JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

"3D LED CUBE"

Submitted to
Jawaharlal Nehru Technological University Kakinada
in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
ELECTRONICS & COMMUNICATION ENGINEERING

Submitted by

K. THARUNIMA (13H71A0446) **Y. VIJAYA PHANEENDRA (13H71A0452)**

Ch. ASLESHA (13H71A0402) **J. GOPI BRAHMAM (14H75A0404)**

Under the Esteemed Guidance of
Mr. M.GOPALA KRISHNA M.Tech, [Ph.D]
Assistant Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
NBA Accredited

Devineni Venkata Ramana & Dr. Hima Sekhar
MIC College of Technology
Kanchikacherla, Krishna Dist, PIN: 521180, A.P, India.

2016-2017

ISO 9001:2008

Devineni Venkata Ramana & Dr. Hima Sekhar
MIC College of Technology
Kanchikacherla, Krishna Dist, PIN: 521180, A.P, India.

ISO 9001:2008

CERTIFICATE

This is to certify that the Main Project entitled **"3D LED CUBE"** is a bonafide work carried out by **K. THARUNIMA (13H71A0446), Y. VIJAYA PHANEENDRA (13H71A0452), Ch. ASLESHA (13H71A0402) and J. GOPIBRAHMAM (14H75A0404)** partial fulfillment for the award of degree Bachelor of Technology in Electronics and Communication Engineering of **Jawaharlal Nehru Technological University Kakinada** during the year 2016-2017.

(Mr. M.GOPALA KRISHNA)
Project Guide

(Mrs. A.SARADA)
Head of the Department

(Dr. Y.SUDHEER BABU)
Principal

Examiner 1

Examiner 2

ACKNOWLEDGEMENT

We would like to take this opportunity to express our deepest gratitude to the following people for their valuable contributions and assistance with this project.

Initially, we would like to thank our project supervisor, **Mr. M.GOPALA KRISHNA** **M.Tech., [Ph.D], Assistant Professor, Department of Electronics and Communication Engineering** for his guidance and support, especially for his valuable ideas and knowledge provided throughout the project. He is expertise and experience in **Embedded Systems & Signal Processing** makes valuable comments and suggestions have been very useful in solving problems encountered during the project.

We have the immense pleasure in expressing our thanks and deep sense of gratitude to**, Ms. A.SARADA, Head of the Department, Electronics and Communication Engineering** for extending necessary facilities for the completion of the Project.

We whole heartedly acknowledge **Dr.Y.SUDHEER BABU, Principal and Prof. D.PANDURANGA RAO, CEO** for giving opportunity to execute this Project. We also extend our thanks to all faculty members of **Electronics and Communication Engineering**, for their valuable guidance and encouragement in this Project.

We would like to extend our warm appreciation to all our friends for sharing us their knowledge, valuable contributions and help with this project.

Finally, our special thanks go to our families for their continuous support and help throughout our academics years and for their continual support and encouragement for this project.

K.THARUNIMA (13H71A0446)
Y.V.PHANEENDRA (13H71A0452)
CH. ASLESHA (13H71A0402)
J. GOPI BRAHMAM (14H75A0404)

DECLARATION

We **K.THARUNIMA (13H71A0446), Y.V.PHANEENDRA (13H71A0452), CH. ASLESHA (13H71A0402), J. GOPI BRAHMAM (14H75A0404)** of the main-Project **"3D LED CUBE",** hereby declare that the matter embodied in this Project is the genuine work done by us and has not been submitted either to this university or to any other university/institute for the fulfilment of the requirement of any course of study.

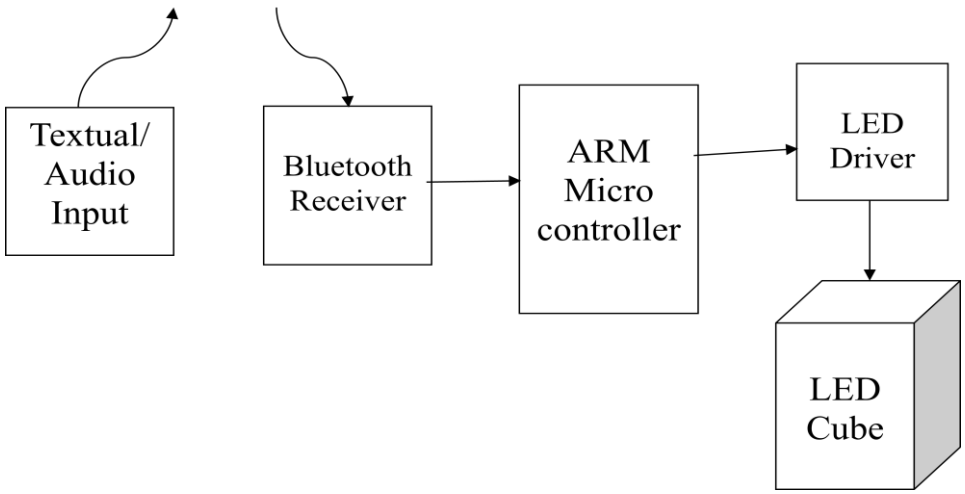
K.THARUNIMA (13H71A0446)
Y.V.PHANEENDRA (13H71A0452)
CH. ASLESHA (13H71A0402)
J. GOPI BRAHMAM (14H75A0404)

ABSTRACT

The goal of this project is to create a three dimensional 8x8x8 array of LEDs powered by the ARM7 microcontroller. This array of LEDs creates a cube shape that, with simple C programming, can display various graphics through the use of multiplexing and serial in, parallel out shift registers to create persistence of vision. The cube can also produce visual effects that correspond to an audio input in real time through the use of the MSGEQ7 integrated circuit.

There will be several options for display including non-directional animations and direction focused graphics. The processor will, through several inputs, decide what graphic to present and will feed it to an LED Driver Board (LDB). The LDB will then process the necessary data and output to the 512 LEDs to be used in the 3D array. The cube also has commercial potential due to its unique and interesting design, though not at its current cost of construction. All design goals were met for this project, and future improvement will most likely consist of interactivity between the cube and the user.

The block diagram of this project is as shown below:



CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

LED	Light Emitting Diode
ARM	Advanced RISC Machines
LDB	LED Driver Board
FPGA	Field Programmable Gate Array
PMOS	P-type metal-oxide-semiconductor logic
LDB	LED Driver Board
PCB	Printed Circuit Board
SVGA	Super Video Graphics Array
SMT	Surface Mount Technology
PC	Personal Computer
CMOS	Complemented metal-oxide-semiconductor
NMOS	N-type metal-oxide-semiconductor
TTL	Transistor Transistor Logic
CPU	Central Processing Unit
RISC	Reduced Instruction Set Computing
RTOS	Real Time Operating System
MSP	Managed Service Provider
PSP	Personal Software Process
CPSR	Computer Professional for Social Responsibility
VMSA	Virtual Memory System Architecture
PMSA	Protected Memory System Architecture
CISC	Complex Instruction Set Computing
SOC	System-on-Chip
IDM	Integrated device manufacturer
JTAG	Joint Test Action Group
TDMI	Thumb,Instruction, Debugger, Multiplier,
UART	Universal Asynchronous Receiver/Transmitter
PIO	Programmed input/output

MCU	Multipoint Control Unit Computing
GPS	Global Positioning System
AFH	Adaptive Frequency Hopping Feature
CSR	Communication Service Request
VDR	Voltage Detector Reset
SPP	Standard Parallel Port
ARQ	Automatic Retransmission on Request
SCO	Synchronous Connection Oriented
ACL	Access Control List
TDD	Time Division Duplex
ISM	International Safety Management
ASEE	American Society for Engineering Education
AVR	Augmented and Virtual Reality

CHAPTER-1

INTRODUCTION

Light organs have been a popular Do-It-Yourself electrical engineering design ever since their implementation in the 1970's. In their most basic design, light organs consisted of three lights: one to show low frequencies, one to show midrange frequencies, and one to show high frequencies. As technologies have advanced, improvements in this basic design have been made such as displaying the frequencies over two-dimensional banners as well as being able to select a greater number of frequencies to display.

This project takes the simple concepts of light organs and electronic banners and builds off of them to create an LED light organ in three dimensions. This audio-visual LED cube (3D LED light organ) consists of a 8x8x8 array of LEDs that is able to display low-resolution images as well as interact with different frequencies of an audio input through the use of an android application called ARM voice . The primary hardware design goal was to come up with a way to control all 512 LEDs (each with 2 leads that needed individual control) using the least amount of wires and components necessary but without sacrificing performance and controllability of the cube. The software design aspect of this project consisted of creating visually appealing light and animations. Hypothetically, this project has no definitive endpoint. An endless number of animations can be created, and a cleaner user interface could also be continuously developed. The LED cube is a viable commercial product with limitless potential in design applications.

This Manual shows step by step instructions for building Images 3D LED Cube. The LED matrix we decided to use is an 8x8x8 monochromatic LED Matrix. This is a total of 512 LEDs. The reason we chose this size, is that it provides the best of overall cube size, construction time & easier programming.

The advantage to using mono-chromatic LEDs allows us to use bright LEDs. Bright LEDs in our LED cube are viewable in a well lighted room. Mono-chromatic LEDs from well known manufacturer are cheap and affordable; usage of mono-chromatic LEDs allows easier construction and programming of Cube. While the obvious disadvantage of not using LEDs is that the Cube is only limited to one color. The current crop of LED does don't have the light intensity punch and when used in a 3D LED cube

must be viewed in a dark room. If you look at a 3D cube in a well lighted room chances are you can't even tell that it's on. In addition affordable LEDs are four times costlier than the mono-chromatic ones; the usage of LEDs would increase the difficulty level of construction and programming.

3D LED Displays come in all manner of shapes and sizes. The one thing they all share in common is the ability to display changing light patterns in three dimensional spaces.

A Typical 3D LED Display is a collection of LEDs, anywhere from 8 to thousands, somehow connected and arranged in a 3D pattern and controlled so that the LEDs can be made to turn on and off or vary in brightness in a controlled manner, thus creating interesting and pleasant patterns of light.

Probably the most common type of 3D LED display is the LED cube, though there have been several notable exceptions. The typical 3D LED display has LEDs that are either single color, multi color or full color RGB. The 3D matrix is constructed either by soldering the legs of the LEDs directly together or by constructing a matrix of wires which form a frame that the LEDs are connected to. There are a few exceptions to the rigid frame method including hanging from string or wires. There are also many different ways to control a 3D LED display which vary in complexity, the simplest is probably a single micro-controller, with more complicated methods using special LED Driver ICs or even direct PC control.

The other thing 3D LED displays have in common is that they are really cool! There is a lot of info on the internet and a lot of people who have started and finished 3D LED Display projects. Lots of those people have been very generous and have shared what they spent a lot of time and effort on so other people can learn and benefit from their work. The main problem I found is that most of this info is scattered widely around the net without a central place where people can easily find info and resources, or a community.

3D LED Displays have been steadily growing in popularity. There have now been several professional and commercial 3D LED display ventures which are still growing in

size, complexity and wonder! The focus of this site however is for projects undertaken by amateurs and hobbyists. Of course that is NOT to say that discussion and admiration of the amazing commercial projects isn't welcome here.

This conveys the design of a 3-Diemensional LED cube. The LED is constructed as an 8x8x8 cube with blue LEDs.

- The cube has some light up features: each LED flashes in sequence, the cube lights up entirely, a plane is generated at the middle of the cube and is rotated and a central axis like a teeter-totter, the plane is tipped the other direction, and lastly a plane is generated and moved from top to bottom of the cube.
- The details of electric and software design are expounded in this document. This includes the hardware components, how they are connected and detailed description of the code. Specifics of requirements and testing also appear in this document.
- For the cube design, the LED anodes are connected together to form the columns in the cube, and the LED cathodes are connected together to form the rows of the cube. To turn on any single LED, the column where it is located is powered by the PMOS corresponding to that column and the row where it is located is grounded by another PMOS corresponding to that row. Each column and row is powered by their own independent PMOS chips, allowing control over each LED independent of the others. Which LED is turned on at any given time is controlled by the microcontroller.
- Columns of LED anodes are connected together and rows LED cathodes are connected. Each column is powered separately and each row grounded separately. The current supplied to the LEDs is controlled through use of PMOS chips. To limit the number of pins required on the microcontroller, 3-to-8 decoders were used to map the 12 output pins to all 30 PMOS cube controls (25 columns and 5 rows). The power supply used for the decoders was 5V, which was synced to the board while the power supply for the cube was external since the board was unable to supply enough current to power the cube.

- Eight display sequences have been programmed for demonstration of this project. The code written simply cycles through all eight. With the hardware that has been built, a large variety of display sequence could be programmed using the methodologies that appear for the display sequences designed for this demonstration. The programming done for demonstration will be discussed in detail to illustrate these methodologies. See Appendix B for complete code.
- The next phase occurred once the interfacing of the cube with chips and microcontroller was completed. The function cube on worked as expected first try. The function Plane up down mostly worked aside from when to change the increment value. This was a simple fix. Flash was implemented after the testing procedure. It was a simple for loop that ran through each column and every row in that column before moving to the next column. A delay was inserted to leave the LEDs on for a recognizable amount of time.

1.1 Reaching Final Design:

The final project design was an 8x8x8 LED cube. The size of this finished product was decided mostly due to cost of LEDs. To reach this final goal, incremental steps were first taken to gain familiarity with the design and coding. The first step of this project was to gain the basic tools and knowledge to build a cube by first building a simpler size 3x3x3 single color LED cube. Although the 3x3x3 cube is more simplistic, there were many design similarities between it and larger cubes. Building this allowed us to foresee some problems with a larger-scale design and it also made coding simpler, which made the process of learning how to use the microcontroller easier and quicker. Figure 1 below shows the complete 3x3x3 single color cube connected to the microcontroller.

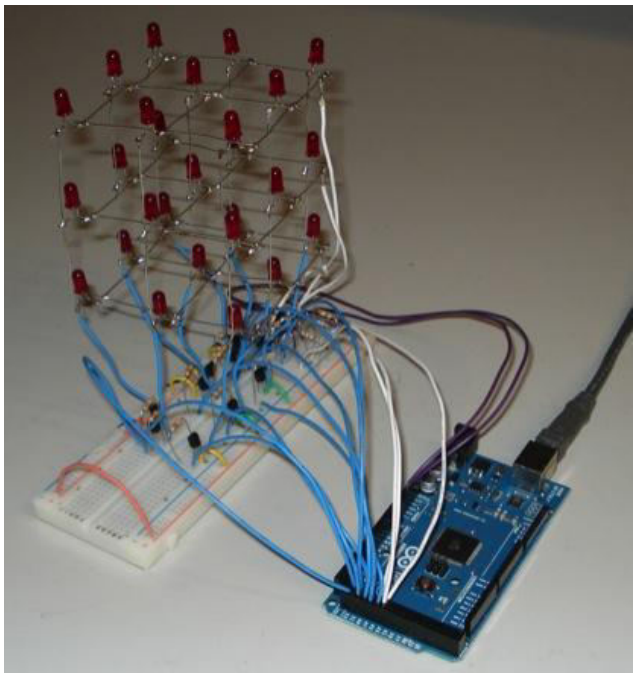


Figure 1.1: Figure showing of 3x3x3 LED matrix connected to Arduino UNO

CHAPTER-2

LITERATURE REVIEW

Over View:

Literature review has conducted before development to acquire sufficient knowledge and information on implementation of the wireless sensor. This chapter will briefly discuss about other projects on LED cube. Most of the technical part for constructing the LED cube is the same. The only difference that has been observed among the projects is the control circuit designed and the type of the microcontroller used.

2.1 Control circuit for 3x3x3:

Due to the size of the LED Cube that can be considered small, the circuit actually can be constructed on the base of the LED board. The resistors are placed into the proto board first. Then all the legs of the cube are soldered in to the board. Now all the connections of the circuit are done.

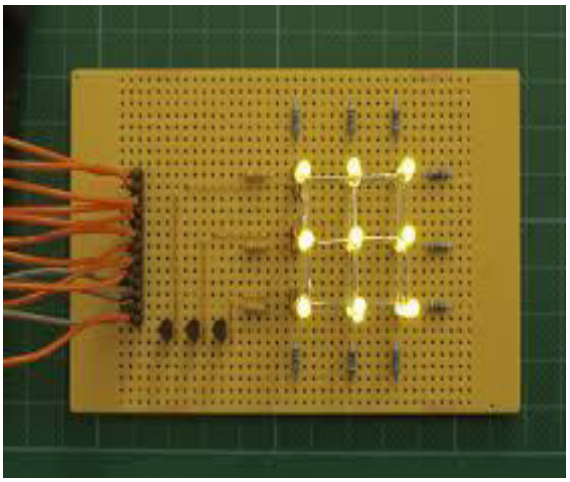


Figure 2.1.1: First stage of 3D LED CUBE

There are many different technologies that go into a 3D LED cube. Our literature survey addresses different technologies used in most available designs. A light-emitting (LED) diode is a semiconductor light source. LEDs are used extensively in everyday life. There are LEDs in TVs, Computer Monitors, traffic lights, signs, electronic gadgets, etc. LEDs contain no filament nor do they get particularly hot. They are illuminated solely by electrons moving through a semiconducting material. The lifespan of LEDs surpass incandescent lights by thousands of hours.

LEDs are also very efficient because they have a high luminous efficacy, which is the level of efficiency electricity is converted into visible light. LEDs are constructed to release the light from the diode outward. The diode is encased in a plastic bulb, which makes the light more efficient. There is an issue with LEDs that needs to be considered in all designs using Led. A way to circumvent this is to use a resistor. This type of LED has two pins; one anode pins and one cathode. The cathode is connected to ground while the anode is connected to a power supply (through a resistor). There are 256 different possible voltage levels.

The current will control the brightness of the LEDs. A vital part of any LED design is the central control system. This is what communicates to the LED grid what animations to execute. Below, we discuss a few of the options we looked into. ARDUINO, An Arduino is a microcontroller that offers access to an open-source platform and development environment. This system is based on the Atmel ATMega chip and offers a number of additional breakouts and features. These boards are designed to be inexpensive compared to other top microcontrollers or computer platforms. Arduino can also run cross platform allowing for development on Windows, Macintosh, or Linux operating systems.

This microcontroller is programmed through an AVR C based environment that includes large libraries of helpful functions. The board can be directly programmed through AVR C or used by expanding through C++ libraries. Arduino is known for its vast development community that makes learning and getting started easy. Raspberry Pi Raspberry Pi is a credit-card sized computer that can perform many standard desktop PC functions. This system was originally designed to offer low cost computers for education

purposes and is also a perfect research tool for many electronic projects. The Raspberry Pi is built around an ARM 11 processor. The microcomputer also includes standard USB 2.0 ports, the option of AV or HDMI video, standard 3.5mm audio output jack, Ethernet, several general purpose IO ports, and an I2C communication bus. The Raspberry Pi runs a standard Debian Linux distribution.

The system is programmed using Python by default, but can compile several other programming languages that support ARMv6 assembly. Standard PC A standard PC offers an abundance of processing power and memory compared to a typical microcontroller. It also offers the flexibility to use a wide range of well-developed programming languages and operating systems. However, the ability to interface with hardware through the typical general purpose input and output pins is not available. The cost of a machine is also much greater than that of a microcontroller, which is solely designed for integrated electronics projects. 4 Proceedings of the 2014 ASEE North Central Section Conference Copyright © 2014, American Society for Engineering Education Custom Microcontroller A custom microcontroller includes the selection of individual hardware parts to create a microcontroller specific to an electronics project. The use of a custom microcontroller offers the ability to meet exact requirements and is often ideal for mass-production projects in which the cost can be significantly lowered.

However, in single applications you can often get more for your money by purchasing a prebuilt system. The custom microcontroller approach will offer the advantage of working items such as the required voltages and number of output pins directly into the board itself. This prevents the need for workarounds or circuit modifications down the line. This approach also lacks the community examples and troubleshooting that many common microcontrollers have to offer. Without a shift register and demultiplexers, building an LED cube would be very difficult and very costly. Each LED would need a wire connecting it from the controller. Below, we discuss the technology behind each Shift Registers. A shift register is a sequential logic system that uses past inputs in addition to current inputs to create a specific output. Serial-in/parallel-out shift registers shift data into internal storage elements one clock cycle at a time and can output all data values at any given clock cycle.

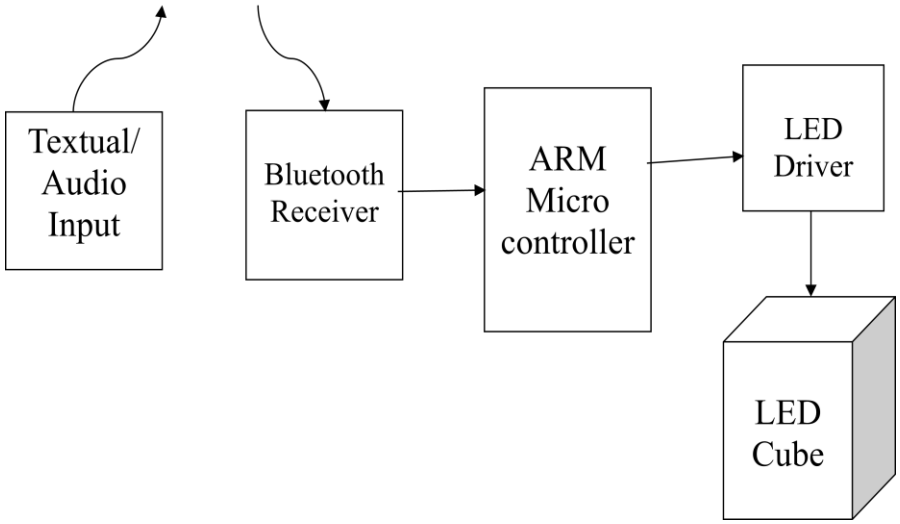
This allows data to be sent 1 bit per clock cycle and then allows for n bits to be outputted after n clock cycles. This is one method of expanding output ports for a system while not increasing bandwidth.

Demultiplexer: A demultiplexer is a system that converts bitwise input numbers into driven control lines for each number. This allows for n general purpose IO pins to be converted into 2n output pins. These output pins are driven based on the input value. For example, 2 input pins could be set to binary 2 which would drive the 3rd of the 4 output pins high. The downside of demultiplexers is that only one output can be set at any given time. This makes that transition rate an important factor for the overall bandwidth of the end system.

CHAPTER-3

BLOCK DIAGRAM

The block diagram of this project is as shown below.



3.1 BLUETOOTH

3.1.1 Introduction:

Bluetooth is a radio frequency specification for short range, point to point and point to multi point voice and data transfer. Bluetooth technology facilitates the replacement of cables normally used to connect one device to another by a short range radio link. With the help of blue tooth we can operate our keyboard and mouse without direct connection of CPU. Printers, fax machines, headphone, mouse, keyboard or any other digital devices can be part of Bluetooth system. In spite of facilitating the replacement of cables, Bluetooth technology works as a universal medium to bridge the existing data networks, a peripheral interface for existing devices and provide a mechanism to form short ad-hoc network of connected devices away from fixed network infrastructures.

Due to their independence on short range radio link, Bluetooth devices do not require a line of site connection in order to communicate. Therefore a computer can print information on a printer if printer is in inside the room. Two blue tooth devices can talk to each other when they come within range of 10 meters to each other. Bluetooth

technology represents an opportunity for the industry to deliver wireless solutions that are ubiquitous across a broad range of devices.

3.1.2 Nomenclature of Bluetooth:

While many new technologies bear technical names, like RS-232 or IEEE 802.11b, Bluetooth, the wireless technology, is different. Bluetooth was named for the 10th Century Viking king, Herald Blatand (A.K.A., Bluetooth) who peacefully united all the tiny island kingdoms of Denmark, southern Sweden, and southern Norway into one country. In keeping with its namesake, Bluetooth, the new low-cost radio technology, is designed to unite or connect all different types of devices to effectively work as one. By uniting devices, Bluetooth eliminates the need for cabling in a wide range of products, including cellular phones, PCs, headphones, audio equipment, printers, and many more.

Bluetooth Definitions:

- **Piconet:** Devices connected in an ad hoc fashion, which is, not requiring predefinition and planning, as with a standard network. Two to eight devices can be networked into a piconet. It is a peer network, which is, once connected; each device has equal access to the others. However, one device is defined as master, and the others as slaves.
- **Scatternet:** Several piconets may form a larger scatternet, with each piconet maintaining independence.
- **Master unit:** The master in a piconet whose clock and hopping sequence synchronizes the other devices.
- **Slave unit:** Devices in a piconet that are not the master.
- **MAC address:** Three bit address that distinguishes each unit in a piconet.
- **Parked units**: Piconet devices that are synchronized but don't have MAC addresses.
- **Sniff and hold mode:** Power-saving mode of a piconet device.

3.1.3 Working of Bluetooth Technology:

The technology of Bluetooth centers on a 9mm x 9mm microchip, which functions as a low cost and short range radio link. Bluetooth Technology provides a 10 meter personal bubble that support simultaneous transmission of both voice and data for multiple devices. Up to 8 devices can be connected in a piconet, and up to 10 piconets can exist within the 10 meter bubble. Each piconet support up to 3 simultaneous full duplex voice devices. The gross data rate is 1 Mb/s, but the actual data rate is 432 kbps for full duplex transmission, 721/56kbps for asymmetric transmission, and 384 kbps for tms2000 transmission. Bluetooth wireless technology is designed to be as secure as a wire with up to 128-bit public/private key authentication, and streaming cipher up to 64 bit based on a5 security.

Transmission types and rates:

The baseband (single channel per line) protocol combined circuit and packet switching. To assure that packets do not arrive out of order, slots (up to five) can be reserved for synchronous packets. As noted earlier, a different hop signal is used for each packet. Circuit switching can be either asynchronous or synchronous. Up to three synchronous (voice) data channels, or one synchronous and one asynchronous data channel, can be supported on one channel. Each synchronous channel can support a 64 Kb/s transfer rate, which is fully adequate for voice transmissions. An asynchronous channel can transmit as much as 721 Kb/s in one direction and 57.6 Kb/s in the opposite direction. It is also possible for an asynchronous connection to support 432.6 Kb/s in both directions if the link is symmetric.

Radio frequency and spectrum hopping:

What if there's a lot of radio noise? Won't that interfere with Bluetooth connections? As a rule, the answer is no. It is designed to use fast acknowledgement and frequency hopping, which will make connections robust. It is packet-based, and will jump to a new frequency after each packet is received, which not only helps limit interference problems, but also adds to security. Data rates are one megabyte/second, including headers. Full duplex transmissions (both directions at once) are accomplished via time division multiplexing.

The Bluetooth radio chip functions at 2.4 gigahertz, which is in the unlicensed ISM (Industrial Scientific Medical) band. It separates the 2.4 gigahertz frequency band into 79 hops one megahertz apart, starting with 2.402 and ending with 2.480 (though this bandwidth is narrower in Japan, France, and Spain). This spread spectrum is used to hop from one channel to another, pseudo-randomly, which adds a strong layer of security. Up to 1600 hops per second can be made. The standard frequency range is 10 centimeters to 10 meters, and can be extended to at least 100 meters by increasing transmission power.

3.1.4 Data Transmission:

Data can be transmitted both synchronously and asynchronously. The Synchronous Connection Oriented (SCO) method is used primarily for voice, and Asynchronous Connectionless (ACL) is primarily for data. Within a piconet, each master-slave pair can use a different transmission mode, and modes can be changed at any time. Time Division Duplex (TDD) is used by both SCO and ACL, and both support 16 types of packets, four of which are control packets that are the same in each type. Because of the need for smoothness in data transmission, SCO packets are generally delivered via reserved intervals, that is, the packets are sent in groups without allowing other transmissions to interrupt. SCO packets can be transmitted without polling by the sending unit. ACL links support both symmetric and asymmetric transmissions.

Bandwidth is controlled by the master unit, which determines how much of the total each slave unit can use. Slaves cannot transmit data until they have been polled by the master, and the master can broadcast messages to the slave units via ACL link.

Network arrangement:

Bluetooth network arrangements (topology) can be either point-to-point or point-to-multipoint. Any unit in a piconet can establish a connection to another piconet to form a scatternet. See the figure, which diagrams a scatternet in which piconet A, which consists of four units, is connected to piconet B, consisting of two units. Note that the master unit of A is not the link Bluetooth network arrangements (topology) can between the two piconets.

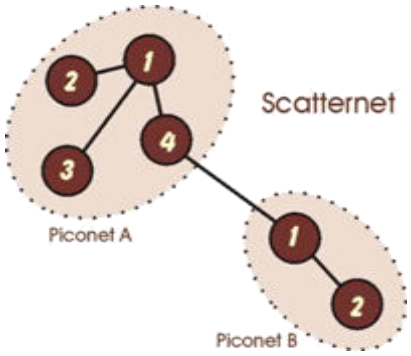


Figure 3.1.4.1 Piconets A&B

Error Correction and Security:

The FEC methods are designed to reduce the number of retransmissions. However, the over **hear** Three error correction techniques have been defined: 1/3 rate forward error corrected significantly slows transmissions, so is generally not used in relatively error-free environments, with the exception of packet headers. The ARQ scheme requires that the header error and cyclic redundancy checks are okay. When they are, an acknowledge is sent. When they aren't, the data is resent.

Security is provided in three ways: through pseudo-random frequency band hops, authentication, and encryption. Frequency band hops make it difficult for anyone to eavesdrop. Authentication allows a user to control connectivity to only devices specified. Encryption uses secret key lengths of 1, 40, and 64bits. The quality of security is excellent for most applications. However, it is not the highest level available, and for those users who require it, the suggestion is to investigate separate network transfer protocols and security software.

3.1.5 Bluetooth Characteristics:

These are the features of the Bluetooth technology:

- It separates the frequency band into hops. This spread spectrum is used to hop from one channel to another, which adds a strong layer of security.
- Up to eight devices can be networked in a piconet.
- Signals can be transmitted through walls and briefcases, thus eliminating the need for line-of-sight.
- Devices do not need to be pointed at each other, as signals are Omni-directional.

- Both synchronous and asynchronous applications are supported, making it easy to implement on a variety of devices and for a variety of services, such as voice and Internet.
- Governments worldwide regulate it, so it is possible to utilize the same standard wherever one travels.

3.1.6 Advantages of Bluetooth Technology:

1. No line of site restrictions as with IrDA.
2. Low power consumption makes integrated in battery powered devices very practical.
3. 2.4 GHz radio frequency ensures worldwide operability.
4. Tremendous momentum not only within the computer industry but other industries like cellular telephones and transportation.

3.1.7 HC-05 Bluetooth Module:

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication.

Figure 3.1.7.1 Chip Diagram of BLUETOOTH HC-05

This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and

baseband. It uses CSR Blue core 04-External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature).

The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc. Just go through the datasheet for more details

Hardware Features:

- Typical -80dBm sensitivity.
- Up to +4dBm RF transmits power.
- 3.3 to 5 V I/O.
- PIO (Programmable Input/output) control.
- UART interface with programmable baud rate.
- With integrated antenna.
- With edge connector.

Software Features:

- Slave default Baud rate: 9600, Data bits: 8, Stop bit: 1, Parity: No parity.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"1234" as default.

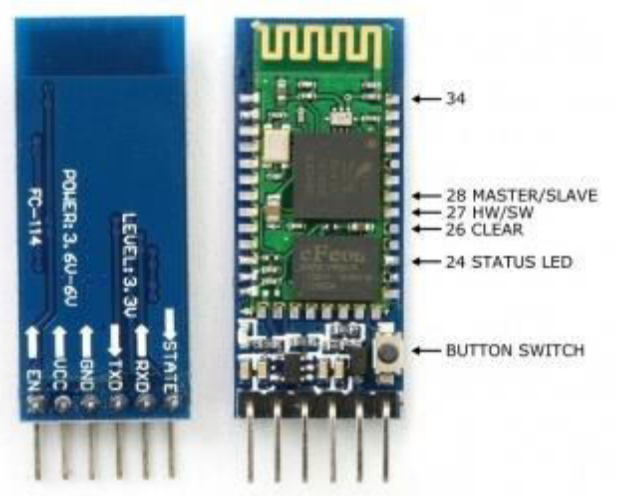


Figure 3.1.7.2 Diagrams for Pin Description of BLUETOOTH HC-05

Pin Description:

The HC-05 Bluetooth Module has 6pins. They are as follows:

ENABLE:

When enable is pulled LOW, the module is disabled which means the module will not turn on and it fails to communicate. When enable is left open or connected to 3.3V, the module is enabled i.e. the module remains on and communication also takes place.

Vcc:

Supply Voltage 3.3V to 5V

GND:

Ground pin

TXD & RXD:

These two pins acts as an UART interface for communication

STATE:

It acts as a status indicator. When the module is not connected to / paired with any other Bluetooth device, signal goes Low. At this low state, the led flashes continuously which denotes that the module is not paired with other device. When this module is connected to/paired with any other Bluetooth device, the signal goes high. At this high state, the LED blinks with a constant delay which indicates that the module is paired.

Button Switch:

This is used to switch the module into AT command mode. To enable AT command mode, press the button switch for a second. With the help of AT commands, the user can change the parameters of this module but only when the module is not paired with any other BT device. If the module is connected to any other Bluetooth device, it starts to communicate with that device and fails to work in AT command mode.

3.1.8 Interfacing Bluetooth Module with Micro-controller:

The interface of Bluetooth Module with LPC2148 involves the connection between the HC – 05 Bluetooth Module and the ARM7 based MCU LPC2148. The details of the complete circuit design are as follows. The HC – 05 Bluetooth module is an UART based device. Hence, the connection between the LPC2148 and Bluetooth module requires only two wires. In LPC2148, there are two UART modules: UART 0 and UART 1. The UART 0 in our development board is connected to the USB to serial converter.

So, we are connecting the Bluetooth Module to the UART 1 i.e. PORT0 Pins P0.8 (Pin 33) and P0.9 (Pin 34). The TX pin of the Bluetooth Module is connected to RXD1 (P0.9 – Pin 34) of LPC2148. Similarly, the RX pin of the Bluetooth Module is connected to TXD1 (P0.8 – Pin 33) of LPC2148.

Note: The TX and RX pins of Bluetooth Module can tolerate only 3.3V. Since the output of the LPC2148 MCU is only 3.3V, the connections can be made directly.

The Bluetooth Module has on – board 3.3V regulator... As the board already consists of LEDs, we are not connecting any additional LEDs. On the development board, the LEDs are connected to PORT1 Pins P1.18 to P1.25. So, we will write the program as per these pins.

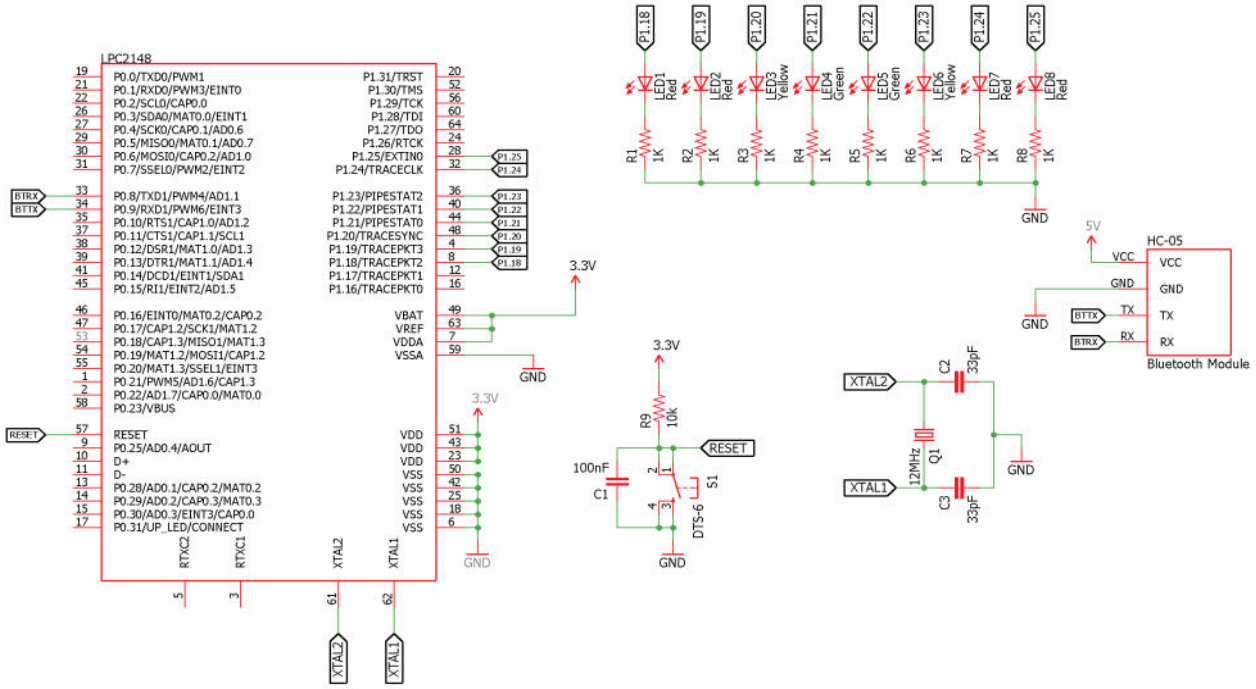


Figure 3.1.8.1 Interfacing of LPC2148 with Bluetooth Module

3.2 ARM7:

ARM7 is a group of older 32-bit RISC ARM processor cores licensed by ARM Holdings for microcontroller use. ARM7 cores are no longer recommended for new designs, instead ARM Cortex-M or ARM Cortex-R cores should be considered. It is a versatile processor designed for mobile devices and other low power electronics. This processor architecture is capable of up to 130 MIPS on a typical 0.13 µm process. The ARM7TDMI processor core implements ARM architecture **v4T**. The processor supports both 32-bit and 16-bit instructions via the ARM and Thumb instruction sets. ARM licenses the processor to various semiconductor companies, which design full chips based on the ARM processor architecture.

Overview of ARM7:

This generation introduced the Thumb 16-bit instruction set providing improved code density compared to previous designs. The most widely used ARM7 designs

implement the ARMv4T architecture, but some implement ARMv3 or ARMv5TEJ. ARMv7 has 37 registers (31 GPR and 6 SPR).All these designs use Von Neumann architecture, thus the few versions comprising a cache do not separate data and instruction caches.

Some ARM7 cores are obsolete. One historically significant model, the **ARM7DI** is notable for having introduced JTAG based on-chip debugging; the preceding ARM6 cores did not support it. The "D" represented a JTAG TAP for debugging; the "I" denoted an ICE Breaker debug module supporting hardware breakpoints and watch points, and letting the system.

ARM license:

ARM Holdings neither manufactures nor sells CPU devices based on its own designs, but rather licenses the processor architecture to interested parties. ARM offers a variety of licensing terms, varying in cost and deliverables.

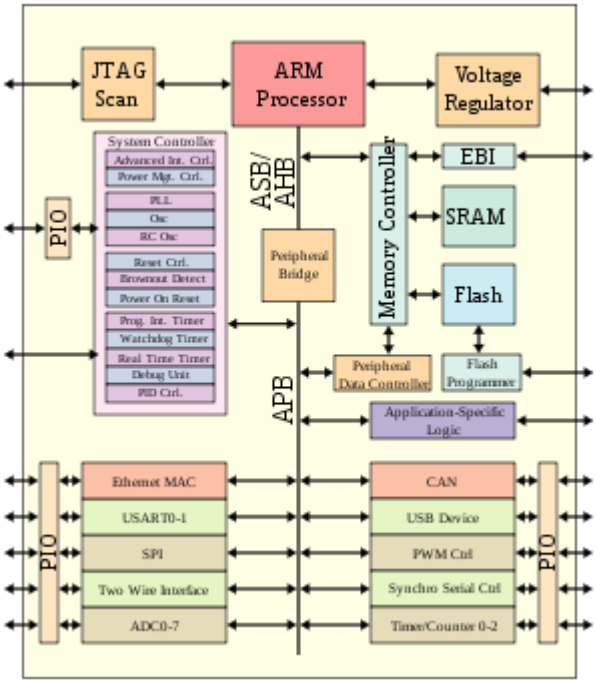


Figure 3.2.1 Microprocessor based System on Chip

To all licensees, ARM provides an integrable hardware description of the ARM core, as well as complete software development toolset and the right to sell manufactured silicon containing the ARM CPU.

Silicon customization:

Integrated device manufacturer (IDM) receive the ARM Processor IP as synthesizable RTL (written in Verilog). In this form, they have the ability to perform architectural level optimizations and extensions.

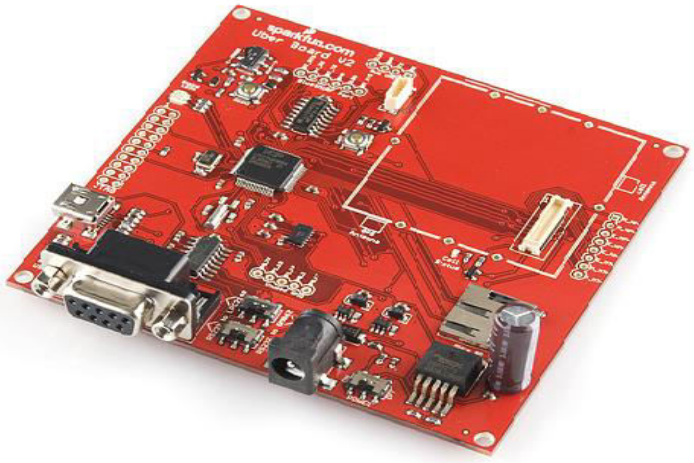


Figure 3.2.2 ARM Board

This allows the manufacturer to achieve custom design goals, such as higher clock speed, very low power consumption, instruction set extensions, optimizations for size, debug support, etc. To determine which components have been included in a particular ARM CPU chip, consult the manufacturer datasheet and related documentation.

3.2.1 ARM Architecture:

ARM, originally **Acorn RISC Machine**, later **Advanced RISC Machine**, is a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments. British company ARM Holdings develops the architecture and licenses it to other companies, who design their own products that implement one of those architectures—including System-on-

Chip (SoC) that incorporate memory, interfaces, radios, etc. It also designs cores that implement this instruction set and licenses these designs to a number of companies that incorporate those core designs into their own products.

A RISC-based computer design approach means processors require fewer transistors than typical Complex Instruction Set Computing (CISC) x86 processors in most personal computers. This approach reduces costs, heat and power use. These characteristics are desirable for light, portable, battery-powered devices-including smart phones, laptops and tablet computers, and other embedded systems. For supercomputers, which consume large amounts of electricity, ARM could also be a power-efficient solution.

The ARM7 family is a range of low-power 32-bit RISC microprocessor cores optimized for cost and power-sensitive consumer applications. The ARM7 family incorporates the Thumb 16-bit instruction set - enabling 32-bit performance at 8/16-bit system cost. About the ARMv7 architecture, and architecture profiles ARMv7 is documented as a set of architecture profiles.

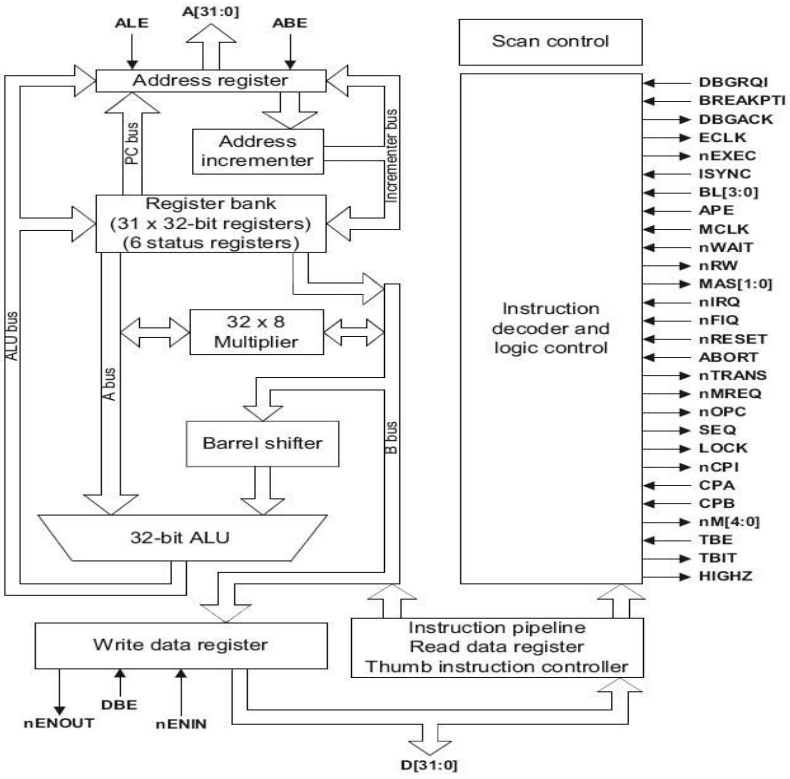


Figure 3.2.1.1 ARM Architecture Diagram

The profiles are defined as follows:

ARMv7-A: The application profile for systems supporting the ARM and Thumb instruction sets, and requiring virtual address support in the memory management model.

ARMv7-R: The real-time profile for systems supporting the ARM and Thumb instruction sets, and requiring physical addresses only support in the memory management model.

ARMv7-M: The microcontroller profile for systems supporting only the Thumb instruction set, and where overall size and deterministic operation for an implementation are more important than absolute performance. While profiles were formally introduced with the ARMv7 development, the A-profile and R-profile have implicitly existed in earlier versions, associated with the Virtual Memory System Architecture (VMSA) and Protected Memory System Architecture (PMSA) respectively

Instructions for ARM Holdings cores have 32-bit fixed-length instructions, but later versions of the architecture also support a variable-length instruction set that provides both 32- and 16-bit instructions for improved code density. Some cores can also provide hardware execution of java code density. With over 50 billion ARM processors produced as of 2014, ARM is the most widely used instruction set architecture in terms of quantity produced. Currently, the widely used Cortex cores, older "classic" cores, and specialized secure core cores variants are available for each of these to include or exclude optional capabilities.

32- Bit architecture:

The 32-bit ARM architecture, such as **ARMv7-A**, is the most widely used architecture in mobile devices. Since 1995, the ARM architecture reference manual has been the primary source of documentation on the ARM processor architecture and instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction semantics) from implementation details that may vary. The architecture has evolved over time, and version seven of the architecture, ARMv7, defines three architecture "profiles":

- A-profile, the "Application" profile, implemented by 32-bit cores in the Cortex-A series and by some non-ARM cores.
- R-profile, the "Real-time" profile, implemented by cores in the Cortex-R series.
- M-profile, the "Microcontroller" profile, implemented by most cores in the Cortex-M series.

Although the architecture profiles were first defined for ARMv7, ARM subsequently defined the ARMv6-M architecture (used by the Cortex M0/M0+/M1) as a subset of the ARMv7-M profile with fewer instructions.

3.2.2 CPU Modes:

Except in the M-profile, the 32-bit ARM architecture specifies several CPU modes, depending on the implemented architecture features. At any moment in time, the CPU can be in only one mode, but it can switch modes due to external events (interrupts) or programmatically.

- User mode: The only non-privileged mode.
- FIQ mode: A privileged mode that is entered whenever the processor accepts an FIQ interrupt.
- IRQ mode: A privileged mode that is entered whenever the processor accepts an IRQ interrupt.
- Supervisor (svc) mode: A privileged mode entered whenever the CPU is reset or when an SVC instruction is executed.
- Abort mode: A privileged mode that is entered whenever a prefetch abort or data abort exception occurs.
- Undefined mode: A privileged mode that is entered whenever an undefined instruction exception occurs.
- System mode (ARMv4 and above): The only privileged mode that is not entered by an exception. It can only be entered by executing an instruction that explicitly writes to the mode bits of the CPSR.
- Monitor mode (ARMv6 and ARMv7 Security Extensions, ARMv8 EL3): A monitor mode is introduced to support Trust Zone extension in ARM cores.

- Hype mode (ARMv7 Virtualization Extensions, ARMv8 EL2): A hypervisor mode that supports Popek and Goldberg virtualization requirements for the non-secure operation of the CPU.
- Thread mode (ARMv6-M, ARMv7-M, and ARMv8-M): A mode which can be specified as either privileged or unprivileged, while whether Main Stack Pointer (MSP) or Process Stack Pointer (PSP) is used can also be specified in CONTROL register with privileged access. This mode is designed for user tasks in RTOS environment but it's typically used in bare-metal for super-loop.
- Handler mode (ARMv6-M, ARMv7-M, and ARMv8-M): A mode dedicated for exception handling (except the RESET which are handled in Thread mode). Handler mode always uses MSP and works in privileged level.

Instruction set:

The original (and subsequent) ARM implementation was hardwired without microcode, like the much simpler 8-bit 6502 processor used in prior Acorn microcomputers. The 32-bit ARM architecture (and the 64-bit architecture for the most part) includes the following RISC features:

- Load/store architecture.
- No support for unaligned memory accesses in the original version of the architecture. ARMv6 and later, except some microcontroller versions, support unaligned accesses for half-word and single-word load/store instructions with some limitations, such as no guaranteed atomicity.
- Uniform 16× 32-bit register file (including the program counter, stack pointer and the link register).
- Fixed instruction width of 32 bits to ease decoding and pipelining, at the cost of decreased code density. Later, the Thumb instruction set added 16-bit instructions and increased code density.
- Mostly single clock-cycle execution.

To compensate for the simpler design, compared with processors like the Intel 80286 and Motorola 68020, some additional design features were used:

- Conditional execution of most instructions reduces branch overhead and compensates for the lack of a branch predictor.
- Arithmetic instructions alter condition codes only when desired.
- 32-bit barrel shifter can be used without performance penalty with most arithmetic instructions and address calculations.
- Has a powerful indexed addressing mode.
- A link register supports fast leaf function calls.
- A simple, but fast, 2-priority-level interrupt subsystem has switched register banks.

3.2.3 Arithmetic Instructions:

ARM includes integer arithmetic operations for add, subtract, and multiply; some versions of the architecture also support divide operations. ARM supports 32-bit x 32-bit multiplies with either a 32-bit result or 64-bit result, though Cortex-M0 / M0+ / M1 cores don't support 64-bit results. Some ARM cores also support 16-bit x 16-bit and 32-bit x 16-bit multiplies.

The divide instructions are only included in the following ARM architectures:

- ARMv7-M and ARMv7E-M architectures always include divide instructions.
- ARMv7-R architecture always includes divide instructions in the Thumb instruction set, but optionally in its 32-bit instruction set.
- ARMv7-architecture optionally includes the divide instructions. The instructions might not be implemented, or implemented only in the Thumb instruction set, or implemented in both the Thumb and ARM instruction sets, or implemented if the Virtualization Extensions are included.

Registers:

Registers R0 through R7 are the same across all CPU modes; they are never banked. Registers R8 through R12 are the same across all CPU modes except FIQ mode. FIQ mode has its own distinct R8 through R12 registers. R13 and R14 are banked across all privileged CPU modes except system mode. That is, each mode that can be entered because of an exception has its own R13 and R14. These registers generally contain the stack pointer and the return address from function calls, respectively.

- R13 is also referred to as SP, the Stack Pointer.
- R14 is also referred to as LR, the Link Register.
- R15 is also referred to as PC, the Program Counter.

The Current Program Status Register (CPSR) has the following 32 bits.

- M (bits 0–4) is the processor mode bits.
- T (bit 5) is the Thumb state bit.
- F (bit 6) is the FIQ disable bit.
- I (bit 7) is the IRQ disable bit.
- A (bit 8) is the imprecise data abort disable bit.
- E (bit 9) is the data endianness bit.
- IT (bits 10–15 and 25–26) is the if-then state bits.
- GE (bits 16–19) is the greater-than-or-equal-to bits.
- DNM (bits 20–23) is the do not modify bits.
- J (bit 24) is the Java state bit.
- Q (bit 27) is the sticky overflow bit.
- C (bit 29) is the carry/borrow/extend bit.
- Z (bit 30) is the zero bit.

Key criteria for ARMv7 implementations are as follows:

- Enable implementations with industry leading power, performance and area constraints:
  — provides opportunities for simple pipeline designs offering leading edge system performance levels in a broad range of markets and applications.

• Highly deterministic operation:

  — Single or low cycle count execution
  — Minimal interrupt latency, with short pipelines
  — cache less operation.

• Excellent C/C++ target. This aligns with the ARM programming standards in this area:

— Exception handlers are standard C/C++ functions entered using standard calling conventions.

• Designed for deeply embedded systems:

— Low pin count devices

— enables new entry level opportunities for the ARM architecture.

• Provides debug and software profiling support for event driven systems.

This manual is specific to the ARMv7 profile.

3.3 LED Driver Board:

LEDs, being PN junctions, are non linear devices which require specific care to ensure that the device is operated within its physical limits. PN junctions have an exponential increase in current dissipation with a linear increase in bias voltage, making the current an extremely important factor for powering the LEDs. For this reason, the 3D LED Cube uses constant current technology when driving the LEDs.

This ensures that the thermal and physical characteristics of the LEDs are not passed, and breakdown does not occur.

There are numerous LED drivers available on the market, targeting specific segments and applications, some meant for driving only a small number of LEDs, some meant to power very large LEDs, and some meant to power high quantities of LEDs. For the 3D LED cube, the group is looking to find a LED driver that can handle both adequate current, and the number of outputs necessary, while being available in an easy to assemble package.

Most LED drivers have the same basic architecture. Typically, some form of a back-plane serial connection provides input data to the registers within the chip. An internal oscillator drives a counter with which each register is compared with to produce the outputs necessary. When the register equals the counter, the output is switched to logic on. The serial nature of these LED drivers often run in the mega-hertz range, requiring a high speed link in order to be most effective. A communication of this nature may best be suited to a FPGA.

The output of LED drivers come in two major variants, open drain/collector, and standard TTL. An open drain works by connecting the output of the chip to the drain of the internal MOSFET, or collector of the internal BJT, depending on what the IC is built with. The open drain can then be used to sink current, or connect the output to ground, allowing a large range of voltages to be used with the output.

The LED driver board is a simple component of the LED cube, controlling only current, brightness, and. Each of these factors will be tested individually from each channel. The channels will be calibrated so that the same brightness appears for all LED's across each LED driver channel and device.

3.4 LEDs:

A **light-emitting diode** (**LED**) is a two-lead semiconductor light source. It is a p-n junction diode, which emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor. LEDs are typically small (less than 1 mm^2) and integrated optical components may be used to shape the radiation pattern.

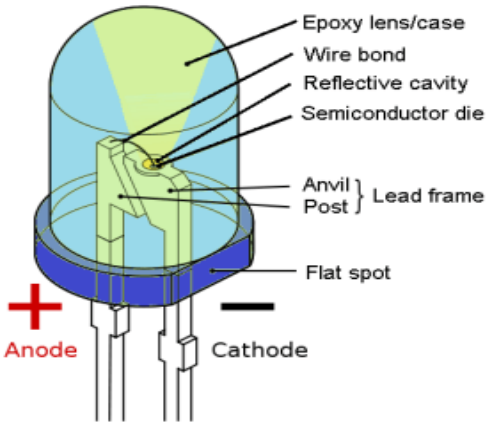


Figure 3.4.1 Parts of an LED

Appearing as practical electronic components in 1962, the earliest LEDs emitted low-intensity infrared light. Infrared LEDs are still frequently used as transmitting elements in remote-control circuits, such as those in remote controls for a wide variety of consumer electronics. The first visible-light LEDs were also of low intensity and limited to red. Modern LEDs are available across the visible, ultraviolet and infrared wavelengths, with very high brightness.

Early LEDs were often used as indicator lamps for electronic devices, replacing small incandescent bulbs. They were soon packaged into numeric readouts in the form of seven segment displays and were commonly seen in digital clocks. Recent developments in LEDs permit them to be used in environmental and task lighting. LEDs have allowed new displays and sensors to be developed, while their high switching rates are also used in advanced communications technology.

LEDs have many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved physical robustness, smaller size, and faster switching. Light-emitting diodes are now used in applications as diverse as aviation lighting, automotive headlamps, advertising, general lighting, traffic signals, camera flashes, and lighted wallpaper. As of 2017, LED lights home room lighting is as cheap or cheaper than compact fluorescent lamp sources of comparable output. They are also significantly more energy efficient and, arguably, have fewer environmental concerns linked to their disposal.

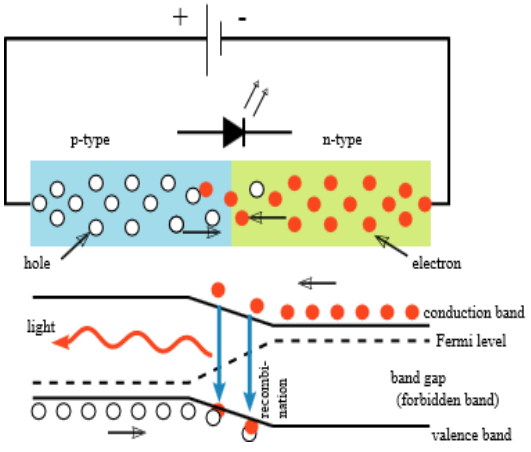


Figure 3.4.2 The inner workings of an LED, showing circuit (top) and band diagram (bottom)

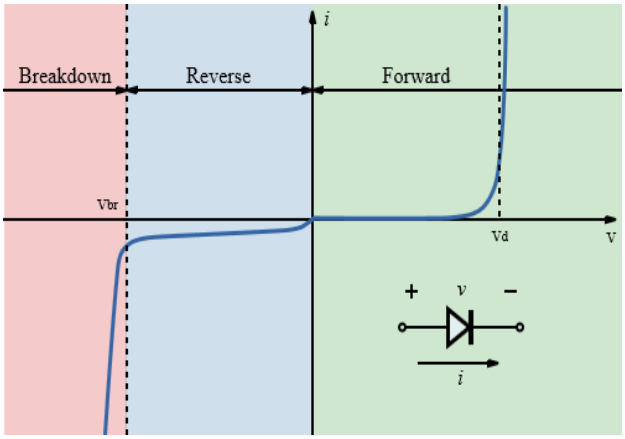


Figure 3.4.1.1 I-V diagram for a diode

3.4.1 Working of LED

A P-N junction can convert absorbed light energy into a proportional electric current. The same process is reversed here (i.e. the P-N junction emits light when electrical energy is applied to it). This phenomenon is generally called electroluminescence which can be defined as the emission of light from a semi conductor under the influence of an electric field. The charge carriers recombine in a forward-biased P-N junction as the electrons cross from the N-region and recombine with the holes existing in the P-region. Free electrons are in the conduction band of energy levels, while holes are in the valence energy band. Thus the energy level of the holes will be lesser than the energy levels of the electrons. Some portion of the energy must be dissipated in order to recombine the electrons and the holes. This energy is emitted in the form of heat and light.

The electrons dissipate energy in the form of heat for silicon and germanium diodes but in gallium arsenide phosphide (GaAsP) and gallium phosphide (GaP) semiconductors, the electrons dissipate energy by emitting photons. If the semiconductor is translucent, the junction becomes the source of light as it is emitted, thus becoming a light-emitting diode, but when the junction is reverse biased no light will be produced by the LED and, if the potential is great enough, the device will be damaged.

3.4.2 Blue LED

Blue LEDs were first developed by Herbert Paul Maruska at RCA in 1972 using gallium nitride (GaN) on a sapphire substrate. SiC-types were first commercially sold in the United States by Cree in 1989. However, neither of these initial blue LEDs was very bright.

The first high-brightness blue LED was demonstrated by Shuji Nakamura of Nichia Corporation in 1994 and was based on InGaAs In 2001 and 2002; processes for growing gallium nitride (GaN) LEDs on silicon were successfully demonstrated. In January 2012, Osram demonstrated high-power InGaN LEDs grown on silicon substrates commercially.

Table1. physical parameters of Blue LED

Color	**Wave length [nm]**	**Voltage drop [ΔV]**	**Semiconductor material**
Blue	$450 < \lambda < 500$	$2.48 < \Delta V < 3.7$	Zinc selenide (ZnSe) Indium gallium nitride(InGaN) Silicon carbide (SiC) as substrate Silicon(Si) as substrate—under development

Efficiency and operational parameters:

Typical indicator LEDs are designed to operate with no more than 30–60 mill watts (mW) of electrical power. a conventional incandescent light bulb of 60–100 watts emits around 15 lm/W, and standard fluorescent light emit up to 100 lm/W.In September 2003, a new type of blue LED was demonstrated by Cree that consumes 24 mW at 20 mill amperes (mA).

Lifetime and failure:

Solid-state devices such as LEDs are subject to very limited wear and tear if operated at low currents and at low temperatures. Typical lifetimes quoted are 25,000 to 100,000 hours, but heat and current settings can extend or shorten this time significantly.

The most common symptom of LED failure is the gradual lowering of light output and loss of efficiency. Sudden failures, although rare, can also occur. Early red LEDs were notable for their short service life. With the development of high-power LEDs, the devices are subjected to higher junction temperatures and higher current densities than traditional devices. This causes stress on the material and may cause early light-output degradation. To quantitatively classify useful lifetime in a standardized manner it has been suggested to use L70 or L50, which are the runtimes (typically given in thousands of hours) at which a given LED reaches 70% and 50% of initial light output, respectively.

3.5 SHIFT REGISTER

The 74HC595 is an 8-bit serial-in/serial or parallel-out shift register with a storage register and 3-state outputs. Both the shift and storage register have separate clocks. The device features a serial input and a serial output to enable cascading and an asynchronous reset MR input. A LOW on MR will reset the shift register. Data is shifted on the LOW-to-HIGH transitions of the SHCP input.

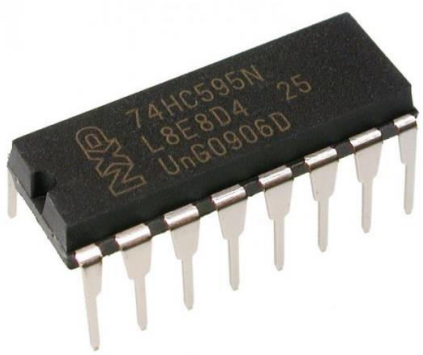


Figure 3.5.1 74HC595N Shift Register Diagram

The data in the shift register is transferred to the storage register on a LOW-to-HIGH transition of the STCP input. If both clocks are connected together, the shift register will always be one clock pulse ahead of the storage register. Data in the storage register appears at the output whenever the output enable input is LOW. A HIGH on OE causes the outputs to assume a high-impedance OFF-state. Operation of the OE input does not affect the state of the registers. Inputs include clamp diodes. This enables the use of current limiting resistors to interface inputs to voltages in excess of VCC.

Features and Benefits:

• 8-bit serial input

• 8-bit serial or parallel output

• Storage register with 3-state outputs

• Shift register with direct clear

• 100 MHz (typical) shift out frequency

• Input levels: – For 74HC595: CMOS level – For 74HCT595: TTL level

• Multiple package options

• Specified from -40 °C to +85 °C and from -40 °C to +125 °C

PIN Assignment:

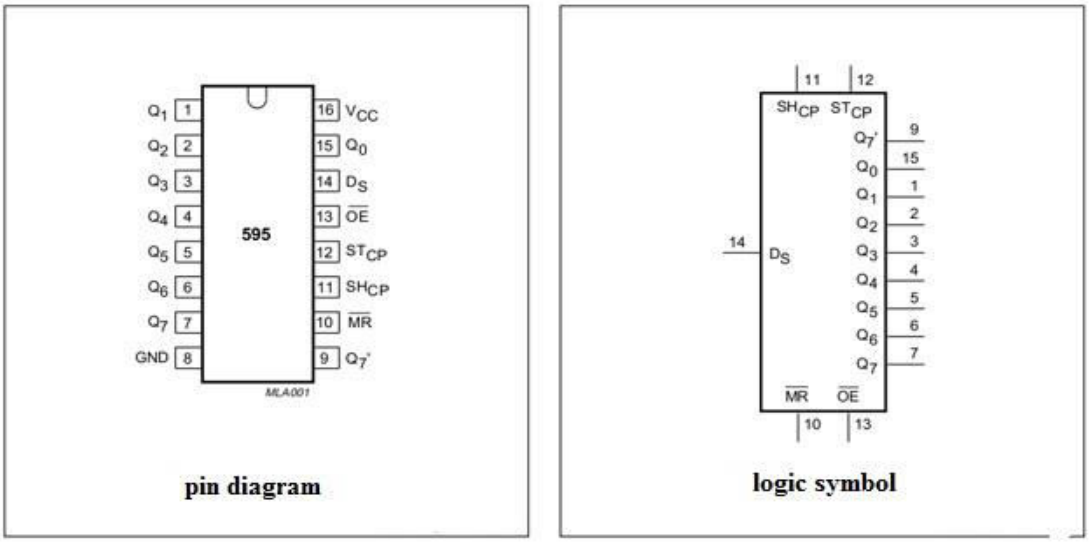


Figure 3.5.2 PIN Diagram and Logic symbol of 74HC595N

Logic Diagram:

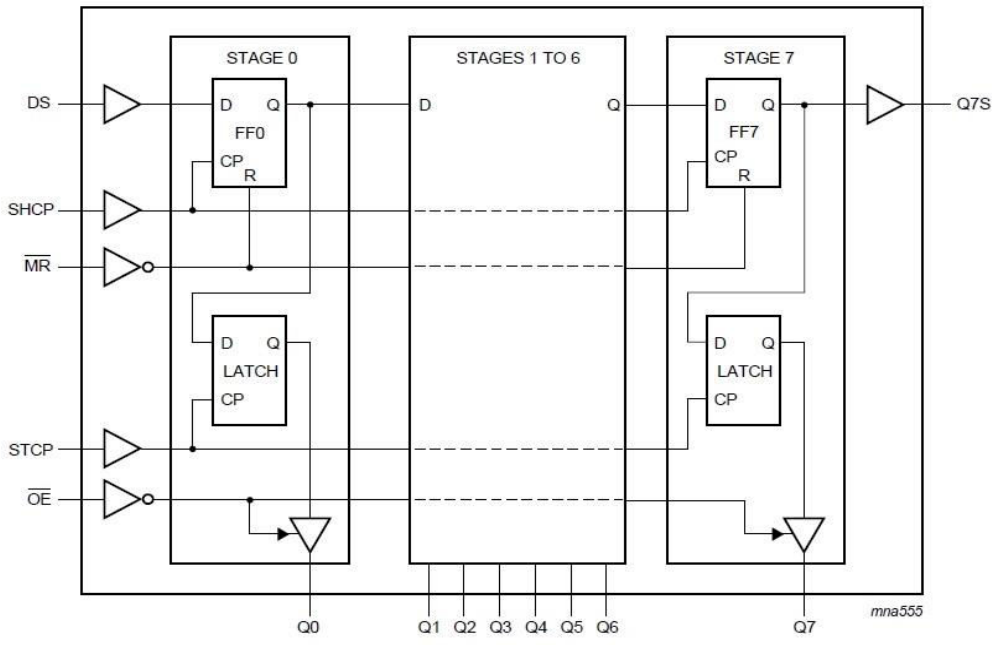
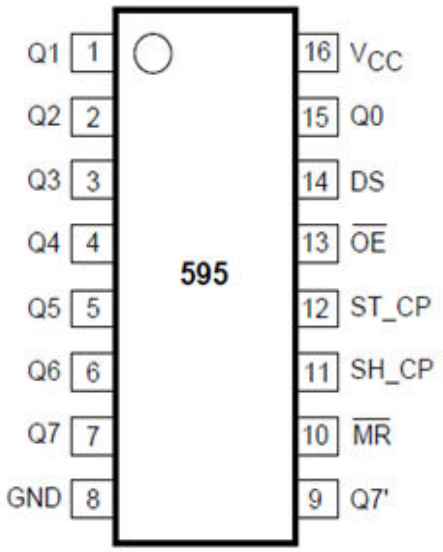


Figure 3.5.3 Logic Diagram of 74HC595N Shift Register

Pin Description:



PIN	SYMBOL	DESCRIPTION
1	Q1	parallel data output
2	Q2	parallel data output
3	Q3	parallel data output
4	Q4	parallel data output
5	Q5	parallel data output
6	Q6	parallel data output
7	Q7	parallel data output
8	GND	ground (0 V)
9	Q7'	serial data output
10	$\overline{MR}$	master reset (active LOW)
11	SH_CP	shift register clock input
12	ST_CP	storage register clock input
13	$\overline{OE}$	output enable (active LOW)
14	DS	serial data input
15	Q0	parallel data output
16	V_{CC}	positive supply voltage

Figure 3.5.4 Pin Description of 74HC595N

Table2. Functional Description of Shift Register

INPUT					OUTPUT		FUNCTION
SH_CP	ST_CP	$\overline{OE}$	$\overline{MR}$	DS	Q7'	Qn	
X	X	L	L	X	L	n.c.	a LOW level on $\overline{MR}$ only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6'	n.c.	logic high level shifted into shift register stage 0; contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6') appears on the serial output (Q7')
X	↑	L	H	X	n.c.	Qn'	contents of shift register stages (internal Qn') are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6'	Qn'	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

Note: L-LOW, H-HIGH, ↑-From LOW to HIGH, ↓-From HIGH to LOW, X-Don't Care, NC-No Change

Timing Diagram:

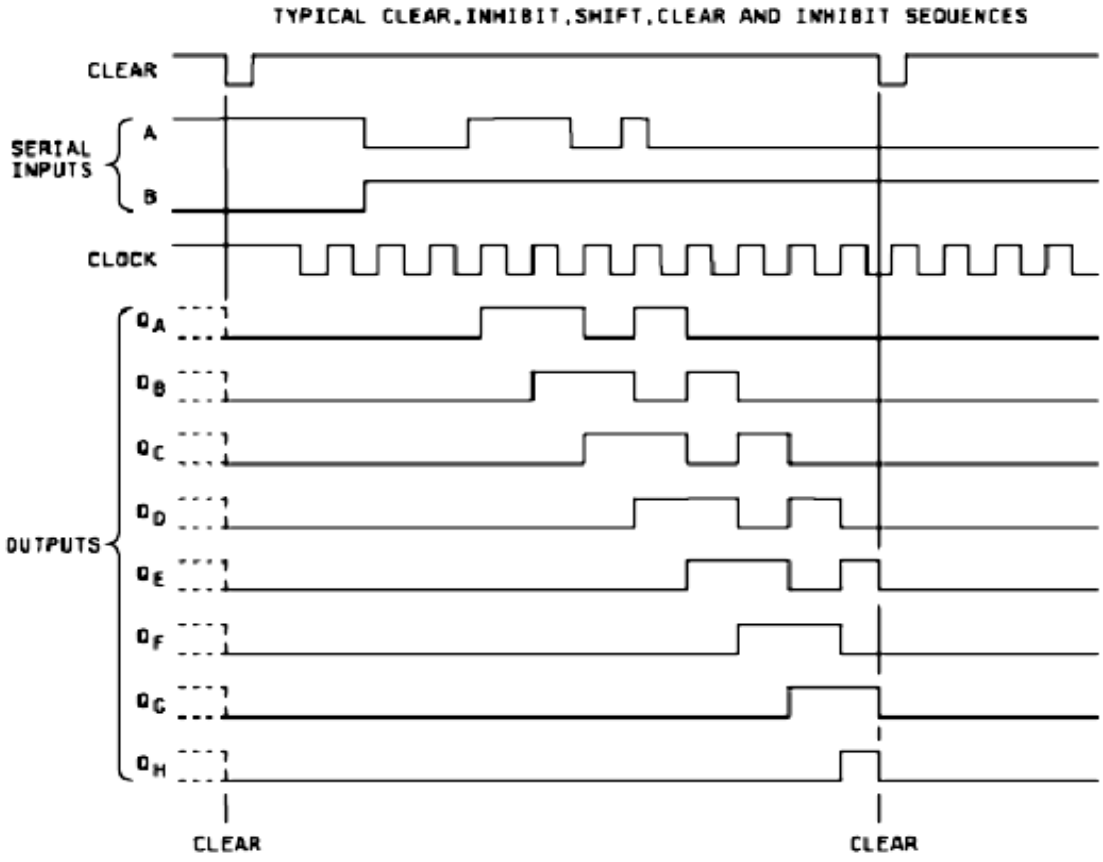


Figure 3.5.5 Timing Diagram of Shift Register

The shift register holds what can be thought of as eight memory locations, each of which can be a 1 or a 0. To set each of these values on or off, we feed in the data using the 'Data' and 'Clock' pins of the chip. The clock pin needs to receive eight pulses, at the time of each pulse, if the data pin is high, then a 1 gets pushed into the shift register, otherwise a 0.

When all eight pulses have been received, then enabling the 'Latch' pin copies those eight values to the latch register. This is necessary, otherwise the wrong LEDs would flicker as the data was being loaded into the shift register. You could attach this to a PWM capable ARM pin and use 'analog Write' to control the brightness of the LEDs. This pin is active low, so we tie it to GND.

Limiting Values:

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect the device reliability.

Table3. Limiting values of 74HC595N

Symbol	Parameter		Min.	Max.	Unit
V_{CC}	Supply Voltage		-0.5	7.0	V
V_{IN}	DC Input Voltage		-1.5 to V_{CC+}	1.5	V
V_{OUT}	DC Output Voltage		-0.5 to V_{CC+}	0.5	V
I_{IK}, I_{OK}	Clamp Diode Current			±20	mA
I_{OUT}	DC Output Current, per Pin			±35	mA
I_{CC}	DC VCC or GND Current, per Pin			±70	mA
T_{STG}	Storage Temperature Range		-65	+150	°C
P_D	Power Dissipation	PDIP(2)		600	mW
		SOIC Package Only		500	
T_L	Lead Temperature			+260	°C
ESD	Electrostatic Discharge Capability	Human Body Model, JESD22-A114		4000	V

Notes:
1. Unless otherwise specified all voltages are referenced to ground.
2. Power dissipation temperature derating, plastic package (PDIP);12mW/°C from -65 to +85°C.

Table4. Recommended Operating Conditions

Symbol	Parameter		Min.	Max.	Unit
V_{CC}	Supply Voltage		-0.5	7.0	V
V_{IN}	DC Input Voltage		-1.5 to V_{CC+}	1.5	V
V_{OUT}	DC Output Voltage		-0.5 to V_{CC+}	0.5	V
I_{IK}, I_{OK}	Clamp Diode Current			±20	mA
I_{OUT}	DC Output Current, per Pin			±35	mA
I_{CC}	DC VCC or GND Current, per Pin			±70	mA
T_{STG}	Storage Temperature Range		-65	+150	°C
P_D	Power Dissipation	PDIP(2)		600	mW
		SOIC Package Only		500	
T_L	Lead Temperature			+260	°C
ESD	Electrostatic Discharge Capability	Human Body Model, JESD22-A114		4000	V

Notes:
1. Unless otherwise specified all voltages are referenced to ground.
2. Power dissipation temperature derating, plastic package (PDIP);12mW/°C from -65 to +85°C.

3.6 Amplifier:

ULN2803 is a High voltage, high current Transistor Array IC used especially with Microcontrollers where we need to drive high power loads. This IC consists of an eight NPN Darlington connected transistors with common Clamp diodes for switching the loads connected to the output. This IC is widely used to drive high loads such Lamps, relays, motors etc. It is usually rated at 50v/500mA. This article brings out the working of ULN2803 IC and how to use it in a circuit.

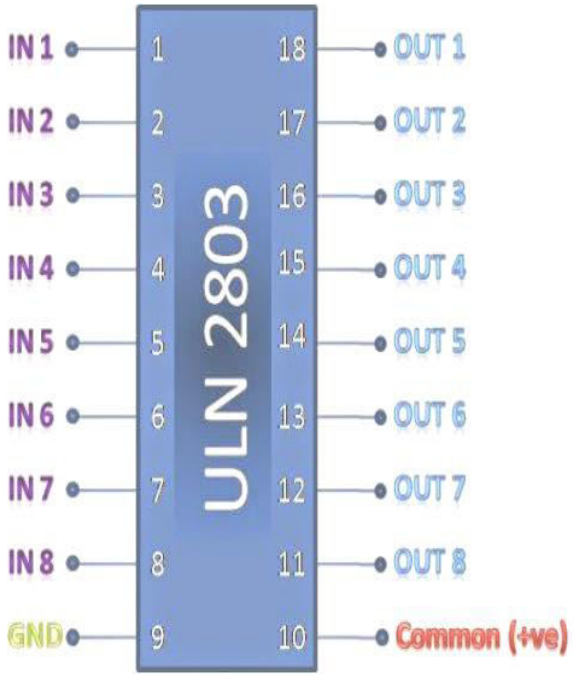


Figure 3.6.1 IC ULN2803 Pin Diagram

Purpose of ULN2803:

Most of the Chips operates with low level signals such as TTL, CMOS, PMOS, and NMOS which operates at the range of (0-5) v and are incapable to drive high power inductive loads. However this chip takes low level input signals (TTL) and uses that to switch/turn off the higher voltage loads that are connected to the output side.

3.6.1 Working of ULN2803 IC

The ULN2803 IC consists of eight NPN Darlington pair which provides the proper current amplification required by the loads. We all know that the transistors are used to amplify the current but here Darlington transistor pairs are used inside the IC to make the required amplification.

Darlington Pair

A Darlington pair is two transistors that act as a single transistor providing high current gain. In this pair the current amplified by the first transistor is further amplified by the next transistor providing high current to the output terminal.

When no base voltage is applied that when no signal is is given to the input pins of the IC, there will be no base current and transistor remains in off state. When high logic is fed to the input both the transistors begin to conduct providing a path to ground for the external load that the output is connected. Thus when an input is applied corresponding output pin drops down to zero there by enabling the load connected to complete its path.

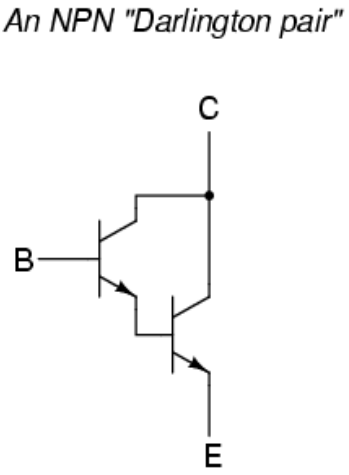


Figure 3.6.1.1 Darlington pair Transistor

CHAPTER 4
CONSTRUCTION & WORKING

4.1 Making of LED Array:

The dimensions of the cube will be 8x8x8 for a total of 512 LEDs. The whole structure will be based on a wood board.

In this project we will exploit the persistence of vision. This phenomenon is a feature of the human eye and is responsible for the illusion that a movie is not composed by individual still images. All the animations are based on this biological functioning of the human eye. If we can switch on and off the leds in a sufficiently quick time, (a few milliseconds) they will appear as if they were simultaneously on.

We use the persistence of vision because simultaneously switching the entire cube LEDs on would require a very high electric current (in ampere). Actually, if we consider that the high brightness leds of this project require about 20 mA, we can calculate that 512 LEDs would require 10.24 A. This high current is hard to manage. So, what we do? We switch on a cube layer at each time! In this way the current consumption will never exceed 1.28 A (20 mA x 64 leds), which is easily supplied by a good voltage transformer. Let us suppose that we want to make the cube appear as completely switched on. To achieve that we simply have to switch on the different layers one by one at high speed. To a human eye the cube will appear as completely lighted.

4.2 Controlling of LEDs:

In order to independently control each LED; we divide the cube into horizontal layers and vertical columns. All the LEDs of a horizontal layer will have the cathodic contact (-) in common. All the LEDs of a vertical column will have the anodic contact (+) in common.Overall, it will be necessary to control 8 cathodic contacts to select the layers and 64 anodic contacts to select the columns. The combination layers by column will select and switch a single LED.

The 8-bit shift registers:

The shift registers are composed by 1-bit memory cells connected one each other. At each clock impulse, they allow the bit flowing from a cell to the next-one. The registers used in this project are SIPO (serial input-parallel output) type. The data are charged one by one through the input bits and the output bits are simultaneously collected from the 8 outlets.

Power supply:

To power the cube and the control circuit it is necessary a power supplier with the following specifications:

-voltage: 5volts (stabilized)

- current: 2 ampere (4 is better)

4.3 Building of Cube

Step 1: Let us build the cube (first part)

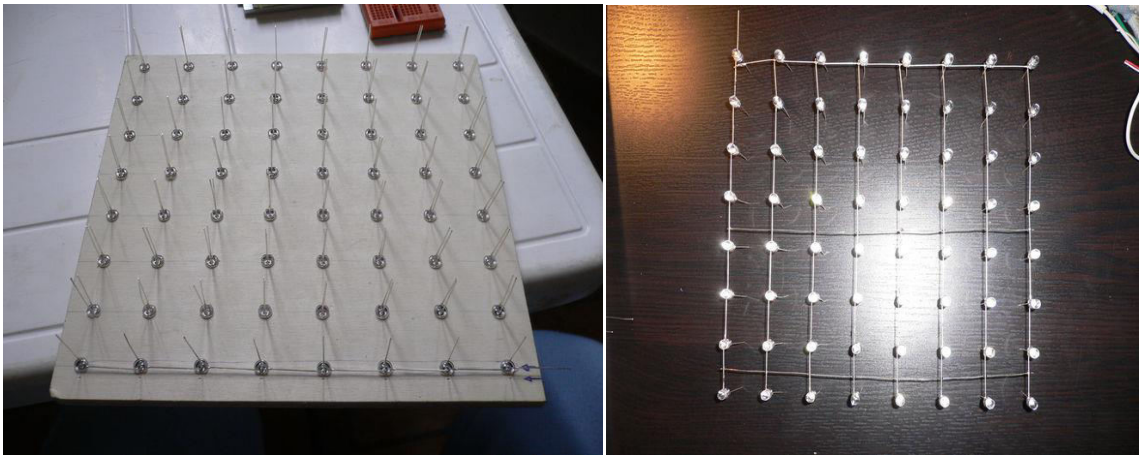


Figure 4.3.1 showing the construction of LED Cube

Needed tools:

- Soldering iron
- Soldering tin alloy
- Hot glue gun
- Jigsaw

Needed materials:

- 512 LEDs of your favourite color (we chose to take blue)
- 1 40 pin flat cable (to connect the IDE hard disk to the pc)
- 1 34 pin flat cable (to connect floppy disk to the PC)
- 1 prototype board
- 1 plywood board 20x20 cm, 8 mm thick
- Electric wire suitable for soldering (thin, flexible, and resistant)

Building:

Draw an 8x8 square grid (2.5 cm side) on the plywood board. At the crossings of the grid lines drill 64 holes with a mesh diameter as the LED size, generally 5 mm. This board will be the basis on which all the LEDs will be soldered. By this solid grid the LEDs will be evenly spaced and perfectly aligned. As explained in the intro, the cathodic contacts of each layer will be combining together. Insert the 64 LEDs in the board holes and bend the cathodic (-) terminal to obtain an interconnected grid.

Step 2: Building the cube with Blue LEDs

- Once the 8 layers have been made, we are going to connect them together. Place the LED layer on the plywood board, one layer at each time. Bend the anodic terminals (+) as shown in the picture. Bending the terminals is necessary to solder the entire anodic terminal on the column with a straight wire, without the need of wire bending to overcome the led.
- Once all the LED layers are ready, cut an electric wire into 64 pieces 20 cm long. Take the insulation plastic off these wire pieces and make them as straight as possible (surely you do not want to build the Pisa cube!). The straighter the wire the more squared will be the cube. The LED layers must be spaced 1.5 cm in vertical.
- Place the top layer on the plywood board with the LEDs corresponding to the holes. Solder all the previously prepared 64 wires to the anodic terminals of the

LEDs. After finished this layer, insert from the top the next layer and continue soldering the terminals.

Important tips:

To obtain an even spacing between the layers, prepare 4 small wood pieces 1.5 cm thick. You will insert them between the layers and they will help in holding the layer in place during soldering.

After soldering each layer, always test all the LEDs. If you discover some bad LEDs at the end of the project, it will be very hard to replace them. It is far better to spend some time in preliminary tests than completely un mounting the cube to fix a single LED. You will need some patience for these tests, but, as the cube building goes on, the structure will become more and steadier.

- Once all the layers have been connected, you have to bring all the cathodic terminals to the cube bottom. Remember that at this step the cube is up-side down. So, bend at 90° all the terminals of a layer and solder them to the wires, as already done for the columns. Now remove the plywood board and drill 8 small holes on it to let the cathodic wires pass through. Place the board over the cube, where the column wires (anodic contacts) go out vertically. Let these wires pass through the 64 holes. You can use the 1.5 cm wood pieces to hold the board over the LED cube. Cut a prototype board into 64 squares of 3 x 3 holes. These boards will serve to fix the terminals of columns and layers to the plywood board.
- Fill the holes of the plywood board with hot glue. For each hole, fill in the glue and then insert the prototype board square at the right place, with column wire in the central hole. When the glue is cold and solid, the wires will be steadily fixed in place. Remark: 8 prototype board squares will host both a layer wire (cathodic) and column wire (anodic). After all the 64 board square have been fixed, solder all the wires to the prototype boards and cut the exceeding wires. In this way the connection with the control cables will be easy.

- At this moment, the cube is almost finished. It only remains to solder the control wires to the LED terminals. However, this step will be carried out later, after defining the connections between the led terminals and the pins of control circuit.

Step 3: Build the control circuit

Short introduction

The control circuit basically consists of these following components:
- 8shift registers of 8 bits (total 64 bits) that enable / disable each column of the cube
- 1shift register connected to 8 NPN transistors to enable / disable each layer.
-2connectors with 40 and 34 pins for connection to the cube
-1connector with 2 pins for connection to the parallel port we connect the 8 NPN transistors to the shift register of layers. As we know, the LEDs of each layer have a common cathode (-), then we must enable them to connect the signal mass. The problem is that the output of the layer shift register have a high level signal (logic level 1 is equivalent to Vcc) in any pin enabled. We need a component that, given a signal at logic level 1, enables the passage of the current. This component is the transistor. It works as a switch: if the pin of the base is set to logic level 1, the pin collectors and emitter are connected together.

Tools needed:
- Soldering iron
- Soldering tin alloy
- Pincers

Materials needed:
- 9 shift register of 8 bit (I have used 74HC595)
- 9 slots for shift register
- 70 resistors 100 ohm
- 8 resistors 1500 ohm
- 1 prototype board
- 2 connectors with 40 and 34 pins for connecting to the cube
- 1 connector with 12 pins for connection to the parallel port

- 3 connectors with 2 pins for connecting the power switch, the LED power and power supply
- 1 adapter
- 1 button (power switch)
- 1 male connector for power supply
- 3 little cables with 2 wires, ended with a connector (like those used in PCs to connect the LEDs of the front panel to the mother board)

- Electric wire suitable for soldering (thin, malleable, and resistant at the same time)

Once you have got all the components, you can begin the construction of the control circuit.

First we should begin by placing the components above the base (or use the same disposition that I have chosen), to realize their actual size. Once you have positioned all the components, you should draw rectangles around them with a pencil to outline their actual space. So, when you re-position them back, you will be sure of their place. This is useful as you have to flip the base for soldering.

Components in this sequence to be shouldered:
1 - Slots for shift register
2 - Resistances of 100 ohms and 1.5 Kohm
3 - Transistors
4 - Connectors

Now, the connections should be between the main components. In a second stage, connect each resistor/transistor to the connectors for connecting the cube. Take note of all matches OUTPUT SHIFT REGISTER -> CONNECTORS PIN. This will be essential when we will solder the various wires to each terminal of the cube.

- For connections between 2 or more adjacent points use soldering iron to connect directly these points each chatter.
-should not insist too much with the iron on a single hole of the base, otherwise the pitch copper could fall off.

Step 4: Build the control circuit:

Connecting the wires to the cube:

Go back to the cube. Take the flat cables with 40 and 34 pin and cut it by removing the connect or from one side. We must separate for about 10 cm the wire to the side where we cut the connector. Those wires will be connected to different pins of cathodes and anodes of the cube according to the scheme that we made during the construction of the board. Each pin of an output shift register will be connected to a whole row of connectors in the order of pins (so not at random). In this way the first shift register controller the first row, the second controller second, etc of the 74 pins available (40 + 34) we will serve only 72 (64 columns, 8 layers), 2 remain not connected.

Build the cable to connect to the parallel port:
It is connected by a flat cable to the pins of mother board.
If you do not have a way to retrieve a cable and want to build it, this is the scheme of the parallel port pins:
You'll have to use the pin from D0 to D5 connecting like this:
D0 -> input shift register columns
D1 -> clear shift register columns
D2 -> clock shift register columns
D3 -> input shift register columns
D4 -> clear shift register layers
D5 -> clock shift register layers

D6 -> clear shift register layers
D7 -> clock shift register layers

CAUTION: If the pins are not connected as described above, the control software will not work. Construction of cables for the power switch, power LED and power supply: For all 3 cables you must only solder the components to the wire.

CHAPTER 5
ADVANTAGES

- **Single LED package:** By using a single LED package with two individual PN junctions will be used for the construction of this project. The choice to use a single package will greatly reduce the construction cost and time, the complexity of the project, and will increase the visual appeal of the cube by allowing for much better color blending and saturation.
- **Packaging of LEDs:** Commercially available LEDs are offered in many various package sizes, and formats. As with most electronic components, surface mount and through hole (leaded) package variants are available.
- **Surface mount device:** Surface mount device LEDs, commonly abbreviated SMD, and are flat chips that emit light in approximately an 180◦ field of view. These LEDs commonly have no lens, and are intended to provide illumination, rather than illuminate itself.
- **Quality of led:** LED quality can be measured in two very separate but important factors, efficiency, and color wavelength. During construction of LEDs, every effort is made to produce the highest quality product.
- **Quality Assurance:** During quality control, a process called binning occurs where each LED is tested and measured to ensure that it matches the quality standards. The higher the efficiency and the more accurate the color wavelength, the higher a price the LED will command.
- LEDs are inherently temperature and voltage sensitive devices. LEDs, when biased at a higher temperature emit less light.
- In larger LEDs, this effect causes drastic reductions in lumen output. Due to the low power nature of the LEDs used in this design, and the operating conditions of the project, thermal efficiency should not
- To explore programming user iterative games.
- To achieve the light and music shows.
- Creating an animation visually.

CHAPTER 6
APPLICATIONS

Animation Features:

The animation feature set will act as a visual representation of the power and utility of the animation component of the software. A rich animation feature set will be directly correlated with an audience's immediate perception of project success, and for this reason, the feature set must be robust and vast. The following animations are all generated through the Programmable Animation Framework API. The animation features will be available for selection from the Animation Controller with animation specific variables exposed to change the behavior and appearance of the animation. Each specific animation feature may require additional data structures to be maintained to support each individual animation in order to produce the intended features.

Raining Voxels:

A raining effect can be achieved from the LED cube by having fully lit voxels appear to drop from the top to the bottom of the cube. This is implemented simply by decreasing the color intensity of the previous layer by a constant factor before adding in the new droplets. With an appropriate fade rate, the improved raining voxel animation can mimic the famous falling characters from the movie The Matrix.

Fireworks:

The illusion of fireworks can be created by combining some functions of previously described animation features. . A firework animation consists of a firework shot rising up a column of the LED cube. The firework explosion is created by generating an expanding hollow sphere of voxels centered at the peak of the shot. After the explosion has completed, gravity pulling the exploded particles to the ground is implemented similarly to how falling raindrops are implemented. The firework animations will feature multiple explosion colors, and the falling exploded particles will turn to resemble an orange ember that fades at it falls to the ground.

Scrolling Figures:

The LED cube can display scrolling figures similar to a scrolling LED banner in multiple manners that complement as well as extend the common features of the traditional 2D LED banner. The scrolling figure is represented internally as an array of the same dimensions as the LED cube with the one exception that the axis of motion is extended to support the length of the scrolling animation. The implemented scrolling figures features recreate but also extend the capabilities of similar 2D LED displays, highlighting the capabilities of the 3D LED cube.

Rotating Figures:

A solid figure can be rotated within the LED cube to provide an additional feature as well as to possibly add support for other features. Incremental changes of rotation to the figure are made directly to the orientation vector and the displayed figure is recomputed from this orientation vector at every iteration to prevent the unintended walking or degradation of voxels from the original figure.

Mathematical Patterns:

Seemingly complex patterns and visually appealing animations can be created by defining math functions in 3-dimensional space and using the cube to visualize the result. The generation of sinusoidal functions with variable amplitude, frequency, and phase across dependent on both x and y and possibly time can create a vast set of animations. Additional animations can be created by expanding or contracting a sphere or other volumes from various coordinates within the LED cube. The numbers of possible animations that utilize combinations of mathematical functions are limited only by the unit display resolution of the LED cube.

Gravity Simulation:

As an extension of the rotating figures rotating figures animation, an interesting feature can be added by adding an accelerometer on the LED cube to provide orientation data. An additional effect would be to allow for an open upper face, and to allow the

water level to decrease as water poured from the top of the cube. For this feature to be implemented, it requires the addition of an extra physical component to the LED cube and a channel for communication back to the animation software.

Audio Visualization:

A neat and interactive animation feature to include is an audio visualizer. By taking an audio sample as input and applying a Fast Fourier Transform to the waveform, the waveform will decompose to a spectrum of frequencies and their associated amplitudes. This animation gives a viewer the ability to visualize the audio signal they are hearing and identify interesting features of the audio signal that appear in the spectrum across time. The audio signals can be input in both real-time through a microphone peripheral or from a saved audio file input to the animation software.

CHAPTER 7
CONCLUSION

The cube overall was a success. As a team, a great deal was learned about the process of creating hardware and correctly interfacing with the microcontroller and software. The process involved moving between the hardware, microcontroller and software to ensure it all combines into a successful result. The functions were successfully implemented that demonstrate control over the cube lighting and mastery of the principles behind the software to hardware interface. This gives a current overview of how our team will create a visual display that can be used across campus to showcase the talents of the ECE department. The detailed background search relevant to this topic will help to ensure that our team possesses the necessary knowledge to effectively tackle this problem. The restraints and objectives of this project allowed for several potential solutions for today's use. The best solution was selected using a decision matrix and discussed in great detail above. This was determined to best meet the economic and qualitative requirements for this project. The testing plan will help ensure that this design meets all the desired requirements. Overall, this plan will help guide the creation of a 3D LED display that will be readily available and used for years to come.

CHAPTER 8
FUTURE SCOPE

The completion of this project culminated in a clear and precise direction moving toward the completion of a final prototype of the 3D LED cube. The design phase was where the individual skills of the group members were put to the test - creating a highly detailed and formulated design plan to achieve the required specifications for the completed prototype.

When the design was completed, we had a clear and functional outline: a software interface to send instructions to an on-board control PCB, containing an embedded processor to control the driver board containing LED drivers and MOSFETs which in turn modulated the current to each individual LED with a high level of speed and precision.

For future designs, a better approach would be to use different PMOS chips with a greater range of Vdd that maintains the desired output performance. Another approach would be to reduce the hardware required by using different control devices than decoders and PMOS. This design required 14 different integrated chips.

This resulted in a dozens of wires and a difficult structure to build and debug. Another approach would be to use LED drivers, each of which can drive up to 16 LEDs. LED drivers work by lighting LEDs by writing a byte to the driver. This would allow for greatly varied and more complex lighting designs as well as less hardware and wires.

Variation in size can give better resolution. Use of Bit Angle Modulation can improve brightness of LEDs. Full integration with MATLAB to plot 3D interactive graphs. Use of RGB LEDs.

Possibilities for future expansion:

- Low-level (as opposed to application-level) pulse width modulation brightness control of the LEDs, with corresponding intensity variations on the SVGA output
- Implementation of the Bresenham line drawing algorithm to allow projection angles other than 45°.
- Display of 3D data stored on a Compact Flash card - may be used as initial conditions for cellular automata
- True 3D rendering (rather than orthographic projection) of the cube on SVGA output, rotatable in real time by user
- Modification of the cube to increase resolution/enhance visibility – use larger lattice spacing or smaller LEDs (ideally SMT, but this would require a new construction technique.

REFERENCES

1. Bergesen, Joseph D.; Tähkämö, Leena; Gibon, Thomas; Suh, Sangwon (2016). "Potential Long-Term Global Environmental Implications of Efficient Light-Source Technologies". Journal of Industrial Ecology. 20 (2): 263.
2. Damir, B (2012). "Longevity of light bulbs and how to make them last longer". RobAid. Retrieved 10 August 2015.
3. Whelan, M. (2013). "Arc Lamps". Edison Tech Center. Retrieved November 22, 2014.
4. Nakamura, S.; Mukai, T.; Senoh, M. (1994). "Candela-Class High-Brightness InGaN/AlGaN Double-Heterostructure Blue-Light-Emitting-Diodes". Appl. Phys. Lett. 64 (13): 1687. Sakr, Sharif. "ARM co-founder John Biggs". Engadget. Retrieved 23 December 2011. [...] the ARM7-TDMI was licensed by Texas Instruments and designed into the Nokia 6110, which was the first ARM-powered GSM phone.
5. "Remembering the Sega Dreamcast".29 September 2009.
6. Shiro Hagiwara; Ian Oliver (1999). "Sega Dreamcast: Creating a Unified Entertainment World".. IEEE Micro.. 19 (6): 29–35.
7. Rood, George. "Music Concerns Seek New Volume With Amplifier" New York Times. Retrieved 23 February 2015.
8. Amplifier Fills Need in Picture: Loud Speaker Only Method Found to Carry Directions During Turmoil". Los Angeles Times.
9. This table is a "Zwicky box", in particular, it encompasses all possibilities.
10. bitsavers.org, DataPoint 3300 Maintenance Manual, December 1976.
11. Flowers, Thomas H.(1983), "The Design of Colossus".Annals of the History of Computing, 5 (3): 246
12. ‘Introduction, Architecture, Interfacing of ARM’ by Raj Kamal
13. ‘ARM architecture reference manual’ by David Seal
14. ‘ARM Microcontroller Interfacing: Hardware and Software’ by Warwick A. Smith
15. ‘ARM Microcontrollers for Beginners’ by Bert Van Dam

16. https://en.wikipedia.org/wiki/Amplifier
17. http://www.electronics-tutorials.ws/amplifier/amp_1.html
18. http://www.futureelectronics.com/en/drivers/led-driver.aspx
19. https://en.wikipedia.org/wiki/Shift_register
20. https://www.allaboutcircuits.com/textbook/digital/chpt-12/introduction-to-shift-registers/
21. https://www.slideshare.net/ECprojects/led-cube-slidesecee.colorado.edu/~ecen4610/expos11/STONECAP_PDR.ppt
22. students.iitk.ac.in/eclub/assets/documentations/summer13/LED%20cube.pdf
23. http://www.instructables.com/id/3D-LED-Cube
24. http://www.instructables.com/id/How-to-build-a-8x8x8-led-cube-English-version
25. https://developer.android.com/guide/topics/connectivity/bluetooth.html
26. https://www.bluetooth.com/specifications/adopted-specifications
27. https://www.itead.cc/wiki/Serial_Port_Bluetooth_Module_(Master/Slave):HC-05
28. http://www.instructables.com/id/Led-Cube-8x8x8/?ALLS TEPS
29. http://www.hownottoengineer.com/projects/lc.html

APPENDICES

APPENDIX-1:

Flash Magic Software Tool:

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology. Flash Magic is Windows software from the Embedded Systems Academy that allows easy access to all the ISP features provided by the devices. These features include:

• Erasing the Flash memory (individual blocks or the whole device)

• Programming the Flash memory

• Modifying the Boot Vector and Status Byte

• Reading Flash memory

• Performing a blank check on a section of Flash memory

• Reading the signature bytes

• Reading and writing the security bits

• Direct load of a new baud rate (high speed communications)

• Sending commands to place device in Boot loader mode

Flash Magic provides a clear and simple user interface to these features and more as described in the following sections.

Under Windows, only one application may have access the COM Port at any one time, preventing other applications from using the COM Port. Flash Magic only obtains access to the selected COM Port when ISP operations are being performed. This means that other applications that need to use the COM Port, such as debugging tools, may be used while Flash Magic is loaded.

Minimum Requirements:

- Windows NT/2000/XP/Vista/7/8 32-bit or 64-bit.
- COM Port or Ethernet interface.
- 10Mb Disk Space.

Main Window:

The following is a screenshot of the main Flash Magic window. The appearance may differ slightly depending on the device selected.

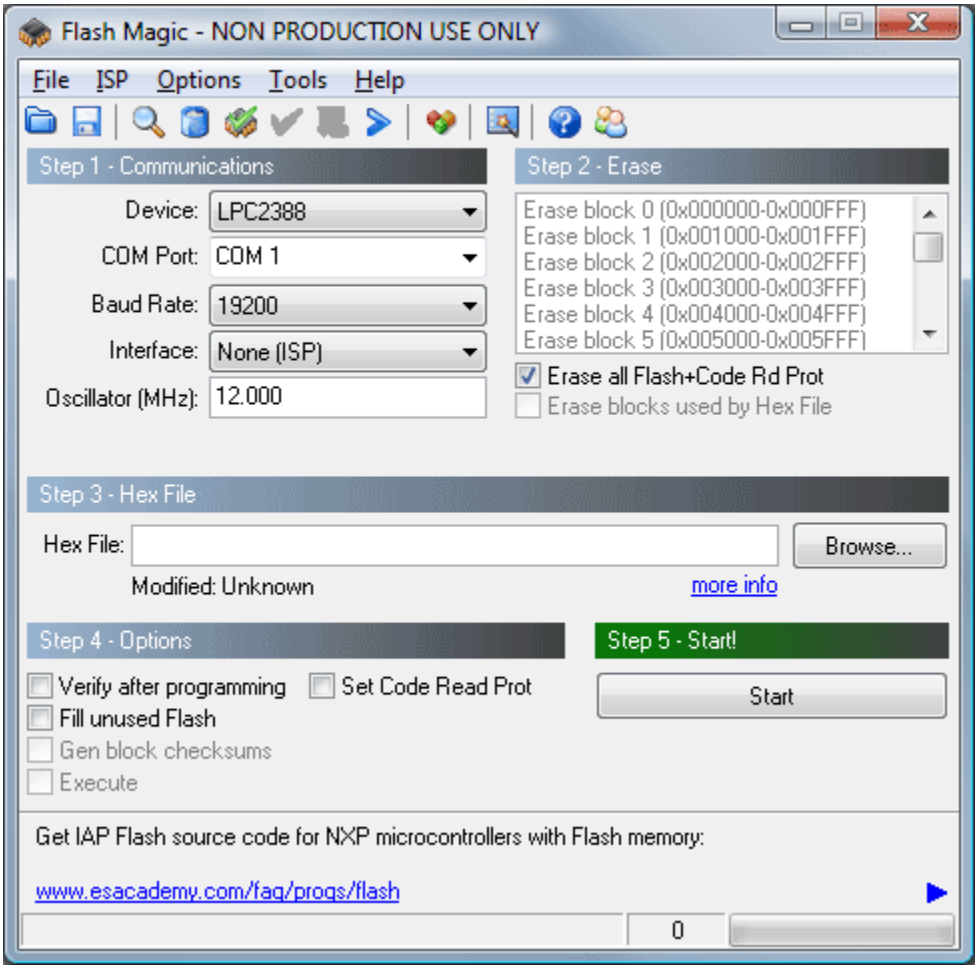


Figure: Main Flash Magic Window

The window is divided up into five sections. Work your way from section 1 to section 5 to program a device using the most common functions. Each section is described in detail in the following sections.

At the very bottom left of the window is an area where progress messages will be displayed and at the very bottom right is where the progress bar is displayed. In between the messages and the progress bar is a count of the number of times the currently selected hex file has been programmed since it was last modified or selected.

Just above the progress information Embedded Hints are displayed. These are rotating Internet links that you can click on to go to a web page using your default browser. If you wish to quickly flick through all the hints then you can click on the fast forward button: 

Menus:

There are five menus, File, ISP, Options, Tools and Help.

- The File menu provides access to loading and saving Hex Files, loading and saving settings files and exiting the application.
- The ISP menu provides access to the less commonly used ISP features.
- The Options menu allows access to the advanced options and includes an item to reset all options.
- The Tools menu provides features that support the operation and use of Flash Magic.
- The Help menu contains items that link directly to useful web pages and also open the Help About window showing the version number.
- The Loading and Saving of Hex Files and the other ISP features are described in the following sections.

Tooltips:

Throughout the Flash Magic user interface extensive use has been made of tooltips. These are small text boxes that appear when you place the pointer over something and keep it still for a second or two.

Saving Options:

The options in the main window and the Advanced Options window are automatically saved to the registry whenever Flash Magic is closed. This removes the need for an explicit save

operation. When Flash Magic is restarted the main window and the Advanced Options window will appear as you left it, so you do not have to repeatedly make the same selections every time you start the application. If you wish to reset the options to the original defaults then choose Reset from the Options menu.

Five step programming:

For each step there is a corresponding section in the main window as shown in above figure

Step 1 – Connection Settings

Before the device can be used the settings required to make a connection must be specified. For Communications, select the PC port with the serial cable connected from the board as the COM Port, specify 9600 as the **Baud Rate**, and select the LPC2148 as the **Device**.

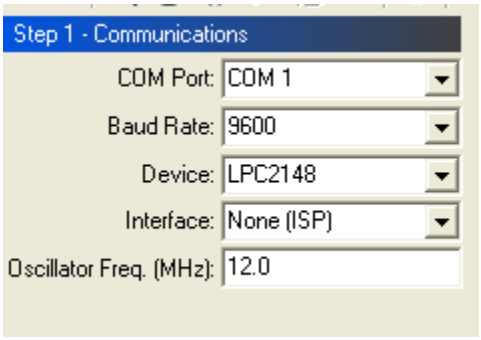


Figure: Window showing Step 1

Step 2 - Erasing

This step is optional, however if you attempt to program the device without first erasing at least one Flash block, then Flash Magic will warn you and ask you if you are sure you want to program the device.

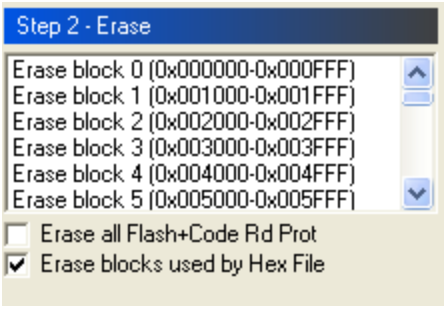


Figure: Window showing Step 2

Step 3 – Selecting the Hex File

This step is optional. If you do not wish to program a Hex File then do not select one. You can either enter a path name in the text box or click on the Browse button to select a Hex File by browsing to it. Also you can choose Open… from the File menu.

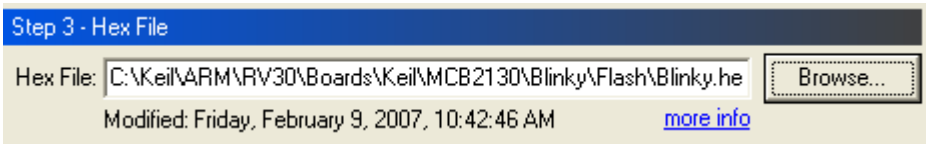


Figure: Window showing Step 3

If the Hex File is modified, you do not have to reselect it in Flash Magic. Every time the Hex File is programmed it is first re-read from the location specified in the main window. The date the Hex file was last modified is shown in this section. This information is updated whenever the hex file is modified. The hex file does not need to be reselected.

Step 4 – Options

Flash Magic provides various options that may be used after the Hex File has been programmed.

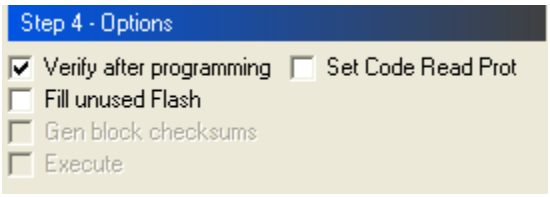


Figure: Window showing Step 4

Step 5 – Performing the Operations

It contains a Start button. Clicking the Start button will result in all the selected operations in the main window taking place.

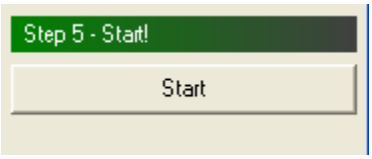


Figure: Window showing Step 5

Once started progress information and a progress bar will be displayed at the bottom of the main window. In addition the Start button will change to a cancel button. Click on the cancel button to cancel the operation. If you cancel during erasing all the Flash, it may take a few seconds before the operation is cancelled.

Reset and Execute:

Selecting the Reset item on the ISP menu will cause a reset command to be sent to the device. Depending on the hardware and the status byte, the devices with either reset and execute code or reset to the Boot loader. If the Reset command is sent after successfully programming the device then the reset will execute the downloaded code. If the Reset command is sent after erasing the device then the device will reset to the Boot loader.

Selecting the Execute item on the ISP menu will cause Flash Magic to program the Boot Vector to the default and the Status Byte to zero (for devices that have this feature), followed by sending a reset command to the device. This will force any downloaded code to be executed.

KEIL Software Tool:

Keil development tools for the ARM Microcontroller Architecture support every level of software developer from the professional applications engineer to the student just learning keil.

The industry-standard Keil C Compilers, Macro Assemblers, Debuggers, Real-time Kernels, Single-board Computers, and Emulators support all ARM derivatives and help you get your projects completed on schedule.

Figure: Keil C51 9.51 Compiler-uVision 4 IDE Full Version Crack

The Keil Development Tools are designed to solve the complex problems facing embedded software developers.

- When starting a new project, simply select the microcontroller you use from the Device Database and the µVision IDE sets all compiler, assembler, linker, and memory options for you.
- Numerous example programs are included to help you get started with the most popular embedded ARM devices.
- The Keil µVision Debugger accurately simulates on-chip peripherals (I²C, CAN, UART, SPI, Interrupts, I/O Ports, A/D Converter, D/A Converter, and PWM Modules) of your Microcontroller device. Simulation helps you understand hardware configurations and avoids time wasted on setup problems. Additionally, with simulation, you can write and test applications before target hardware is available.
- When you are ready to begin testing your software application with target hardware, use the MON51, MON390, MONADI, or FlashMON51 Target Monitors, the ISD51 In-System Debugger, or the ULINK USB-JTAG Adapter to download and test program code on your target system.
- The ARM Compiler is specifically designed to optimize software running on ARM processors. It is the result of 20 years of development alongside the ARM Architecture. The ARM Compiler tool chain incorporates a highly optimizing C/C++ compiler, assembler, linker and libraries for embedded software development.

- Billions of devices containing software built with ARM tools have shipped to date, covering all embedded markets. Whether your industry is **avionics**, consumer electronics, mobile, industrial automation, automotive or medical, the ARM compiler can play a key role in optimizing your code.

Superior Performance:

Figure: performance

- The highly efficient ARM Compiler invokes powerful optimization techniques such as loop unrolling, function in lining, idiom recognition, and architecture-specific instruction scheduling.

Superior code size Reduction:

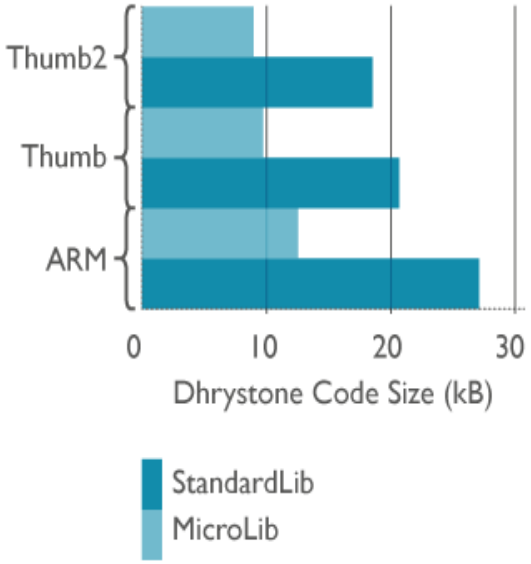


Figure: Graphical Analysis of code size Reduction

- The ARM Compiler has a long embedded heritage, where memory space is a prized commodity. It incorporates techniques that can reduce your application footprint by up to 30% compared to other compilers. The ARM Compiler reduces the best code size by up to 5% compared to the RVDS 4.0 compiler.

Figure: Code Size

- In addition, the ARM Compiler includes an optional MicroLib C library for Cortex-M series microcontrollers, which provides up to a 50% reduction in code size compared to the full standard C library.
- The MicroLib C library provides a completely C-based development environment without the need to revert to assembly language - even for interrupt service routines. This removes the need for specific knowledge of the ARM architecture.

Figure: Value

Superior value:

Whether you are creating a modern application targeting a Cortex-M7 microcontroller or rebuilding a 10-year-old library targeting the ARM7TDMI®, you only need a single license. An MDK-ARM license works with older compiler versions, making modifications to legacy code easy.

Libraries:

The ARM Compiler features full support for C90, C99 and C++2003 with optimized routines for ARM and Thumb-2 which can greatly improve the performance of your code.

Safety:

The ARM Compiler Qualification Kit consists of a significant body of supporting evidence derived from our development process including defect reports, C90/C99 test reports and a compiler safety manual, so that you can increase your confidence in providing a justification argument for compliance.

Figure: Safety

To complement this, we also provide Extended Maintenance and Support to protect your safety-critical projects against tool chain obsolescence. Initially supported in ARM Compiler v4.04, Extended Maintenance and Support is intended to maximize the stability of a fixed branch of the compiler tool chain by providing bug fixes and patches for a minimum of 5 years.

BT Voice Control for ARM:

This application uses android mobiles internal voice recognition to pass voice commands to your device.This will pair with serial modules and sends in the recognized voice as a string.

For example if u say “hello” the android phone will return a string *hello# to your Bluetooth module *and# indicates the start and stop bits can be used with any micro controller which can handle strings.

Example Platforms: Arduino, ARM, PICAXE, MSP430, 8051 based and many other processors and controllers.

It's really easy and quick to add voice control to your arduino project. Whether it is home automation or door lock, or robots, voice control could be one eye catching feature in an arduino project. In this tutorial I'll show you how to voice control arduino projects without voice recognition shield. We'll be using a HC-05 Bluetooth module. We'll connect an Android device with HC-05 Bluetooth. Android Phone will convert voice into string of data using Google voice recognition software. This string of data will be sent to HC-05 Bluetooth module and then to arduino Uno. After that, Arduino decodes and process it.

On the Android device we use an Application called as AMR_Voice to convert voice into string of data and send it over bluetooth. You can download this app from Google Play store.

Hardware:

1.Serial Port Bluetooth Module(TTL)

2.Arduino Microcontroller Board

3.MCU serial communication

4.Can search for Bluetooth low energy devices (nothing more)

Software features:

1. Search for Bluetooth devices, and displays the class and RSSI (signal strength);

2. The use of serial communication, receiving and sending data;

3. Can be set to ASCII and HEX input and output mode;

4. The data results can be saved to the SD card (/sdcard/Bluetooth spp pro/...).

5. Can search for Bluetooth low energy devices (nothing more)

This tool has three modules:

1. Byte stream mode: the basic input-output model;

2. Keyboard mode: Can customize the output value of 12 buttons; each button has three states (respectively: Down | Long-press | Up), each state can send commands event.

3. Command Line: Set the command terminator for communication debugging.

If the connected Bluetooth device is not paired, the system will automatically prompt you for pairing. Bluetooth pairing is successful, try to connect again.

This can only connect Bluetooth serial module devices, Bluetooth devices are generally used for MCU serial communication.

System using the Bluetooth pairing means: [menu-> Settings -> Wireless and Network -> Bluetooth Settings], open the Bluetooth feature, and to search for Bluetooth devices to pair, paired with a device only once.

System Configuration (includes keyboard mode button settings) file can be found in the SD card Bluetooth spp pro directory. You can back up the configuration file, or copy the configuration file to terminal equipment, covering his profile to complete recovery.

Special cases:

Non-normal end of the Bluetooth function, may lead to not be able to connect Bluetooth devices such as this is the case, please restart the phone can often return to normal.

APPENDIX-2 (SAMPLE CODE):

```
#include "LPC214x.h"
#include <stdio.h>
#include "uart.h"
#include "uart0.c"
#include"uart1.c"
#include<string.h>
#include "lcd.c"
#define  latchPin1     0x20000000
#define clockPin1     0x40000000
#define dataPin1      0x10000000
#define latchPin2     0x00000020
#define clockPin2     0x00000040
#define dataPin2      0x00000010
int kl;
int x,y,kl=0;
int x1,i,j,k;
int y1,kl1=0,stat=1;
int index;
char inData[800],ch;
int refreshCount=0;
//unsigned char inputString[800];
unsigned int is,js,ltr,ct,count=0,mm=0;
int alphabets[8][64] =
 {
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},// sp
```

```
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1},//!
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1}, // "

void long delay(unsigned int v)

{

 int i,j;

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

 for(j=0;j<=10000;j++);

}

void shiftOut1(int  *myDataOut)

{

   int i=0;

  int pinState;

  IOCLR1=dataPin1;

  IOCLR1=clockPin1;

  //digitalWrite(myDataPin, 0);

  //digitalWrite(myClockPin, 0);

  for (i=0; i<=63; i++)

 {

  IOCLR1=clockPin1;

   // digitalWrite(myClockPin, 0);

    if ( myDataOut[i] & (0x01) )

{

 pinState= 0;

IOCLR1=dataPin1;

}

else
```

```
{
pinState= 1;
IOSET1=dataPin1;
}
 //   digitalWrite(myDataPin, pinState);
 IOSET1=clockPin1;
 IOCLR1=dataPin1;
  //  digitalWrite(myClockPin, 1);
  //  digitalWrite(myDataPin, 0);
   }
  //stop shifting
  IOCLR1=clockPin1;
 // digitalWrite(myClockPin, 0);
}
void shiftOut2(char  *myOut)
{
 int i=0;
 int pinState;

 IOCLR0=dataPin2;
 IOCLR0=clockPin2;
// digitalWrite(myDataPin, 0);
// digitalWrite(myClockPin, 0);
for (i=63; i>=0; i--)
{
IOCLR0=clockPin2;
//digitalWrite(myClockPin, 0);
```

```
if ( myOut[i])
{
pinState= 1;
IOSET0=dataPin2;
}
else
{
pinState= 0;
IOCLR0=dataPin2;
}
//digitalWrite(myDataPin, pinState);
IOSET0=clockPin2;
IOCLR0=dataPin2;
//  digitalWrite(myClockPin, 1);
// digitalWrite(myDataPin, 0);
}
//digitalWrite(myClockPin, 0);
IOCLR0=clockPin2;
}
void RefreshDisplay()
{
char out[8]={0,0,0,0,0,0,0,0};
int row=0;
for (row = 0; row < 8; row++)
{
if(row>0)
out[row-1]=0;
```

```
out[row]=1;
IOCLR0=latchPin2;
// digitalWrite(latchPin2, LOW);
//Hold latchPin LOW for as long as we're transmitting datashiftOut2(out);
//Transmit data IOCLR1=latchPin1;
//digitalWrite(latchPin1, LOW);
//Hold latchPin LOW for as long as we're transmitting datashiftOut1(alphabets[row]);
IOSET1=latchPin1;
//digitalWrite(latchPin1, HIGH);
IOSET0=latchPin2;
// digitalWrite(latchPin2, HIGH);
// delayMicroseconds(1);
//delay(100);
}
}
```