



Jaypee University of Information Technology  
Solan (H.P.)

**LEARNING RESOURCE CENTER**

Acc. Num. SP06145 Call Num:

**General Guidelines:**

- ◆ Library books should be used with great care.
- ◆ Tearing, folding, cutting of library books or making any marks on them is not permitted and shall lead to disciplinary action.
- ◆ Any defect noticed at the time of borrowing books must be brought to the library staff immediately. Otherwise the borrower may be required to replace the book by a new copy.
- ◆ The loss of LRC book(s) must be immediately brought to the notice of the Librarian in writing.

Learning Resource Centre-JUIT



SP06145



**SOFTWARE CONTROLLED ROBOTIC  
ARM MOVEMENT**

**A DISSERTATION**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE AWARD OF THE DGREE  
OF**

**BACHELORE OF TECHNOLOGY  
IN COMPUTER SCIENCE AND INFORMATION  
TECHNOLOGY ENGINEERING**

**BY**



**NISHANT RANA- 061267**

**ATTIN KUMAR VERMA-061412**



**DEPARTMENT OF COMPUTER SCIENCE AND  
INFORMATION TECHNOLOGY**

**JAYPEE UNIVERSITY OF INFORMATION  
TECHNOLOGY  
WAKNAGHAT, SOLAN-173215(HP), INDIA**

**DEC 2010**

## CERTIFICATE

This is to certify that the work entitled, "SOFTWARE  
CONTROLLED ROBOTIC ARM MOVEMENT  
" submitted by NISHANT  
RANA 061267, AHIR KUMAR VERMA 061412

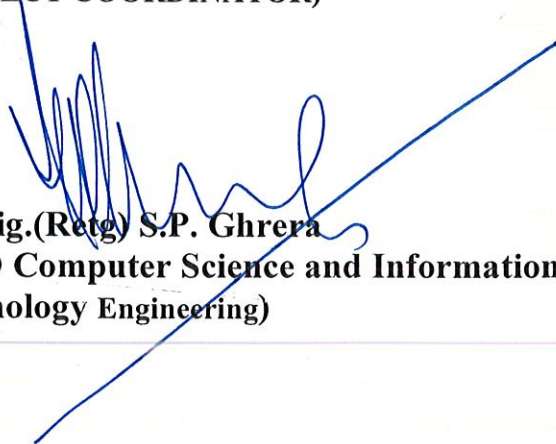
in partial fulfillment for the award of degree of bachelor of  
technology in COMPUTER SCIENCE

-----engineering  
of Jaypee University of Information and Technology has been  
carried out under my supervision. This work has not been  
submitted partially or wholly to any other University or Institute  
for the award of this or any other degree or diploma.

*Ravi Rastogi*

**MR. RAVI RASTOGI**

**(PROJECT COORDINATOR)**



**Brig.(Retg) S.P. Ghreya**  
**(HOD Computer Science and Information**  
**Technology Engineering)**

## ACKNOWLEDGMENT

It has been an amazing experience working on the project SOFTWARE CONTROLLED ROBOTIC ARM MOVEMENT.

While working on a project that involves in depth knowledge of language

proves that we as a student have imbibed in us the quality that an engineer should have.

We thanks Mr. **Ravi Rastogi** , our project guide ,who has always encouraged us to put in our best efforts and deliver a quality and professional output .We really thank him for his time and efforts. We are deeply indebted to all those who provided reviews and suggestions for improving the materials and topic covered in our package, and we extend our apologies to anyone we may have failed to mention .

**Thank you**

**Nishant Rana**

**061267**

Department of Computer Science Engineering

Jaypee University of Information Technology, Solan

**Attin Kumar Verma**

**061412**

Department of Information Technology Engineering

Jaypee University of Information Technology, Solan



## TABLE OF CONTENTS

LIST OF FIGURES	PAGE NO.
Fig . 1.1 (Pin diagram).....	16
Fig. 2.1 (Block Diagram).....	24
Fig. 3.1 ( Diode) .....	28
Fig. 3.2 (Diode DC Characteristics) .....	29
Fig. 3.3 (Rectifier Characteristics).....	30
Fig. 3.4 (Full Wave Rectifier) .....	32
Fig. 3.5 (Transformer) .....	34
Fig. 3.6 (Resistor).....	36
Fig. 3.7 (Resistor Working) .....	39
Fig. 4.1 (AVR MCU Architecture).....	41
Fig. 4.2 (AVR Status Register) .....	45
Fig. 4.3 (AVR CPU General Purpose register).....	48
Fig. 4.4 (Stack Pointer).....	50
Fig. 5.1 (Program Memory map) .....	55
Fig. 5.2 (Data Memory map).....	57
Fig. 5.3 (EPROM Read /Write Access) .....	59
Fig. 5.4 (System Clock) .....	61
Fig. 5.5 (Crystal Oscillator).....	64
Fig. 5.6 (External Clock) .....	66
Fig. 6.1 (78xx).....	69
Fig. 6.2 .....	69
Fig. 6.3 (ATMEGA16) .....	74

Fig. 6.4 (Block Diagram).....	75
Fig. 6.5 (8086/88 Pin out) .....	78
Fig. 6.6 (Rotor Alignment) .....	85
Fig. 6.7 (Stator Winding) .....	86
Fig. 6.8 (Step Alignment ) .....	88
Fig.7.1 .....	92
Fig. 7.2 (Pin Diagram of L298D) .....	93



## INDEX

<u>CHAPTER</u>	<u>Page No.</u>
----------------	-----------------

ABSTRACT .....	9
----------------	---

### Chapter 1: Approach

1.1 CISC (Complex Instruction).....	10
1.2 RISC (Reduced Instruction).....	11
1.3 Comparison between CISC and RISC.....	12
1.4 The Performance Equation .....	13
1.5 Pin Description .....	17

### Chapter 2 : Introduction

2.1 PCB Design.....	21
2.2 Power Supply .....	23
2.3 Controller Section .....	23
2.4 Driver Circuit .....	23

### Chapter 3 : System Design And Implementation

3.1 Semiconductor Diode .....	25
3.2 Rectifiers .....	30
3.3 Transformer .....	32
3.4 Resistor .....	36
3.5 Capacitor .....	37

## **Chapter 4 : AVR CPU Core**

4.1 ALU- Arithmetic Logic Unit .....	44
4.2 Status Register .....	44
4.3 General Purpose Register File .....	47
4.4 Stack pointer .....	49

## **Chapter 5 : Reset and Interrupt Handling**

5.1 Memories .....	54
5.2 SRAM Data Memory .....	55
5.3 EPROM Data Memory.....	58
5.4 System Clock and Clock Distribution .....	60
5.5 Crystal Oscillator .....	63
5.6 External RC Oscillator .....	65

## **Chapter 6 : Other Components**

6.1 78xx Voltage Converter .....	68
6.2 ATMEGA16.....	72
6.3 8086/88 Device Specification .....	76
6.4 Stepper Motor .....	82



**Chapter 7 : Power Management and Sleep mode**

7.1 Project Working and Implementation ..... 89

**Chapter 8 : Algorithm .....94**

**Chapter 9 : Discussion and Future Scope**

9.1 Discussion.....99

9.2 Future scope.....99

**Chapter 10 : conclusion.....101**

**Chapter 11: References.....102**

## ABSTRACT

Robots and embedded control systems are traditionally programmed to perform automated tasks, yet accomplishing complicated and dynamically changing tasks will typically require the use of specialized computational resources and software systems. In order to design these sophisticated control systems the use of computational modeling and simulation techniques is necessary to ensure that final implementations of the system perform optimally. Since even the simplest robotic control systems are intrinsically a mixture of heterogeneous sub-systems, each focusing on a particular task, we need a means of modeling the different aspects of the system accurately and intuitively. Using the Ptolemy II software environment, one can visually design and model elaborate control systems by incorporating heterogeneous models of computation that govern the interaction of multiple components within the system. In this paper we design a robot arm controller component for Ptolemy II that enables various other Ptolemy components to interface and control the robot arm in novel ways, hereby expanding the scope of innovative uses and future applications. We then illustrate an implementation of a Ptolemy model that uses an X-10 remote control, communicating via an X-10 network, to control the movements of a Lynx-5 Robot Arm.



## **CHAPTER 1 : APPRAOCHES**

### **Microcontroller**

A microcontroller (sometimes abbreviated  $\mu\text{C}$ ,  $\text{uC}$  or  $\text{MCU}$ ) is a small computer on a single Integrated Circuit containing a processor core, memory and programmable input/output peripherals..Program memory in the form of NOR Flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in Personal Computers or other general purpose applications.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools and toys. By reducing the size and cost compared to a design that uses a separate microprocessor, memory and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

### **1.1: CISC Approach**

The primary goal of CISC architecture is to complete a task in as few lines of assembly as possible. This is achieved by building processor hardware that is capable of understanding and executing a series of operations. For this particular task, a CISC processor would come prepared with a specific instruction. When

executed, this instruction loads the two values into separate registers, multiplies the operands in the execution unit, and then stores the product in the appropriate register. Thus, the entire

task of multiplying two numbers can be completed with one instruction:

MULT 2:3, 5:2

MULT is what is known as a "complex instruction." It operates directly on the computer's memory banks and does not require the programmer to explicitly call any loading or storing functions. It closely resembles a command in a higher level language. For instance, if we let "a" represent the value of 2:3 and "b" represent the value of 5:2, then this command is identical to the C statement "a = a \* b."

One of the primary advantages of this system is that the compiler has to do very little work to translate a high-level language statement into assembly. Because the length of the code is relatively short, very little RAM is required to store instructions. The emphasis is put on building complex instructions directly into the hardware.

## **1.2: RISC Approach**

RISC processors only use simple instructions that can be executed within one clock cycle. Thus, the "MULT" command described above could be divided into three separate commands: "LOAD," which moves data from the memory bank to a register, "PROD," which finds the product of two operands located within the registers, and "STORE," which moves data from a register to the memory banks. In order to perform the exact series of steps described in the CISC approach, a programmer would need to code four lines of assembly:

LOAD A, 2:3  
LOAD B, 5:2  
PROD A, B  
STORE 2:3, A

At first, this may seem like a much less efficient way of completing the operation. Because there are more lines of code, more RAM is needed to store the assembly level instructions.

The compiler must also perform more work to convert a high-level language statement into code of this form.

### 1.3: Comparison

CISC	RISC
1) Emphasis on hardware	1) Emphasis on software
2) Includes multi-clock complex instructions	2) Single-clock, reduced instruction only
3) Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	3) Register to register: "LOAD" and "STORE" are independent instructions
4) Small code sizes, high cycles per second	4) Low cycles per second, large code sizes



5) Transistors used for storing complex instructions	5) Spends more transistors on memory registers
--	--

However, the RISC strategy also brings some very important advantages. Because each instruction requires only one clock cycle to execute, the entire program will execute in approximately the same amount of time as the multi-cycle "MULT" command. These RISC "reduced instructions" require less transistors of hardware space than the complex instructions, leaving more room for general purpose registers. Because all of the instructions execute in a uniform amount of time (i.e. one clock), pipelining is possible.

Separating the "LOAD" and "STORE" instructions actually reduces the amount of work that the computer must perform. After a CISC-style "MULT" command is executed, the processor automatically erases the registers. If one of the operands needs to be used for another computation, the processor must re-load the data from the memory bank into a register. In RISC, the operand will remain in the register until another value is loaded in its place.

#### 1.4: The Performance Equation

The following equation is commonly used for expressing a computer's performance ability:

$$\frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$$

The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program.

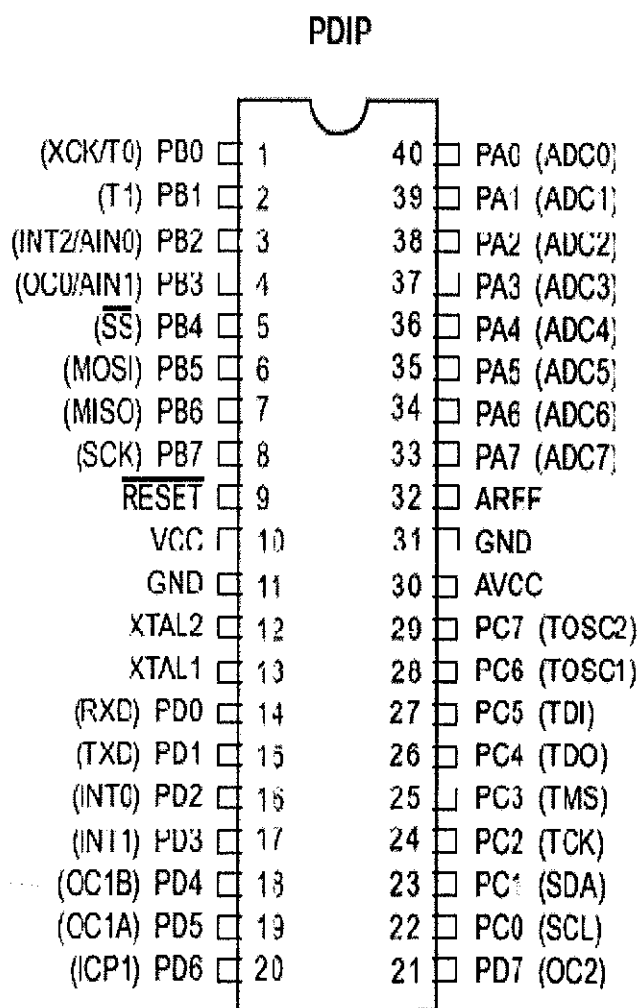
The controller which we use in our project is Atmega16 it have following feature

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16 Kbytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1 Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
- In-System Programming by On-chip Boot Program
- True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
- 8 Single-ended Channels
- 7 Differential Channels in TQFP Package Only

2 Differential Channels with Programmable Gain at 1x, 10x, or 200x

- Byte-oriented Two-wire Serial Interface
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega16L
  - 4.5V - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
    - Power-down Mode: < 1 Ma

**Fig. 1.1: PIN DIAGRAM:**





## **1.5: Pin Descriptions**

**VCC:** Digital supply voltage.

**GND:** Ground.

**Port A (PA7.....PA0) :** Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**Port B (PB7.....PB0) :** Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active even if the clock is not running.

**Port C (PC7.....PC0) :** Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active even if the clock is not running. If the JTAG

interface is enabled, the pull-up resistors on pins PC5 (TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed.

**Port D (PD7.....PD0)** : Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active even if the clock is not running.

**RESET:** Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running.

**XTAL1:** Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

**XTAL2:** Output from the inverting Oscillator amplifier.

**AVCC:** AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.

**AREF:** AREF is the analog reference pin for the A/D Converter

## CHAPTER 2 : INTRODUCTION

There are many definitions of robot and no real consensus has been attained so far. We loosely define a robot as follows:

**Robot:** An electromechanical device which is capable of reacting in some way to its environment, and take autonomous decisions or actions in order to achieve a specific task.

These days the industry is depend on the production of the goods at high pace. As we know in Auto industry have so many different task which involve the very hard environment like RIG welding , spray painting etc. so we try to make our employ safe to work. Even in nuclear plant to deploy nuclear fuel in the reactors, which is one of the dangerous job of the world for men and also we need somebody which not got effected by the radiation emitted by the radioactive material.

Robots used to manipulate objects like bombs or hazardous materials need a robotic arm. Robotic arms usually have several points of articulation or joints. The arm might have the same capabilities of a human arm, with shoulder, elbow and wrist articulation, or it may have many more joints, allowing the operator to reach places he wouldn't be able to get to on his own. On the end of the arm is a manipulator, usually a gripping device in the form of a two-fingered claw.

Because the officer controlling the robot is at least several meters away from the robot, he needs a way to see the robot's

environment independent of his own perspective. For this reason, police robots use video cameras to broadcast images back to the operator's laptop or console. Most robots use at least two or three cameras so that the operator can stay aware of the robot's surroundings. Some models have a camera mounted on every point of articulation, as well as stationary cameras attached to the body of the robot. Video camera systems range from black-and-white to night vision and infrared.

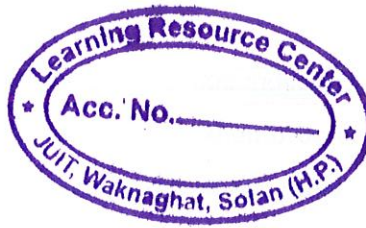
Another feature many police robots have is a two-way audio system. Manufacturers mount microphones and speakers on the robot, allowing police to listen to sounds in the robot's environment or communicate with suspects or hostages in a dangerous situation. The robot can become the eyes and ears of the police force without a single officer being put in harm's way.

Every robot have arm which make him work easily. As every robot arm is free to move in all direction, can hold the object . so it is very important to make robotic arm and make it work manually.

These are the following things which is main part of the any embedded system these are as follows

- 1. PCB design**
- 2. Power supply.**
- 3. Controller circuit**
- 4. Driver circuit**





## **2.1: PCB design**

In PCB design we keep in mind that the circuit track should be kept at the mini distance that two track signal should not interfere with the adjusting track .Which result in the induction of the voltage over the other track which ride over the signal. The thickness of the track also decides of the current carrying capacity of the track. So the power track sometime needs to be soldered with extra solder. Printed circuit board (PCB) is a component made of one or more layers of insulating material with electrical conductors. The insulator is typically made on the base of fiber reinforced resins, ceramics, plastic, or some other dielectric materials. During manufacturing, the portions of conductors that are not needed are etched off, leaving printed circuits that connect electronic components.

Currently the main generic standard for the design of printed circuit boards, regardless of materials, is IPC-2221A. Whether a PCB board is single-sided, double-sided or multilayer, this standard provides rules for manufacturability and quality such as requirements for material properties, criteria for surface plating, conductor thickness, component placement, dimensioning and tolerance rules, and more. For a specific technology, the designer can then choose the appropriate sectional standard from the IPC-2220 series. For power conversion devices,

additional parameters are recommended by IPC-9592. The width of the circuit conductors should be chosen based on the maximum temperature rise at the rated current and the acceptable impedance. The spacing between the PC traces is determined by peak working voltage, the coating and the product application. The minimum possible widths of the traces and of the spacing between them are both limited by the manufacturing capabilities of your fabricator and in any case should not be less than 2 mils. Typical minimum values are 6/6 mils. Depending on the application and product end use, other standards may also apply. For example, for mains-powered or battery-powered information technology equipment, the creep age and clearance requirements of IEC/UL 60950-1 shall take precedence over IPC. IPC and other standards do not tell you how to properly route the board. Good PCB layout techniques require an understanding of the effects of non-zero trace impedance and the coupling of signals from one circuit to another through parasitic capacitances and radio transmission, as well as a basic understanding of circuit operation. Auto placement may be done for most parts of control circuits, but power, ground and high di/dt circuits should be routed by hand. Here you will find guidelines for designing PCB, links to software downloads, trace calculators and other info, tools and online resources.

## **2.2: Power supply**

The power supply is the important part of any electronics circuit. if we select under rated power supply (for example if we have circuit which required 12 V, 1 A, if we took 12 V, 750 m A then when it need full current then it result in the dip of voltage supply from 12 to lower voltage and if the circuit is working on the 12 volt and it got low voltage then it effect the working of the electronics chips which are design for particular voltage level. So deign of the power supply according to the circuit is very important.

## **2.3: Controller selection**

Now the selection of the controller depend on the number of input and out to be sense and control. So we must keep in mind that the controls should of pin number according to the it. So if there is only two in and out to be sense and control than we must took low pin controller as if we select controller with more pin it make circuit design then we are going to make bigger circuit which is not the best concept for the embedded.

## **2.4: Driver circuit**

The driver circuit is very important in case of operating the relay a, interface of the two different type of IC with each other as CMOS need less current and other side TTL IC need more current as they are Transistor based IC .also for long distance communication we needed buffer driver circuit in between then we must keep in mind the maxi. Distance that a cable can take signal from one side to other side cable.

#### Block diagram

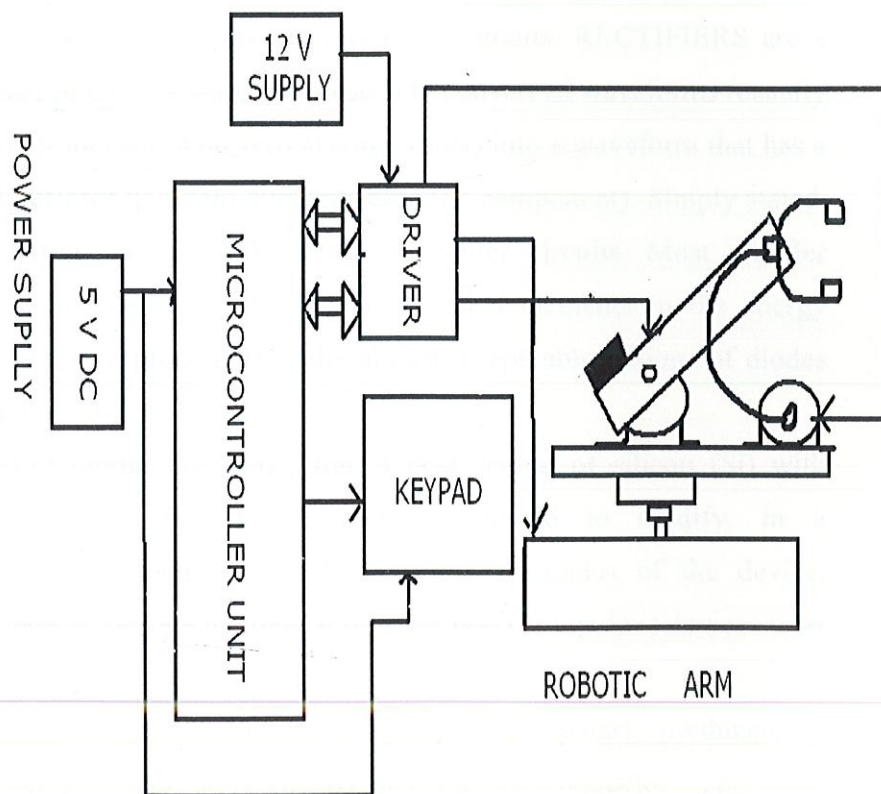


Fig. 2.1

## CHAPTER 3: SYSTEM DESIGNE AND IMPLEMENTATION

### **3.1: SEMICONDUCTOR DIODE**

Generally refers to a two-terminal solid-state semiconductor device that presents a low impedance to current flow in one direction and a high impedance to current flow in the opposite direction. These properties allow the diode to be used as a one-way current valve in electronic circuits. RECTIFIERS are a class of circuits whose purpose is to convert ac waveforms (usually sinusoidal and with zero average value) into a waveform that has a significant non-zero average value (dc component). Simply stated, rectifiers are ac-to-dc energy converter circuits. Most rectifier circuits employ diodes as the principal elements in the energy conversion process; thus the almost inseparable notions of diodes and rectifiers.

Most diodes are made from a host crystal of silicon (Si) with appropriate impurity elements introduced to modify, in a controlled manner, the electrical characteristics of the device. These diodes are the typical *pn*-junction (or bipolar ) devices used in electronic circuits.

Another type is the Schottky diode (unipolar), produced by placing a metal layer directly onto the semiconductor . The metal semiconductor interface serves the same function as the *pn* -



junction in the common diode structure. Other semiconductor materials such as gallium-arsenide (GaAs) and silicon-carbide (SiC) are also in use for new and specialized applications of diodes. Detailed discussion of diode structures and the physics of their operation can be found in later paragraphs of this section. The electrical circuit symbol for a bipolar diode is shown in Fig.1.

The polarities associated with the forward voltage drop for forward current flow are also included. Current or voltage opposite to the polarities are considered to be negative values with respect to the diode conventions shown. The current-voltage dependencies of typical diodes. The diode conducts forward current with a small forward voltage drop across the device, simulating a closed switch.

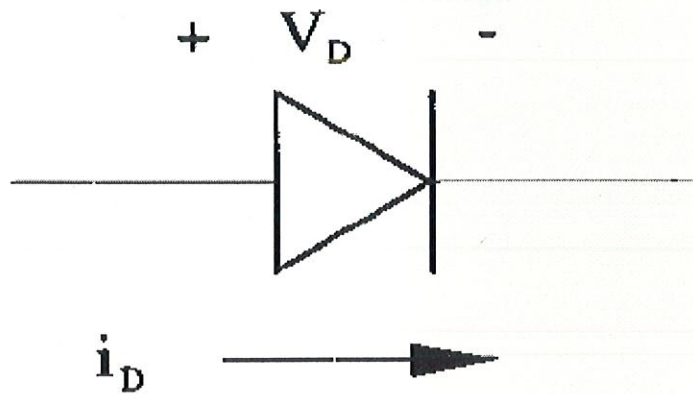
The relationship between the forward current and forward voltages is approximately given by the Shockley diode equation [Shockley, 1949] :  $k$  Boltzmann's constant, and  $T$  is the temperature of the semiconductor. Around the knee of the curve in Fig.2 is a positive voltage that is termed the turn-on or sometimes the threshold voltage for the diode. This value is an approximate voltage above which the diode is considered turned "on" and can be modeled to first degree as a closed switch with constant forward drop. Below the threshold voltage value the diode is considered weakly conducting and approximated as an open switch.

.Reverse voltage applied to the diode causes a small leakage current (negative according to the sign convention) to flow that is typically orders of magnitude lower than current in the forward direction. The diode can withstand reverse voltages up to a limit determined by its physical construction and the semiconductor material used beyond this value the reverse voltage imparts enough

energy to the charge carriers to cause large increases in current. The mechanisms by which this current increase occurs are impact ionization (avalanche) [McKay, 1954] and a tunneling phenomenon (Zener breakdown) [Moll, 1964]. Avalanche breakdown results in large power dissipation in the diode, is generally destructive, and should be avoided at all times. Both breakdown regions are superimposed in Fig.3.2 for comparison of their effects on the shape of the diode characteristic curve. Avalanche breakdown occurs for reverse applied voltages in the range of volts to kilovolts depending on the exact design of the diode. Zener breakdown occurs at much lower voltages than the avalanche mechanism. Diodes specifically designed to operate in the Zener breakdown mode are used extensively as voltage regulators in

During forward conduction the power loss in the diode can become excessive for large current flow. Schottky diodes have an inherently lower turn-on voltage than *pn*-junction diodes and are therefore more desirable in applications where the energy losses in the diodes are significant (such as output rectifiers in switching power supplies). Other considerations such as recovery characteristics from forward conduction to reverse blocking may also make one diode type more desirable than another. Schottky diodes conduct current with one type of charge carrier and are therefore inherently faster to turn off than bipolar diodes. However, one of the limitations of Schottky diodes is their excessive forward voltage drop when designed to support reverse biases above about 200 V. Therefore, high-voltage diodes are the *pn*-junction type. The effects due to an increase in the temperature in a bipolar diode are many. The forward voltage drop during

conduction will decrease over a large current range, the reverse leakage current will increase, and the reverse avalanche breakdown voltage will increase as the device temperature climbs. A family of static characteristic curves highlighting these effects is shown in Fig.5.3 where in addition, a major effect on the switching characteristic is the increase in the reverse recovery time during turn-off. Some of the key parameters to be aware of when choosing a diode are its repetitive peak inverse voltage rating, (relates to the avalanche breakdown value), the peak forward surge current rating, (relates to the maximum allowable transient heating in the device), the average or rms current rating,



**FIGURE 3.1** Circuit symbol for a bipolar diode indicating the polarity associated with the forward voltage and current directions

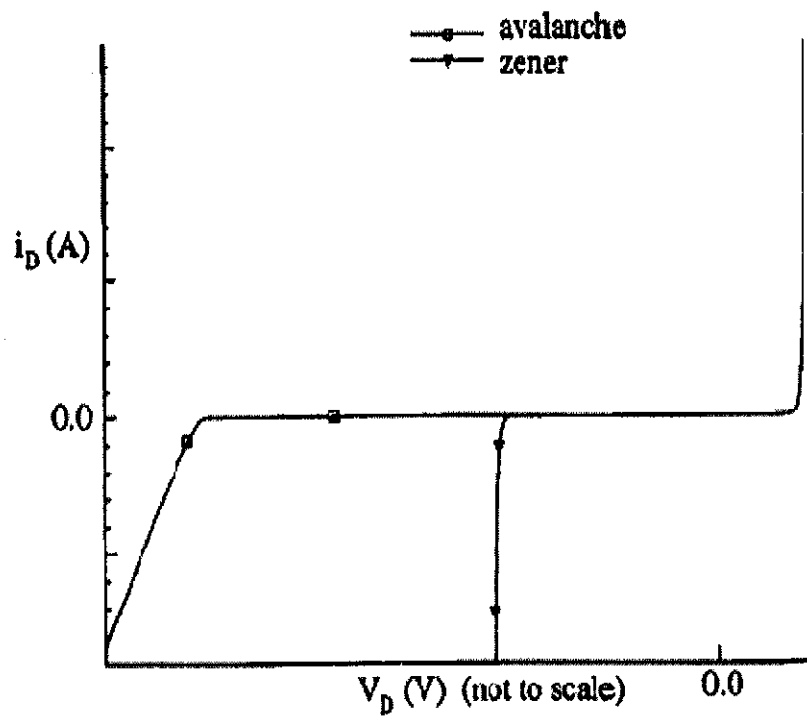


FIGURE 3.2 A typical diode dc characteristic curve showing the current dependence on voltage.



### 3.2: RECTIFIERS

This section discusses some simple **uncontrolled rectifier** circuits that are commonly encountered. The term *uncontrolled* refers to the absence of any control signal necessary to operate the primary switching elements (diodes) in the rectifier circuit. The discussion of controlled rectifier circuits, and the controlled switches themselves, is more appropriate in the context of power electronics applications. Rectifiers are the fundamental building block in dc power supplies of all types and in dc power transmission used by some electric utilities. A single-phase full wave rectifier circuit with the accompanying input and output voltage waveform is shown in FIGURE 3.3.

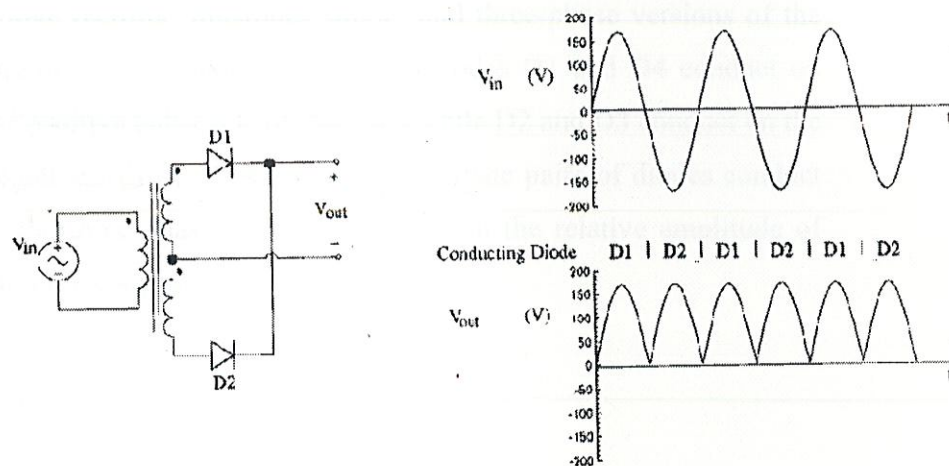


FIGURE 3.3

This topology makes use of a center-tapped transformer with each diode conducting on opposite half-cycles of the input voltage. The forward drop across the diodes is ignored on the output graph, which is a valid approximation if the peak voltages of the input and output are large compared to 1 V. The circuit changes a sinusoidal waveform with no dc component (zero

average value) to one with a dc component of  $2 V$  peak. The rms value of the output is  $0.707 V$  peak. The dc value can be increased further by adding a low-pass filter in cascade with the output.

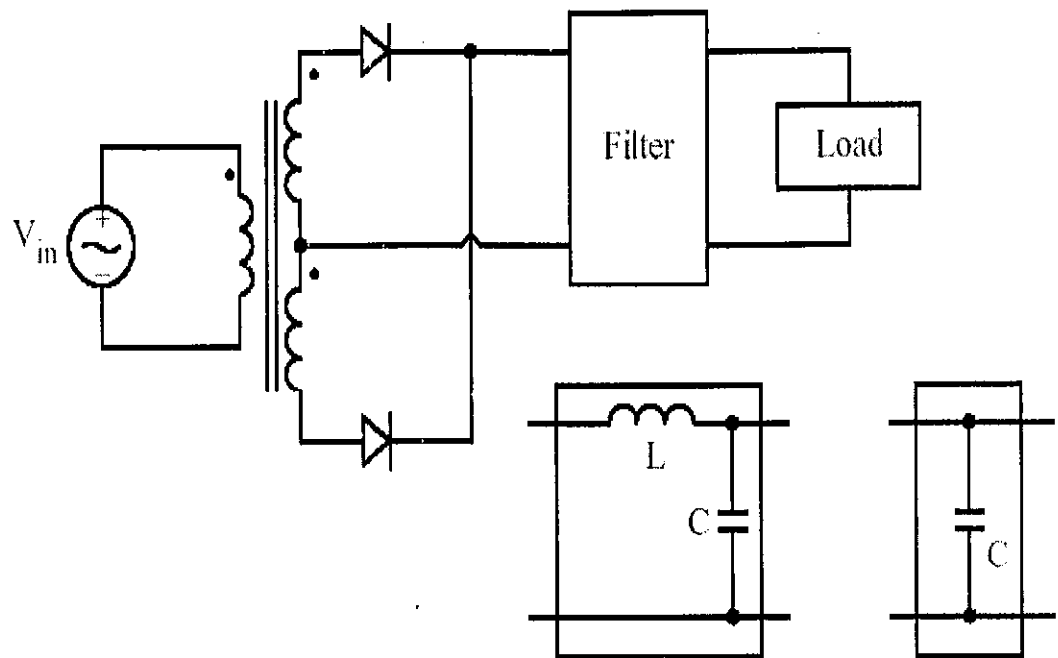
The usual form of this filter is a shunt capacitor or an LC filter as shown in Fig. 3.4. The resonant frequency of the LC filter should be lower than the fundamental frequency of the rectifier output for effective performance. The ac portion of the output signal is reduced while the dc and rms values are increased by adding the filter. The remaining ac portion of the output is called the **ripple**. Though somewhat confusing, the transformer, diodes, and filter are often collectively called the rectifier circuit.

Another circuit topology commonly encountered is the bridge rectifier. illustrates single- and three-phase versions of the circuit. In the single-phase circuit diodes D1 and D4 conduct on the positive half-cycle of the input while D2 and D3 conduct on the negative half-cycle of the input. Alternate pairs of diodes conduct in the three phase circuit depending on the relative amplitude of the source signals.



**FIGURE 3. 4:**

A single-phase , full-wave rectifier circuit using a center-tapped transformer with the associated input and output waveforms.



### 3.3 : TRANSFORMER

Transformer is an electric instrument used for convert one voltage level to other voltage level using the phenomenon of electromagnet induction.

#### Types of Transformers

Transformers are broadly grouped into two main categories: dry-type and liquid-filled transformers. Dry-type

transformers are cooled by natural or forced circulation of air or inert gas through or around the transformer enclosure. Dry-type transformers are further subdivided into ventilated, sealed, or encapsulated types depending upon the construction of the transformer. Dry transformers are extensively used in industrial power distribution for rating up to 5000 kVA and 34.5 kV.

Liquid-filled transformers are cooled by natural or forced circulation of a liquid coolant through the windings of the transformer. This liquid also serves as a **dielectric** to provide superior voltage-withstand characteristics. The most commonly used liquid in a transformer is a mineral oil known as transformer oil that has a continuous operating temperature rating of 105°C, a flash point of 150°C, and a fire point of 180°C. A good grade transformer oil has a **breakdown strength** of 86.6 kV/cm (220 kV/in.) that is far higher than the breakdown strength of air, which is 9.84 kV/cm (25 kV/in.) at atmospheric pressure. Silicone fluid is used as an alternative to mineral oil.

The breakdown strength of silicone liquid is over 118 kV/cm (300 kV/in.) and it has a flash point of 300°C and a fire point of 360°C. Silicone-fluid-filled transformers are classified as less flammable. The high dielectric strengths and superior thermal conductivities of liquid coolants make them ideally suited for large high voltage power transformers that are used in modern power generation and distribution.

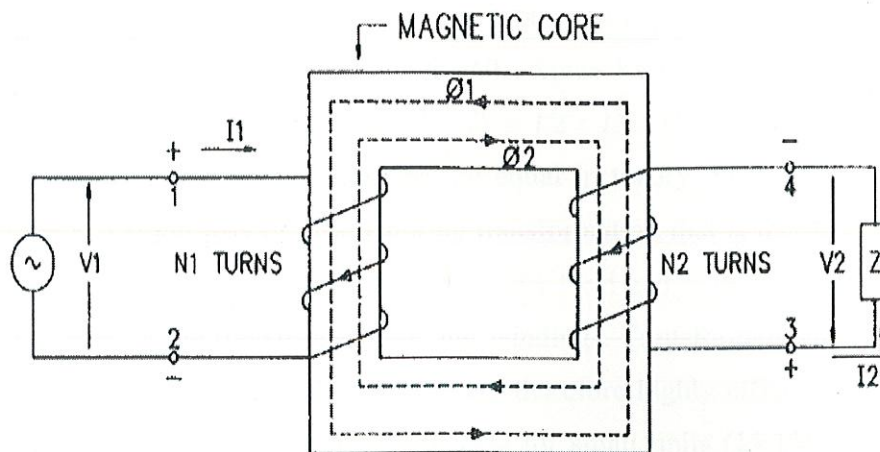


FIGURE 3. 5

### Principle of Transformation

The actual process of transfer of electrical power from a voltage of  $V_1$  to a voltage of  $V_2$  is explained with the aid of the simplified transformer representation shown in Fig. Application of voltage across the primary winding of the transformer results in a **magnetic field** of  $\Phi_1$  Wb in the magnetic core, which in turn induces a voltage of  $V_2$  at the secondary terminals.  $V_1$  and  $V_2$  are related by the expression  $V_1/V_2 = N_1/N_2$ , where  $N_1$  and  $N_2$  are the number of turns in the primary and secondary windings, respectively. If a load current of  $I_2$  A is drawn from the secondary terminals, the load current establishes a magnetic field of  $\Phi_2$  Wb in the core and in the direction shown.

Since the effect of load current is to reduce the amount of primary magnetic field, the reduction in  $\Phi_1$  results in an increase in the primary current  $I_1$  so that the net magnetic field is almost restored to the initial value and the slight reduction in the field is

due to leakage **magnetic flux**. The currents in the two windings are related by the expression  $I_1/I_2 = N_2/N_1$ . Since  $V_1/V_2 = N_1/N_2 = I_2/I_1$ , we have the expression  $V_1 \cdot I_1 = V_2 \cdot I_2$ . Therefore, the volt-amperes in the two windings are equal in theory. In reality, there is a slight loss of power during transformation that is due to the energy necessary to set up the magnetic field and to overcome the losses in the transformer core and windings. Transformers are static power conversion devices and are therefore highly efficient. Transformer efficiencies are about 95% for small units (15 kVA and less), and the efficiency can be higher than 99% for units rated above 5 MVA.

### Electromagnetic Equation

Figure 3.5 shows a magnetic core with the area of cross section  $A = W \cdot D$  m<sup>2</sup>. The transformer primary winding that consists of  $N$  turns is excited by a sinusoidal voltage  $v = V \sin(\omega t)$ , where  $\omega$  is the angular frequency given by the expression  $\omega = 2\pi f$  and  $f$  is the frequency of the applied voltage waveform.  $\phi$  is magnetic field in the core due to the excitation current  $i$ :

Induced voltage in the winding

$$\phi = \Phi \sin\left(\omega t - \frac{\pi}{2}\right) = -\Phi \cos(\omega t)$$

Maximum value of the induced voltage

$$e = -N \frac{d\phi}{dt} = N \frac{d[\Phi \cos(\omega t)]}{dt} = -N\omega\Phi \sin(\omega t)$$

The root-mean-square value

$$E = N\omega\Phi$$

$$E_{\text{rms}} = \frac{E}{\sqrt{2}} = \frac{2\pi f N\Phi}{\sqrt{2}} = 4.44 f NBA$$

Where flux  $\Phi$ (webers) is replaced by the product of the flux density  $B$  (teslas) and the area of cross section the core. This fundamental design equation determines the size of the transformer for any given voltage and frequency. Power transformers are normally operated at flux density levels of 1.5 T.

### 3.4: Resistors

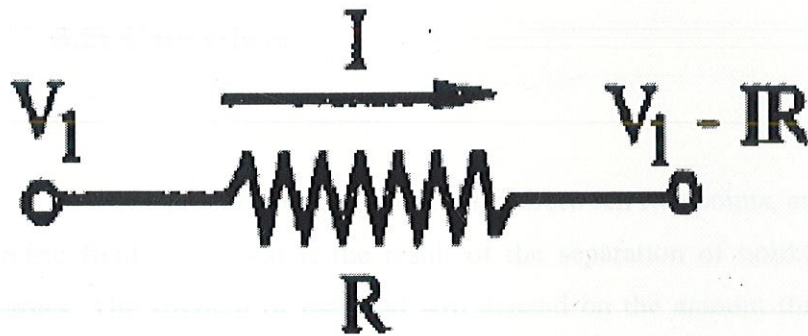


FIGURE 3.6

The resistor is an electrical device whose primary function is to introduce resistance to the flow of electric current. The magnitude of opposition to the flow of current is called the resistance of the resistor. A larger resistance value indicates a greater opposition to current flow. The resistance is measured in ohms. An ohm is the



resistance that arises when a current of one ampere is passed through a resistor subjected to one volt across its terminals.

The various uses of resistors include setting biases, controlling gain, fixing time constants, matching and loading circuits, voltage division, and heat generation.

The following sections discuss resistor characteristics and various resistor types. **FIGURE 3.6** shows a resistor with resistance  $R$  having a current  $I$  flowing through it will have a voltage drop of  $IR$  across it.

### 3.5: Capacitors

If a potential difference is found between two points, an electric **field** exists that is the result of the separation of unlike charges. The strength of the field will depend on the amount the charges have been separated. *Capacitance* is the concept of energy storage in an electric field and is restricted to the area, shape, and spacing of the **capacitor** plates and the property of the material separating them. When electrical current flows into a capacitor, a force is established between two parallel plates separated by a **dielectric**. This energy is stored and remains even after the input is removed. By connecting a **conductor** (a resistor, hard wire, or even air) across the capacitor, the charged capacitor can regain electron balance, that is, discharge its stored energy.



$$C = \frac{x\epsilon[(N - 1)A]}{d} \times 10^{-13}$$

The value of a parallel-plate capacitor can be found with the equation

Where

$C$  = capacitance, F;

$\epsilon$  = dielectric constant of insulation;

$d$  = spacing between plates;

$N$  = number of plates;

$A$  = area of plates; and  $x = 0.0885$

when  $A$  and  $d$  are in centimeters, and  $x = 0.225$  when  $A$  and  $d$  are in inches. The work necessary to transport a unit charge from one plate to the other is  $e = kg$  (1.22)

where  $e$  = volts expressing energy per unit charge,  $g$  = coulombs of charge already transported, and  $k$  = proportionality factor between work necessary to carry a unit charge between the two plates and charge already transported. It is equal to  $1/C$ , where  $C$  is the capacitance, F.

The value of a capacitor can now be calculated from the equation.

$$C = \frac{q}{e}$$

### WORKING:-

THE STEP INVOLVED IN WOKRING OF THIS PROJECT ARE AS FOLLOWS

- 1) 220 VOLT AC IS CONVERTED TO 12 VOLT AC USING STEP DOWN TRANSFORMER.
- 2) THEN THIS 12 V AC IS CONVETED INTO DC BY USING FULL WAVE RECTIFIER.
- 3) THIS DC OUT FORM FULL WAVE RECTIFIER IS PULSATED NOT PURE DC SUPPLY. THIS PULSATED DC IS THEN FEED TO CAPACITOR WHICH MAKE IT SMOOTH BY REMOVEING RIPPESAS SHOW IN FIGURE.

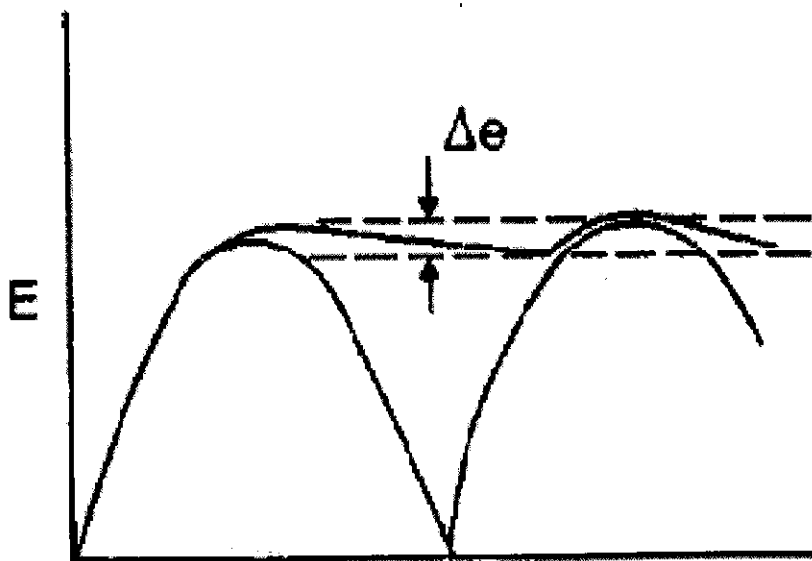
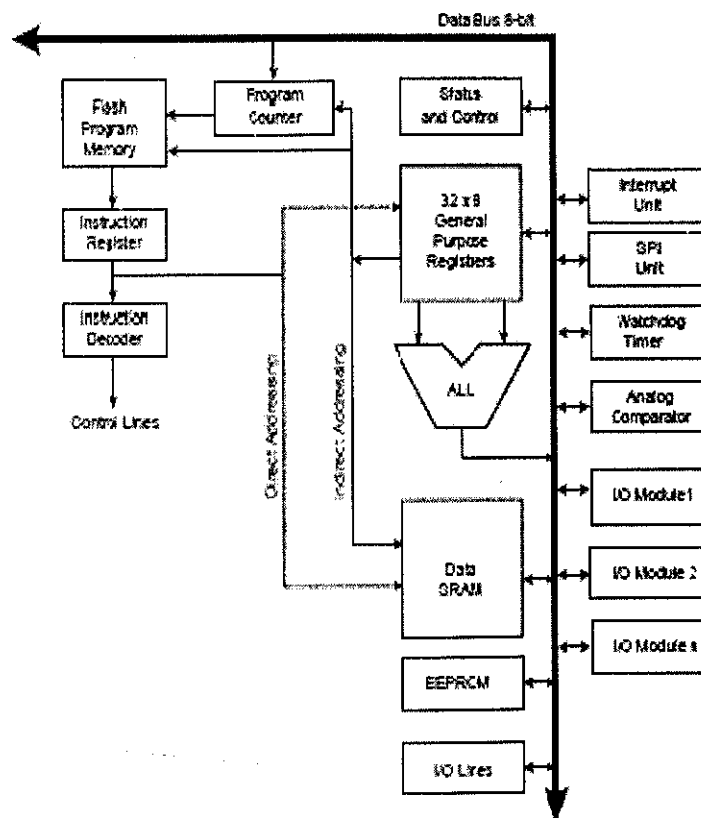


FIGURE 3.7

- 4) AFTER SMOOTHING THS DC SUPPLY WE FEED IT INTO  
VOLTAGE REGULATOR LM7805.
- 5) NOW BY VARING THE VARIABLE RESISTANCE WE  
GET VARIABLE DC SUPPLY AT OUT PUT.
- 6) WHICH RANGE FROM 0-12 VOLT.

## CHAPTER 4: AVR CPU CORE

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.



**(Fig. 4.1) Block Diagram of the AVR MCU Architecture**

In order to maximize performance and parallelism, the AVR uses Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle.

The program memory is In-System Reprogrammable Flash memory. The fast-access Register File contains  $32 \times 8$ -bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle. Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations.

One of these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-register, Y-register, and Z-register, described later in this section. The ALU supports arithmetic and logic operations between registers or between a constant and a register.

Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation. Program



flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16-bit or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section. During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack.

The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps. A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the Status Register. All interrupts have a separate interrupt vector in the interrupt vector table.

The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority. The I/O memory space contains 64 addresses for CPU

peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data space locations following those of the Register File, \$20 - \$5F .

#### **4.1: ALU – Arithmetic Logic Unit**

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

#### **4.2: Status Register**

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference.

This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code. The Status Register is not automatically stored

when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Fig. 4.2**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.



- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit Load) and BST (Bit Store) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit,  $S = N \oplus V$**

The S-bit is always an exclusive or between the Negative Flag N and the Two’s Complement Overflow Flag V.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetic.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation.

### **4.3: General Purpose Register File**

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Shows the structure of the 32 general purpose working registers in The CPU.



**(Fig. 4.3)** AVR CPU General Purpose Working Registers

General Purpose Working Registers	7	0	Addr.	
	R0		\$00	
	R1		\$01	
	R2		\$02	
	...			
	R13		\$0D	
	R14		\$0E	
	R15		\$0F	
	R16		\$10	
	R17		\$11	
	...			
	R26		\$1A	X-register Low Byte
	R27		\$1B	X-register High Byte
	R28		\$1C	Y-register Low Byte
	R29		\$1D	Y-register High Byte
	R30		\$1E	Z-register Low Byte
	R31		\$1F	Z-register High Byte

Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions. As shown in Figure, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.

#### 4.4: Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer. If software reads the Program Counter from the Stack after a call or an interrupt, unused bits (15:13) should be masked out. The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above \$60.

The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI. The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH register will not be present .

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

**Fig. 4.4**



## CHAPTER 5 : RESET AND INTERRUPT HANDLING

The AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed.



The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request

0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). Refer to “Interrupts” on page 45 for more information. The Reset Vector can also be moved to the start of the boot Flash section by programming the BOOTRST Fuse, see “Boot Loader Support – Read-While-Write Self- Programming” on page 246. When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user

software can write logic one to the I-bit to enable nested interrupts.

1. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed. There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.

If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software.

Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the global interrupt enable bit is set, and will then be executed by order of priority. The second type of interrupts will trigger as long as the interrupt condition is present.

These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served. Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine.



This must be handled by software. When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

### Assembly Code Example

```
in r16, SREG      ; store SREG value
cli              ; disable interrupts during timed sequence
sbi EECR, EEMWE   ; start EEPROM write
sbi EECR, EEWE
out SREG, r16     ; restore SREG value (I-bit)
```

### C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
_cli();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```



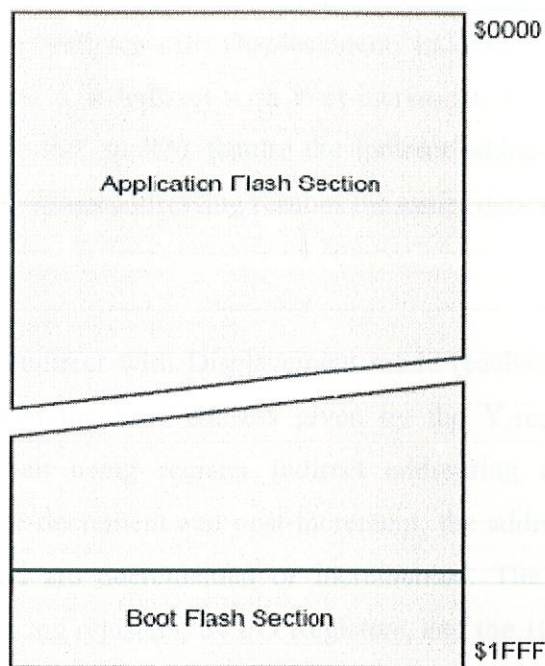
## **5.1: Memories**

This section describes the different memories in the ATmega16. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega16 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

### **In-System Reprogrammable Flash Program Memory**

The ATmega16 contains 16 Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8K ×

16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section. The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations.



**(Fig. 5.1)** Program Memory Map

## 5.2: SRAM Data Memory

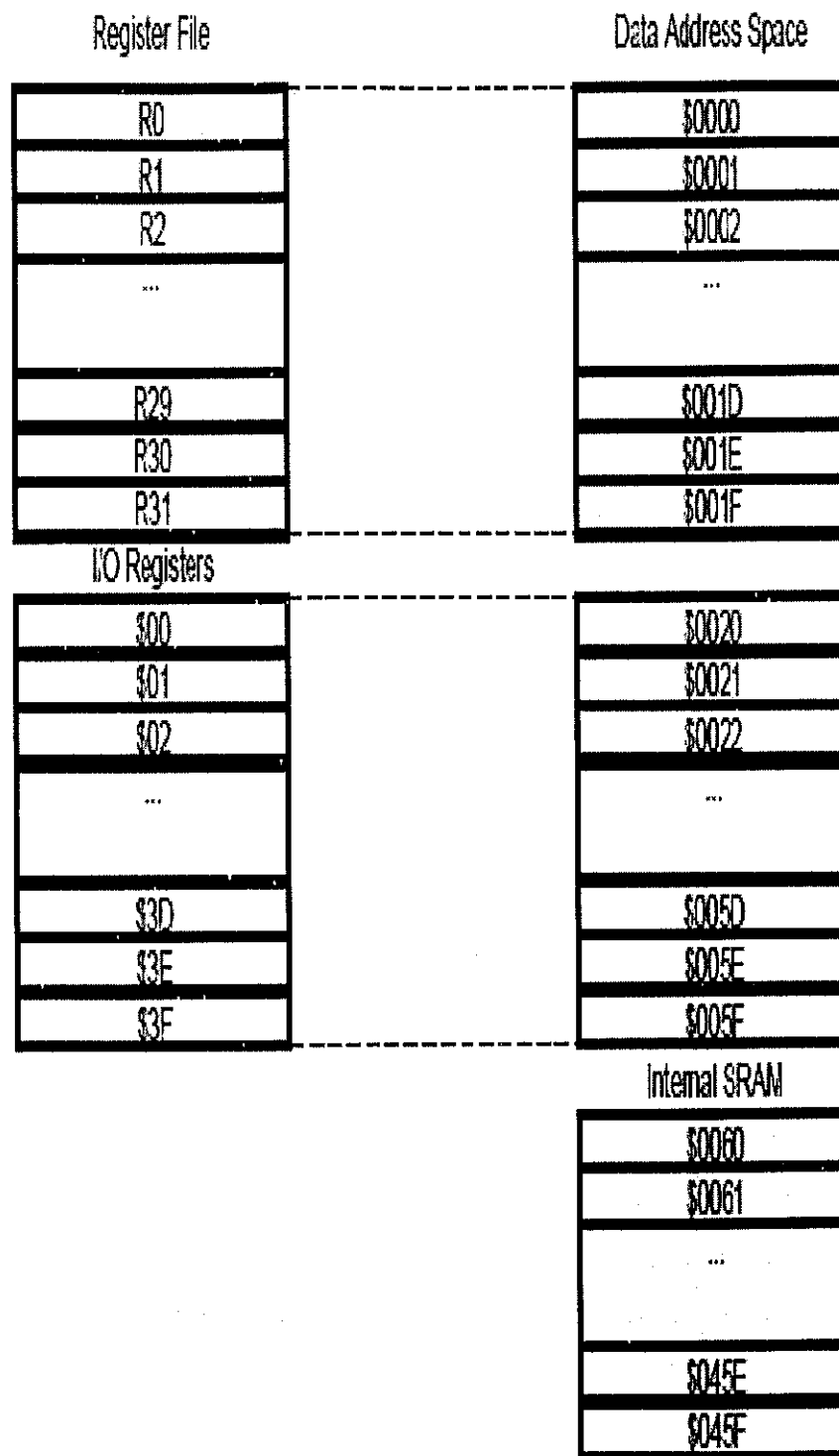
The above figure shows how the ATmega16 SRAM Memory is organized. The lower 1120 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM.

The first 96 locations address the Register File and I/O Memory, and the next 1024 locations address the internal data

SRAM. The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers. The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y-register or Z-register. When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented. The 32 general purpose working registers, 64 I/O Registers, and the 1024 bytes of internal data SRAM in the ATmega16 are all accessible through all these addressing modes.

**(Fig. 5.2)** Data Memory Map





### 5.3 : EEPROM Data Memory

The ATmega16 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

#### EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space. The write access time for the EEPROM is given in Table . A self-timing function, however, lets the user software detect when the next byte can be written.

If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this. When the EEPROM is read, the CPU is halted for four clock cycles before the next

instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is execute

Bit	15	14	13	12	11	10	9	3	
	-	-	-	-	-	-	-	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	RW	
	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

**(Fig. 5.3)**



## **Bits 15.9 – Res: Reserved Bits**

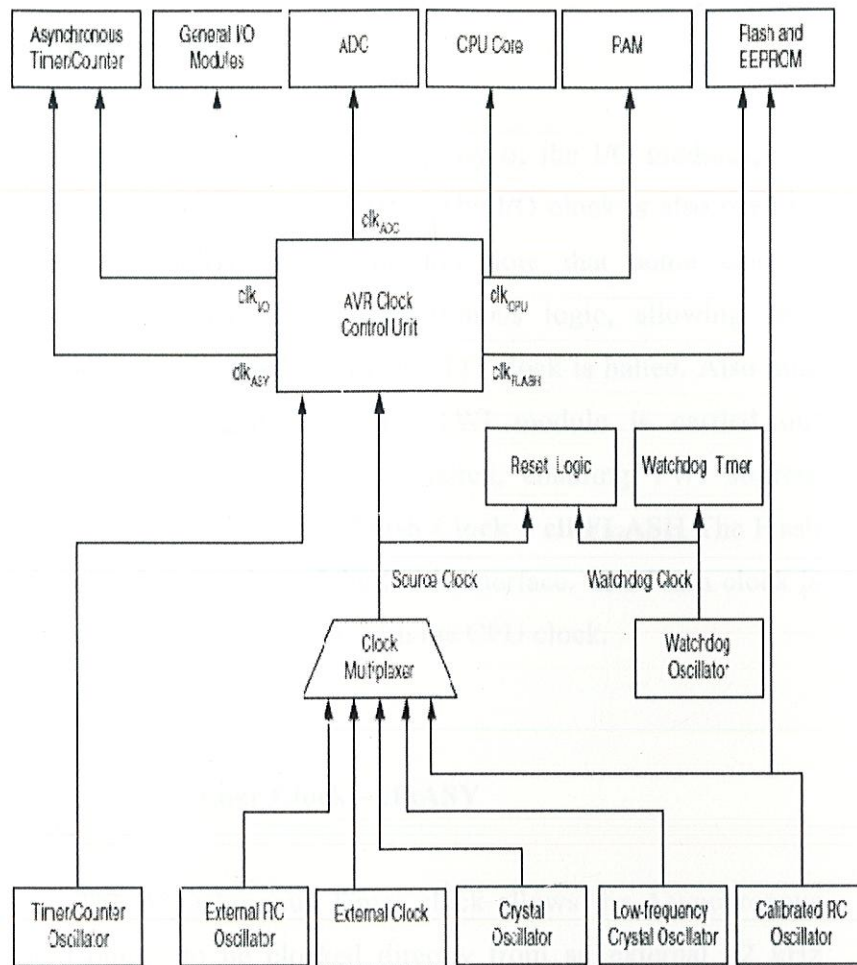
These bits are reserved bits in the ATmega16 and will always read as zero.

### **• Bits 8.0 – EEAR8..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL – specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

## **5.4: System Clock and Clock Options Clock Systems and their Distribution**

Figure 5.4 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described.



(Fig. 5.4)

### CPU Clock – $clk_{CPU}$

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.



## **I/O Clock – clkI/O**

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that address recognition in the TWI module is carried out asynchronously when clkI/O is halted, enabling TWI address reception in all sleep modes. **Flash Clock – clkFLASH** The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

## **Asynchronous Timer Clock – clkASY**

The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

## **ADC Clock – clkADC**

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

## **Clock Sources**

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

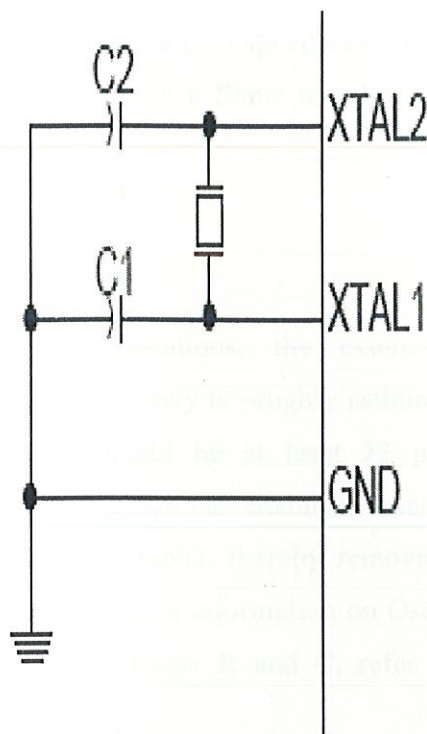
### **Default Clock Source**

The device is shipped with CKSEL = "0001" and SUT = "10". The default clock source setting is therefore the 1 MHz Internal RC Oscillator with longest startup time. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel Programmer .

#### **5.5: Crystal Oscillator**

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in FIGURE 5.5

## Crystal Oscillator Connections



**(Fig. 5.5)**

Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate with a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range.

When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably. This mode has a limited frequency range and it can not be used to drive other clock buffers.

For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed.



C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for

## **5.6 : External RC Oscillator**

For timing insensitive applications, the external RC configuration can be used. The frequency is roughly estimated by the equation  $f = 1/(3RC)$ . C should be at least 22 pF. By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND, thereby removing the need for an external capacitor. For more information on Oscillator operation and details on how to choose R and C, refer to the External RC Oscillator

## **Oscillator Calibration Register – OSCCAL**

### **• Bits 7.0 – CAL7..0: Oscillator Calibration Value**

Writing the calibration byte to this address will trim the Internal Oscillator to remove process variations from the Oscillator frequency. This is done automatically during Chip Reset. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the Internal Oscillator. Writing \$FF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the

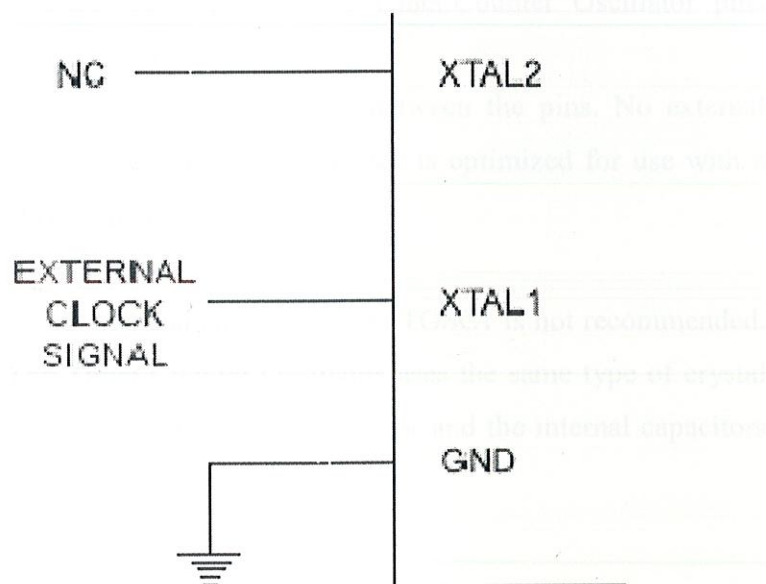


Oscillator is intended for calibration to 1.0 MHz, 2.0 MHz, 4.0 MHz, or 8.0 MHz Tuning to other values is not guaranteed

## External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 5.6

(Fig. 5.6)



To run the device on an external clock, the CKSEL Fuses must be programmed to "0000". By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND. Figure. External Clock Drive Configuration When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU.

A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in reset during such changes in the clock frequency.

### **Timer/Counter Oscillator**

For AVR microcontrollers with Timer/Counter Oscillator pins (TOSC1 and TOSC2),

the crystal is connected directly between the pins. No external capacitors are needed. The Oscillator is optimized for use with a 32.768 kHz watch crystal.

Applying an external clock source to TOSC1 is not recommended.

Note: The Timer/Counter Oscillator uses the same type of crystal oscillator as Low-Frequency Oscillator and the internal capacitors have the same nominal value of 36 pF.

## **CHAPTER 6: OTHER COMPONENTS**

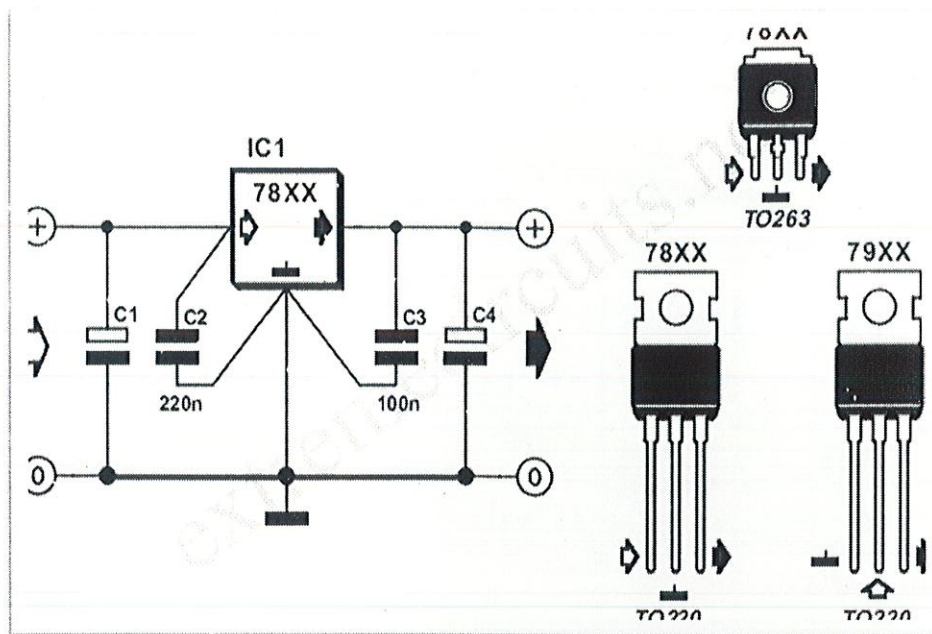
### **6.1: 78XX Voltage Converter**

#### **Description:**

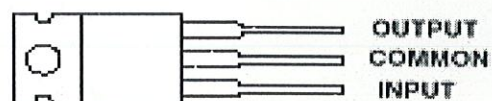
This series of fixed-voltage monolithic integrated-circuit voltage regulators is designed for a wide range of applications. These applications include on-card regulation for elimination of noise and distribution problems associated with single-point regulation. In addition, they can be used with power-pass elements to make high-current voltage regulators. Each of these regulators can deliver up to 1.5 A of output current. The internal limiting and thermal shutdown features of these regulators make them essential immune to overload.

#### **Features**

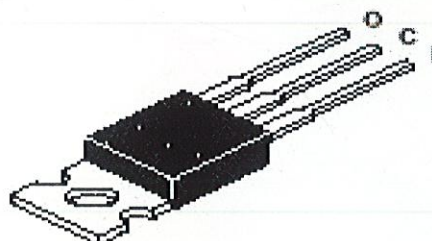
- 1 - 3-Terminal Regulators
- 2 - Output Current Up to 1.5 A
- 3 - No External Components
- 4 - Internal Thermal Overload Protection
- 5 - High Power Dissipation Capability
- 6 - Internal Short-Circuit Current Limiting
- 7 - Output Transistor Safe-Area Compensation



( Fig. 6.1) 78xx  
(TOP VIEW)



The common terminal is in electrical contact with the mounting base.



(Fig. 6.2)



## Electrical characteristics 7805

Electrical characteristics at specified virtual junction temperature,  $V_i = 10V$ ,  $I_o = 500mA$  (unless otherwise noted)

Parameter	Test Conditions*	7805			Units
		Min	Typ	Max	
Output voltage**	25°	4.8	5	5.2	V
	$I_o = 5mA$ to 1A, $V_i = 7V$ to 20V, $P_{\Sigma} \leq 15W$	4.75	5	5.25	
Input regulation	25°		3	100	mV
	$V_i = 7V$ to 25V		1	50	
Ripple rejection	$V_i = 8V$ to 18V, $f = 120Hz$		78		dB
Output regulation	$I_o = 5mA$ to 1.5A		15	100	mV
	$I_o = 250mA$ to 750mA		5	50	
Output resistance	$f = 1KHz$		0.017		$\Omega$
Temperature coefficient of output voltage	$I_o = 5mA$		-1.1		mV/°
Output noise voltage	$f = 10Hz$ to 100 KHz		40		$\mu V$
Dropout voltage	$I_o = 1A$		2.0		V
Bias current	25°		4.2	8	mA
Bias current change	$V_i = 7V$ to 25V			1.3	
	$I_o = 5mA$ to 1A			0.5	
Short-circuit output current	25°		750		
Peak output current	25°		2.2		A

\* Pulse testing techniques are used to maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately.

\*\* This specification applies only for dc power dissipation permitted by absolute maximum ratings.



## Recommended Operating Conditions

Parameter		Min	Max	Units
Input voltage $V_i$	7805	7	25	V
	7806	8	25	
	7808	10.5	25	
	7885	10.5	25	
	7809	11.5	27	
	7810	12.5	28	
	7812	14.5	30	
	7815	17.5	30	
	7818	21	33	
	7820	23	36	
	7824	27	38	
	7827	30	40	
Output current, $I_O$			1.5	
A Operating virtual junction temperature, $T_J$		0	125	$^{\circ}\text{C}$

## Device Selection Guide

Device	Output Voltage
7805	5V
7806	6V
7808	8V
7885	8.5V
7809	9V
7810	10V
7812	12V
7815	15V
7818	18V
7820	20V
7824	24V
7827	27V

## 6.2: ATMEGA16

### Features:

- High-performance, Low-power AVR<sup>®</sup> 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 16K Bytes of In-System Self-Programmable Flash
    - Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - 512 Bytes EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits

through the JTAG Interface

- Peripheral Features

- Two 8-bit Timer/Counters with Separate Rescales and Compare Modes
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Real Time Counter with Separate Oscillator
- Four PWM Channels
- 8-channel, 10-bit ADC
  - 8 Single-ended Channels
  - 7 Differential Channels in TQFP Package Only
  - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
- Byte-oriented Two-wire Serial Interface
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator

- Special Microcontroller Features

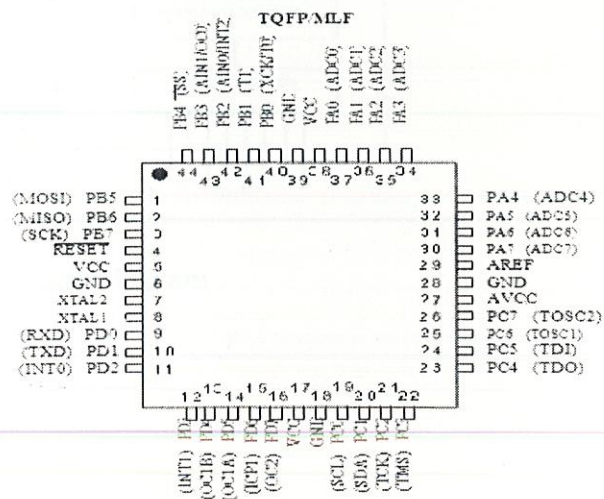
- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated RC Oscillator
- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby

- I/O and Packages



- 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
- Operating Voltages
    - 2.7 - 5.5V for ATmega16L
    - 4.5 - 5.5V for ATmega16
  - Speed Grades
    - 0 - 8 MHz for ATmega16L
    - 0 - 16 MHz for ATmega16
  - Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
    - Active: 1.1 mA
    - Idle Mode: 0.35 mA
    - Power-down Mode: < 1  $\mu$ A

**(Fig. 6.3)**



[illegible]

75



### 6.3: 8086/8088 Device Specifications

- Both are packaged in DIP (Dual In-Line Packages).

8086: 16-bit microprocessor with a 16-bit data bus

8088: 16-bit microprocessor with an 8-bit data bus.

- Both are 5V parts:

8086: Draws a maximum supply current of 360mA.

8088: Draws a maximum supply current of 340mA.

80C86/80C88: CMOS version draws 10mA with temp spec -40 to 225degF.

- Input/output current levels:

- Yields a 350mV noise immunity for logic 0 (Output max can be as high as 450mV while input max can be no higher than 800mV).
- This limits the loading on the outputs.

INPUT			OUTPUT		
Logic level	Voltage	Current	Logic level	Voltage	Current
0	0.8V max	+/- 10uA max	0	0.45V max	+2mA max
1	2.0V min	+/- 10uA max	1	2.4V min	- 400uA max

- Yields a 350mV noise immunity for logic 0 (Output max can be as high as 450mV while input max can be no higher than 800mV).
- This limits the loading on the outputs.

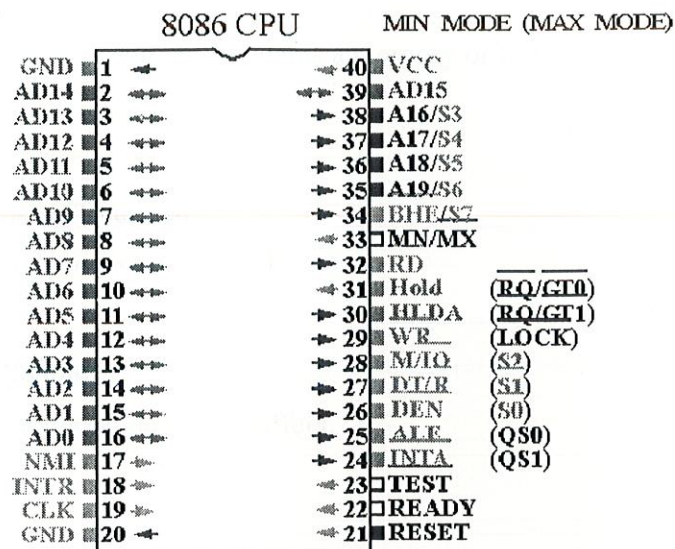
## DC Characteristics Recommended Logic Fan-out

### DC Characteristics Recommended Logic Fan-out

**TABLE 9-3** Recommended fan-out from any 8086/8088 pin connection.

<i>Family</i>	<i>Sink Current</i>	<i>Source Current</i>	<i>Fan-out</i>
TTL (74)	-1.6 mA	40 $\mu$ A	1
TTL (74LS)	-0.4 mA	20 $\mu$ A	5
TTL (74S)	-2.0 mA	50 $\mu$ A	1
TTL (74ALS)	-0.1 mA	20 $\mu$ A	10
TTL (74AS)	-0.5 mA	25 $\mu$ A	10
TTL (74F)	-0.5 mA	25 $\mu$ A	10
CMOS (74HC)	-10 $\mu$ A	10 $\mu$ A	10
CMOS (CD4)	-10 $\mu$ A	10 $\mu$ A	10
NMOS	-10 $\mu$	10 $\mu$ A	10

8086/88 Pin out



(Fig. 6.5) 8086/88 Pin out

### Pin functions:

#### AD15-AD0

Multiplexed address (ALE=1)/data bus(ALE=0).

#### A19/S6-A16/S3 (multiplexed)

High order 4 bits of the 20-bit address OR status bits S6-S3.

#### M/IO

Indicates if address is a Memory or IO address.

#### RD

When 0, data bus is driven by memory or an I/O device.

#### WR

Microprocessor is driving data bus to memory or an I/O device. When 0, data bus contains valid data.

ALE (Address latch enable)

When 1, address data bus contains a memory or I/O address.

DT/R (Data Transmit/Receive)

Data bus is transmitting/receiving data.

DEN (Data bus Enable)

Activates external data bus buffers.

### 8086/88 Pin out

- Pin functions:

S7, S6, S5, S4, S3, S2, S1, S0

- S7: Logic 1, S6: Logic 0.
- S5: Indicates condition of IF flag bits.
- S4-S3: Indicate which segment is accessed during current bus cycle:

S4	S3	Function
0	0	Extra segment
0	1	Stack segment
1	0	Code or no segment
1	1	Data segment

- S2, S1, S0 : Indicate function of current bus cycle (decoded by 8288).



$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Function	$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Function
0	0	0	Interrupt Ack	1	0	0	Opcode Fetch
0	0	1	I/O Read	1	0	1	Memory Read
0	1	0	I/O Write	1	1	0	Memory Write
0	1	1	Halt	1	1	1	Passive

## 8086/88 Pin out

Pin functions:

### INTR

When  $IF=1$ , microprocessor prepares to service interrupt.  $INTA$  becomes active after current instruction completes

### INTA

Interrupt Acknowledge generated by the microprocessor in response to  $INTR$ . Causes the interrupt vector to be put onto the data bus.

### NMI

Non-maskable interrupt. Similar to  $INTR$  except  $IF$  flag bit is not consulted and interrupt is vector 2.

### CLK

Clock input must have a duty cycle of 33% (high for  $1/3$  and low for  $2/3$ s)

### VCC/GND

Power supply (5V) and GND (0V).



## 8086/88 Pin out

Pin functions:

MN/ MX

Select minimum (5V) or maximum mode (0V) of operation.

BHE

Bus High Enable. Enables the most significant data bus bits (D 15 -D 8 ) during a read or write operation.

READY

Used to insert wait states (controlled by memory and IO for reads/writes) into the microprocessor.

RESET

Microprocessor resets if this pin is held high for 4 clock periods.

Instruction execution begins at FFFF0H and IF flag is cleared.

TEST

An input that is tested by the WAIT instruction.

Commonly connected to the 8087 coprocessor.

## 8086/88 Pin out

Pin functions:

### HOLD

Requests a direct memory access (DMA). When 1, microprocessor stops and places address, data and control bus in high-impedance state.

### HLDA (Hold Acknowledge)

Indicates that the microprocessor has entered the hold state.

### RO/GT1 and RO/GT0

Request/grant pins request/grant direct memory accesses (DMA) during maximum mode operation.

### LOCK

Lock output is used to lock peripherals off the system.

Activated by using the LOCK: prefix on any instruction.

### QS1 and QS0

The queue status bits show status of internal instruction queue. Provided for access by the numeric coprocessor (8087).

## 6.4: STEPPER MOTOR

Motion Control, in electronic terms, means to accurately control the movement of an object based on either speed, distance, load, inertia or a combination of all these factors. There are numerous types of motion control systems, including; Stepper Motor, Linear Step Motor, DC Brush, Brushless, Servo, Brushless Servo and more.

A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. Stepper motor is a form of ac. motor. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motor's rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied [39].

For every input pulse, the motor shaft turns through a specified number of degrees, called a step. Its working principle is one step rotation for one input pulse. The range of step size may vary from 0.72 degree to 90 degree. In position control application, if the number of input pulses sent to the motor is known, the actual position of the driven job can be obtained.

A stepper motor differs from a conventional motor (CM) as under:

- a. Input to SM is in the form of electric pulses whereas input to a CM is invariably from a constant voltage source.
- b. A CM has a free running shaft whereas shaft of SM moves through angular steps.
- c. In control system applications, no feedback loop is required when SM is used but a feedback loop is required when CM is used.
- d. A SM is a digital electromechanical device whereas a CM is an analog electromechanical device [40].

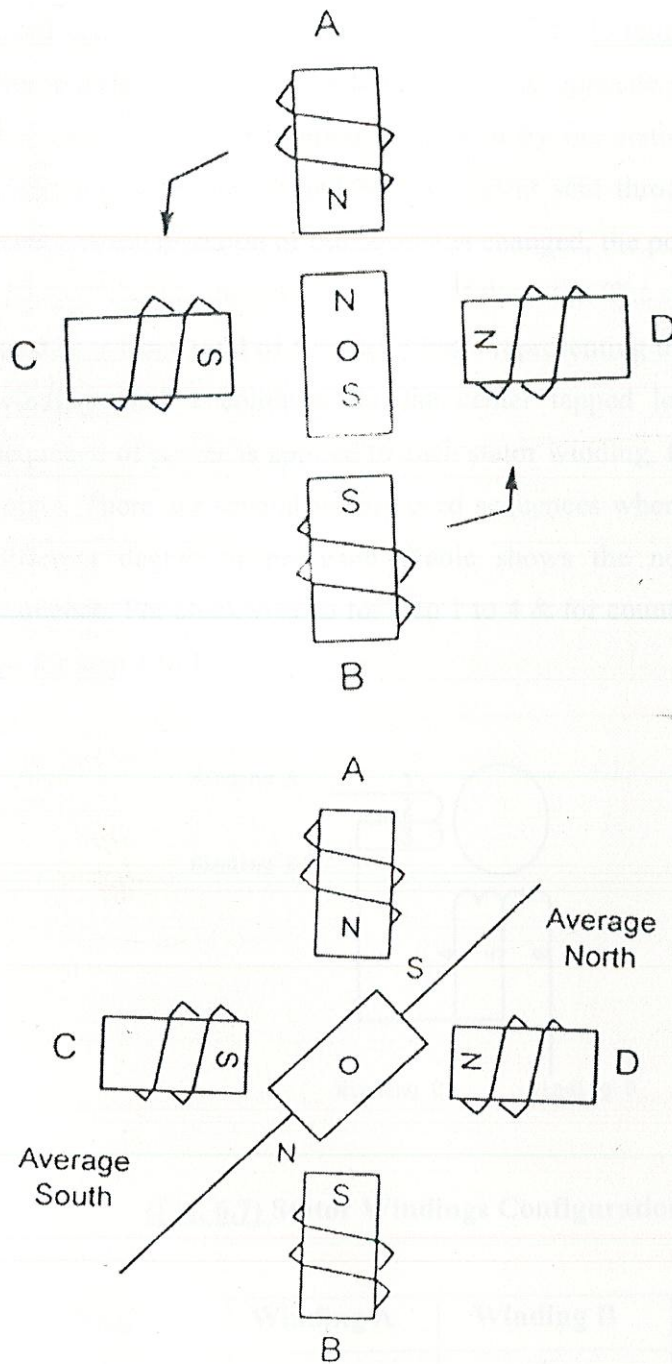


## Open Loop Operation

One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system. Open loop control means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. Control position is known simply by keeping track of the input step pulses [39].

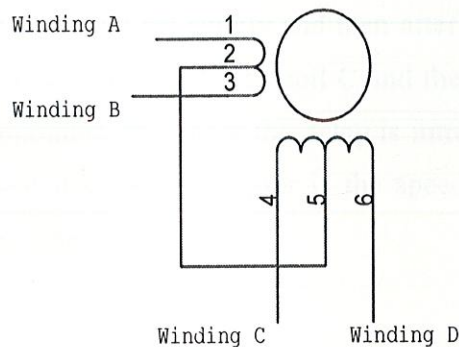
Every stepper motor has a permanent magnet rotor (shaft) surrounded by a stator. The most common stepper motor has four stator windings that are paired with a center-tapped common. This type of stepper motor is commonly referred to as a four- phase stepper motor. The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator. Notice that while a conventional motor shaft runs freely, the stepper motor shaft moves in a fixed repeatable increment which allows one to move it to a precise position. This repeatable





**(Fig. 6.6) Rotor Alignment**

fixed movement is possible as a result of basic magnetic theory where poles of the Same polarity repel and opposite poles attract. The direction of the rotation is dictated by the stator poles. The stator poles are determined by the current sent through the wire coils. As the direction of the current is changed, the polarity is also changed causing the reverse motion of the rotor. The stepper motor used here has a total of 5 leads: 4 leads representing the four stator windings and 1 common for the center tapped leads. As the sequence of power is applied to each stator winding, the rotor will rotate. There are several widely used sequences where each has a different degree of precision. Table shows the normal 4-step sequence. For clockwise go for step 1 to 4 & for counter clockwise go for step 4 to 1.



**(Fig. 6.7) Stator Windings Configuration**

Step	Winding A	Winding B	Winding C	Winding D
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

#### Input Sequence to the Windings

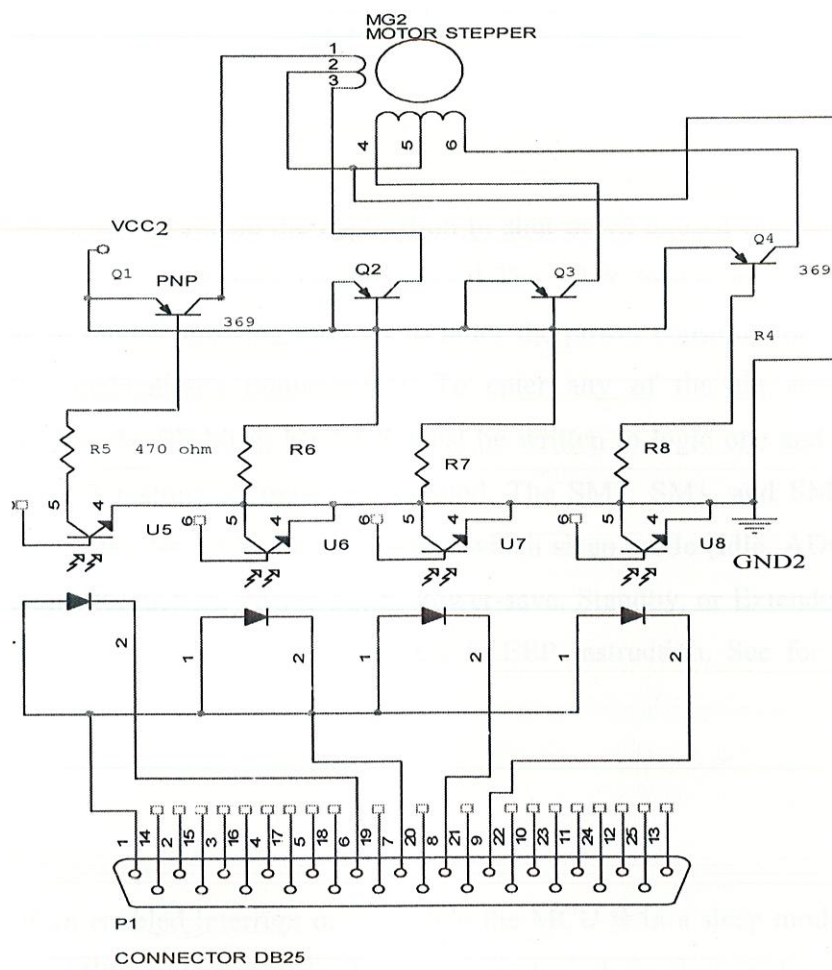
### **Step Angle & Steps per Revolution**

Movement associated with a single step, depends on the internal construction of the motor, in particular the number of teeth on the stator and the rotor. The step angle is the minimum degree of rotation associated with a single step.

Step per revolution is the total number of steps needed to rotate one complete rotation or 360 degrees (e.g., 180 steps \* 2 degree = 360) [31].

Since the stepper motor is not ordinary motor and has four separate coils, which have to be energized one by one in a stepwise fashion. We term them as coil A, B, C and D. At a particular instant the coil A should get supply and then after some delay the coil B should get a supply and then coil C and then coil D and so on the cycle continues. The more the delay is introduced between the energizing of the coils the lesser is the speed of the stepper motor and vice versa.





**(Fig. 6.8)**



## **CHAPTER 7: POWER MANAGEMENT AND SLEEP MODES**

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements. To enter any of the six sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the MCUCR Register select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, Standby, or Extended Standby) will be activated by the SLEEP instruction. See for a summary.

If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, it executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a Reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector. The presents the different clock systems in the ATmega16, and their distribution.

### **7.1: Project working and implementation**

In this project we have design the following circuit n the Orcad capture a it provide very easiness to make it as in which there are

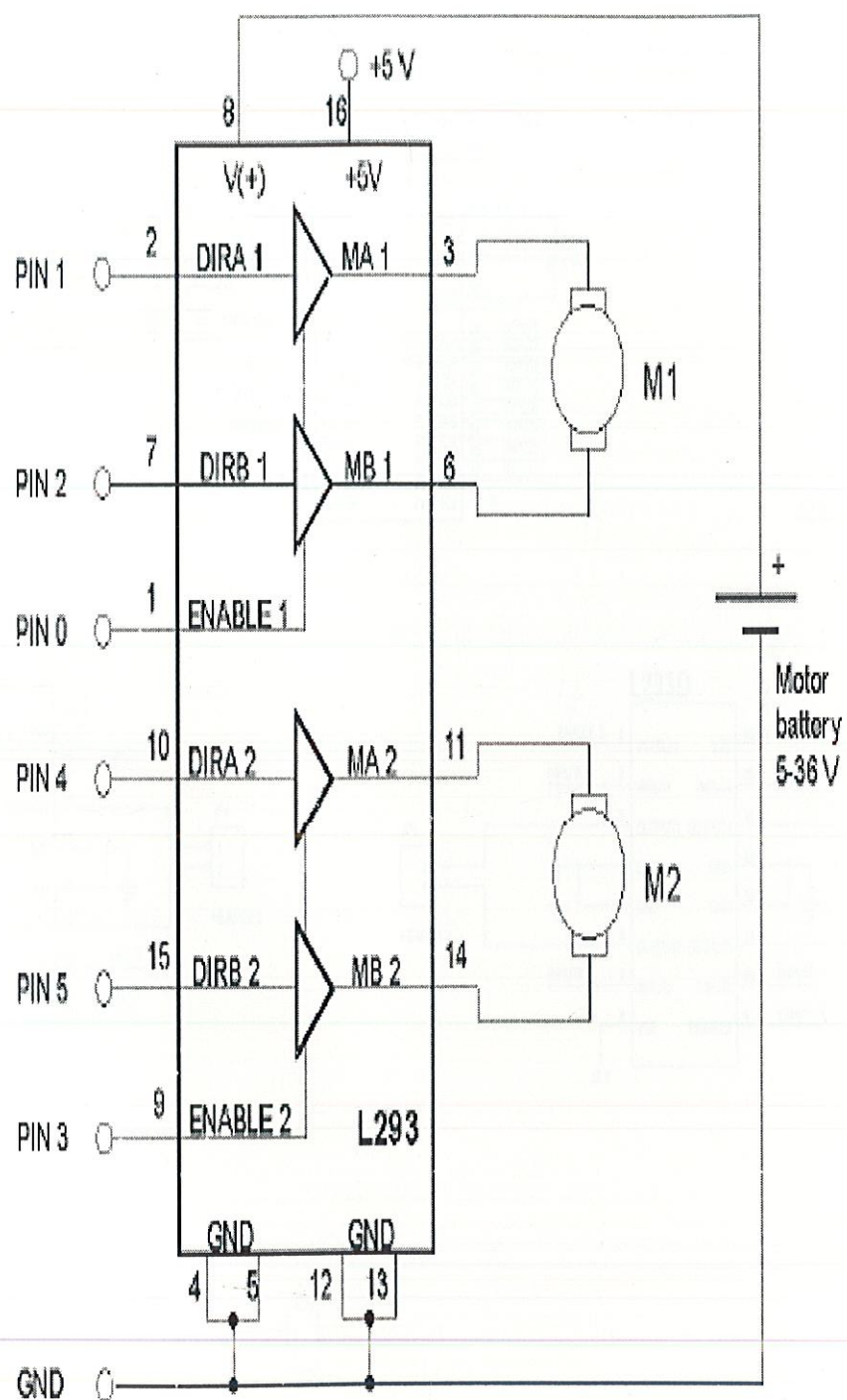
very special tools which help in designing the PCB but in our project we just deigned our project schematic in the Orcad.

In our project we use driver circuit because as microcontroller is not capable of driving the motor as it can source and sink only 20 milliamp here current .so we have to use the driver circuit which help the motor to provide the required voltage and current to make it run well. So in this circuit we use L298D driver IC which have the control pins which help us in controlling the direction of the motor, and these are as given blow the control signal provided for their control.

ENABLE	DIRA	DIRB	Function
H	H	L	Turn right
H	L	H	Turn left
H	L/H	H/L	Fast stop
L	either	either	Slow stop

The keypad is used to send the control signal to the microcontroller to control the direction of the motors .so when we

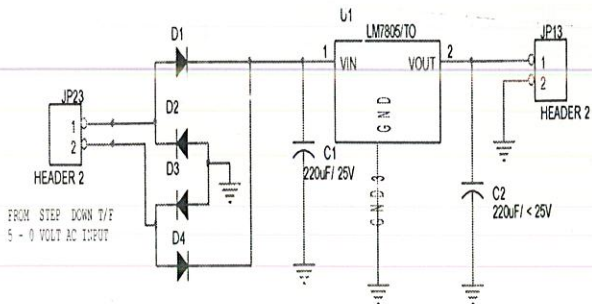
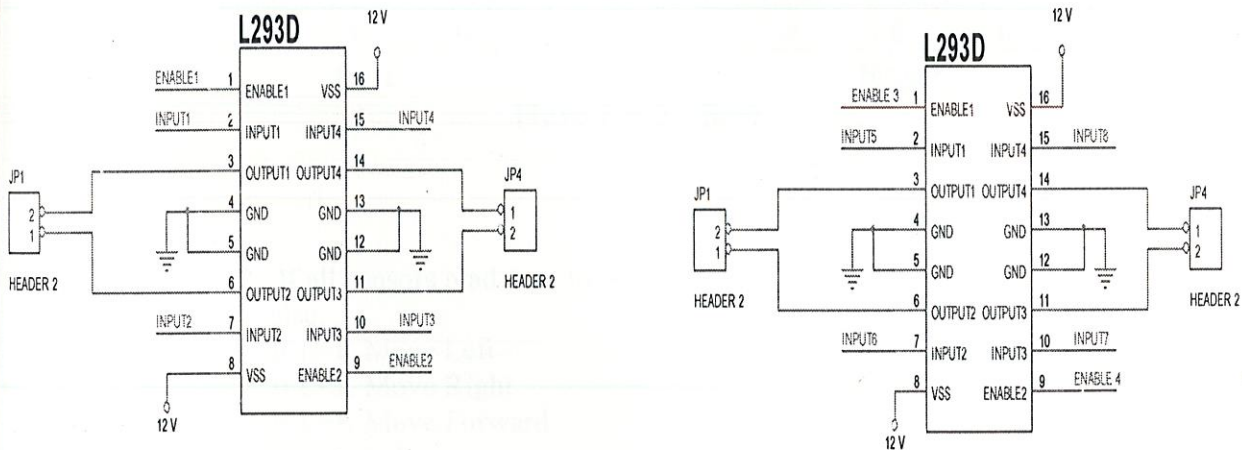
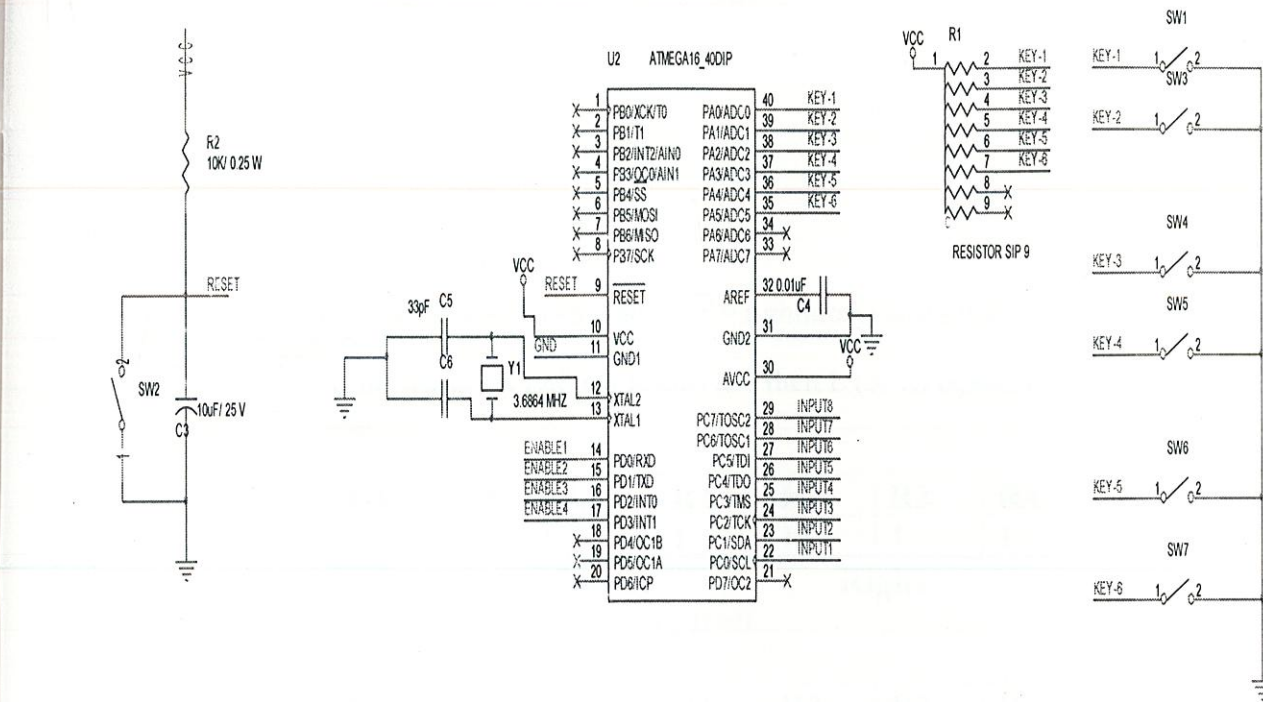
press the key the microcontroller get low signal at its keys and which is normally at high. So keep on detecting the low signal on the pin of the controller and whenever we press the key the controller send the required signal to the driver L298 and which make work according to the control signal we give it to them. The driver L298 will work up to 40 V, 1.5 ampere and it is the basic requirement of the motor to work on.



(Fig. 7.1)



**(Fig. 7.2) Pin diagram of the driver L298D**



## CHAPTER 8: ALGORITHM

### ALGORITHM

1. L= leftmost sensor which reads 0; R= rightmost sensor which reads 0.

If no sensor on Left (or Right) is 0 then L (or R) equals 0;  
Eg.

L4	L3	L2	L1	R1	R2	R3	R4
1	0	0	1	1	1	1	1

Left

Center

Right

Here L=3 , R=0

L4	L3	L2	L1	R1	R2	R3	R4
1	1	0	0	0	0	0	0

Left

Center

Right

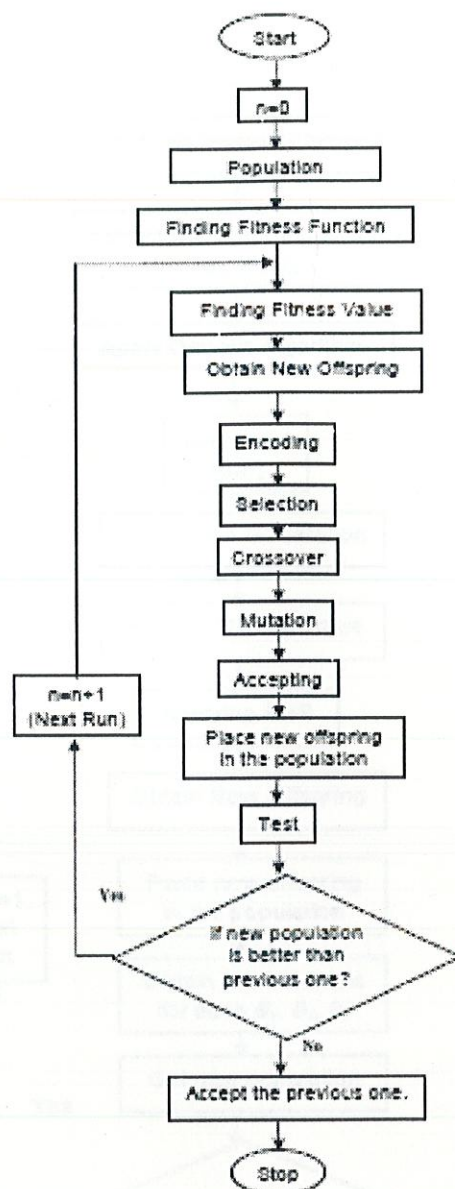
Here L= 2 , R=4

2. If all sensors read 1 go to step 3,  
else,  
If L>R Move Left  
If L<R Move Right  
If L=R Move Forward  
Goto step 4
3. Move Clockwise if line was last seen on Right  
Move Counter Clockwise if line was last seen on Left  
Repeat step 3 till line is found.
4. Goto step 1.

## GENETIC ALGORITHM

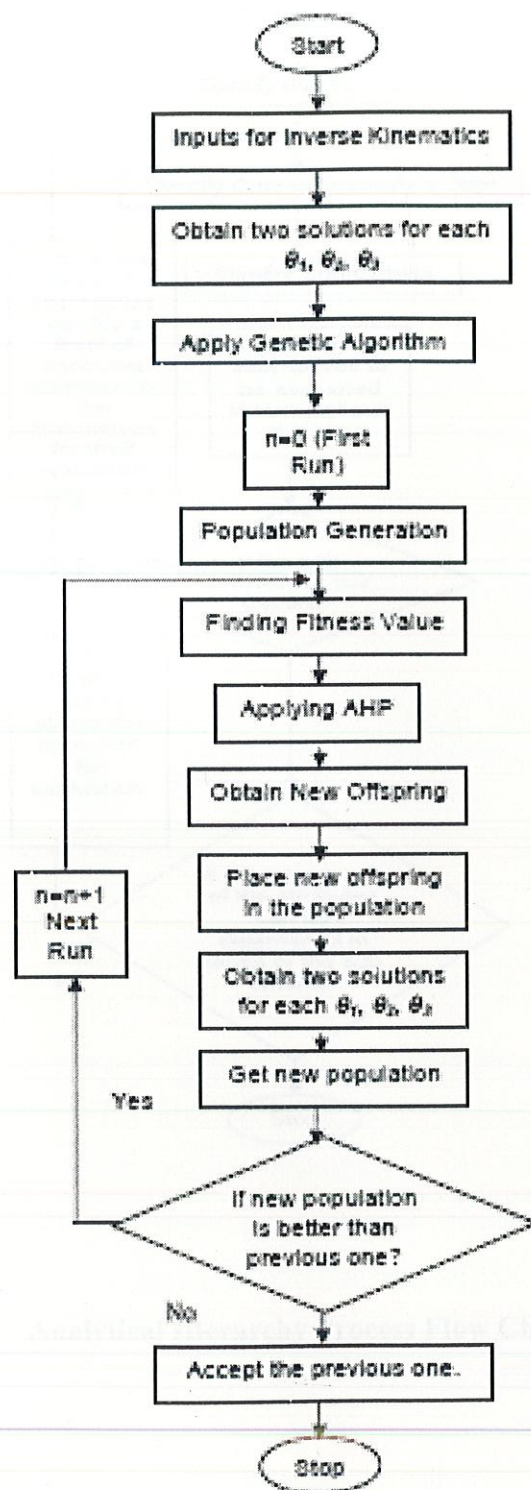
Now these results are fed to the Genetic Algorithm for generating the population of chromosomes having optimized values.

- (1) **Start**     Generate random population of  $n$  chromosomes (suitable solutions for the problem).
- (2) **Fitness**   Evaluate the fitness  $f(x)$  of each chromosome  $x$  in the population.
- (3) **New population**   Create a new population by repeating following steps until the new population is complete.
  - (a). **Selection.**   Select two parent hromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
  - (b) . **Crossover.**   With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
  - (c) . **Mutation.**   With a mutation probability, mutate new offspring at each locus (position in chromosome).
  - (d) . **Accepting.**   Place new offspring in the new population.
- (4) **Replace**     Use new generated population for a further run of the algorithm.
- (5) **Test**        If the end condition is satisfied, stop, and return the best solution in current population.
- (6) **Loop**        Go to step 2.

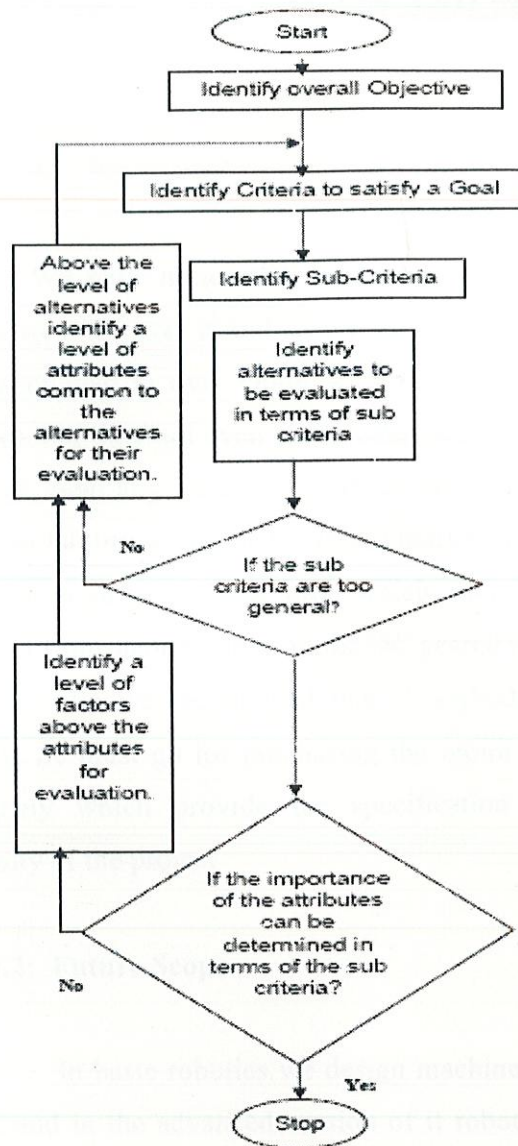


**Genetic Algorithm Flow Chart**





System Flow Chart



**Analytical Hierarchy Process Flow Chart**

## **CHAPTER 9 : DISCUSSION AND FUTURE SCOPE**

### **9.1: Discussion**

We have notice that in our project we not got standard motor to work over it and so we go lot of problem regarding the lifting of load in over motor we also use stepper motor for our project. but they not even move when we use it and when we not provide them any load they rotate so we come to know that their non availability of the motor in the market which help us making the project in the perfect way, which also make us change our concept from stepper motor to the DC geared motor.

So we recommend that if anybody like to make that project he must go for purchasing the motor online from foreign company which provide the specification and load carrying capacity of the project

### **9.2: Future Scope**

In basic robotics we design machines to do the specified tasks and in the advanced version of it robots are designed to be adaptive, that is, respond according to the changing environment and even autonomous, that is, capable to make decisions on their own. While designing a robot the most important thing to be taken in consideration is, obviously, the function to be performed. Here comes into play the discussion about the scope of the robot and robotics. Robots have basic levels of complexity and each level has its scope for performing the requisite function.

The levels of complexity of robots is defined by the members used in its limbs, number of limbs, number of actuators and sensors used and for advanced robots the type and number of microprocessors and microcontrollers used. Each increasing component adds to the scope of functionality of a robot. With every joint added, the degrees of freedom in which a robot can work increases and with the quality of the microprocessors and microcontrollers the accuracy and effectiveness with which a robot can work is enhanced.



## **Chapter 10 : Conclusion**

I hope that you enjoyed this robotic arm unit. Not only did you apply math and science to your project, but you also learned the importance of shop safety and working together with your team mates. Each team mate had a job to do. Engineers were in charges of the final design and oversight of building your robot arm. Accountants were in charge of your business expenses. Finally Project Directors were in charge of your teams' public affairs and overall completion of your project. Hopefully you were able to see how there is more to just having the best design. Working together as a team is just as important in accomplishing the different challenges you may face in your life. I hope that you will be able to apply the methods and ideas from this project in your day to day lives.

## **Chapter 11 : References**

- (1) Craig J.J, Introduction to Robotics: Mechanics and control, Pearson Education Asia , 2004.
- (2) Fu K.S, Gonzalez R.C., Lee C.S.G., Robotics : Control, Sensing , Vision and Intelligence. Mc-Graw Hill International Editions , 1987.
- (3) Schilling R.J., Fundamental of Robotics: Analysis and Control, Prentice Hall if India Pvt. Ltd., 2002.
- (4) Deb S.R., Robotics Technology and Flexible Automation. Tata Mc-GrawHill , 2002.
- (5) Seshadri V., Jankiraman P.A., Nagarajan T., Robotics – Principles and Applications. AICTE Updates, code no 382.
- (6) Whitley Darrell, A Genetic Algorithm Tutorial, Computer Science Department, Colorado State University Fort Collins, CO 80523.
- (7) P. Dupont , “Friction Modeling in Dynamic Robot Simulation” Proceedings 1990 IEEE Int. Robotics and Automation Conf., Cincinnati, OH, May, 1990.
- (8) W. Townsend and K. Salisbury, “Mechanical Bandwidth as a Guideline to High-Performance Manipulator Design” Proceedings 1990 IEEE Int. Robotics and Automation Conf., Scottsdale, April, 1989.
- (9) Lee G. L. and Goldenberg A. A., “Comparative Study of Robust Saturation Control of Robot Manipulator: Analysis and Experiments,” International Journal of Robotics Research, Vol. 15, No. 5, 1996, pp. 473-491.
- (10) Wikipedia also helps us a lot.