphabets and numbers such as dog, bag, cat, jug and 000-999 was actualizeable, making it easy and attractive for children to appreciate the basis and fundamental aspect of their work which serves as the foundation of every learning.

5.2 RECOMMENDATIONS

For better assistance to the user, an audio interactivity scheme should be considered. This would involve an audio chip programmed with verbal representations of the corresponding displayed symbols. The design can also be improved on in future by using more LED arrays in order to display more than three digits.

Another recommendation is the inclusion of larger customized symbols. This requires the incorporation of a semi-permanent storage. User customization offers a wider symbol database that can be programmed into the unit at design time as it allows the user to update/edit the symbol tables in the field (during usage).

REFERENCES

[1]. Whalley, Benefits of computer aided learning in industrial and commercial training 1998, pp 14- 23.

[2]. Rosecielle, Distributed intelligent tutoring, 2005, pp 33- 37.

[3]. Hooper R. and Toye I., Computer Assisted Learning in the united kingdom, June 1975.

[4]. Frye C. H., CAL Languages, their capabilities and applications, April 1969.

[5]. Hensen D.N., An investigation of computer based science testing, 1969, pp 13-23.

[6]. Lawrence M., Factors in the design of systems for computer, 1969.

[7]. Fishman E. J, Keller L. and Atkinson R. C., Distributed practice in computerized spelling drills, 1969, pp 34 – 39.

[8]. Gillian J., Computer assisted learning in ICL London Council for Educational Technology, 1975.

[9]. Atkinson R. C and Wilson H.A., Computer Assisted Instruction, 1969, pp 15 - 25.

[10]. Mayadas F.A., Asynchronous learning networks, 1997.

[11]. Everinggham B.T., Troscianko T., Thomas B.T., and Easty D., Neural network virtual reality mobility aid, 1998, pp 30 -38.

[12]. Halff H.M., Curriculum and instruction in automatic tutors, 1998.

[13]. Nakabayashi K., Murayama M., Koike Y., Kato Y., Touhei H. and Fukuhara Y. Architecture of an intelligent tutoring system, 1997.

[14]. Terauchi M. and Osaki H., A system for learning Japanese sign language with 3D placement, 1997, pp 18-29.

[15]. Brusilovsky P., Ritter S. and Schwaz E., Distributed intelligent tutoring on the web, 1997.

[16]. Henry J., Resistance to computer based technology, 1994, pp 23 -39.

[17]. Fast and LS TTL data.

[18] AT89C51 available at "http: //www.Atmel.com/89C51.pdf

APPENDIX I

MANUAL OF OPERATION

The unit should be kept away from sources of heat to prevent deformation of the unit casing. If the unit fails to come on, replace the battery and operate the unit again.

To use the unit, select the display mode using the buttons. Pressing the mode button causes the display of either Alphabets number or memory, depending on what the user wants. The user could cycle through 000-999, A-Z, a-z, and spell words like Car, Cat, Mug, Bag etc. When the desired symbol or word has been stored, use the scan up (+) and scan down (-) keys to scan through the available programmed character sets.

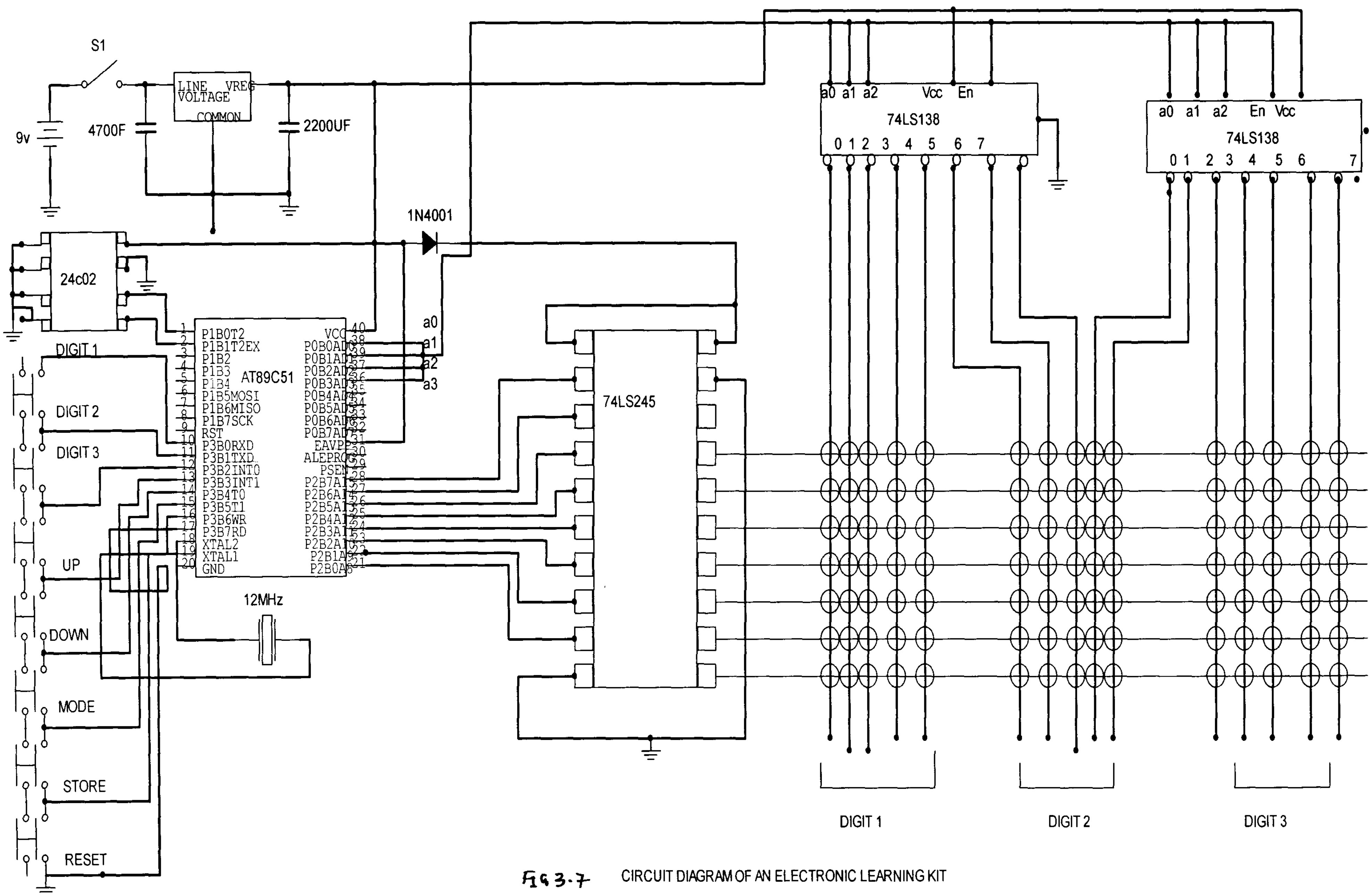
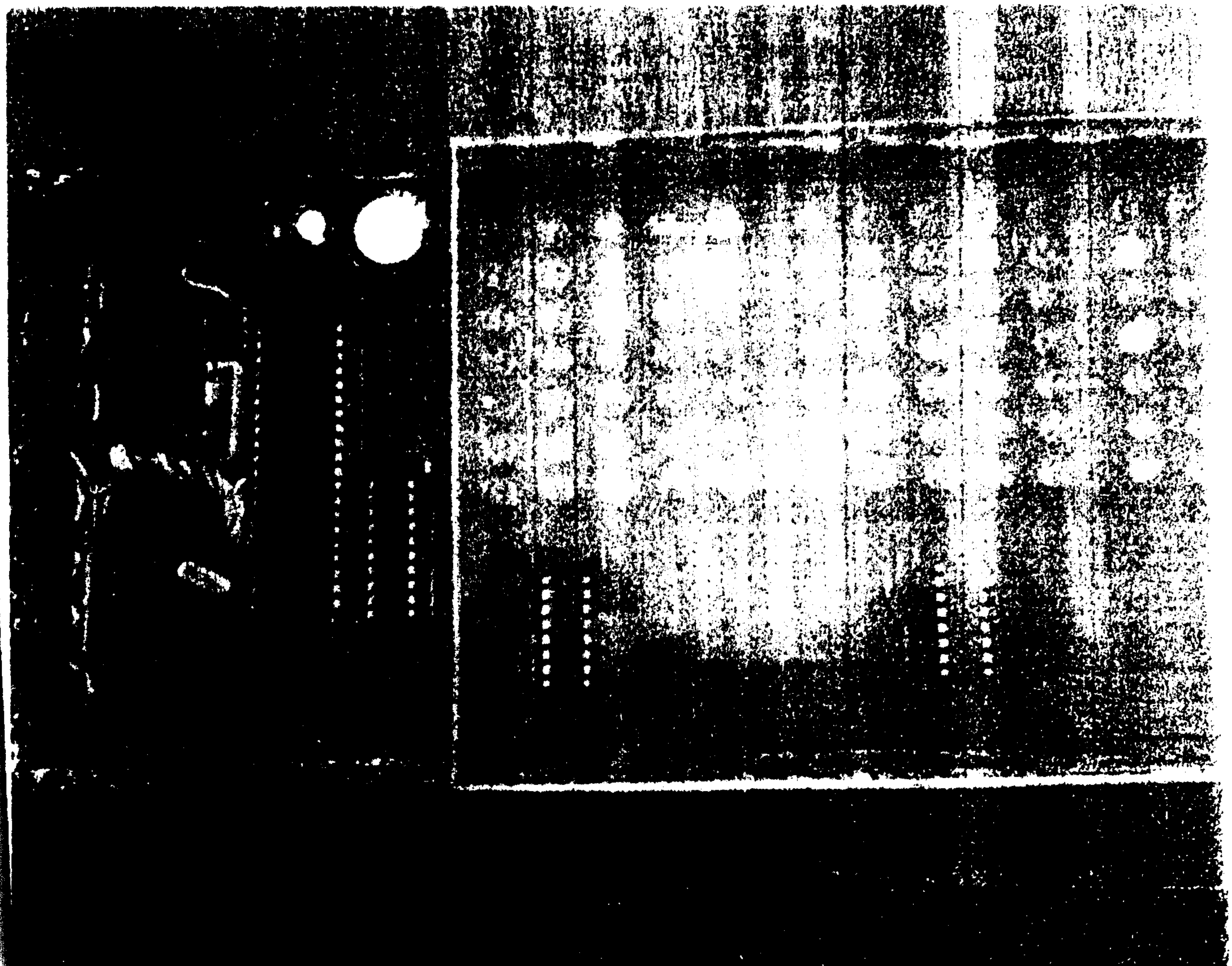


Fig 3.7 CIRCUIT DIAGRAM OF AN ELECTRONIC LEARNING KIT



```
INCLUDE 89c51.mc
;*************************************
stack equ 60
;************************************
ptr1 data 8
ptr2 data 9
ptr3 data 10
temp data 11
temp1 data 12
temp2 data 13
temp3 data 14
address data 15
data_read data 16
data_2_write data 17
mode data 18
digit1 data 19
digit2 data 20
digit3 data 21
max_ptr data 22
nvm_ptr DATA 23
buffer DATA 40
;**********************************
update BIT 00h
;**************************************
slave_address equ 10100000b
read_flag equ 00000001b
write_flag equ 00000000b
sda BIT p1.1
scl bit p1.0
;**************************************
digit1_key bit p3.0
digit2_key bit p3.1
digit3_key bit p3.2
up_key bit p3.3
down_key bit p3.4
mode_key bit p3.5
store_key bit p3.6
delete_key bit p3.7
;**************************************
data_port EQU p2
control_port equ p0
;****************************************
max_digit_ptr equ 10
max_char_ptr equ 52
offset EQU 20h
;****************************************

org 0000h
start_up:       MOV sp,#stack
                ACALL sys_init
;*****************************************
mainloop:       ACALL show_key
                ACALL scan_key
                sjmp mainloop
;******************************************
sys_init:       MOV data_port,#0
                MOV control_port,#0
                MOV mode,#01h
                MOV ptr1,#1
                MOV ptr2,#2
                MOV ptr3,#3
                MOV address,#0
                MOV nvm_ptr,#0
```

```
                SETB update
                RET
;*******************************************
;*******************************************
show_key:       JNB update, skip_show_key       ; nuevo
                CLR update              ; nuevo
                MOV A, mode
                JB acc.0, show_digit
                JB acc.1, show_char
                jb acc.2, show_nvm
                MOV mode,#01h
                JMP show_key

show_digit:     MOV DPTR,#digit_table
show_back:      ACALL load_ptr
                ACALL load_char
skip_show_key:  ACALL show_long
                        RET
;**********************************************
show_char:      mov dptr,#char_table
                sjmp show_back
;**********************************************
show_nvm:       ACALL scan_up
                RET
;*************************************************
load_ptr:       mov r1,#digit1
                mov r0,#ptr1
load_ptr_loop:  mov a,@r0
                movc a,@a+dptr
                mov @r1, a
                inc r0
                inc r1
                cjne r0,#ptr1+3, load_ptr_loop
                mov temp1, digit1
                mov temp2, digit2
                mov temp3, digit3
                ret
;*************************************************
load_char:      mov r1,#digit1
                MOV R0,#buffer
load_char_loop: mov a,@r1
                ACALL load_Ascii
                inc r1
                cjne r1,#digit1+3, load_char_loop
                ret
;*************************************************
load_ascii:     mov dptr,#ascii_table
                clr c
                subb a,#offset
                mov b,#5
                mul ab
                add a, dpl
                mov dpl,a
                clr a
                addc a, b
                addc a, dph
                mov dph, a
                mov r7,#5
load_Ascii_loop:clr a
                movc a,@a+dptr
                mov @r0, a
                inc r0
                inc dptr
```

```
                djnz r7, load_Ascii_loop
                ret
;*******************************************************
show_long:      MOV R2,#45
show_long2:     MOV R0,#buffer
                MOV data_port,#0
                MOV control_port,#0
show_loop:      MOV data_port,@R0
                call dly_2_show
                mov data_port,#0
                mov r7,#15
                djnz r7,$
                INC R0
                INC control_port
                CJNE R0,#buffer+15, show_loop
                DJNZ R2, show_long2
                RET
;*****************************************************
store_mem:      MOV A, mode
                jnb acc.2, store_mem2
                ACALL show_long
                ret

store_mem2:     ACALL get_next_Free
                jnc store_mem3
                ACALL mem_full
                ret

store_mem3:     MOV DATA_2_WRITE, TEMP1
                CALL WRITE
                INC ADDRESS
                MOV DATA_2_WRITE, TEMP2
                CALL WRITE
                INC ADDRESS
                MOV DATA_2_WRITE, TEMP3
                CALL WRITE
                ret
;**************************************************
get_next_Free:mov R6,#84
                mov address,#0
get_loop:       call read
                mov a, data_read
                jz exit_get_next
                mov a, address
                add a,#3
                mov address, a
                djnz r6, get_loop
                setb c
                ret

exit_Get_next:  clr c
                ret
;****************************************************
scan_key:       JB digit1_key, scan2
                SETB update
                call get_key1

                jnb digit1_key, scan_key

scan2:  JB digit2_key, scan3
                SETB update
                call get_key2
                jnb digit2_key, scan2
```

```
scan3:  JB digit3_key, scan4
                setb update
                call get_key3
                jnb digit3_key, scan3

scan4:  JB up_key, scan5
                SETB update
                call scan_up
                jnb up_key, scan4

scan5:  JB down_key, scan6
                SETB update
                call scan_down
                jnb down_key, scan5

scan6:  JB mode_key, scan7
                SETB update
                call get_mode
                jnb mode_key, scan6

scan7:  JB delete_key, scan8
                setb update
                call delete_mem

                JB delete_key, scan8
scan7b: call show_long
                JNB delete_key, scan7b

scan8:  JB store_key, scan9
                setb update
                call store_mem
                JB store_key, scan9

scan8b: call show_long
                JNB STORE_KEY,scan8b

scan9:  RET
;*****************************************************
get_key1:       MOV R1,#ptr1
                mov r0,#buffer
                call get
                ret

get_key2:       MOV R1,#ptr2
                mov r0,#buffer+5
                call get
                ret

get_key3:       MOV R1,#ptr3
                mov r0,#buffer+10
                call get
                ret
;**************************************************
get:            mov a, mode
                jb acc.0, get_digit
                jb acc.1, get_Char
                call show_long
                ret
;****************************************************
get_digit:      mov max_ptr,#max_digit_ptr
                mov dptr,#digit_table
back3:          call get_next_ptr
```

```
                call load_ptr
                call load_char
                call show_long
                ret

;***************************************************
get_char:       mov max_ptr,#max_char_ptr
                mov dptr,#char_table
                sjmp back3
;**************************************************
get_next_ptr:   inc @R1
                mov a,@r1
                cjne a,max_ptr, chk1
back1:          MOV @R1, #0
                sjmp skip1
chk1:           jnc back1
skip1:          ret
;***************************************************
;get_mode:      mov ptr1,#0
;               mov ptr2,#0
;               mov ptr3,#0
;               MOV A, mode
;               clr c
;               rlc a
;               ANL A,#00001111b                ; nuevo!!!
;               mov mode, a
;               JNB acc.3, exit_get_mode
;               MOV mode,#01h
;exit_get_mode: call show_key
;                       RET
;
;
;****************************************
;****************************************
get_mode:       MOV A, mode
                clr c
                rlc a
                ANL A,#00000111b                ; nuevo!!!
                MOV mode, A

back_mode:      JB acc.0, skip_mode_num
                JB acc.1, skip_mode_char
                JB acc.2, skip_mode_mem
                MOV mode,#01h
                MOV A, mode
                SJMP back_mode

skip_mode_num:  MOV ptr1,#1
                        MOV ptr2,#2
                        MOV ptr3,#3
                        call show_key
                        RET

skip_mode_char: MOV ptr1,#0
                        MOV ptr2,#1
                        MOV ptr3,#2
                        call show_key
                        RET

skip_mode_mem:  MOV address,#0
                        MOV nvm_ptr,#0
                        call show_key
                        RET
```

```
;*******************************************
;*******************************************
mem_full:       MOV R0,#buffer
mem_loop1:      mov a,@r0
                push acc
                inc r0
                cjne r0,#buffer+14, mem_loop1
                push digit1
                push digit2
                push digit3
                mov digit1,#"f"
                mov digit2,#"u"
                mov digit3,#"l"
                call load_char
                call show_long
                call show_long
                call show_long
                pop digit3
                pop digit2
                pop digit1
                mov r0,#buffer
mem_loop2:      pop acc
                mov @r0, a
                inc r0
                cjne r0,#buffer+14, mem_loop2
                ret
;**************************************************
digit_Table:    db "0123456789"
char_table:     db "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
;*************************************************************
ascii_Table:
DB              000H,000H,000H,000H,000H,       ; 20 space
DB              000H,000H,05fH,000H,000H,       ; 21 !
DB              000H,007H,000H,007H,000H,       ; 22 "
DB              014H,07fH,014H,07fH,014H,       ; 23 #
DB              024H,02aH,07fH,02aH,012H,       ; 24 $
DB              023H,013H,008H,064H,062H,       ; 25 %
DB              036H,049H,055H,022H,050H,       ; 26 &
DB              000H,005H,003H,000H,000H,       ; 27 '
DB              000H,01cH,022H,041H,000H,       ; 28 (
DB              000H,041H,022H,01cH,000H,       ; 29 )
DB              014H,008H,03eH,008H,014H,       ; 2a *
DB              008H,008H,03eH,008H,008H,       ; 2b +
DB              000H,050H,030H,000H,000H,       ; 2c ,
DB              008H,008H,008H,008H,008H,       ; 2d -
DB              000H,060H,060H,000H,000H,       ; 2e .
DB              020H,010H,008H,004H,002H,       ; 2f /
DB              03eH,051H,049H,045H,03eH,       ; 30 0
DB              000H,042H,07fH,040H,000H,       ; 31 1
DB              042H,061H,051H,049H,046H,       ; 32 2
DB              021H,041H,045H,04bH,031H,       ; 33 3
DB              018H,014H,012H,07fH,010H,       ; 34 4
DB              027H,045H,045H,045H,039H,       ; 35 5
DB              03cH,04aH,049H,049H,030H,       ; 36 6
DB              001H,071H,009H,005H,003H,       ; 37 7
DB              036H,049H,049H,049H,036H,       ; 38 8
DB              006H,049H,049H,029H,01eH,       ; 39 9
DB              000H,036H,036H,000H,000H,       ; 3a :
DB              000H,056H,036H,000H,000H,       ; 3b ;
DB              008H,014H,022H,041H,000H,       ; 3c <
```

```
DB                014H,014H,014H,014H,014H,        ; 3d =
DB                000H,041H,022H,014H,008H,        ; 3e >
DB                002H,001H,051H,009H,006H,        ; 3f ?
DB                032H,049H,079H,041H,03eH,        ; 40 @
DB                07eH,011H,011H,011H,07eH,        ; 41 A
DB                07fH,049H,049H,049H,036H,        ; 42 B
DB                03eH,041H,041H,041H,000H,        ; 43 C
DB                07fH,041H,041H,022H,01cH,        ; 44 D
DB                07fH,049H,049H,049H,000H,        ; 45 E
DB                07fH,009H,009H,009H,001H,        ; 46 F
DB                03eH,041H,049H,049H,07aH,        ; 47 G
DB                07fH,008H,008H,008H,07fH,        ; 48 H
DB                000H,041H,07fH,041H,000H,        ; 49 I
DB                020H,040H,041H,03fH,001H,        ; 4a J
DB                07fH,008H,014H,022H,041H,        ; 4b K
DB                07fH,040H,040H,040H,000H,        ; 4c L
DB                07fH,002H,00cH,002H,07fH,        ; 4d M
DB                07fH,004H,008H,010H,07fH,        ; 4e N
DB                03eH,041H,041H,041H,03eH,        ; 4f O
DB                07fH,009H,009H,009H,006H,        ; 50 P
DB                03eH,041H,051H,021H,05eH,        ; 51 Q
DB                07fH,009H,019H,029H,046H,        ; 52 R
DB                046H,049H,049H,049H,031H,        ; 53 S
DB                001H,001H,07fH,001H,001H,        ; 54 T
DB                03fH,040H,040H,040H,03fH,        ; 55 U
DB                01fH,020H,040H,020H,01fH,        ; 56 V
DB                03fH,040H,038H,040H,03fH,        ; 57 W
DB                063H,014H,008H,014H,063H,        ; 58 X
DB                007H,008H,070H,008H,007H,        ; 59 Y
DB                061H,051H,049H,045H,043H,        ; 5a Z
DB                000H,07fH,041H,041H,000H,        ; 5b [
DB                002H,004H,008H,010H,020H,        ; 5c Yen Currency Sign
DB                000H,041H,041H,07fH,000H,        ; 5d ]
DB                004H,002H,001H,002H,004H,        ; 5e ^
DB                040H,040H,040H,040H,040H,        ; 5f _
DB                000H,001H,002H,004H,000H,        ; 60 `
DB                020H,054H,054H,054H,078H,        ; 61 a
DB                07fH,048H,044H,044H,038H,        ; 62 b
DB                038H,044H,044H,044H,020H,        ; 63 c
DB                038H,044H,044H,048H,07fH,        ; 64 d
DB                038H,054H,054H,054H,018H,        ; 65 e
DB                008H,07eH,009H,001H,002H,        ; 66 f
DB                00cH,052H,052H,052H,03eH,        ; 67 g
DB                07fH,008H,004H,004H,078H,        ; 68 h
DB                000H,044H,07dH,040H,000H,        ; 69 i
DB                020H,040H,044H,03dH,000H,        ; 6a j
DB                07fH,010H,028H,044H,000H,        ; 6b k
DB                000H,041H,07fH,040H,000H,        ; 6c l
DB                07cH,004H,018H,004H,078H,        ; 6d m
DB                07cH,008H,004H,004H,078H,        ; 6e n
DB                038H,044H,044H,044H,038H,        ; 6f o
DB                07cH,014H,014H,014H,008H,        ; 70 p
DB                008H,014H,014H,018H,07cH,        ; 71 q
DB                07cH,008H,004H,004H,008H,        ; 72 r
DB                048H,054H,054H,054H,020H,        ; 73 s
DB                004H,03fH,044H,040H,020H,        ; 74 t
DB                03cH,040H,040H,020H,07cH,        ; 75 u
DB                01cH,020H,040H,020H,01cH,        ; 76 v
DB                03cH,040H,030H,040H,03cH,        ; 77 w
DB                044H,028H,010H,028H,044H,        ; 78 x
DB                00cH,050H,050H,050H,03cH,        ; 79 y
DB                044H,064H,054H,04cH,044H,        ; 7a z
DB                000H,008H,036H,041H,000H,        ; 7b <
```

```
DB              000H,000H,07fH,000H,000H,         ; 7c |
DB              000H,041H,036H,008H,000H,         ; 7d >
DB              010H,008H,008H,010H,008H,         ; 7e Right Arrow        ->
DB              078H,046H,041H,046H,078H          ; 7f Left Arrow <-

;*******************************************************
dly_2_Show:     MOV R7,#100
                DJNZ R7,$
                RET
;*******************************************
;*******************************************
compute_address:        MOV A, nvm_ptr
                        MOV B,#3
                        MUL ab
                        MOV address, A
                        RET
;*******************************************
;*******************************************
delete_mem:             call init_nvm
                        RET

;**********************************************************
include "c:\program files\batronix\prog-studio\include\jatto.asm"
;*********************************************
init_nvm:       MOV digit1,#"-"
                MOV digit2,#"-"
                MOV digit3,#"-"

                call load_char
                MOV address,#0
                MOV R1,#0
                MOV data_2_Write,#0
init_loop:      call write
                INC address
                DJNZ R1, init_loop

                MOV ADDRESS,#0
                MOV DATA_2_wRITE,#"M"
                CALL WRITE
                INC ADDRESS
                MOV DATA_2_WRITE,#"e"
                CALL WRITE
                INC ADDRESS
                MOV DATA_2_wRITE,#"m"
                CALL WRITE
                MOV mode,#01h
                RET
;**************************************
;****************************************
scan_up:
SCAN_DOWN:      MOV A, mode
                jb acc.2, scan_up2
                call show_long
                RET
;**************************************
;**************************************
scan_up2:
re_Scan:        MOV R0,#digit1
                MOV A, nvm_ptr
                call compute_address
                MOV address, a
scan_up_loop:call read
```

```
                MOV A, data_read
                JZ chk_down2
                MOV @R0, A
                call show_long_nvm
                INC R0
                INC address
                call read
                MOV @R0,data_Read
                call show_long_nvm
                INC address
                INC R0
                call read
                MOV @R0, data_read
                call load_char
                call show_long

not_yet:        JB up_key, CHK_DOWN
                INC nvm_ptr
                mov a, nvm_ptr
                cjne a,#85, chk2
back10: MOV nvm_ptr,#0
                jmp re_Scan

chk2:           JNC back10
                jmp re_scan

exit_scan_up:RET

CHK_DOWN:       JB DOWN_KEY, EXIT_SCAN_UP
chk_down2:      MOV A, NVM_PTR
                JZ RE_sCAN
                DEC NVM_PTR
                JMP RE_sCAN

;***************************************
show_long_nvm:  PUSH 00h
                        MOV R2,#5
                        call show_long2
                        POP 00h
                        RET
;*****************************************
;*****************************************
```