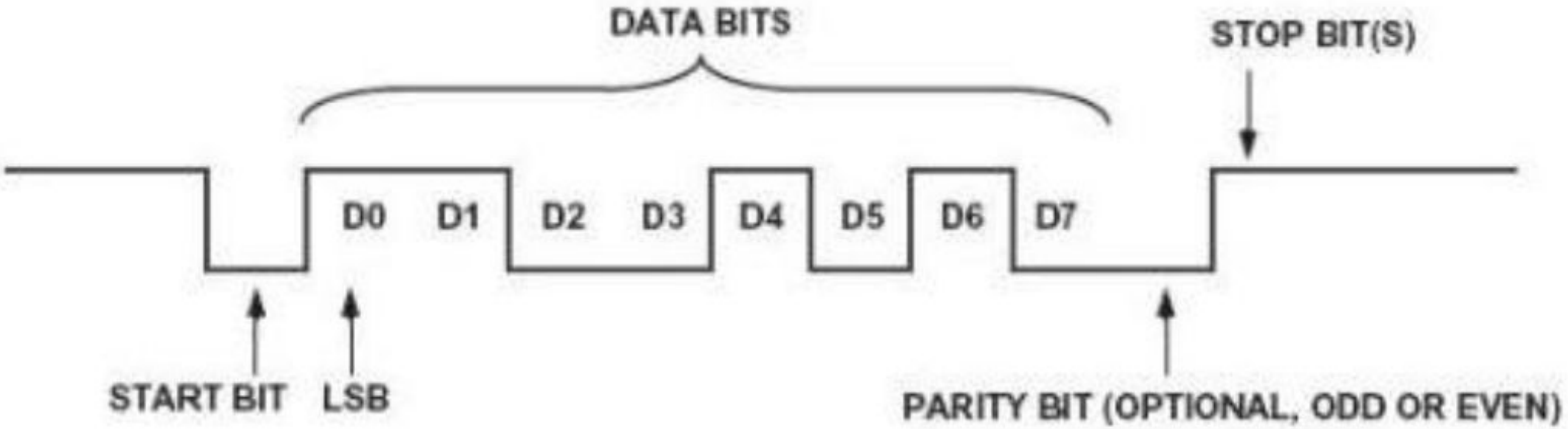
19Z604 Embedded Systems

Dr.N.Arulanand,
Professor
Dept of CSE,
PSG College of Technology

Agenda

- RS-232 Serial Programming

RS-232 Communication Parameters

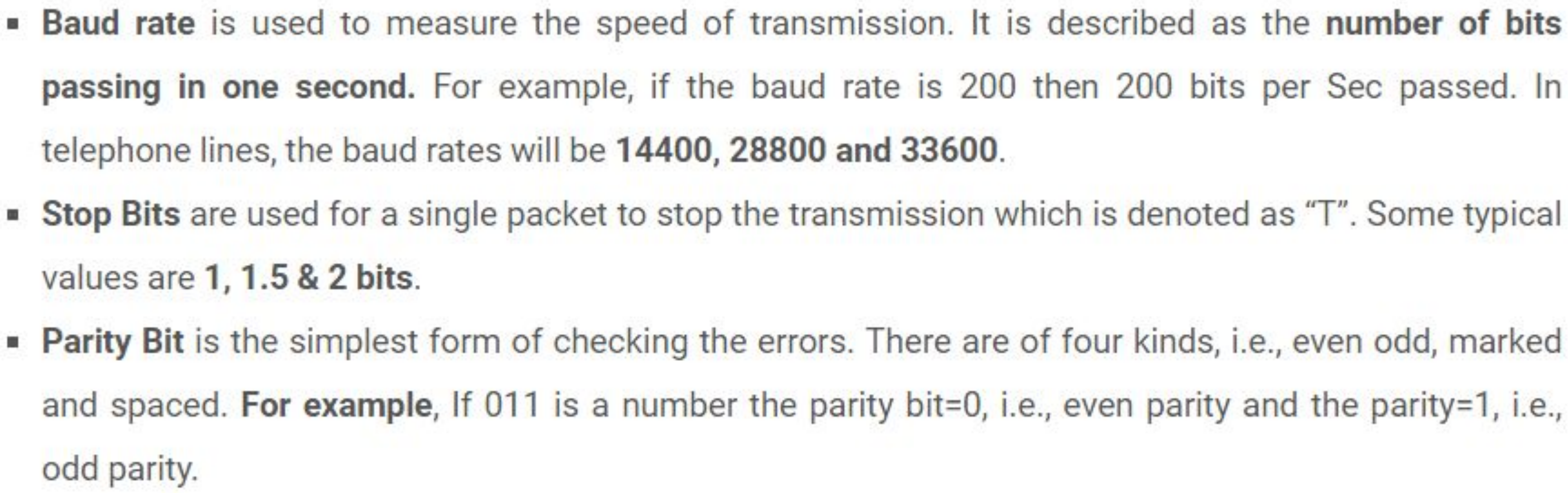


Asynchronous Transmission: UART data transfers are asynchronous. The transmitter transmits each bit (of the word being transmitted) for a fixed duration (defined by baud rate).

UART

- **Universal Asynchronous Data Receiver &Transmitter (UART)** used in connection with RS232 for transferring data between printer and computer. The microcontrollers are not able to handle such kind of voltage levels, connectors are connected between RS232 signals.

Characteristics of Serial Communication



- **Baud rate** is used to measure the speed of transmission. It is described as the **number of bits passing in one second.** For example, if the baud rate is 200 then 200 bits per Sec passed. In telephone lines, the baud rates will be **14400, 28800 and 33600**.
- **Stop Bits** are used for a single packet to stop the transmission which is denoted as "T". Some typical values are **1, 1.5 & 2 bits**.
- **Parity Bit** is the simplest form of checking the errors. There are of four kinds, i.e., even odd, marked and spaced. **For example**, If 011 is a number the parity bit=0, i.e., even parity and the parity=1, i.e., odd parity.

RS-232 Recap

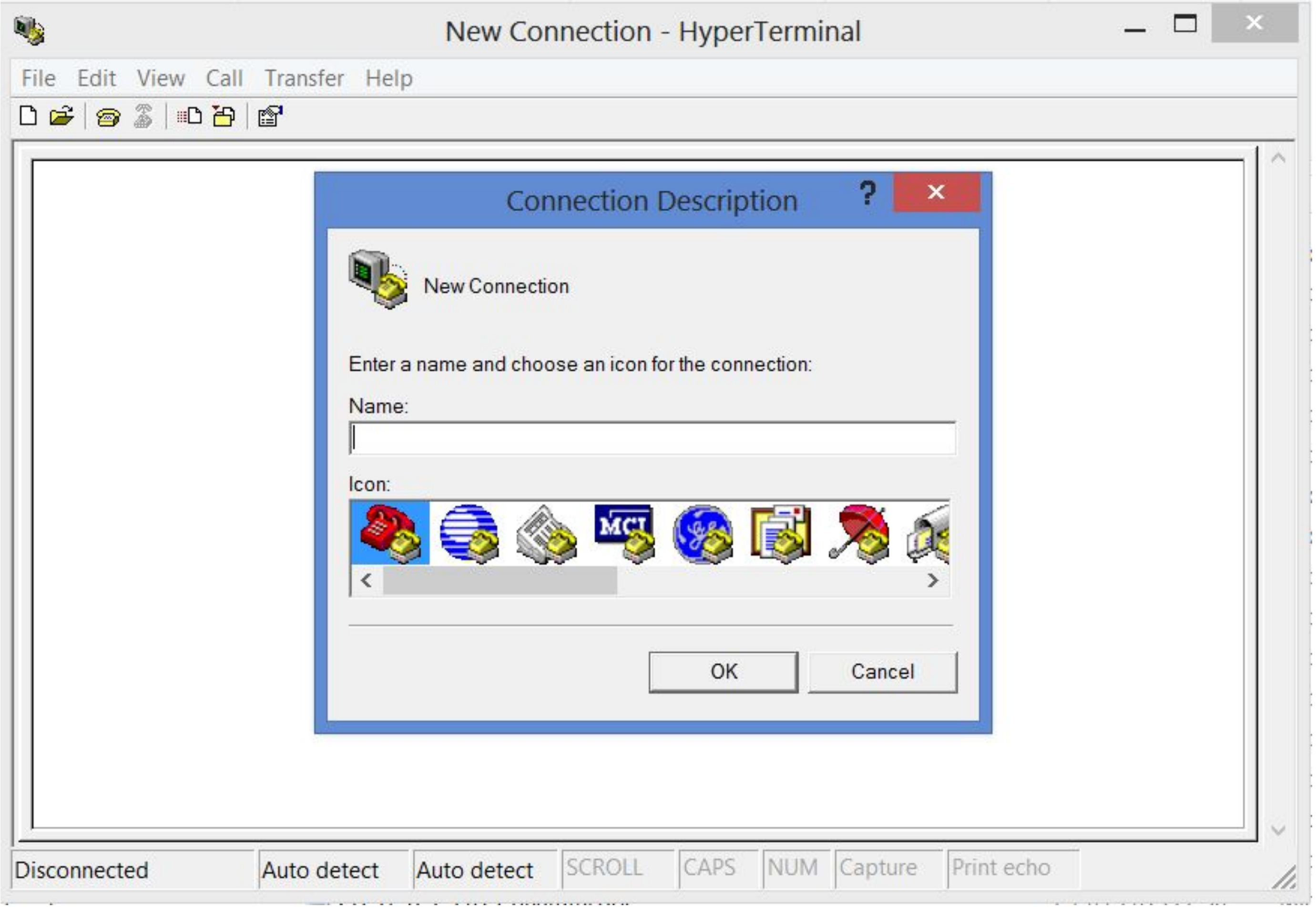
- Asynchronous
- Full-duplex
- RS-232 character by character transmission. Each character will have a start and stop bits
- RS-232 Communication parameters
  - Baudrate, start bits, stop bits, parity bits, data bits
  - Flow control (RTS/CTS)
- Null modem cable

RS232 Programming - Projector

- Look at the example. See how it works ?

- Why we need our own Protocol for any RS-232 Communication?

Hyperterminal screen shots

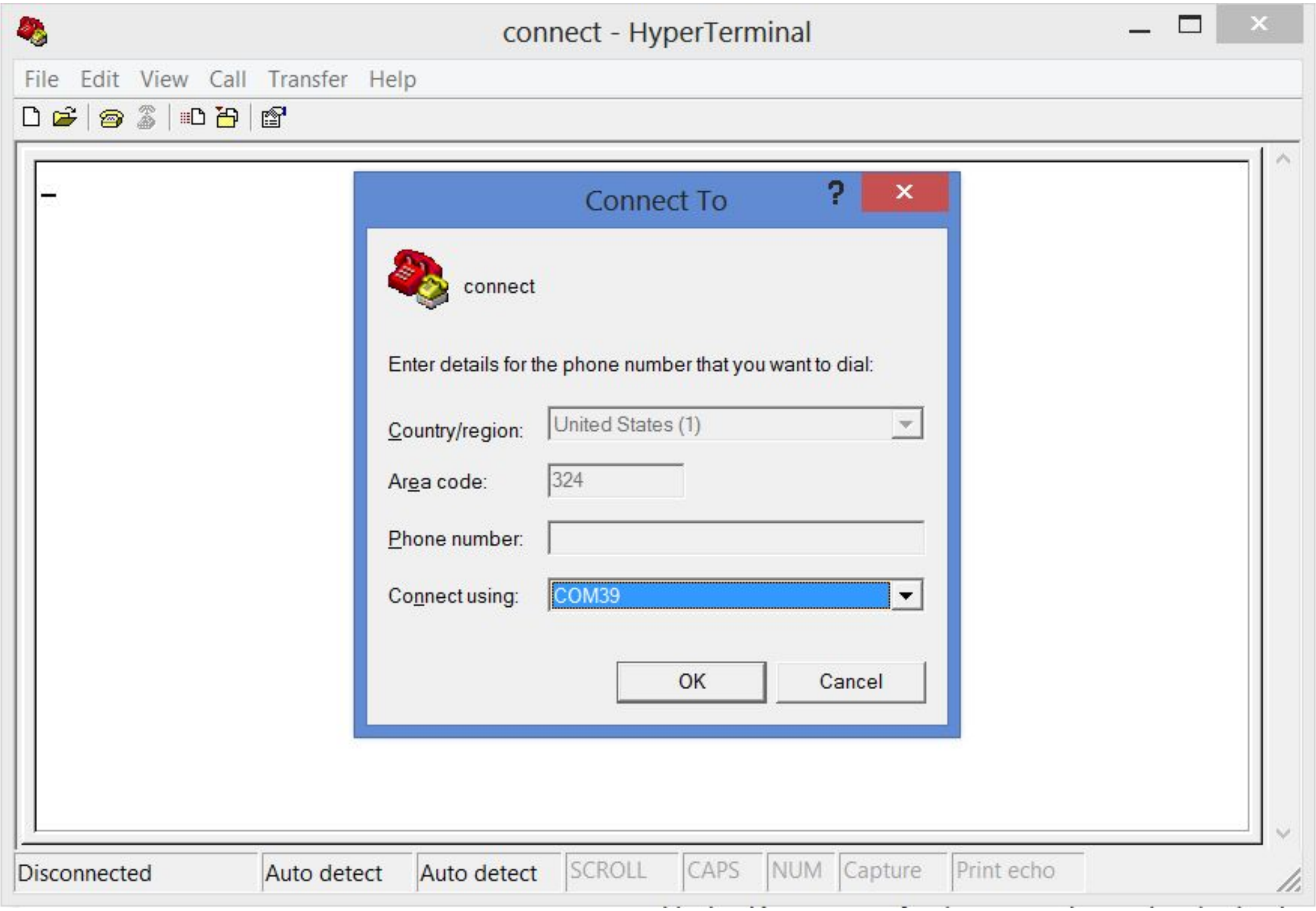


Enter a name and choose an icon for the connection:

Name:

Icon:

Hyperterminal screen shots



Enter details for the phone number that you want to dial:

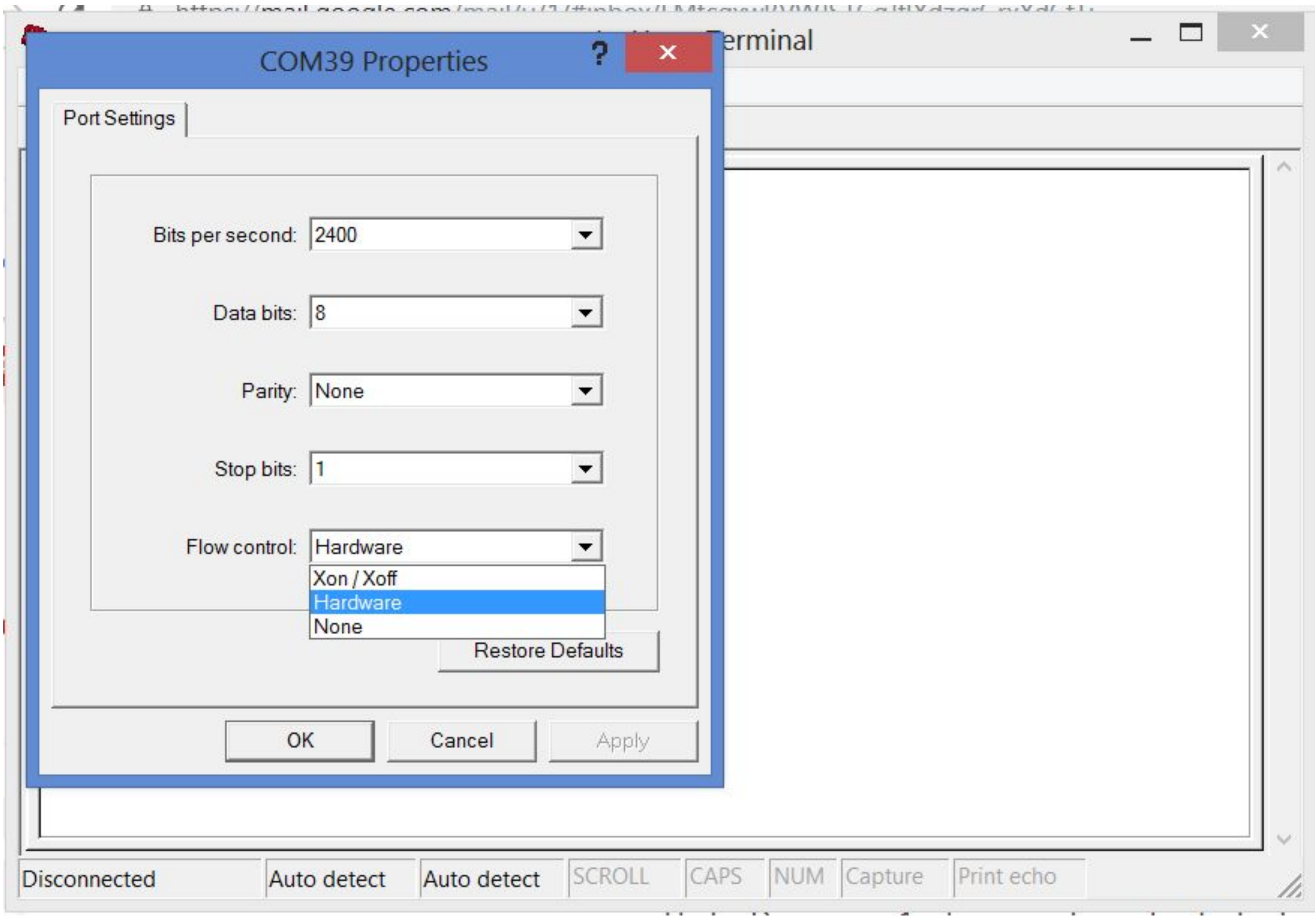
Country/region: United States (1)

Area code: 324

Phone number:

Connect using: COM39

Hyperterminal screen shots



COM39 Properties

Port Settings

Bits per second: 2400

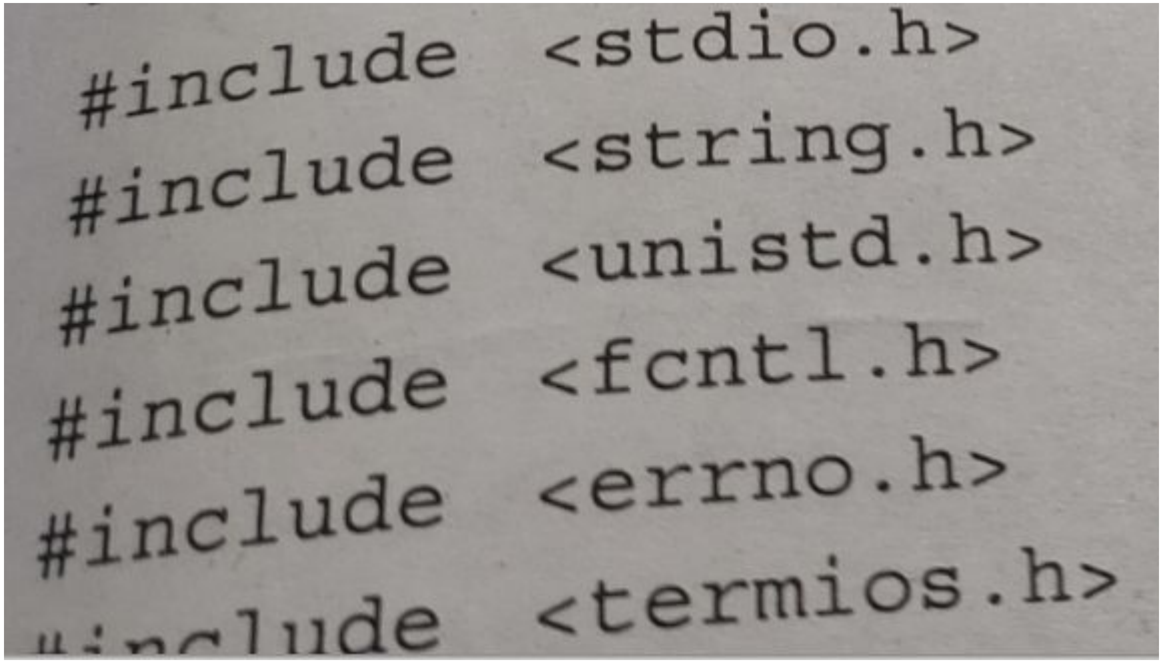
Data bits: 8

Parity: None

Stop bits: 1

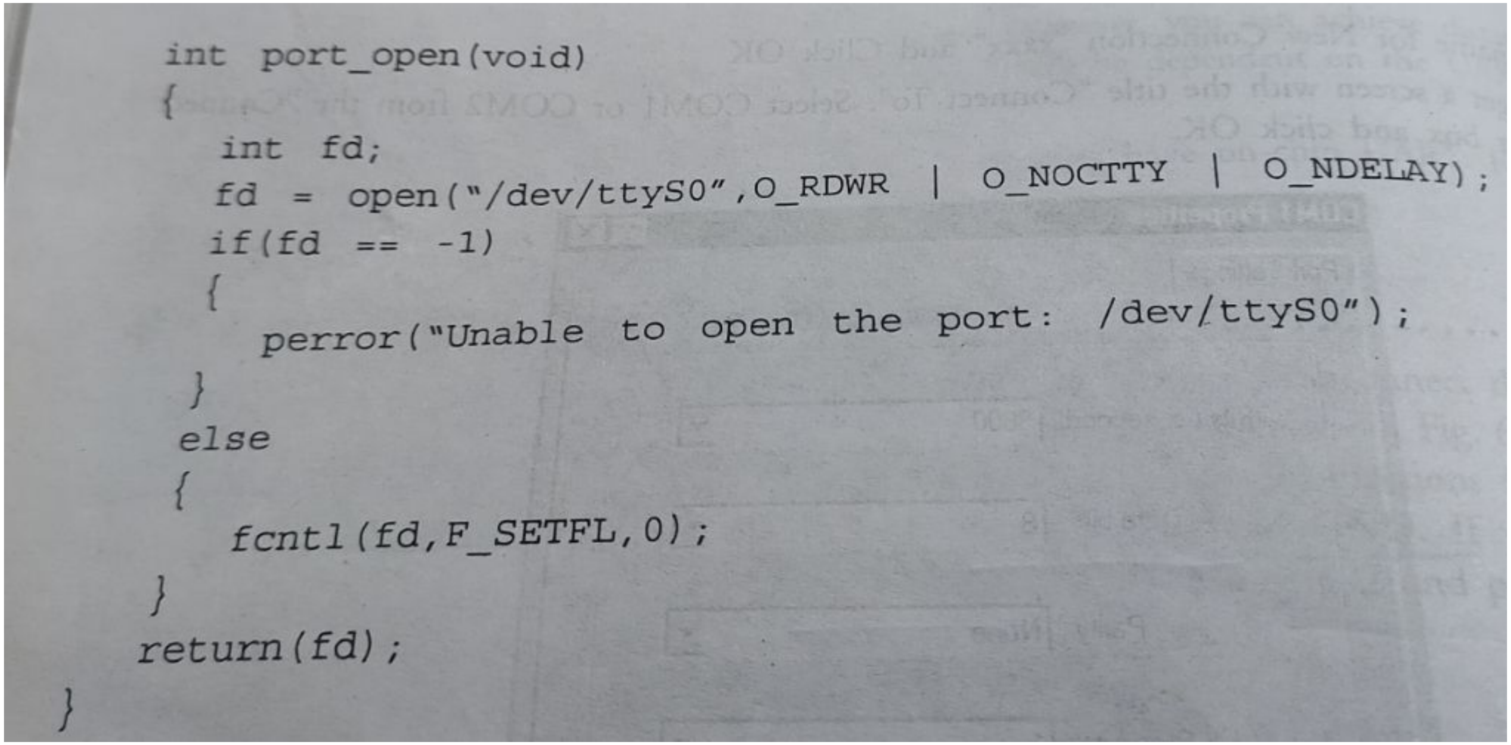
Flow control: Hardware

Terminal I/O settings

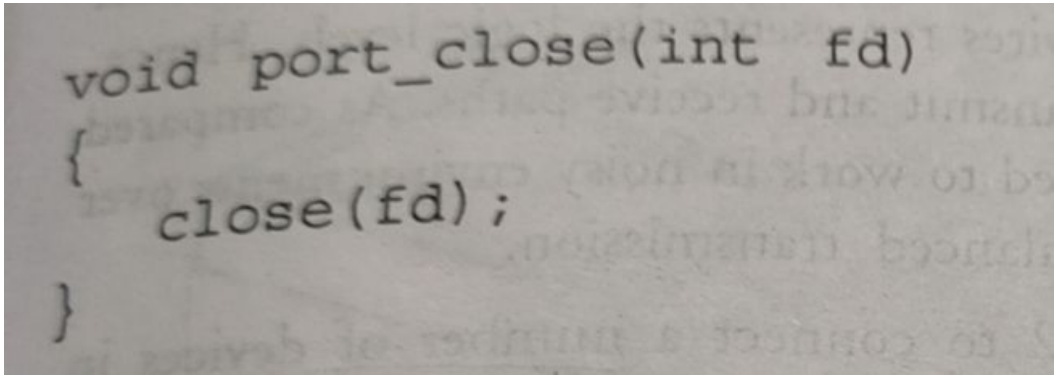


```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
```

Open the port

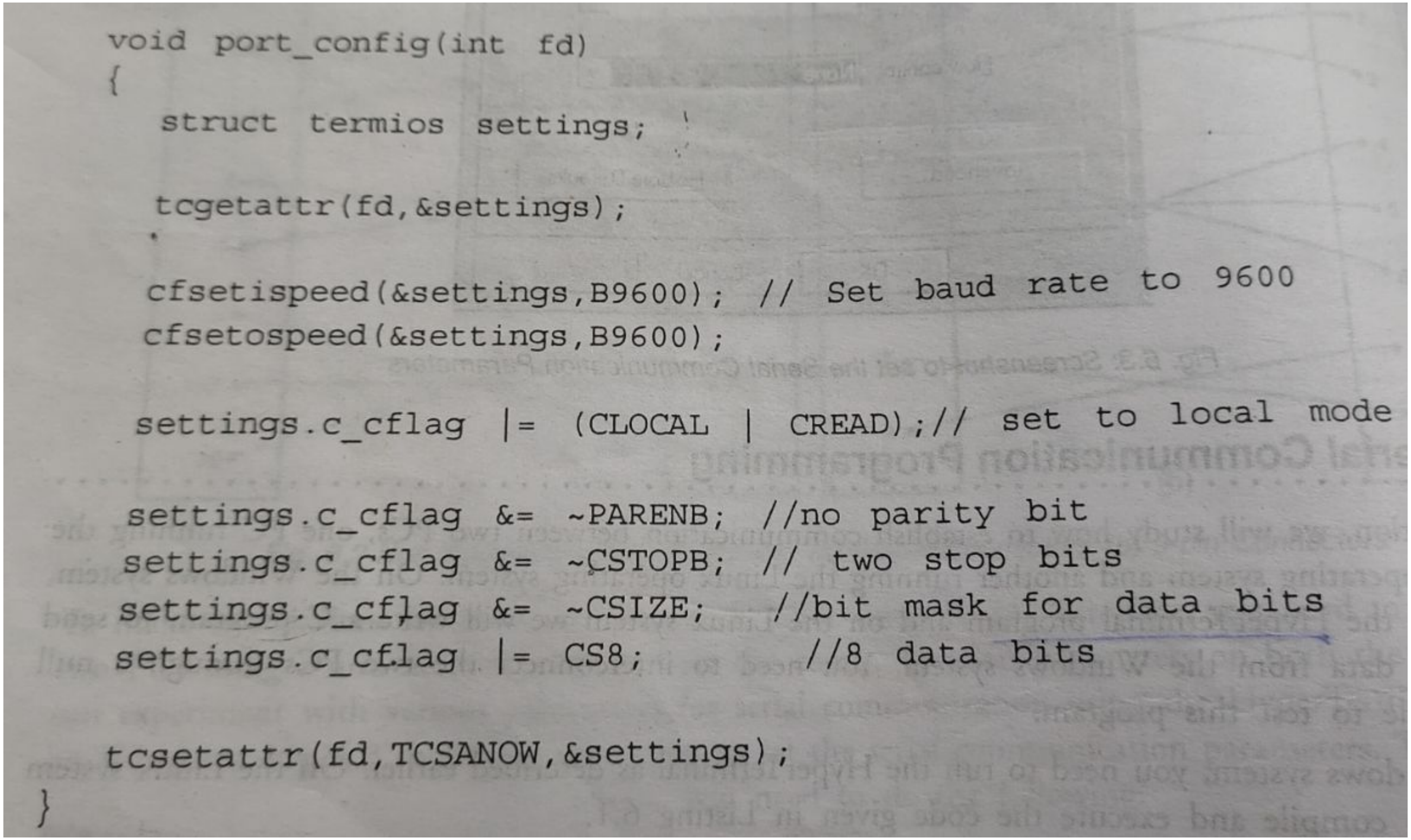


```
  int  port_open(void)
  {
     int  fd;
     fd  =  open(“/dev/ttyS0”,O_RDWR  |  O_NOCTTY  |  O_NDELAY);
     if(fd  ==  -1)
     {
        perror(“Unable  to  open  the  port:  /dev/ttyS0”);
     }
    else
    {
       fcntl(fd,F_SETFL,0);
    }
   return(fd);
}
```



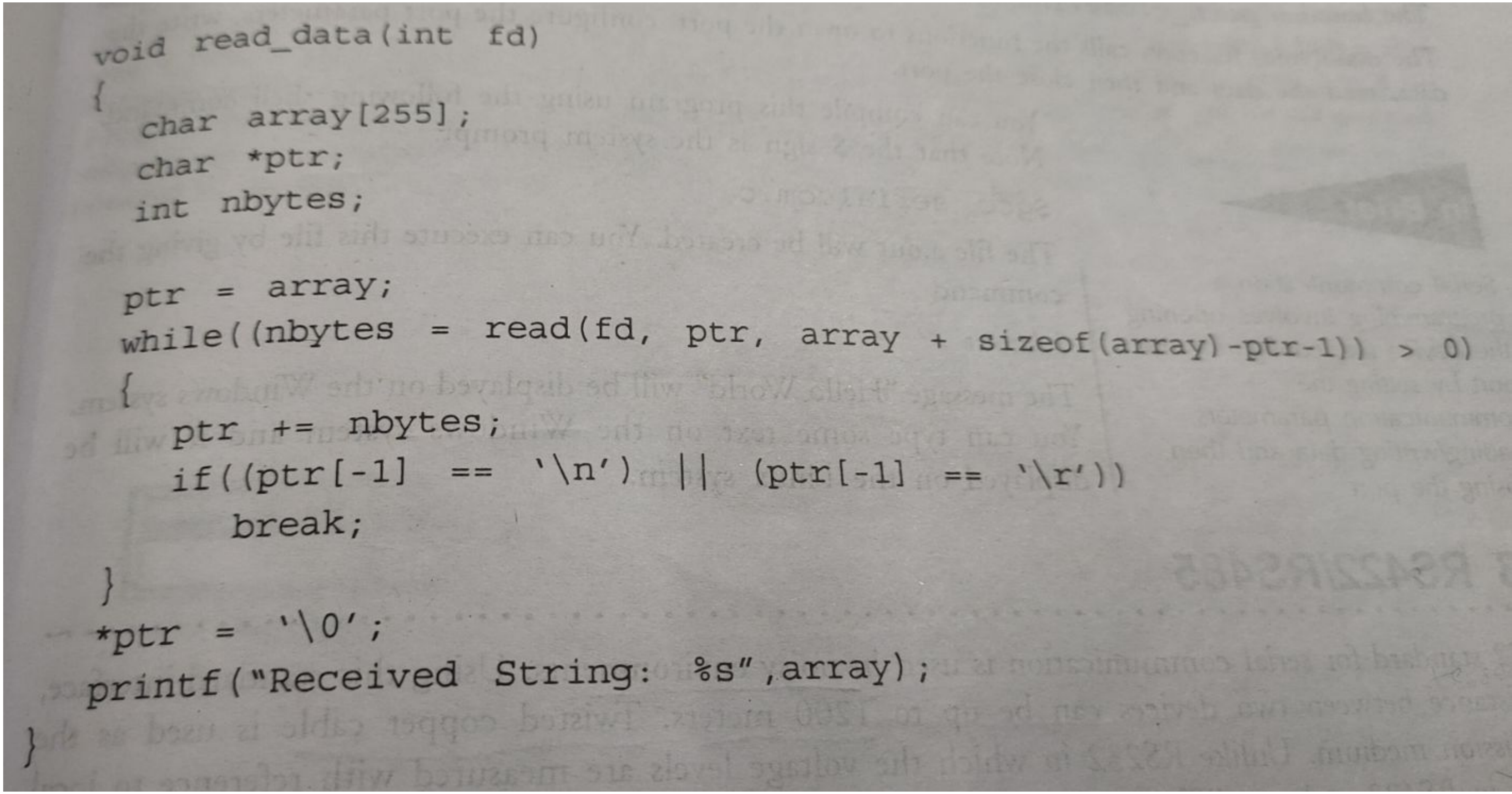
```
void  port_close(int  fd)
{
    close(fd);
}
```

Configuration Settings



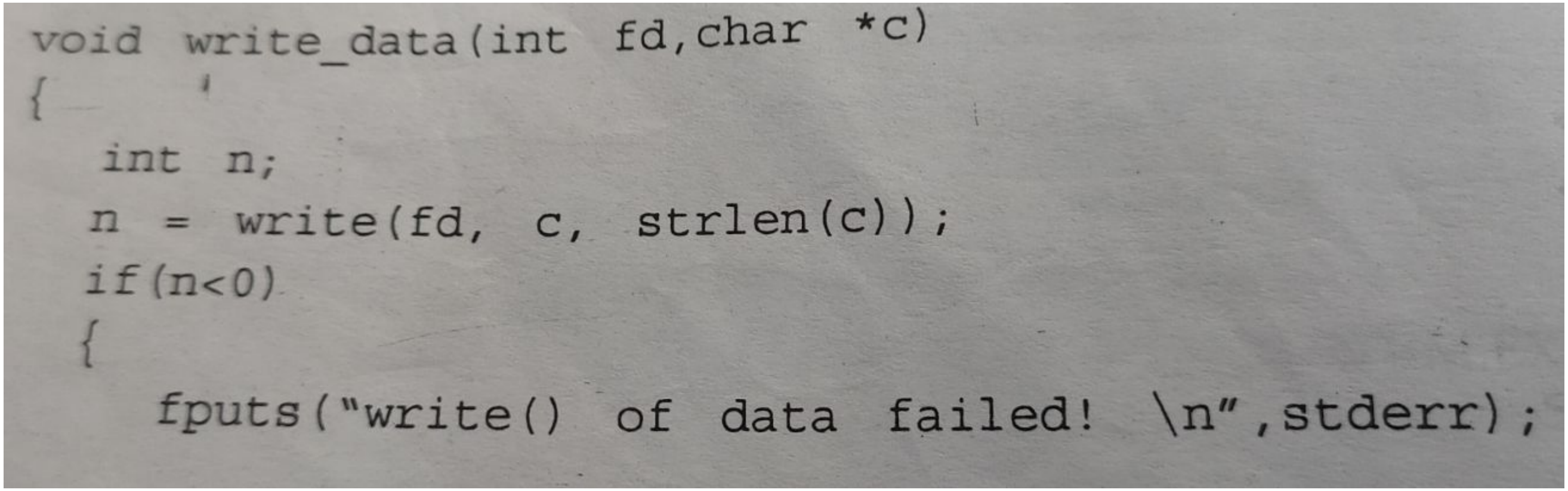
```
void port_config(int fd)
{
   struct termios settings;

   tcgetattr(fd,&settings);

   cfsetispeed(&settings,B9600); // Set baud rate to 9600
   cfsetospeed(&settings,B9600);

   settings.c_cflag |= (CLOCAL | CREAD);// set to local mode

  settings.c_cflag &= ~PARENB; //no parity bit
  settings.c_cflag &= ~CSTOPB; // two stop bits
  settings.c_cflag &= ~CSIZE;  //bit mask for data bits
  settings.c_cflag |= CS8;       //8 data bits

 tcsetattr(fd,TCSANOW,&settings);
}
```

Read the data



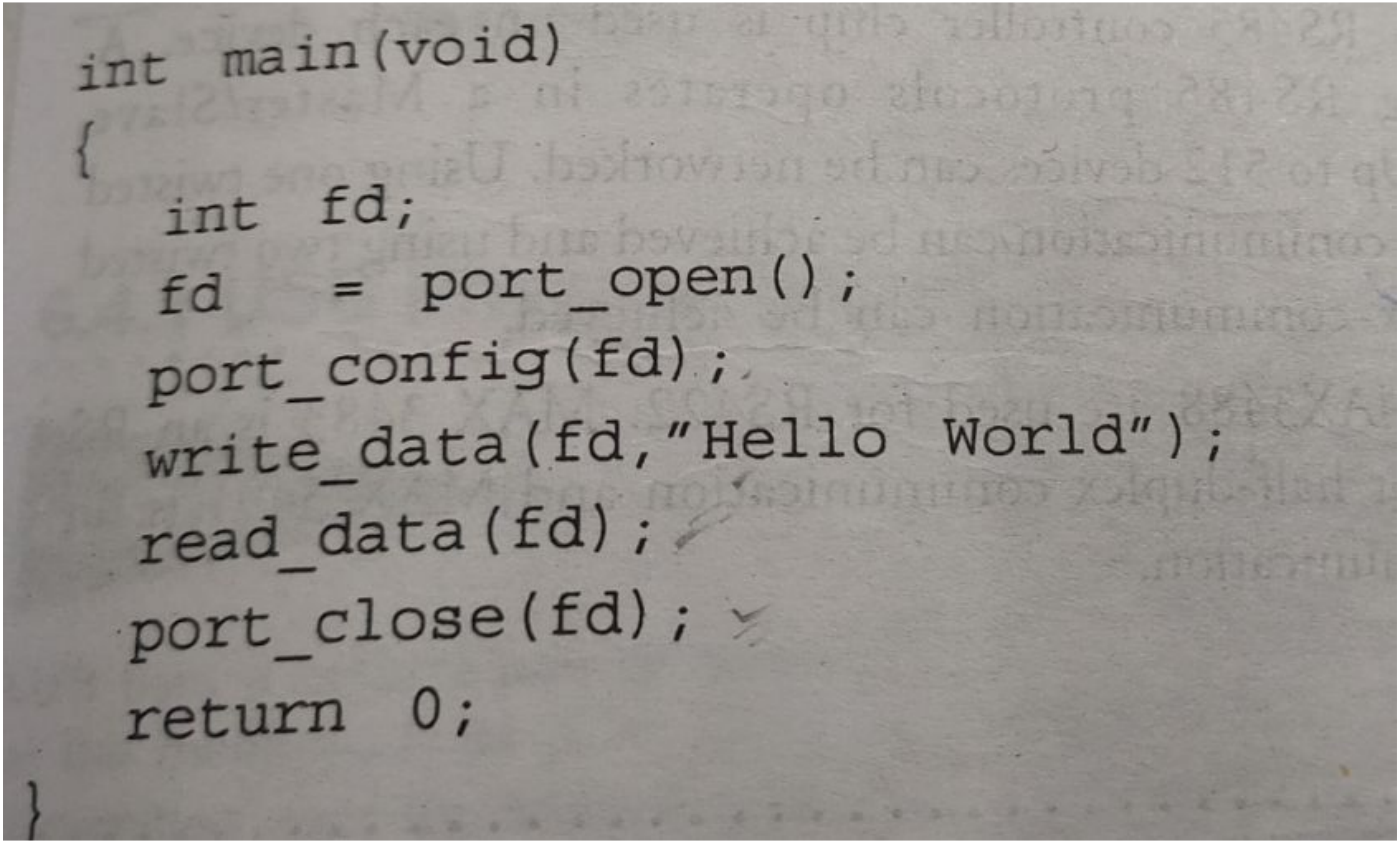
```
void read_data(int fd)
{
   char array[255];
   char *ptr;
   int nbytes;

  ptr = array;
  while((nbytes = read(fd, ptr, array + sizeof(array)-ptr-1)) > 0)
  {
     ptr += nbytes;
     if((ptr[-1] == ‘\n’) || (ptr[-1] == ‘\r’))
        break;
  }
 *ptr = ‘\0’;
printf(“Received String: %s”,array);
}
```

Write the data



```
void write_data(int fd,char *c)
{
  int n;
  n = write(fd, c, strlen(c));
  if(n<0)
  {
    fputs(“write() of data failed! \n”,stderr);
```

Serial Programming



```
int main(void)
{
    int fd;
    fd   = port_open();
    port_config(fd);
    write_data(fd,”Hello World”);
    read_data(fd);
    port_close(fd);
    return 0;
}
```

Control System for WindTurbine RS-232 Interface

- Measure
- Windspeed
- Power generated
- Faults

WindTurbine Protocol

Request Command

SOF (0x01)	COMMUNICATION ID (8 bytes)	MACHINE ID (8 bytes)	CMD_ID (2 bytes)	CHKSUM (2 bytes)	EOF (ox03)

```
TEMP_CMDID        = {0x44, 0x3a};
STATUS_CMDID       = {0x44, 0x3b};
PROG_CMDID        = {0x44, 0x3c};
FAULT_CMDID       = {0x44, 0x3d};
GRID_CMDID        = {0x44, 0x3e};
UP_CMDID          = {0x44, 0x3f};
START_CMDID       = {0x44,0x40};
STOP_CMDID        = {0x44,0x41};
DOWN_CMDID        = {0x44, 0x42};
YAWSTOP_CMDID      = {0x44, 0x44};
```

Example 1: Calculator Program

- Client / Server communication using RS-232 interface
- What is difference Checksum, CRC ?
- Calculator program
- Two PC – RS-232 interface
- PC 1 – client
- PC 2 - server
- Request Command: ADD, SUB, MUL, DIV
- Response Command: R-ADD (81), R-SUB(82), R-MUL(83), R-DIV(84)
- Design the protocol ???

Example 1: Calculator Program

- Sending: SoC SeqNo NoOfChar CmdID CmdParams CRC EoC

  1 1 2 4 1

- Response:
- SoC SeqNo NoOfChar RspCmdID ErrorFlag Result CRC EoC
- ErrrorFlag : 0 success 1 overflow 2 non
- Result – 2 bytes

Example 1: Calculator Program

- Start of Command (0x01) – 1 byte
- No of bytes to be transmitted – 1 byte
- Command ID : 2 bytes
  - ADD 01
  - SUB 02
  - MUL 03
  - DIV 04
- Command Parameters: 4 bytes
  - PARAM1 – value1 (example: 32)
  - PARAM2 – value2 (example : 10)
- End of Command (0x02) - 1 byte
- 01 32 10
- 42

Tic Tac Toe

Tic-Tac-Toe Protocol Message Format

- Request and Response Commands
- 3 *3 Grid
  - START the game
  - MOVE (place the X and O on the Grid)
  - END the Game – Win, Lose or Draw
  - Any other ?

REQUEST Commands

SOF	SEQUENCE_ID	COMMAND_ID	LEN_OF_DATA	DATA	CHKSUM	EOF
(0x01)	(1 byte)	(1 byte)	(1 byte)	(0 or more bytes)	(2 bytes)	(ox03)

START_COMMAND	0x01
START_RSP_COMMAND	0x81
MOVE_COMMAND	0x02
MOVE_RESP_COMMAND	0x81
END_COMMAND	0x03
END_RESP_COMMAND	0x83

RESPONSE Commands

SOF	SEQUENCE_ID	COMMAND_ID	LEN_OF_DATA	DATA	ERROR_CODE	CHKSUM	EOF
(0x01)	(1 byte)	(1 byte)	(1 byte)	(0 or more bytes)	(1 byte)	(2 bytes)	(ox03)

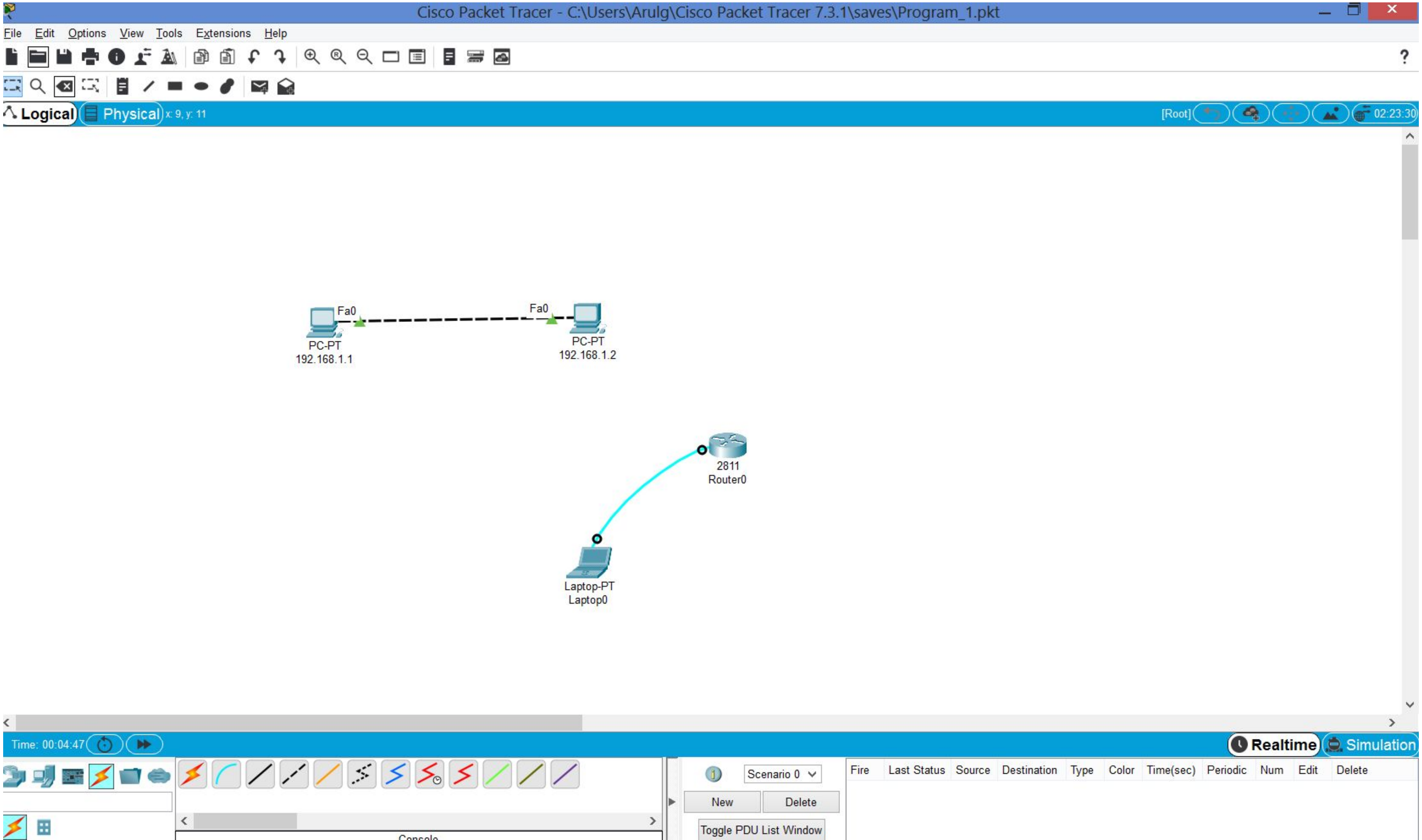
NO_ERROR	0x00	//no error
ERROR_1	0x01	//Checksum error
ERROR_2	0x03	//invalid move
ERROR_3	0x04	//invalid result

LOSE – 0x00
WIN – 0x01
DRAW – 0x02

API's used

- Open_port()
- Port_settings()
- receive_data()
- send_data()
- Parse_data() //implement a state machine
- Close_port()

Practical use of RS-232


Cisco Packet Tracer - C:\Users\Arulg\Cisco Packet Tracer 7.3.1\saves\Program_1.pkt
File
Edit
Options
View
Tools
Extensions
Help
Logical
Physical
x: 9, y: 11
[Root]
02:23:30
Fa0
Fa0
PC-PT
192.168.1.1
PC-PT
192.168.1.2
2811
Router0
Laptop-PT
Laptop0
Time: 00:04:47
Realtime
Simulation
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete
Console