Appendix A

Practical RS-232 programming

A.1 INTRODUCTION

RS-232 provides an excellent opportunity to investigate practical communications systems. It is a standard device on all new PC and has many practical applications, especially in remote communication, instrumentation and control.

The Intel 8250 IC is commonly used in serial communications. It is mounted onto the PC motherboard or fitted to an I/O card. This section discusses how it is programmed.

A.2 ISOLATED I/O

The RS-232 device(s) map into the PCs isolated I/O memory area, this is separate from the computers main memory. These isolated I/O locations store a single byte and are referred to as ports. Their addresses range from `0000h` to `FFFFh`. Table A.1 shows some of the standard devices which mapped into this memory. For example, the primary serial communications port (`COM1:`) maps into addresses `3F8h-3FFh`.

Table A.1: Typical isolated I/O memory map

Address	Device	Address	Device
000h-01Fh	DMA controller	278h-27Fh	Second parallel port (LPT2:)
020h-03Fh	Interrupt controller	2F8h-2FFh	Second serial port (COM2:)
040h-05Fh	Counter/timer	300h-31Fh	Prototype card
060h-07Fh	Digital I/O	378h-37Fh	Primary parallel port (LPT1:)
080h-09Fh	DMA controller	380h-38Ch	SDLC interface
0A0h-0BFh	NMI reset	3A0h-3AFh	Primary binary synchronous port
0C0h-0DFh	DMA controller	3B0h-3BFh	Monochrome display
0E0h-0FFh	Math co-processor	3D0h-3DFh	Colour/graphics adapter
1F0h-1F8h	AT fixed disk	3F0h-3F7h	Floppy disk controller
200h-20Fh	Game I/O adapter	3F8h-3FFh	Primary serial port (COM1:)
210h-217h	Expansion unit		

A.2.1 Inputting a byte

In Turbo/Borland C/C++, a byte can be read from a port using the `inportb()` function. The general syntax is:

```
value=inportb(PORTADDRESS);
```

where `PORTADDRESS` is the address of the input port and `value` is loaded with the 8-bit value from this address. This function is prototyped in the header file `dos.h`.

For Turbo Pascal the equivalent is accessed via the `port[]` array. Its general syntax is as follows:

```
value:=port[PORTADDRESS];
```

where `PORTADDRESS` is the address of the input port and `value` the 8-bit value at this address. To gain access to this function the statement `uses  dos` needs to be placed near the top of the program.

A.2.2 Outputting a byte

In Turbo/Borland C/C++, a byte can be outputted to a port using the `outportb()` function. The general syntax is

```
outportb(PORTADDRESS,value);
```

where `PORTADDRESS` is the address of the output port and `value` is the 8-bit value to be send to this address. This function is prototyped in the header file `dos.h`.

For Turbo Pascal the equivalent is accessed via the `port[]` array. Its general syntax is as follows:

```
port[PORTADDRESS]:=value;
```

where `PORTADDRESS` is the address of the output port and `value` is the 8-bit value to be sent to that address. To gain access to this function the statement `uses  dos` requires to be placed near the top of the program.

Note that some C compilers use the functions `inp()` and `outp()` instead of `inportb()` and `outportb()`. If this is the case either replace all the occurrences of these functions in this appendix with their equivalent or insert the following lines after the header files have been included.

```
#define inportb(portid)         inp(portid)
#define outportb(portid,ch)     outp(portid,ch)
```

This replaces all occurrences of `inportb()` and `outportb()` with `inp()` and `outp()`, respectively. In C hexadecimal constants are preceded by a `0` (zero) and the character 'x' (`0x`). In Pascal a dollar sign is used to signify a hex number, for example `$C4`.

A.3 PROGRAMMING THE SERIAL DEVICE

The main registers used in RS-232 communications are the Line Control Register (LCR), the Line Status Register (LSR) and the Transmit and Receive buffers (see Figure A.1). The Transmit and Receive buffers share the same addresses.

The base address of the primary port (`COM1:`) is normally set at 3F8h and the secondary port (`COM2:`) at 2F8h. A standard PC can support up to four COM ports. These addresses are set in the BIOS memory and the address of each of the ports is stored at address locations `0040:0000` (`COM1:`), `0040:0002` (`COM2:`), `0040:0004` (`COM3:`) and `0040:0008` (`COM4:`). Program A.1 can be used to identify these addresses. The statement:

```
ptr=(int far *)0x0400000;
```

initializes a far pointer to the start of the BIOS communications port addresses. Each address is 16 bits thus the pointer points to an integer value. A far pointer is used as this can access the full 1 MB of memory, a non-far pointer can only access a maximum of 64 kB.

Program A.1

```
#include <stdio.h>
#include <conio.h>
int  main(void)
{
int  far *ptr; /* 20-bit pointer */

   ptr=(int far *)0x0400000; /* 0040:0000   */ clrscr();

   printf("COM1: %04x\n",*ptr);
   printf("COM2: %04x\n",*(ptr+1));
   printf("COM3: %04x\n",*(ptr+2));
   printf("COM4: %04x\n",*(ptr+3));

   return(0);
}
```

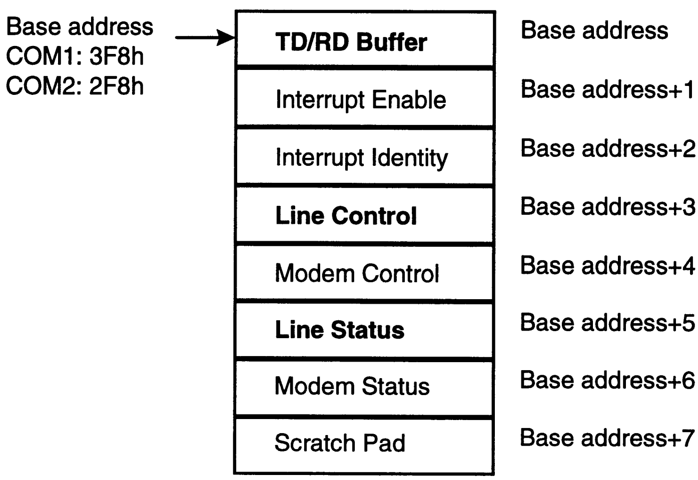

Figure A.1 Serial communication registers

Test run A.1 shows a sample run. In this case there are four COM ports installed on the PC. If any of the addresses is zero then that COM port is not installed on the system.

Test run A.1

```
COM1: 03f8
COM2: 02f8
COM3: 03e8
COM4: 02e8
```

A.3.1 Line Status Register (LSR)

The LSR determines the status of the transmitter and receiver buffers. It can only be read-from, and all the bits are automatically set by hardware. The bit definitions are given in Figure A.2. When an error occurs in the transmission of a character one (or several) of the error bits is (are) set to a '1'.

One danger when transmitting data is that a new character can be written to the transmitter buffer before the previous character has been sent. This overwrites the contents of the character being transmitted. To avoid this the status bit S_6 is tested to determine if there is still a character still in the buffer. If there is then it is set to a '1', else the transmitter buffer is empty.

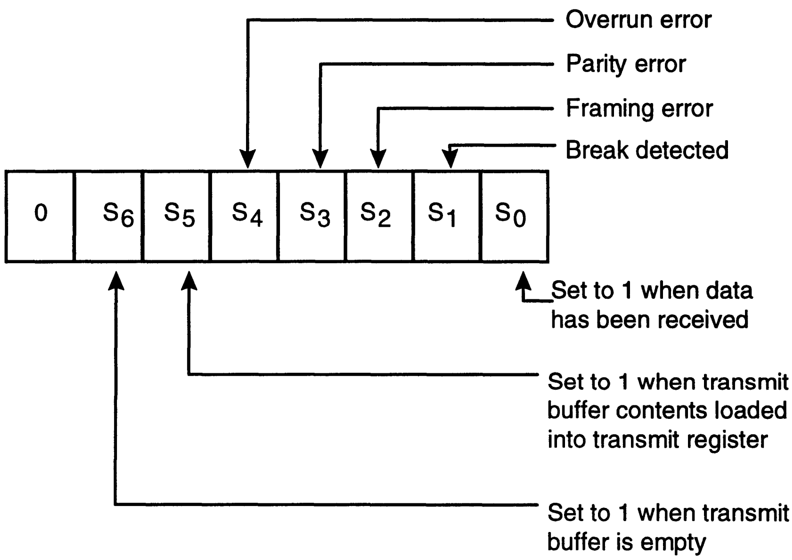

Figure A.2 Line Status Register

The routine to send a character is:

Test Bit 6 until set;
Send character;

The equivalent Turbo Pascal routine is:

```
repeat
    status := port[LSR] and $40;
until (status=$40);
```

When receiving data the S_0 bit is tested to determine if there is a bit in the receiver buffer. To receive a character:

Test Bit 0 until set;
Read character;

The equivalent Turbo Pascal routine is:

```
repeat
    status := port[LSR] and $01;
until (status=$01);
```

Figure A.3 shows how the LSR is tested for the transmission and reception of characters.

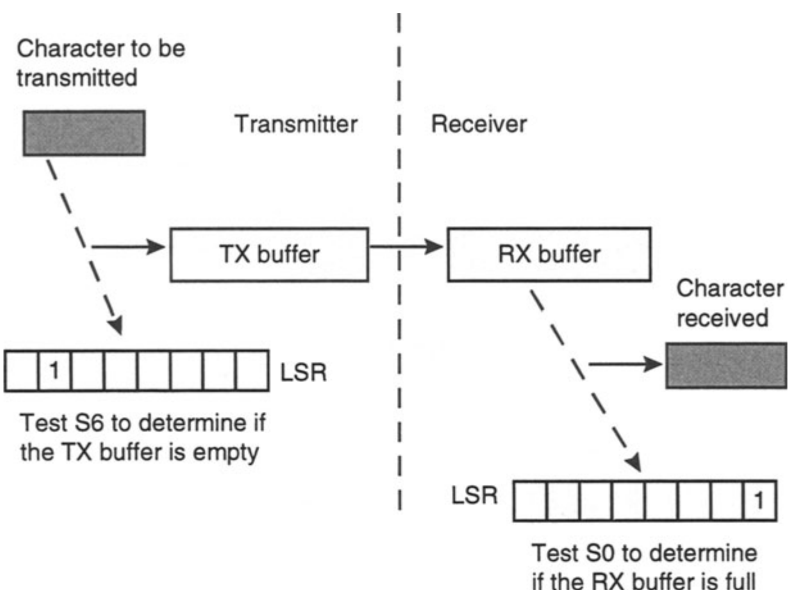

Figure A.3 Testing for the transmission and reception of characters

A.3.2 Line Control Register (LCR)

The LCR sets up the communications parameters. These include the number of bits per character, the parity and the number of stop bits. It can be written to or read from and has a similar function to that of the control registers used in other I/O devices. The bit definitions are given in Figure A.4.

The msb, C_7, must to be set to a '0' in order to access the transmitter and receiver buffers, else if it is set to a '1' the baud rate divider is set up. The baud rate is set by loading an appropriate 16-bit divisor into the addresses of transmitter/receiver buffer address and the next address. The value loaded depends on the crystal frequency connected to the IC. Table A.2 shows divisors for a crystal frequency is 1.8432 MHz. In general the divisor, *N*, is related to the baud rate by:

$$Baud\ rate = \frac{Clock\ frequency}{16 \times N}$$

For example, for 1.8432 MHz and 9600 baud $N = 1.8432\times10^6/(9600\times16) = 12$ (000Ch).

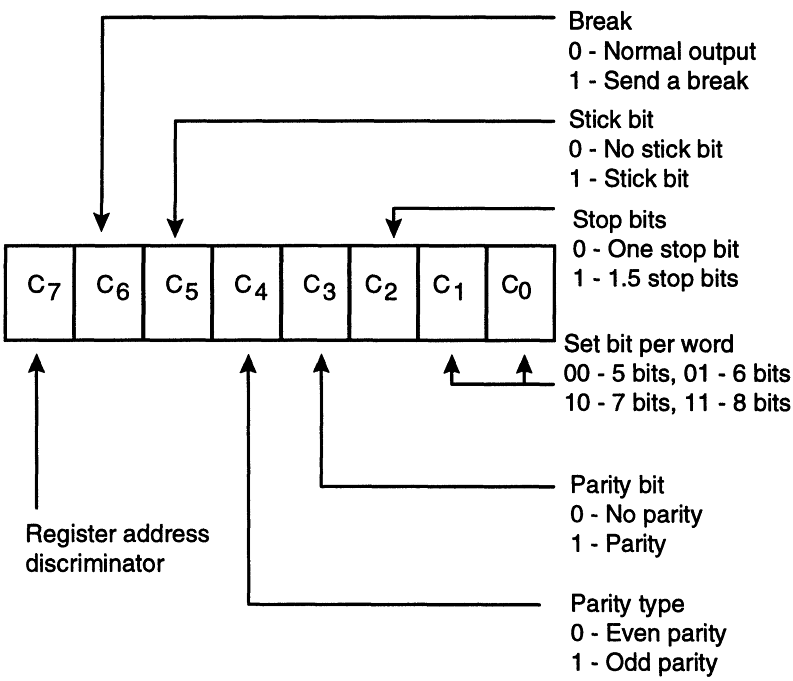

Figure A.4 Line Control Register

Table A.2 Baud rate divisors

Baud rate	*Divisor (value loaded into Tx/Rx buffer)*
110	0417h
300	0180h
600	00C0h
1200	0060h
1800	0040h
2400	0030h
4800	0018h
9600	000Ch
19200	0006h

A.3.3 Register addresses

The addresses of the main registers are given in Table A.3. To load the baud rate divisor, first the LCR bit 7 is set to a '1', then the LSB is loaded into divisor LSB and the MSB into the divisor MSB register. Finally, bit 7 is set back to a '0'. For example, for 9600 baud, `COM1` and 1.8432 MHz clock then `0Ch` is loaded in `3F8h` and `00h` into `3F9h`.

When bit 7 is set at a '0' then a read from base address reads from the RD buffer and a write operation writes to the TD buffer. An example is this is shown in Figure A.5.

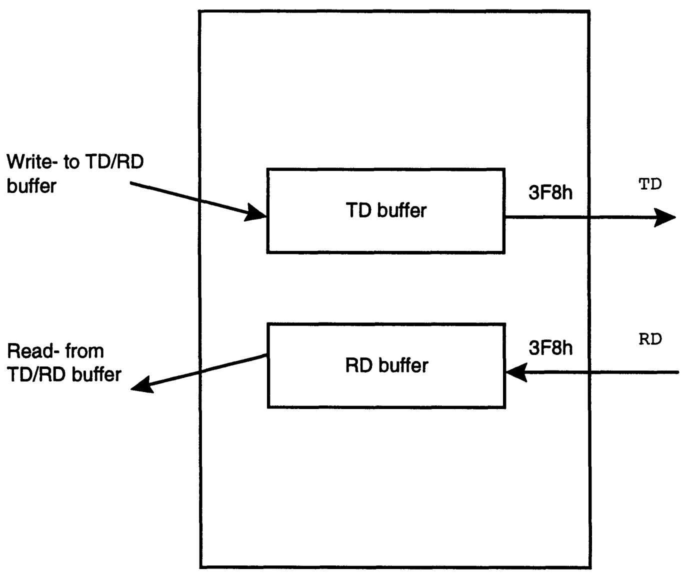

Figure A.5 Read and write-from TD/RD buffer

Table A.3 Serial communications addresses

Primary	Secondary	Register	Bit 7 of LCR
3F8h	2F8h	TD buffer	'0'
3F8h	2F8h	RD buffer	'0'
3F8h	2F8h	Divisor LSB	'1'
3F9h	2F9h	Divisor MSB	'1'
3FBh	2FBh	Line Control Register	
3FDh	2FDh	Line Status Register	

A.3.4 Programming RS232 via DOS

The DOS command `mode` (or `md` for DOS Version 6.0) can be used to set the parameters of the serial port. The general format is shown next, options in square brackets ([ ]) are optional.

```
MODE COMn[:]baud[, parity[ ,word_size[ ,stopbits[ ,P]]]]
```

The `mode` command can also be used for other functions such as setting up the parallel port, text screen, and so on.

```
C:\DOCS\NOTES>mode /?
Configures system devices.
MODE COMm[:][BAUD=b][PARITY=p][DATA=d][STOP=s][RETRY=r]
```

For example

```
C>  mode com2:2400,e,8,1
```

sets up COM2: with 2400 baud, even parity, 8 data bits and 1 stop bit.

```
C>  mode com1:9600
```

changes the baud rate to 9600 on COM1.

A.4 PROGRAMS

Program A.2 uses a loop back on the TD/RD lines so that a character sent by the computer will automatically be received into the receiver buffer and Program A.3 is the Turbo Pascal equivalent.

This set-up is useful in testing the transmit and receive routines. The character to be sent is entered via the keyboard. A *CNTRL-D* (^D) keystroke exits the program. Figure A.6 shows system connections for a 9- and 25-pin connector.

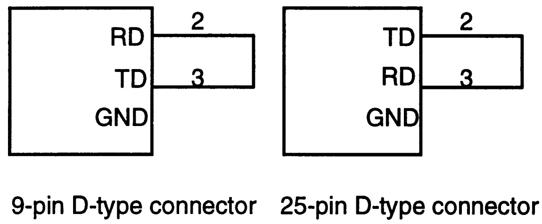

Figure A.6 System connections

Program A.2
```
/* This program transmits a character from COM1: and receives */
/* it via this port. The TD is connected to RD.               */

#define  COM1BASE 0x3F8
#define  COM2BASE 0x2F8
#define  TXDATA   COM1BASE
#define  LCR      (COM1BASE+3) /*  0x3FB line control     */
#define  LSR      (COM1BASE+5) /*  0x3FD line status      */

#include <conio.h> /* required for getch()                */
#include <dos.h>   /* required for inportb() and outportb() */
#include <stdio.h>

/* Some ANSI C prototype definitions  */
void   setup_serial(void);
void   send_character(int ch);
int    get_character(void);

int    main(void)
{
int    inchar,outchar;

  setup_serial();
  do
  {
    puts("Enter char to be transmitted (Cntrl-D to end)");
    outchar=getch();
    send_character(outchar);
    inchar=get_character();
    printf("Character received was %c\n",inchar);
  } while (outchar!=4);
  return(0);
}

void   setup_serial(void)
{
  outportb( LCR, 0x80);
  /* set up bit 7 to a 1  to set Register address bit */

  outportb(TXDATA,0x0C);
  outportb(TXDATA+1,0x00);
```

```
   /* load TxRegister with 12, crystal frequency is 1.8432MHz */

   outportb(LCR, 0x0A);
   /* Bit pattern loaded is 00001010b, from msb to lsb these are:   */
   /* 0 - access TD/RD buffer , 0 - normal output                   */
   /* 0 - no stick bit , 0 - even parity                            */
   /* 1 - parity on, 0 - 1 stop bit                                 */
   /* 10 - 7 data bits                                              */
}

void   send_character(int ch)
{
char   status;
   do
   {
     status = inportb(LSR) & 0x40;
   } while (status!=0x40);
   /*repeat until Tx buffer empty ie bit 6 set*/

   outportb(TXDATA,(char) ch);
}

int  get_character(void)
{
int  status;
   do
   {
     status = inportb(LSR) & 0x01;
   } while (status!=0x01);
   /* Repeat until bit 1 in LSR is set */

   return( (int)inportb(TXDATA));
}
```

Program A.3

```
program RS232_1(input,output);
{    This program transmits a character from COM1: and receives }
{    it via this port. The TD is connected to RD.                }
uses crt;
const  TXDATA =    $3F8;  LSR    =    $3FD;
       LCR    =    $3FB;  CNTRLD =    #4;
var    inchar, outchar:char;

procedure   setup_serial;
begin
  port[LCR] := $80;  { set up bit 7 to a 1                       }
  port[TXDATA] := $0C;
  port[TXDATA+1] := $00;
  { load TxRegister with 12, crystal frequency is 1.8432 MHz     }
  port[LCR] := $0A
  { Bit pattern loaded is 00001010b, from msb to lsb these are:}
  { Access TD/RD buffer, normal output, no stick bit            }
  { even parity, parity on, 1 stop bit, 7 data bits             }
end;

procedure   send_character(ch:char);
var      status:byte;
begin
```

```
    repeat
      status := port[LSR] and $40;
    until (status=$40);
      {repeat until bit Tx buffer is empty                       }
    port[TXDATA] := ord(ch); {send ASCII code                    }
  end;

  function    get_character:char;
  var       status,inbyte:byte;
  begin

    repeat
      status := port[LSR] and $01;
    until (status=$01);
    inbyte := port[TXDATA];
    get_character:= chr(inbyte);
  end;
  begin
    setup_serial;

    repeat
      outchar:=readkey;
      send_character(outchar);
      inchar:=get_character;
      writeln('Character received was ',inchar);
    until (outchar=CNTRLD);
  end.
```

In the next two programs a transmitter and receiver program are used to transmit data from one PC to another. Program A.4 should run on the transmitter PC and Program A.5 runs on the receiver. The cable connections for 9-pin to 25-pin, 9-pin to 9-pin and 25-pin to 25-pin connectors are given in Figure A.7.

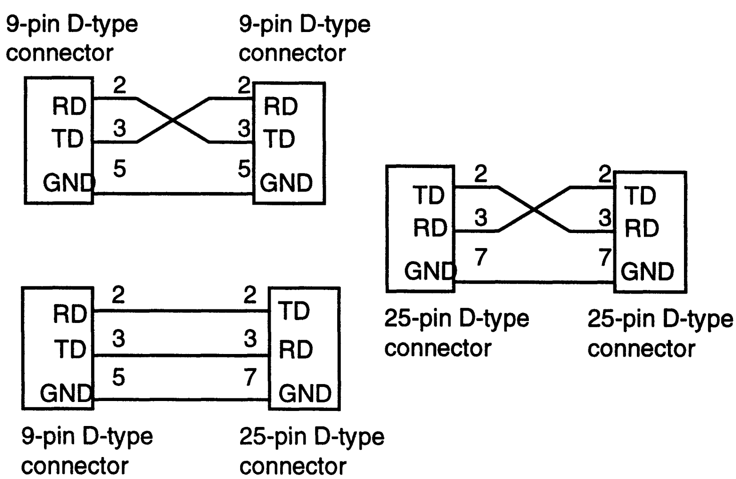

Figure A.7 Connection from one PC to another

Program A.4

```
/*   send.c                                              */
#define  TXDATA  0x3F8
#define  LSR     0x3FD
#define  LCR     0x3FB

#include  <stdio.h>
#include  <conio.h>   /* included for getch              */
#include  <dos.h>     /* included for inputb and outputb  */

void   setup_serial(void);
void   send_character(int ch);

int    main(void)
{
int    ch;

  puts("Transmitter program. Please enter text (Cntl-D to end)");
  setup_serial();
  do
  {
    ch=getche();
    send_character(ch);
  } while (ch!=4);

  return(0);
}

void setup_serial(void)
{

  outportb( LCR, 0x80);
  /* set up bit 7 to a 1  to set Register address bit        */

  outportb(TXDATA,0x0C);
  outportb(TXDATA+1,0x00);
  /* load TxRegister with 12, crystal frequency is 1.8432MHz */

  outportb(LCR, 0x0A);
  /* Bit pattern loaded is 00001010b, from msb to lsb these are:    */
  /* Access TD/RD buffer, normal output, no stick bit               */
  /* even parity, parity on, 1 stop bit, 7 data bits                */
}

void   send_character(int ch)
{
char   status;

  do
  {
    status = inportb(LSR) & 0x40;
  } while (status!=0x40);
  /*repeat until Tx buffer empty ie bit 6 set*/


  outportb(TXDATA,(char) ch);
}
```

Program A.5

```
/*   receive.c                                              */

#define  TXDATA  0x3F8
#define  LSR     0x3FD
#define  LCR     0x3FB

#include  <stdio.h>
#include  <conio.h>   /* included for getch                 */
#include  <dos.h>     /* included for inputb and outputb    */

void   setup_serial(void);
int    get_character(void);


int    main(void)
{
int    inchar;
setup_serial();
  do
  {
    inchar=get_character();
    putchar(inchar);
  } while (inchar!=4);
  return(0);

}
void setup_serial(void)
{
  outportb( LCR, 0x80);
  /* set up bit 7 to a 1  to set Register address bit        */

  outportb(TXDATA,0x0C);
  outportb(TXDATA+1,0x00);
  /* load TxRegister with 12, crystal frequency is 1.8432MHz      */

  outportb(LCR, 0x0A);
  /* Bit pattern loaded is 00001010b, from msb to lsb these are:  */
  /* Access TD/RD buffer, normal output, no stick bit             */
  /* even parity, parity on, 1 stop bit, 7 data bits              */
}

int  get_character(void)
{
int  status;

  do
  {
    status = inportb(LSR) & 0x01;
  } while (status!=0x01);

  /* Repeat until bit 1 in LSR is set */

  return( (int)inportb(TXDATA));
}
```

A.5 USING BIOS

The previous section discussed how the 8250 IC is programmed. Some machines may use a different IC, such as the 8251. An improved method of programming the RS-232 device is to use the BIOS commands. These are device independent and contain programs that can control the RS-232 hardware. The function used, in C, is

```
int bioscom(int cmd,char abyte,int port)
```

where

`port` corresponds to the port to use, 0 for `COM1:`, 1 for `COM2:`;

`cmd` 0 – set communications parameters to the value given by `abyte`;
1 – send a character;
2 – receive a character;
3 – return status of communications.

When `cmd` is set to 0 the device is programmed. In this mode the definition of the bits in `abyte` is given in Figure A.8. For example if the function call is `bioscom(0,0x42,0)` then the RS-232 parameters for `COM1:` will be 300 baud, no parity, 1 stop bit and 7 data bits.

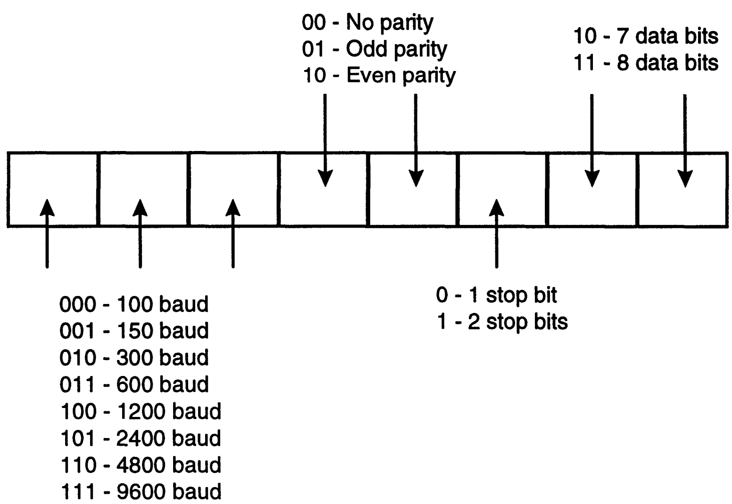

Figure A.8 Bit definitions for `bioscom()` function

If cmd is set to 3 then the return value from the function is a 16-bit unsigned integer. Bits 8 to 15 are defined as:

Bit 15 – Time out
Bit 14 – Transmit shift register empty (character sent)
Bit 13 – Transmit holding register empty
Bit 12 – Break detect
Bit 11 – Framing error
Bit 10 – Parity error
Bit 9 – Overrun error
Bit 8 – Data ready

Program A.6 uses bioscom() to create a transmit/receive program. To test it either loop the TD to the RD or connect two PCs together. The BIOS functions use the RTS and CTS lines in their operation. Thus, connect the RTS to the CTS on the transmitter and receiver or connect the RTS of the transmitter to the CTS of the receiver and the CTS on the transmitter to the RTS of the receiver (see Figure 8.21).

Program A.6

```
#include <stdio.h>
#include <bios.h>
#include <conio.h>

#define   COM1         0
#define   COM2         1

#define   DATA_READY   0x100
#define   DATABITS7    0x02
#define   DATABITS8    0x03

#define   STOPBIT1     0x00
#define   STOPBIT2     0x04

#define   NOPARITY     0x00
#define   ODDPARITY    0x08
#define   EVENPARITY   0x18

#define   BAUD110      0x00
#define   BAUD150      0x20
#define   BAUD300      0x40
#define   BAUD600      0x60
#define   BAUD1200     0x80
#define   BAUD2400     0xA0
#define   BAUD4800     0xC0
#define   BAUD9600     0xE0
```

```
#define   ESC           0x1B

int     main(void)
{
int     RS232_setting,status,in,ch;

   RS232_setting=BAUD2400 | STOPBIT1 | NOPARITY | DATABITS7;

   bioscom(0,RS232_setting,COM1);

   puts("RS-232 COMBIOS press ESC to exit");

    do
    {

      status = bioscom(3, 0, COM1);

      if (kbhit())
      {
        ch = getch();
        bioscom(1, ch, COM1); /* send character */
      }

      if (status & DATA_READY)
        if ((in = bioscom(2, 0, COM1) & 0x7F) != 0) /* receive char */
          putch(in);

    }  while (ch!=ESC);

    return 0;
}
```

A.6 TUTORIAL

A.1 Write a program that continuously sends the character 'A' to the serial line. Observe the output on an oscilloscope and identify the bit pattern and the baud rate.

A.2 Write a program that continuously sends the characters from 'A' to 'Z' to the serial line. Observe the output on an oscilloscope.

A.3 Complete Table A.4 to give the actual time to send 1000 characters for the given baud rates. Compare these values with estimated values.

Note that approximately 10 bits are used for each character thus 960 characters/sec will be transmitted at 9600 baud.

A.4 Modify Program A.2 or A.3 so that the program prompts the user for the baud rate when the program is started. A sample run is shown in Test run A.2.

Table A.4 Baud rate timing

Baud rate	*Time to send 1000 characters (sec)*
110	
300	
600	
1200	
2400	
4800	
9600	
19200	

Test run A.2

```
Enter baud rate required:
1  110
2  150
3  300
4  600
5  1200
6  2400
7  4800
8  9600
>> 8
RS232 transmission set to 9600 baud
```

A.5 Modify the `setup_serial()` routine so that the RS-232 parameters can be passed to it. These parameters should include the comport (either `COM1:` or `COM2:`), the baud rate, the number of data bits and the type of parity. An outline of the modified function is given in Program A.7.

Program A.7

```
#define   COM1BASE  0x3F8
#define   COM2BASE  0x2F8

#define   COM1      0
#define   COM2      1

enum   baud_rates  {BAUD110,BAUD300,BAUD600,BAUD1200,
                    BAUD2400,BAUD4800,BAUD9600};

enum   parity    {NO_PARITY,EVEN_PARITY,ODD_PARITY};

enum   databits {DATABITS7,DATABITS8};

#include <conio.h>
```

```
#include <dos.h>
#include <stdio.h>


/* Some ANSI C prototype definitions  */
void setup_serial(int comport, int baudrate, int parity,
                  int databits);
void send_character(int ch);
int  get_character(void);

int  main(void)
{
int  inchar,outchar;

   setup_serial(COM1,BAUD2400,EVEN_PARITY,DATABITS7);
   ::::::::::etc.

}


void setup_serial(int comport, int baudrate,
                  int parity, int databits)
{
int  tdreg,lcr;

   if (comport==COM1)
   {
      tdreg=COM1BASE;
      lcr=COM1BASE+3;
   }
   else
   {
      tdreg=COM2BASE;
      lcr=COM2BASE+3;
   }

   outportb( lcr, 0x80);
   /* set up bit 7 to a 1  to set Register address bit */

   switch(baudrate)
   {

   case BAUD110: outportb(tdreg,0x17);outportb(tdreg+1,0x04);
                 break;
   case BAUD300: outportb(tdreg,0x80);outportb(tdreg+1,0x01);
                 break;
   case BAUD600: outportb(tdreg,0x00);outportb(tdreg+1,0xC0);
                 break;
   case BAUD1200:
      outportb(tdreg,0x00);outportb(tdreg+1,0x40); break;
   case BAUD2400:
      outportb(tdreg,0x00);outportb(tdreg+1,0x30); break;
   case BAUD4800:
      outportb(tdreg,0x00);outportb(tdreg+1,0x18); break;
   case BAUD9600:
      outportb(tdreg,0x00);outportb(tdreg+1,0x0C); break;
   }
      :::::::::: etc.
}
```

A.6 One problem with Programs 7.4 and 7.5 is that when the return key is pressed only one character is sent. The received character will be a carriage return which returns the cursor back to the start of a line and not to the next line. Modify the receiver program so that a line feed will be generated automatically when a carriage return is received. Note a carriage return is an ASCII 13 and line feed is a 10.

A.7 Modify the `get_character()` routine so that it returns an error flag if it detects an error or if there is a time-out. Table A.5 lists the error flags and the returned error value. An outline of the C code is given in Program 7.10. If a character is not received within 10 seconds an error message should be displayed.

Program A.8

```
#include <stdio.h>
#include <dos.h>

#define  TXDATA  0x3F8
#define  LSR     0x3FD
#define  LCR     0x3FB

void   show_error(int ch);
int    get_character(void);

enum      RS232_errors  {PARITY_ERROR=-1, OVERRUN_ERROR=-2,
       FRAMING_ERROR=-3,  BREAK_DETECTED=-4, TIME_OUT=-5};

int    main(void)
{
int    inchar;
  do
  {
    inchar=get_character();

    if (inchar<0) show_error(inchar);
    else printf("%c",inchar);
  } while (inchar!=4);

  return(0);
}

void   show_error(int ch)
{
  switch(ch)
  {
  case PARITY_ERROR: printf("Error: Parity error/n"); break;
  case OVERRUN_ERROR: printf("Error: Overrun error/n"); break;
```

```
    case FRAMING_ERROR: printf("Error: Framing error/n"); break;
    case BREAK_DETECTED: printf("Error: Break detected/n");break;
    case TIME_OUT: printf("Error: Time out/n"); break;
    }
}

int  get_character(void)
{
int  instatus;
    do
    {
        instatus = inportb(LSR) & 0x01;
        if (instatus & 0x02) return(BREAK_DETECTED);
                            :::: etc
    } while (instatus!=0x01 );

    return( (int) inportb(TXDATA) );

}
```

Table A.5 Error returns from get_character()

Error condition	*Error flag return*	*Notes*
Parity error	–1	
Overrun error	–2	
Framing error	–3	
Break detected	–4	
Time-out	–5	`get_character()` should time-out if no characters are received with 10 seconds.

Test the routine by connecting two PCs together and set the transmitter with differing RS232 parameters.

A.7 PROJECTS

A.7.1 Project 1: Half-duplex link

Design and implement a half-duplex link between two computers, that is, only one computer can talk at a time. The same program should run on both computers but one should automatically go into talk mode when a key is pressed on the keyboard and the other as a listener. When the talker transmits an ASCII code 04 (^D) the mode of the computers should swap, that is, the talker should listen and the listener should talk. Figure A.9 shows a sample conversation.

A.7.2 Project 2: Full-duplex link

Design and implement a *simulated* full-duplex link between two computers, that is, both computers can talk and listen at the same time. The same program should run on both computers. *Hint*: one possible implementation is to loop within the `get_character()` routine and break out of it if a character is received from the line or if a character has been entered from the keyboard. If a character is entered from the keyboard it should be sent and the program then returns to the `get_character()` routine.

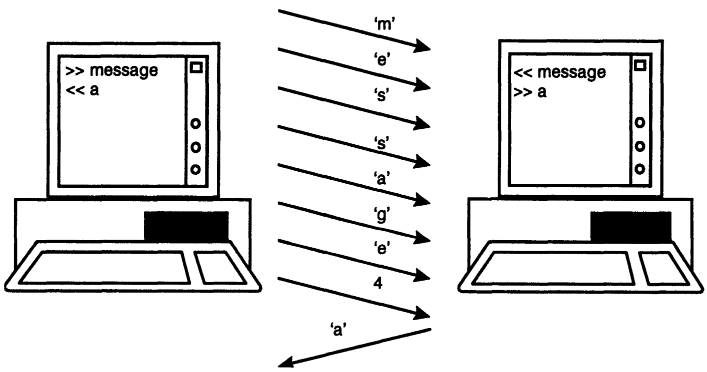

Figure A.9 System operation

A.7.3 Project 3: Simulated software handshaking

Set up two PCs so that one PC transmits characters to the other and the receiving PC displays them. If the space bar is pressed on the receiver PC then it should send an X-OFF character to the transmitter. The transmitter should then display a message informing the user that the receiver is busy. When the spacebar is pressed again on the receiver it should transmit an X-ON character and the transmitter is free to transmit more characters to the receiver. So it continues, with the receiver using the space bar to toggle its busy/idle state.

A.7.4 Project 4: File transfer

Design and implement a program that transfers a file between two PCs. The name of the file should be initially sent following a NULL character (to delimit the filename). Next, the contents of the file are sent and finally an EOF character.

Appendix B

Data communications standards

B.1 STANDARDS

Table B.1 lists some of standards relating to data communications. The CCITT (now known as the ITU) and the ISO are the main international standards organizations. The CCITT standard that relate to the transmission of data over telephone circuits are defined in the V. series, Packet-switched networks standards are defined in the X. series and Integrated Digital Services Network (ISDN) standards are defined in the I. series.

Table B.1: Data communications standards

ISO/CCITT standard	*Other standard*	*Description*
	ANSI X3T9.5	FDDI standard
CCITT I.430 CCITT I.431		Physical layer interface to an ISDN network
CCITT I.440 CCITT I.441		Data layer interface to an ISDN network
CCITT I.450/ CCITT I.451		Network layer interface to an ISDN network
CCITT V.10	EIA RS-423	Serial transmission up to 300 kbps/ 1200 m
CCITT V.11	EIA RS-422	Serial transmission up to 10 Mbps/ 1200 m
CCITT V.21		Full-duplex modem transmission at 300 bps
CCITT V.22		Half-duplex modem transmission at 600/1200 bps
CCITT V.22bis		Full-duplex modem transmission at 1200/2400 bps

CCITT V.23		Full-duplex modem transmission at 1200 bps and receive at 75 bps
CCITT V.24	EIA RS-232C	Serial transmission up to 20 kbps/ 20 m
CCITT V.25bis		Modem command language
CCITT V.27		Full-duplex modem transmission at 2400/ 4800 for leased lines
CCITT V.29		Full-duplex modem transmission at 9600 bps over leased lines
CCITT V.32		Full-duplex modem transmission at 4800/9600 bps
CCITT V.32bis		Full-duplex modem transmission at 7200, 12000 and 14400 bps
CCITT V.35	EIA RS-449	CCITT standard for the RS-449 interface
CCITT V.42		Error control protocol
CCITT X.21		Physical layer interface to connection for synchronous transfer on PSDN
CCITT X.25		Connection to a packet-switched network
ISO 8802.4	IEEE 802.2	Token passing in a token ring LAN network
ISO 8802.5	IEEE 802.3	Token ring topology

B.2 INTERNATIONAL ALPHABET NO. 5

ANSI defined a standard alphabet known as ASCII. This has since been adopted by the CCITT as a standard, known as IA5 (International Alphabet No. 5). The following tables define this alphabet in binary, as a decimal, as a hexadecimal value and as a character.

Binary	*Decimal*	*Hex.*	*Character*	*Binary*	*Decimal*	*Hex.*	*Character*
00000000	0	00	NUL	00010000	16	10	DLE
00000001	1	01	SOH	00010001	17	11	DC1
00000010	2	02	STX	00010010	18	12	DC2
00000011	3	03	ETX	00010011	19	13	DC3
00000100	4	04	EOT	00010100	20	14	DC4
00000101	5	05	ENQ	00010101	21	15	NAK
00000110	6	06	ACK	00010110	22	16	SYN
00000111	7	07	BEL	00010111	23	17	ETB
00001000	8	08	BS	00011000	24	18	CAN
00001001	9	09	HT	00011001	25	19	EM
00001010	10	0A	LF	00011010	26	1A	SUB
00001011	11	0B	VT	00011011	27	1B	ESC
00001100	12	0C	FF	00011100	28	1C	FS
00001101	13	0D	CR	00011101	29	1D	GS
00001110	14	0E	SO	00011110	30	1E	RS
00001111	15	0F	SI	00011111	31	1F	US

Binary	*Decimal*	*Hex.*	*Character*	*Binary*	*Decimal*	*Hex.*	*Character*
00100000	32	20	SPACE	00110000	48	30	0
00100001	33	21	!	00110001	49	31	1
00100010	34	22	"	00110010	50	32	2
00100011	35	23	£/#	00110011	51	33	3
00100100	36	24	$	00110100	52	34	4
00100101	37	25	%	00110101	53	35	5
00100110	38	26	&	00110110	54	36	6
00100111	39	27	/	00110111	55	37	7
00101000	40	28	(	00111000	56	38	8
00101001	41	29	)	00111001	57	39	9
00101010	42	2A	*	00111010	58	3A	:
00101011	43	2B	+	00111011	59	3B	;
00101100	44	2C	,	00111100	60	3C	<
00101101	45	2D	-	00111101	61	3D	=
00101110	46	2E	.	00111110	62	3E	>
00101111	47	2F	/	00111111	63	3F	?

Binary	*Decimal*	*Hex.*	*Character*	*Binary*	*Decimal*	*Hex.*	*Character*
01000000	64	40	@	01010000	80	50	P
01000001	65	41	A	01010001	81	51	Q
01000010	66	42	B	01010010	82	52	R
01000011	67	43	C	01010011	83	53	S
01000100	68	44	D	01010100	84	54	T
01000101	69	45	E	01010101	85	55	U
01000110	70	46	F	01010110	86	56	V
01000111	71	47	G	01010111	87	57	W
01001000	72	48	H	01011000	88	58	X
01001001	73	49	I	01011001	89	59	Y
01001010	74	4A	J	01011010	90	5A	Z
01001011	75	4B	K	01011011	91	5B	[
01001100	76	4C	L	01011100	92	5C	\
01001101	77	4D	M	01011101	93	5D	]
01001110	78	4E	N	01011110	94	5E	`
01001111	79	4F	O	01011111	95	5F	_

Binary	*Decimal*	*Hex.*	*Character*	*Binary*	*Decimal*	*Hex.*	*Character*
01100000	96	60		01110000	112	70	p
01100001	97	61	a	01110001	113	71	q
01100010	98	62	b	01110010	114	72	r
01100011	99	63	c	01110011	115	73	s
01100100	100	64	d	01110100	116	74	t
01100101	101	65	e	01110101	117	75	u
01100110	102	66	f	01110110	118	76	v
01100111	103	67	g	01110111	119	77	w
01101000	104	68	h	01111000	120	78	x
01101001	105	69	i	01111001	121	79	y
01101010	106	6A	j	01111010	122	7A	z
01101011	107	6B	k	01111011	123	7B	{
01101100	108	6C	l	01111100	124	7C	:
01101101	109	6D	m	01111101	125	7D	}
01101110	110	6E	n	01111110	126	7E	~
01101111	111	6F	o	01111111	127	7F	DEL

Appendix C

Data communications connections

C.1 RS-232C INTERFACE

Table C.1: RS-232C connections

9-pin D-type	25-pin D-type	Name	RS-232 name	Description	Signal Direction on DCE
	1		AA	Protective GND	
3	2	TXD	BA	Transmit Data	IN
2	3	RXD	BB	Receive Data	OUT
7	4	RTS	CA	Request to Send	IN
8	5	CTS	CB	Clear to Send	OUT
6	6	DSR	CC	Data Set Ready	OUT
5	7	GND	AB	Signal GND	
1	8	CD	CF	Received line signal detect	OUT
	9		-	RESERVED	-
	10		-	RESERVED	-
	11			UNASSIGNED	-
	12		SCF	Secondary Received Line Signal Detector	OUT
	13		SCB	Secondary Clear To Send	OUT
	14		SBA	Secondary Transmitted Data	IN
	15		DB	Transmission Signal Element Detector	OUT
	16		SBB	Secondary Received Data	OUT
	17		DD	Receiver Signal Element Time	OUT
	18			UNASSIGNED	-
	19		SCA	Secondary Request To Send	IN
4	20	DTR	CD	Data Terminal Ready	IN
	21		CG	Signal Quality Detector	OUT
9	22	RI	CE	Ring Indicator	OUT
	23		CH/CI	Data Signal Rate Selector	IN/OUT
	24		DA	Transmit Signal Element Timing	IN
	25			UNASSIGNED	-

C.2 RS-449 INTERFACE

RS-449 defines a standard for the function/mechanical interface for DTEs/DCEs for serial communications and is usually used with synchronous transmissions. Table C.2 lists the main connections.

Table C.2: RS-449 connections

Pin number	Mnemonic	Description
1		Shield
2	SI	Signalling Rate Indicator
3,21		Spare
4,22	SD	Sending Time
5,23	ST	Receive Data
6,24	RD	Receive Data
7,25	RS	Request to Send
8,26	RT	Receive Timing
9,27	CS	Clear To Send
10	LL	Local Loopback
11,29	DM	Data Mode
12,30	TR	Terminal Ready
13,31	RR	Receiver Ready
14	RL	Remote Loopback
15	IC	Incoming Call
16	SF/SR	Select Frequency/ Signalling Rate Select
17,37	TT	Terminal Timing
18	TM	Test Mode
19	SG	Signal Ground
20	RC	Receive Common
28	IS	Terminal in Service
32	SS	Select Standby
33	SQ	Signal Quality
34	NS	New Signal
36	SB	Standby Indicator
37	SC	Send Common

Appendix D

Ethernet voltages and Fast Ethernet

D.1 ETHERNET VOLTAGE LEVELS

Baseband Ethernet transmits onto a single ether at a rate of 10 Mbps. This gives a bit period of around 100 ns. When the ether is not busy the voltage on the ether is nominally +0.7 V which provides a carrier sense signal for all nodes on the network and is also known as the heartbeat. A low voltage is nominally –0.7 V.

When transmitting, a transceiver unit transmits a preamble of consecutive 1s and 0s. The coding used is Manchester coding which represents a 0 as a high to a low voltage transition and a 1 as a low to high transition. Thus when the preamble is transmitted the voltage changes between +0.7 and –0.7 V, as illustrated in Figure D.1. If after the transmission of the preamble no collisions are detected then the rest of the frame is sent.

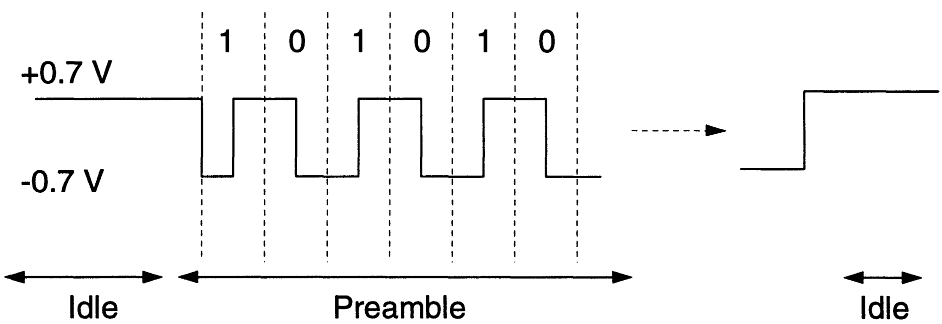

Figure D.1 Ethernet digital signal

Unfortunately, the ether and transceiver electronics are not perfect. The transmission line contains resistance and capacitance which distorts the shape of the pulse. There is also a delay period in the time that a pulse takes to travel along the line to all the nodes on a segment. The number of transceivers which connect to the segment also has an effect on the electrical loading on the ether. Figure D.2 shows a practical measurement of the voltages on an Ethernet segment with coaxial cable (that is, 10Base2). It shows that the pulses are rounded and do not have square edges, as the pulses in Figure D.1. Measurements also show that the idle voltage on the line is approximately +1 V and the pulse amplitude is ±0.8 V.

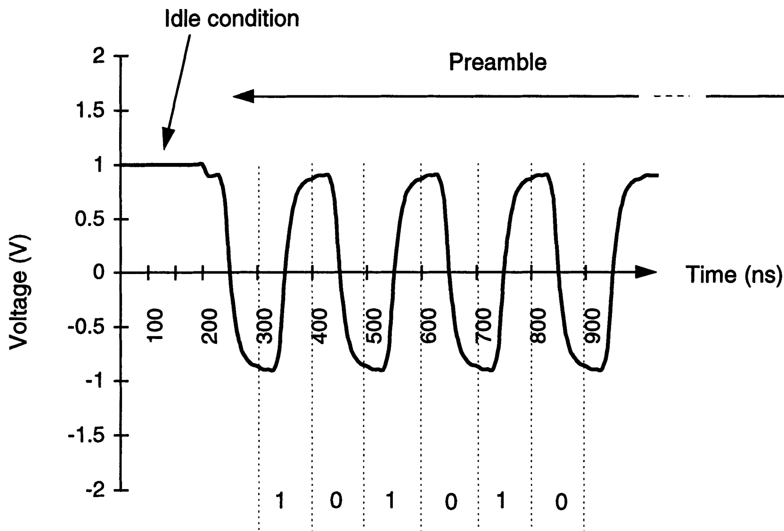

Figure D.2 Practical Ethernet voltages

The start of frame delimiter is identified by two consecutive logic 1s, this is shown in Figure D.3.

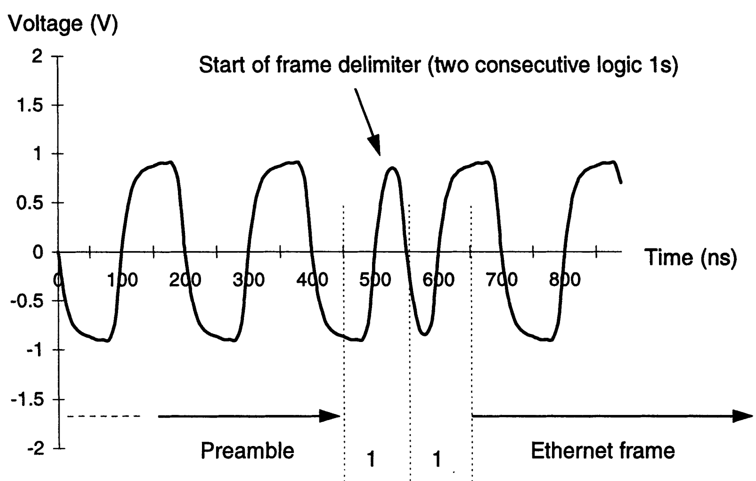

Figure D.3 Start of frame delimiter

D.2 FAST ETHERNET AND 100VG-ANYLAN

Standard 10 Mbps Ethernet does not performance well when many users running multi-media applications. Two improvements to the standard are Fast Ethernet and 100VG-AnyLAN. The IEEE have defined standards for these, IEEE 802.3u for Fast Ethernet and 802.12 for 100VG-AnyLAN. Both are supported by many manufacturers and use a bit rates of 100 Mbps. This gives, at least, ten times the performance of standard Ethernet.

D.2.1 Fast Ethernet

Fast Ethernet, or 100BASE-T, is simply 10BASE-T running at ten times the bit rate. It is a natural progression from standard Ethernet and thus allows existing Ethernet networks to be easily upgraded. Unfortunately, as with standard Ethernet, nodes contend for the network which reduces the network efficient when there are high traffic rates. Also, as it uses collision detect, the maximum segment length is limited by the amount of time for the farthest nodes on a network to properly detect collisions. On a Fast Ethernet network with twisted-pair copper cables this distance is 100 m and for a fibre optic link this is 400 m.

D.2.2 100VG-Any LAN

The 100VG-AnyLAN standard (IEEE 802.12) was developed mainly by Hewlett Packard and overcomes the contention problem by using a priority based, round robin arbitration method, known as Demand Priority Access Method (DPAM). Unlike Fast Ethernet, nodes always connect to a hub which regularly scans its input ports to determine it any nodes have requests pending.

It has an in-built priority mechanism with two priority levels: a high priority request and a normal priority request. A normal priority request is used for non-real time data, such as data files, and so on. High priority requests are used for real-time data, such as speech or video data. At present there is limited usage of this feature hub and there is no support mechanism for this facility after the data has left the hub.

100VG-AnyLAN allows up to seven level hubs (that is, one root and six cascaded hubs) with a maximum distance of 150 m between nodes.

Unlike, other forms of Ethernet it allows any number of node to be connected to a segment.

D.2.3 Migration from Ethernet to Fast Ethernet or 100VG-AnyLAN

If an existing network is based on standard Ethernet then, in most cases, the best network upgrade is either to Fast Ethernet or 100VG-AnyLAN. Since the protocols and access methods are the same there is no need to change any of network management software or application programs. The upgrade path for Fast Ethernet is simple and could be:

- Upgrade high data rate nodes, such as servers or high powered workstations to Fast Ethernet;
- Gradually upgrade NICs (Network Interface Cards) on Ethernet segments to cards which support both 10BASE-T and 100BASE-T. These cards automatically detect the transmission rate to give either 10 or 100 Mbps.

The upgrade path to 100VG-AnyLAN is less easy as it relies on hubs and, unlike Fast Ethernet, most NICs have different network connectors, one for 10BASE-T and the other for 100VG-AnyLAN (although it is likely that more NICs will have automatic detection). A possible path could be:

- Upgrade high data rate nodes, such as servers or high powered workstations to 100VG-AnyLAN;
- Install 100VG-AnyLAN hubs;
- Connect nodes to 100VG-AnyLAN hubs and change-over connectors.

D.2.4 Network performance

It is difficult to assess the performance differences between Fast Ethernet and 100VG-AnyLAN. Fast Ethernet uses a well proven technology but suffers from network contention. 100VG-AnyLAN is a relatively new technology and the handshaking with the hub increases delay time.

Glossary

Address	A unique label for the location of data or the identity of a communications device.
Address Resolution Protocol (ARP)	A TCP/IP process which maps an IP address to an Ethernet address.
American National Standards Institute (ANSI)	ANSI is a non-profit organization which is made up of expert committees that publish standards for national industries.
American Standard Code for Information Interchange (ASCII)	An ANSI-defined character alphabet which has since been adopted as a standard international alphabet for the interchange of characters.
Amplitude modulation (AM)	Information is contained in the amplitude of a carrier.
Amplitude-Shift Keying (ASK)	Uses two, or more, amplitudes to represent binary digits. Typically used to transmit binary data over speech-limited channels.
Application layer	The highest layer of the OSI model.
Asynchronous transmission	Transmission where individual characters are sent one-by-one. Normally each character is delimited by a start and stop bit. With asynchronous communication the transmitter and receiver only have to be roughly synchronized.
Bandwidth	The range of frequencies contained in a signal. As an approximation it is the difference between the highest and lowest frequency in the signal.
Baseband	Data transmission using unmodulated signals.
Baud rate	The number signalling elements sent per second with a RS-232, or modem, communications. In RS-232 the baud rate is equal to the bit-rate. With modems, two or more bits can be encoded as a single signalling element, such as 2 bits being represented by four different phase shifts (or one signalling element). The signalling element could change its amplitude, frequency or phase-shift to increase the bit-rate. Thus the bit-rate is a better measure of information transfer.

Bit stuffing	The insertion of extra bits to stop the appearance of a defined sequence. In HDLC the bit sequence 01111110 delimits the start and end of a frame. Bit stuffing stops this bit sequence from occurring anywhere in the frame by the receiver inserting a 0 whenever their are five consecutive 1's transmitted. At the receive if five consecutive 1's are followed by a 0 then the 0 is deleted.
Bridge	A device which physically links two or more networks using the same communications protocols, such as Ethernet/ Ethernet or token ring/ token ting.
Broadband	Data transmission using multiplexed data using an analogue signal or high-frequency electromagnetic waves.
Buffer	A temporary-storage space in memory.
Bus	A network topology where all nodes share a common transmission medium.
Byte	A group of eight bits, see octet.
Carrier Sense Multiple Access/ Carrier Detect (CSMA/CD)	A network where all nodes share a common bus. Nodes must contend for the bus and if a collision occurs then all colliding node back-off for a random time period.
CCITT	The Consultative Committee for International Telephone and Telegraph (now known at the ITU-TSS) is an advisory committee established by the United Nations. They attempt to establish standards for inter-country data transmission on a wide-wide basis.
Checksum	An error-detection scheme in which bits are grouped to form integer values and then each of the value is summated. Normally, the negative of this value is then added as a checksum. At the receiver, all the grouped values and the checksum are summated and, in the absence of errors, the result should be zero.
CRC	Cyclic Redundancy Check. An error-detection scheme. Used in most HDLC-related data link applications.
Cross-talk	Interference noise caused by conductors radiating electromagnetic radiation to couple into other conductors.

Data Communications Equipment (DCE)	Devices which establish, maintain and terminate a data communications conversation.
Data Terminal Equipment (DTE)	Devices at the end of the data communications connection.
Digital modulation	Method of converting digital data into a form which can be transmitted over a band-limited channel. Methods use either ASK, FSK, PSK or a mixture of ASK, FSK and PSK.
Direct Distance Dialling (DDD)	Allows modems to communicate directly without going through operator services.
Distributed system	A computer system in which computing, storage and other resources are dispersed throughout a network.
Electronic Industries Association (EIA)	US standards organization specializing in electrical interfaces.
Ethernet	A local area network which uses coaxial, twisted-pair or fibre optic cable as a communication medium. It transmits at a rate of 10 Mbps and was developed by DEC, Intel and Xerox Corporation. The IEEE 802.3 network standard is based upon Ethernet.
Ethernet address	A 48-bit number that identifies a node on an Ethernet network. Ethernet addresses are assigned by the Xerox Corporation.
Even parity	An error-detection scheme where defined bit-grouping have an even number of 1's.
Extended Binary Coded Decimal Interchange Code (EBCDIC)	An 8-bit code alphabet developed by IBM allowing 256 different bit patterns for character definitions.
Fibre Distributed Data Interface (FDDI)	The specification for a high-speed fibre-optic ring network.
File Transfer Protocol (FTP)	A protocol for transmitting files between host computers using the TCP/IP protocol.

Flow control	Procedure to regulate the flow of data between two nodes.
Frequency Modulation (FM)	Information is contained in the frequency of a carrier.
Frequency-Division Multiplexing (FDM)	Simultaneous transmission of several information channels using different frequencies for each channel.
Frequency-shift Keying (FSK)	Uses two, or more, frequencies to represent binary digits. Typically used to transmit binary data over speech-limited channels.
Full-Duplex (FDX)	Simultaneous, two-way communications.
Gateway	A device that connects networks using different communications protocols, such as Ethernet/FDDI or Ethernet/ token ring. It provides protocol translation, in contrast to a bridge which connects two networks that are of the same protocol.
Half-duplex (HDX)	Two-way communications, one at a time.
Handshaking	A reliable method for two devices to pass data.
High-Level Data Link Control (HDLC)	ISO standard for the data link layer.
Host	A computer that communicates over a network. A host can both initiate communications and respond to communications that are addressed to it.
Hub	A hub is a concentration point for data and repeats data from one node to all other connected nodes.
IEEE 802.2	A set of IEEE-defined specifications for Logical Link Control (LLC) layer. It provides some network functions and interfaces the IEEE 802.5, or IEEE 802.3, standards to the transport layer.
IEEE 802.3	A set of IEEE-specifications for CSMA/CD networks. It was developed by the IEEE 802.3 committee and has since been adopted by ANSI. Its specifications includes network protocol and hardware specifications.

IEEE 802.4	Token bus specifications.
IEEE 802.5	Token ring specifications.
Institute of Electrical and Electronic Engineers (IEEE)	An international professional society which issues standards.
International Telegraph Union Telecommunications Standards Sector (ITU-TSS)	Organization which has replaced the CCITT.
Internet	Connection of nodes on a global network which use a DARPA-defined Internet address.
internet	Two or more connected networks that may, or may not, use the same communication protocol. The device that connects the networks may be a router, bridge or a gateway.
Internet address	An address that conforms to the DARPA-defined Internet protocol. A unique, four byte number identifies a host or gateway on the Internet. This consists of a network number followed by a host number. The host number can be further divide into a subnet number. Internet addresses are normally expressed as four decimal numbers, ranging between 0-255, separated by periods.
ISO	International Standards Organization
Leased line	A permanent telephone line connection reserved exclusively by the leased customer. There is no need for any connection and disconnection procedures.
Light-emitting diode (LED)	A device which converts electrical current into light.
Line driver	A device which converts an electrical signal to a form that is transmittable over a transmission line. Typically, it provides the required power, current and timing characteristics.

Link layer	Layer 2 of the OSI model.
Link segment	A point-to-point link terminated on either side by a repeater. Nodes can not be attached to a link segment.
Local Area Network (LAN)	A data communications system that allows a number of independent devices to communicate.
Logical Link Control (LLC)	see IEEE 802.2
Manchester coding	Digital coding technique where each bit period is divided in a positive and a negative half. This helps to embed timing information in the transmitted signal.
Media Access Control (MAC)	Media-specific access-control for token ring and Ethernet.
Media Interface Controller (MIC)	The connector used to attach dual-attachment FDDI stations to a fibre-optic ring.
Medium Attachment Unit (MAU)	A device which connects nodes to the IEEE 802.3 network.
Modem (Modulator-Demodulator)	A device which converts binary digits into a form which can be transmitted over a speech-limited transmission channel.
Network Architecture	The organization of communications devices and their interconnection.
Network controller	In Ethernet, it is device which passes bit frames from the network and the local memory of the computer. Coupled with a network transceiver, it also handles signal processing, encoding and network media access.
Node	Any point in a network which provides communications services or where devices interconnect.
Octet	Same as a byte, a group of eight bits (typically used in communications terminology)
Odd parity	An error-detection scheme where a defined bit-grouping has an even number of 0's.

Optical Repeater	A device that receives, restores, and re-times signals from one optical fibre segment to another.
Packet	A sequence of binary digits that is transmitted as a unit in a computer network. A packet usually contains control information and data.
Phase-Locked Loop (PLL)	Converts FM signals, with a certain range, back into the unmodulated signal. The range at which the PLL 'picks-up' the FM signal is known as the capture range.
Phase-Shift Keying (PSK)	Uses two, or more, phase-shifts to represent binary digits. Typically used to transmit binary data over speech-limited channels.
Protocol	A specification for coding of messages exchanged between two communications processes.
Repeater	A device that receives, restores, and re-times signals from one segment of a network and passes them on to another. Both segments must have the same type of transmission medium and share the same set of protocols. A repeater cannot translate protocols.
RJ-45	Connector used with US telephones and with twisted-pair cables. It is also used in ISDN networks.
Routing node	A node that transmits packets between similar networks. A node that transmits packets between dissimilar networks is called a gateway.
RS-232C	EIA-defined standard for serial communications.
RS-422, 423	EIA-defined standard which uses a higher transmission rates and cable lengths than RS-232.
RS-449	EIA-defined standard for the interface between a DTE and DCE for 9- and 37-way D-type connectors.
RS-485	EIA-defined standard which is similar to RS-422 but uses a balanced connection.
Segment	A segment is any length of LAN cable terminated at both

	ends. In a bus network, segments are electrically continuous pieces of the bus, connected at by repeaters.
Simplex	One-way communication.
Synchronous transmission	Transmission where the transmitter and receiver synchronize their to each other. The bits are sent at a fixed rate which means that no start and stop bits are required (see asynchronous communications)
TCP/IP Internet	An Internet is made up of networks of nodes that can communicate with each other using TCP/IP protocols.
Time-Division Multiplexing (TDM)	Simultaneous transmission of several information channels using different time-slots for each channel.
Token	A token transmits data around a token ring network.
Topology	The physical and logical geometry governing placement of nodes on a network.
Transceiver	A device that transmits and receives signals.
Transmission Control Protocol and Internet Protocol (TCP/IP)	A standard protocol, defined by the Defence Advanced Research Projects Agency (DARPA), to allow different host computers to communicate over a variety of network types.
V.24	CCITT-defined specification, similar to RS-232C.
World-Wide Web (WWW)	The interconnection of networks on the Internet.
X-ON/ X-OFF	The Transmitter On/ Transmitter Off characters are used to control the flow of information between two nodes.
X.21	CCITT-defined specification for the interconnection of DTEs and DCEs for synchronous communications.
X.25	CCITT-defined specification for packet-switched network connections.

Common abbreviations

AA	auto answer
ABM	asynchronous balanced mode
AC	access control
ACK	acknowledge
ADC	analogue-to-digital converter
ADPCM	adaptive pulse code modulation
AFI	authority and format identifier
AM	amplitude modulation
AMI	alternative mark inversion
ANSI	American National Standard Institute
ARM	asynchronous response mode
ASCII	American standard code for information exchange
ASK	amplitude-shifting keying
AT	attention
ATM	asynchronous transfer mode
BCD	binary coded decimal
BIOS	basic input/output system
bps	bits per second
CASE	common applications service elements
CCITT	International Telegraph and Telephone Consultative Committee
CD	carrier detect
CPU	central processing unit
CRC	cyclic redundancy
CRT	cathode ray tube
CSDN	circuit-switched data network
CSMA	carrier sense multiple access
CSMA/CA	CSMA with collision avoidance
CSPDN	circuit-switched public data network
CTS	clear to send
DA	destination address
DAC	digital-to-analogue convertor
DARPA	Defence Advanced Research Projects Agency
dB	decibel
DC	direct current
DCD	data carrier detect
DCE	data circuit-terminating equipment
DNS	domain name server
DPSK	differential phase-shift keying
DR	dynamic range
DSP	domain specific part
DTE	data terminal equipment

DTR	data terminal ready
EaStMAN	Edinburgh/ Stirling MAN
EBCDIC	extended binary coded decimal interchange code
ENQ	enquiry
EOT	end of transmission
ETB	end of transmitted block
ETX	end of text
FAX	facsimile
FC	frame control
FCS	frame check sequence
FDDI	fibre distributed data interface
FDM	frequency division multiplexing
FEC	forward error control
FM	frequency modulation
FSK	frequency-shift keying
FTP	file transfer protocol
GFI	group format identifier
GUI	graphical user interface
HDB3	high-density bipolar code no. 3
HDLC	high-level data link control
HF	high frequency
Hz	Hertz
I/O	input/output
IA5	international alphabet no. 5
ICP	interconnection protocol
IDI	initial domain identifier
IDP	initial domain part
IEEE	Institute of Electrical and Electronic Engineers
ILD	injector laser diode
IP	internet protocol
ISDN	integrated services digital network
ISO	International Standards Organization
ITU	International Telecommunications Union
JANET	joint academic network
LAN	local area network
LAPB	link access procedure balanced
LCN	logical channel number
LED	light emitting diode
LGN	logical group number
LLC	logical link control
MAC	media access control
MAN	metropolitan area network
NAK	negative acknowledge
NSAP	network service access point

OH	off-hook
OSI	open systems interconnection
PA	point of attachment
PC	personal computer
PCM	pulse-coded modulation
PDN	public data network
PPSDN	public packet-switched data network
PSDN	packet-switched data network
PSE	packet switched exchange
PSK	phase-shift keying
PSR	packet switched router
PSTN	public-switched telephone network
QAM	quadrature amplitude modulation
RD	receive data
SAPI	service access point identifier
SD	sending data
SDLC	synchronous data link control
SEL	selector/extension local address
SNA	systems network architecture (IBM)
SNR	signal-to-noise ratio
STM	synchronous transfer mode
TCP	transmission control protocol
TDM	time-division multiplexing
TEI	terminal equipment identifier
TR	transmit data
VCI	virtual circuit identifier
WAN	wide area network
WIMPs	windows, icons, menus and pointers

Index

C

D

E

I

L

Q

R

S

Answers

1.1 B
1.2 D
1.3 C
1.4 D
1.5 B
1.6 A
1.7 D
1.8 A
1.8 C
1.9 A
1.10 B
1.11 B
2.1 Data link, Network, Transport, Data link, Session, Physical, Prestation
2.2 A
2.3 C
2.4 B
2.5 D
2.6 A
2.7 A
2.8 C
3.1 C
3.2 D
3.3 A
3.4 B
3.5 B
3.6 C
3.7 D
3.8 A
3.9 C
3.10 B
3.11 B
3.12 D
3.13 C
3.14 C
3.15 B
3.16 A
3.17 B
3.18 A
3.19 D
3.28 3F5F:8850:087F
5.1 D
5.2 A
5.3 B
5.4 C
5.5 140.113.1.9 (Type B), 64.125.65.233 (Type A), 174.113.29.57 (Type B)
6.1 C
6.2 A
6.3 C
6.4 C
6.5 A
6.6 B
6.7 C
6.8 B
6.15 10001100, 00001000, 1111101100011111
7.1 B
7.2 A
7.3 D
7.4 A
7.5 C
7.6 B
7.7 C
7.8 B
7.9 A
8.1 D
8.2 C
8.3 C
8.4 A
8.5 C
8.6 C
8.7 B
8.8 C
8.9 A
8.10 C
8.11 samX yrreM
10.1 40 kHz
10.2 60.2 dB
10.3 14 bits
10.4 0.488 μs
10.5 (i) 24 channel in 125 μs gives 1.536 Mbps
(ii) $8\times24+1$ bits in 125 μs gives 1.544 Mbps
10.6 0.648 μs
10.7 20 dB
11.2 0.4 ms
13.2 4 bits
13.3 00000000000 11000110010 01101001111 11111111111
13.6 Hello world

Refer to WWW page for more information.