CHAPTER 1

PROJECT OVERVIEW

PROJECT OVERVIEW

1.1 INTRODUCTION

Modern age is the age of computers and all recent equipment are working on the principles of digital communication techniques and efforts have been made to upgrade older equipments to the new standards, hence increasing their service life cycle. This in addition to enhanced efficiency has a pleasant effect on the economy. In this context, one area which has been neglected and holds great potential is military inventory. It includes equipment which may be as big as aeroplanes to radars, tanks and other communication systems. It will save not only foreign exchanges but also a step towards self sufficiency and decrease our dependence on other countries. So we decided to focus our efforts on this department which has been ignored in the past. One such example is AN/TPQ-36 artillery locating radar currently used by artillery.

AN/TPQ-36 is an American radar and a very costly equipment acquired in late 80's. It locates enemy artillery guns and acquires their exact location. Currently it is being operated by an operator who performs all the functions manually i.e. the operator manually plots the coordinates on a map. And then through a radio set passes the information to the concerned parties. This consumes a lot of time and is prone to human errors. The solution to problem is either to acquire the digital map kit from the manufacturer at a handsome cost or to try and develop a much cheaper solution locally. Apart from the cost factor there were other hurdles in the form of the imposed sanctions. C4I has been working on this project and the project undertaken is the endeavour in the same direction.

The first and the most important task is to acquire the data generated by the radar so that it can be used with digital maps. Developing an interface with the radar is a gigantic task . It is so because of the complexity of radar circuitry it self

and non compatibility of its components with modern equipment. Most of the IC's used are unknown and hence in case of malfunction cannot be repaired. Prior to our involvement projects had been undertaken on the same lines which were unsuccessful. Hence there were two options available to us; either to continue with incomplete work of the previous attempts or to develop and implement a design of our own. We decided to go with the second option.

The second important issue is transfer of data generated by the radar after it was transferred to the computer. It requires developing an interface which could link the data with the digital maps. Our problem was some what simplified because we didn't have to digitize the map as we were using the digitized map that is currently being used by the C4I. For this purpose they were using a GIS (Geographical Information System) Geomedia. Our objective was to develop the software that will use the data useful software. The design phase was completed in visual c++ and the code for this is given at the end .

1.2 OVERVIEW

FIGURE 1.1 PROJECT OVERVIEW

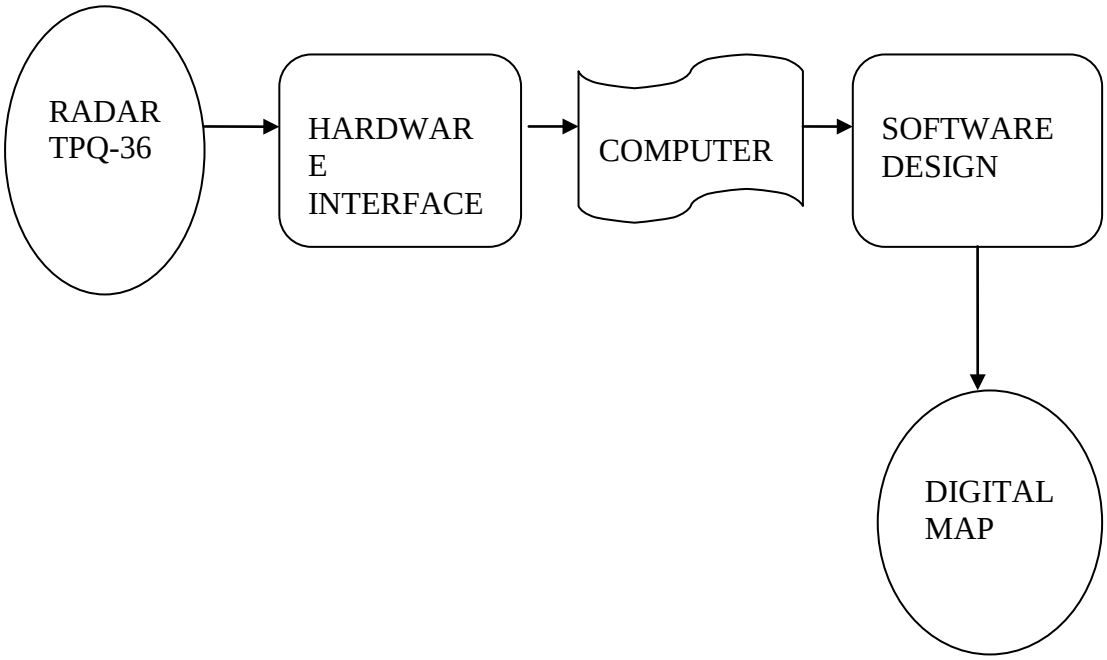


FIGURE 1.1 shows the modular diagram of the project. The information from the radar (AN/TPQ-36) is fed to the computer through a hardware interface. The main project effort was concentrated in designing and implementing this module. The data was transferred to the computer through its serial port. The emphasis was given to synchronization of the source and the receiver. The parameter which governs the transfer of data is the baud rate. In order to use the data received correctly it had to be converted to a specific format. For this an algorithm in c/visual c++ was written so that it is compatible with the digital map. A software that includes codes for converting data into required format and displaying the location of the gun on the digital map was developed and successfully implemented. The main challenge in the project was its real time implementation. The main issues adhered were synchronization of the transmitter and the receiver.

1.3 PROJECT MODULES.

The project comprised of the following modules.

- Studying the radar
- Selection of data source
- Designing Hardware interface
- Studying computer communication ports
- Study and implementation of Software interface
- C/VC++ coding
- MATLAB based simulation
- Real time implementation

CHAPTER 2

STUDY OF RADAR

STUDY OF RADAR

2.1. INTRODUCTION

AN/TPQ-36 Firefinder Weapon Locating System general properties.

- Benefits
- Early threat detection from medium standoff ranges
- Detection of multiple threats— mortars, artillery, and rockets
- Support of friendly weapons, reduced fratricide
- Exceptional accuracy
- Superior mobility
- . Rapid emplacement and displacement .
- High system operational availability .
- Reduced maintenance costs and downtime
- Medium-Range Surveillance Raytheon's compact, mobile, combat-proven AN/TPQ-36
- Firefinder radar accurately, rapidly, and automatically locates medium-range enemy mortars, artillery, and rocket launchers. It can handle simultaneous fire from weapons at multiple locations, detecting and reporting their positions on the first round. No longer must front-line troops and armor be exposed to long periods of enemy mortar, artillery, and rocket attacks. The AN/TPQ-36 directs accurate counterfire to neutralize enemy positions. The AN/TPQ-36's automatic detection, tracking, and locating process is so fast that an enemy weapon's position can often be pinpointed before its projectile impacts. Locations are automatically corrected for Electronic Systems Sometimes the best offense is a good defense. In seconds, the mediumrange AN/TPQ-36 Firefinder Weapon Locating System detects and backplots enemy projectiles and provides pinpoint targeting data for the counterattack. altitude differences, using computer-stored digital maps, and presented to the operator in northing, easting, and altitude coordinates. The system is so automatic and simple to operate that, once set up, the operator need not be present in the operation control shelter to determine a weapon's location. Rounds from friendly weapons also can be tracked, for more accurate delivery. Defeats Enemy Firepower, Supports Friendly Weapons The AN/TPQ-36's stationary antenna sweeps a rapid sequence of beams along the horizon, forming an electronic radar curtain over a 90° area. Any target enetrating the curtain triggers an immediate verification beam. On verification, an automatic tracking sequence begins.AN/TPQ-36 Firefinder Weapon Locating System While tracking any single target, the radar continues scanning, locating, and tracking others. The AN/TPQ-36 can detect and report the positions of up to 10 different weapons in seconds, at a maximum range of 24 km. The system also corrects and improves the delivery of friendly fire. Signal and data processors test each track to filter out birds, aircraft, and other unwanted returns, giving the AN/TPQ-36 an extremely low false-location rate and a very high probability of location. Once the computer establishes a target's validity, it smooths the easured track data, deriving a trajectory that it extrapolates to establish the target's firing

position and impact location. Those data are displayed on a visual map and printed out in map coordinates. From the operation control shelter, the power-driven antenna can be tilted or rotated to any azimuth position. The system also offers a 360° sectoring mode Maximum range: 24 km

2.2 System Capabilities Specifications

Locates mortars, artillery, and rocket launchers Highly Mobile Compact and highly mobile, the AN/TPQ-36 supports rapid deployment of forces and close combat. It can be positioned and ready for operation in 15 minutes. It can be readied for movement in 5 minutes by a five-man crew. Because it can move quickly from one position to another, it is typically located close to the forward battle line in direct support of brigade operations. The AN/TPQ-36 comprises an antenna-transceiver trailer, a generator, and an operation control shelter that contains processing equipment, the weapon-locating unit, and communications equipment. For the U.S. army, the 2,500-lb shelter is carried on a HMMWV or a 2.5-ton truck; however, it can also be carried by other tactical vehicles. The manned operation control shelter can be located as far as Locates 10 weapons simultaneously Locates targets on first round Performs high-burst, datum-plane, and impact registrations Adjusts friendly fire Interfaces with tactical fire Predicts impact of hostile projectiles search one sector for a short period, then automatically rotate in turn to the other sectors.

Effective range

Artillery: 18 km
Rockets: 24 km
Azimuth sector: 90°
Frequency: X-band, 32 frequencies
Prime power: 115/200 VAC, 400 Hz, 3-phase, 8 kW
Peak transmitted power: 23 kW, minimum 50 m away from the unmanned antenna trailer.

Exceptionally Reliable With high system reliability and maintainability simplified by computer-controlled, built-in test equipment, Raytheon's AN/TPQ-36 provides unusually high system availability. On-line fault detection and off-line fault diagnostics alert the operator to system faults, directing repair action to the unit that must be replaced. Ninety percent of all repairs required in the field can be performed by the crew, with a mean-time-to-repair of only 30 minutes. The AN/TPQ-36's cost effectiveness is enhanced by its 90°–360° sector, small crew, ease of operation, and high availability. The operation control shelter can be used interchangeably with either the AN/TPQ-36 or the longer-range AN/TPQ-37, with only a software change, thereby providing operational flexibility and much greater life-cycle cost effectiveness. Features Permanent storage for 99 targets Field exercise mode Digital data interface .

FIGURE 2 AN/TPQ-36 FIREFINDER

2.3 DATA GENERATED BY THE RADAR

AN/TPQ-36 has a computer and supporting PCB which interface the computer with various components such as the line printer , magnetic tape unit,B-scope etc. when operational AN/TPQ-36 locates the exact location of enemy guns and generates the data which is provided to various peripherals through a controller called “Cartridge and Printer controller”. Cartridge and Printer controller (CPC) basically controls all the operations regarding the transfer of data to these devices .

2.3.1 DATA AT B-SCOPE

The computer sends the data to the B-scope where it is displayed as Easting, northing and altitude. The radar also generates the location of the impact as well.

2.3.2 DATA AT WLU(Weapon Location Unit)

The computer sends the same data to the weapon location unit . the format of the data is the same at all three terminals

2.3.3 DATA AT LINE PRINTER

computer sends seven bit data and a strobe to the line printer. In return the line printer sends two signals back to the line printer i.e. fault and ready. In simple words handshaking occurs between the printer and the computer through the cartridge and printer controller.

2.4 SELECTION OF DATA INPUT DEVICE

Computer sends data to

- B-scope
- WLU
- Line printer

So we had to select a point from which we can extract data with ease . after extensive study we came to the conclusion that line printer was the easiest and safest approach. The reason for the selection of line printer was that B-scope and WLU were difficult to access moreover PCB connected to them were very sensitive and the probability of damaging the components was higher. There was also a possibility of extracting data directly from the computer instead of the peripherals. But this proposal was rejected because an attempt had already been made by AWC(Air Weapon Complex) which was unsuccessful and they damaged some of the PCB's which were expensive. After keeping all these factors in mind we decided to extract data at the input of the line printer.

CHAPTER 3

INTERFACING THROUGH COMPUTER PORTS.

a. PARALLEL PORT.

3.1 Introduction to Parallel Ports

The Parallel Port is the most commonly used port for interfacing home made projects. This port will allow the input of up to 9 bits or the output of 12 bits at any one given time, thus requiring minimal external circuitry to implement many simpler tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. This will be a serial RS-232 port and thus, is a totally incompatible port. Newer Parallel Port's are standardized under the IEEE 1284 standard first released in 1994. This standard defines 5 modes of operation which are as follows,

1. Compatibility Mode.
2. Nibble Mode. *(Protocol not Described in this Document)*
3. Byte Mode. *(Protocol not Described in this Document)*
4. EPP Mode *(Enhanced Parallel Port).*
5. ECP Mode *(Extended Capabilities Port).*

The aim was to design new drivers and devices which were compatible with each other and also backwards compatible with the Standard Parallel Port (SPP). Compatibility, Nibble & Byte modes use just the standard hardware available on the original Parallel Port cards while EPP & ECP modes require additional hardware which can run at faster speeds, while still being downwards compatible with the Standard Parallel Port.

Compatibility mode or "Centronics Mode" as it is commonly known, can only send data in the forward direction at a typical speed of 50 kbytes per second but can be as high as 150+ kbytes a second. In order to receive data, you must change the mode to either

1. Nibble or
2. Byte mode.

1. **Nibble mode** can input a nibble *(4 bits)* in the reverse direction. E.g. from device to computer.
2. **Byte mode** uses the Parallel's bi-directional feature *(found only on some cards)* to input a byte *(8 bits)* of data in the reverse direction.

Extended and Enhanced Parallel Ports use additional hardware to generate and manage handshaking. To output a byte to a printer (or anything in that matter) using compatibility mode, the software must.

1. *Write the byte to the Data Port.*
2. *Check to see is the printer is busy. If the printer is busy, it will not accept any data, thus any data which is written will be lost.*
3. *Take the Strobe (Pin 1) low. This tells the printer that there is the correct data on the data lines. (Pins 2-9)*

4. *Put the strobe high again after waiting approximately 5 microseconds after putting the strobe low. (Step 3)*

This limits the speed at which the port can run at. The EPP & ECP ports get around this by letting the hardware check to see if the printer is busy and generate a strobe and /or appropriate handshaking. This means only one I/O instruction need to be performed, thus increasing the speed.

These ports can output at around 1-2 megabytes per second. The ECP port also has the advantage of using DMA channels and FIFO buffers, thus data can be shifted around without using I/O instructions.

3.2 Hardware Properties

The D-Type 25 pin connector is the most common connector found on the Parallel Port of the computer, while the Centronics Connector is commonly found on printers. The IEEE 1284 standard however specifies 3 different connectors for use with the Parallel Port. The first one, 1284 Type A is the D-Type 25 connector found on the back of most computers. The 2nd is the 1284 Type B which is the 36 pin Centronics Connector found on most printers.

IEEE 1284 Type C however, is a 36 conductor connector like the Centronics, but smaller.

Pin #(D-TYPE 25)	SPP SIGNAL	DIRECTION IN/OUT	REGISTER	HARDWARE INVERTED
1	NStrobe	In/Out	Control	Yes
2	Data 0	Out	Data	
3	Data 1	Out	Data	
4	Data 2	Out	Data	
5	Data 3	Out	Data	
6	Data 4	Out	Data	
7	Data 5	Out	Data	
8	Data 6	Out	Data	
9	Data 7	Out	Data	
10	NAck	In	Status	
11	Busy	In	Status	Yes
12	Paper-out PaperEnd	In	Status	
13	Select	In	Status	
14	nAuto Linefeed	In/Out	Control	Yes
15	nError/nFault	In	Status	
16	NInitialize	In/Out	Control	
17	nselect-Printer nselect-In	In/Out	Control	Yes
18-25	Ground	Gnd		

Table 3.1 Pin Assignments of the D-Type 25 pin Parallel Port Connector.

The above table uses "n" in front of the signal name to denote that the signal is active low. e.g. nError. If the printer has occurred an error then this line is low. This line normally is high, should the printer be functioning correctly. The "Hardware Inverted" means the signal is inverted by the Parallel card's hardware. Such an example is the Busy line. If +5v (Logic 1) was applied to this pin and the status register read, it would return back a 0 in Bit 7 of the Status Register.

The output of the Parallel Port is normally TTL logic levels. The voltage levels are the easy part. The current you can sink and source varies from port to port. Most Parallel Ports implemented in ASIC, can sink and source around 12mA. However these are just some of the figures taken from Data sheets, Sink/Source 6mA, Source 12mA/Sink 20mA, Sink 16mA/Source 4mA, Sink/Source 12mA. As you can see they vary quite a bit. The best bet is to use a buffer, so the least current is drawn from the Parallel Port.

3.3 Port Addresses

The Parallel Port has three commonly used base addresses. These are listed in table 2, below. The 3BCh base address was originally introduced used for Parallel Ports on early Video Cards. This address then disappeared for a while, when Parallel Ports were later removed from Video Cards. They has now reappeared as an option for Parallel Ports integrated onto motherboards, upon which their configuration can be changed using BIOS.

LPT1 is normally assigned base address 378h, while LPT2 is assigned 278h. However this may not always be the case as explained later. 378h & 278h have always been commonly used for Parallel Ports. The lower case h denotes that it is in hexadecimal. These addresses may change from machine to machine.

ADDRESS	NOTES:
3BCh-3BFh	Used for Parallel Ports which were incorporated in to Video Cards and now, commonly an option for Ports controlled by BIOS. - Doesn't support ECP addresses.
378h-37Fh	Usual Address For LPT 1
278h-27Fh	Usual Address For LPT 2

Table 3.2 Port Addresses

When the computer is first turned on, BIOS (Basic Input/Output System) will determine the number of ports you have and assign device labels LPT1, LPT2 & LPT3 to them. BIOS first looks at address 3BCh. If a Parallel Port is found here, it is assigned as LPT1, then it searches at location 378h. If a Parallel card is found there, it is assigned the next free device label. This would be LPT1 if a card wasn't found at 3BCh or LPT2 if a card was found at 3BCh. The last *port of call*, is 278h and follows the same procedure than the other two ports. Therefore it is possible to have a LPT2 which is at 378h and not at the expected address 278h.

What can make this even confusing, is that some manufacturers of Parallel Port Cards, have jumpers which allow you to set your Port to LPT1, LPT2, LPT3. Now what address is

LPT1? - On the majority of cards LPT1 is 378h, and LPT2, 278h, but some will use 3BCh as LPT1, 378h as LPT1 and 278h as LPT2. *Life wasn't meant to be easy.*

The assigned devices LPT1, LPT2 & LPT3 should not be a worry to people wishing to interface devices to their PC's. Most of the time the base address is used to interface the port rather than LPT1 etc. However should you want to find the address of LPT1 or any of the **L**ine **P**rin**T**er Devices, you can use a lookup table provided by BIOS. When BIOS assigns addresses to your printer devices, it stores the address at specific locations in memory, so we can find them.

START ADDRESS	FUNCTION
0000-0408	LPT1's Base Address
0000-040A	LPT2's Base Address
0000-040C	LPT3's Base Address
0000-040E	LPT4's Base Address (Note 1)

Table 3.3 - LPT Addresses in the BIOS Data Area

Note 1 : Address 0000:040E in the BIOS Data Area may be used as the Extended Bios Data Area in PS/2 and newer Bioses, and thus this field may be invalid.

The above table, table 3.3, shows the address at which we can find the Printer Port's addresses in the BIOS Data Area. Each address will take up 2 bytes. The following sample program in C, shows how you can read these locations to obtain the addresses of your printer ports.

```
#include <stdio.h>
#include <dos.h>
void main(void)
{
unsigned int far *ptraddr; /* Pointer to location of Port Addresses */
unsigned int address; /* Address of Port */
int a;
ptraddr=(unsigned int far *)0x00000408;
for (a = 0; a < 3; a++)
      {
      address = *ptraddr;
      if (address == 0)
      printf("No port found for LPT%d \n",a+1);
      else
      printf("Address assigned to LPT%d is %Xh\n",a+1,address);
      *ptraddr++;
      }
}
```

3.4 Software Registers - Standard Parallel Port (SPP)

OFFSET	NAME	READ/WRITE	BIT NO	PROPERTIES
BASE +0	DATA PORT	WRITE (Note 1)	Bit 7	Data 7(Pin 9)
			Bit 6	Data 6(Pin 8)
			Bit 5	Data 5(Pin 7)
			Bit 4	Data 4(Pin 6)
			Bit 3	Data 3(Pin 5)
			Bit 2	Data 2(Pin 4)
			Bit 1	Data 1(Pin 3)
			Bit 0	Data 0(Pin 2)

Table 3.4 Data Port

Note 1 : If the Port is bi-directional then Read and Write Operations can be performed on the Data Register.

The base address, usually called the Data Port or Data Register is simply used for outputting data on the Parallel Port's data lines (Pins 2-9). This register is normally a write only port. If you read from the port, you should get the last byte sent. However if your port is bi-directional, you can receive data on this address. See *Bi-directional Ports* for more detail.

BASE +1	STATUS PORT	READ ONLY	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper out
			Bit 4	Select in
			Bit 3	Error
			Bit 2	IRQ
			Bit 1	Reserved
			Bit 0	Reserved

Table 3.5 Status Port

The Status Port (base address + 1) is a read only port. Any data written to this port will be ignored. The Status Port is made up of 5 input lines (Pins 10,11,12,13 & 15), a IRQ status register and two reserved bits. Please note that Bit 7 (Busy) is a active low input. E.g. If bit 7 happens to show a logic 0, this means that there is +5v at pin 11. Likewise with Bit 2. (nIRQ) If this bit shows a '1' then an interrupt has **not** occurred.

BASE +2	CONTROL PORT	READ\WRITE	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable bi-directional port
			Bit 4	Enable IRQ via Ack line
			Bit 3	Select printer
			Bit 2	Initialize printer (Reset)
			Bit 1	Auto linefeed
			Bit 0	Strobe

Table 3.6 Control Port

The Control Port (base address + 2) was intended as a write only port. When a printer is attached to the Parallel Port, four "controls" are used. These are Strobe, Auto Linefeed, Initialize and Select Printer, all of which are inverted except Initialize.

The printer would not send a signal to initialize the computer, nor would it tell the computer to use auto linefeed. However these four outputs can also be used for inputs. If the computer has placed a pin high (e.g. +5v) and your device wanted to take it low, you would effectively short out the port, causing a conflict on that pin. Therefore these lines are "open collector" outputs *(or open drain for CMOS devices).* This means that it has two states. A low state (0v) and a high impedance state (open circuit).

Normally the Printer Card will have internal pull-up resistors, but as you would expect, not all will. Some may just have open collector outputs, while others may even have normal totem pole outputs. In order to make your device work correctly on as many Printer Ports as possible, you can use an external resistor as well. Should you already have an internal resistor, then it will act in Parallel with it, or if you have Totem pole outputs, the resistor will act as a load.

An external 4.7k resistor can be used to pull the pin high. I wouldn't use anything lower, just in case you do have an internal pull up resistor, as the external resistor would act in parallel giving effectively, a lower value pull up resistor. When in high impedance state the pin on the Parallel Port is high (+5v). When in this state, your external device can pull the pin low and have the control port change read a different value. This way the 4 pins of the Control Port can be used for bi-directional data transfer. However the Control Port must be set to xxxx0100 to be able to read data, that is all pins to be +5v at the port so that you can pull it down to GND (logic 0).

Bits 4 & 5 are internal controls. Bit four will enable the IRQ *(See Using the Parallel Ports IRQ)* and Bit 5 will enable the bi-directional port meaning that you can input 8 bits using (DATA0-7). This mode is only possible if your card supports it. Bits 6 & 7 are reserved. Any writes to these two bits will be ignored.

b. SERIAL PORT

3.5 Introduction

The Serial Port is harder to interface than the Parallel Port. In most cases, any device you connect to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. On the software side of things, there are many more registers that you have to attend to than on a Standard Parallel Port. (SPP)

So what are the advantages of using serial data transfer rather than parallel?

1. Serial Cables can be longer than Parallel cables. The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts where as a parallel port transmits a '0' as 0v and a '1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables than they are for parallel.
2. You don't need as many wires than parallel transmission. If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper that running 19 or 25 core cable. However you must take into account the cost of the interfacing at each end.
3. Infra Red devices have proven quite popular recently. You may of seen many electronic diaries and palmtop computers which have infra red capabilities build in. However could you imagine transmitting 8 bits of data at the one time across the room and being able to (from the devices point of view) decipher which bits are which? Therefore serial transmission is used where one bit is sent at a time. IrDA-1 (The first infra red specifications) was capable of 115.2k baud and was interfaced into a UART. The pulse length however was cut down to 3/16th of a RS232 bit length to conserve power considering these devices are mainly used on diaries, laptops and palmtops.
4. Microcontroller's have also proven to be quite popular recently. Many of these have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial Communication reduces the pin count of these MPU's. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use a 8 bit Parallel method (You may also require a Strobe).

3.6 Hardware Properties

Devices which use serial cables for their communication are split into two categories. These are DCE (Data Communications Equipment) and DTE (Data Terminal Equipment.) Data Communications Equipment are devices such as your modem, TA adapter, plotter etc while Data Terminal Equipment is your Computer or Terminal.

The electrical specifications of the serial port is contained in the EIA (Electronics Industry Association) RS232C standard. It states many parameters such as -

1. A "Space" (logic 0) will be between +3 and +25 Volts.
2. A "Mark" (Logic 1) will be between -3 and -25 Volts.
3. The region between +3 and -3 volts is undefined.
4. An open circuit voltage should never exceed 25 volts. (In Reference to GND)
5. A short circuit current should not exceed 500mA. The driver should be able to handle this without damage. (Take note of this one!)

Above is no where near a complete list of the EIA standard. Line Capacitance, Maximum Baud Rates etc are also included. For more information please consult the EIA RS232-E standard. It is interesting to note however, that the RS232C standard specifies a maximum baud rate of 20,000 BPS!, which is rather slow by today's standards. Revised standards, EIA-232D & EIA-232E were released, in 1987 & 1991 respectively.

Serial Ports come in two "sizes", There are the D-Type 25 pin connector and the D-Type 9 pin connector both of which are male on the back of the PC, thus you will require a female connector on your device. Below is a table of pin connections for the 9 pin and 25 pin D-Type connectors.

3.6.1 Serial Pinouts (D25 and D9 Connectors)

D-Type 25 Pin No	D-Type 9Pin No	Abbreviations	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request to send
Pin 5	Pin 8	CTS	Clear to send
Pin 6	Pin 6	DSR	Data send ready
Pin 7	Pin 5	SG	Signal ground
Pin 8	Pin 1	CD	Carrier detect
Pin 20	Pin 4	DTR	Data terminal ready
Pin 22	Pin 9	RI	Ring indicator

Table 3.8 : D Type 9 Pin and D Type 25 Pin Connectors

3.6.2 Pin Functions

Abbreviations	Full Name	Functions
TD	Transmit Data	Serial data output (TXD)
RD	Receive Data	Serial data input (RXD)
DCD	Data carrier detect	When the data detects Carrier from the modem at the other end of the phone line, this Line becomes active.
CTS	Clear to send	This line indicate that modem is ready to exchange data.
DSR	Data send ready	This tells the UART that the modem is ready to establish the link.

DTR	Data terminal ready	This is opposite to DSR. This tells the modem that the UART is ready to link.
RTS	Request to send	This line informs the modem that the UART is ready to exchange data.
RI	Ring indicator	Goes active when the modem detects the ringing signal from the PSTN.

3.7 Null Modems

A Null Modem is used to connect two DTE's together. This is commonly used as a cheap way to network games or to transfer files between computers using Zmodem Protocol, Xmodem Protocol etc. This can also be used with many Microprocessor Development Systems.

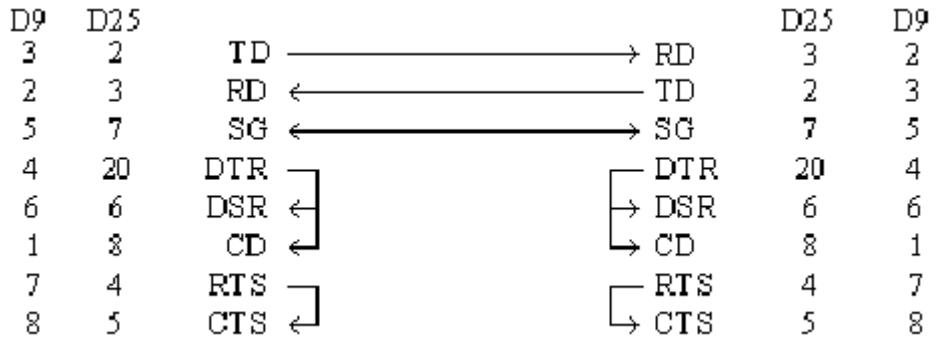


Figure 3.4 : Null Modem Wiring Diagram

Above is my preferred method of wiring a Null Modem. It only requires 3 wires (TD, RD & SG) to be wired straight through thus is more cost effective to use with long cable runs. The theory of operation is reasonably easy. The aim is to make to computer think it is talking to a modem rather than another computer. Any data transmitted from the first computer must be received by the second thus TD is connected to RD. The second computer must have the same set-up thus RD is connected to TD. Signal Ground (SG) must also be connected so both grounds are common to each computer.

The Data Terminal Ready is looped back to Data Set Ready and Carrier Detect on both computers. When the Data Terminal Ready is asserted active, then the Data Set Ready and Carrier Detect immediately become active. At this point the computer thinks the Virtual Modem to which it is connected is ready and has detected the carrier of the other modem.

All left to worry about now is the Request to Send and Clear To Send. As both computers communicate together at the same speed, flow control is not needed thus these two lines are also linked together on each computer. When the computer wishes to send data, it asserts the Request to Send high and as it's hooked together with the Clear to Send, It immediately gets a reply that it is ok to send and does so.

Notice that the ring indicator is not connected to anything of each end. This line is only used to tell the computer that there is a ringing signal on the phone line. As we don't have a modem connected to the phone line this is left disconnected.

3.8 LoopBack Plug

This loopback plug can come in extremely handy when writing Serial / RS232 Communications Programs. It has the receive and transmit lines connected together, so that anything transmitted out of the Serial Port is immediately received by the same port. If you connect this to a Serial Port an load a Terminal Program, anything you type will be immediately displayed on the screen. This can be used with the examples later in this tutorial.

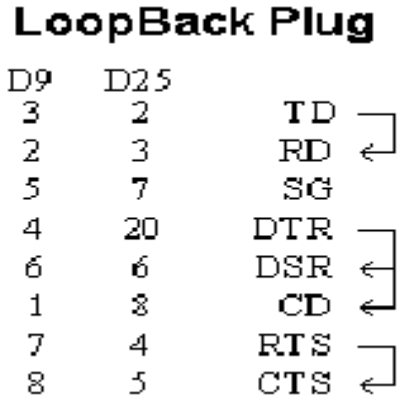


Figure 3.5

3.9 DTE / DCE Speeds

We have already talked briefly about DTE & DCE. A typical Data Terminal Device is a computer and a typical Data Communications Device is a Modem. Often people will talk about DTE to DCE or DCE to DCE speeds. DTE to DCE is the speed between your modem and computer, sometimes referred to as your terminal speed. This should run at faster speeds than the DCE to DCE speed. DCE to DCE is the link between modems, sometimes called the line speed.

Most people today will have 28.8K or 33.6K modems. Therefore we should expect the DCE to DCE speed to be either 28.8K or 33.6K. Considering the high speed of the modem we should expect the DTE to DCE speed to be about 115,200 BPS. (Maximum Speed of the 16550a UART) This is where some people often fall into a trap. The communications program which they use have settings for DCE to DTE speeds. However they see 9.6 KBPS, 14.4 KBPS etc and think it is your modem speed.

Today's Modems should have Data Compression build into them. This is very much like PK-ZIP but the software in your modem compresses and decompresses the data. When set up correctly you can expect compression ratios of 1:4 or even higher. 1 to 4 compression would be typical of a text file. If we were transferring that text file at 28.8K (DCE-DCE), then when the modem compresses it you are actually transferring 115.2 KBPS between computers and thus have a DCE-DTE speed of 115.2 KBPS.

Thus this is why the DCE-DTE should be much higher than your modem's connection speed. Some modem manufacturers quote a maximum compression ratio as 1:8. Lets say for example its on a new 33.6 KBPS modem then we may get a maximum 268,800 BPS transfer between modem and UART. If you only have a 16550a which can do 115,200 BPS tops, then

you would be missing out on a extra bit of performance. Buying a 16C650 should fix your problem with a maximum transfer rate of 230,400 BPS.

However don't abuse your modem if you don't get these rates. These are MAXIMUM compression ratios. In some instances if you try to send a already compressed file, your modem can spend more time trying the compress it, thus you get a transmission speed less than your modem's connection speed. If this occurs try turning off your data compression. This should be fixed on newer modems. Some files compress easier than others thus any file which compresses easier is naturally going to have a higher compression ratio.

3.10 Flow Control

So if our DTE to DCE speed is several times faster than our DCE to DCE speed the PC can send data to your modem at 115,200 BPS. Sooner or later data is going to get lost as buffers overflow, thus flow control is used. Flow control has two basic varieties, Hardware or Software.

Software flow control, sometimes expressed as Xon/Xoff uses two characters Xon and Xoff. Xon is normally indicated by the ASCII 17 character where as the ASCII 19 character is used for Xoff. The modem will only have a small buffer so when the computer fills it up the modem sends a Xoff character to tell the computer to stop sending data. Once the modem has room for more data it then sends a Xon character and the computer sends more data. This type of flow control has the advantage that it doesn't
require any more wires as the characters are sent via the TD/RD lines. However on slow links each character requires 10 bits which can slow communications down.

Hardware flow control is also known as RTS/CTS flow control. It uses two wires in your serial cable rather than extra characters transmitted in your data lines. Thus hardware flow control will not slow down transmission times like Xon-Xoff does. When the computer wishes to send data it takes active the Request to Send line. If the modem has room for this data, then the modem will reply by taking active the Clear to Send line and the computer starts sending data. If the modem does not have the room then it will not send a Clear to Send.

c. PARALLEL TO SERIAL DATA CONVERSION

3.11 The UART (8250 and Compatibles)

UART stands for Universal Asynchronous Receiver / Transmitter. Its the little box of tricks found on your serial card which plays the little games with your modem or other connected devices. Most cards will have the UART's integrated into other chips which may also control your parallel port, games port, floppy or hard disk drives and are typically surface mount devices. The 8250 series, which includes the 16450, 16550, 16650, & 16750 UARTS are the most commonly found type in your PC. Later we will look at other types which can be used in your homemade devices and projects.

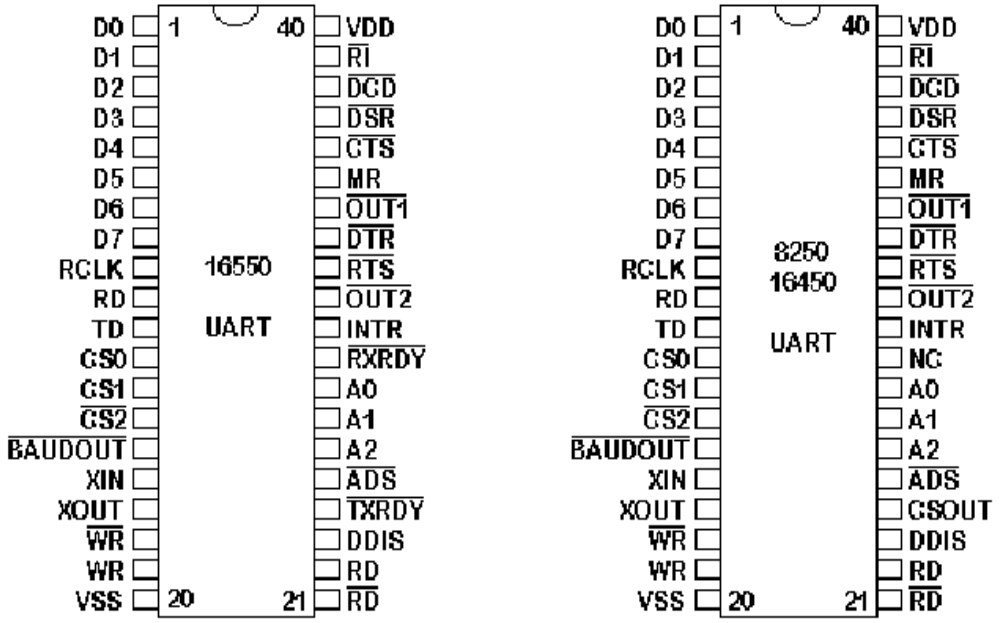


Figure 3.6 : Pin Diagrams for 16550, 16450 & 8250 UARTs

The 16550 is chip compatible with the 8250 & 16450. The only two differences are pins 24 & 29. On the 8250 Pin 24 was chip select out which functioned only as a indicator to if the chip was active or not. Pin 29 was not connected on the 8250/16450 UARTs. The 16550 introduced two new pins in their place. These are Transmit Ready and Receive Ready which can be implemented with DMA (Direct Memory Access). These Pins have two different modes of operation. Mode 0 supports single transfer DMA where as Mode 1 supports Multi-transfer DMA.

Mode 0 is also called the 16450 mode. This mode is selected when the FIFO buffers are disabled via Bit 0 of the FIFO Control Register or When the FIFO buffers are enabled but DMA Mode Select = 0. (Bit 3 of FCR) In this mode RXRDY is active low when at least one character (Byte) is present in the Receiver Buffer. RXRDY will go inactive high when no more characters are left in the Receiver Buffer. TXRDY will be active low when there are no characters in the

Transmit Buffer. It will go inactive highafter the first character / byte is loaded into the Transmit Buffer.

Mode 1 is when the FIFO buffers are active and the DMA Mode Select = 1. In Mode 1, RXRDY will go active low when the trigger level is reached or when 16550 Time Out occurs and will return to inactive state when no more characters are left in the FIFO. TXRDY will be active when no characters are present in the Transmit Buffer and will go inactive when the FIFO Transmit Buffer is completely Full.

All the UARTs pins are TTL compatible. That includes TD, RD, RI, DCD, DSR, CTS, DTR and RTS which all interface into your serial plug, typically a D-type connector. Therefore RS232 Level Converters (which we talk about in detail later) are used. These are commonly the DS1489 Receiver and the DS1488 as the PC has +12 and -12 volt rails which can be used by these devices. The RS232 Converters will convert the TTL signal into RS232 Logic Levels.

The UART requires a Clock to run. If you look at your serial card a common crystal found is either a 1.8432 MHZ or a 18.432 MHZ Crystal. The crystal in connected to the XIN-XOUT pins of the UART using a few extra components which help the crystal to start oscillating. This clock will be used for the Programmable Baud Rate Generator which directly interfaces into the transmit timing circuits but not directly into the receiver timing circuits. For this an external connection mast be made from pin 15 (BaudOut) to pin 9 (Receiver clock in.) Note that the clock signal will be at Baudrate 16.

Pin No.	Name	Notes
Pin 1:8	D0:D7	Data Bus
Pin 9	RCLK	Receiver Clock Input. The frequency of this input should equal the receivers baud rate * 16
Pin 10	RD	Receive Data
Pin 11	TD	Transmit Data
Pin 12	CS0	Chip Select 0 - Active High
Pin 13	CS1	Chip Select 1 - Active High
Pin 14	nCS2	Chip Select 2 - Active Low
Pin 15	nBAUDOUT	Baud Output - Output from Programmable Baud Rate Generator. Frequency = (Baud Rate x 16)
Pin 16	XIN	External Crystal Input - Used for Baud Rate Generator Oscillator
Pin 17	XOUT	External Crystal Output
Pin 18	nWR	Write Line - Inverted
Pin 19	WR	Write Line - Not Inverted
Pin 20	VSS	Connected to Common Ground
Pin 21	RD	Read Line - Inverted
Pin 22	nRD	Read Line - Not Inverted
Pin 23	DDIS	Driver Disable. This pin goes low when CPU is reading from UART. Can be connected to Bus Transceiver in case of high capacity data bus.
Pin 24	nTXRDY	Transmit Ready
Pin 25	nADS	Address Strobe. Used if signals are not stable during read or write cycle
Pin 26	A2	Address Bit 2
Pin 27	A1	Address Bit 1
Pin 28	A0	Address Bit 0

Pin 29	nRXRDY	Receive Ready
Pin 30	INTR	Interrupt Output
Pin 31	nOUT2	User Output 2
Pin 32	nRTS	Request to Send
Pin 33	nDTR	Data Terminal Ready
Pin 34	nOUT1	User Output 1
Pin 35	MR	Master Reset
Pin 36	nCTS	Clear To Send
Pin 37	nDSR	Data Set Ready
Pin 38	nDCD	Data Carrier Detect
Pin 39	nRI	Ring Indicator
Pin 40	VDD	+ 5 Volts

Table 3.9 : Pin Assignments for 16550A UART

3.11.1 Types of UARTS (For PC's)

8250	First UART in this series. It contains no scratch register. The 8250A was an improved version of the 8250 which operates faster on the bus side.
8250A	This UART is faster than the 8250 on the bus side. Looks exactly the same to software than 16450.
8250B	Very similar to that of the 8250 UART.
16450	Used in AT's (Improved bus speed over 8250's). Operates comfortably at 38.4KBPS. Still quite common today.
16550	This was the first generation of buffered UART. It has a 16 byte buffer, however it doesn't work and is replaced with the 16550A.
16550A	Is the most common UART use for high speed communications eg 14.4K & 28.8K Modems. They made sure the FIFO buffers worked on this UART.
16650	Very recent breed of UART. Contains a 32 byte FIFO, Programmable X-On / XOff characters and supports power management.
16750	Produced by Texas Instruments. Contains a 64 byte FIFO.

3.11.2 Serial Port's Registers (PC's) Port Addresses & IRQ's

NAME	ADDRESS	IRQ
COM 1	3F8	4
COM 2	2F8	3
COM 3	3E8	4
COM 4	2E8	3

Table 3.10 : Standard Port Addresses

Above is the standard port addresses. These should work for most P.C's. If you just happen to be lucky enough to own a IBM P/S2 which has a micro-channel bus, then expect a different set of addresses and IRQ's. Just like the LPT ports, the base addresses for the COM ports can be read from the BIOS Data Area.

START ADDRESS	FUNCTION
0000 : 0400	COM 1's Base Address
0000 : 0402	COM 2's Base Address
0000 : 0404	COM 3's Base Address
0000 : 0406	COM 4's Base Address

Table 3.11 - COM Port Addresses in the BIOS Data Area;

The above table shows the address at which we can find the Communications (COM) ports addresses in the BIOS Data Area. Each address will take up 2 bytes. The following sample program in C, shows how you can read these locations to obtain the addresses of your communications ports.

```
#include <stdio.h>
#include <dos.h>
void main(void)
{
unsigned int far *ptraddr; /* Pointer to location of Port Addresses */
unsigned int address; /* Address of Port */
int a;
ptraddr=(unsigned int far *)0x00000400;
for (a = 0; a < 4; a++)
{
address = *ptraddr;
if (address == 0)
printf("No port found for COM%d \n",a+1);
else
printf("Address assigned to COM%d is %Xh\n",a+1,address);
*ptraddr++;
}
}
```

3.11.3 Table of Registers

Base Address	DLAB	Read/Write	Abr.	Register Name
+ 0	=0	Write	-	Transmitter Holding Buffer
	=0	Read	-	Receiver Buffer
	=1	Read/Write	-	Divisor Latch Low Byte
+ 1	=0	Read/Write	IER	Interrupt Enable Register
	=1	Read/Write	-	Divisor Latch High Byte
+ 2	-	Read	IIR	Interrupt Identification Register
	-	Write	FCR	FIFO Control Register
+ 3	-	Read/Write	LCR	Line Control Register
+ 4	-	Read/Write	MCR	Modem Control Register
+ 5	-	Read	LSR	Line Status Register
+ 6	-	Read	MSR	Modem Status Register
+ 7	-	Read/Write	-	Scratch Register

Table 3.12 : Table of Registers

3.11.4 DLAB ?

You will have noticed in the table of registers that there is a DLAB column. When DLAB is set to '0' or '1' some of the registers change. This is how the UART is able to have 12 registers (including the scratch register) through only 8 port addresses. DLAB stands for Divisor Latch Access Bit. When DLAB is set to '1' via the line control register, two registers become available from which you can set your speed of communications measured in bits per second.

The UART will have a crystal which should oscillate around 1.8432 MHZ. The UART incorporates a divide by 16 counter which simply divides the incoming clock signal by 16. Assuming we had the 1.8432 MHZ clock signal, that would leave us with a maximum, 115,200 hertz signal making the UART capable of transmitting and receiving at 115,200 Bits Per Second (BPS). That would be fine for some of the faster modems and devices which can handle that speed, but others just wouldn't communicate at all. Therefore the UART is fitted with a Programmable Baud Rate Generator which is
controlled by two registers.
Lets say for example we only wanted to communicate at 2400 BPS. We worked out that we would have to divide 115,200 by 48 to get a workable 2400 Hertz Clock. The "Divisor", in this case 48, is stored in the two registers controlled by the "Divisor Latch Access Bit". This divisor

can be any number which can be stored in 16 bits (ie 0 to 65535). The UART only has a 8 bit data bus, thus this is where the two registers are used. The first register (Base + 0) when DLAB = 1 stores the "Divisor latch low byte" where as the second register (base + 1 when DLAB = 1) stores the "Divisor latch high byte."

Below is a table of some more common speeds and their divisor latch high bytes & low bytes. Note that all the divisors are shown in Hexadecimal.

Speed (BPS)	Divisor (Dec)	Divisor Latch High Byte	Divisor Latch Low Byte
50	2304	09h	00h
300	384	01h	80h
600	192	00h	C0h
2400	48	00h	30h
4800	24	00h	18h
9600	12	00h	0Ch
19200	6	00h	06h
38400	3	00h	03h
57600	2	00h	02h
115200	1	00h	01h

Table 3.13 : Table of Commonly Used Baudrate Divisors

3.11.5 Interrupt Enable Register (IER)

Bits	Notes
Bit 7	Reserved
Bit 6	Reserved
Bit 5	Enable low power mode (16750)
Bit 4	Enables sleep mode(16750)
Bit 3	Enable modem status interrupt
Bit 2	Enable receive line status interrupt
Bit 1	Enable transmit holding register empty interrupt
Bit 0	Enable received data available interrupt

Table 3.14 : Interrupt Enable Register

The Interrupt Enable Register could possibly be one of the easiest registers on a UART to understand. Setting Bit 0 high enables the Received Data Available Interrupt which generates an interrupt when the receiving register/FIFO contains data to be read by the CPU.

Bit 1 enables Transmit Holding Register Empty Interrupt. This interrupts the CPU when the transmitter buffer is empty. Bit 2 enables the receiver line status interrupt. The UART will interrupt when the receiver line status changes. Likewise for bit 3 which enables the modem

status interrupt. Bits 4 to 7 are the easy ones. They are simply reserved. (If only everything was that easy!)

3.11.6 Interrupt Identification Register (IIR)

Bit	Notes		
Bits 6 : 7	Bit 6	Bit 7	
	0	0	No FIFO
	0	1	FIFO Enabled but Unusable
	1	1	FIFO Enabled
Bit 5	64 Byte Fifo Enabled (16750 only)		
Bit 4	Reserved		
Bit 3	0	Reserved on 8250, 16450	
	1	16550 Time-out Interrupt Pending	
Bits 1 : 2	Bit 2	Bit 1	
	0	0	Modem Status Interrupt
	0	1	Transmitter Holding Register Empty Interrupt
	1	0	Received Data Available Interrupt
	1	1	Receiver Line Status Interrupt
Bit 0	0	Interrupt Pending	
	1	No Interrupt Pending	

Table 3.15 : Interrupt Identification Register

The interrupt identification register is a read only register. Bits 6 and 7 give status on the FIFO Buffer. When both bits are '0' no FIFO buffers are active. This should be the only result you will get from a 8250 or 16450. If bit 7 is active but bit 6 is not active then the UART has it's buffers enabled but are unusable. This occurs on the 16550 UART where a bug in the FIFO buffer made the FIFO's unusable. If both bits are '1' then the FIFO buffers are enabled and fully operational.

Bits 4 and 5 are reserved. Bit 3 shows the status of the time-out interrupt on a 16550 or higher. Lets jump to Bit 0 which shows whether an interrupt has occurred. If an interrupt has occurred it's status will shown by bits 1 and 2. These interrupts work on a priority status. The Line Status Interrupt has the highest Priority, followed by the Data Available Interrupt, then the

Transmit Register Empty Interrupt and then the Modem Status Interrupt which has the lowest priority.

3.11.7 First In / First Out Control Register (FCR)

Bit	Notes		
Bits 6 : 7	Bit 7	Bit 6	Interrupt Trigger Level
	0	0	1 Byte
	0	1	4 Bytes
	1	0	8 Bytes
	1	1	14 Bytes
Bit 5	Enable 64 Byte FIFO (16750 only)		
Bit 4	Reserved		
Bit 3	DMA Mode Select. Change status of RXRDY & TXRDY pins from mode 1 to mode 2.		
Bit 2	Clear Transmit FIFO		
Bit 1	Clear Receive FIFO		
Bit 0	Enable FIFO's		

Table 3.16 : FIFO Control Register

The FIFO register is a write only register. This register is used to control the FIFO (First In / First Out) buffers which are found on 16550's and higher.

Bit 0 enables the operation of the receive and transmit FIFO's. Writing a '0' to this bit will disable the operation of transmit and receive FIFO's, thus you will loose all data stored in these FIFO buffers.

Bit's 1 and 2 control the clearing of the transmit or receive FIFO's. Bit 1 is responsible for the receive buffer while bit 2 is responsible for the transmit buffer. Setting these bits to 1 will only clear the contents of the FIFO and will not affect the shift registers. These two bits are self resetting, thus you don't need to set the bits to '0' when finished.

Bit 3 enables the DMA mode select which is found on 16550 UARTs and higher. More on this later. Bits 4 and 5 are those easy type again, Reserved.

Bits 6 and 7 are used to set the triggering level on the Receive FIFO. For example if bit 7 was set to '1' and bit 6 was set to '0' then the trigger level is set to 8 bytes. When there is 8 bytes of data in the receive FIFO then the Received Data Available interrupt is set. See (IIR)

3.11.8 Line Control Register (LCR)

The Line Control register sets the basic parameters for communication. Bit 7 is the Divisor Latch Access Bit or DLAB for short. We have already talked about what it does. (See DLAB?) Bit 6 Sets break enable. When active, the TD line goes into "Spacing" state which causes a break in the receiving UART.Setting this bit to '0' Disables the Break.Bits 3,4 and 5 select parity. If you study the 3 bits, you will find that bit 3 controls parity. That is, if it is set to '0' then no parity is used, but if it is set to '1' then parity is used. Jumping to bit 5, we can see that it controls sticky parity. Sticky parity is simply when the parity bit is always transmitted and checked as a '1' or '0'. This has very little success in checking for errors as if the first 4 bits contain errors but the sticky parity bit contains the appropriately set bit, then a parity error will not result. Sticky high parity is the use of a '1' for the parity bit, while the opposite, sticky low parity is the use of a '0' for the parity bit.If bit 5 controls sticky parity, then turning this bit off must produce normal parity provided bit 3 is still set to '1'. Odd parity is when the parity bit is transmitted as a '1' or '0' so that there is a odd number of 1's. Even parity must then be the parity bit produces and even number of 1's. This provides better error checking but still is not perfect, thus CRC-32 is often used for software error correction. If one bit happens to be inverted with even or odd parity set, then a parity error will occur, however if two bits are flipped in such a way that it produces the correct parity bit then an parity error will no occur.

Bit 2 sets the length of the stop bits. Setting this bit to '0' will produce one stop bit, however setting it to '1' will produce either 1.5 or 2 stop bits depending upon the word length. Note that the receiver only checks the first stop bit.Bits 0 and 1 set the word length. This should be pretty straight forward. A word length of 8 bits is most commonly used today.

Bit	Notes			
Bit 7	1	Divisor Latch Access Bit		
	0	Access to Receiver buffer, Transmitter buffer & Interrupt Enable Register		
Bit 6	Set Break Enable			
Bits 3 : 5	Bit 5	Bit 4	Bit 3	Parity Select
	X	X	0	No Parity
	0	0	1	Odd Parity
	0	1	1	Even Parity
	1	0	1	High Parity (Sticky)
	1	1	1	Low Parity (Sticky)

Bit 2	Length of Stop Bit		
	0	One Stop Bit	
	1	2 Stop bits for words of length 6,7 or 8 bits or 1.5 Stop Bits for Word lengths of 5 bits.	
Bits 0 : 1	Bit 1	Bit 0	Word Length
	0	0	5 Bits
	0	1	6 Bits
	1	0	7 Bits
	1	1	8 Bits

Table 3.17 : Line Control Register

3.11.9 Modem Control Register (MCR)

Bit	Notes
Bit 7	Reserved
Bit 6	Reserved
Bit 5	Autoflow Control Enabled (16750 only)
Bit 4	LoopBack Mode
Bit 3	Aux Output 2
Bit 2	Aux Output 1
Bit 1	Force Request to Send
Bit 0	Force Data Terminal Ready

Table 3.18 : Modem Control Register

The Modem Control Register is a Read/Write Register. Bits 5,6 and 7 are reserved. Bit 4 activates the loopback mode. In Loopback mode the transmitter serial output is placed into marking state. The receiver serial input is disconnected. The transmitter out is looped back to the receiver in. DSR, CTS, RI & DCD are disconnected. DTR, RTS, OUT1 & OUT2 are connected to the modem control inputs. The modem control output pins are then place in an inactive state. In this mode any data which is placed in the transmitter registers for output is received by the receiver circuitry on the same chip and is available at the receiver buffer. This can be used to test the UARTs operation.

Aux Output 2 maybe connected to external circuitry which controls the UART-CPU interrupt process. Aux Output 1 is normally disconnected, but on some cards is used to switch between a 1.8432MHZ crystal to a 4MHZ crystal which is used for MIDI. Bits 0 and 1 simply control their relevant data lines. For example setting bit 1 to '1' makes the request to send line active.

3.11.10 Line Status Register (LSR)

The line status register is a read only register. Bit 7 is the error in received FIFO bit. This bit is high when at least one break, parity or framing error has occurred on a byte which is contained in the FIFO.When bit 6 is set, both the transmitter holding register and the shift register are empty. The UART's holding register holds the next byte of data to be sent in parallel fashion. The shift register is used to convert the byte to serial, so that it can be transmitted over one line. When bit 5 is set, only the transmitter holding register is empty. So what's the difference between the two? When bit 6, the transmitter holding and shift registers are empty, no serial conversions are taking place so there should be no activity on the transmit data line. When

bit 5 is set, the transmitter holding register is empty, thus another byte can be sent to the data port, but a serial conversion using the shift register may be taking place.

Bit	Notes
Bit 7	Error in Received FIFO
Bit 6	Empty Data Holding Registers
Bit 5	Empty Transmitter Holding Register
Bit 4	Break Interrupt
Bit 3	Framing Error
Bit 2	Parity Error
Bit 1	Overrun Error
Bit 0	Data Ready

Table 3.19 : Line Status Register

The break interrupt (Bit 4) occurs when the received data line is held in a logic state '0' (Space) for more than the time it takes to send a full word. That includes the time for the start bit, data bits, parity bits and stop bits.

A framing error (Bit 3) occurs when the last bit is not a stop bit. This may occur due to a timing error. You will most commonly encounter a framing error when using a null modem linking two computers or a protocol analyzer when the speed at which the data is being sent is different to that of what you have the UART set to receive it at.

A overrun error normally occurs when your program can't read from the port fast enough. If you don't get an incoming byte out of the register fast enough, and another byte just happens to be received, then the last byte will be lost and a overrun error will result.

Bit 0 shows data ready, which means that a byte has been received by the UART and is at the receiver buffer ready to be read.

3.11.11 Modem Status Register (MSR)

Bit	Notes
Bit 7	Carrier Detect
Bit 6	Ring Indicator
Bit 5	Data Set Ready
Bit 4	Clear To Send
Bit 3	Delta Data Carrier Detect
Bit 2	Trailing Edge Ring Indicator
Bit 1	Delta Data Set Ready
Bit 0	Delta Clear to Send

Table 3.20 : Modem Status Register

Bit 0 of the modem status register shows delta clear to send, delta meaning a change in, thus delta clear to send means that there was a change in the clear to send line, since the last read of this register. This is the same for bits 1 and 3. Bit 1 shows a change in the Data Set Ready line where as Bit 3 shows a change in the Data Carrier Detect line. Bit 2 is the Trailing Edge Ring Indicator which indicates that there was a transformation from low to high state on the Ring Indicator line.

Bits 4 to 7 show the current state of the data lines when read. Bit 7 shows Carrier Detect, Bit 6 shows Ring Indicator, Bit 5 shows Data Set Ready & Bit 4 shows the status of the Clear To Send line.

3.11.12 Scratch Register

The scratch register is not used for communications but rather used as a place to leave a byte of data. The only real use it has is to determine whether the UART is a 8250/8250B or a 8250A/16450 and even that is not very practical today as the 8250/8250B was never designed for AT's and can't hack the bus speed.

3.12 Interfacing Devices to RS-232 Ports

3.12.1 RS-232 Waveforms

So far we have introduced RS-232 Communications in relation to the PC. RS-232 communication is asynchronous. That is a clock signal is not sent with the data. Each word is synchronized using it's start bit, and an internal clock on each side, keeps tabs on the timing.

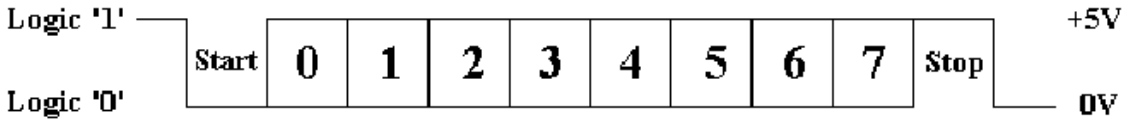


Figure 3.7 : TTL/CMOS Serial Logic Waveform

The diagram above, shows the expected waveform from the UART when using the common 8N1 format. 8N1 signifies 8 Data bits, No Parity and 1 Stop Bit. The RS-232 line, when idle is in the Mark State (Logic 1). A transmission starts with a start bit which is (Logic 0). Then each bit is sent down the line, one at a time. The LSB (Least Significant Bit) is sent first. A Stop Bit (Logic 1) is then appended to the signal to make up the transmission.

The diagram, shows the next bit after the Stop Bit to be Logic 0. This must mean another word is following, and this is it's Start Bit. If there is no more data coming then the receive line will stay in it's idle state(logic 1). We have encountered something called a "Break" Signal. This is when the data line is held in a Logic 0 state for a time long enough to send an entire word. Therefore if you don't put the line back into an idle state, then the receiving end will interpret this as a break signal.

The data sent using this method, is said to be *framed*. That is the data is *framed* between a Start and Stop Bit. Should the Stop Bit be received as a Logic 0, then a framing error will occur. This is common, when both sides are communicating at different speeds.

The above diagram is only relevant for the signal immediately at the UART. RS-232 logic levels uses +3 to +25 volts to signify a "Space" (Logic 0) and -3 to -25 volts for a "Mark" (logic 1). Any voltage in between these regions (ie between +3 and -3 Volts) is undefined. Therefore this signal is put through a "RS-232 Level Converter". This is the signal present on the RS-232 Port of your computer, shown below.

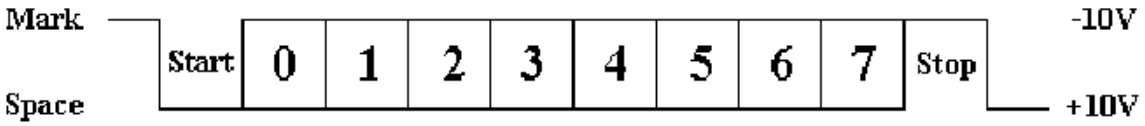


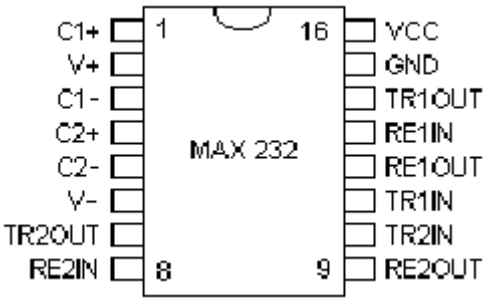
Figure 3.8 : RS-232 Logic Waveform

The above waveform applies to the Transmit and Receive lines on the RS-232 port. These lines carry serial data, hence the name Serial Port. There are other lines on the RS-232 port which, in essence are *Parallel* lines. These lines (RTS, CTS, DCD, DSR, DTR, RTS and RI) are also at RS-232 Logic Levels.

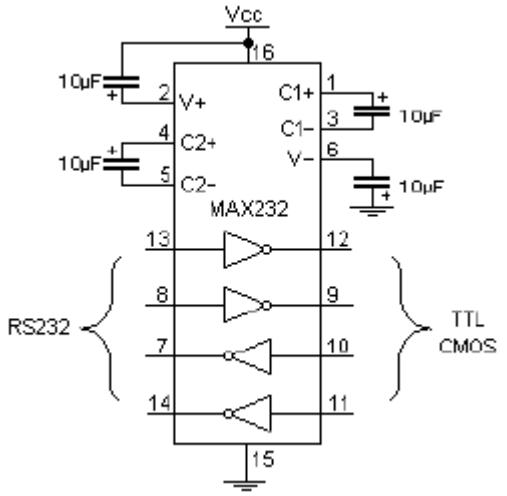
3.12.2 RS-232 Level Converter

Almost all digital devices which we use require either TTL or CMOS logic levels. Therefore the first step to connecting a device to the RS-232 port is to transform the RS-232 levels back into 0 and 5 Volts. As we have already covered, this is done by RS-232 Level Converters.

Two common RS-232 Level Converters are the 1488 RS-232 Driver and the 1489 RS-232 Receiver. Each package contains 4 inverters of the one type, either Drivers or Receivers. The driver requires two supply rails, +7.5 to +15v and -7.5 to -15v. As you could imagine this may pose a problem in many instances where only a single supply of +5V is present. However the advantages of these I.C's are they are cheap.



(Figure 3.9) Pinouts for the MAX-232, RS-232 Driver/Receiver.



(Figure 3.10) Typical MAX-232 Circuit.

Another device is the MAX-232. It includes a Charge Pump, which generates +10V and -10V from a single 5v supply. This I.C. also includes two receivers and two transmitters in the same package. This is handy in many cases when you only want to use the Transmit and Receive data Lines. You don't need to use two chips, one for the receive line and one for the transmit. However all this convenience comes at a price, but compared with the price of designing a new power supply it is very cheap.

There are also many variations of these devices. The large value of capacitors are not only bulky, but also expensive. Therefore other devices are available which use smaller capacitors and even some with inbuilt capacitors. *(Note : Some MAX-232's can use 1 micro farad Capacitors).* However the MAX-232 is the most common, and thus we will use this RS-232 Level Converter in our examples.

3.12.3 Making use of the Serial Format

In order to do anything useful with our Serially transmitted data, we must convert it back to Parallel. *(You could connect an LED to the serial port and watch it flash if you really want too, but it's not extremely useful).* This in the past has been done with the use of UART's. However with the popularity of cheap Microcontroller's, these can be more suited to many applications. We will look into the advantages and disadvantages of each method.

3.13 8250 and Compatible UARTs

We have already looked at one type of UART, the 8250 and compatibles found in your PC. These devices have configuration registers accessible via the data and address buses which have to be initialized before use. This is not a problem if your device which you are building uses a Microprocessor.

However if you are making a stand alone device, how are you going to initialize it? Most Microprocessors / Microcontrollers these days can be brought with build-in Serial Communication Interfaces (SCI). Therefore there is little need to connect a 40 pin 16550 to, for example a 68HC11 when you can buy one built in. If you are still in love with the Z-80 or 8086 then an 16550 may be option!

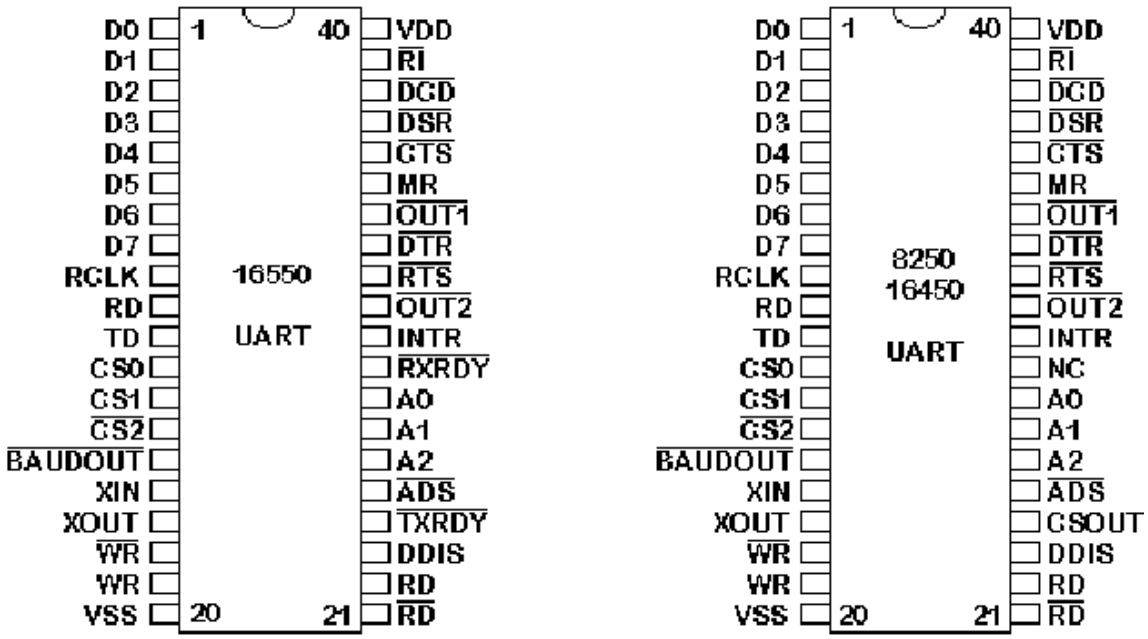


Figure 3.11 : Pin Diagrams for 16550, 16450 & 8250 UARTs

3.14 Microcontrollers

It is also possible to use microcontrollers to transmit and receive Serial data. As we have already covered, some of these MCU's (Micro Controller Units) have built in UART's among other things Microcontroller and thus reducing the chip count and the cost of the project.

It only supports slow transmission speeds, commonly 2400, 9600 or maybe even 19,200 BPS if you are lucky. The other disadvantage is that it's really only effective in half duplex mode. That is, it can only communicate in one direction at any one given time. However in many applications this is not a problem.

As there are many different types of Micro-Controllers all with their different instruction sets, it is very hard to give examples here which will suit everyone.

3.14.1An Introduction to MicroControllers & Embedded Systems

\What is a Controller?

A machine to control various kinds of devices at a time or independently

What is a MicroController?

As name implies a controller fabricated in a small scale to control the devices at hand simultaneously or independently

Embedded Systems

Most of the MicroControllers have built in ROM; so code is rather “embedded” in it. MicroProcessors need external RAM and ROM for a complete system; while most of the MicroControllers have built in RAM and ROMs (Nowadays EEPROMs)

General purpose Vs Dedicated Machine

A MicroProcessor is a general purpose machine while a MicroController is a dedicated machine.

“No Direct Talk” Vs “Direct Talk” (Interface)

MicroProcessors cannot directly “talk” or in other words interfaced to outside world; they need chips to do so e.g. 8255 for Parallel Communication and 8251 for Serial Communication (for Intel Systems). MicroController have dedicated ports for these interfaces e.g. Port 1 and RxD & TxD of 89C51

Comparisons of various MicroControllers in MCS51 Series

Part Number	On-Chip Code Memory	On-Chip Data Memory	No of Timers
xx31	0K ROM	128 bytes	2
xx51	4K ROM	128 bytes	2
xx32	0K ROM	256 bytes	3
xx52	8K ROM	256 bytes	3

3.14.2 AT89C51 Basic Features

- 4 K Bytes ROM
- 128 Bytes RAM
- Four 8-bit I/O Ports
- Two 16 Bit Timers
- Serial Interface
- 64 K External Code Memory Space
- 64 K External Data Memory Space
- Boolean processor (operates on single bits)
- 210 Bit Addressable Locations
- 4 Microseconds Multiply / Divide

3.14.2.1 A Brief Description of Pinouts of AT89C51

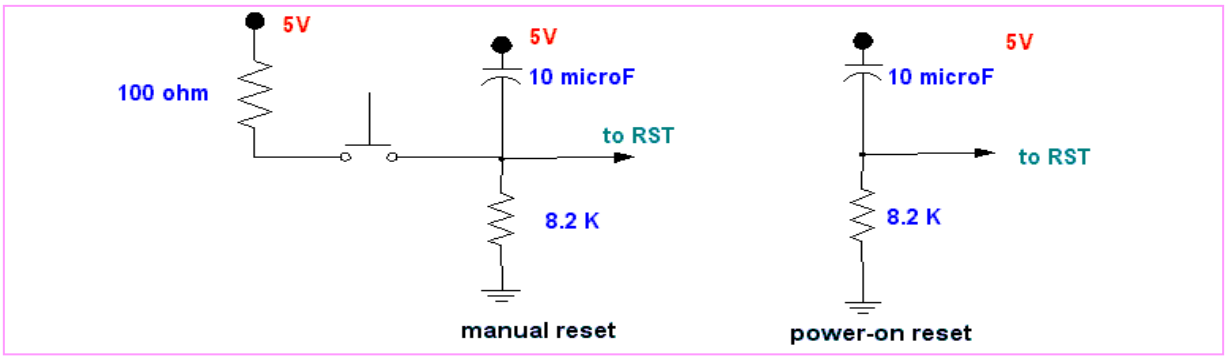
- Pins 1-8 :

Pins 1 through 8 are the pins of Port 1. Port 1 is a dedicated I/O port; so these pins are available for interfacing external devices as required. No alternate function is assigned to these pins.

- Pin 9 :

Pin Number 9 is the system RESET (RST) of CPU of AT89C51. AT89C51 is reset by holding RST high for at least two machine cycles and then returning it low. The Reset may be manually activated using a switch, or may be activated upon power-up using RC network. After a system reset, Program Counter is loaded with 0000H. When RST returns low, program execution begins at the first location in code memory at address 0000H. The contents of on-chip RAM are *not* affected by a reset operation.

Below are the RC networks connected with RST pin:



- Pins 10-17 :

Pins' numbers 10 through 17 constitute Port 3 which is a dual-purpose port. As well as general purpose I/O, these pins are multifunctional with each having an alternate purpose related to special features of C51.

These features along with pins are summarized in the coming table

Pin #	Bit #	Symbol	Bit Add	Alternate Function
10	P3.0	RxD	B0 H	Receive data for Serial Port
11	P3.1	TxD	B1 H	Transmit data for Serial Port
12	P3.2	INT0#	B2 H	External Interrupt 0
13	P3.3	INT1#	B3 H	External Interrupt 1
14	P3.4	T0	B4 H	Timer/Counter 0 external input
15	P3.5	T1	B5 H	Timer/Counter 1 external input
16	P3.6	WR#	B6 H	External Memory write strobe
17	P3.7	RD#	B7 H	External Memory read strobe

- Pins 18-19 :

Pins' numbers 18 and 19 comprise the inputs of crystal to be connected to the on-chip oscillator of AT89C51. Two Stabilizing capacitors of 30 pF each are also required.

- Pin 20 :

It is the common ground of 89C51 and accompanying networks.

- Pins 21-28 :

Pins 21 through 28 are of Port 2.

Port 2 is a also a dual purpose port. It can serve as a general purpose I/O port or as the high byte of the address bus for designs with external code memory or more than 128 bytes of data memory.

- Pin 29 and Pin 31 :

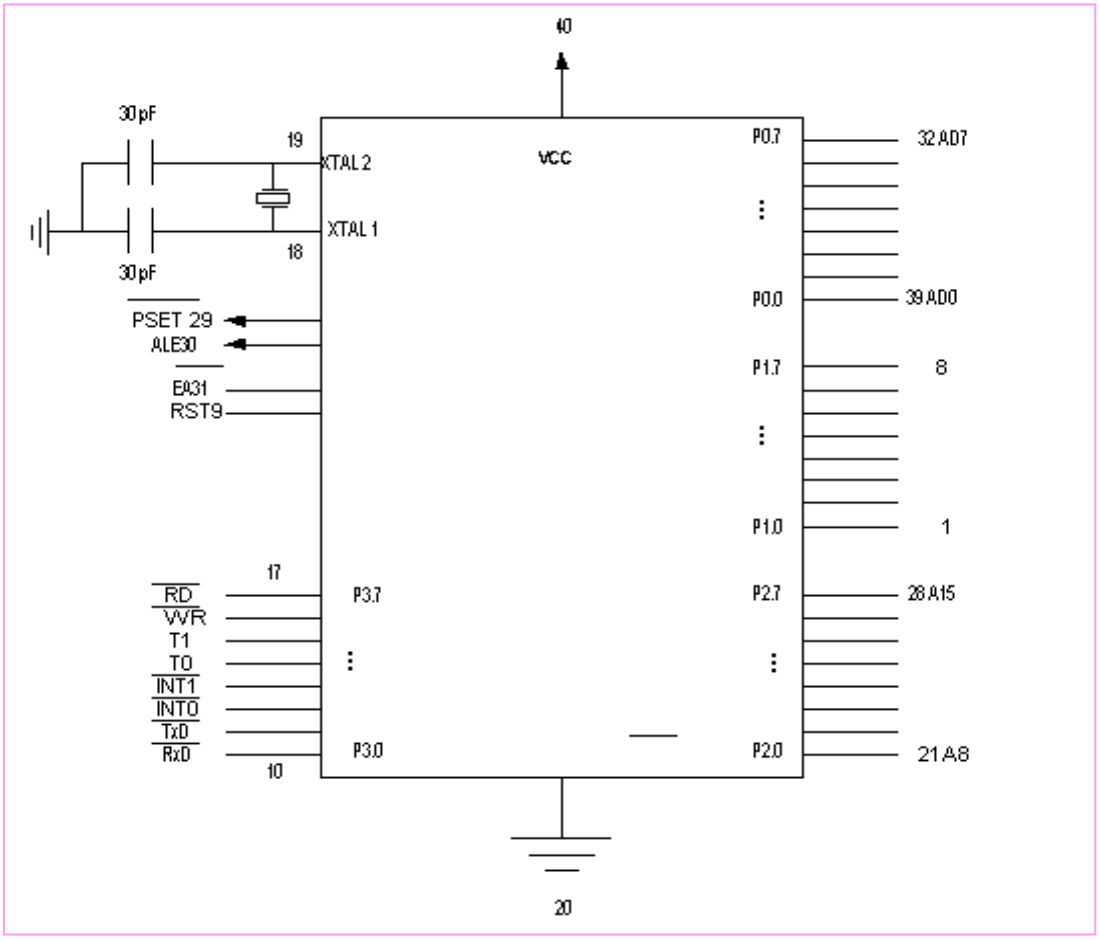
These pins are used in conjunction with external code memory being used or else. On Pin number 29 is a control signal PSEN# (Program Store Enable) that enables external code (Program) memory. It is usually connected to an EEPROMs Output Enable

(OE#) pin to permit reading of program bytes. Pin 31 i.e. EA# (External Access) is either tied high (+5V) or low (ground). If high, the C51 executes programs from internal ROM otherwise from external code memory (and then PSEN# comes into play).

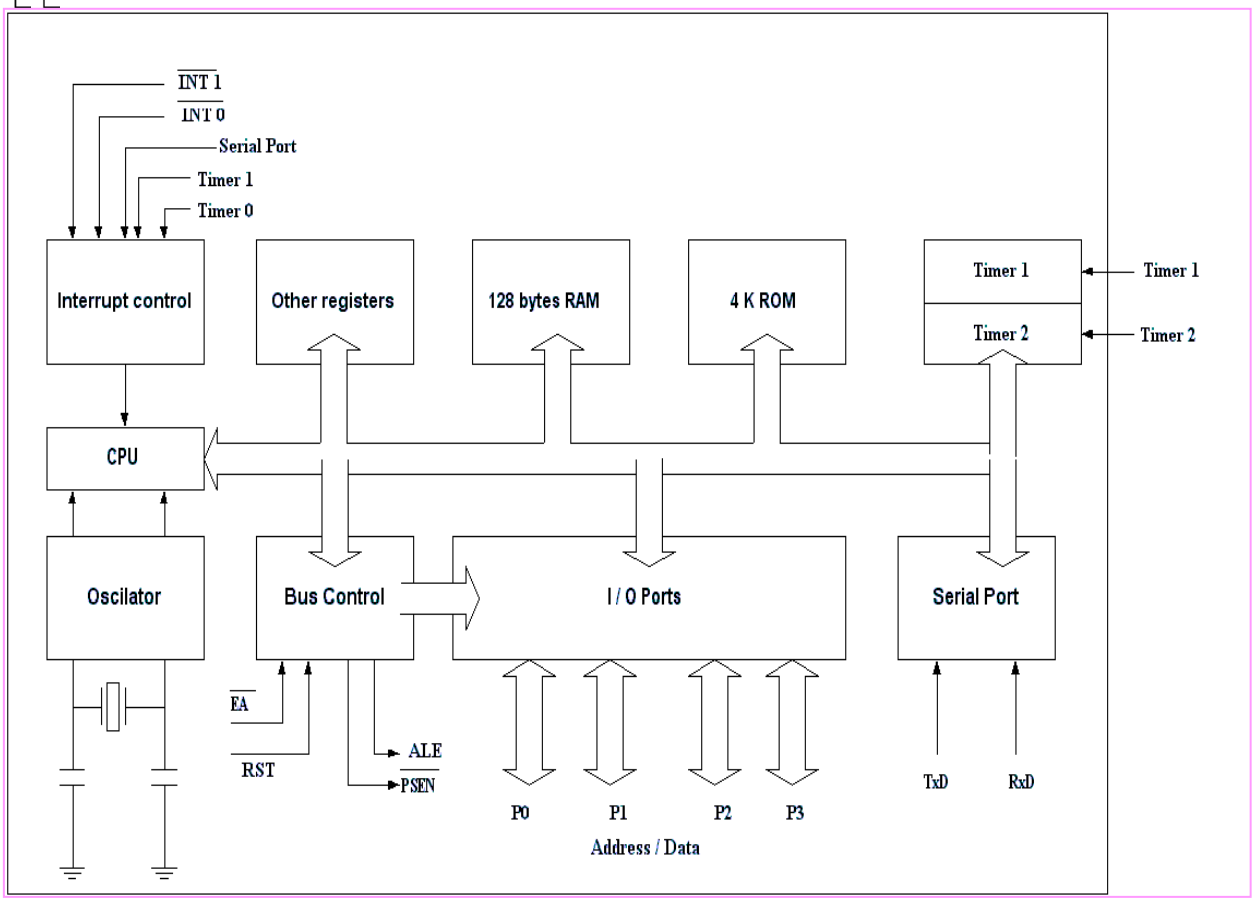
- Pins 32 - 39 and Pin Number 30:

Pins 32 through 39 make up Port 0. Port 0 ,in addition to being used as an I/O port, has the capacity to act as multiplexed data and address bus. The discrimination of data and address is provided through ALE (Address Latch Enable) which is Pin number 30.

3.14.2.2 AT89C51 Pinouts Diagram



3.14.3 C51 Internal Architecture



3.14.4 A Brief Description of SFRs of C51

- Program Control Register … PSW
  (Program Status Word)
- General Purpose Registers … ACC (ACCumulator) and B Registers
- Ports' Registers … P0, P1, P2 and P3
- Data Flow Register … DPH and DPL or DPTR (the only 16 bit register)
  (Data PoinTeR
  … addressable as 16 bit)
- Stack Operation Register … SP (Stack Pointer)
- Power Control Register … PCON (Power CONtrol)

- Timer/Counter Registers … TCON (Timer CONtrol) ,
  TMOD (Timer MODe),
  Timer #0 Registers TL0 & TH0
  Timer #1 Registers TL1 & TH1

- Serial Interface Registers … SCON (Serial CONtrol)
  SBUF (Serial BUFfer)
- Interrupt System Registers … IE (Interrupt Enable)
  IP (Interrupt Priority

3.14.4.1 PSW (Program Status Word)

PSW Bit No	Symbol	Address	Meaning
PSW.0	P	D0.H	Even parity flag
PSW.1	-	D1.H	Reserved flag
PSW.2	0V	D2.H	Overflow flag
PSW.3	RS0	D3.H	Register bank select
PSW.4	RS1	D4.H	Register bank select
PSW.5	F0	D5.H	Flag 0 for user apply
PSW.6	AC	D6.H	Auxiliary carry flag
PSW.7	CY	D7.H	Carry flag

3.14.4.2 ACC (ACCumulator) & B Registers

ACC (ACCumulator) and B Registers are general purpose Registers; their combined use lies mostly in Arithmetic Instructions while ACC being used in most of the other instructions (like Logic and Program Flow Control Instructions) of C51 Instruction Set. Both registers are Bit-Addressable. ACC is at address E0H while B Register is at address F0H.

3.14.4.3 Ports' Registers…P0, P1, P2 and P3

Ports' Registers P0 for Port #0; P1 for Port #1; P2 for Port #2 and P3 for Port #3; are used in accordance with the functionality used in these ports. Writing data to any Port Registers causes an immediate transfer of the data to the respective port while reading data from any Port Register is analogues to reading the data from the respective physical port. All of the four registers are Bit-Addressable. P0 is at 80H, P1 at 90H, P2 at A0H and P3 at B0H.

3.14.4.4 DPTR (Data PoinTeR) Register

DPTR is the only Register in AT89C51 which is accessible as 16 bit register. It is also byte addressable as DPL (Data Pointer Low Byte) and DPH (Data Pointer High Byte). DPTR is

mostly used in addressing blocks of data I.e. as a pointer for a large block of data in Instructions like MOVX. DPTR is Not-Bit-

3.14.4.5 SP (Stack Pointer) Register

Pushing data onto the stack increments the SP. Likewise, popping data from the stack, decrements the SP. If the application software does not re-initialize the SP, then register bank 1 (and perhaps 2 and 3) is not available, since this area of internal RAM is in stack.

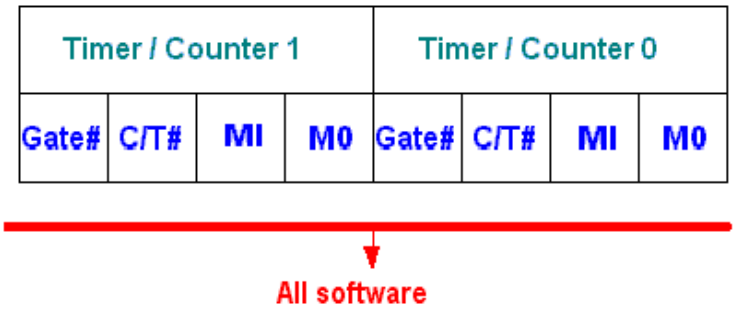
3.14.4.6 PCON (Power CONtrol) Register

PCON Bit No	Symbol	Meaning
PCON.0	IDL	Idle mode ,Set to active idle mode
PCON.1	PD	Power down mode
PCON.2	GF0	General purpose Flag bit 0
PCON.3	GF1	General purpose Flag bit 1
PCON.4	-	Not defined
PCON.5	-	Not defined
PCON.6	-	Not defined
PCON.7	SMOD	Baud rate bit, when set baud rate is
-	-	doubled in Serial Port Modes 1,2 or 3.

3.14.4.7 Timer/Counter Operation & its Registers

The Data Registers for the two timers are TL0 and TH0 for Timer #1 while TL1 and TH1 for the other one. TMOD and TCON control their operation and modes. There are four modes of operation of the two timers set by TMOD.

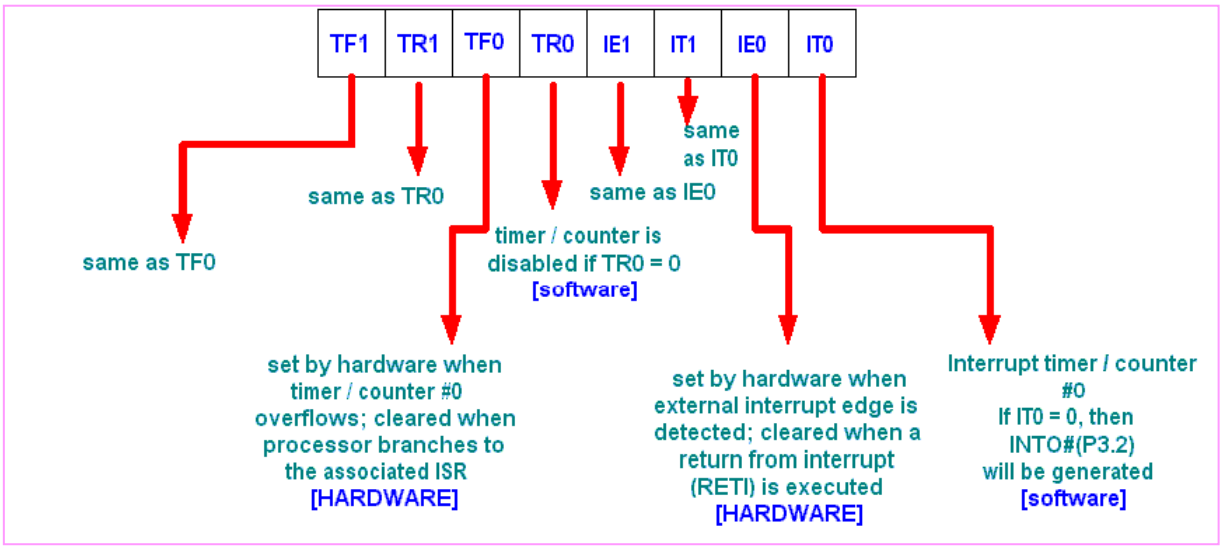
A description of TMOD and TCON follows :

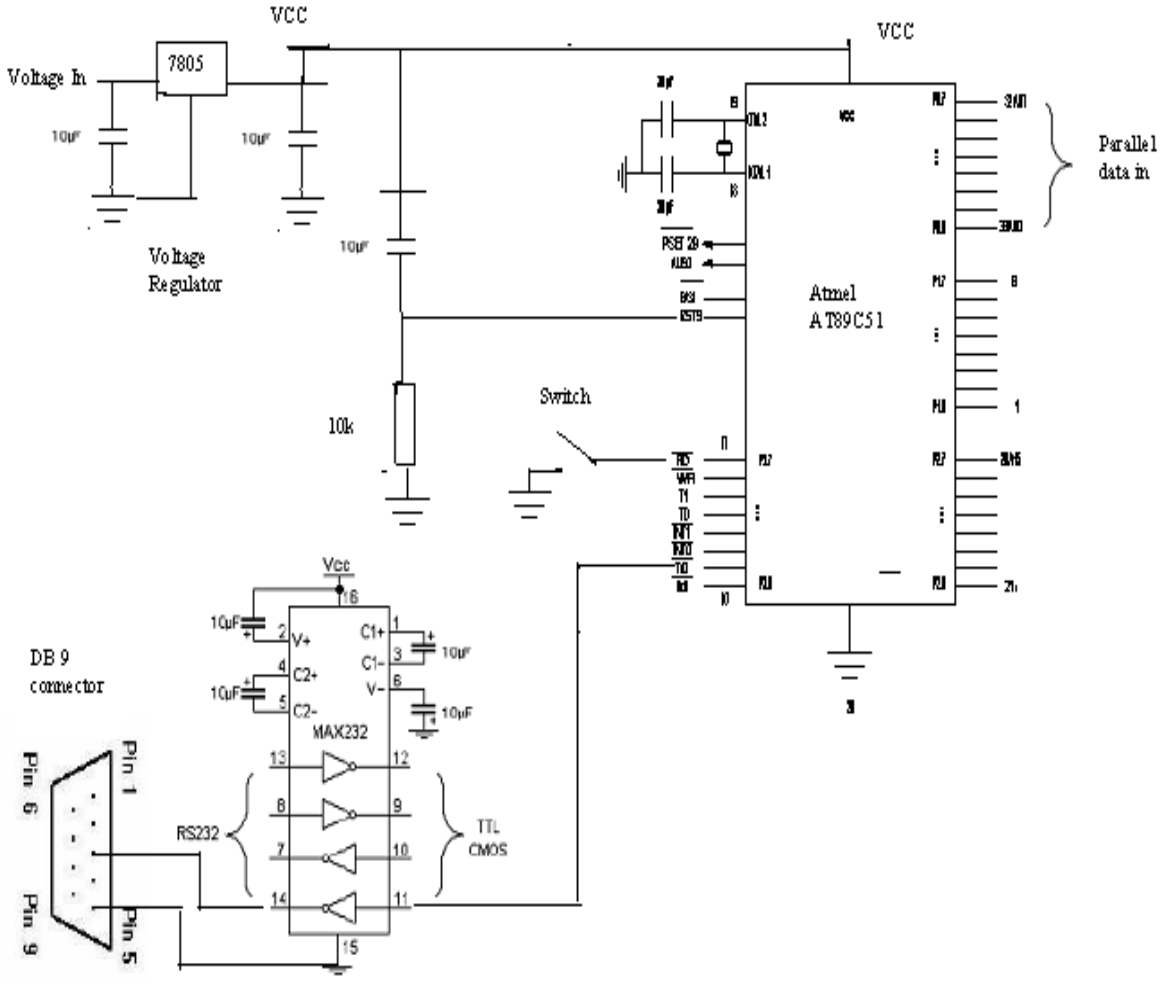


TMOD Bit No	Symbol	Meaning
TMOD.0	M0	Timer#0 mode select bit#0
TMOD.1	M1	Timer#0 mode select bit#0
TMOD.2	C/T#	Timer #0 Counter/Timer (Low Enable) Select Bit
TMOD.3	GATE	Timer #0 GATE Bit , set, timer only rubs INT1 is high
TMOD.4	M0	Timer#1 mode select bit#0
TMOD.5	M1	Timer#1 mode select bit#0
TMOD.6	C/T#	Timer #1 Counter/Timer (Low Enable) Select Bit
TMOD.7	GATE	Timer #1 GATE Bit , set, timer only rubs INT1 is high

M0	M1	Mode	Description
0	**0**	**0**	13-bit Timer Mode
0	**0**	**1**	16-bit Timer Mode
1	**0**	**2**	**8**-bit **A**uto-**R**eload **M**ode
1	**1**	**3**	**S**plit **T**imer **M**ode; TL0 8-bit timer by its
-	**-**	**-**	mode bits, TH0 same xcept by
-	**-**	**-**	timer 1 mode bits ,Timer 1 stopped

TCON Register




VCC
VCC
Voltage In
7805
10uF
10uF
10uF
Voltage
Regulator
10k
Switch
Atmel
AT89C51
Parallel
data in
Vcc
10uF
V+
C1+
C2+
C1-
C2-
V-
MAX232
DB 9
connector
Pin 1
Pin 6
Pin 9
Pin 5
RS232
TTL
CMOS

CHAPTER 4

SOFTWARE STUDY SIMULATION AND CODING

GIS – Concepts

4.1 What IS GIS - An Overview

In the past 10 years or so Geographic Information System (GIS) are gaining importance for storing and organizing spatial information in a computer. GIS empowered by dropping costs of workstations and increasing power of graphics, are beginning to make their way from urban planning, natural resource management, and cartographic production shops.

Geographic Information System are characterized by a wide variety of architectures/platforms, applications, and processing requirements due to a heterogeneous universe of users including geographic, natural resource management environment sciences and archeology.

4.2 Definitions of a GIS

Different definitions provide different perspectives of GIS following is a *blend view* of GIS:

A system of hardware, software, data, people, organizations and institutional arrangements for collecting, storing, analyzing, and disseminating information about areas of the earth.

A particular form of information system applied to geographical data. It was developed from the need to handle and manipulate geographically referenced spatial data.

4.3 How GIS Works

While there is a large range of the sophistication in GIS, a typical GIS has to perform following few functions:

Data Capture
Data Transfer
Validate & Edit the data
Store & Structure
Generalize
Transform
Query
Analyze
Output

These functions, when considered without reference to spatial versus attribute distinction, are very generic, and could be used to describe high-level functionalities of many typical GIS.

4.4 Components of GIS

Two basic components required for any GIS application are:

Computer hardware
Sets of application software

4.4.1 Computer hardware for GIS

The computer or central processing unit is linked to a disk drive storage unit, which provides space frosting data programs. A digitizer or another device is used to convert data from maps and documents into digital form and send them to computer. A plotter or other kind of display device is used to present the result of the data or program on magnetic tape, or for communicating with other systems. Inter computer communication can also take place via a network system over special data lines, or over telephone lines using a Modem. The user controls the computer and peripherals (general term for plotters, printers, digitizer and other apparatus linked to the computer) via a Visual Display Unit (VDU), known as terminal. The user's terminal might itself be a microcomputer, or it might incorporate special hardware to allow maps to be displayed quickly. There is a very wide range of devices that can be used to fill these general hardware requirements.

4.4.2 GIS software component

Data input module/Data capture.
Data storage and database management.
Data output and presentation.
Data transformation.
Interaction with user.

Data Input

Data input covers all aspects of transforming data captured in the form of existing maps, field observation and sensors (including aerial photography, satellites and recording interments) into a compatible digital form. A wide range of computer tools are available for this purpose, including the interactive terminal or Visual Display Unit (VDU), the digitizer, lists of data in text files, scanners (n satellites or airplane for direct recording of data or for covering maps and photographic images) already written on a magnetic media such as tapes, drums and disks. Data input and the verification of data needed to build a geographical database.

Putting the information into the system is the time consuming component of GIS work. Identities of the objects on the map must be specified, as well as their spatial relationships. Editing of information that is automatically captured can also be difficult. Electronic scanners record blemishes on a map just as faithfully as they record the map features. For example, a fleck of dirt might connect two lines that should not be connected. Extraneous data must be edited, or removed from the digital data file.

Data Storage

Topology and attributes of geographical elements (points, lines and areas representing objects on the earth's surface) are structured and organized, both with respect to the way they are handled in the computer and how they are perceived by the users of the system. The computer program used to organize the database is known as a Database Management System (DBMS). In our case this system is *Oracle* database or *Ms Access* Database files.

Data transformation

Data transformation embraces two classes of operation.

1. Transformation needed to remove errors from the data or to bring them up to date or to match them to other data sets.

2. The large array of analysis methods that can be applied to the data in order to achieve answers to the questions asked to the GIS.

Transformation can operate on the spatial and the non-spatial aspects of the data, either separately or in combination. Many of these transformation such as those associated with scale-changing, fitting data to new projections, logical retrieval of data, and calculation of area and perimeters, are of such a general nature that one should not expect to find them in every kind of GIS in one form or another. Other kinds of manipulation may be extremely application specific, and their incorporation into a particular GIS may be only to satisfy the particular users of that system. The kinds of transformation method available, their optimum use and misuse, the ways in which sets of simple transformations can be combined in order achieve certain types of geographical or spatial modeling.

Interaction with the users

This module is concerned with the interaction with the user and query input, which are essential for the acceptance of the use of nay geographical information system. Certainly it is an aspect that until relatively recently has received less attention than it deserves. It is only in the last few years that the average user has received less attention than it deserves. It is only in the last few years that the average user has been able to make direct contact with computer other than via the impersonal and unforgiving media of punched paper tapes and cad handed into the computing center. The widespread introduction of the personnel computer and of programs that are operated by commands chosen from a menu (a list), or that are initiated by a response to requests in an English like command language of verbs, nouns, and modifiers has broken down the barriers that once frightened many a would be computer user away for life.

4.3.3 Querying the GIS

The GIS stores both spatial and non-spatial data in a database system which links the two types of data to provide flexible and powerful ways of querying or asking questions about the data. An example of a spatial query might look like this:

"*Locate and display all playgrounds downstream of landfills within 100 year flood plain"*

This type of query is answered by a set of commands to the GIS that then generates a map display of all sites meeting the criteria expressed in the query. The user may also query the GIS by the textual attributes in the tabular database and then display the map features that correspond to these attributes. An example of this type of query is as follows:

"*Display all water mains installed before 1950 with a diameter less than 12 inches"*

This query results in a map display of the water mains in the study area with the specific mains in the query highlighted. Alternatively a report could be generated which lists the complete information on each segment of water mains which meet the criteria in the query.

GIS data integration

Many Geographic Information Systems handle both vector and raster data from a wide variety of sources including satellite imagery, cadastral information, hand digitized maps and scanned images.

Data output and presentation

Data output and presentation concerns the ways data is displayed and the results of analysis are reported to the users. Data may be presented as maps, table and figures (graphs and charts) in a variety of ways, ranging from the ephemeral image on cathode

ray tube (CRT) through hard copy output drawn on printer or plotter to information recorded on magnetic media in digital form.

4.3.4 GIS Applications

GIS are now used extensively in government, business, and research for a wide range of applications including environmental resource analysis, land use planning, location analysis, tax appraisal, utility and infrastructure planning, real estate analysis, marketing and demographic analysis, habitat studies, archaeological analysis, and military planning.

Natural resources management

This is one of the first major areas of application it includes management of Wildlife habitat.

Wild and scenic rivers.
Recreation resources.
Floodplains.
Wetlands.
Agricultural lands.
Aquifers.
Forests.
Facilities management

One of the largest areas of application has been in facilities management. Uses for GIS in this area have included

Locating underground pipes and cables.
Balancing loads in electrical networks.
Planning facility maintenance.
Tracking energy use.
Land management

Local, state, and federal governments have found GIS particularly useful in land management. GIS has been commonly applied in areas like

Zoning and subdivision planning.
Land acquisition.
Environmental impact policy.
Water quality management.
Maintenance of ownership.
Street networks
Address matching.
Location analysis / site selection.

4.4 Research And Analysis

4.4.1 Graphics and GIS

Geographical information systems differ from computer graphics because the latter are largely concerned with the display and manipulation of visible material. Computer graphics systems do not pay much attention to the non-graphic attributes that the visible entities might or might not have, and which are useful data for analysis. Good computer graphics packages are neither by itself sufficient for performing the tasks expected, nor such drawing packages necessarily a good basis for developing such a system.

4.4.2 CADD and GIS

Computer Aided Design and Drafting (CADD) technology is widely used by many professionals (including engineers, architects, and planners) to help them design and produce design drawings. Data in a CADD system is organized on layers that are conceptually like registered overlays. The layers can be used to organize map features by theme, such as streams versus roads versus structures, or by type of data, such as line work versus text. The table represents the elements stored in a CADD data file. Plotted CADD data rivals scribed cartographic products in graphic quality and precision. CADD can greatly reduce map production time and save money over the traditional cartographic process. For instance, corrections are much easier to make. To make changes to the map, the manual process requires erasing and re-inking. A CADD system allows the user to quickly modify a single element without affecting other features. CADD offers many other benefits over traditional film-based, manually prepared mapping techniques; data is better organized easier to store and retrieve, and so on. For these reasons, many civil engineers use CADD to store and manipulate map- ping data used in the design process.

Similarly, map atlas producers and other cartographers can use CADD-based systems to create maps. However, CADD is not suited for analyzing map data. In a CADD system, map features are associated by theme, using layers, and the features are all referenced to a common geographic coordinate system, but further relationships among the data elements are not defined.

4.4.3 COMPARASION

GIS is similar to CADD in that it references graphic data elements to a xy coordinate system, and it separates map features by layer (also referred to as a map "theme" or "coverage"). Although the GIS may divide the entire area being mapped into separate files, much like map sheets, it handles all of the data in these files as though they were in one large "seamless" map file. In addition to graphic (often referred to as "spatial") data, a GIS also stores attribute data. These are associated with the spatial data and provide further descriptive information about them.

GIS differs from CADD in that the spatial relationships among all data elements are defined. This convention, known as data topology, goes beyond merely describing the location and geometry of map features. Topology also describes how linear map features are connected, how areas are bounded, and which areas are contiguous. To define map topology, a GIS uses a special database structure. As in a CADD system, all map features are related to a geographic coordinate system. However, unlike a CADD system (which defines map features simply as lines, line strings, and symbols), a GIS defines map features as nodes, lines, and areas. (Other terms-such as points, arcs, and polygon are also frequently used).

4.4.4 Existing GIS

In order to get a better idea of the types of features found in a GIS system, this chapter reviews the core products of the four leading GIS vendors according to the GIS industry: Environmental Systems Research Institute, Inc. (ESRI), Intergraph, MapInfo, and AutoDesk.

ArcInfo

ArcInfo provides a series of out-of-the-box GIS applications, as well as a customization capability. ArcInfo can be customized using drag-and-drop and menu driven tools. Industry-standard Microsoft Visual Basic for Applications (VBA) is

provided for all scripting and application customization jobs. Any component object model (COM)-compliant programming language can also be used to customize and extend ArcInfo.

In ArcInfo , display, query, and analysis of map data are supported by the ArcMap application. ArcMap provides a direct data read capability, which allows data sets to be used on the fly without translation or use of an intermediate format. ArcMap supports ArcInfo coverages, ESRI shapefiles, Spatial Database Engine (SDE) layers, map libraries, ArcStorm layers, DXF and DWG, DGN, and many image types. In addition, ArcMap supports on-the-fly projection at the individual layer level.

ArcMap is a menu-driven application for working with map data. It is an integrated application for creating and editing spatial databases, displaying and querying geographic data, performing complex analysis, generating quality reports and charts, and making high-quality maps. Features include the following.

Integrated map display, editing, and production environment

Windows user interface

Data visualization for interpretation/analysis

Out-of-the-box usability

Creation of charts and reports

Mapping capabilities Computer-aided design (CAD) editing on intelligent GIS databases (ArcSDE databases, coverages, and shapefiles)

Drag-and-drop customization

ArcMap includes an integrated Object Editor capable of multiuser geographic and attribute entry and update. The Object Editor can work with coverages, shapefiles, and geodatabases stored in a database management system (DBMS) using ArcSDE. The Object Editor supports the following.

Creation and updating of shapefiles, coverages, and geodatabases

Editing of features according to rules/behavior (i.e., network connectivity, attribute consistency, and so on)

Display of raster and vector data

Snapping to vector data (including CAD files)

Versioning and conflict resolution across work groups

CAD/sketching function directly on the GIS database

Editing within magnify windows

Integrated tracing

Customization for user-defined tasks/tools

The Object Editor includes many of the graphic editing functions of CAD editing packages for editing map features, using rule-based tools for creating and maintain- ing spatial databases. The Object Editor also enables users to directly edit data in a DBMS via ArcSDE.

Arc View

ArcView is a desktop mapping and GIS tool that enables the user to select and display various combinations of data. ArcView is integrated with other applications, creating an environment for analysis and desktop publishing, and spreadsheets, databases, word processing, publication graphics, and other software applications that extend the functionality of ArcView. ArcView works directly with ArcInfo, Arc- CAD, PC ArcInfo, and SDE databases. It can import

a variety of data sources, including CAD files, spreadsheets, raster data, and other databases. The key features of ArcView include the following.

"Windows" graphical interface
Integrated charts, maps, tables, graphics, and multimedia
Visual mapping
Wizards for map composition
Labeling and text tools
Industry/ application -specific symbols
Geographic hot links to supported data formats
Analysis wizards for geoprocessing operations
Address matching and geocoding
Utility for projection and datum transformation
Geographic and tabular data editing
Integration of images and CAD data
Client/server access to data warehouses

Five CDs of data included

Report writing using Crystal Reports
Self-paced tutorial
User manual
Online help
Customizable with developer environment

Expanded analysis capabilities using optional extensions

ArcView runs on Microsoft Windows and UNIX computers. ArcView can map tabular data residing in Microsoft Access, dBASE, FoxPro, ASCII, INFO, SQL, Open Database Connectivity (ODBC), and/or SAP R/3 databases. ArcView supports the following GIS data formats. ESRI shapefiles, Arclnfo coverages, Arclnfo dynamic segmentation coverages, route systems, PC Arclnfo coverages, AutoCAD (DXF and DWG), MicroStation (DGN and MSG), TIFF 6.0 (including GeoTIFF), VPF, ADRG, CADRG, CIB, NITF, MrSID, JPEG (JFIF), ERDAS IMAGINE, ERDAS LAN and GIS, BSQ, BIL, BIP, SunRaster files, BMP , GRID (as image data), and DIGEST (ASRP and USRP) ArcSDE with Spatial Data- base Engine (SDE), or Oracle Spatial Data Option (SDO) with SDE.

GeoMedia

GeoMedia is a GIS visualization and analysis tool, as well as a platform for custom GIS solutions, providing the following tools.

Data integration with major GIS vendor formats
Spatial analysis tools
Map layout
An open development platform for creating custom applications
Microsoft Windows-standard user interface

GeoMedia provides access to leading GIS product formats through "live" data connections to "native" GIS data repositories. GeoMedia can integrate data using on- the-fly coordinate transformation and feature definition. GeoMedia data server technology supports views of multiple GIS data sets in various formats, and analyzes this information by running queries, buffer zones, and thematics across multiple GIS formats. Accessible GIS formats

include MGE, FRAMME, MGE Segment Man- ager, Maplnfo, AutoCAD, Oracle SC Relational, MicroStation, Arclnfo, ArcView Shapefiles, Microsoft SQL Server, and Oracle8i Spatial. GeoMedia supports Oracle, including Oracle8i Spatial, as the geospatial warehouse.

GeoMedia's analysis tools include nine types of spatial functions, such as "entirely contained by" or "touches"; a suite of arithmetic operations; and the tools necessary to create "what if' queries. It can create buffer zones and spatial overlays, and query spatial data within a specified area. GeoMedia's dynamic segmentation capabilities can query and segment linear and point data from multiple MGE databases.

GeoMedia's thematic mapping features create maps with color-coded and patterned attribute data, and integrate multimedia with GIS, including hyperlinks to files that contain sounds, images, text, and satellite imagery. GeoMedia also supports geocod- ing, which translates tabular data, such as street addresses, into spatial data.

GeoMedia can create maps and presentations with tools for map layout. This includes tools for the placement of map information, as well as the typical graphics found on printed maps and charts. Marginalia such as legends, scale bars, and North arrows are automatically generated and dynamically linked to the map information. Changes to the map are automatically reflected in the layout. GeoMedia provides multiple selection methods for controlling the map data to plot: map window, exist- ing area feature, rectangle, polygon, paper size, geographic frame, or projected frame.

AutoCAD Map

AutoCAD Map is software for mapping and GIS analysis in the AutoCAD environment. It provides GIS analysis tools and features for creating, maintaining, and producing maps and geographic information, along with the underlying functionality of AutoCAD. The following are some of its key features.

Data integration: Import and export maps from standard CAD and GIS file formats. Provides database-linking capability to add "intelligence" to maps. Link internally and externally to Microsoft Access, dBase, Oracle, FoxPro, Paradox, and other ODBC-compliant databases with drag-and-drop configuration. Raster tools provide support for multiple raster file formats.

Multiple map access: Work with large data sets and multiple maps, simulta- neously sharing them with other users without version conflicts. Work with multiple maps, in different coordinate systems, in a single session. Save map projects and the settings for the session are saved automatically for recall at a later time. Provide multiuser access to single or multiple maps with entity- level locking.

Map creation and cleanup: Digitizing,coordinate transformation, and drawing cleanup. Digitizing tools attach object data or SQL links. Transform maps with different coordinate systems into one standard or customized system. Edit map data with rubber sheeting, edge and boundary trim, and so on. Prepare for topology, with automated drawing cleanup tools.

Querying: Access and query data from different sources and combine the information into one map, alter object properties during queries, save routine queries for future use, and perform topology queries.

GIS spatial analysis: Topological operations, path tracing, and other analytical capabilities. Create, edit, and save node, network, and polygon topologies. Perform flood analysis and buffer, shortest, and optimal path trace. Conduct polygon overlay and dissolve operations. Create thematic maps based on associated data.

Coordinate conversion: Support for global and local coordinate systems. Combine multiple maps into a common coordinate system. Define a custom coordinate system.

Construction tools: Build new maps using construction tools. Use polar construction tools to place field locations of objects the way they were measured. Switch between multiple editing alignments.

Presentation and plotting: Thematic mapping and plotting. Flexible plot styles support changes to the visual properties of any object, including color, line type, line weight, and pen assignment at plot time. Predefine map's appearance and reapply the same settings.

4.4.6 Assessment

Requirement was to make a decision support system having following characteristics:

Working in a GIS environments

Integration of an expert system to help in decision.

4.5.1 Tools available

Following were available for our use:

Existing GIS: Commonly used GIS like Arcview and Geomedia

Expert system: Control structures of an expert system for making a communication planning have been developed earlier and were available for use.

4.5.2 Analysis

Available GIS have their own standard format in which these deal and understand.

Almost all available GIS (Arcview and Geomedia) have been developed using special languages (e.g., Arcview use avenue script).

4.5.3 Approach Followed

Make our own GIS environment using language in which the skill level is better.
The GIS should be easy and have the specific necessary functions required.
Map the control structures on the GIS which can easily interface with these control structures.

4.6 SOFTWARE PORTION

The software of a radar interface system essentially needs to do the following tasks: -

1. Display the map of the location.
2. To receive the navigation data through serial port.
3. Convert the Radar co-ordinates into screen co-ordinates.
4. Display the firing and target location of enemy gun.

I will cover all the parts one by one. But initially the basic principle of the software will be explained.

4.7 PRINCIPLE OF TRACKING

This portion of the software works on a very simple principle that is described in the following: -

If we have the map of the location of the field, then we just need the exact coordinates of only two points to calibrate the map and track the vehicle.

Let two points be Point1 and Point2. By point I mean that each point comprises latitude, longitude, altitude, x-position and y-position on the map. The x and y positions will be x and y-coordinates on the screen.

Now we can define a scale in the x and y directions as follows.

x_Scale = (Point1.Longitude - Point2.Longitude)/(Point1.x - Point2.x)
y_Scale = (Point1.Latitude - Point2.Latitude)/(Point1.y - Point2.y)

From the Point1 and the scales, we now can guess the latitude and longitude of the Origin (0,0). Now,

Origin.Latitude = Point1.Latitude - (Point1.y * y_Scale)
Origin.Longitude = Point1.Longitude - (Point1.x * x_Scale)

After we have got the coordinates of the point (0,0), we can track the point. To do this, assume that the point obtained by the radar is the position obtained by the PC as CurrPosition.

Now the x and y-position of the point on the map will be calculated as follows:

x = (CurrPosition.Longitude - Origin.Longitude)/x_Scale
y = (CurrPosition.Latitude - Origin.Latitude)/y_Scale

But in our project as we have acquired the map which has already been digitized, so we do not go into the digitization (geo-referencing) portion of the map.

Now the four portions of the software will be explained one by one.

4.8 CHOICE OF THE PROGRAMING ENVOIRNMENT

The choices available with us for programming were: -

1. Microsoft Visual Basic
2. Turbo C++
3. Microsoft Visual C++

The advantages and shortcomings of each will be discussed one by one.

4.8.1 VISUAL BASIC

Visual Basic is a very good programming tool with respect to making forms and GUI's. The GUI in our project as described above is an overlay window interacting with a working environment i.e Geomedia. Visual Basic is easy and can handle the graphical images very easily. So it must be a language of choice as for as GUI and loading of a map are concerned. But the only problem with visual basic is its lack of support for communicating with the ports. Whereas a very important part of our software is to retrieve data from the serial port. Thus this choice was ruled out.

4.8.2 TURBO C++

The second good option was Turbo C++. Turbo C++ is very good at dealing with the serial and other ports. So it solved our problem of port I/O. But again it was a partial solution because Turbo C++ is poor at dealing with GUI's and graphics. In connection to graphics it is worth mentioning here that Turbo C++ does not carry the capability of displaying more than 16 colors whereas a bitmap is 24-bit colored image. Thus this choice was also ruled out.

4.8.3 VISUAL C++

Coming on to the last choice i.e. Microsoft Visual C++6.0. This language had all the desired capabilities for us. It is extremely strong at making GUI's, displaying graphics and has the inherited capability of hardware access from Turbo C++ (its predecessor). Thus we decided to use Visual C++ for our project and successfully implemented the principle in it.

4.9 CHOICE OF PROGRAMING FRAMEWORK

For using Visual C++ we had the choice to use: -

1. API's (Application Programming Interfaces)
2. MFC's (Microsoft Foundation Classes)

In the following a brief description of both is given and then in the end our design approach is given.

4.9.1 API's

API's are a set of more than one thousand functions especially designed for Visual C++ programmers. These functions come as a set of API's for implementing the Win 32 programming. The programming style using API's is just like programming in Turbo C++. I also used a bit of it until I had shifted to MFC's. API's are very strong but the problem with them is that as the program length grows, their complexity increases tremendously to a point where it becomes extremely difficult to find something of your concern. The second problem with the API's is that before starting any work of your concern you have to spend a lot of time developing the main window of your program. So a good portion of your time is wasted in making a GUI.

4.9.2 MFC's

MFC's also made by Visual C++ programmers provide you with the following advantages: -

1. After its six steps you have a working application and you don't have to worry about making a GUI.
2. Class Wizard is always there to assist you in making a program.
3. The SDI/MDI support is available which is otherwise very difficult in API's.

The document/view architecture and framework is extremely strong and once understood provides you with a lot of help conceptually.

Two things Class Wizard and document/view architecture need a little explanation here.

Class Wizard is a dialog box that has information bout all your classes, icons, menu resources. It also has the definition of all the windows messages that can be dealt by a specific class. So whenever you need to add a message handler to your program, you can get the help from Class Wizard.

The second thing is document/view architecture. This concept is base on the fact that document is a universal set of application's data and view is its subset and displays part of the document. So whatever data handling is done, that is done in the document class and the display functions are written in the view class. The view class has the knowledge about its document through a function GetDocument() and similarly document also has knowledge about its all the connected view. I have also used this concept in my application and kept the data away from the display.

Thirdly this option was most feasible as the source code of Geomedia is written using MFCs.

```
// parser.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "parser.h"

#include "MainFrm.h"
#include "parserDoc.h"
#include "parserView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CParserApp

BEGIN_MESSAGE_MAP(CParserApp, CWinApp)
	//{{AFX_MSG_MAP(CParserApp)
	ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
		// NOTE - the ClassWizard will add and remove mapping macros
here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG_MAP
	// Standard file based document commands
	ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
	ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CParserApp construction

CParserApp::CParserApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CParserApp object

CParserApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CParserApp initialization

BOOL CParserApp::InitInstance()
{
	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
```

```
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif

	// Change the registry key under which our settings are stored.
	// TODO: You should modify this string to be something appropriate
	// such as the name of your company or organization.
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));

	LoadStdProfileSettings();  // Load standard INI file options (including MRU)

	// Register the application's document templates.  Document templates
	//  serve as the connection between documents, frame windows and views.

	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CParserDoc),
		RUNTIME_CLASS(CMainFrame),	// main SDI frame window
		RUNTIME_CLASS(CParserView));
	AddDocTemplate(pDocTemplate);

	// Parse command line for standard shell commands, DDE, file open
	CCommandLineInfo cmdInfo;
	ParseCommandLine(cmdInfo);

	// Dispatch commands specified on the command line
	if (!ProcessShellCommand(cmdInfo))
		return FALSE;

	// The one and only window has been initialized, so show and update it.
	m_pMainWnd->ShowWindow(SW_SHOW);
	m_pMainWnd->UpdateWindow();

	return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
```

```
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
		// No message handlers
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CParserApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
```

```
}

//////////////////////////////////////////////////////////////////////////
// CParserApp message handlers
```

Now the program for accessing the ports is

```
// Port.cpp: implementation of the CPort class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "PortIo.h"


//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CPortIo::CPortIo()
{
m_hComm = NULL;

// create events
m_ov.hEvent = NULL ;
m_IsOpen=FALSE ;
}

CPortIo::~CPortIo()
{
}

BOOL CPortIo::WriteToPort(CPortIo*port,int batch,int size)
{
if(m_IsOpen==TRUE)
{
ASSERT(m_hComm != 0);
// memset(m_WriteBuffer, 0,
//       sizeof(m_WriteBuffer));
//for(int i=0;i<size;i++)
//{
// m_WriteBuffer[i]=Buffer[i];
//            WriteChar(port,m_WriteBuffer[i]);
//}
WriteChar(port,batch,size);
return TRUE;
```

```
}
else
AfxMessageBox("Please Open COM First");
return FALSE;
}

void CPortIo::WriteChar(CPortIo*port,int batch,int length)//unsigned char byte)
{

BOOL bWrite = TRUE;
BOOL bResult = TRUE;

DWORD BytesSent = 0;

if (bWrite)
{
// Initailize variables
port->m_ov.Offset = 0;
port->m_ov.OffsetHigh = 0;

// Clear buffer
PurgeComm(port->m_hComm,  PURGE_TXCLEAR );

bResult = WriteFile( port->m_hComm,  // Handle to COMM Port
&port->target[batch],
length,//1,strlen((char*)port->m_WriteBuffer),      // Length of message to send
&BytesSent,                                          // Where to store the number of
bytes sent
&port->m_ov );

// deal with any error codes
if (!bResult)
{
//        AfxMessageBox(port->m_WriteBuffer);

DWORD dwError = GetLastError();
switch (dwError)
{
case ERROR_IO_PENDING:
{
// continue to GetOverlappedResults()
BytesSent = 0;
bWrite = FALSE;
break;
}
case ERROR_INVALID_HANDLE:
```

```
{
AfxMessageBox("invalid handle");
break;
}
default:
{
// all other error codes
port->ProcessErrorMessage("WriteFile()");
}
}
}
else
{
AfxMessageBox("Send Failure");
}
} // end if(bWrite)

if (!bWrite)
{
bWrite = TRUE;
bResult = GetOverlappedResult(port->m_hComm, // Handle to COMM port
&port->m_ov,          // Overlapped structure
&BytesSent,           // Stores number of bytes sent
TRUE);
// Wait flag returns the results of an overlapped operation
// communications device.
// deal with the error code

if (!bResult)
{
port->ProcessErrorMessage("GetOverlappedResults() in WriteFile()");
}
} // end if (!bWrite)

// Verify that the data size send equals what we tried to send
//        if (BytesSent != strlen((char*)port->m_WriteBuffer))
//        {
//                TRACE("WARNING: WriteFile() error.. Bytes Sent: %d; Message Length:
%d\n", BytesSent, strlen((char*)port->m_WriteBuffer));
//        }
}

BOOL CPortIo::InitPort(CWnd *pPortOwner, UINT portno, UINT baud,
char parity, UINT databits, UINT stopbits,
DWORD dwCommEvents)
{
```

```
ASSERT(portno>0 && portno<3);
ASSERT(pPortOwner!=NULL);

if(m_ov.hEvent!=NULL)
ResetEvent(m_ov.hEvent);
m_ov.hEvent=CreateEvent(NULL,TRUE,FALSE,NULL);

m_pOwner=pPortOwner;
m_portno=portno;
m_dwCommEvents=dwCommEvents;
char *szPort=new char[50];
char *szBaud=new char[50];

if(m_hComm!=NULL)
{
CloseHandle(m_hComm);
m_hComm=NULL;
}
sprintf(szPort, "COM%d", portno);
sprintf(szBaud, "baud=%d parity=%c data=%d stop=%d",
baud, parity, databits, stopbits);

m_dcb.BaudRate=baud;
m_dcb.ByteSize=databits;
m_dcb.Parity=parity;
m_dcb.StopBits=stopbits;
m_dcb.fBinary=TRUE;
if(parity!='N')
m_dcb.fParity=TRUE;

m_hComm=CreateFile(szPort,
GENERIC_READ|GENERIC_WRITE,
0,
NULL,
OPEN_EXISTING,
FILE_FLAG_OVERLAPPED,
0);
if(m_hComm==INVALID_HANDLE_VALUE)
{
delete [] szPort;
delete [] szBaud;
return FALSE;
}
SetupComm(m_hComm,512,1024);
m_CommTimeouts.ReadIntervalTimeout = 0;// time limit every 2 char in comunication line
m_CommTimeouts.ReadTotalTimeoutMultiplier = 0;
```

```
m_CommTimeouts.ReadTotalTimeoutConstant = 0;
m_CommTimeouts.WriteTotalTimeoutMultiplier = 0;
m_CommTimeouts.WriteTotalTimeoutConstant = 0;

// configure
if (SetCommTimeouts(m_hComm, &m_CommTimeouts))
{
if (SetCommMask(m_hComm, dwCommEvents))
{
if (GetCommState(m_hComm, &m_dcb))
{
m_dcb.fRtsControl = RTS_CONTROL_ENABLE;          // set RTS bit high!
if (BuildCommDCB(szBaud, &m_dcb))
{
if (SetCommState(m_hComm, &m_dcb))
; // normal operation... continue
else
ProcessErrorMessage("SetCommState()");
}
else
ProcessErrorMessage("BuildCommDCB()");
}
else
ProcessErrorMessage("GetCommState()");
}
else
ProcessErrorMessage("SetCommMask()");
}
else
ProcessErrorMessage("SetCommTimeouts()");

delete [] szPort;
delete [] szBaud;

PurgeComm(m_hComm, PURGE_RXCLEAR | PURGE_TXCLEAR | PURGE_RXABORT |
PURGE_TXABORT);

TRACE("Initialisation for communicationport %d completed.\nUse Startmonitor to
communicate.\n", portno);
m_IsOpen=TRUE;
return TRUE;
}

void CPortIo::ProcessErrorMessage(char *ErrorText)
{
char *Temp = new char[200];
```

```
LPVOID lpMsgBuf;

FormatMessage(
FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
NULL,
GetLastError(),
MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
(LPTSTR) &lpMsgBuf,
0,
NULL
);

sprintf(Temp, "WARNING:  %s Failed with the following error: \n%s\nPort: %d\n",
(char*)ErrorText, lpMsgBuf, m_portno);
MessageBox(NULL, Temp, "Application Error", MB_ICONSTOP);

LocalFree(lpMsgBuf);
delete[] Temp;
}
```

References

www.beyondlogic.com
www.google.com.pk
www.raytheon.com
www.programmersheaven.com
www.logix4u.com
www.atmel.com
www.intel.com
www.mathworks.com
Interfacing 8086 microprocessor.
Microprocessor Architecture by D.V.Hall.
Programming in C++ by Dietel & Dietel.
Programming in Visual C++ by Ivor Horton.