1. INTRODUCTION

The project is mainly used to control the on-off action of a motor in the field based on the dry and wet conditions of the field using GSM Technology.

The Main Components of Project

- Microcontroller
- Vehicle or Robot
- DC Motors, to run the Vehicle
- Sensing logic
- LCD, to display the status of the field and the communication between circuit and the user mobile.
- Power Supply

Working:

This project is developed based on EMBEDDED and GSM Technology. When a field is in the dry condition, the sensing logic senses the state of the field and intimates it to the microcontroller. It in response makes the motor on. We can know the status of the field by sending a message to the GSM modem which is placed at the field. Through our mobile we can switch on-off the motor by sending the respective commands to the kit through the GSM modem. Thus the irrigation motor can be controlled through our mobiles using GSM technology.

BLOCK DIAGRAM

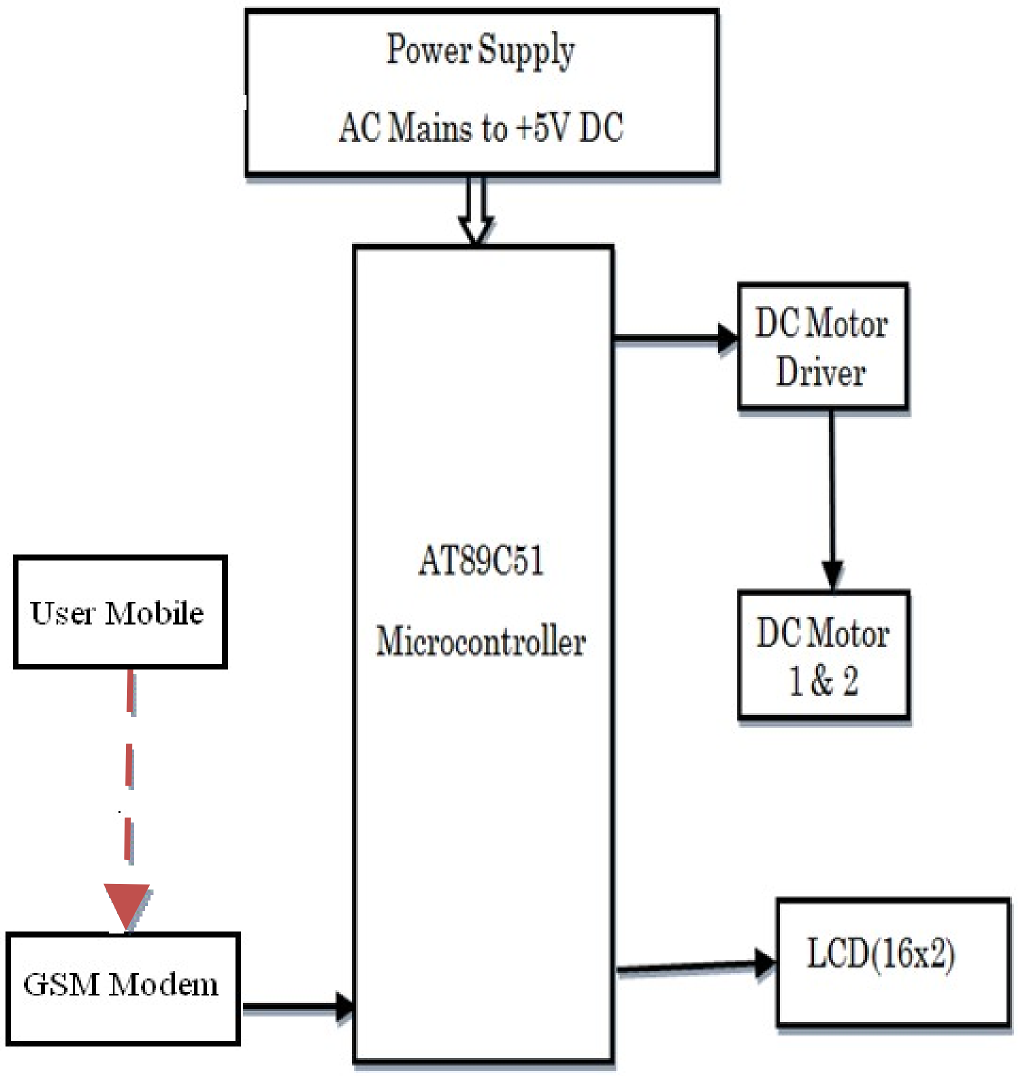


Figure 1.1: Block Diagram of Project

2. EMBEDDED SYSTEMS

An **Embedded System** is a special-purpose computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is usually *embedded* as part of a complete device including hardware and mechanical parts..

Embedded system controls many of the common devices. Physically, embedded systems range from portable devices such as digital watches and MP4 players. Now it ranges to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

Embedded processors can be broken into two broad categories: ordinary microprocessors (µP) and microcontrollers (µC), which have many more peripherals on chip, reducing cost and size.

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reason such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Examples of Embedded Systems

- Automatic teller machines (ATMs)
- Cellular telephones and telephone switches
- Home automation products, such as thermostats, air conditioners, sprinklers, and security monitoring systems
- Handheld calculators and computers
- Household appliances, including microwave ovens, washing machines, television sets, DVD players and recorders
- Medical equipment
- Industrial controllers for remote machine operation.

An embedded system is not always a separate block - very often it is physically built-in to the device it is controlling the software written for embedded systems is often called firmware, and is stored in read-only memory or Flash memory chips rather than a disk drive. It often runs with limited computer hardware resources: small or no keyboard, screen, and little memory.

User Interfaces

Embedded systems range from no user interface at all - dedicated only to one task - to full user interfaces similar to desktop operating systems in devices such as PDAs.

A full graphical screen, with touch sensing or screen-edge buttons provides flexibility while minimizing space used the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what's desired.

The rise of the World Wide Web has given embedded designers another quite different option providing a web page interface over a network connection. This avoids the cost of a sophisticated display, yet provides complex input and display capabilities when needed, on another computer. This is successful for remote, permanently installed equipment. In particular, routers take advantage of this ability.

Figure 2.1: Embedded Systems

3. POWER SUPPLY

The power supply unit is used to provide a constant 5V of DC supply from a 230V of AC supply. These 5V DC will acts as power to different standard circuits. It mainly uses 3 devices

1. Bridge wave rectifier
2. Voltage regulator

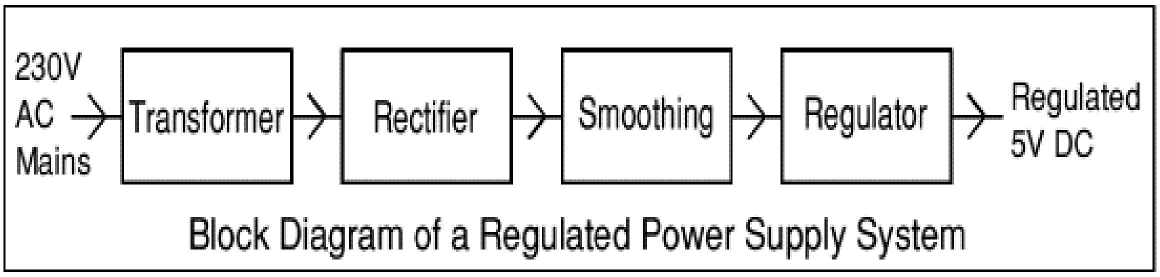


Figure 3.1: Block Diagram Of Power Supply

BRIDGE WAVE RECTIFIER

A **rectifier** is an electrical device that converts alternating current (AC) to direct current (DC), a process known as rectification. The term *rectifier* describes a *diode* that is being used to convert AC to DC.

A bridge-wave rectifier converts the whole of the input waveform to one of constant polarity (positive or negative) at its output. Bridge-wave rectifier converts both polarities of the input waveform to DC (direct current), and is more efficient. However, in a circuit with a center tapped transformer (9-0-9) is used.

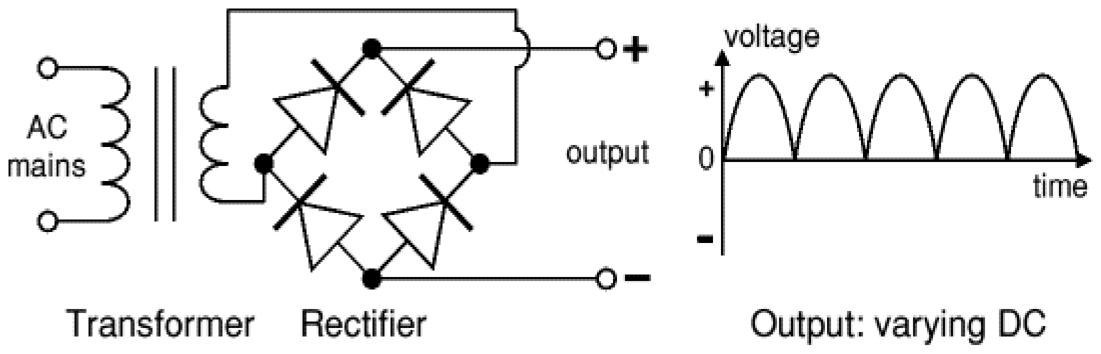


Figure 3.2: Bridge Wave Rectifier

For single-phase AC, if the transformer is center-tapped, then two diodes back-to-back(i.e. anodes-to-anode or cathode-to-cathode) can form a full-wave rectifier. Many windings are required on the transformer secondary to obtain the same output voltage.

In this only two diodes are activated at a time i.e. D1 and D3 activate for positive cycle and D2 and D4 activates for negative half cycle. D2 and D4 convert negative cycle to positive cycle as it as negative supply and negative cycle as positive cycle at its output.

VOLTAGE REGULATOR

This is most common voltage regulator that is still used in embedded designs. LM7805 voltage regulator is a linear regulator. With proper heat sink these LM78xx types can handle even more than 1A current. They also have Thermal overload protection, Short circuit protection.

This will connect at the output of rectifier to get constant Dc supply instead of ripple voltages. It mainly consists of 3 pins

1. Input voltage
2. Output voltage
3. Ground

The capacitor C2 is used to get thee ripple voltage as input to regulator instead of full positive cycles.

Vr = I load/Xc

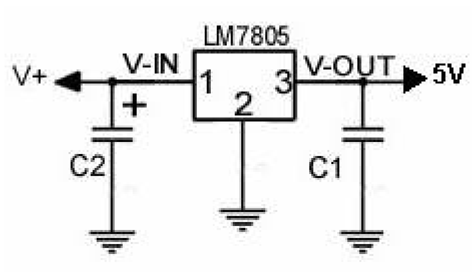


Figure 3.3: Voltage Regulator

For some devices we require 12V/9V/4V Dc supply at that time we go for 7812/7809/7804 regulator instead of 7805 regulator. It also have same feature and pins has 7805 regulator except output is of 12V/9V/4V instead of 5V.

The general circuit diagram for total power supply to any embedded device is as shown below.

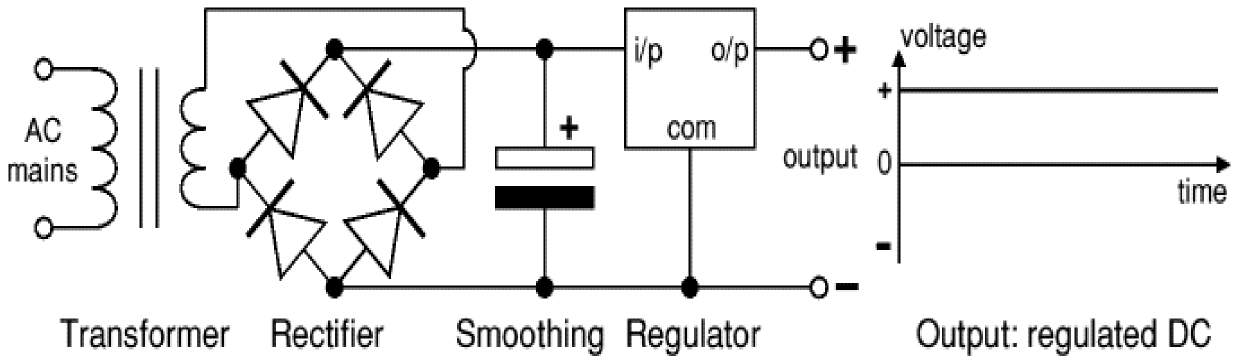


Figure 3.4: Circuit Diagram Of Power Supply

4. MICRO CONTROLLER(P89V51RD2)

General description

The P89V51RD2 are 80C51 microcontrollers with 16kB Flash and 1024 bytes of data RAM. A key feature of the P89V51RD2 is its X2 mode option. The design engineer can choose to run the application with the conventional 80C51 clock rate (12 clocks per machine cycle) or select the X2 mode (6 clocks per machine cycle) to achieve twice the throughput at the same clock frequency. Another way to benefit from this feature is to keep the same performance by reducing the clock frequency by half, thus dramatically reducing the EMI.

The Flash program memory supports both parallel programming and in serial **In-System Programming** (ISP). Parallel programming mode offers gang-programming at high speed, reducing programming costs and time to market. ISP allows a device to be reprogrammed in the end product under software control. The capability to field/update the application firmware makes a wide range of applications possible.

The P89V51RD2 is also **In-Application Programmable** (IAP), allowing the Flash program memory to be reconfigured even while the application is running.

Features

80C51 Central Processing Unit

- 5 V Operating voltage from 0 MHz to 40 MHz
- 16/32/64 kB of on-chip Flash user code memory with ISP and IAP Supports 12-clock (default) or 6-clock mode selection via software
- SPI (Serial Peripheral Interface) and enhanced UART
- PCA (Programmable Counter Array) with PWM and Capture/Four 8-bit I/O ports with three high-current Port 1 pins (16Three 16-bit timers/counters
- Programmable watchdog timer
- Eight interrupt sources with four priority levels
- Second DPTR register

- Low EMI mode (ALE inhibit)
- TTL- and CMOS-compatible logic levels
- Brown-out detection
- Low power modes
  - Power-down mode with external interrupt wake-up
  - Idle mode
- DIP40, PLCC44 and TQFP44 packages

Block diagram

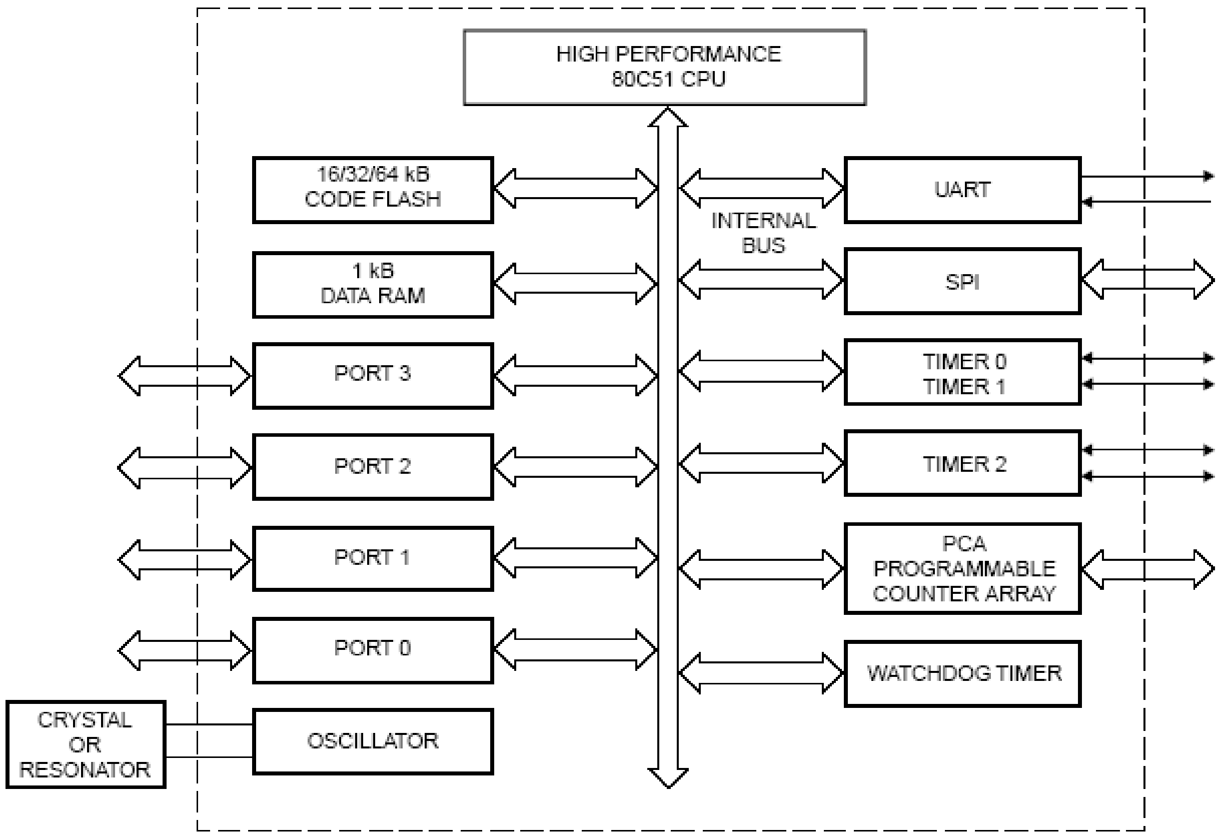


Figure4.1: P89V51RB2/RC2/RD2 block diagram.

Pin Configuration:

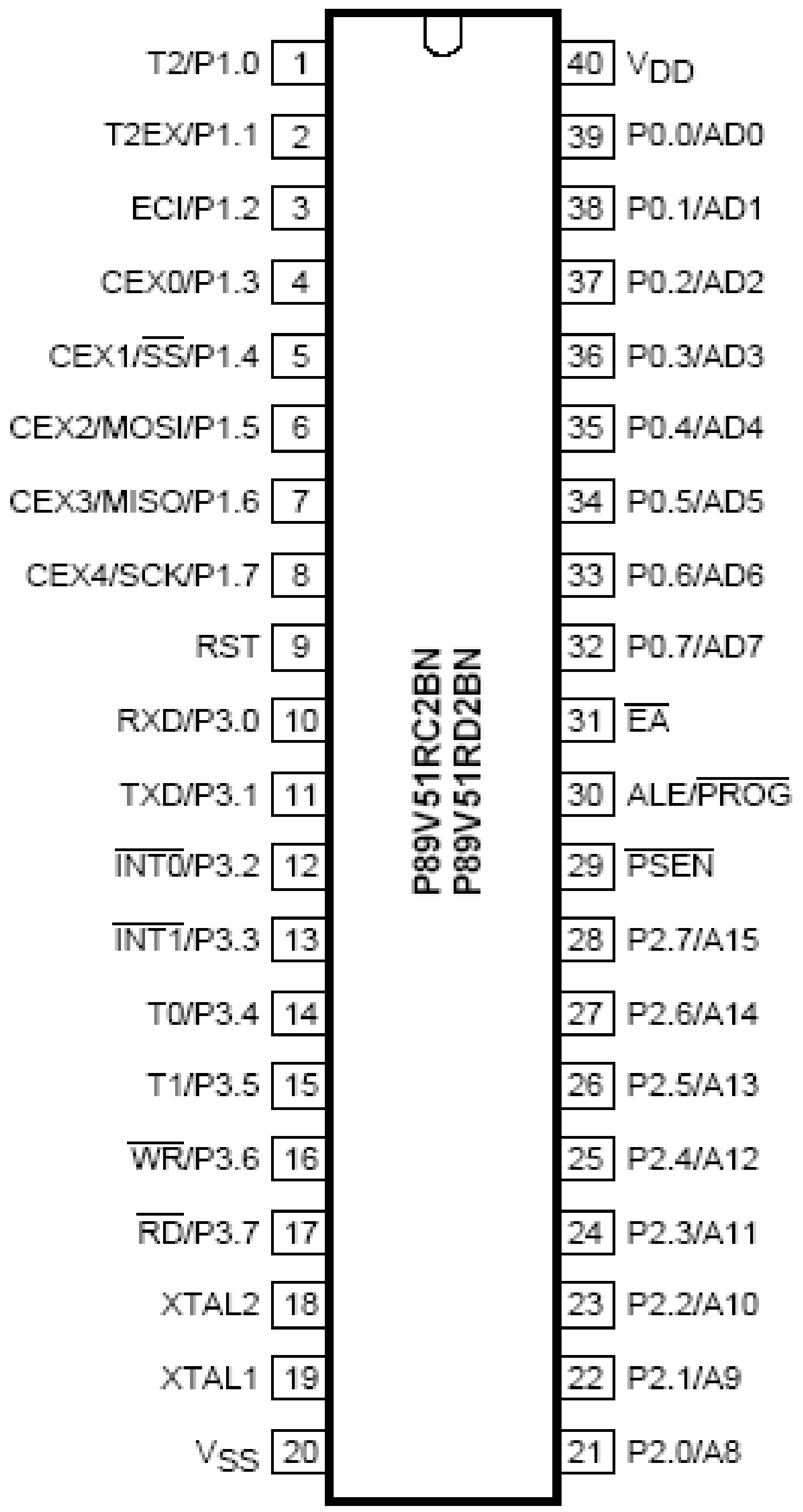


Figure 4.2; Pin configuration of P89V51RD2BN

PIN 1–8: PORT 1: (p1.0 to P1.7): Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 pins are pulled high by the internal pull-ups when '1's are written to them and can be used as inputs in this state. As inputs, Port 1 pins that are externally pulled LOW will source current (IIL) because of the internal pull-ups. P1.5, P1.6, P1.7 have high current drive of 16 mA. Port 1 also receives the low-order address bytes during the external host mode programming and verification.

P1.0: T2: External count input to Timer/Counter 2 or Clock-out from Timer/Counter 2

P1.1: T2EX: Timer/Counter 2 capture/reload trigger and direction control

P1.2: ECI: External clock input. This signal is the external clock input for the PCA.

P1.3: CEX0: Capture/compare external I/O for PCA Module 0. Each capture/compare module connects to a Port 1 pin for external I/O. When not used by the PCA, this pin can handle standard I/O.

P1.4: SS: Slave port select input for SPI. **CEX1**: Capture/compare external I/O for PCA Module 1

P1.5: MOSI: Master Output Slave Input for SPI **CEX2**: Capture/compare external I/O for PCA Module 2

P1.6: **MISO**: Master Input Slave Output for SPI **CEX3**: Capture/compare external I/O for PCA Module 3

P1.7: **SCK**: Master Output Slave Input for SPI **CEX4**: Capture/compare external I/O for PCA Module 4

PIN 9: RESET SIGNAL: High logical state on this input halts the MCU and clears all the registers. Bringing this pin back to logical state zero starts the program a new as if the power had just been turned on. In another words, positive voltage impulse on this pin resets the MCU. Depending on the device's purpose and environs, this pin is usually connected to the push-button, reset-upon-start circuit or a brown out reset circuit (covered in the previous chapter). The image shows one simple circuit for safe reset upon starting the controller. It is utilized in situations when power fails to reach its optimal voltage.

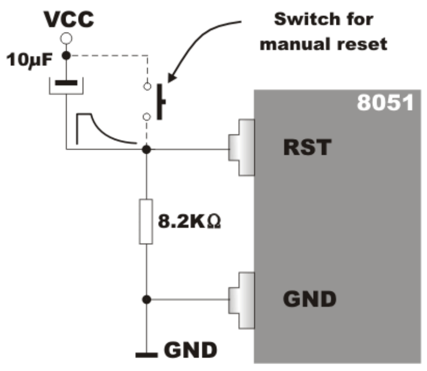


Figure 4.3: Reset Switch

PIN 10-17: Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins are pulled HIGH by the internal pull-ups when '1's are written to them and can be used as inputs in this state. As inputs, Port 3 pins that are externally pulled LOW will source current (IIL) because of the internal pull-ups. Port3 also receives some control signals and a partial of high-order address bits during the external host mode programming and verification.

P3.0: RXD: serial input port

P3.1: TXD: serial output port

P3.2: INT0: external interrupt 0 input

P3.3: INT1: external interrupt 1 input

P3.4: T0: external count input to Timer/Counter 0

P3.5: T1: external count input to Timer/Counter 1

P3.6: WR: external data memory write strobe

P3.7: RD: external data memory read strobe

PIN 18: XTAL2: Crystal 2: Output from the inverting oscillator amplifier.

PIN 19: XTAL1: Crystal 1: Input to the inverting oscillator amplifier and input to the internal clock generator circuits.

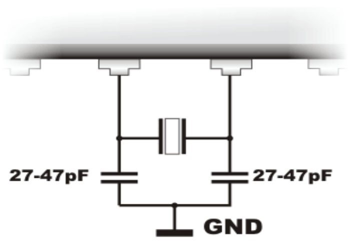


Figure 4.4: Description of XTAL1 and XTAL2 pins

PIN 20 : Ground

PIN 21-28: (P2.0 toP2.7): **Port 2**: Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins are pulled HIGH by the internal pull-ups when ‘1’s are written to them and can be used as inputs in this state. As inputs, Port 2 pins that are externally pulled LOW will source current (IIL) because of the internal pull-ups. Port 2 sends the high-order address byte during fetches from external program memory and during accesses to external Data Memory that use 16-bit address (MOVX@DPTR). In this application, it uses strong internal pull-ups when transitioning to ‘1’s. Port 2 also receives some control
signals and a partial of high-order address bits during the external host mode programming and verification.

PIN 29: Program Store Enable: PSEN is the read strobe for external program memory. When the device is executing from internal program memory, PSEN is inactive (HIGH). When the device 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. A forced HIGH-to-LOW input transition on the PSEN pin while the RST input is continually held HIGH for more than 10 machine cycles will cause the device to enter external host mode programming.

PIN 30: Address Latch Enable: ALE is the output signal for latching the low byte of the address during an access to external memory. This pin is also the programming pulse input (PROG) for flash programming. Normally the ALE[1] is emitted at a constant rate of 1 ■6 the crystal frequency[2] and can be used for external timing and clocking. One ALE pulse is skipped during each access to external data memory. However, if AO is set to ‘1’, ALE is disabled.

PIN 31: External Access Enable: EA must be connected to VSS in order to enable the device to fetch code from the external program memory. EA must be strapped to VDD for internal program execution. However, Security lock level 4 will disable EA, and program execution is only possible from internal program memory. The EA pin can tolerate a high voltage of 12 V.

PIN 32 TO 39: P0.0 – P0.7: Port 0: Port 0 is an 8-bit open drain bi-directional I/O port. Port 0 pins that have ‘1’s written to them float, and in this state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external code and data memory. In this application, it uses strong internal pull-ups when transitioning to ‘1’s. Port 0 also receives the code bytes during the external host mode programming, and outputs the code bytes during the external host mode verification. External pull-ups are required during program verification or as a general purpose I/O port.

PIN 40: VDD: Power supply

Special function registers

Remark: Special Function Registers (SFRs) accesses are restricted in the following ways:

• User must **not** attempt to access any SFR locations not defined.

• Accesses to any defined SFR locations must be strictly for the functions for the SFRs.

• SFR bits labeled '-', '0' or '1' can **only** be written and read as follows:

– '-' Unless otherwise specified, **must** be written with '0', but can return any value when read (even if it was written with '0'). It is a reserved bit and may be used in future derivatives.

– '0' **must** be written with '0', and will return a '0' when read.

– '1' **must** be written with '1', and will return a '1' when read.

4.2 TYPES OF MEMORY

Memory organization

The device has separate address spaces for program and data memory.

Flash program memory bank selection

There are two internal flash memory blocks in the device. Block 0 has 16/32/64 kB and is organized as 128/256/512 sectors, each sector consists of 128 Bytes. Block contains the IAP/ISP routines and may be enabled such that it overlays the first 8 kB of the user code memory. The overlay function is controlled by the combination of the Software Reset Bit (SWR) at FCF.1 and the Bank Select Bit (BSEL) at FCF.0. The combination of these bits and the memory source used for instructions is shown in the below Table.

Table: Code memory bank selection

SWR (FCF.1)	**BSEL (FCF.0)**	**addresses from 0000h to 1FFFh**	**addresses above 1FFFh**
0	0	Bootcode (in Block 1)	User code (in Block 0)
0	1	User code (in Block 0)	
1	0		
1	1		

Access to the IAP routines in Block 1 may be enabled by clearing the BSEL bit (FCF.0), provided that the SWR bit (FCF.1) is cleared. Following a power-on sequence, the bootcode is automatically executed and attempts to autobaud to a host. If no autobaud occurs within approximately 400 ms and the SoftICE flag is not set, control will be passed to the user code. A software reset is used to accomplish this control transfer and as a result the SWR bit will remain set. **Therefore the user's code will need to clear the SWR bit in order to access the IAP routines in Block**

However, caution must be taken when dynamically changing the BSEL bit. Since this will cause different physical memory to be mapped to the logical program address space, the user must avoid clearing the BSEL bit when executing user code within the address range 0000H to 1FFFH.

Power-on reset code execution

At initial power up, the port pins will be in a random state until the oscillator has started and the internal reset algorithm has weakly pulled all pins high. Powering up the device without a valid reset could cause the MCU to start executing instructions from an indeterminate location. Such undefined states may inadvertently corrupt the code in the flash. A system reset will not affect the 1 kB of on-chip RAM while the device is running, however, the contents of the on-chip RAM during power up are indeterminate.

When power is applied to the device, the RST pin must be held high long enough for the oscillator to start up (usually several milliseconds for a low frequency crystal), in addition to two machine cycles for a valid power-on reset. An example of a method to extend the RST signal is to implement a RC circuit by connecting the RST pin to VDD through a 10 F capacitor and to VSS through an 8.2KW resistor as shown in the above table.

Note that if an RC circuit is being used, provisions should be made to ensure the VDD rise time does not exceed 1 millisecond and the oscillator start-up time does not exceed 10 milliseconds.

For a low frequency oscillator with slow start-up time the reset signal must be extended in order to account for the slow start-up time. This method maintains the necessary relationship between VDD and RST to avoid programming at an indeterminate location, which may cause corruption in the code of the flash. The power-on detection is designed to work during initial power up, before the voltage reaches the brown-out detection level. The POF flag in the PCON register is set to indicate an initial power up condition. The POF flag will remain active until cleared by software.

Following a power-on or external reset the P89V51RB2/RC2/RD2 will force the SWR and BSEL bits (FCF[1:0]) = 00. This causes the bootblock to be mapped into the lower 8 kB of code memory and the device will execute the ISP code in the boot block and attempt to autobaud to the host. If the autobaud is successful the device will remain in ISP mode. If, after approximately 400 ms, the autobaud is unsuccessful the boot block code will check to see if the SoftICE flag is set (from a previous programming operation). If the SoftICE flag is set the device will enter SoftICE mode. If the SoftICE flag is cleared, the bootcode will execute a software reset causing the device to execute the user code from block 0 starting at address 0000h. Note that an external reset applied to the RST pin has the same effect as a power-on reset.

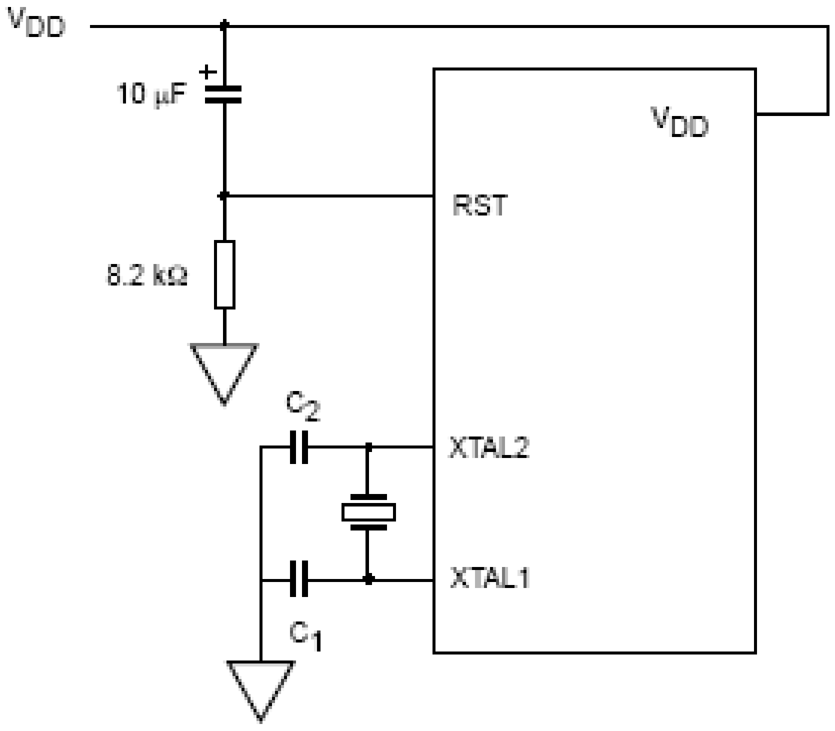


Figure4.5:Power-on reset circuit.

Software reset

A software reset is executed by changing the SWR bit (FCF.1) from '0' to '1'. A software reset will reset the program counter to address 0000H and force both the SWR and BSEL bits (FCF[1:0]) =10. This will result in the lower 8 kB of the user code memory being mapped into the user code memory space. Thus the user's code will be executed starting at address 0000h. A software reset will not change WDTC.2 or RAM data. Other SFRs will be set to their reset values.

Brown-out detect reset

The device includes a brown-out detection circuit to protect the system from severe supply voltage fluctuations. The P89V51RB2/RC2/RD2's brown-out detection threshold is 2.35 V. When VDD drops below this voltage threshold, the brown-out detect triggers the circuit to generate a brown-out interrupt but the CPU still runs until the supplied voltage returns to the brown-out detection voltage VBOD. The default operation for a brown-out detection is to cause a processor reset.

VDD must stay below VBOD at least four oscillator clock periods before the brown-out detection circuit will respond.

Brown-out interrupt can be enabled by setting the EBO bit (IEA.3). If EBO bit is set and a brown-out condition occurs, a brown-out interrupt will be generated to execute the program at location 004BH. It is required that the EBO bit be cleared by software after the brown-out interrupt is serviced. Clearing EBO bit when the brown-out condition is active will properly reset the device. If brown-out interrupt is not enabled, a brown-out condition will reset the program to resume execution at location 0000H.

A brown-out detect reset will clear the BSEL bit (FCF.0) but will not change the SWR bit (FCF.1) and therefore will not change the banking of the lower 8 kB of user code memory space.

Data RAM memory

The data RAM has 1024 bytes of internal memory. The device can also address up to 64 kB for external data memory.

Expanded data RAM addressing

The P89V51RB2/RC2/RD2 has 1 kB of RAM.

The device has four sections of internal data memory:

1. The lower 128 bytes of RAM (00H to 7FH) are directly and indirectly addressable.

2. The higher 128 bytes of RAM (80H to FFH) are indirectly addressable.
3. The special function registers (80H to FFH) are directly addressable only.
4. The expanded RAM of 768 bytes (00H to 2FFH) is indirectly addressable by the move external instruction (MOVX) and clearing the EXTRAM bit.

Since the upper 128 bytes occupy the same addresses as the SFRs, the RAM must be accessed indirectly. The RAM and SFRs space are physically separate even though they have the same addresses.

When instructions access addresses in the upper 128 bytes (above 7FH), the MCU determines whether to access the SFRs or RAM by the type of instruction given. If it is indirect, then RAM is accessed. If it is direct, then an SFR is accessed. See the examples below.

Indirect Access:

MOV@R0, #data; R0 contains 90H

Register R0 points to 90H which is located in the upper address range. Data in '#data' is written to RAM location 90H rather than port 1.

Direct Access:

MOV90H, #data; write data to P1

Data in '#data' is written to port 1. Instructions that write directly to the address write to the SFRs.

To access the expanded RAM, the EXTRAM bit must be cleared and MOVX instructions must be used. The extra 768 bytes of memory is physically located on the chip and logically occupies the first 768 bytes of external memory (addresses 000H to 2FFH).

When EXTRAM = 0, the expanded RAM is indirectly addressed using the MOVX instruction in combination with any of the registers R0, R1 of the selected bank or DPTR. Accessing the expanded RAM does not affect ports P0, P3.6 (WR), P3.7 (RD), or P2. With EXTRAM = 0, the expanded RAM can be accessed as in the following example.

Expanded RAM Access (Indirect Addressing only):

MOVX@DPTR, A DPTR contains 0A0H

DPTR points to 0A0H and data in 'A' is written to address 0A0H of the expanded RAM rather than external memory. Access to external memory higher than 2FFH using the MOVX instruction will access external memory (0300H to FFFFH) and will perform in the same way as the standard 8051, with P0 and P2 as data/address bus, and P3.6 and P3.7 as write and read timing signals.

When EXTRAM = 1, MOVX @Ri and MOVX @DPTR will be similar to the standard 8051. Using MOVX @Ri provides an 8-bit address with multiplexed data on Port 0.

Other output port pins can be used to output higher order address bits. This provides external paging capabilities. Using MOVX @DPTR generates a 16-bit address. This allows external addressing up the 64 kB. Port 2 provides the high-order eight address bits (DPH), and Port 0 multiplexes the low order eight address bits (DPL) with data.

Both MOVX @Ri and MOVX @DPTR generates the necessary read and write
signals (P3.6 - WR and P3.7 - RD) for external memory use. Table 9 shows external data memory RD, WR operation with EXTRAM bit.

The stack pointer (SP) can be located anywhere within the 256 bytes of internal RAM (lower 128 bytes and upper 128 bytes). The stack pointer may not be located in any part of the expanded RAM.

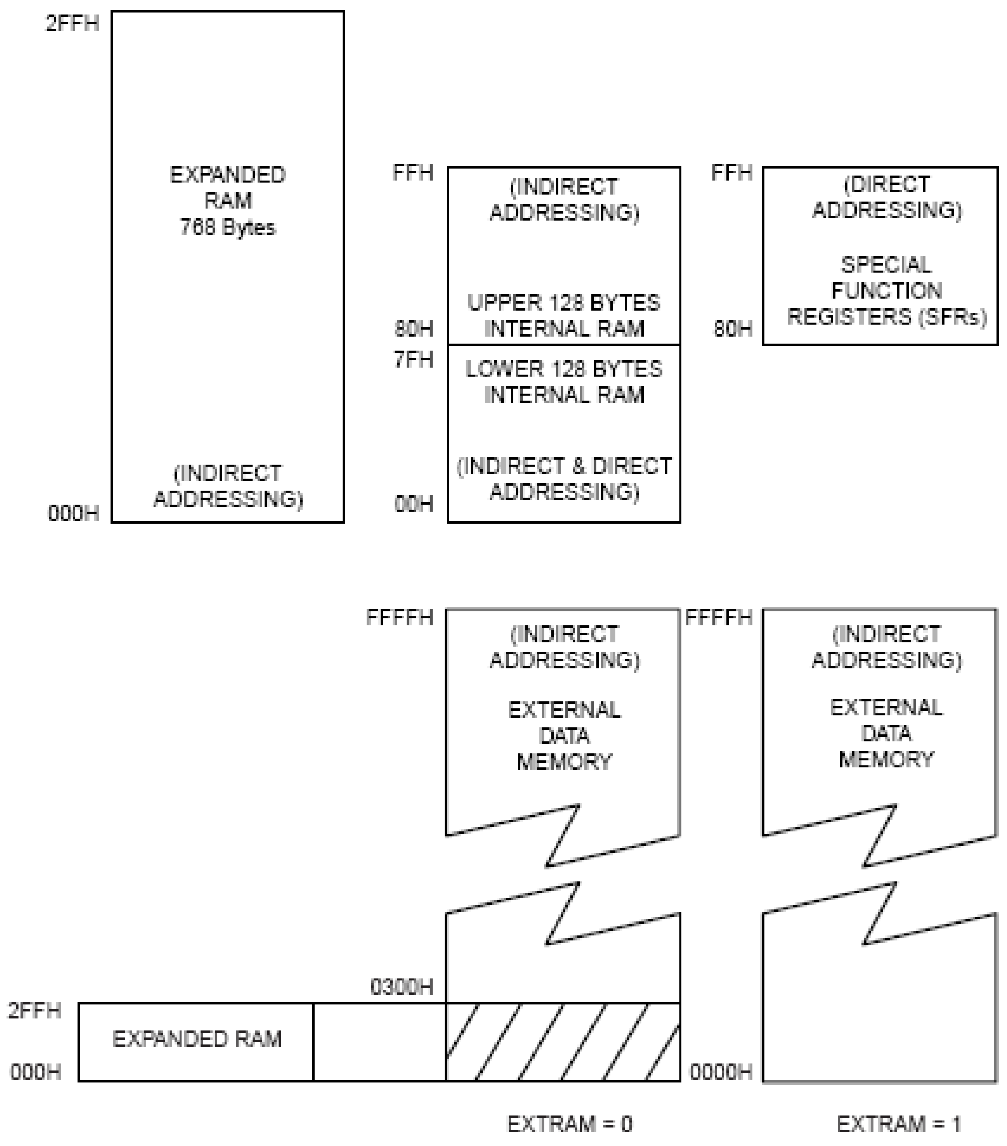


Figure4.6: Internal and external data memory structure.

Flash memory In-Application Programming

Flash organization

The P89V51RB2/RC2/RD2 program memory consists of a 16/32/64 kB block. An In-System Programming (ISP) capability, in a second 8 kB block, is provided to allow the user code to be programmed in-circuit through the serial port. There are three methods of erasing or programming of the Flash memory that may be used. First, the Flash may be programmed or erased in the end-user application by calling low-level routines through a common entry point (IAP). Second, the on-chip ISP boot loader may be invoked. This ISP boot loader will, in turn, call low-level routines through the same common entry point that can be used by the end-user application. Third, the Flash may be programmed or erased using the parallel method by using a commercially available EPROM programmer which supports this device.

In-System Programming (ISP)

In-System Programming is performed without removing the microcontroller from the system. The In-System Programming facility consists of a series of internal hardware resources coupled with internal firmware to facilitate remote programming of the P89V51RB2/RC2/RD2 through the serial port. This firmware is provided by Philips and embedded within each P89V51RB2/RC2/RD2 device. The Philips In-System Programming facility has made in-circuit programming in an embedded application possible with a minimum of additional expense in components and circuit board area.
The ISP function uses five pins (VDD, VSS, TxD, RxD, and RST). Only a small
connector needs to be available to interface your application to an external circuit in order to use this feature.

Using the In-System Programming

The ISP feature allows for a wide range of baud rates to be used in your application, independent of the oscillator frequency. It is also adaptable to a wide range of oscillator frequencies. This is accomplished by measuring the bit-time of a single bit in a received character. This information is then used to program the baud rate in terms of timer counts based on the oscillator frequency. The ISP feature requires that an initial character (an uppercase U) be sent to the P89V51RB2/RC2/RD2 to establish the baud rate. The ISP firmware provides auto-echo of received characters.

The P89V51RB2/RC2/RD2 will accept up to 32 data bytes. The 'AAAA' string represents the address of the first byte in the record. If there are zero bytes in the record, this field is often set to 0000. The 'RR' string indicates the record type. A record type of '00' is a data record. A record type of '01' indicates the end-of-file mark. In this application, additional record types will be added to indicate either commands or data for the ISP facility.

The maximum number of data bytes in a record is limited to 32 (decimal). ISP commands are summarized in Table given below.. As a record is received by the P89V51RB2/RC2/RD2, the information in the record is stored internally and a
checksum calculation is performed. The operation indicated by the record type is not performed until the entire record has been received. Should an error occur in the checksum, the P89V51RB2/RC2/RD2 will send an ‘X’ out the serial port indicating a checksum error. If the checksum calculation is found to match the checksum in the record, then the command will be executed. In most cases, successful reception of the record will be indicated by transmitting a ‘.’ character out the serial port.

Using the serial number

This device has the option of storing a 31-byte serial number along with the length of the serial number (for a total of 32 bytes) in a non-volatile memory space. When ISP mode is entered, the serial number length is evaluated to determine if the serial number is in use. If the length of the serial number is programmed to either 00H or FFH, the serial number is considered not in use. If the serial number is in use, reading, programming, or erasing of the user code memory or the serial number is blocked until the user transmits a ‘verify serial number’ record containing a serial number and length that matches the serial number and length previously stored in the device. The user can reset the serial number to all zeros and set the length to zero by sending the ‘reset serial number' record. In addition, the ‘reset serial number’ record will also erase all user code.

UARTs

The UART operates in all standard modes. Enhancements over the standard 80C51
UART include Framing Error detection, and automatic address recognition.

Mode 0

Serial data enters and exits through RxD and TxD outputs the shift clock. Only 8 bits
are transmitted or received, LSB first. The baud rate is fixed at ı ▪ 6 of the CPU clock
frequency. UART configured to operate in this mode outputs serial clock on TxD line
no matter whether it sends or receives data on RxD line.

Mode 1

10 bits are transmitted (through TxD) or received (through RxD): a start bit (logical 0),
8 data bits (LSB first), and a stop bit (logical 1). When data is received, the stop bit is stored in RB8 in Special Function Register SCON. The baud rate is variable and is determined by the Timer 1 ▪ 2 overflow rate.

Mode 2

11 bits are transmitted (through TxD) or received (through RxD): start bit (logical 0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (logical 1). When data is transmitted, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or (e.g. the parity bit (P, in the PSW) could be moved into TB8). When data is received, the 9th data bit goes into RB8 in Special Function Register SCON, while the stop bit is ignored. The baud rate is programmable to either 1 ▪ 16 or 1 ▪ 32 of the CPU clock frequency, as determined by the SMOD1 bit in PCON.

Mode 3

11 bits are transmitted (through TxD) or received (through RxD): a start bit (logical 0),
8 data bits (LSB first), a programmable 9th data bit, and a stop bit (logical 1). In fact,
Mode 3 is the same as Mode 2 in all respects except baud rate. The baud rate in Mode 3 is variable and is determined by the Timer 1 ▪ 2 overflow rate.

4.3 <u>SPECIAL FUNCTION REGISTERS</u>

The microcontroller consists of eight bit Arithmetic Logic Unit (ALU). Associated Register Array means registers like register A, register B, PSW (program status word), SP (stack pointer), and a 16-bit PC (program counter) and a 16-bit DPTR (data pointer) register, 8-bit four PORT registers, Two 8-bit timer registers TCON and TMOD, Two serial communication 8-bit registers SCON and SBUF, power mode register PCON, and two interrupt registers IP and IE.

ADDR. (HEX.)	MARK	FULL NAME
80	P0	PORT 0
81	SP	STACK POINTER
	DPTR	DATA POINTER
82	DPL	DATA LOW POINTER
83	DPH	DATA HIGH POINTER
87	PCON	POWER CONTROL
88	TCON	TIMER/COUNTER CONTROL
89	TMOD	TIMER/COUNTER MODE CONTROL
8A	TL0	TIMER/COUNTER0 LOW BYTE
8B	TL1	TIMER/COUNTER1 LOW BYTE
8C	TH0	TIMER/COUNTER0 HIGH BYTE
8D	TH1	TIMER/COUNTER1 HIGH BYTE
90	P1	PORT 1
98	SCON	SERIAL PORT CONTROL
99	SBUF	SERIAL DATA PORT
A0	P2	PORT 2
A8	IE	INTERRUPT ENABLE
B0	P3	PORT 3
B8	IP	INTERRUPT PRIORITY CONTROL
D0	PSW	PROGRAM STATUS WORD
E0	ACC(A)	ACCUMULATOR
F0	B	B REGISTER

Table 4.1: Special Function Registers

ACCUMULATOR

The Accumulator, as its name suggests, is used as a general register to accumulate the results of a large number of instructions. It can hold an 8-bit (1-byte) value and is the most versatile register, the microcontroller has due to the shear number of instructions that make use of the accumulator. Accumulator holds a source of operand and stores the result of the arithmetic operations such as addition, subtraction, multiplication and division. The accumulator has several exclusive functions such as rotate, parity computation; testing for 0, sign acceptor etc. and so on.

DPL/DPH (Data Pointer Low/High, Addresses 82h/83h)

The SFRs DPL and DPH work together to represent a 16-bit value called the *Data Pointer*. The data pointer is used in operations regarding external RAM and some instructions involving code memory. Since it is an unsigned two-byte integer value, it can represent values from 0000h to FFFFh (0 through 65,535 decimal).

SCON (Serial Control, Addresses 98h, Bit-Addressable)

The Serial Control SFR is used to configure the behavior of the 8051's on-board serial port. This SFR controls the baud rate of the serial port, whether the serial port is activated to receive data, and also contains flags that are set when a byte is successfully sent or received.
Bit addressable.

7	6	5	4	3	2	1	0
SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI

REN	set or cleared by software to enable or disable reception.
TB 8	not widely used.
RB 8	not widely used.
TI	transmits interrupt flag. Set by hardware at the beginning of the stop bit in mode 1. It must be cleared by software.
RI	received interrupts flag. Set by hardware halfway through the stop bit time in mode 1. It must be cleared by software.

SM0	SM1	Serial mode 0
0	0	Synchronous mode
0	1	8-bit data, 1 start bit, 1 stop bit, variable baud rate
1	0	9- bit data, 1 start bit, 1 stop bit, fixed baud rate
1	1	9- bit data, 1 start bit, 1 stop bit,

		variable baud rate

SBUF (Serial Control, Addresses 99h)

The Serial Buffer SFR is used to send and receive data via the on-board serial port. Any value written to SBUF will be sent out the serial port's TXD pin. Likewise, any value which the 8051 receives via the serial port's RXD pin will be delivered to the user program via SBUF. In other words, SBUF serves as the output port when written to and as an input port when read from.

TCON (Timer Control, Addresses 88h, Bit-Addressable)

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 the TCON SFR. These bits are used to configure the way in which the external interrupts are activated and also contain the external interrupt flags which are set when an external interrupt has occurred.

TMOD (Timer Mode, Addresses 89h)

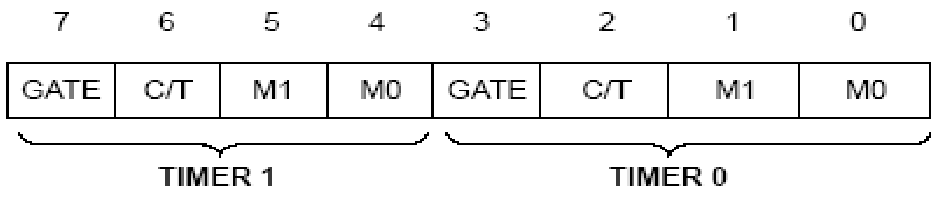
The Timer Mode SFR is used to configure the mode of operation of each of the two timers. Using this SFR your program may configure each timer to be a 16-bit timer, an 8-bit auto reload timer, a 13-bit timer, or two separate timers. Additionally, you may configure the timers to only count when an external pin is activated or to count "events" that are indicated on an external pin.

TIMER 0 AND TIMER 1

The "timer” or "counter "function is selected by control bits C/T in the special function register TMOD. These two timer/counters have for operating modes, which are selected by bit-pairs (M1/M0) in TMOD. Modes 0, 1, and 2 are the same for both timers/counters. Mode 3 is different.

TL0/TH0 (Timer 0 Low/High, Addresses 8Ah/8Ch)

These two SFRs, 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.



GATE: When set, start and stop of timer by hardware
When reset, start and stop of timer by software

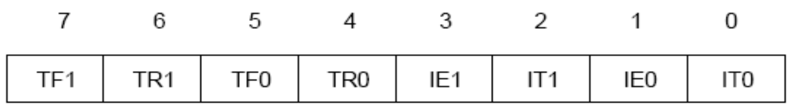
C/T : Cleared for timer operation
Set for counter operation

M1	M0	MODE	OPERATING MODE
0	0	0	13-bit timer mode
0	1	1	16-bit timer mode
1	0	2	8-bit auto reload mode
1	1	3	Split timer mode

TL1/TH1 (Timer 1 Low/High, Addresses 8Bh/8Dh)

These two SFRs, 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.
Address =88H.

Bit addressable.



7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF: Timer overflow flag: Set by hardware when the timer/counter overflows. It is cleared by hardware, as the processor vectors to the interrupt service routine.

TR: timer run control bit: Set or cleared by software to turn timer or counter on/off.

IE: set by CPU when the external interrupt edge (H-to-L transition) is detected. It is cleared by CPU when the interrupt is processed.

IT: set/cleared by software to specify falling edge/low-level triggered external interrupt.

4.4 <u>INPUT – OUTPUT (I/O) PORTS</u>

Every MCU from 89C51 families has 4 I/O ports of 8 bits each. This provides the user with 32 I/O lines for connecting MCU to the environs. Unlike the case with other controllers, there is no specific SFR register for designating pins as input or output. Instead, the port itself is in charge: 0=output, 1=input. If particular pin on the case is needed as output, the appropriate bit of I/O port should be cleared. This will generate 0V on the specified controller pin. Similarly, if particular pin on the case is needed as input, the appropriate bit of I/O port should be set. This will designate the pin as input, generating +5V as a side effect (as with every TTL input).

PORT 0

Port 0 has two-fold role: if external memory is used, it contains the lower address byte (addresses A0-A7); otherwise all bits of the port are either input or output. Another feature of this port comes to play when it has been designated as output. Unlike other ports, Port 0 lacks the "pull up" resistor (resistor with +5V on one end). This seemingly insignificant change has the following consequences: When designated as input, pin of Port 0 acts as high impedance offering the infinite input resistance with no "inner" voltage. When designated as output, pin acts as "open drain". Clearing a port bit grounds the appropriate pin on the case (0V). Setting a port bit makes the pin act as high impedance. Therefore, to get positive logic (5V) at output, external "pull up" resistor needs to be added for connecting the pin to the positive pole. Therefore, to get one (5V) on the output, external "pull up" resistor needs to be added for connecting the pin to the positive pole.

PORT 1

This is "true" I/O port, devoid of dual function characteristic for Port 0. Having the "pull up" resistor, Port 1 is fully compatible with TTL circuits. This is input/output port 1. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 1 is pin P1.0, bit 7 is pin P1.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

PORT 2

When external memory is used, this port contains the higher address byte (addresses A8–A15), similar to Port 0. Otherwise, it can be used as universal I/O port.

This is input/output port 2. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 2 is pin P2.0, bit 7 is pin P2.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

PORT 3

Beside its role as universal I/O port, each pin of Port 3 has an alternate function. In order to use one of these functions, the pin in question has to be designated as input, i.e. the appropriate bit of register P3 needs to be set. From a hardware standpoint, Port 3 is similar to Port 0.

5. SERIAL COMMUNICATION

In order to connect micro controller or a PC to any modem a serial port is used. Serial, is a very common protocol for device communication that is standard on almost every PC .Most computers including RS-232 based serial ports. Serial is also a common communication protocol that is used by many devices for instrumentation.

In serial communication, the data is sent one bit at a time in contrast parallel communication, in which the data is sent a byte or more at time. Serial communication uses a single data line where as the parallel communication uses 8 bit data line , this makes serial communication not only inexpensive but also makes it possible for two computers located in two different cities to communicate over the telephone.

Serial data communication uses two methods, asynchronous and synchronous. the synchronous method transfers a block of data at a time while the asynchronous transfers a single byte at a time. The 8051 have an in built UART (Universal Asynchronous Receiver-Transmitter).

Typically, serial is used to transmit ASCII data. Communication is completed using 3 transmission lines: (1) Transmitter, (2) Receiver and (3) Ground. Since serial is asynchronous, the port is able to transmit data on one line while receiving data on another. Other lines are available for handshaking, but are not required. The important characteristics are Data Transfer Rate, Start and Stop bits, Data bits and Parity bits. For two ports to communicate, these parameters must match.

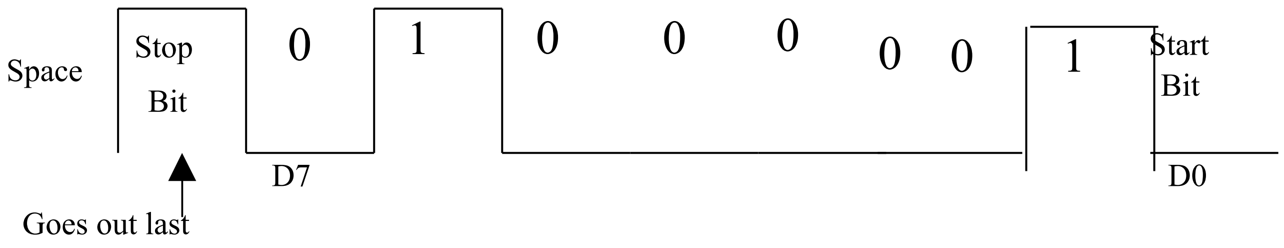


Figure 5.1: Serial communication transmission method

Data Transfer Rate

The rate of data transfer in serial data communication is stated in bps (bits per second) or baud rate. However the baud rate and bps are not necessarily equal. Baud rate is defined as the number of signal changes per second. In modems, there are occasions when a single change of signal transfers several bits of data .As far as conductor wire is considered bps and baud rate is the same.

Data Framing

Asynchronous serial data communication is used for character oriented transmissions, each character is placed in between start and stop bits. This is called framing .In data framing for asynchronous communications , the data ,such as ASCII characters ,are packed in between a start bit and a stop bit .The start bit is always one bit but the stop bit can be one or more bits .The start bit is always 0(low) whereas stop bit is 1(high).Since the data is clocked across the lines and each device has its own clock , it is possible for the two devices to come out slightly out of synchronous .Therefore ,the stop bits not only indicate the end of transmission but also give the computers some room for error in clock speeds .The more the stop bits the greater the lenience in synchronizing the different clocks, but slower the data transmission rate.

Parity Bits

In order to maintain data integrity, parity bit of the character byte is included in the data frame .The parity bit is odd or even .In the case of an odd parity bit the number of data bits, including the parity bits has an odd number of 1's.Similarly, in an even parity bit system the total number of bits, including the parity bits has an even number of 1's.UART chips allow programming of the parity bit for odd-, even- and no- parity options.

5.1 RS232 STANDARD

RS denotes “Recommended Standard” and refers to official standards of Electronics Industries Association. RS-232 is the most known serial port used in transmitting the data in communication and interface. Even though serial port is harder to program than the parallel port, this is the most effective method in which the data transmission requires less that yields to the less cost. Serial RS-232 communication works with voltages (-15V ... -3V for *high* [sic]) and +3V ... +15V for *low* [sic]) which are not compatible with normal computer logic voltages.

The maximum RS-232 signal levels are far too high for computer logic electronics, and the negative RS-232 voltage for *high* cant be applicable at all by computer logic. Therefore, to receive serial data from an RS-232 interface the voltage has to be reduced, and the *low* and *high* voltage level inverted. In the other direction (sending data from some logic over RS-232) the low logic voltage has to be bumped up”, and a negative voltage has to be generated, too.

Independent channels are established for two-way (full-duplex) communications. The RS232 signals are represented by voltage levels with respect to a system common (power *I* logic ground). The “idle” state (MARK) has the signal level negative with respect to common, and the “active” state (SPACE) has the signal level positive with respect to common. RS232 has numerous handshaking lines (primarily used with modems), and also specifies a communications protocol.

The RS-232 interface presupposes a common ground between the DTE and DCE. This is a reasonable assumption when a short cable connects the DTE to the DCE, but with longer lines and connections between devices that may be on different electrical busses with different grounds, this may not be true.

RS232 data is bi-polar.... +3 to +12 volts indicate an “ON or 0-state (SPACE) condition’s while A -3 to -12 volts indicates an “OFF” 1-state (MARK) condition.... Modern computer equipment ignores the negative level and accepts a zero voltage level as the “OFF” state. In fact, the “ON” state may be achieved with lesser positive potential.

The output signal level usually swings between +12V and -12V. The “dead area” between +3v and -3v is designed to absorb line noise. In the various RS-232-like definitions this dead area may vary.

This can cause problems when using pin powered widgets - line drivers, converters, modems etc. These types of units need enough voltage & current to power them self's up. Typical URART (the RS-232 I/O chip) allows up to 50ma per output pin - so if the device needs 70ma to run we would need to use at least 2 pins for power. The number of output lines, the type of interface driver IC, and the state of the output lines are important considerations.

The types of driver ICs used in serial ports can be divided into three general categories:

- Drivers which require plus (+) and minus (-) voltage power supplies such as the 1488 series of interface integrated circuits. (Most desktop and tower PCs use this type of driver.)

- Low power drivers which require one +5 volt power supply. This type of driver has an internal charge pump for voltage conversion. (Many industrial microprocessor controls use this type of driver.)

- Low voltage (3.3 v) and low power drivers which meet the EIA-562 Standard. (Used on notebooks and laptops.)

Data is transmitted and received on pins 2 and 3 respectively. Data Set Ready (DSR) is an indication from the Data Set (i.e., the modem or DSU/CSU) that it is on. Similarly, DTR indicates to the Data Set that the DTE is on. Data Carrier Detect (DCD) indicates that a good carrier is being received from the remote modem.

Pins **4** RTS (Request to Send - from the transmitting computer) and **5** CTS (Clear to Send - from the Data set) are used to control. In most Asynchronous situations, RTS and CTS are constantly on throughout the communication session. However where the DTE is connected to a multipoint line. RTS is used to turn carrier on the modem on and *off.* On a multipoint line, it's imperative that only one station is transmitting at a time (because they share the return phone pair). When a station wants to transmit, it raises RTS. The modem turns on carrier, typically waits a few milliseconds for carrier to stabilize, and then raises

CTS. The DTE transmits when it sees CTS up. When the station has finished its transmission, it drops RTS and the modem drops CTS and carrier together.

Clock signals (pins 15, 17, & 24) are only used for synchro/.
nous communications. The modem or DSU extracts the clock from the data stream and provides a steady clock signal to the DTE. The transmit and receive clock signals do not have to be the same, or even at the same baud rate.

To allow compatibility among data communication equipment made by various Manufacturers, an interfacing standard called RS232 was set by the Electronics and Industries Association in 1960.Today, RS232 is the most widely used serial I/O interfacing standard. RS232 standard is not TTL compatible; therefore it requires a line driver such as MAX232chip to convert RS232 voltage levels to TTL levels and vice versa. One advantage of the MAX 232 chip is that it uses +5V power source that has same source voltage as that of 8051.

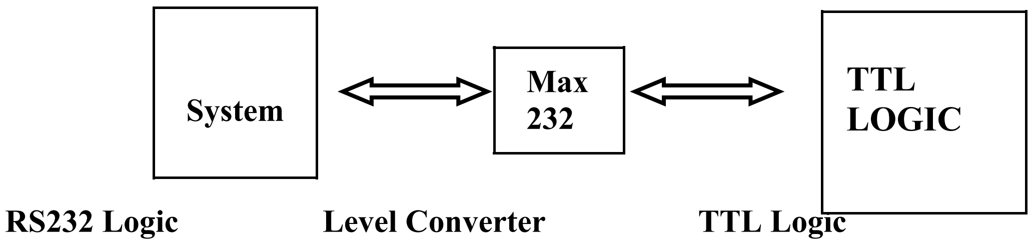


Figure 5.2: Block Diagram of RS232

Figure 5.3: RS232 cable

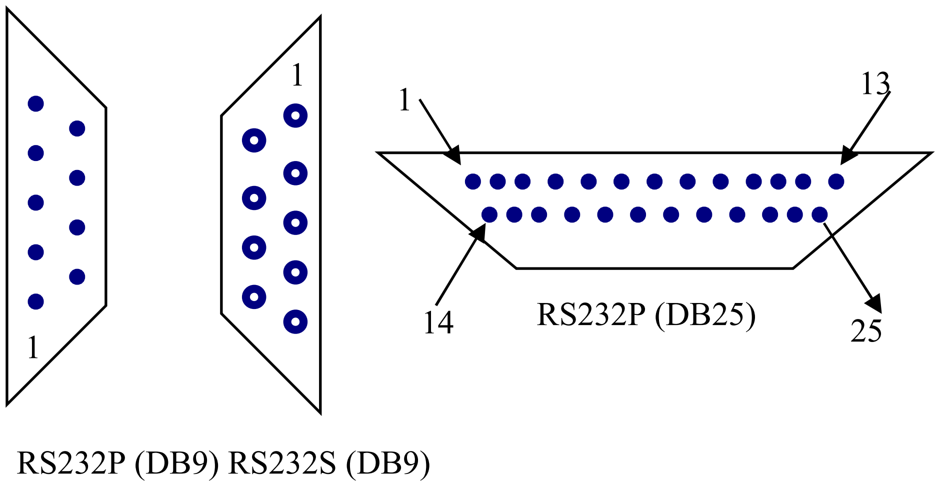


Figure: 5.2 Pin Diagram of RS232 Connector

RS232 PINS

Pin	Description
1	Protective ground
2	Transmitted data (TxD)
3	Received data (RxD)
4	Request to send (RTS)
5	Clear to send (CTS)
6	Data set ready (DSR)
7	Signal ground (GND)
8	Data carrier detect (DCD)
9/10	Reserved for data setting
11	Unassigned
12	Secondary data carrier
13	Secondary clear send
14	Secondary transmitted data
15	Transmit signal element timing
16	Secondary received data
17	Receive signal element timing
18	Unassigned
19	Secondary request to send
20	Data terminal ready (DTR)
21	Signal quality detector

22	Ring indicator
23	Data signal rate select
24	Transmit signal element timing
25	Unassigned

MAX 232

The RS 232 is not compatible with micro controllers, so a line driver converts the RS 232's signals to TTL voltage levels. It is a 16 pin DIP package.

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels.

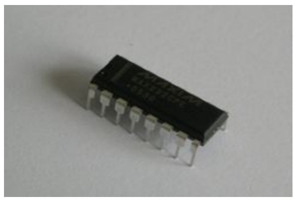
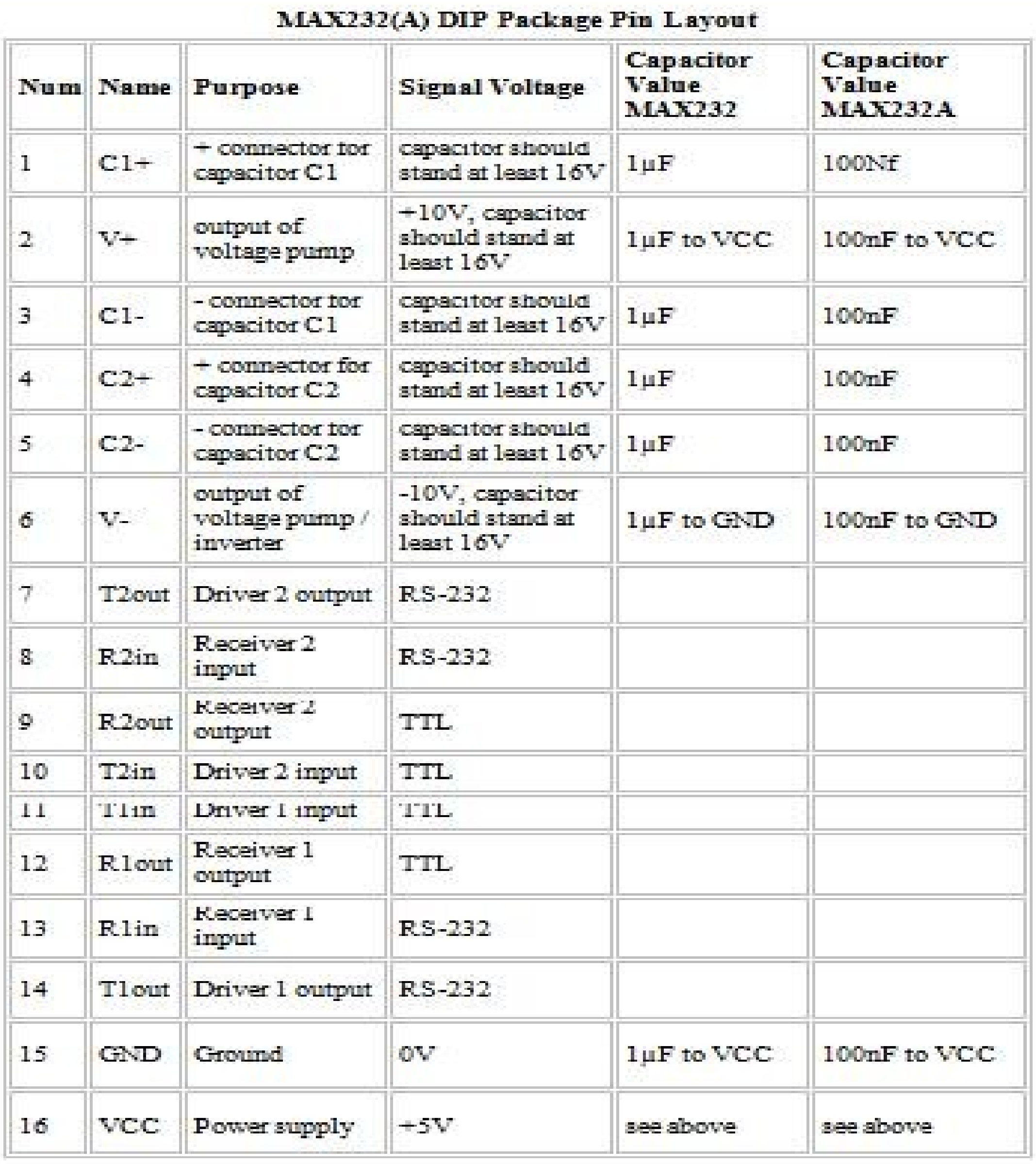


Figure 5.4: MAX 232 IC



MAX232(A) DIP Package Pin Layout

Num	Name	Purpose	Signal Voltage	Capacitor Value MAX232	Capacitor Value MAX232A
1	C1+	+ connector for capacitor C1	capacitor should stand at least 16V	1µF	100Nf
2	V+	output of voltage pump	+10V, capacitor should stand at least 16V	1µF to VCC	100nF to VCC
3	C1-	- connector for capacitor C1	capacitor should stand at least 16V	1µF	100nF
4	C2+	+ connector for capacitor C2	capacitor should stand at least 16V	1µF	100nF
5	C2-	- connector for capacitor C2	capacitor should stand at least 16V	1µF	100nF
6	V-	output of voltage pump / inverter	-10V, capacitor should stand at least 16V	1µF to GND	100nF to GND
7	T2out	Driver 2 output	RS-232		
8	R2in	Receiver 2 input	RS-232		
9	R2out	Receiver 2 output	TTL		
10	T2in	Driver 2 input	TTL		
11	T1in	Driver 1 input	TTL		
12	R1out	Receiver 1 output	TTL		
13	R1in	Receiver 1 input	RS-232		
14	T1out	Driver 1 output	RS-232		
15	GND	Ground	0V	1µF to VCC	100nF to VCC
16	VCC	Power supply	+5V	see above	see above

Table 5.1: Pin Description of MAX232

Registers Used For Communication

SBUF Register

SBUF is an 8-bit register used solely for serial communication in the 8051. For byte of data to be transferred via TxD line, it must be placed in *SBUF* register. *SBUF* also holds the byte of data when it is received by the 8051's RxD line.

The moment a byte is written into *SBUF*, it is framed with the start and stop bits and transferred serially via TxD line. Similarly when bits r received serially via RxD, the 8051 defames it by eliminating a byte out of the received, and then placing it in the *SBUF*.

6.SENSING LOGIC

In this section we will discuss how the sensing action takes place in the project. In this we use the simple principle of conduction between two wires through a medium. The logic consists of two wires one with the 5V supply and the other one was the ground. When there is water in the field, a medium is being established between the wires and the conduction will take place between the wires and the voltage at the microcontroller will go low indicating the wet condition of the field. In this wet condition the mpotor will be in off position.

If there is no water in the field, the re is no conduction between the wires and the voltage at the microcontroller will go high indicating the dry condition of the field and the motor will be made on position.

We can not only know the status of the field but also control the motor action from our home itself by using the GSM technology. The following are the commands or messages to be passed to the GSM modem from our user moblie for the above mentioned operations.

- To know the status of the field, **GIFMOD**
- To make the motor off postion, we use **MTR OF**
- To make the motor on.. **MTR ON**

7.GSM MODEM(SIM300S)

INTRODUCTION

Scope of the document

This document presents the AT Command Set for SIMCOM cellular engine SIM300S

Conventions and abbreviations

In this document, the GSM engines are referred to as following term:

1) ME (Mobile Equipment);
2) MS (Mobile Station);
3) TA (Terminal Adapter);
4) DCE (Data Communication Equipment) or facsimile DCE(FAX modem, FAX board);

In application, the controlling device controls the GSM engine by sending AT Command via its serial interface. The controlling device at the other end of the serial line is referred to as following term:

1) TE (Terminal Equipment);
2) DTE (Data Terminal Equipment) or plainly "the application" which is running on an embedded system;

AT Command syntax:

The "AT" or "at" prefix must be set at the beginning of each command line. To terminate a command line enter <CR>.Commands are usually followed by a response that includes."<CR><LF><response><CR><LF>" Throughout this document, only the responses are presented, <CR><LF> are omitted intentionally.
The AT command set implemented by SIM100S is a combination of GSM07.05, GSM07.07 and ITU-T recommendation V.25ter and the AT commands developed by SIMCOM.

Note: Only enter AT command through serial port after SIM100S is power on and Unsolicited Result Code "RDY" is received from serial port. And if Unsolicited Result Code"SCKS: 0" returned it indicates SIM card isn't present.

All these AT commands can be split into three categories syntactically: "**basic**", "**S parameter**", and "**extended**". These are as follows:

Basic syntax

These AT commands have the format of "**AT<*x*><*n*>**", or "**AT&<x><n>**", where "**<*x*>**"is the command, and "**<*n*>**"is/are the argument(s) for that command. An example of this is "**ATE<*n*>**", which tells the DCE whether received characters should be echoed back to the DTE according to the value of "**<*n*>**". "**<*n*>**" is optional and a default will be used if missing.

S parameter syntax

These AT commands have the format of "**ATS<*n*>=<*m*>**", where "**<*n*>**" is the index of the **S** register to set, and "**<*m*>**"is the value to assign to it. "**<*m*>**" is optional; if it is missing, then a default value is assigned.

Extended Syntax

These commands can operate in several modes, as following table:

Table 1: Types of AT commands and responses

Test command	AT+<*x*>=?	The mobile equipment returns the list of parameters and value ranges set with the corresponding Write command or by internal processes.
Read command	AT+<*x*>?	This command returns the currently set value of the parameter or parameters.
Set command	AT+<x>=<…>	This command sets the user-definable parameter values.
Execution command	AT+<x>	The execution command reads non-variable parameters affected by internal processes in the GSM engine

Combining AT commands on the same command line

You can enter several AT commands on the same line. In this case, you do not need to type the "**AT**" or "**at**" prefix before every command. Instead, you only need type "**AT**" or "**or**" at the beginning of the command line. Please note to use a semicolon as command delimiter.

The command line buffer can accept a maximum of 256 characters. If the characters entered exceeded this number then none of the command will executed and TA will returns "**ERROR**".

Entering successive AT commands on separate lines

When you need to enter a series of AT commands on separate lines, please note that you need to wait the final response (for example OK, CME error, CMS error) of last AT command you entered before you enter the next AT command.

Supported character sets

The SIM300S AT command interface defaults to the **GSM** character set. The SIM300S supports the following character sets:

- GSM format
- UCS2

The character set can be set and interrogated using the "**AT+CSCS**" command (GSM 07.07). The character set is defined in GSM specification 07.05.

The character set affects transmission and reception of SMS and SMS Cell Broadcast messages, the entry and display of phone book entries text field and SIM Application Toolkit alpha strings.

Flow control

Flow control is very important for correct communication between the GSM engine and DTE. For in the case such as a data or fax call, the sending device is transferring data faster than the receiving side is ready to accept. When the receiving buffer reaches its capacity, the receiving device should be capable to cause the sending device to pause until it catches up.

There are basically two approaches to achieve data flow control: software flow control and hardware flow control. SIM100S support both two kinds of flow control.

In Multiplex mode, it is recommended to use the hardware flow control.

Software flow control (XON/XOFF flow control)

Software flow control sends different characters to stop (XOFF, decimal 19)

and resume (XON, decimal 17) data flow. It is quite useful in some applications that only use three wires on the serial interface.
The default flow control approach of SIM100S is hardware flow control (RTS/CTS flow control), to enable software flow control in the DTE interface and within GSM engine, type the following AT command:

AT+IFC=1,1

This setting is stored volatile, for use after restart, **AT+IFC=1,1** should be stored to the user profile with **AT&W**.

Ensure that any communications software package (e.g. ProComm Plus, HyperTerminal or WinFax Pro) uses software flow control.

Software Flow control should not be used for data calls where binary data will be transmitted or received (e.g. TCP/IP) as the DTE interface may interpret binary data as flow control characters.

Hardware flow control (RTS/CTS flow control)

Hardware flow control achieves the data flow control by controlling the RTS/CTS line. When the data transfer should be suspended, the CTS line is set inactive until the transfer from the receiving buffer has completed. When the receiving buffer is ok to receive more data, CTS goes active once again.

To achieve hardware flow control, ensure that the RTS/CTS lines are present on your application platform.

AT Commands Additional To Sim300S II

This section lists the AT commands and responses that are additional to SIM100S. For each of the commands listed in the table below a more detailed description is provided in the following section. Some commands are SIMCOM proprietary as none currently exist in the GSM specifications for certain functions. If an AT command is SIMCOM proprietary it is indicated in the table below.

7.1 Overview

Command	Description	SIMCOM Proprietary
+CLTS	GET LOCAL TIMESTAMP (TIME/DATE COMES FORMNITZ)	Y
+CEXTHS	EXTERNAL HEADSET JACK CONTROL	Y
+CEXTBUT	HEADSET BUTTON STATUS REPORTING	Y
+CMUT	MUTE CONTROL	
+CLVL	LOUDSPEAKER VOLUME LEVEL	
+CBC	BATTERY CHARGE	
+CSSN	SUPPLEMENTARY SERVICES NOTIFICATION	
+CSIM	GENERIC SIM ACCESS	
+CMUX	GSM 07.10 MULTIPLEXER CONTROL	
+CPOL	PREFERRED OPERATOR LIST	
+COPN	READ OPERATOR NAMES	
+CNUM	READ SUBSCRIBER NUMBER	
+CSMINS	SIM INSERTED STATUS REPORTING	Y
+CCLK	CLOCK	
+CALM	ALERT SOUND MODE(RINGER TYPE)	
+CRSL	RINGER SOUND LEVEL	
+CPUC	PRICE PER UNIT AND CURRENCY TABLE	
+CCWE	CALL METER MAXIMUM EVENT	
+CLDTMF	LOCAL DTMF TONE GENERATION	Y
+CDRIND	CS Call/GPRS PDP CONTEXT TERMINATION INDICATION	Y
+CSPN	GET SERVICE PROVIDER NAME FROM SIM	Y
+CCVM	GET AND SET THE VOICE MAIL NUMBER ON THE SIM	Y
+CGURC	GENERIC UNSOLICITED RESULT CODES	Y
+CHFA	SWAP THE AUDIO CHANNELS	Y
+CPCS	CHOOSE THE FREQUENCY BAND	Y
+CDFL	DELETE THE FPLMN LIST	Y

+RADC		Y
+SPIC	TIMES REMAIN TO INPUT SIM PIN/PUK	Y
+CHUP	DISCONNECT ALL CALLS(NOT INCLUDE GPRS CALLS)	Y
+HUPG	DISCONNECT GPRS CALLS	Y
+CBAND	SELECT BAND MODE	Y
+CSNS	SELECT DATA TRANSFERS MODE	Y
+UART	CONFIGURE UART	Y
+CDTMT	SET DTMF TIME	Y
+ECPBS	SELECT DIAL PHONE TYPE	Y
+CCID	SHOW ICCID	Y
+CPOWD	POWER DOWN	Y
+CALARM	SET THE REAL-TIME CLOCK OF THE ME	Y
+CDSCB	RESET CELLBROADCAST	Y
+SMURC	CONFIG SMS READY INDICATION	Y
+CMIC	CHANGE THE MICROPHONE GAIN LEVEL	Y
+SIDET	CHANGE THE SIDE TONE GAIN LEVEL	Y
+ECHO	ECHO CANCELLATION CONTRLO	Y

Supported unsolicited result codes

This section lists the unsolicited result codes supported in the Data Services software. The AT commands specific to SIM100S III implementation which are defined in this document include details of the relevant values supported.

Unsolicited Result Code	Description	SIM100S III Specific?	SIMCOM Proprietary
+CME ERROR	Error report	N	
+CR	Service reporting control	N	
+DR	Data compression control	N	
+ILRR	Determines whether the used local TE-TA data rate is informed using intermediate result code +ILRR: <rate> before going online data state after call answering or originating	N	
+CMTI	New SMS indication	N	
+CMT	New SMS indication including message content	N	
+CBM	New CBS indication including message content	N	
+CDS	SMS-STATUS-REPORT indication	N	

+CMS ERROR	SMS error report	N	
+CCWA	Call waiting indication	N	
+CLIP	Calling line identification presentation	N	
+COLP	Connected Line Identification Presentation	N	
+CREG	Network registration	N	
+CRING	Extended format: incoming call indication	N	
+CSSI	intermediate result indication / Supplementary service notifications	Y	
+CSSU	unsolicited result indication / Supplementary service notifications	Y	
+CUSD	Unstructured supplementary service data	Y	
+CEXTHS	External headset jack state reporting	Y	Y
+CEXTBUT	External headset button state reporting	Y	Y
+CGEV	GPRS event reporting information	Y	
+CSMINS	SIM insertion and removal reporting	Y	Y
+CCWV	Call Meter Maximum Event	Y	
+CDRIND	CSD call or GPRS PDP context termination reporting	Y	Y
+CGURC	Generic unsolicited result code	Y	Y

Summary of CME ERROR Codes:

Final result code +CME ERROR: <err> indicates an error related to mobile equipment or network. The operation is similar to ERROR result code. None of the following commands in the same command line is executed. Neither ERROR nor OK result code shall be returned.<err> values used by common messaging commands:

Code of <err>	Meaning
0	phone failure
1	no connection to phone
2	phone-adaptor link reserved
3	operation not allowed
4	operation not supported
5	PH-SIM PIN required
6	PH-FSIM PIN required
7	PH-FSIM PUK required
10	SIM not inserted
11	SIM PIN required
12	SIM PUK required
13	SIM failure
14	SIM busy
15	SIM wrong
16	incorrect password
17	SIM PIN2 required
18	SIM PUK2 required
20	memory full
21	invalid index
22	not found
23	memory failure
24	text string too long
25	invalid characters in text string
26	dial string too long
27	invalid characters in dial string
30	no network service
31	network timeout
32	network not allowed - emergency calls only
40	network personalization PIN required
41	network personalization PUK required
42	network subset personalization PIN required
43	network subset personalization PUK required
44	service provider personalization PIN required
45	service provider personalization PUK required
46	corporate personalization PIN required

47	corporate personalization PUK required
48	SIM invalid - network reject
100	unknown
103	illegal MS
106	illegal ME
107	gprs services not allowed
111	plmn not allowed
112	location area not allowed
113	roaming not allowed in this location area
114	service type not yet available
115	SMS service pref outof range
116	mode value not in range
117	buffer value not in range
118	buffer value not present
119	buffer value not supported
120	mobile not ready
121	invalid activation state (0-1)
122	invalid cid value
123	profile (cid) not defined
124	processing of multiple cids not supported
125	pdp type not supported
126	GPRS - invalid dial string length
127	GPRS - QOS validation fail
128	GPRS - invalid character in address string
129	GPRS - address element out of range
130	GPRS - invalid address length
132	service option not supported
133	requested service option not subscribed
134	service option temporarily out of order
135	GPRS - LLC or SNDCP failure
136	GPRS - insufficient resources
137	GPRS - missing or unknown APN
138	GPRS - unknown PDP address or type
139	GPRS - user authorization failed
140	GPRS - activation rejected by GGSN
141	GPRS - unspecified activation rejection
142	GPRS - NSAPI already used
143	GPRS - regular deactivation
144	GPRS - QOS not accepted
145	GPRS - network failure
146	GPRS - reactivation required
147	GPRS - feature not supported
148	unspecified gprs error
149	PDP authentication failure
150	invalid mobile class

160	invalid input value
161	volume level out of range
162	invalid command length
163	+CSCS type not supported
164	+CSCS type not found
165	must include <format> with <oper>
166	incorrect <oper> format
167	<oper> length too long
168	SIM full
169	unable to change PLMN list
170	network operator not recognized
171	invalid input string
172	unsupported value or mode
173	operation failed
174	audio manager not ready
175	audio format cannot be configured
202	multiplexer already active
203	sim toolkit menu has not been configured
204	GPRS - semantic error in TFT operation
205	GPRS - syntactical error in TFT operation
206	GPRS - unknown PDP context
207	GPRS - PDP context w/o TFT already activated
208	GPRS - semantic errors in packet filter
209	GPRS - syntactical errors in packet filter
210	GPRS - invalid TI value
211	GPRS - semantically incorrect message
212	GPRS - invalid MAND information
213	GPRS - unknown message from network
214	GPRS - message type incompatible with state
215	GPRS - unknown IE from network
216	GPRS - conditional IE error
217	GPRS - message incompatible with state
218	GPRS - unspecified protocol error
219	GPRS - duplicate TI received
220	GPRS - LLC error
221	GPRS - peer refuses our MRU
222	GPRS - peer refuses our ACCM
223	GPRS - peer refuses our IP address
224	GPRS - peer rerequested CHAP
225	GPRS - LCP negotiation timeout
226	GPRS - IPCP negotiation timeout

227	GPRS - PAP close
228	GPRS - CHAP close
229	GPRS - NCP close
230	GPRS - normal termination
231	GPRS - bad code or protocol rejection
232	GPRS - no echo reply
233	GPRS - can't modify address
234	GPRS - can't modify address
235	GPRS - too many RXJs
236	GPRS - combined services not allowed
237	GPRS - MS identity not in network
238	GPRS - implicitly detached
239	GPRS - MSC temporarily not reachable
240	GPRS - no PDP context activated
241	GPRS - lower layer failure
242	GPRS - no free NSAPIs
243	GPRS - service not available

Table7.2: CME code errors

8. DC MOTOR

DC motors are fairly simple to understand. They are also simple to make and only require a battery or dc supply to make them run.

The brushed DC motor will generate torque directly from DC power applied to the motor leads. Brushed DC motors require a significant amount of maintenance to work properly. This involves replacing the brushes and springs which carry the electric current as well as cleaning or replacing the commutator.

Many of the limitations of the classic commutator DC motor are due to the need for brushes to press against the commutator. This creates friction. At higher speeds, brushes have increasing difficulty in maintaining contact. Brushes may bounce off the irregularities in the commutator surface, creating sparks. This limits the maximum speed of the machine. The current density per unit area of the brushes limits the output of the motor. Brushes eventually wear out and require replacement, and the commutator itself is subject to wear and maintenance. The commutator assembly on a large machine is a costly element, requiring precision assembly of many parts.

Brushless DC motor

In this motor, the mechanical "rotating switch" or commutator/ brush gear assembly is replaced by an external electronic switch synchronized to the rotor's position. Brushless motors are typically 85-90% efficient, whereas DC motors with brush gear are typically 75-80% efficient.

Figure 7.1: Brushless Dc Motor

Synchronous types, like the brushless DC motor and the stepper motor will lock up on DC power, and require external commutation to generate torque. Advantages of the brushless motor include long life span, little or no maintenance, and good efficiency. Disadvantages include high cost and more complicated motor speed controllers. Brushless motors use a rotating permanent magnet and with stationary electrical magnets on the motor housing. This eliminates the complication of getting power to a rotating system.

It has a permanent magnet external rotor, three phases of driving coils, one or more Hall effect sensors to sense the position of the rotor, and the associated drive electronics. The coils are activated, one phase after the other, by the drive electronics as cued by the signals from the Hall effect sensors. In effect, they act as three-phase synchronous motors containing their own variable-frequency drive electronics. A specialized class of brushless DC motor controllers utilize EMF feedback through the main phase connections instead of Hall effect sensors to determine position and velocity.

In a BLDC motor, the electromagnets do not move, instead, the permanent magnets rotate and the armature remains static. The brush-system/commutator assembly is replaced by an electronic controller. The controller performs the same power distribution found in a brushed DC motor, but using a solid-state circuit rather than a commutator/brush system.

BLDC motors are often more efficient at converting electricity into mechanical power than brushed DC motors. This improvement is largely due to the absence of electrical and friction losses due to brushes. The enhanced efficiency is greatest in the no-load and low-load region of the motor's performance curve.

BLDC motors offer several advantages over brushed DC motors, including higher efficiency and reliability, reduced noise, longer lifetime (no brush erosion), elimination of ionizing sparks from the commutator, and overall reduction of electromagnetic interference (EMI). With no windings on the rotor, they are not subjected to centrifugal forces, and because the electromagnets are attached to the casing, the electromagnets can be cooled by conduction, requiring no airflow inside the motor for cooling. This in turn means that the motor's internals can be entirely enclosed and protected from dirt or other foreign matter. The maximum power that can be applied to a BLDC motor is exceptionally high, limited almost exclusively by heat, which can damage the magnets. BLDC's main disadvantage is higher cost, which arises from two issues.

Brushless DC motors are commonly used where precise speed control is necessary, computer disk drives or in video cassette recorders the spindles within CD, CD-ROM (etc.) drives, and mechanisms within office products such as fans, laser printers and photocopiers. They have several advantages over conventional motors:

- Compared to AC fans using shaded-pole motors, they are very efficient, running much cooler than the equivalent AC motors. This cool operation leads to much-improved life of the fan's bearings.
- Without a commutator to wear out, the life of a DC brushless motor can be significantly longer compared to a DC motor using brushes and a commutator. Commutation also tends to cause a great deal of electrical and RF noise; without a commutator or brushes, a brushless motor may be used in electrically sensitive devices like audio equipment or computers.
- The same Hall effect sensors that provide the commutation can also provide a convenient tachometer signal for closed-loop control (servo-controlled) applications. In fans, the tachometer signal can be used to derive a "fan OK" signal.
- The motor can be easily synchronized to an internal or external clock, leading to precise speed control.
- Brushless motors have no chance of sparking, unlike brushed motors, making them better suited to environments with volatile chemicals and fuels.
- Brushless motors are usually used in small equipment such as computers and are generally used to get rid of unwanted heat.
- They are also very quiet motors which is an advantage if being used in equipment that is affected by vibrations.
- Modern DC brushless motors range in power from a fraction of a watt to many kilowatts. Larger brushless motors up to about 100 kW rating are used in electric vehicles. They also find significant use in high-performance electric model aircraft.

FEATURES BRUSHLESS DC MOTOR

- Long life span and no maintenance
- High efficiency(85-90)
- High reliability
- Noise reduction and elimination of commutator losses

- High cost
- Complexity of motor speed control
- Require external communication like controller to generate torque
- Transferring of power from driver to rotor is easy
- It consists of permanent magnets external to rotor
- 3-phase driving coils
- One or more hall effect sensor
- Uses where exact speed control is necessary

7.1 L293D DRIVER

Because of induction of the windings, power requirements, and temperature management, some glue circuitry is necessary between digital controllers and motor. In our project to interface DC motor with microcontroller we use L293D driver.

L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo- Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state.

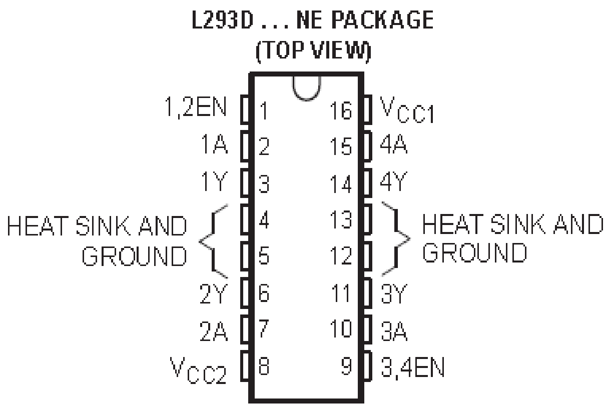


Figure 8.2: Pin Diagram Of L293D

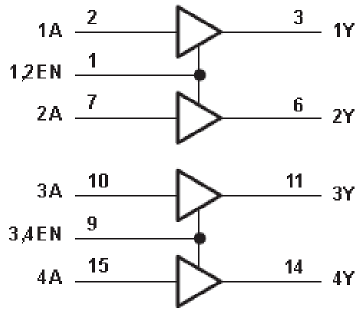


Figure 8.3: Internal Block Diagram Of L293D

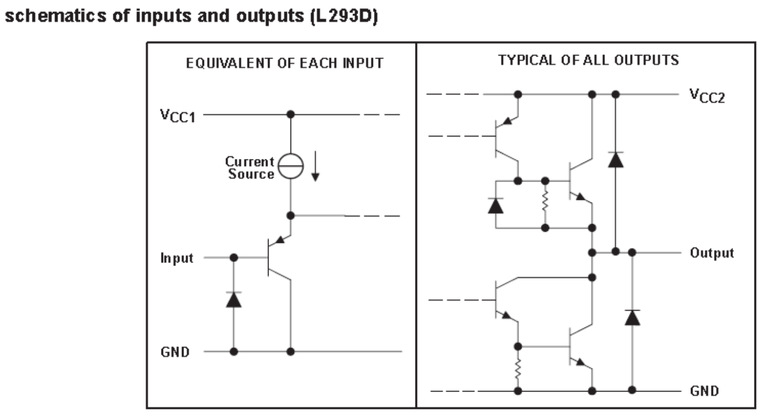


Figure 8.4: Input And Output Compatibles

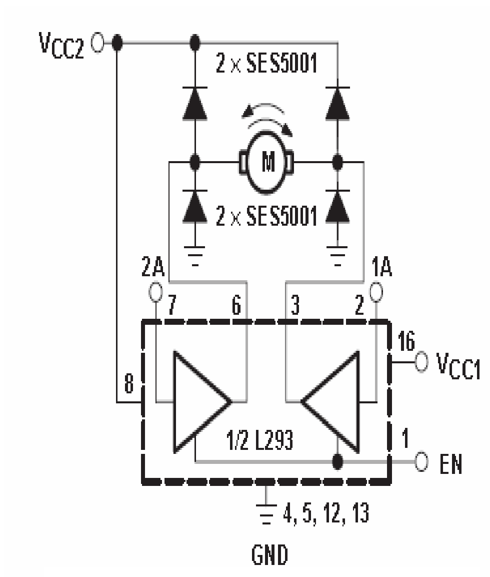
FEATURES OF L293D

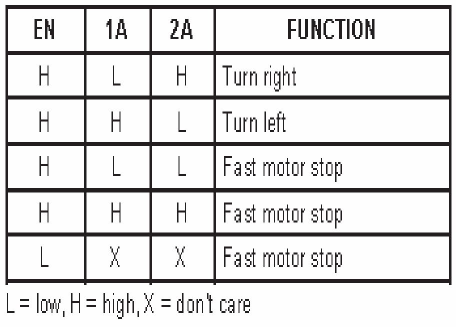
- Used has Dc motor driver
- Supply voltage range 4.5V to 36V
- High current half H drivers

- High noise immunity input
- Out put current of 600mA per channel
- Out put peak current of 1.2mA per channel
- Thermal shut down
- Output clamp diodes for inductive transient suppression
- Input circuit are TTL compatible
- Output circuit are totem pole with Darlington transistor pair
- Maximum input voltage is 7V
- Output voltage range is -3V to 39V
- Storage temperature range is -65 to 150C
- It is a 16 pin DIP IC configuration

DC MOTOR INTERFACING WITH L293D

By using single 16 pin L293D driver we can interface 2 dc motor. In our project we use 2 Dc motors one for front wheels used to turn left or right and other for movement of motor as forward and reverse direction. For one motor we give two inputs to rotate forward and backward directions or to turn right/left. Below figure is an example for interfacing a single Dc motor to L293D driver. Where 2 & 7 pin of L293D are inputs from microcontroller ports these input are enable only when 1pin of L293D is high and 3 & 6 pins of L293D are inputs to dc motor. Diodes used externally for inductive transient suppression. Supply is from power supply generator. In this when enable low then motor stops and when enable is high and both inputs are same that is either high or low then also motor stops. It works only when one input is high and enable is high.





EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

Figure 8.5: Interfacing of DC Motor With L293D

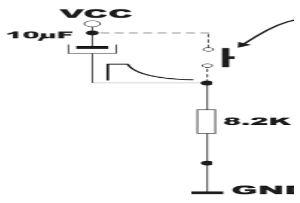
9. LEDS AND SWITCHS

LED

A light-emitting diode (LED) is an electronic light source. LEDs are based on the semiconductor diode. When the diode is forward biased, electrons are able to recombine with holes and energy is released in the form of light. This effect is called electroluminescence and the color of the light is determined by the energy gap of the semiconductor. The LED is usually small in area with integrated optical components to shape its radiation pattern and assist in reflection. Applications of LEDs are diverse. They are used as low-energy and also for replacements for traditional light sources in well-established applications such as indicators and automotive lighting. The compact size of LEDs has allowed new text and video displays and sensors to be developed, while their high switching rates are useful in communications technology.

Figure9.1: Led

Led's are connect to the ports of microcontroller by using transistors and resisters. Transistor is used to decrease power dissipation and Led's are glow from external supply instead of microcontroller.



SWITCH

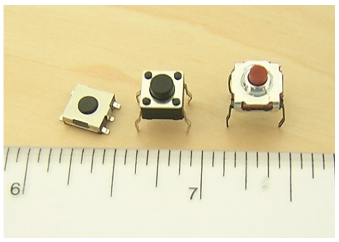
In electronics, a **switch** is an electronic electronics, a **switch** is an electrical component that can break an electrical circuit, interrupting the current or diverting it from one conductor to another. The most familiar form of switch is a manually operated electromechanical device with one or more sets of electrical contacts. Each set of contacts can be in one of two states: either 'closed' meaning the contacts are touching and electricity can flow between them, or 'open', meaning the contacts are separated and non conducting.In this at the time of switch pressed(supply applied) the voltage across resister 8.2Kohms is VCC as capacitor is short circuit. And this switches are connected to LEDs by using microcontroller program. When we release the switch the capacitor get charges to VCC.

Figure 9.2: Internal Circuit Of A Manual Switch

Figure 9.3: Switches

9. Liquid Crystal Display [LCD]

We examine an intelligent LCD display of two lines, 16 characters per line that is interfaced to the 8051.The protocol (handshaking) for the display is as shown. The display contains two internal byte-wide registers, one for commands (instructions) (RS=0) and the second for characters (data) to be displayed (RS=1).It also contains a user-programmed RAM area (the character RAM) that can be programmed to generate any desired character that can be formed using a dot matrix. To distinguish between these two data areas, the hex command byte 80 will be used to signify that the display RAM address 00h will be chosen Port1 is used to furnish the command or data type, and ports 3.2 to 3.4 furnish register select and read/write levels.



The display takes varying amounts of time to accomplish the functions as listed. LCD bit 7 is monitored for logic high (busy) to ensure the display is overwritten. A slightly more complicated LCD display (4 lines*40 characters) is currently being used in medical diagnostic systems to run a very similar program.

Liquid Crystal Display

Figure: 10.1 Liquid Crystal Display

\

Pins Description

1 Ground	2 Vcc
3 Contrast Voltage	4"R/S"_Instruction(0)/data(1) Select
5 "R/W" Read(1)/Write(0) LCD Registers	6 "E" Clock
7 - 14 Data I/O Pins	8 A(anode) back light power supply 5V
9 K(cathode) back light power supply GND	

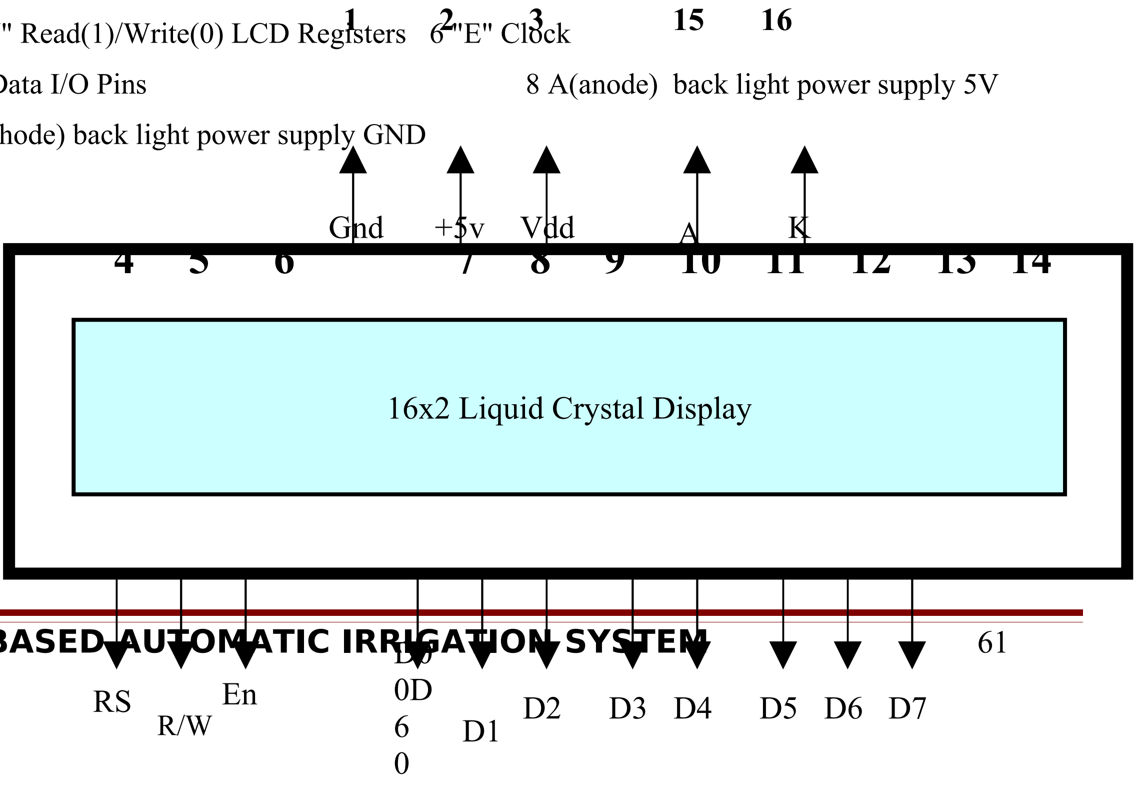
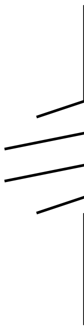


Figure: 10.2 Showing LCD Pins



11.RELAY

Introduction:-

A relay is an electrical switch that opens and closes under control of another electrical circuit. In the original form, the switch is operated by an electromagnet to open or close one or many sets of contacts. It was invented by Joseph Henry in 1835. Because a relay is able to control an output circuit of higher power than the input circuit, it can be considered, in a broad sense, to be a form of an electrical amplifier.

Operation:-

When a current flows through the coil, the resulting magnetic field attracts an armature that is mechanically linked to a moving contact. The movement either makes or breaks a connection with a fixed contact. When the current to the coil is switched off, the armature is returned by a force approximately half as strong as the magnetic force to its relaxed position. Usually this is a spring, but gravity is also used commonly in industrial motor starters. Most relays are manufactured to operate quickly. In a low voltage application, this is to reduce noise. In a high voltage or high current application, this is to reduce arcing.

If the coil is energized with DC, a diode is frequently installed across the coil, to dissipate the energy from the collapsing magnetic field at deactivation, which would otherwise generate a spike of voltage and might cause damage to circuit components. If the coil is designed to be energized with AC, a small copper ring can be crimped to the end of the solenoid. This "shading ring" creates a small out-of-phase current, which increases the minimum pull on the armature during the AC cycle.
By analogy with the functions of the original electromagnetic device, a solid-state relay is made with a thyristor or other solid-state switching device. To achieve electrical isolation, a light-emitting diode (LED) is used with a photo transistor.

Types of relay:-

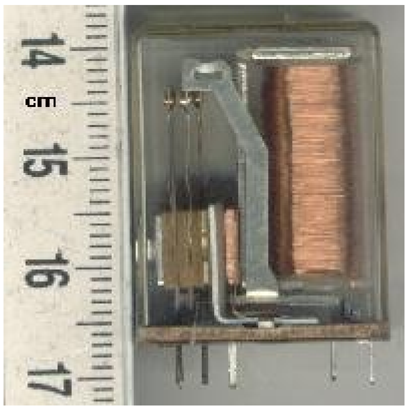


Figure: 11.1: latching relay

1) Latching relay:-

A latching relay has two relaxed states (bistable). These are also called 'keep' relays. When the current is switched off, the relay remains in its last state. This is achieved with a solenoid operating a ratchet and cam mechanism, or by having two opposing coils with an over-center spring or permanent magnet to hold the armature and contacts in position while the coil is relaxed, or with a remnant core. In the ratchet and cam example, the first pulse to the coil turns the relay on and the second pulse turns it off.

In the two coil example, a pulse to one coil turns the relay on and a pulse to the opposite coil turns the relay off. This type of relay has the advantage that it consumes power only for an instant, while it is being switched, and it retains its last setting across a power outage.

2) Reed relay:-

A reed relay has a set of contacts inside a vacuum or inert gas filled glass tube, which protects the contacts against atmospheric corrosion. The contacts are closed by a magnetic field generated when current passes through a coil around the glass tube. Reed relays are capable of faster switching speeds than conventional relays. See also reed switch.

2.1) Mercury-wetted relay:-

A mercury-wetted relay is a form of reed relay in which the contacts are wetted with mercury. Such relays are used to switch low-voltage signals (one volt or less) because of its low contact resistance, or for high-speed counting and timing applications where the mercury eliminated contact bounce. Mercury wetted relays are position-sensitive and must be mounted vertically to work properly. Because of the toxicity and expense of liquid mercury, these relays are rarely specified for new equipment. See also mercury switch.

3) Polarized relay:-

A Polarized Relay placed the armature between the poles of a permanent magnet to increase sensitivity. Polarized relays were used in middle 20th Century telephone exchanges to detect faint pulses and correct telegraphic distortion. The poles were on screws, so a technician could first adjust them for maximum sensitivity and then apply a bias spring to set the critical current that would operate the relay.

4) Machine tool relay:-

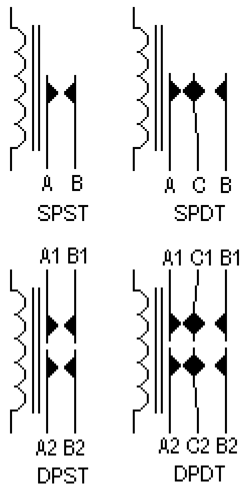
A machine tool relay is a type standardized for industrial control of machine tools, transfer machines, and other sequential control. They are characterized by a large number of contacts (sometimes extendable in the field) which are easily converted from normally-open to normally-closed status, easily replaceable coils, and a form factor that allows compactly installing many relays in a control panel. Although such relays once were the backbone of automation in such industries as automobile assembly, the programmable logic controller mostly displaced the machine tool relay from sequential control applications.

5) Contactor relay:-

A contactor is a very heavy-duty relay used for switching electric motors and lighting loads. With high current, the contacts are made with pure silver. The unavoidable arcing causes the contacts to oxidize and silver oxide is still a good conductor. Such devices are often used for motor starters. A motor starter is a contactor with an overload protection devices attached. The overload sensing devices are a form of heat operated relay where a coil heats a bi-metal strip, or where a solder pot melts, releasing a spring to operate auxiliary contacts. These auxiliary contacts are in series with the coil. If the overload senses excess current in the load, the coil is de-energized. Contactor relays can be extremely loud to operate, making them unfit for use where noise is a chief concern.

Pole & Throw:-

Circuit symbols of relays. "C" denotes the common terminal in SPDT and DPDT types. Since relays are switches, the terminology applied to switches is also applied to relays. According to this classification, relays can be of the following types:



* SPST - Single Pole Single Throw. These have two terminals which can be switched on/off. In total, four terminals when the coil is also included.

* SPDT - Single Pole Double Throw. These have one row of three terminals. One terminal (common) switches between the other two poles. It is the same as a single change-over switch. In total, five terminals when the coil is also included.

* DPST - Double Pole Single Throw. These have two pairs of terminals. Equivalent to two SPST switches or relays actuated by a single coil. In total, six terminals when the coil is also included. This configuration may also be referred to as DPNO.

* DPDT - Double Pole Double Throw. These have two rows of change-over terminals. Equivalent to two SPDT switches or relays actuated by a single coil. In total, eight terminals when the coil is also included.

* QPDT - Quadruple Pole Double Throw. Often referred to as Quad Pole Double Throw, or 4PDT. These have four rows of change-over terminals. Equivalent to four SPDT switches or relays actuated by a single coil or two DPDT relays. In total, fourteen terminals when the coil is also included.

The contacts can be either Normally Open (NO), Normally Closed (NC), or change-over (CO) contacts.

* Normally-open contacts connect the circuit when the relay is activated; the circuit is disconnected when the relay is inactive. It is also called Form A contact or "make" contact. Form A contact is ideal for applications that require to switch a high-current power source from a remote device.
* Normally-closed contacts disconnect the circuit when the relay is activated; the circuit is connected when the relay is inactive. It is also called Form B contact or "break" contact. Form B contact is ideal for applications that require the circuit to remain closed until the relay is activated.
* Change-over contacts control two circuits: one normally-open contact and one normally-closed contact with a common terminal. It is also called Form C contact or "transfer" contact.

Applications:-

Relays are used:-

* To control a high-voltage circuit with a low-voltage signal, as in some types of modems,
* To control a high-current circuit with a low-current signal, as in the starter solenoid of an automobile,
* To detect and isolate faults on transmission and distribution lines by opening and closing circuit breakers (protection relays),
* To isolate the controlling circuit from the controlled circuit when the two are at different potentials, for example when controlling a mains-powered device from a low-voltage switch. The latter is often applied to control office lighting as the low voltage wires are easily installed in partitions, which may be often moved as needs change.

* To perform logic functions. For example, the Boolean AND function is realized by connecting NO relay contacts in series, the OR function by connecting NO contacts in parallel. The change-over or Form C contacts perform the XOR (exclusive or) function.

Similar functions for NAND and NOR are accomplished using NC contacts. Due to the failure modes of a relay compared with a semiconductor, they are widely used in safety critical logic, such as the control panels of radioactive waste handling machinery.

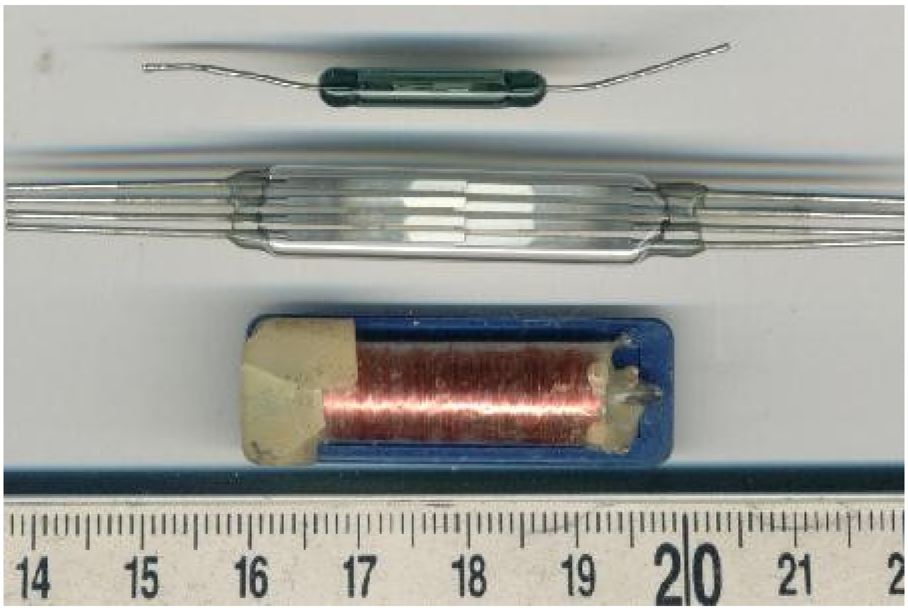
* To perform time delay functions. Relays can be modified to delay opening or delay closing a set of contacts. A very shorts (a fraction of a second) delay would use a copper disk between the armature and moving blade assembly. Current flowing in the disk maintains magnetic field for a short time, lengthening release time. For a slightly longer (up to a minute) delay, a dashpot is used. A dashpot is a piston filled with fluid that is allowed to escape slowly. The time period can be varied by increasing or decreasing the flow rate. For longer time periods, a mechanical clockwork timer is installed.

Protective relay:-

A protective relay is a complex electromechanical apparatus, often with more than one coil, designed to calculate operating conditions on an electrical circuit and trip circuit breakers when a fault was found. Unlike switching type relays with fixed and usually ill-defined operating voltage thresholds and operating times, protective relays had well-established, selectable, time/current (or other operating parameter) curves.

Such relays were very elaborate, using arrays of induction disks, shaded-pole magnets, operating and restraint coils, solenoid-type operators, telephone-relay style contacts, and phase-shifting networks to allow the relay to respond to such conditions as over-current, over-voltage, reverse power flow, over- and under- frequency, and even distance relays that would trip for faults up to a certain distance away from a substation but not beyond that point. An important transmission line or generator unit would have had cubicles dedicated to protection, with a score of individual electromechanical devices.

Each of the protective functions available on a given relay is denoted by standard ANSI Device Numbers. For example, a relay including function 51 would be a timed over current protective relay.



Design and theory of these protective devices is an important part of the education of an electrical engineer who specializes in power systems. Today these devices are nearly entirely replaced (in new designs) with microprocessor-based instruments (numerical relays) that emulate their electromechanical ancestors with great precision and convenience in application. By combining several functions in one case, numerical relays also save capital cost and maintenance cost over electromechanical relays. However, due to their very long life span, tens of thousands of these "silent sentinels" are still protecting transmission lines and electrical apparatus all over the world.

12. KEIL SOFTWARE

Simulator/Debugger

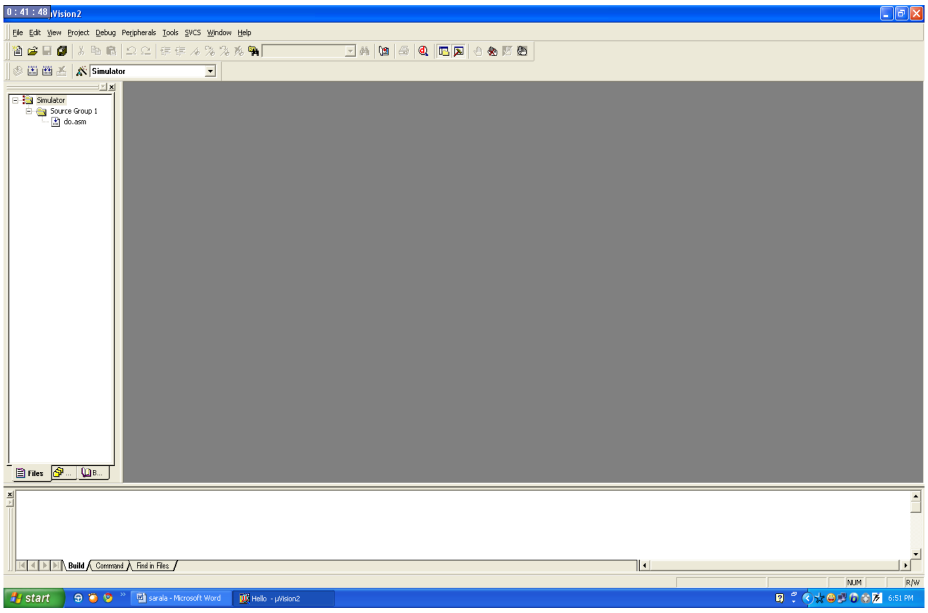
The simulator/ debugger in KEIL can perform a very detailed simulation of a micro controller along with external signals. It is possible to view the precise execution time of a single assembly instruction, or a single line of C code, all the way up to the entire application, simply by entering the crystal frequency. A window can be opened for each peripheral on the device, showing the state of the peripheral. This enables quick trouble shooting of mis-configured peripherals. Breakpoints may be set on either assembly instructions or lines of C code, and execution may be stepped through one instruction or C line at a time. The contents of all the memory areas may be viewed along with ability to find specific variables. In addition the registers may be viewed allowing a detailed view of what the microcontroller is doing at any point in time.

The Keil Software 8051 development tools listed below are the programs you use to compile your C code, assemble your assembler source files, link your program together, create HEX files, and debug your target program. µVision2 for Windows™ Integrated Development Environment: combines Project Management, Source Code Editing, and Program Debugging in one powerful environment.

- C51 ANSI Optimizing C Cross Compiler: creates relocatable object modules from your C source code,
- A51 Macro Assembler: creates relocatable object modules from your 8051 assembler source code,
- BL51 Linker/Locator: combines relocatable object modules created by the compiler and assembler into the final absolute object module,
- LIB51 Library Manager: combines object modules into a library, which may be used by the linker,
- OH51 Object-HEX Converter: creates Intel HEX files from absolute object modules.

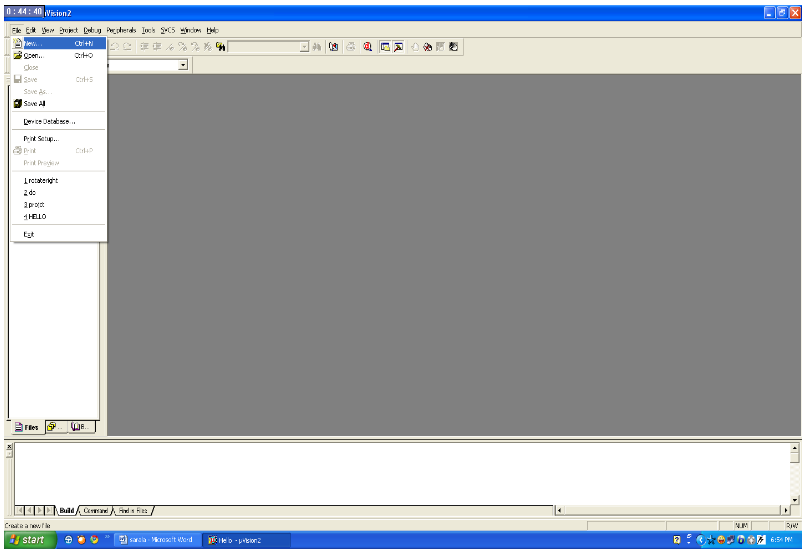
EVALUATION OF KEIL SOFTWARE

1. Start the µVision Program

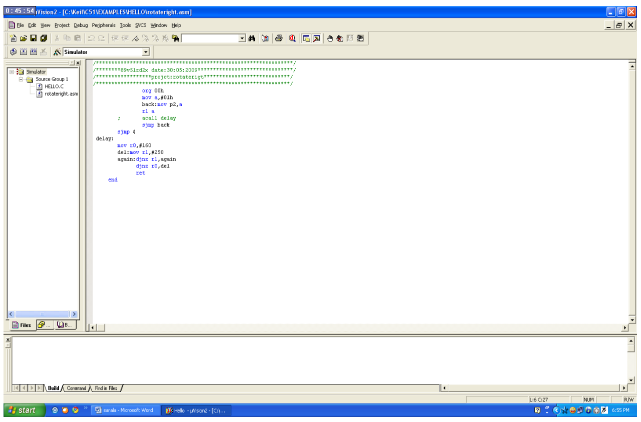


2. After the program has started:

Select File, New… from the program menu

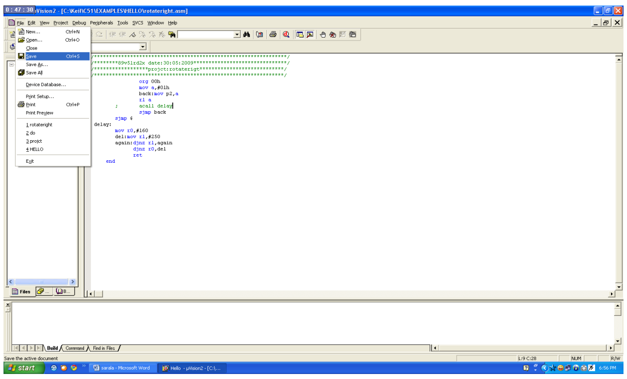


Type your assembly file. The following is an example of a toggle program.



```
/**************************************************************/
/********89v51rd2x date:30:05:2009******************************/
/*****************projct:rotaterigt****************************/
/**************************************************************/
                org 00h
                mov a,#01h
                back:mov p2,a
                rl a
        ;       acall delay
                sjmp back
        sjmp $
 delay:
        mov r0,#160
        del:mov r1,#250
        again:djnz r1,again
              djnz r0,del
              ret
     end
```

3.   Select File, Save… from the program menu



```
/**************************************************************/
/********89v51rd2x date:30:05:2009******************************/
/*****************projct:rotaterigt****************************/
/**************************************************************/
                org 00h
                mov a,#01h
                back:mov p2,a
                rl a
        ;       acall delay
                sjmp back
        sjmp $
 delay:
        mov r0,#160
        del:mov r1,#250
        again:djnz r1,again
              djnz r0,del
              ret
     end
```

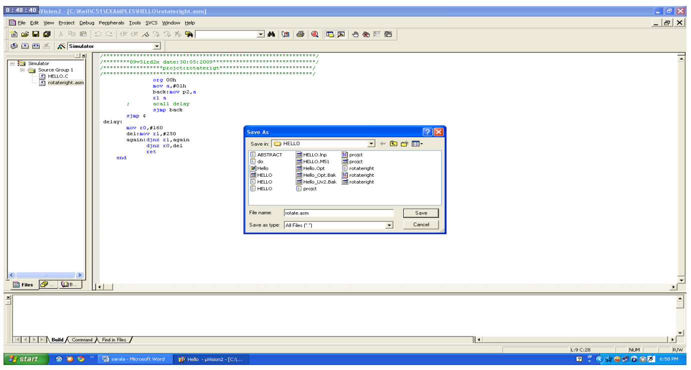
The first time you save the program a dialog box will popup and allow you to name your file and file type.

Save program with filename: xxxxx.asm

The File type is mentioned at last (.asm) means assembly language

4.   Select Project, New Project… from the program menu

Give some project name: xxxx.prj



Save in: HELLO

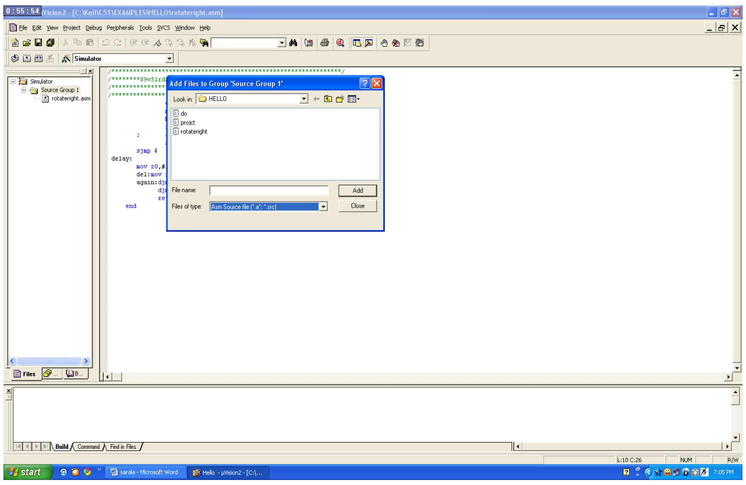
File name: rotate.asm

Save as type: All Files (*.*)

5. Click on the Add button

A dialog-box appears, allowing you to add files to the project

Change the file type to Assembly.


0 : 55 : 54
µVision2 - [C:\Keil\C51\EXAMPLES\HELLO\rotateright.asm]
File Edit View Project Debug Peripherals Tools SVCS Window Help
Simulator
Source Group 1
rotateright.asm
Build
Command
Find in Files
start

Add Files to Group 'Source Group 1'

Look in: HELLO

- do
- projct
- rotateright

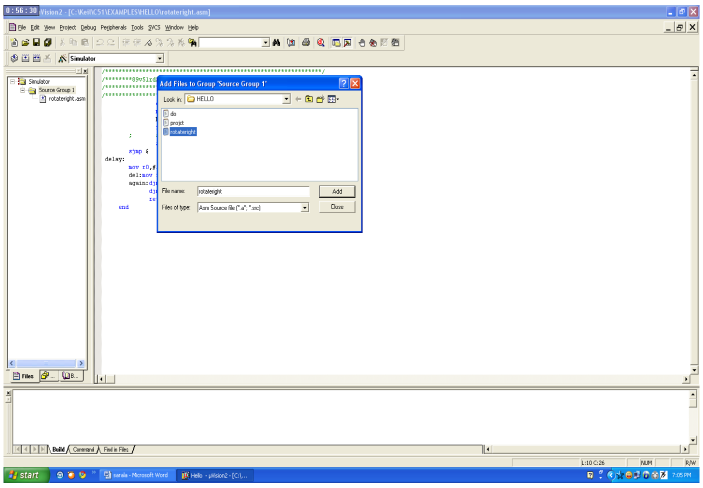
File name:

Files of type: Asm Source file (*.a*; *.src)

Add | Close

6. Select your assembly file.

Click on the Add button then close the Add dialog box.


0 : 56 : 30
µVision2 - [C:\Keil\C51\EXAMPLES\HELLO\rotateright.asm]
File Edit View Project Debug Peripherals Tools SVCS Window Help
Simulator
Source Group 1
rotateright.asm
Build
Command
Find in Files
start

Add Files to Group 'Source Group 1'

Look in: HELLO

- do
- projct
- rotateright

File name: rotateright

Files of type: Asm Source file (*.a*; *.src)

Add | Close

13. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- This project controls the on-off action of the motor in the field
- Low cost and easy to implement.
- Can cover maximum area in a field

DISADVANTAGES

- Difficult in case of failure of GSM modem
- Kit is to be protected from reaching water.

14. CODE

```
#include <at89c51xd2.h>
#include <string.h>
#include <stdio.h>
#include "usart.h"
#include "lcd.h"
#include "gsm.h"

// Inputs
#define moistureSensor1 P1_4
#define moistureSensor2 P1_5
#define moistureSensor3 P1_6
#define moistureSensor4 P1_7

//Outputs
#define relayMotor P0_0
#define NOR_CONDITION 1
#define DRY_CONDITION 2
#define WET_CONDITION 3

unsigned char fieldStatus = 0;



unsigned char motorOnFlag = 0;
unsigned char motorOffFlag = 0;
unsigned char moisSenCnt = 0;
unsigned char mtOnSmsFlag = 0;
unsigned char mtOffSmsFlag = 0;

unsigned char smsSendFlag = 0;
xdata unsigned char smsMessage[ 100 ];

void main( void )
```

```
{
            xdata unsigned char messageBuffer[200];
            int retValue = 0;
            unsigned char *myString1 = "Starting........";
            unsigned char *myString2 = "................";
            unsigned char *myString3 = "Wting 4 Command ";
            unsigned char *myString4 = "Command Recieved";
            unsigned char colunCount = 0, index = 0;
            xdata unsigned char gsmCommand[7];
            unsigned int waitCnt = 0;

            P2 = 0xFF;
            P3 = 0xFF;
            relayMotor=0;
            DelayMs(5);

            Lcd_Init();
            SenStringToLcd ( 1, myString1 );
            SenStringToLcd ( 2, myString2 );

            smsSendFlag = 0;

            USART_Init_9600();

            retValue = DeleteSms();
            if( retValue == 0 ){

                        SenStringToLcd ( 1, "GSM Modem is    " );
                        SenStringToLcd ( 2, "Not Functioning " );
            }

            while(1){

                        moisSenCnt = 0;
```

```
motorOnFlag = 0;
motorOffFlag = 0;

SenStringToLcd ( 1, " Sensing field  " );
SenStringToLcd ( 2, "   Condition    " );
DelayMs( 100 );

if( moistureSensor1 == 1 )
        moisSenCnt++;
if( moistureSensor2 == 1 )
        moisSenCnt++;
if( moistureSensor3 == 1 )
        moisSenCnt++;
if( moistureSensor4 == 1 )
        moisSenCnt++;

SenStringToLcd ( 1, "  Field Sensed  " );
SenStringToLcd ( 2, "****************" );

fieldStatus = NOR_CONDITION;

if( moisSenCnt <= 2 ){
        fieldStatus = DRY_CONDITION;
        motorOnFlag = 1;
        motorOffFlag = 0;
        relayMotor=1;
        SenStringToLcd ( 1, "Field Is In Dry " );
        SenStringToLcd ( 2, "    Condition   " );
        DelayMs( 300 );
}
else{
        fieldStatus = WET_CONDITION;
        motorOffFlag = 1;
        motorOnFlag = 0;
        relayMotor=0;
        SenStringToLcd ( 1, "Field Is In Wet " );
        SenStringToLcd ( 2, "    Condition    " );
        DelayMs( 00 );
}
SenStringToLcd ( 1, "****************" );
SenStringToLcd ( 2, "****************" );
DelayMs( 100 );

SenStringToLcd ( 2, myString3 );
DelayMs( 100 );
SenStringToLcd ( 2, myString2 );
DelayMs( 100 );

retValue = 0;
strcpy( messageBuffer, "\0" );
retValue = ReadSms( messageBuffer );
if( retValue == 1 ){

        index = 0;
        colunCount = 0;
        waitCnt = 0;
        while( (colunCount < 6) && (waitCnt < 10000) ){
                if(     messageBuffer[ index ] == '"' ){

                        colunCount++;
                }
                index++;
                waitCnt++;
        }

        SenStringToLcd ( 1, myString4 );
        strcpy( gsmCommand, "\0" );
        strncpy( gsmCommand, &messageBuffer[index+2],6 );
```

```
gsmCommand[6] = '\0';
SenStringToLcd ( 2, myString2 );
DelayMs( 10 );
SenStringToLcd ( 2, gsmCommand );
DelayMs( 250 );

if ( !strcmp( gsmCommand, "MTR ON" ) ){
        relayMotor=1;
        motorOnFlag = 1;
        motorOffFlag = 0;
        SenStringToLcd ( 1, "    MOTOR ON    " );
        SenStringToLcd ( 2, "****************" );
}
else if ( !strcmp( gsmCommand, "MTR OF" ) ){
        relayMotor=0;
        motorOffFlag = 1;
        motorOnFlag = 0;
        SenStringToLcd ( 1, "    MOTOR OFF    " );
        SenStringToLcd ( 2, "****************" );
}
else if ( !strcmp( gsmCommand, "GIFMOD" ) ){
        smsSendFlag = 1;
        SenStringToLcd ( 1, "Request Received" );
        SenStringToLcd ( 2, "****************" );
        DelayMs( 100 );
}
else{
        SenStringToLcd ( 1, "Invalid Command " );
        DelayMs( 200 );
        SenStringToLcd ( 2, "Received        " );
        DelayMs( 200 );
}

if( smsSendFlag == 1 ){
```

```
                    smsSendFlag = 0;

                    if( fieldStatus == NOR_CONDITION ){
                        strcpy( smsMessage, "Field Is In Normal
Condition." );
                        SendSms( "+919490045729", smsMessage );
                        DelayMs( 500 );
                        SendSms( "+919703935935", smsMessage );
                        DelayMs( 500 );
                    }
                    else if( fieldStatus == DRY_CONDITION ){
                        strcpy( smsMessage, "Field Is In Dry Condition."
);
                        SendSms( "+919490045729", smsMessage );
                        DelayMs( 500 );
                        SendSms( "+919703935935", smsMessage );
                        DelayMs( 500 );
                    }
                    else if( fieldStatus == WET_CONDITION ){
                        strcpy( smsMessage, "Field Is In Wet
Condition." );
                        SendSms( "+919490045729", smsMessage );
                        DelayMs( 500 );
                        SendSms( "+919703935935", smsMessage );
                        DelayMs( 500 );
                    }
                    else{
                    }
                }

                if( motorOnFlag == 1 && mtOnSmsFlag == 0){

                    motorOnFlag = 0;
```

```
                        mtOnSmsFlag = 1;
                        mtOffSmsFlag = 0;
                        strcpy( smsMessage, "Motor On In the field" );

                        SendSms( "+919490045729", smsMessage );
                        DelayMs( 500 );
                        SendSms( "+919703935935", smsMessage );
                        DelayMs( 500 );
                    }
                    if( motorOffFlag == 1 && mtOffSmsFlag == 0){


                        motorOffFlag = 0;
                        mtOffSmsFlag = 1;
                        mtOnSmsFlag = 0;
                        strcpy( smsMessage, "Motor Off In the field" );

                        SendSms( "+919490045729", smsMessage );
                        DelayMs( 500 );
                        SendSms( "+919703935935", smsMessage );
                        DelayMs( 500 );
                    }

                    SenStringToLcd ( 1, "Command Executed");
                    DelayMs( 300 );


                    while( DeleteSms() == 0);
                    SenStringToLcd ( 1, "Command Deleted ");
                    SenStringToLcd ( 2, myString2 );
                }
        } // End Of while(1);
}
```

CONCLUSION

The project ***"GSM BASED AUTOMATIC IRRIGATION SYSTEM"*** has been successfully designed and tested.

It has been developed by integrating features of all the hardware components used. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. Thus monitoring the functioning of the motor automatically using GSM technology got designed with the specific parameters.

Secondly, using highly advanced IC's and with the help of growing technology the project has been successfully implemented.

This is a very useful technique to control the motor functioning.

- By using Microcontroller , we Controlled the on off action of the motor.
- It is mainly useful in the areas where the power fluctuations are high.

BIBILOGRAPHY

1. Theodore S. Rappaport, Wireless Communications Principles and Practices, second edition,2001
2. Lathi, Digital Communications ,g.k publisher,2003
3. Sklar ,Digital Communications, second edition

WEBREFERENCES:

1. www.electronicstutorials.com
2. www.aimglobal.com
3. www.kernel.org
4. ONLamp.com