## V.S.R GOVT DEGREE&P.G COLLEGE, MOVVA

## DEPARTMENT OF ELECTRONICS

V.S.R. GOVT.
DEGREE & P.G. COLLEGE

# SOLAR TRACKING SYSTEM

**A dissertation work Submitted to the department of Electronics In partial fulfilment for the award of under graduate degree in Electronics B.Sc. course - 2016-2019.**

**by**

**E.S.PHANINDRA(Y163223035)**
**G.SANDEEP(Y163223036)**
**K.L.S.V.RAO(Y163223037)**
**K.V.S.RAO(Y163223038)**
**K.D.SAI KRISHNA(Y163223039)**

Department of Electronics
Govt.

**Under the Guidance of**

**Smt.S.KIRANMAYI, M.Sc.,M.Phil.,**NET.,
Lecturer, Department of Electronics

V.S.R. GOVT.
DEGREE & P.G. COLLEGE

A

PROJECT REPORT ON

# SOLAR TRACKING SYSTEM

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE AWARD OF DEGREE OF

BACHELOR OF SCIENCE IN ELECTRONICS

Submitted to the Krishna university

By

K.L.SURYA VARDHAN RAO

Y163223037

Under The Guidance of

S.KIRANMAYI M.Sc.,M.Phil.,NET

(Lecturer in Electronics)

DEPARTMENT OF ELECTRONICS

V.S.R GOVT DEGREE & P.G COLLEGE,MOVVA

V.S.R GOVT DEGREE & P.G COLLEGE,MOVVA

## DEPARTMENT OF ELECTRONICS

V.S.R. GOVT.
DEGREE & P.G. COLLEGE

## CERTIFICATE

This is to Certified that the Project Work

Entitled "SOLAR TRACKING SYSTEM" is a Bonafide work

Carried out by-K.L.Suryavardhan Rao--Regd No-Y163223037

-

Year--2018-2019, in partial fulfilment for the Award of

Undergraduation in ELECTRONICS, of KRISHNA UNIVERSITY , Machilipatnam ,during the Year 2018-19.

It is certified that all Corrections/ Suggestions indicated for Internal Assesment have been incorporated in the report .This project report have been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor Degree in ELECTRONICS.

**Signature of the Guide**

**Signature of the HOD**

**Signature of the Principal**

**Signature of the Examiners**

1)
2) K.Kranth

# ACKNOWLEDGEMENTS

We are the Students of III B.SC(M.E.Cs) Electronics Cluster in V.S.R Govt Degree & P.G College,Movva are preparing a final year project **"SOLAR TRACKING SYSTEM". We Whole heartedly express our Sincere Gratitude to Smt. S.KIRANMAYI who guided us for the completion of this final year project.We are also thankful to our principal Smt V.Rama Jyotshna Kumari garu for giving us an opportunity to do this project.**

**E.S.PHANINDRA(Y163223035)**
**G.SANDEEP(Y163223036)**
**K.L.S.V.RAO(Y163223037)**
**K.V.S.RAO(Y163223038)**
**K.D.SAI KRISHNA(Y163223039)**

Department of Electronics
MOVVA
Govt. Degree [illegible]

# DECLARATION

We E.S.PHANINDRA(Y163223035), G.SANDEEP(Y163223036), K.L.S.V.RAO(Y163223037), K.V.S.RAO(Y163223038), K.D.SAI KRISHNA(Y163223039) students of 6 Semester B.SC in ELECTRONICS, V.S.R Govt Degree & P.G College,Movva here by declare that the project work Entitled **"SOLAR TRACKING SYSTEM" Submitted to the Krishna University,Machilipatnam during the academic year 2018-19 ,is aRecord of original work done by us under the Guidance of Smt S.KIRANMAYI, Lecturer in Department Electronics,V.S.R Govt Degree & p.g College, Movva .This project work is** SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT for the Award of Degree of **BACHELOR OF SCIENCE IN ELECTRONICS. This results and works Embodied in this Thesis have not been submitted to any other University (or) Institute for the award of any Degree.**

Department of Electronics
MOVVA
Govt. Degree College

K.L.Suryavardhan

**STUDENT NAME**

Date: 20/04/2019

Place: MOVVA

# INDEX

# INDEX

| S.NO | Name of the Topic | Page No |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

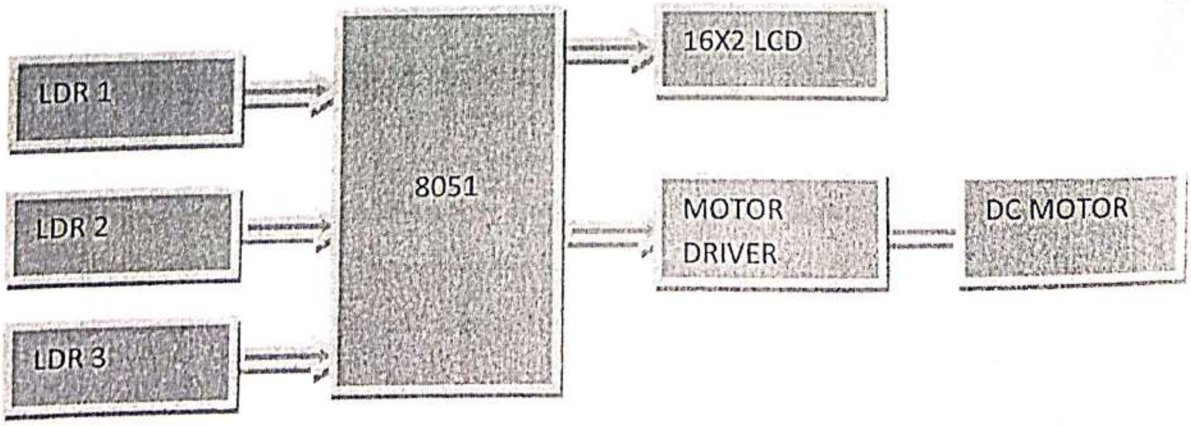
# SOLAR TRACKING SYSTEM

## ABSTRACT:

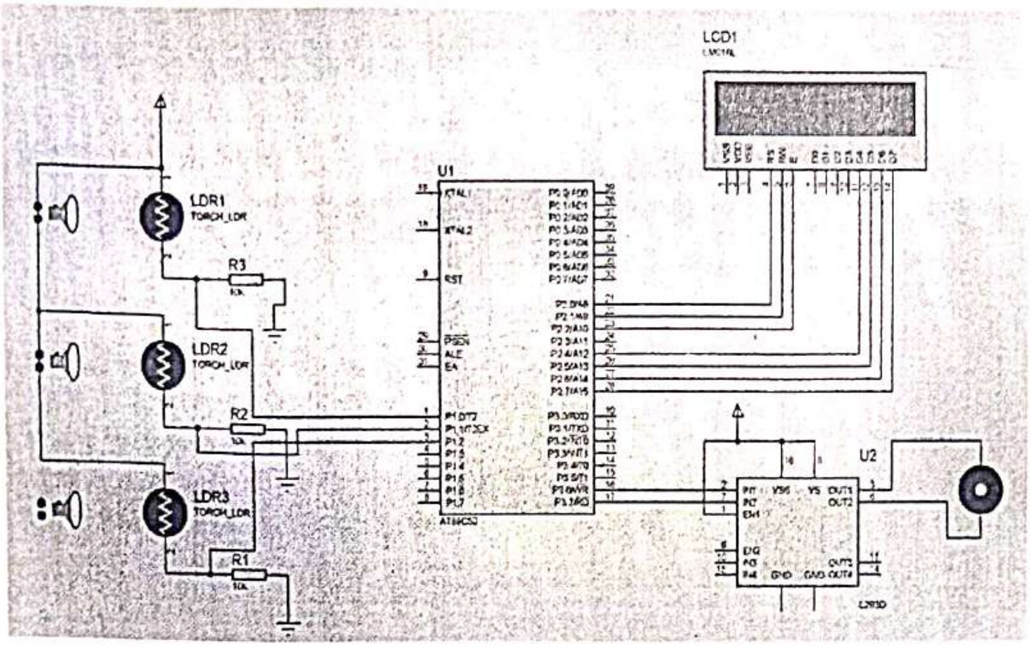
In this project we design the system for tracking the solar panel automatically. Solar tracking is a process of catching up of sunlight for various applications. It is transistor based system . we have used the comparator LM 358 for automatic tracking operation. Two LDR (Light Dependant Resistor) Sensors R1 and R2 are used. When sun rays falls on R1 Motor rotates clockwise direction. When sun rays falls on R 2 Motor rotates anti clockwise direction.

The solar panel converts the incident light of sun into electric signals and then the electricity produced is stored in the 6v rechargeable battery. This stored electricity can be used to glow light or to work any 6v electronic/electric device. Electricity produced from photovoltaic cells does not result in air or water pollution, deplete natural resources, or endanger animal or human health. Small-scale solar plants can take advantage of unused space on roof tops of existing buildings. By using solar tracking system we will get a more surplus energy than fixed photovoltaic cell. The only potential negative impacts are associated with some toxic chemicals, like cadmium and arsenic, that are used in the production process of photovoltaic cell. These environmental impacts are minor and can be easily controlled through recycling and proper disposal.

## BLOCK DIAGRAM:



## CIRCUIT :



### THE MICROCONTROLLER:

A microcontroller is a general purpose device, but that is meant to read data, perform limited calculations on that data and control its environment based on those calculations. The prime use of a microcontroller is to control the operation of a machine using a fixed program that is stored in ROM and that does not change over the lifetime of the system.

The microcontroller design uses a much more limited set of single and double byte instructions that are used to move data and code from internal memory to the ALU. The microcontroller is concerned with getting data from and to its own pins; the architecture and instruction set are optimized to handle data in bit and byte size.

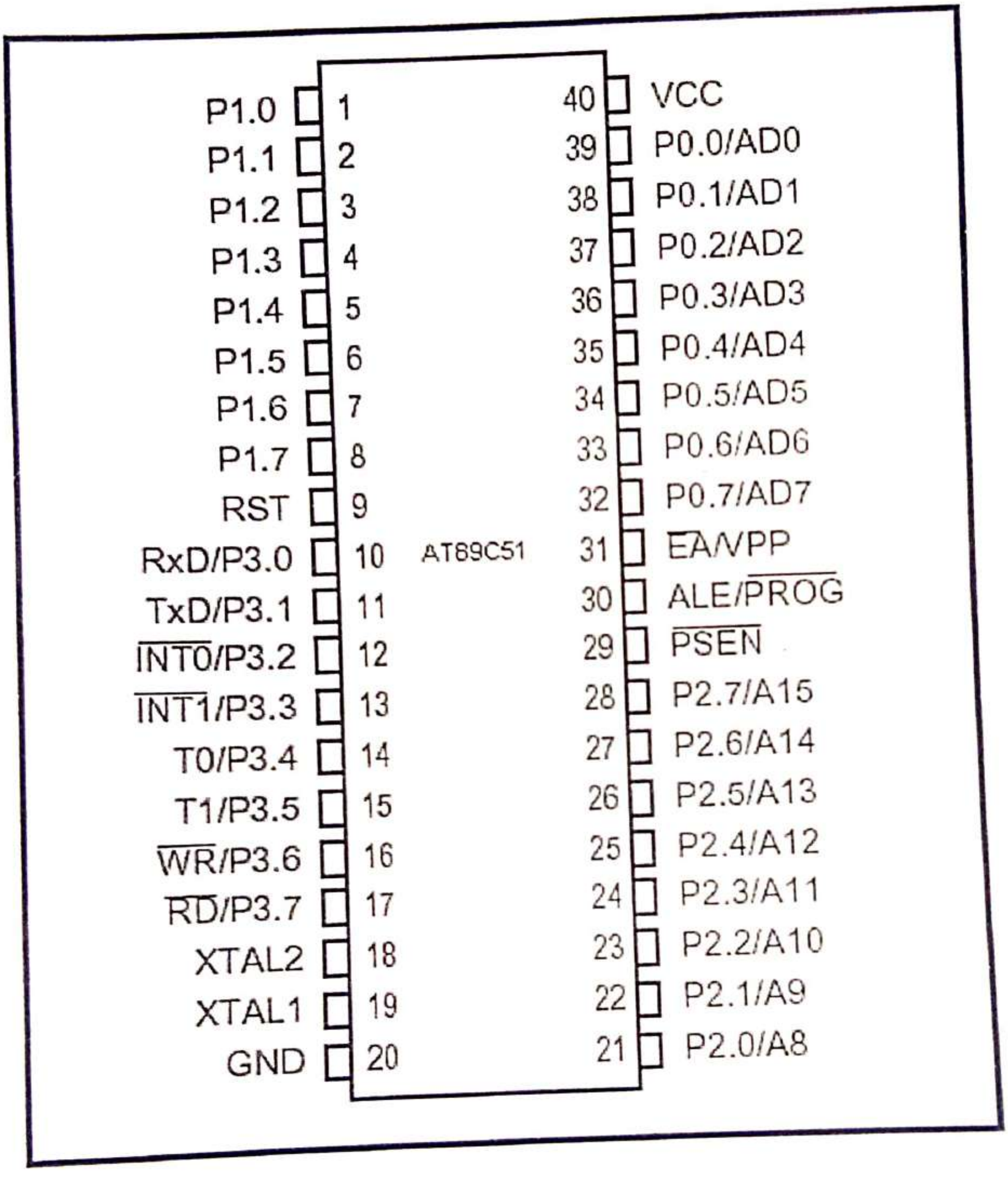
The AT89C51 is a low-power, high-performance CMOS 8-bit microcontroller with 4k bytes of Flash Programmable and erasable read only memory (EROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is functionally compatible with the industry-standard 80C51 microcontroller instruction set and pin out. By combining versatile 8-bit CPU with Flash on a monolithic chip, the Atmel's AT89c51 is a powerful microcomputer, which provides a high flexible and cost-effective solution to many embedded control applications.

## AT89C51 MICROCONTROLLER

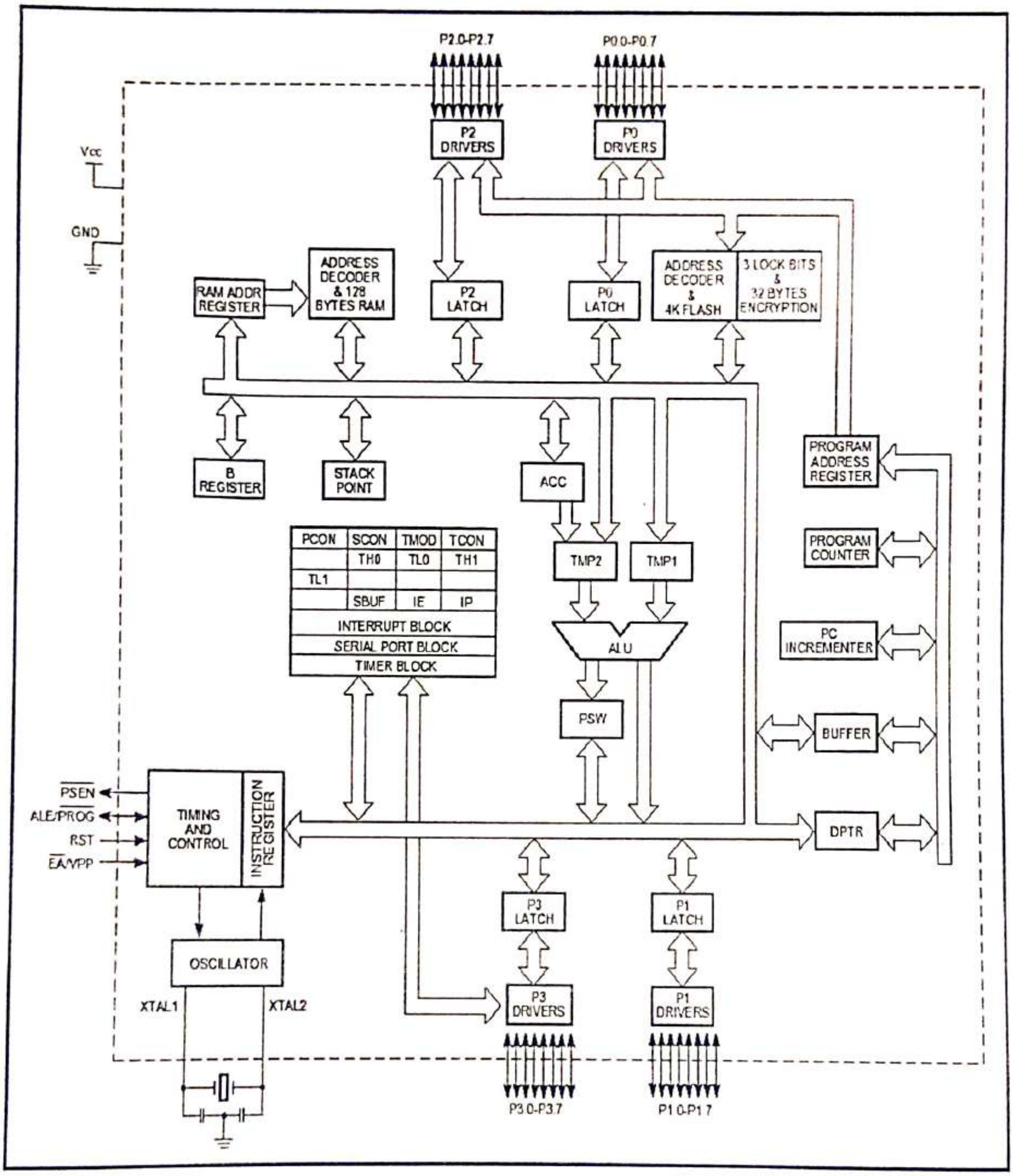
### FEATURES

- 80C51 based architecture
- 4-Kbytes of on-chip Reprogrammable Flash Memory
- 128 x 8 RAM
- Two 16-bit Timer/Counters
- Full duplex serial channel
- Boolean processor
- Four 8-bit I/O ports, 32 I/O lines
- Memory addressing capability
    - 64K ROM and 64K RAM
- Power save modes:
    - Idle and power-down
- Six interrupt sources
- Most instructions execute in 0.3 us
- CMOS and TTL compatible
- Maximum speed: 40 MHz @ Vcc = 5V
- Industrial temperature available
- Packages available:
    - 40-pin DIP
    - 44-pin PLCC
    - 44-pin PQFP

**Pin configuration:**



## AT89C51 Block Diagram



## PIN DESCRIPTION:

### VCC

Supply voltage

### GND

Ground

### Port 0

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high impedance inputs.

Port 0 can also be configured to be the multiplexed low order address/data bus during access to external program and data memory. In this mode, P 0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pull-ups are required during program verification.

### Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The port 1output buffers can sink/source four TTL inputs. When 1s are written to port 1 pins, they are pulled high by the internal pull-ups can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (1) because of the internal pull-ups.

### Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The port 2 output buffers can sink/source four TTL inputs. When 1s are written to port 2 pins, they are pulled high by the internal pull-ups can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current because of the internal pull-ups.

Port 2 emits the high-order address byte during fetches from external program memory and during access to DPTR. In this application Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit data address (MOVX@R1), Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

**Port 3**

Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The port 3 output buffers can sink/source four TTL inputs. When 1s are written to port 3 pins, they are pulled high by the internal pull-ups can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current because of the internal pull-ups.

Port 3 also receives some control signals for Flash Programming and verification.

| Port pin | Alternate Functions |
|---|---|
| P3.0 | RXD(serial input port) |
| P3.1 | TXD(serial input port) |
| P3.2 | INT0(external interrupt 0) |
| P3.3 | INT1(external interrupt 1) |
| P3.4 | T0(timer 0 external input) |
| P3.5 | T1(timer 1 external input) |
| P3.6 | WR(external data memory write strobe) |
| P3.7 | RD(external data memory read strobe) |

RST

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

**ALE/PROG:**

Address Latch Enable is an output pulse for latching the low byte of the address during access to external memory. This pin is also the program pulse input (PROG) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/16 the oscillator frequency and may be used for external timing or clocking purpose. Note, however, that one ALE pulse is skipped during each access to external Data memory.

**PSEN**

Program Store Enable is the read strobe to external program memory when the AT89c51 is executing code from external program memory PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA /VPP

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

XTAL1

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

### XTAL 2

Output from the inverting oscillator amplifier.

## OPERATING DESCRIPTION

The detail description of the AT89C51 included in this description is:

- Memory Map and Registers
- Timer/Counters
- Interrupt System

## MEMORY MAP AND REGISTERS

### Memory

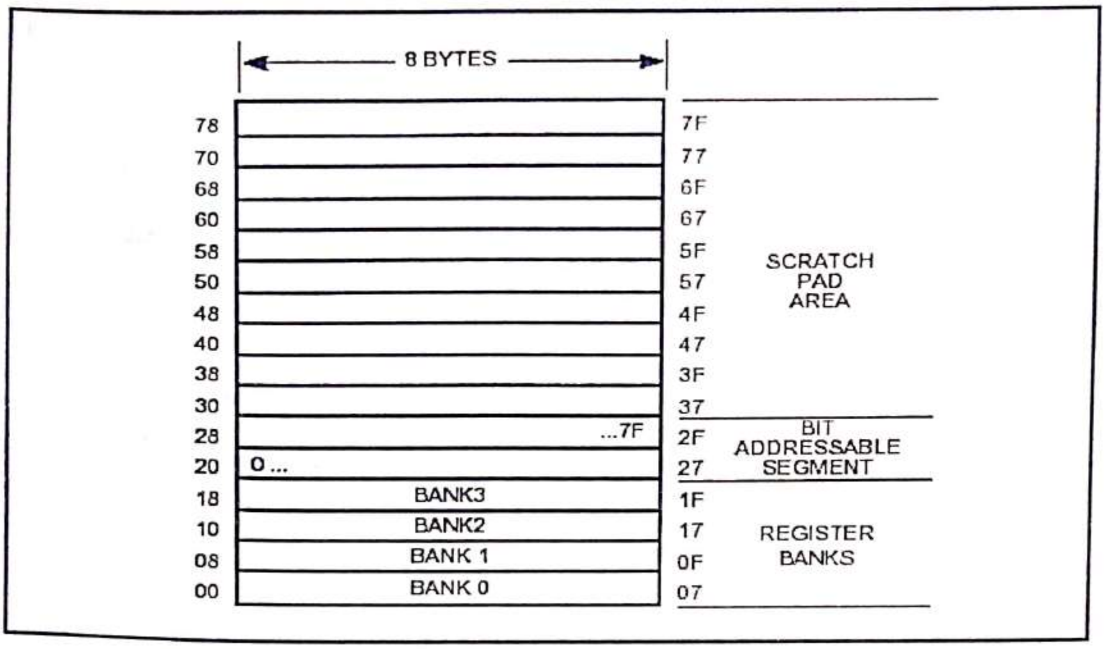
The AT89C51 has separate address spaces for program and data memory. The program and data memory can be up to 64K bytes long. The lower 4K program memory can reside on-chip. The AT89C51 has 128 bytes of on-chip RAM.

The lower 128 bytes can be accessed either by direct addressing or by indirect addressing. The lower 128 bytes of RAM can be divided into 3 segments as listed below

1. **Register Banks 0-3:** locations 00H through 1FH (32 bytes). The device after reset defaults to register bank 0. To use the other register banks, the user must select them in software. Each register bank contains eight 1-byte registers R0-R7. Reset initializes the stack point to location 07H, and is incremented once to start from 08H, which is the first register of the second register bank.

2. **Bit Addressable Area:** 16 bytes have been assigned for this segment 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH). Each of the 16 bytes in this segment can also be addressed as a byte.

3. **Scratch Pad Area:** 30H-7FH are available to the user as data RAM. However, if the data pointer has been initialized to this area, enough bytes should be left aside to prevent SP data destruction.



## SPECIAL FUNCTION REGISTERS:

The Special Function Registers (SFR's) are located in upper 128 Bytes direct addressing area. The SFR Memory Map in shows that.

Not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses in general return random data, and write accesses have no effect. User software should not write 1s to these unimplemented locations, since they may be used in future microcontrollers to invoke new features. In that case, the reset or inactive values of the new bits will always be 0, and their active values will be 1.

**The functions of the SFR's are outlined in the following sections.**

### Accumulator (ACC)

ACC is the Accumulator register. The mnemonics for Accumulator-specific instructions, however, refer to the Accumulator simply as A.

### B Register (B)

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

### Program Status Word (PSW)

The PSW register contains program status information.

### Stack Pointer (SP)

The Stack Pointer Register is eight bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

### Data Pointer (DPTR)

The Data Pointer consists of a high byte (DPH) and a low byte (DPL). Its function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

### Serial Data Buffer (SBUF)

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer, where it is held for serial transmission. (Moving a byte to SBUF initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

### Timer Registers

Register pairs (TH0, TL0) and (TH1, TL1) are the 16-bit Counter registers for Timer/Counters 0 and 1, respectively.

### Control Registers

Special Function Registers IP, IE, TMOD, TCON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port.

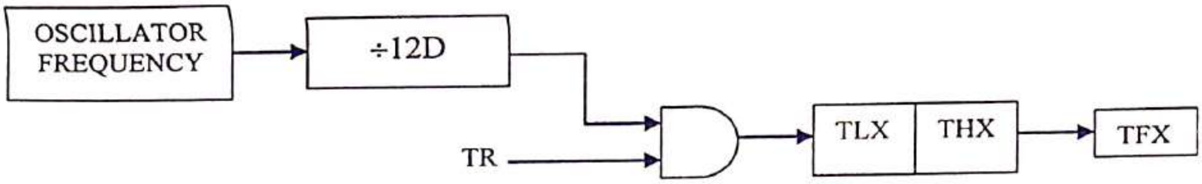
### TIMER/COUNTERS

The IS89C51 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. All two can be configured to operate either as Timers or event counters. As a Timer, the register is incremented every machine cycle. Thus, the register counts machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

As a Counter, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 and T1. The external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but it should be held for at least one full machine cycle to ensure that a given level is sampled at least once before it changes.

In addition to the Timer or Counter functions, Timer 0 and Timer 1 have four operating modes: 13-bit timer, 16-bit timer, 8-bit auto-reload, split timer.

TIMERS:



**SFR'S USED IN TIMERS**

The special function registers used in timers are,

- TMOD Register
- TCON Register
- Timer(T0) & timer(T1) Registers
-

**(i) TMOD Register:**

TMOD is dedicated solely to the two timers (T0 & T1).

- The timer mode SFR is used to configure the mode of operation of each of the two timers. Using this SFR your program may configure each timer to be a 16-bit timer, or 13 bit timer, 8-bit auto reload timer, or two separate timers. Additionally you may configure the timers to only count when an external pin is activated or to count "events" that are indicated on an external pin.
- It can consider as two duplicate 4-bit registers, each of which controls the action of one of the timers.

**(ii) TCON Register:**

- The timer control SFR is used to configure and modify the way in which the 8051's two timers operate. This SFR controls whether each of the two timers is running or stopped and contains a flag to indicate that each timer has overflowed. Additionally, some non-timer related bits are located in TCON SFR.

- These bits are used to configure the way in which the external interrupt flags are activated, which are set when an external interrupt occurs.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

### (iii) TIMER 0 (T0):

- TO (Timer 0 low/high, address 8A/8C h)

  These two SFR's taken together represent timer 0. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

| | |
|---|---|
| TH0 | TL0 |

### (iv) TIMER 1 (T1):

- T1 (Timer 1 Low/High, address 8B/ 8D h)

These two SFR's, taken together, represent timer 1. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is Configurable is how and when they increment in value.

| | |
|---|---|
| TH1 | TL1 |

The Timer or Counter function is selected by control bits C/T in the Special Function Register TMOD. These two Timer/Counters have four operating modes, which are selected by bit pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters, but Mode 3 is different.

The four modes are described in the following sections.

### Mode 0:

Both Timers in Mode 0 are 8-bit Counters with a divide-by-32 pre scalar. Figure 8 shows the Mode 0 operation as it applies to Timer 1. In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or INT1 = 1. Setting GATE = 1 allows the Timer to be controlled by external input INT1, to facilitate pulse width measurements. TR1 is a control bit in the Special Function Register TCON. Gate is in TMOD.

The 13-bit register consists of all eight bits of TH1 and the lower five bits of TL1. The upper three bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for Timer 0 as for Timer 1, except that TR0, TF0 and INT0 replace the corresponding Timer 1 signals. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

### Mode 1

Mode 1 is the same as Mode 0, except that the Timer register is run with all 16 bits. The clock is applied to the combined high and low timer registers (TL1/TH1). As clock pulses are received, the timer counts up: 0000H, 0001H, 0002H, etc. An overflow occurs on the FFFFH-to-0000H overflow flag. The timer continues to count. The overflow flag is the TF1 bit in TCON that is read or written by software.

### Mode 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 10. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves the TH1 unchanged. Mode 2 operation is the same for Timer/Counter 0.

### Mode 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0. Timer 0 in Mode 3 establishes TL0and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 11. TL0 uses the Timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function (counting machine cycles) and over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the Timer 1 interrupt.

Mode 3 is for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, the AT89C51 can appear to have three Timer/Counters. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3. In this case, Timer 1 can still be used by the serial port as a baud rate generator or in any application not requiring an interrupt.

## INTERRUPT SYSTEM

An interrupt is an external or internal event that suspends the operation of micro controller to inform it that a device needs its service. In interrupt method, whenever any device needs its service, the device notifies the micro controller by sending it an interrupt signal. Upon receiving an interrupt signal, the micro controller interrupts whatever it is doing and serves the device. The program associated with interrupt is called as **interrupt service subroutine (ISR)**.Main advantage with interrupts is that the micro controller can serve many devices.

### Baud Rate

The baud rate in Mode 0 is fixed as shown in the following equation. Mode 0 Baud Rate = Oscillator Frequency /12 the baud rate in Mode 2 depends on the value of the SMOD bit in Special Function Register PCON. If SMOD = 0 the baud rate is 1/64 of the oscillator frequency. If SMOD = 1, the baud rate is 1/32 of the oscillator frequency.

Mode 2 Baud Rate = 2SMODx (Oscillator Frequency)/64.

In the IS89C51, the Timer 1 overflow rate determines the baud rates in Modes 1 and 3.

**NUMBER OF INTERRUPTS IN 89C51:**

There are basically five interrupts available to the user. Reset is also considered as an interrupt. There are two interrupts for timer, two interrupts for external hardware interrupt and one interrupt for serial communication.

| Memory location | Interrupt name |
|---|---|
| 0000H | Reset |
| 0003H | External interrupt 0 |
| 000BH | Timer interrupt 0 |
| 0013H | External interrupt 1 |
| 001BH | Timer interrupt 1 |
| 0023H | Serial COM interrupt |

Lower the vector, higher the priority. The External Interrupts INT0 and INT1 can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are the IE0 and IE1 bits in TCON. When the service routine is vectored, hardware clears the flag that generated an external interrupt only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source (rather than the on-chip hardware) controls the request flag.

The Timer 0 and Timer 1 Interrupts are generated by TF0and TF1, which are set by a rollover in their respective Timer/Counter registers (except for Timer 0 in Mode 3).When a timer interrupt is generated, the on-chip hardware clears the flag that is generated.

The Serial Port Interrupt is generated by the logical OR of RI and TI. The service routine normally must determine whether RI or TI generated the interrupt, and the bit must be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though they had been set or cleared by hardware. That is, interrupts can be generated and pending interrupts can be canceled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (interrupt enable) at address 0A8H. There is a global enable/disable bit that is cleared to disable all interrupts or to set the interrupts.

**IE (Interrupt enable register):**

**Steps in enabling an interrupt:**

Bit D7 of the IE register must be set to high to allow the rest of register to take effect. If EA=1, interrupts are enabled and will be responded to if their corresponding bits in IE are high. If EA=0, no interrupt will be responded to even if the associated bit in the IE register is high.

**Description of each bit in IE register:**

D7 bit: Disables all interrupts. If EA =0, no interrupt is acknowledged, if EA=1 each interrupt source is individually enabled or disabled by setting or clearing its enable bit.

D6 bit: Reserved.

D5 bit: Enables or disables timer 2 over flow interrupt (in 8052).

D4 bit: Enables or disables serial port interrupt.

D3 bit: Enables or disables timer 1 over flow interrupt.

D2 bit: Enables or disables external interrupt 1.

D1 bit: Enables or disables timer 0 over flow interrupt.

D0 bit: Enables or disables external interrupt 0.

### Interrupt priority in 89C51:

There is one more SRF to assign priority to the interrupts which is named as interrupt priority (IP). User has given the provision to assign priority to one interrupt. Writing one to that particular bit in the IP register fulfils the task of assigning the priority.

### Description of each bit in IP register:

D7 bit: Reserved.

D6 bit: Reserved.

D5 bit: Timer 2 interrupt priority bit (in 8052).

D4 bit: Serial port interrupt priority bit.

D3 bit: Timer 1 interrupt priority bit.

D2 bit: External interrupt 1 priority bit.

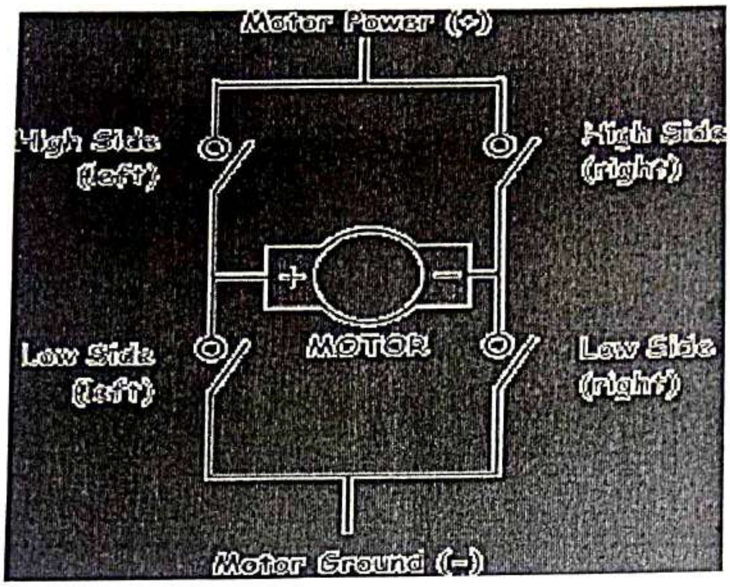
D1 bit: Timer 0 interrupt priority bit.

D0 bit: External interrupt 0 priority bit.

### H-Bridge Theory:

An H-Bridge is an electronic circuit which enables a voltage to be applied across a load in either direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards and backwards. H-bridges are available as integrated circuits, or can be built from discrete components

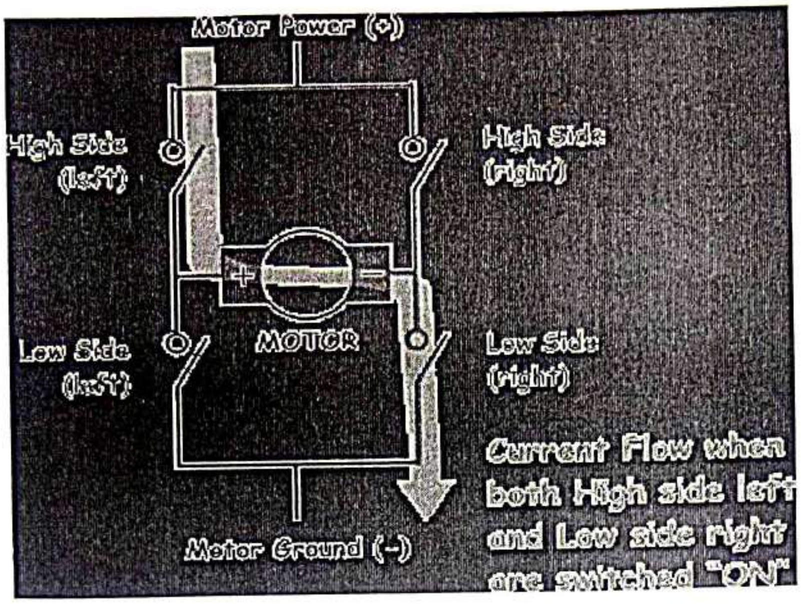
Let's start with the name, H-bridge. Sometimes called a "full bridge" the H-bridge is so named because it has four switching elements at the "corners" of the H and the motor forms the cross bar. The basic bridge is shown in the figure to the right.



The key fact to note is that there are, in theory, four switching elements within the bridge. These four elements are often called, high side left, high side right, low side right, and low side left (when traversing in clockwise order).

The switches are turned on in pairs, either high left and lower right, or lower left and high right, but never both switches on the same "side" of the bridge. If both switches on one side of a bridge are turned on it creates a short circuit between the battery plus and battery minus terminals. This phenomena is called shoot through in the Switch-Mode Power Supply (SMPS) literature. If the bridge is sufficiently powerful it will absorb that load and your batteries will simply drain quickly. Usually however the switches in question melt.

To power the motor, you turn on two switches that are diagonally opposed. In the picture to the right, imagine that the high side left and low side right switches are turned on. The current flow is shown in green.

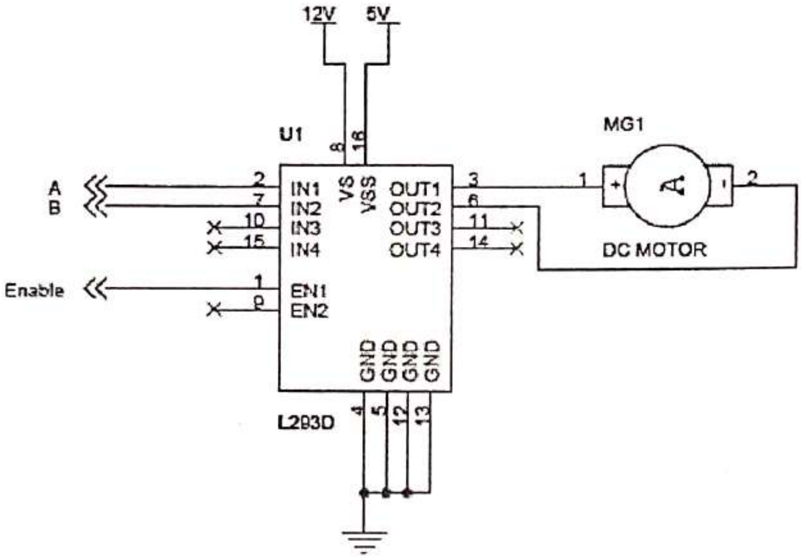


The current flows and the motor begins to turn in a "positive" direction. What happens if you turn on the high side right and low side left switches? You guessed it, current flows the other direction through the motor and the motor turns in the opposite direction.

**L293D Dual H-Bridge Motor Driver**

L293D is a dual H-Bridge motor driver, So with one IC we can interface two DC motors which can be controlled in both clockwise and counter clockwise direction and if you have motor with fix direction of motion the you can make use of all the four I/Os to connect up to four DC motors. L293D has output current of 600mA and peak output current of 1.2A per channel. Moreover for protection of circuit from back EMF ouput diodes are included within the IC. The output supply (VCC2) has a wide range from 4.5V to 36V, which has made L293D a best choice for DC motordriver.

**A simple schematic for interfacing a DC motor using L293D is shown below.**



**Truth Table**

| A | B | Description |
|---|---|---|
| 0 | 0 | Motor stops or Breaks |
| 0 | 1 | Motor Runs Anti-Cloclwise |
| 1 | 0 | Motor Runs Clockwise |
| 1 | 1 | Motor Stops or Breaks |

For above truth table, the Enable has to be Set (1). Motor Power is mentioned 12V, but you can connect power according to your motors.

As you can see in the circuit, three pins are needed for interfacing a DC motor (A, B, Enable). If you want the o/p to be enabled completely then you can connect Enable to VCC and only 2 pins needed from controller to make the motor work.

As per the truth mentioned in the image above its fairly simple to program the microcontroller.

## MOTORS

### DEFINITION

Motor is a device that creates motion, not an engine; it usually refers to either an electrical motor or an internal combustion engine.

It may also refer to:

- Electric motor, a machine that converts electricity into a mechanical motion
    - AC motor, an electric motor that is driven by alternating current
        - Synchronous motor, an alternating current motor distinguished by a rotor spinning with coils passing magnets at the same rate as the alternating current and resulting magnetic field which drives it
        - Induction motor, also called a squirrel-cage motor, a type of asynchronous alternating current motor where power is supplied to the rotating device by means of electromagnetic induction
    - DC motor, an electric motor that runs on direct current electricity
        - Brushed DC electric motor, an internally commutated electric motor designed to be run from a direct current power source
        - Brushless DC motor, a synchronous electric motor which is powered by direct current electricity and has an electronically controlled commutation system, instead of a mechanical commutation system based on brushes
    - Electrostatic motor, a type of electric motor based on the attraction and repulsion of electric charge
    - Servo motor, an electric motor that operates a servo, commonly used in robotics

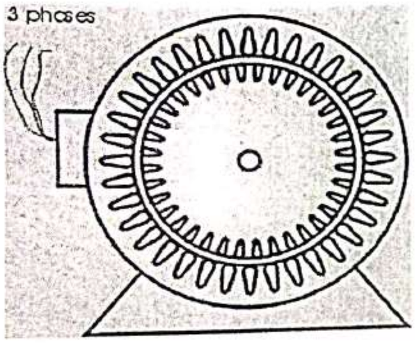
- Internal fan-cooled electric motor, an electric motor that is self-cooled by a fan, typically used for motors with a high energy density

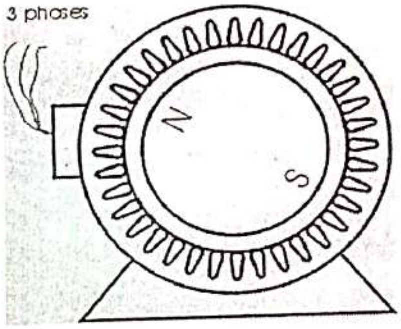
## TYPES OF MOTORS

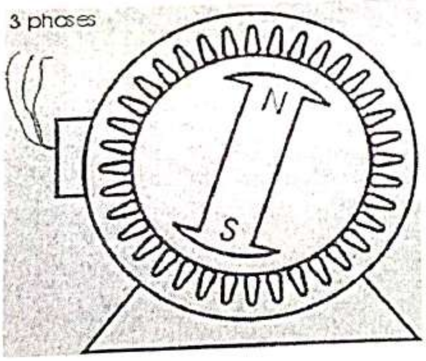
Industrial motors come in a variety of basic types. These variations are suitable for many different applications. Naturally, some types of motors are more suited for certain applications than other motor types are. This document will hopefully give some guidance in selecting these motors.

### AC Motors

The most common and simple industrial motor is the three phase AC induction motor, sometimes known as the "squirrel cage" motor. Substantial information can be found about any motor by checking its (nameplate).







## Advantages

- Simple Design
- Low Cost
- Reliable Operation
- Easily Found Replacements
- Variety of Mounting Styles
- Many Different Environmental Enclosures

## Simple Design

The simple design of the AC motor -- simply a series of three windings in the exterior (stator) section with a simple rotating section (rotor). The changing field caused by the 50 or 60 Hertz AC line voltage causes the rotor to rotate around the axis of the motor.

The speed of the AC motor depends only on three variables:

1. The fixed number of winding sets (known as poles) built into the motor, which determines the motor's base speed.
2. The frequency of the AC line voltage. Variable speed drives change this frequency to change the speed of the motor.
3. The amount of torque loading on the motor, which causes slip.

## Low Cost

The AC motor has the advantage of being the lowest cost motor for applications requiring more than about 1/2 hp (325 watts) of power. This is due to the simple design of the motor. For this reason, AC motors are overwhelmingly preferred for fixed speed applications in industrial applications and for commercial and domestic applications where AC line power can be easily attached. Over 90% of all motors are AC induction motors. They are found in air conditioners, washers, dryers, industrial machinery, fans, blowers, vacuum cleaners, and many, many other applications.

## Reliable Operation

The simple design of the AC motor results in extremely reliable, low maintenance operation. Unlike the DC motor, there are no brushes to replace. If run in the appropriate environment for its enclosure, the AC motor can expect to need new bearings after several years of operation. If the application is well designed, an AC motor may not need new bearings for more than a decade.

## Easily Found Replacements

The wide use of the AC motor has resulted in easily found replacements. Many manufacturers adhere to either European (metric) or American (NEMA) standards. (For Replacement Motors)

## Variety of Mounting Styles

AC Motors are available in many different mounting styles such as:

- □ Foot Mount
- □ C-Face
- □ Large Flange
- □ Vertical
- □ Specialty

## DC Motors

The brushed DC motor is one of the earliest motor designs. Today, it is the motor of choice in the majority of variable speed and torque control applications.

## Advantages

- Easy to understand design
- Easy to control speed
- Easy to control torque
- Simple, cheap drive design

**Easy to understand design**

The design of the brushed DC motor is quite simple. A permanent magnetic field is created in the stator by either of two means:

- Permanent magnets
- Electro-magnetic windings

If the field is created by permanent magnets, the motor is said to be a "permanent magnet DC motor" (PMDC). If created by electromagnetic windings, the motor is often said to be a "shunt wound DC motor" (SWDC). Today, because of cost-effectiveness and reliability, the PMDC motor is the motor of choice for applications involving fractional horsepower DC motors, as well as most applications up to about three horsepower.

At five horsepower and greater, various forms of the shunt wound DC motor are most commonly used. This is because the electromagnetic windings are more cost effective than permanent magnets in this power range.

Caution: *If a DC motor suffers a loss of field (if for example, the field power connections are broken), the DC motor will immediately begin to accelerate to the top speed which the loading will allow. This can result in the motor flying apart if the motor is lightly loaded. The possible loss of field must be accounted for, particularly with shunt wound DC motors.*

Opposing the stator field is the armature field, which is generated by a changing electromagnetic flux coming from windings located on the rotor. The magnetic poles of the armature field will attempt to line up with the opposite magnetic poles generated by the stator field. If we stopped the design at this point, the motor would spin until the poles were opposite one another, settle into place, and then stop -- which would make a pretty useless motor!

However, we are smarter than that. The section of the rotor where the electricity enters the rotor windings is called the commutator. The electricity is carried between the rotor and the stator by conductive graphite-copper brushes (mounted on the rotor) which contact rings on stator. Imagine power is supplied:

The motor rotates toward the pole alignment point. Just as the motor would get to this point, the brushes jump across a gap in the stator rings. Momentum carries the motor forward over this gap. When the brushes get to the other side of the gap, they contact the stator rings again and -- the polarity of the voltage is reversed in this set of rings! The motor begins accelerating again, this time trying to get to the opposite set of poles. (The momentum has carried the motor past the original pole alignment point.) This continues as the motor rotates.

In most DC motors, several sets of windings or permanent magnets are present to smooth out the motion.

**Easy to control speed**

Controlling the speed of a brushed DC motor is simple. The higher the armature voltage, the faster the rotation. This relationship is linear to the motor's maximum speed.

The maximum armature voltage which corresponds to a motor's rated speed (these motors are usually given a rated speed and a maximum speed, such as 1750/2000 rpm) are available in certain standard voltages, which roughly increase in conjuntion with horsepower. Thus, the smallest industrial motors are rated 90 VDC and 180 VDC. Larger units are rated at 250 VDC and sometimes higher.

Specialty motors for use in mobile applications are rated 12, 24, or 48 VDC. Other tiny motors may be rated 5 VDC.

Most industrial DC motors will operate reliably over a speed range of about 20:1 -- down to about 5-7% of base speed. This is much better performance than the comparible AC motor. This is partly due to the simplicity of control, but is also partly due to the fact that most industrial DC motors are designed with variable speed operation in mind, and have added heat dissipation features which allow lower operating speeds.

### Easy to control torque

In a brushed DC motor, torque control is also simple, since output torque is proportional to current. If you limit the current, you have just limited the torque which the motor can achieve. This makes this motor ideal for delicate applications such as textile manufacturing.

### Simple, cheap drive design

The result of this design is that variable speed or variable torque electronics are easy to design and manufacture. Varying the speed of a brushed DC motor requires little more than a large enough potentiometer. In practice, these have been replaced for all but sub-fractional horsepower applications by the SCR and PWM drives, which offer relatively precisely control voltage and current. Common DC drives are available at the low end (up to 2 horsepower) for under US$100 -- and sometimes under US$50 if precision is not important.

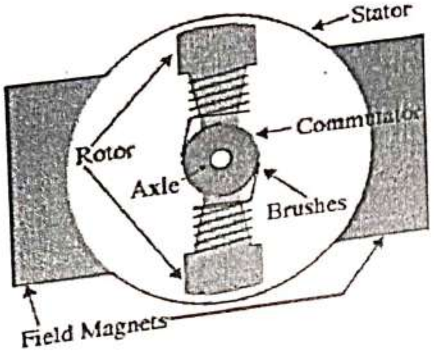
Large DC drives are available up to hundreds of horsepower. However, over about 10 horsepower careful consideration should be given to the price/performance tradeoffs with AC inverter systems, since the AC systems show a price advantage in the larger systems. (But they may not be capable of the application's performance requirments).

### Disadvantages

- Expensive to produce
- Can't reliably control at lowest speeds
- Physically larger
- High maintenance
- Dust

## WORKING OF DC MOTOR

In any electric motor, operation is based on simple electromagnetism. A current-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. As you are well aware of from playing with magnets as a kid, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion.

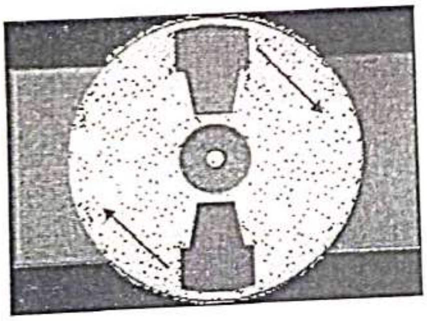


### Principle

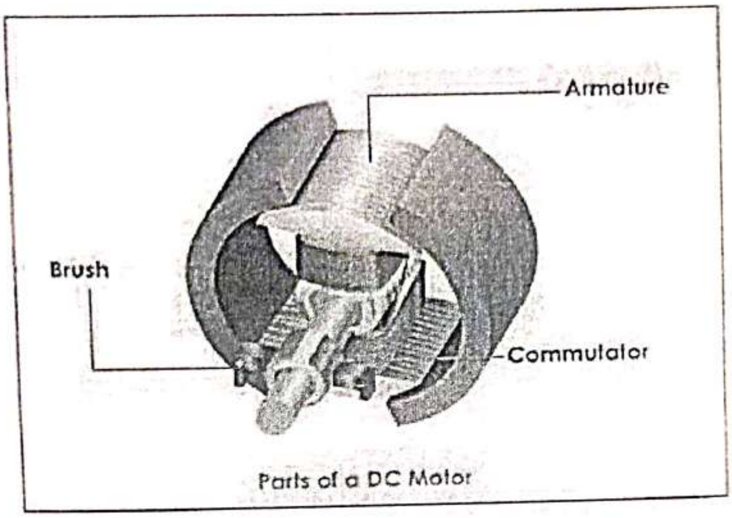
When a rectangular coil carrying current is placed in a magnetic field, a torque acts on the coil which rotates it continuously. When the coil rotates, the shaft attached to it also rotates and thus it is able to do mechanical work.Every DC motor has six basic parts -- axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors (and all that Beamers will see), the external magnetic field is produced by high-strength permanent magnets[1]. The stator is the stationary part of the motor -- this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor (together with the axle and attached commutator) rotate with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator. The above diagram shows a common motor layout -- with the rotor inside the stator (field) magnets.

The geometry of the brushes, commentator contacts, and rotor windings are such that when power is applied, the polarities of the energized winding and the stator magnet(s) are misaligned, and the rotor will rotate until it is almost aligned with the stator's field magnets. As the rotor reaches alignment, the brushes move to the next commentator contacts, and energize the next winding. Given our example two-pole motor, the rotation reverses the direction of current through the rotor winding, leading to a "flip" of the rotor's magnetic field, driving it to continue rotating.

In real life, though, DC motors will always have more than two poles (three is a very common number). In particular, this avoids "dead spots" in the commutator. You can imagine how with our example two-pole motor, if the rotor is exactly at the middle of its rotation (perfectly aligned with the field magnets), it will get "stuck" there. Meanwhile, with a two-pole motor, there is a moment where the commutator shorts out the power supply (i.e., both brushes touch both commutator contacts simultaneously). This would be bad for the power supply, waste energy, and damage motor components as well. Yet another disadvantage of such a simple motor is that it would exhibit a high amount of torque "ripple" (the amount of torque it could produce is cyclic with the position of the rotor).



## Construction and Working



Parts of a DC Motor

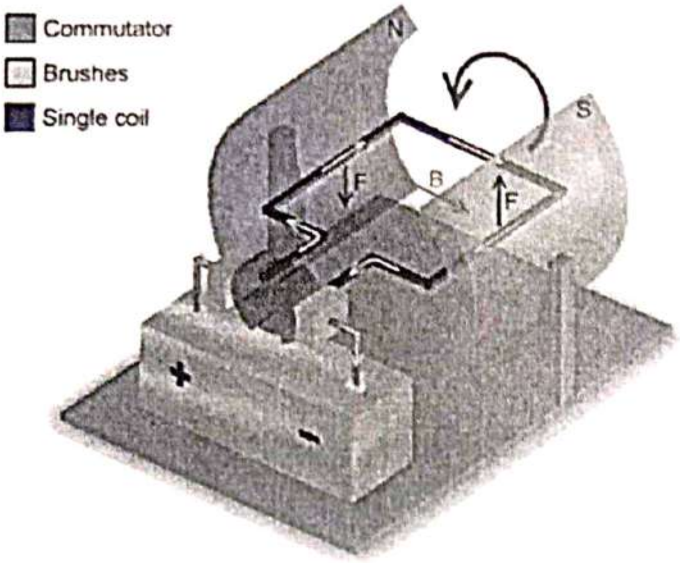
## Parts of a DC Motor

### Armature

A D.C. motor consists of a rectangular coil made of insulated copper wire wound on a soft iron core. This coil wound on the soft iron core forms the armature. The coil is mounted on an axle and is placed between the cylindrical concave poles of a magnet.

### Commutator

A commutator is used to reverse the direction of flow of current. Commutator is a copper ring split into two parts $C_1$ and $C_2$. The split rings are insulated form each other and mounted on the axle of the motor. The two ends of the coil are soldered to these rings. They rotate along with the coil. Commutator rings are connected to a battery. The wires from the battery are not connected to the rings but to the brushes which are in contact with the rings.

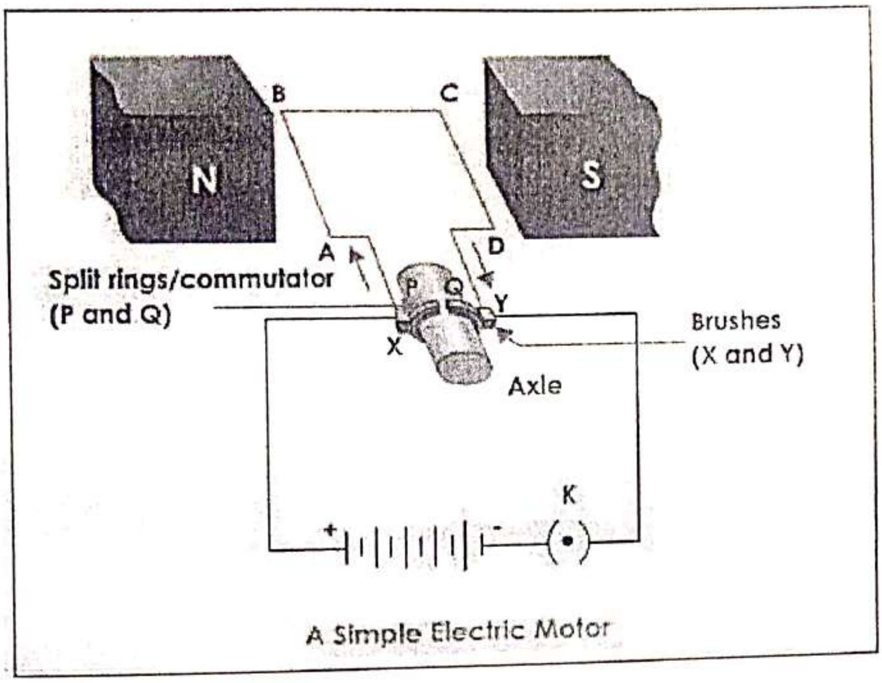


## Brushes

Two small strips of carbon, known as brushes press slightly against the two split rings, and the split rings rotate between the brushes.The carbon brushes are connected to a D.C. source.

## Working of a DC Motor

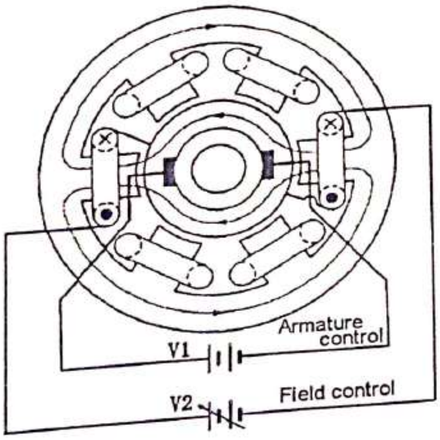
When the coil is powered, a magnetic field is generated around the armature. The left side of the armature is pushed away from the left magnet and drawn towards the right, causing rotation.

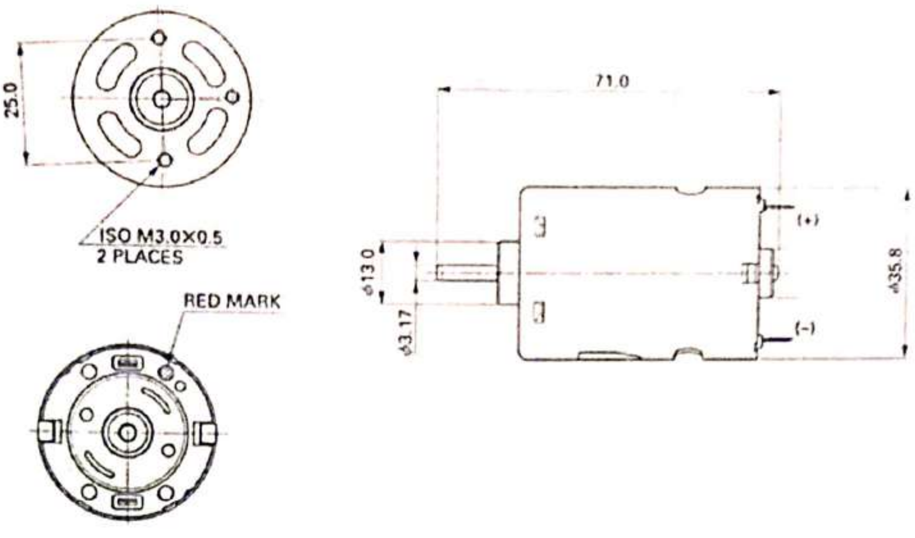


A Simple Electric Motor

When the coil turns through $90^0$, the brushes lose contact with the commutator and the current stops flowing through the coil.However the coil keeps turning because of its own momentum.

Now when the coil turns through $180^0$, the sides get interchanged. As a result the commutator ring $C_1$ is now in contact with brush $B_2$ and commutator ring $C_2$ is in contact with brush $B_1$. Therefore, the current continues to flow in the same direction.



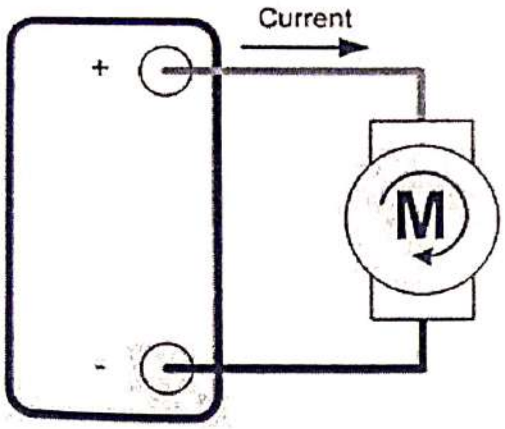


## PARAMETRS OF THE DC MOTRS

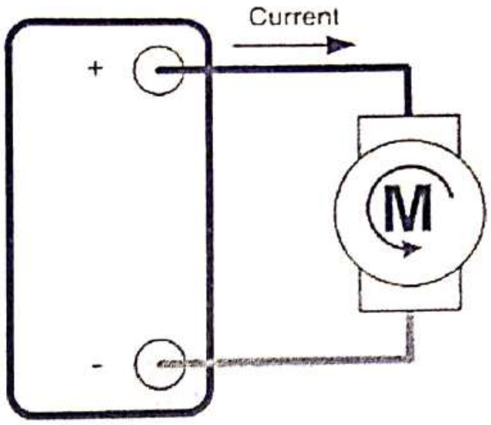
1. Direction of rotation
2. Motor Speed
3. Motor Torque
4. Motor Start and Stop

### Direction of Rotation

A DC Motor has two wires. We can call them the positive terminal and the negative terminal, although these are pretty much arbitrary names (unlike a battery where these polarities are vital and not to be mixed!). On a motor, we say that when the + wire is connected to + terminal on a power source, and the - wire is connected to the - terminal source on the same power source, the motor rotates clockwise (if you are looking towards the motor shaft). If you reverse the wire polarities so that each wire is connected to the opposing power supply terminal, then the motor rotates counter clockwise. Notice this is just an arbitrary selection and that some motor manufacturers could easily choose the opposing convention. As long as you know what rotation you get with one polarity, you can always connect in such a fashion that you get the direction that you want on a per polarity basis.



Positive Polarity
Clockwise rotation



Negative Polarity
Counter - Clockwise rotation

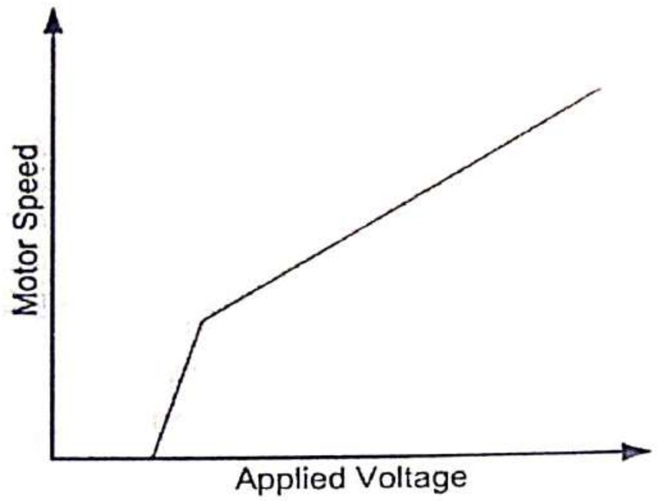
### DC Motor Rotation vs Polarity

**Facts:**

- DC Motor rotation has nothing to do with the voltage magnitude or the current magnitude flowing through the motor.
- DC Motor rotation does have to do with the voltage polarity and the direction of the current flow.

### DC Motor Speed

Whereas the voltage polarity controls DC motor rotation, voltage magnitude controls motor speed. Think of the voltage applied as a facilitator for the strengthening of the magnetic field. In other words, the higher the voltage, the quicker will the magnetic field become strong. Remember that a DC motor has an electromagnet and a series of permanent magnets. The applied voltage generates a magnetic field on the electromagnet portion. This electromagnet field is made to oppose the permanent magnet field. If the electromagnet field is very strong, then both magnetic entities will try to repel each other from one side, as well as atract each other from the other side. The stronger the induced magnetic field, the quicker will this separation/attaction will try to take place. As a result, motor speed is directly proportional to applied voltage.



## Motor Speed Curve

One aspect to have in mind is that the motor speed is not entirely lineal. Each motor will have their own voltage/speed curve. One thing I can guarantee from each motor is that at very low voltages, the motor will simply not move. This is because the magnetic field strength is not enough to overcome friction. Once friction is overcome, motor speed will start to increase as voltage increase.The following video shows the concept of speed control and offers some ideas on how this can be achieved.

### Motor Torque

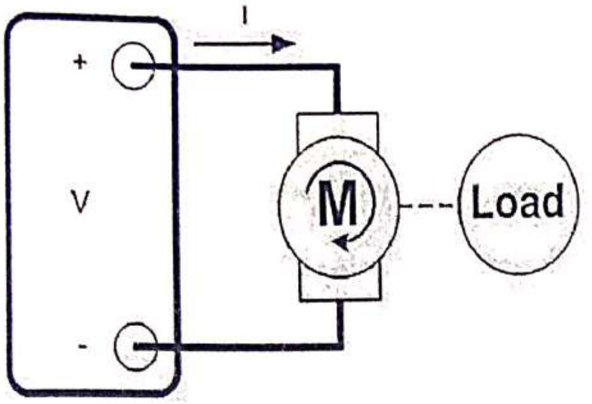
In the previous segment I kind of described speed as having to do with the strength of the magnetic field, but this is in reality misleading. Speed has to do with how fast the magnetic field is built and the attraction/repel forces are installed into the two magnetic structures. Motor strength, on the other hand, has to do with magnetic field strength. The stronger the electromagnet attracts the permanent magnet, the more force is exerted on the motor load.

Per example, imagine a motor trying to lift 10 pounds of weight. This is a force that when multiplied by a distance (how much from the ground we are lifting the load) results in WORK. This WORK when exerted through a predetermined amount of time (for how long we are lifting the weight) gives us power. But whatever power came in, must come out as energy can not be created or destroyed. So that you know, the power that we are supplying to the motor is computed by

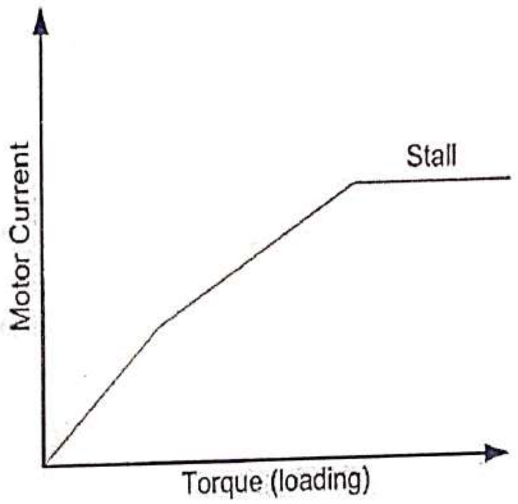
$$P = IV$$

Where P is power, I is motor current and V is motor voltage

Hence, if the voltage (motor speed) is maintained constant, how much load we are moving must come from the current. As you increase load (or torque requirements) current must also increase.



P = IV



## Motor Loading

One aspect about DC motors which we must not forget is that loading or increase of torque can not be infinite as there is a point in which the motor simply can not move. When this happens, we call this loading "Stalling Torque". At the same time this is the maximum amount of current the motor will see, and it is refer to Stalling Current. Stalling deserves a full chapter as this is a very important scenario that will define a great deal of the controller to be used. I promise I will later write a post on stalling and its intricacies.

## Motor Start and Stop

You are already well versed on how to control the motor speed, the motor torque and the motor direction of rotation. But this is all fine and dandy as long as the motor is actually moving. How about starting it and stopping it? Are these trivial matters? Can we just ignore them or should we be careful about these aspects as well? You bet we should!

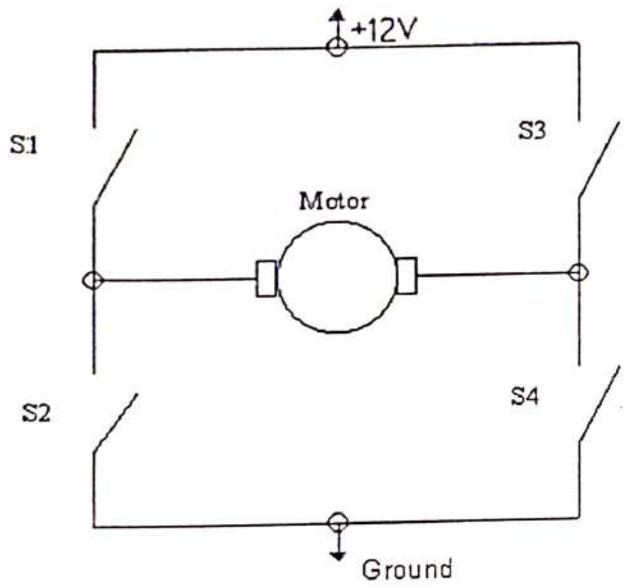
Starting a motor is a very hazardous moment for the system. Since you have an inductance whose energy storage capacity is basically empty, the motor will first act as an inductor. In a sense, it should not worry us too much because current can not change abruptly in an inductor, but the truth of the matter is that this is one of the instances in which you will see the highest currents flowing into the motor. The start is not necessarily bad for the motor itself as in fact the motor can easily take this Inrush Current. The power stage, on the other hand and if not properly designed for, may take a beating.

Once the motor has started, the motor current will go down from inrush levels to whatever load the motor is at. Per example, if the motor is moving a few gears, current will be proportional to that load and according to torque/current curves.

Stopping the motor is not as harsh as starting. In fact, stopping is pretty much a breeze. What we do need to concern ourselves is with how we want the motor to stop. Do we want it to coast down as energy is spent in the loop, or do we want the rotor to stop as fast as possible? If the later is the option, then we need braking. Braking is easily accomplished by shorting the motor outputs. The reason why the motor stops so fast is because as a short is applied to the motor terminals, the Back EMF is shorted. Because Back EMF is directly proportional to speed, making Back EMF = 0, also means making speed = 0.

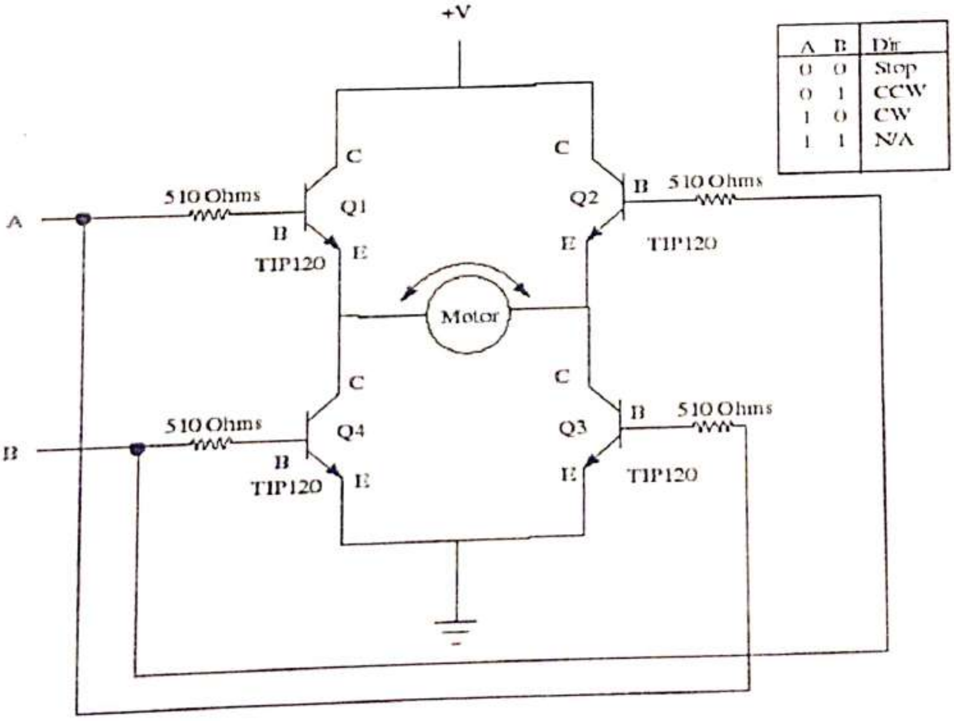
## MOTORDRIVER CIRCUIT

The name "H-Bridge" is derived from the actual shape of the switching circuit which control the motoion of the motor. It is also known as "Full Bridge". Basically there are four switching elements in the H-Bridge as shown in the figure below.



**H-Bridge Switch**

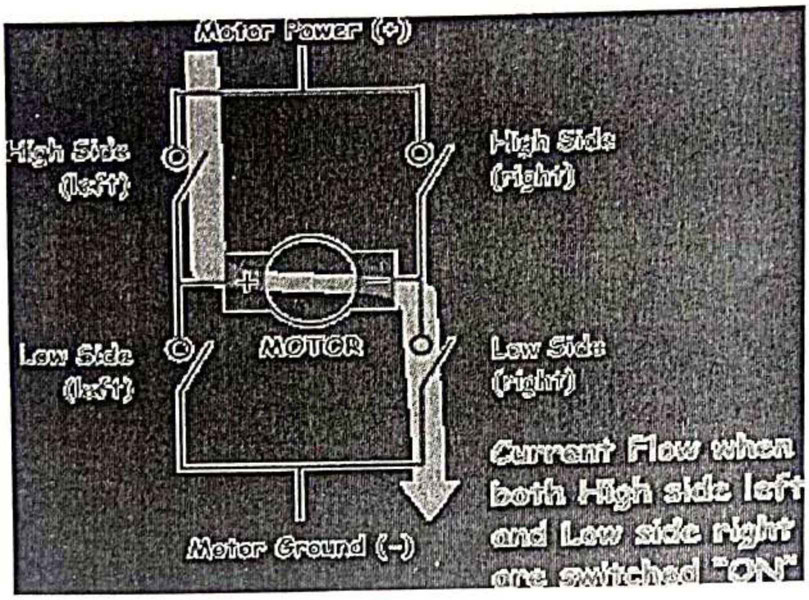
| Switches Closed | Motor |
|---|---|
| S1 S3 | Off |
| S2 S4 | Off |
| S1 S4 | Forward |
| S3 S2 | Reverse |



| A | B | Dir |
|---|---|---|
| 0 | 0 | Stop |
| 0 | 1 | CCW |
| 1 | 0 | CW |
| 1 | 1 | N/A |

As you can see in the figure above there are four switching elements named as "High side left", "High side right", "Low side right", "Low side left".

When these switches are turned on in pairs motor changes its direction accordingly. Like, if we switch on High side left and Low side right then motor rotate in forward direction, as current flows from Power supply through the motor coil goes to ground via switch low side right. This is shown in the figure below.



Similarly, when you switch on low side left and high side right, the current flows in opposite direction and motor rotates in backward direction. This is the basic working of H-Bridge. We can also make a small truth table according to the switching of H-Bridge explained above.

Truth Table

| High Left | High Right | Low Left | Low Right | Description |
|---|---|---|---|---|
| On | Off | Off | On | Motor runs clockwise |
| Off | On | On | Off | Motor runs anti-clockwise |
| On | On | Off | Off | Motor stops or decelerates |
| Off | Off | On | On | Motor stops or decelerates |

As already said, H-bridge can be made with the help of trasistors as well as MOSFETs, the only thing is the power handling capacity of the circuit. If motors are needed to run with high current then lot of dissipation is there. So head sinks are needed to cool the circuit.

Now you might be thinkin why i did not discuss the cases like High side left on and Low side left on or high side right on and low side right on. Clearly seen in the diagra, you don't want to burn your power supply by shorting them. So that is why those combinations are not discussed in the truth table.

## LCD (Liquid Cristal Display):

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

A program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an controller is an LCD display. Some of the most common LCDs connected to the contollers are 16X1, 16x2 and 20x2 displays. This means 16 characters per line by 1 line 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Many microcontroller devices use 'smart LCD' displays to output visual information. LCD displays designed around LCD NT-C1611 module, are inexpensive, easy to use, and it`is even possible to produce a readout using the 5X7 dots plus cursor of the display. They have a standard ASCII set of characters and mathematical symbols. For an 8-bit data bus, the display requires a +5V supply plus 10 I/O lines (RS RW D7 D6 D5 D4 D3 D2 D1 D0). For a 4-bit data bus it only requires the supply lines plus 6 extra lines(RS RW D7 D6 D5 D4). When the LCD display is not enabled, data lines are tri-state and they do not interfere with the operation of the microcontroller.

**Features:**

(1) Interface with either 4-bit or 8-bit microprocessor.

(2)□ Display data RAM

(3) 8□80x bits (80 characters).

(4)□ Character generator ROM

(5). 160 different 7□□5 dot-matrix character patterns.

□(6). Character generator RAM

(7) 8□ different user programmed 7□□5 dot-matrix patterns.

(8).Display data RAM and character generator RAM may be

Accessed by the microprocessor.

(9)□ Numerous instructions

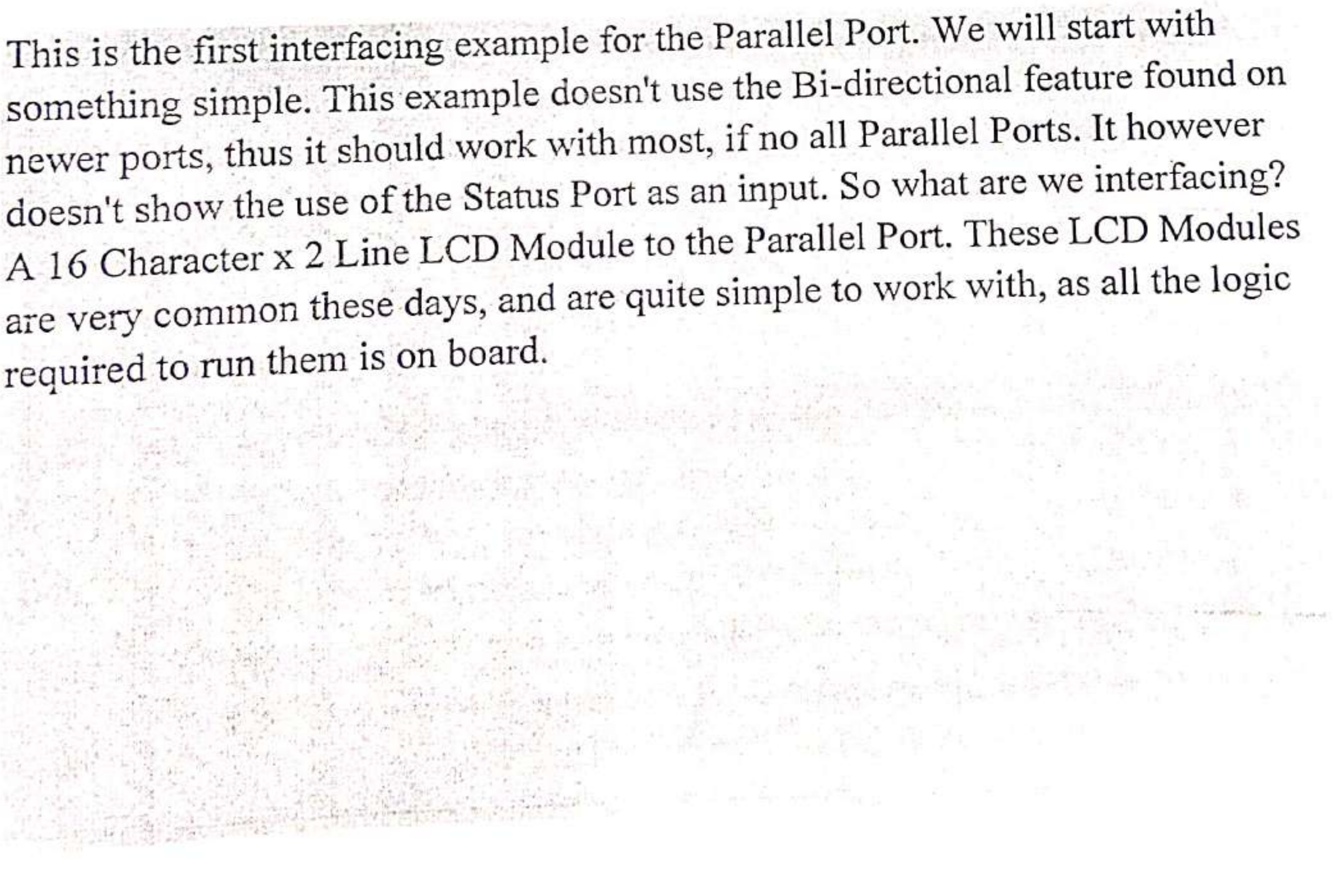
(10)□ .Clear Display, Cursor Home, Display ON/OFF, Cursor ON/OFF,

Blink Character, Cursor Shift, Display Shift.

(11).□ Built-in reset circuit is triggered at power ON.

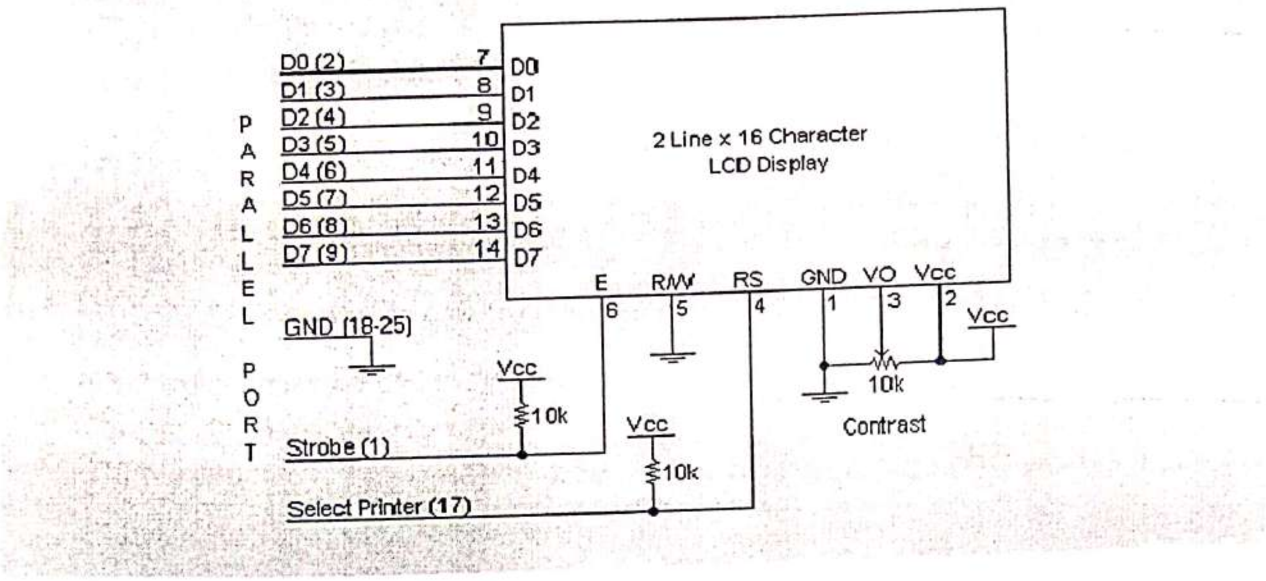
(12).□ Built-in oscillator.

**Description Of 16x2:**



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

**Schematic Diagram:**



- Above is the quite simple schematic. The LCD panel's *Enable* and *Register Select* is connected to the Control Port. The Control Port is an open collector / open drain output. While most Parallel Ports have internal pull-up resistors, there are a few which don't. Therefore by incorporating the two 10K external pull up resistors, the circuit is more portable for a wider range of computers, some of which may have no internal pull up resistors.
- We make no effort to place the Data bus into reverse direction. Therefore we hard wire the *R/W* line of the LCD panel, into write mode. This will cause no bus conflicts on the data lines. As a result we cannot read back the LCD's internal Busy Flag which tells us if the LCD has accepted and finished processing the last instruction. This problem is overcome by inserting known delays into our program.
- The 10k Potentiometer controls the contrast of the LCD panel. Nothing fancy here. As with all the examples, I've left the power supply out. You can use a bench power supply set to 5v or use a onboard +5 regulator. Remember a few de-coupling capacitors, especially if you have trouble with the circuit working properly.
- The 2 line x 16 character LCD modules are available from a wide range of manufacturers and should all be compatible with the HD44780. The one I used to test this circuit was a Power tip PC-

1602F and an old Philips LTN211F-10 which was extracted from a Poker Machine! The diagram to the right, shows the pin numbers for these devices. When viewed from the front, the left pin is pin 14 and the right pin is pin 1

## 16 x 2 Alphanumeric LCD Module Features:

- Intelligent, with built-in Hitachi HD44780 compatible LCD controller and RAM providing simple interfacing
- 61 x 15.8 mm viewing area
- 5 x 7 dot matrix format for 2.96 x 5.56 mm characters, plus cursor line
- Can display 224 different symbols
- Low power consumption (1 mA typical)
- Powerful command set and user-produced characters
- TTL and CMOS compatible
- Connector for standard 0.1-pitch pin headers

## 16 x 2 Alphanumeric LCD Module Specifications:

| Pin | Symbol | Level | Function |
|---|---|---|---|
| 1 | $V_{SS}$ | - | Power, GND |
| 2 | $V_{DD}$ | - | Power, 5V |
| 3 | Vo | - | Power, for LCD Drive |
| 4 | RS | H/L | Register Select Signal<br>H: Data Input<br>L: Instruction Input |

|  |  |  |  |
|---|---|---|---|
| 5 | R/W | H/L | H: Data Read (LCD->MPU)<br>L: Data Write (MPU->LCD) |
| 6 | E | H,H->L | Enable |
| 7-14 | DB0-DB7 | H/L | Data Bus; Software selectable 4- or 8-bit mode |
| **15** | NC | - | **NOT CONNECTED** |
| 16 | NC | - | **NOT CONNECTED** |

**FEATURES:**

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

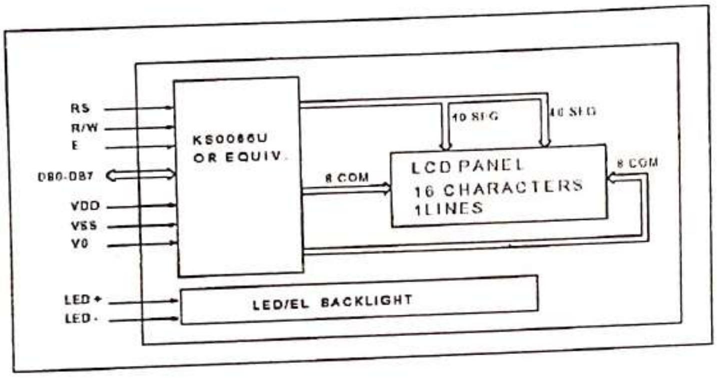
**Data can be placed at any location on the LCD. For 16×1 LCD, the address locations are:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POSITION |  | 1 | 2. | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| ADDRESS | LINE1 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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

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

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



**Electrical Block Diagram**

## Power supply for LCD driving:

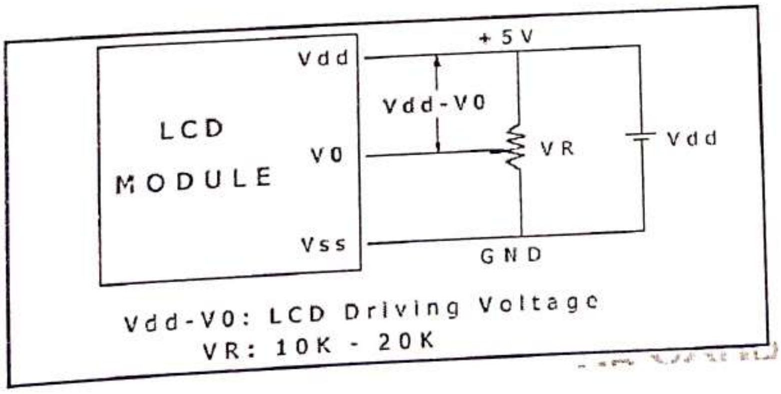


**Fig: 18:power supply for LCD**

## PIN DESCRIPTION:

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

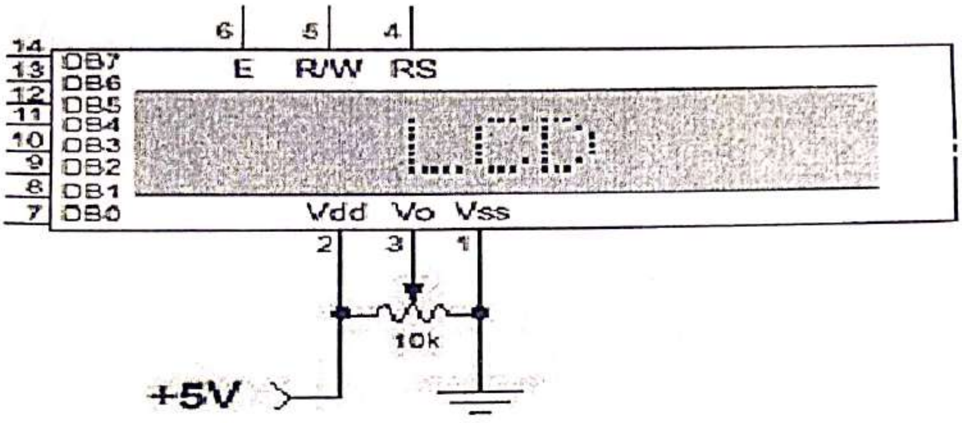


Fig 19: pin diagram of 1x16 lines LCD

| PIN | SYMBOL | FUNCTION |
|---|---|---|
| 1 | Vss | Power Supply(GND) |
| 2 | Vdd | Power Supply(+5V) |
| 3 | Vo | Contrast Adjust |
| 4 | RS | Instruction/Data Register Select |
| 5 | R/W | Data Bus Line |
| 6 | E | Enable Signal |
| 7-14 | DB0-DB7 | Data Bus Line |
| 15 | A | Power Supply for LED B/L(+) |
| 16 | K | Power Supply for LED B/L(-) |

Fig 17: Pin specifications

**EN:**

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

**RS:**

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

**RW:**

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

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

**Logic status on control lines:**

- E - 0 Access to LCD disabled

- 1 Access to LCD enabled

- R/W - 0 Writing data to LCD

- 1 Reading data from LCD

- RS - 0 Instructions

  - 1 Character

**Writing data to the LCD:**

1) Set R/W bit to low

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

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

4) Set E line to high

5) Set E line to low

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

1) Set R/W bit to high

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

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

4) Set E line to high

5) Set E line to low

**Entering Text:**

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

## SWITCHES:

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

All the available characters that are built into the module are shown in Table 3. Studying the table, you will see that codes associated with the characters are quoted in binary and hexadecimal, most significant bits ("left-hand" four bits) across the top, and least significant bits ("right-hand" four bits) down the left.

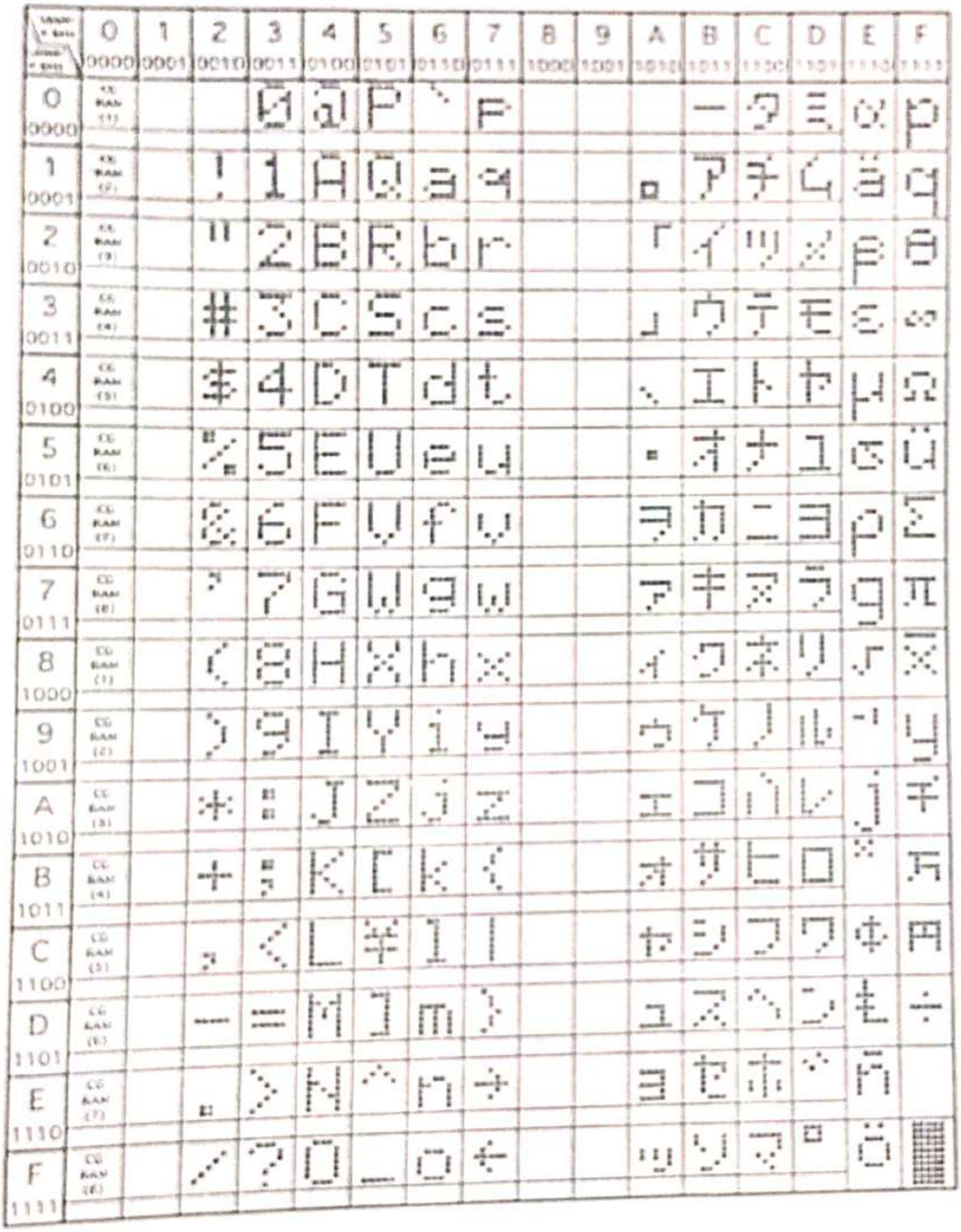
Most of the characters conform to the ASCII standard, although the Japanese and Greek characters (and a few other things) are obvious exceptions. Since these intelligent modules were designed in the "Land of the Rising Sun," it seems only fair that their Katakana phonetic symbols should also be incorporated. The more extensive Kanji character set, which the Japanese share with the Chinese, consisting of several thousand different characters, is not included!

Using the switches, of whatever type, and referring to Table 3, enter a few characters onto the display, both letters and numbers. The RS switch (S10) must be "up" (logic 1) when sending the characters, and switch E (S9) must be pressed for each of them. Thus the operational order is: set RS high, enter character, trigger E, leave RS high, enter another character, trigger E, and so on.

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

offering such capabilities as bar graphs, flashing symbols, even animated characters. Before the user-defined characters are set up, these codes will just bring up strange looking symbols.

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



| Upper 4 bits / Lower 4 bits | 0<br>0000 | 1<br>0001 | 2<br>0010 | 3<br>0011 | 4<br>0100 | 5<br>0101 | 6<br>0110 | 7<br>0111 | 8<br>1000 | 9<br>1001 | A<br>1010 | B<br>1011 | C<br>1100 | D<br>1101 | E<br>1110 | F<br>1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0<br>0000 | CG RAM (1) | | | 0 | @ | P | ` | p | | | | ー | タ | ミ | α | p |
| 1<br>0001 | CG RAM (2) | | ! | 1 | A | Q | a | q | | | 。 | ア | チ | ム | ä | q |
| 2<br>0010 | CG RAM (3) | | " | 2 | B | R | b | r | | | 「 | イ | ツ | メ | β | θ |
| 3<br>0011 | CG RAM (4) | | # | 3 | C | S | c | s | | | 」 | ウ | テ | モ | ε | ∞ |
| 4<br>0100 | CG RAM (5) | | $ | 4 | D | T | d | t | | | 、 | エ | ト | ヤ | μ | Ω |
| 5<br>0101 | CG RAM (6) | | % | 5 | E | U | e | u | | | ・ | オ | ナ | ユ | σ | ü |
| 6<br>0110 | CG RAM (7) | | & | 6 | F | V | f | v | | | ヲ | カ | ニ | ヨ | ρ | Σ |
| 7<br>0111 | CG RAM (8) | | ' | 7 | G | W | g | w | | | ァ | キ | ヌ | ラ | g | π |
| 8<br>1000 | CG RAM (1) | | ( | 8 | H | X | h | x | | | ィ | ク | ネ | リ | √ | x̄ |
| 9<br>1001 | CG RAM (2) | | ) | 9 | I | Y | i | y | | | ゥ | ケ | ノ | ル | ⁻¹ | y |
| A<br>1010 | CG RAM (3) | | * | : | J | Z | j | z | | | ェ | コ | ハ | レ | j | 千 |
| B<br>1011 | CG RAM (4) | | + | ; | K | [ | k | { | | | ォ | サ | ヒ | ロ | ˣ | 万 |
| C<br>1100 | CG RAM (5) | | , | < | L | ¥ | l | \| | | | ャ | シ | フ | ワ | ¢ | 円 |
| D<br>1101 | CG RAM (6) | | - | = | M | ] | m | } | | | ュ | ス | ヘ | ン | £ | ÷ |
| E<br>1110 | CG RAM (7) | | . | > | N | ^ | n | → | | | ョ | セ | ホ | ゛ | ñ | |
| F<br>1111 | CG RAM (8) | | / | ? | O | _ | o | ← | | | ッ | ソ | マ | ゜ | ö | █ |

Fig 20: character details in LCD

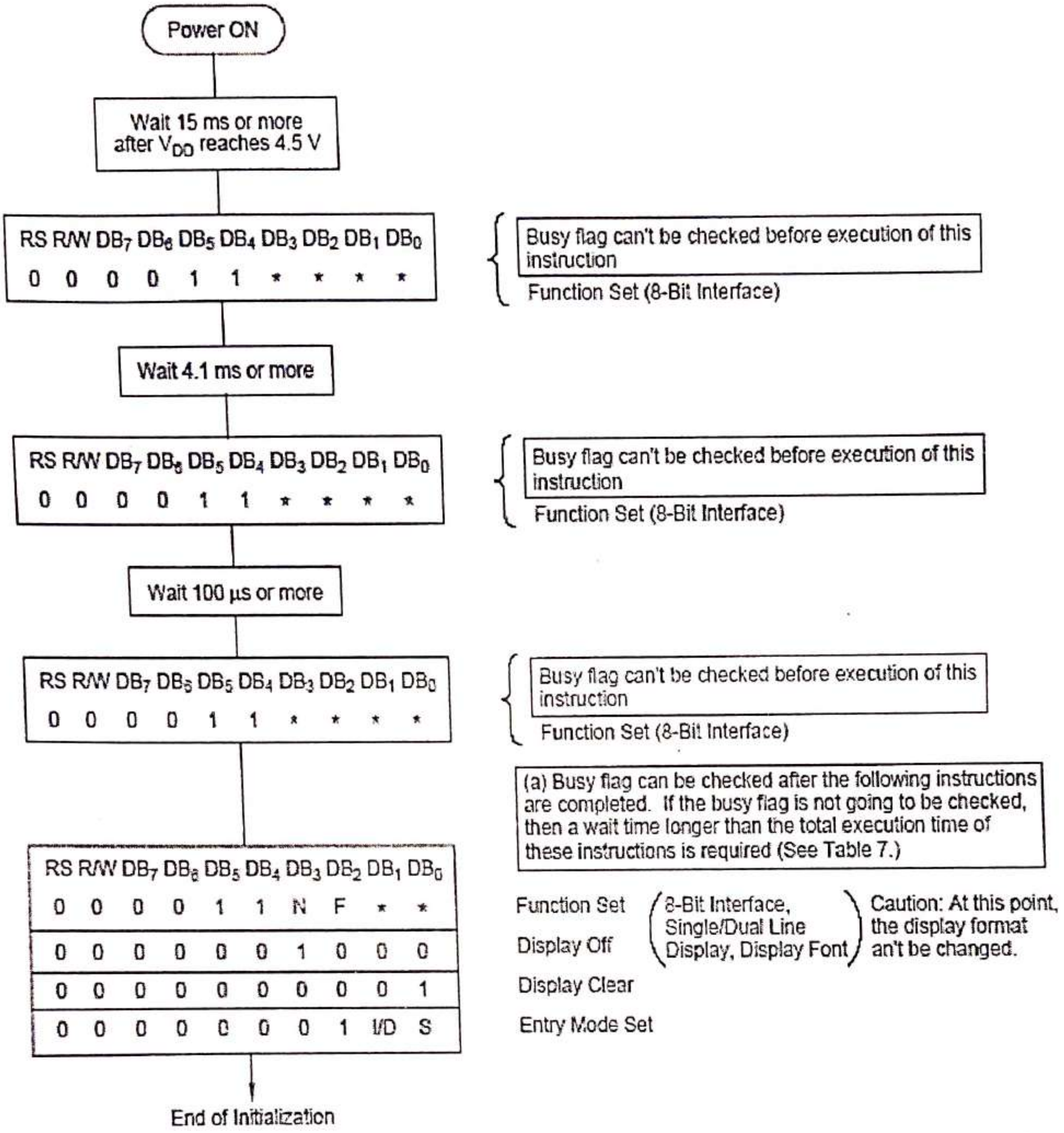
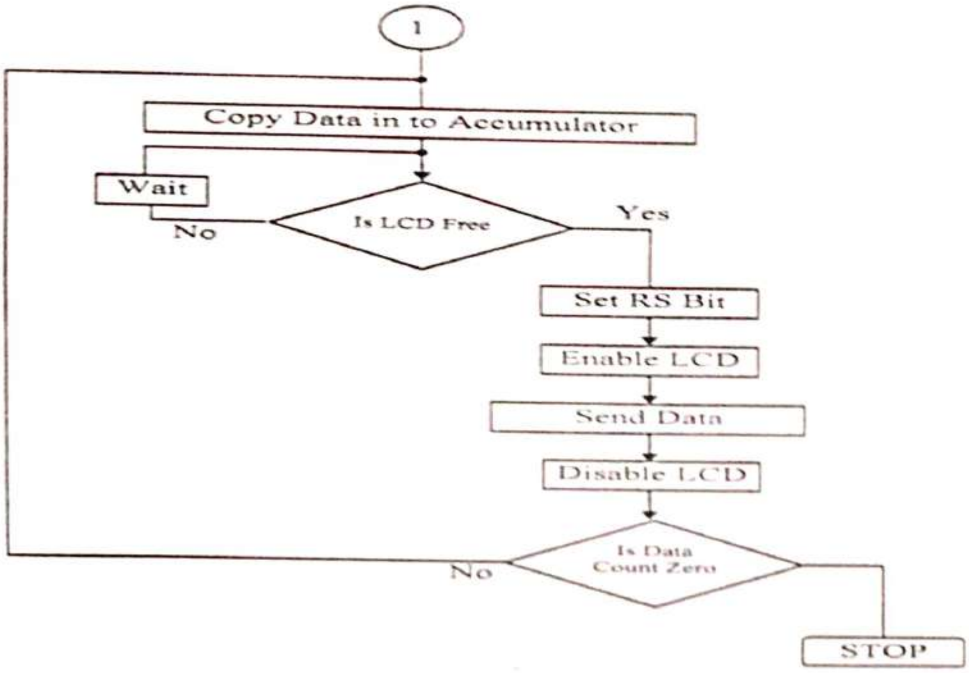
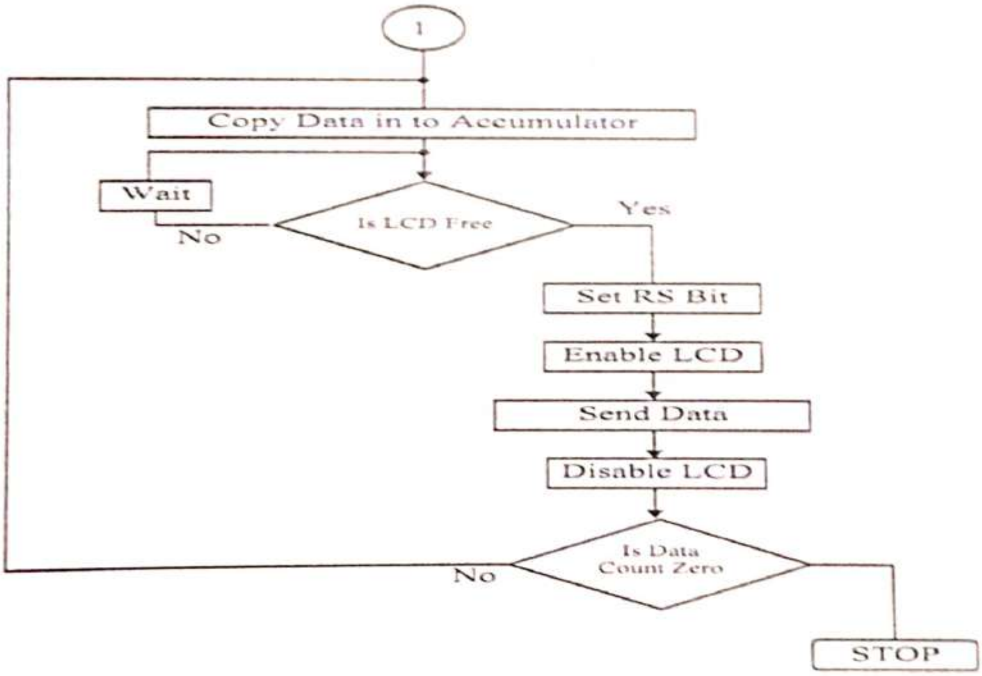

Power ON
Wait 15 ms or more after $V_{DD}$ reaches 4.5 V
RS R/W $DB_7$ $DB_6$ $DB_5$ $DB_4$ $DB_3$ $DB_2$ $DB_1$ $DB_0$
0 0 0 0 1 1 * * * *
Busy flag can't be checked before execution of this instruction
Function Set (8-Bit Interface)
Wait 4.1 ms or more
RS R/W $DB_7$ $DB_6$ $DB_5$ $DB_4$ $DB_3$ $DB_2$ $DB_1$ $DB_0$
0 0 0 0 1 1 * * * *
Busy flag can't be checked before execution of this instruction
Function Set (8-Bit Interface)
Wait 100 μs or more
RS R/W $DB_7$ $DB_6$ $DB_5$ $DB_4$ $DB_3$ $DB_2$ $DB_1$ $DB_0$
0 0 0 0 1 1 * * * *
Busy flag can't be checked before execution of this instruction
Function Set (8-Bit Interface)
(a) Busy flag can be checked after the following instructions are completed. If the busy flag is not going to be checked, then a wait time longer than the total execution time of these instructions is required (See Table 7.)
RS R/W $DB_7$ $DB_6$ $DB_5$ $DB_4$ $DB_3$ $DB_2$ $DB_1$ $DB_0$
0 0 0 0 1 1 N F * *
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 I/D S
Function Set
Display Off
Display Clear
Entry Mode Set
8-Bit Interface, Single/Dual Line Display, Display Font
Caution: At this point, the display format an't be changed.
End of Initialization

**Fig 21: flow chart of lcd**

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

## FLOWCHART:


1
Copy Data in to Accumulator
Wait
Is LCD Free
No
Yes
Set RS Bit
Enable LCD
Send Data
Disable LCD
Is Data Count Zero
No
STOP

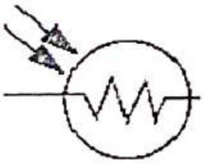

1
Copy Data in to Accumulator
Wait
Is LCD Free
No
Yes
Set RS Bit
Enable LCD
Send Data
Disable LCD
Is Data Count Zero
No
STOP

## Light Dependent Resistor

A photoresistor or light dependent resistor or cadmium sulfide (CdS) cell is a resistor whose resistance decreases with increasing incident light intensity. It can also be referred to as a photoconductor.

A photo resistor is made of a high resistance semiconductor. If light falling on the device is of high enough frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electron (and its hole partner) conduct electricity, there by lowering resistance.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, e.g. silicon. In intrinsic devices the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electron across the entire bandgap. Extrinsic devices have impurities, also called dopants, added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (i.e., longer wavelengths and lower frequencies) are sufficient to trigger the device. If a sample of silicon has some of its atoms replaced by phosphorus atoms (impurities), there will be extra electrons available for conduction. This is an example of an extrinsic semiconductor.



The symbol for a photoresistor

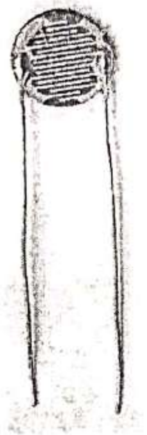
### Applications:

Photoresistors come in many different types. Inexpensive cadmium sulfide cells can be found in many consumer items such as camera light meters, street lights, clock radios, alarms, and outdoor clocks.

They are also used in some dynamic compressors together with a small incandescent lamp or light emitting diode to control gain reduction.

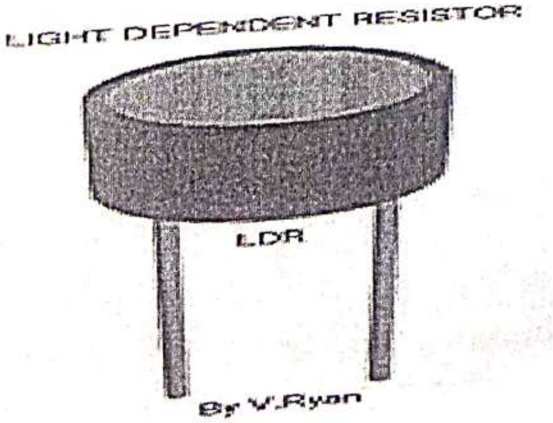
Lead sulfide (PbS) and indium antimonide (InSb) LDRs (light dependent resistor) are used for the mid infrared spectral region. Ge:Cu photoconductors are among the best far-infrared detectors available, and are used for infrared astronomy and infrared spectroscopy. )

**Transducers are used for changing energy types.**



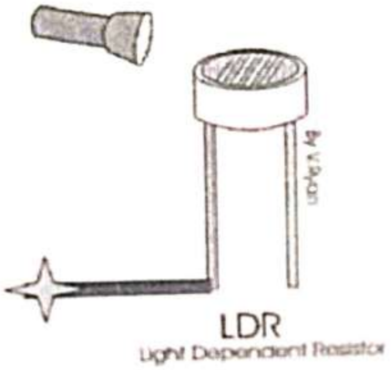
A light dependent resistor

A light dependent resistor is a small, round semiconductor. Light dependent resistors are used to re-charge a light during different changes in the light, or they are made to turn a light on during certain changes in lights. One of the most common uses for light dependent resistors is in traffic lights. The light dependent resistor controls a built in heater inside the traffic light, and causes it to recharge over night so that the light never dies. Other common places to find light dependent resistors are in: infrared detectors, clocks and security alarms.



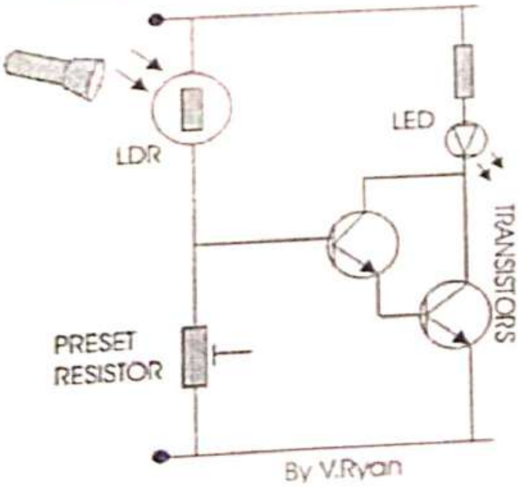
LDRs or Light Dependent Resistors are very useful especially in light/dark sensor circuits. Normally the resistance of an LDR is very high, sometimes as high as 1000 000 ohms, but when they are illuminated with light resistance drops dramatically.

The animation opposite shows that when the torch is turned on, the resistance of the LDR falls, allowing current to pass through it.



Circuit Wizard software has been used to display, the range of values of a ORP12,LDR . When a light level of 1000 lux (bright light) is directed towards it,the resistance is 400R (ohms).

When a light level of 10 lux (very low light level) is directed towards it, the resistance has risen dramatically to 10.43M (10430000 ohms).



**This is an example of a light sensor circuit :**

When the light level is low the resistance of the LDR is high. This prevents current from flowing to the base of the transistors. Consequently the LED does not light.

However, when light shines onto the LDR its resistance falls and current flows into the base of the first transistor and then the second transistor. The LED lights.

The preset resistor can be turned up or down to increase or decrease resistance, in this way it can make the circuit more or less sensitive.

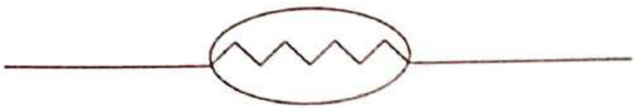


Fig 3.3.1: - Schematic Symbol of LDR

The photo resistor, or Light Dependent Resistor, finds many uses as a low cost photo sensitive element and was used for many years in photographic light meters as well as other applications.such as flame, smoke, and burgler detectors, card readers and lighting controls for street lamps.

**Units for the light intensity are Lux or Lumence.**

**Basic structure:**

Although there are many ways in which LDR's or photo resistors can be manufactured, ther are naturally a few more common methods that are seen. Essentially the LDR or photo resistor consists of a resistive material sensitive to light that is exposed to light. The photo resistive element comprises section of material with contacts at either end. Although many of the material used for light dependent resistors are semiconductors, when used as photo resistors, they are used only as a resistive element and there are no p-n junctions. Accordingly the devices purely passive.

A typical structure for a Light Dependent Resistor uses an active semiconductor layer that is deposited on an insulating substrate. The semiconductor is normally lightly doped to enable it to have the required level of conductivity. Contacts then placed either side of the exposed area. In many instances the area between the contacts is in the form of zig zag, or inter digital pattern. This maximizes the exposed area and by keeping the distance between the contacts small it enhances the gain.

It also possible to use a poly crystalline semiconductor that is deposited onto a substrate such as ceramic. This makes for a very low cost light dependent resistor.

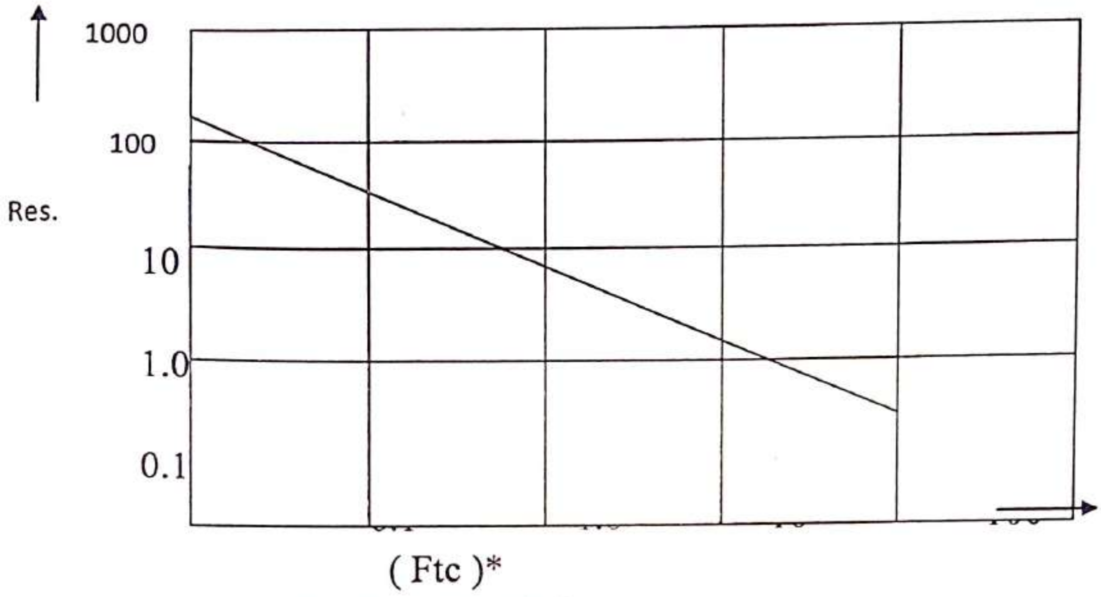
**Operation:**

Light Dependent Resistor made of a high resistance semiconductor, if light falling on the is of high enough efficiently, photon absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electron (and its hole partner) conduct electricity, thereby lowering resistance.

In intrinsic devices, the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electrons across the entire band gap. Extrinsic devices have impurities added , which have a ground state energy closer to the conduction band, since the electrons don't have so far to jump, lower energy photons ( i.e. longer wavelengths and lower frequencies ) will suffice to trigger the device.

## Characteristics of LDR:

The characteristics of LDR are shown below. Here the resistance variations are shown as a function of illumination. The resistance of LDR decreases with increasing incident light intensity



*1 Ftc = 10.764 lumens

Fig 3.3.2: - Characteristics of LDR

## LDR Applications:

LDR's are very useful especially in light/dark sensor circuits. Normally the resistance of LDR is very high, sometimes as high as 1000k ohms,but when they are illuminated with light, resistance drops immediately.

1. Camera light meters.
2. Clock radios.
3. Security alarms.
4. Optical switches.
5. Far infrared detector.
6. Streetlights.

### 3.3.5 Testing circuit of LDR:

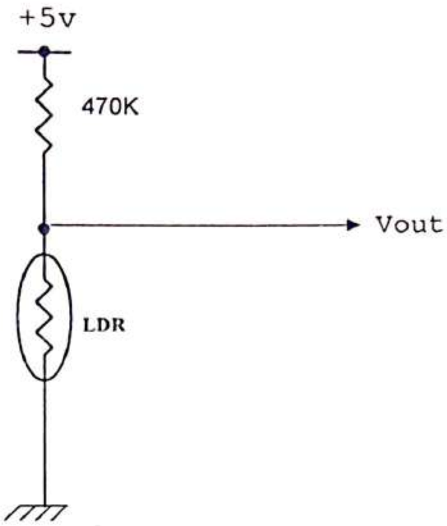


**Fig 3.3.3: - Testing Circuit of LDR**

## SOFTWARE EXPLANATION

### a) ABOUT SOFTWARE

Software's used are:
*Keil software for c programming

*Express PCB for lay out design

*Express SCH for schematic design

### What's New in µVision3?

µVision3 adds many new features to the Editor like Text Templates, Quick Function Navigation, and Syntax Coloring with brace high lighting Configuration Wizard for dialog based startup and debugger setup. µVision3 is fully compatible to µVision2 and can be used in parallel with µVision2.

## What is μVision3?

μVision3 is an IDE (Integrated Development Environment) that helps you write, compile, and debug embedded programs. It encapsulates the following components:

- A project manager.
- A make facility.
- Tool configuration.
- Editor.
- A powerful debugger.

To help you get started, several example programs (located in the **\C51\Examples, \C251\Examples, \C166\Examples**, and **\ARM\...\Examples**) are provided.

- **HELLO** is a simple program that prints the string "Hello World" using the Serial Interface.
- **MEASURE** is a data acquisition system for analog and digital systems.
- **TRAFFIC** is a traffic light controller with the RTX Tiny operating system.
- **SIEVE** is the SIEVE Benchmark.
- **DHRY** is the Dhrystone Benchmark.
- **WHETS** is the Single-Precision Whetstone Benchmark.

Additional example programs not listed here are provided for each device architecture.

## Building an Application in μVision2

To build (compile, assemble, and link) an application in μVision2, you must:

1. Select Project -(forexample,**166\EXAMPLES\HELLO\HELLO.UV2**).
2. Select Project - Rebuild all target files or Build target.

   μVision2 compiles, assembles, and links the files in your project.

## Creating Your Own Application in μVision2

**To create a new project in μVision2, you must:**

1. Select Project - New Project.
2. Select a directory and enter the name of the project file.
3. Select Project - Select Device and select an 8051, 251, or C16x/ST10 device from the Device Database™.
4. Create source files to add to the project.
5. Select Project - Targets, Groups, Files. Add/Files, select Source Group1, and add the source files to the project.
6. Select Project - Options and set the tool options. Note when you select the target device from the Device Database™ all special options are set automatically. You typically only need to configure the memory map of your target hardware. Default memory model settings are optimal for most applications.
7. Select Project - Rebuild all target files or Build target.

### Debugging an Application in µVision2

To debug an application created using µVision2, you must:

1. Select Debug - Start/Stop Debug Session.
2. Use the Step toolbar buttons to single-step through your program. You may enter **G, main** in the Output Window to execute to the main C function.
3. Open the Serial Window using the **Serial #1** button on the toolbar.

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

### Starting µVision2 and Creating a Project

µVision2 is a standard Windows application and started by clicking on the program icon. To create a new project file select from the µVision2 menu

**Project** – New Project…. This opens a standard Windows dialog that asks you

for the new project file name.

We suggest that you use a separate folder for each project. You can simply use the icon Create New Folder in this dialog to get a new empty folder. Then

select this folder and enter the file name for the new project, i.e. Project1.

µVision2 creates a new project file with the name PROJECT1.UV2 which contains

a default target and file group name. You can see these names in the Project

**Window – Files.**

Now use from the menu Project – Select Device for Target and select a CPU for your project. The Select Device dialog box shows the µVision2 device

database. Just select the micro controller you use. We are using for our examples the Philips 80C51RD+ CPU. This selection sets necessary tool options for the 80C51RD+ device and simplifies in this way the tool Configuration

**Building Projects and Creating a HEX Files**

Typical, the tool settings under Options – Target are all you need to start a new application. You may translate all source files and line the application with a click on the Build Target toolbar icon. When you build an application with syntax errors, µVision2 will display errors and warning messages in the Output Window – Build page. A double click on a message line opens the source file on the correct location in a µVision2 editor window. Once you have successfully generated your application you can start debugging.

After you have tested your application, it is required to create an Intel HEX file to download the software into an EPROM programmer or simulator. µVision2 creates HEX files with each build process when Create HEX files under Options for Target – Output is enabled. You may start your PROM programming utility after the make process when you specify the program under the option Run User Program #1.

## CPU Simulation

µVision2 simulates up to 16 Mbytes of memory from which areas can be mapped for read, write, or code execution access. The µVision2 simulator traps and reports illegal memory accesses. In addition to memory mapping, the simulator also provides support for the integrated peripherals of the various 8051 derivatives. The on-chip peripherals of the CPU you have selected are configured from the Device

## Database selection

you have made when you create your project target. Refer to page 58 for more Information about selecting a device. You may select and display the on-chip peripheral components using the Debug menu. You can also change the aspects of each peripheral using the controls in the dialog boxes.

## Start Debugging

You start the debug mode of µVision2 with the Debug – Start/Stop Debug Session command. Depending on the Options for Target – Debug Configuration, µVision2 will load the application program and run the start up

code µVision2 saves the editor screen layout and restores the screen layout of the last debug session. If the program execution stops, µVision2 opens an editor window with the source text or shows CPU instructions in the disassembly window. The next executable statement is marked with a yellow arrow. During debugging, most editor features are still available.

For example, you can use the find command or correct program errors. Program source text of your application is shown in the same windows. The µVision2 debug mode differs from the edit mode in the following aspects:

_ The "Debug Menu and Debug Commands" described on page 28 are Available. The additional debug windows are discussed in the following. _ The project structure or tool parameters cannot be modified. All build Commands are disabled.

### Disassembly Window

The Disassembly window shows your target program as mixed source and assembly program or just assembly code. A trace history of previously executed instructions may be displayed with Debug – View Trace Records. To enable the trace history, set Debug – Enable/Disable Trace Recording.

If you select the Disassembly Window as the active window all program step commands work on CPU instruction level rather than program source lines. You can select a text line and set or modify code breakpoints using toolbar buttons or the context menu commands.

You may use the dialog Debug – Inline Assembly... to modify the CPU instructions. That allows you to correct mistakes or to make temporary changes to the target program you are debugging.

## B)Keil Software

Installing the Keil software on a Windows PC

- Insert the CD-ROM in your computer's CD drive
- On most computers, the CD will "auto run", and you will see the Keil installation menu. If the menu does not appear, manually double click on the Setup icon, in the root directory: you will then see the Keil menu.
- On the Keil menu, please select "Install Evaluation Software". (You will not require a license number to install this software).
- Follow the installation instructions as they appear.

### Loading the Projects

The example projects for this book are NOT loaded automatically when you install the Keil compiler.

These files are stored on the CD in a directory "/Pont". The files are arranged by chapter: for example, the project discussed in Chapter 3 is in the directory "/Pont/Ch03_00-Hello".

Rather than using the projects on the CD (where changes cannot be saved), please copy the files from CD onto an appropriate directory on your hard disk.

**Note:** you will need to change the file properties after copying: file transferred from the CD will be 'read only'.

Configuring the Simulator

Open the Keil µVision2

go to Project – Open Project and browse for Hello in Ch03_00 in Pont and open it.

**Code:**

```
/*#include<reg52.h>

#define lcd_data P0

sbit lcd_rs=P0^0;

sbit lcd_rw=P0^1;

sbit lcd_en=P0^2;

sbit ldr1=P1^0;

sbit ldr2=P1^1;

sbit ldr3=P1^2;

sbit m1=P2^0;

sbit m2=P2^1;

void delay(unsigned int value)

{
```

```
unsigned int x,y;

for(x=0;x<300;x++)

for(y=0;y<value;y++);

}

void lcdcmd(unsigned char value)              // LCD COMMAND

 {



        lcd_data=value&(0xf0); //send msb 4 bits

    lcd_rs=0;          //select command register

    lcd_en=1;          //enable the lcd to execute command

        delay(3);

        lcd_en=0;

    lcd_data=((value<<4)&(0xf0));     //send lsb 4 bits

    lcd_rs=0;          //select command register

    lcd_en=1;          //enable the lcd to execute command

        delay(3);

        lcd_en=0;

 }

void lcd_init(void)

{
```

```
lcdcmd(0x02);

lcdcmd(0x02);

lcdcmd(0x28);  //intialise the lcd in 4 bit mode*/

/* lcdcmd(0x28);  //intialise the lcd in 4 bit mode*/

 /*lcdcmd(0x0c);  //cursor blinking

lcdcmd(0x06);      //move the cursor to right side

lcdcmd(0x01);      //clear the lcd

}

void lcddata(unsigned char value)

  {

     lcd_data=value&(0xf0); //send msb 4 bits

     lcd_rs=1;            //select data register

     lcd_en=1;            //enable the lcd to execute data

         delay(3);

         lcd_en=0;

     lcd_data=((value<<4)&(0xf0));      //send lsb 4 bits

     lcd_rs=1;            //select data register

     lcd_en=1;            //enable the lcd to execute data

         delay(3);

         lcd_en=0;
```

```
   delay(3);
  }
void lcd(unsigned char b[]) // send string to lcd
  {
unsigned char s,count=0;
for(s=0;b[s]!='\0';s++)
 {
  count++;
  if(s==16)
   lcdcmd(0xc0);
   if(s==32)
   {
   lcdcmd(1);
   count=0;
   }
  lcddata(b[s]);
 }
}


void main()
```

```
{
        lcd_rw=0;
        m1=m2=0;
        ldr1=ldr2=ldr3=1;
        lcd_init();
        lcd(" SOLAR TRACKING   ");
        lcdcmd(0xc4);
        lcd("SYSTEM");
        while(1)
        {
                if(ldr1==0 && ldr2==1 &&  ldr3==1)
                {
                        lcdcmd(1);
                        lcd("solar moving to position 1");
                        m1=1;
                        m2=0;
                        delay(100);
                        m1=m2=0;
                while(ldr1==0 && ldr2==1 &&  ldr3==1);
                }
```

```
if(ldr1==1 && ldr2==0 && ldr3==1)
{
        lcdcmd(1);
        lcd("solar moving to position 2");
        m1=1;
        m2=0;
        delay(100);
        m1=m2=0;
        while(ldr1==1 && ldr2==0 && ldr3==1);
}
if (ldr1==1 && ldr2==1 && ldr3==0)
{
        lcdcmd(1);
        lcd("solar moving to position 3");
        m1=1;
        m2=0;

        delay(100);
        m1=0;
        m2=0;
```

```
                delay(1000);

                m2=1;

                m1=0;

                delay(300);

                m1=0;

                m2=0;

                lcdcmd(0x01);

        lcd(" SOLAR TRACKING   ");

        lcdcmd(0xc4);

        lcd("SYSTEM");


            while(ldr1==1 && ldr2==1 && ldr3==0);

            }

        }

}

         */

         #include<reg52.h>

#define lcd_data P0

sbit lcd_rs=P0^0;

sbit lcd_rw=P0^1;
sbit lcd_en=P0^2;

sbit ldr1=P1^0;

sbit ldr2=P1^1;

sbit ldr3=P1^2;

sbit m1=P2^0;

sbit m2=P2^1;

void delay(unsigned int value)

{

unsigned int x,y;

for(x=0;x<300;x++)

for(y=0;y<value;y++);

}

void lcdcmd(unsigned char value)                    // LCD COMMAND

 {

        lcd_data=value&(0xf0); //send msb 4 bits

    lcd_rs=0;          //select command register

    lcd_en=1;          //enable the lcd to execute command

        delay(3);

        lcd_en=0;

    lcd_data=((value<<4)&(0xf0));     //send lsb 4 bits
```

```
   lcd_rs=0;         //select command register
   lcd_en=1;         //enable the lcd to execute command
        delay(3);
        lcd_en=0;
 }
void lcd_init(void)
{
 lcdcmd(0x02);
 lcdcmd(0x02);
 lcdcmd(0x28);  //intialise the lcd in 4 bit mode*/
 lcdcmd(0x28);  //intialise the lcd in 4 bit mode*/
 lcdcmd(0x0c);     //cursor blinking
 lcdcmd(0x06);     //move the cursor to right side
 lcdcmd(0x01);     //clear the lcd
 }
 void lcddata(unsigned char value)
  {
     lcd_data=value&(0xf0); //send msb 4 bits
     lcd_rs=1;         //select data register
     lcd_en=1;         //enable the lcd to execute data
```

```
          delay(3);

          lcd_en=0;

     lcd_data=((value<<4)&(0xf0));     //send lsb 4 bits

     lcd_rs=1;          //select data register

     lcd_en=1;          //enable the lcd to execute data

          delay(3);

          lcd_en=0;

    delay(3);

  }

void lcd(unsigned char b[]) // send string to lcd

  {

unsigned char s,count=0;

for(s=0;b[s]!='\0';s++)

 {

  count++;

  if(s==16)

   lcdcmd(0xc0);

   if(s==32)

   {

   lcdcmd(1);
```

```
    count=0;

    }

   lcddata(b[s]);

  }

 }

void main()

{

        lcd_rw=0;

        m1=m2=0;

        ldr1=ldr2=ldr3=1;

        lcd_init();

        lcd(" SOLAR TRACKING   ");

        lcdcmd(0xc4);

        lcd("SYSTEM");

        lp:

        while(1)

        {

                    while(ldr1==1);

                         lcdcmd(1);

                         lcd("solar moving to position 1");
        m1=1;
        m2=0;
        delay(100);
        m1=m2=0;
    while(ldr2==1);

        lcdcmd(1);
        lcd("solar moving to position 2");
        m1=1;
        m2=0;
        delay(100);
        m1=m2=0;
        while(ldr3==1);

        lcdcmd(1);
        lcd("solar moving to position 3");
        m1=1;
        m2=0;

        delay(100);
```

```
            m1=0;

            m2=0;

            delay(1000);

            m2=1;

            m1=0;

            delay(300);

            m1=0;

            m2=0;

            lcdcmd(0x01);

    lcd(" SOLAR TRACKING   ");

    lcdcmd(0xc4);

    lcd("SYSTEM");

    goto lp;




    }

}
```

# DECLARATION

*I here by declare that the entire work for project contained in this* Dissertation entitled

## "SOLAR TRACKING SYSTEM"

Submitted in the partial fulfilment of B.SC-(ELECTRONICS CLUSTER), of V.S.R GOVT DEGREE&P.G COLLEGE,MOVVA was done by me

Under the guidance of

Smt.S.KIRANMAYI, M.Sc., M.Phil., Lecturer in Electronics.

Department of Electronics
V.S.R GOVT DEGREE& P.G COLLEGE
MOVVA.

Signature of the students

1. E.S. Phanindra
2. G. Sandeep
3. K.L.S.V. Rao
4. K.V.S Rao.
5. K.D. Saikrishna.