Single Board Computer Network

Asist.ing. Mihai Timiş, Ing. Cătălin Galu, Ing. Adrian Cosmici
Technical University "Gh.Asachi" Iaşi

Abstract: This system can be used for control different types of systems like: mineral industry, petrol industry, pharmaceutical industry, cars industry, industrial robots, alarm systems. His performances are: 1000m direct communication, 32 networks nodes connected, low implementation prices.

Using the 80C51 kernel we are using SM75176BP circuits which support RS485 transmission protocol.

We are using this circuit because its designed for bidirectional data communication on multipoint bus transmition lines. They are designed for balance transmission line and meet ANSI standards, EIA/TIA-422-B&RS-485.

SN75176BP combine a 3-state differential line driver and a differential input line receiver.

The driver and receiver have active – high and active-low enables respectively, that can be externally connected together to function as a direction control.

The driver differential outputs and the receiver differential inputs are connected internally to form differential input-output (I/O) bus ports.

We have used RS485 transmission protocol because the differential data transmission is ideal for transmiting at high data rates, overlong distance and thrugh noicy environment.

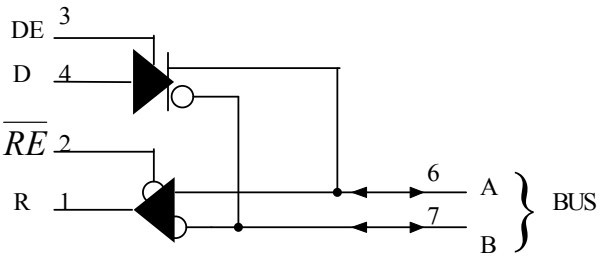

Figure 1

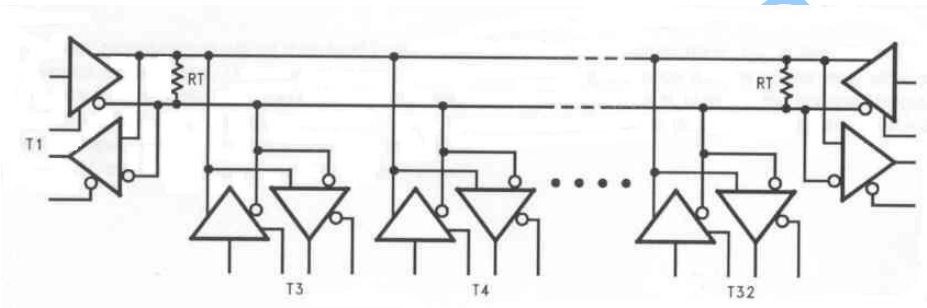

Figure 2

Using this network tophology and the below schematic,

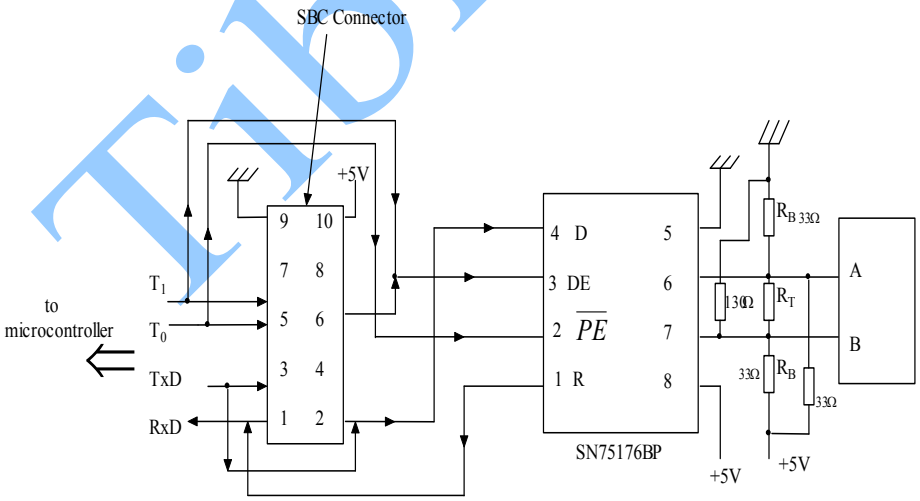

Figure 3

we will develop a SBS network with four 80C51 microcontrollers, like the schematic below:

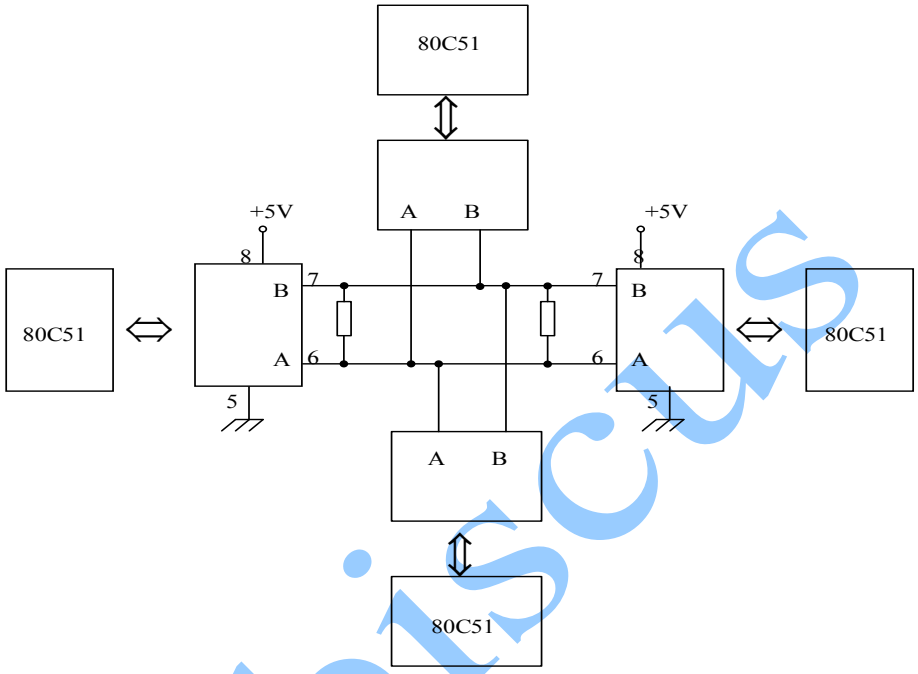

Figure 4

We are looking for realize SBC networks which permits safe data transmision on huge distance using half-duplex buss and only two wires. We have two comunication types:

- Master
- Slave

In the Master – Slave comunication the conflicts problem didn't appear because there are only one emitter and only one receiver. The problems apears in Master-Slave comunications because bus conflicts. There are many methods for resolve that conflicts.

Resolving methods for bus conflicts (collisions)

A collision appears when two or more microcontrollers emits in the same moments.

So, when two or more microcontrollers emits one octet (1 byte) on the bus when the all other listen the network.

One method for resolving this collision could be done using a flexible circular priority system. This method consists in repeat a rotation command at the end of every data packet transmission. It consists in using the 0x55h constant which determine a priority low level of each microcontroller on the network. For "N" microcontrollers, after "N" transmission packets the master microcontroller became the slave microcontroller.

For medium networks, this problem could be resolved by using a fixed system priority.

So, after the internal collision detection, the emitted test code is different than received code.

The emitted test code will be suspended during a precised time who will be the same on the entire system. After this time the transmission could be overloaded.

The microcontrollers will be on the next level priority:

- the master microcontroller will emit after 2*T time period
- the slave microcontroller will emit after 2*N*T time period, where N=number of the networks node(microcontrollers), T=transimtion time of one data octet,

$$T=104us*10=1.04ms \Leftarrow \frac{9600}{|1 \text{ bit start}|\ |8 \text{ biti date}|\ |1 \text{ bit stop}|}.$$

Determination of buss ocupation

We will consider the folowing rules:

1. Determination of buss occupation will be done only by listening
2. Because we have semi-duplex bus, on a fixed moment of time only one node (microcontroller) can emite.

Each microcontroller has only one flag which depends by the occupation scale of the bus. This flag is tested in the main program routine and it has the transmission activation/lock task.

The transmission protocol is activated only if the communication needed.

- we will use a buss WDT (Watch Dog Timer) who on every 3*T time periods generates an interrupt. This interrupt indicate that the buss is free: flag – buss occupation=1.
- „everyone listen", for each received octet (RI=1), the serial port routine will be executed.
- If flag – buss ocupation=0, the buss is busy
- Reset of bus WDT(Watch Dog Timer)

Usually, after 3*T ms when the last octed was sent/received, the transmision mechanism of the entire microcontrollers network could be validated.

Random routine number generator using 80C51 microcontroller

We will define a XDATA pointer type. It is intialize with zero, the value on the zero location is read, the first random number in 0x00 – 0xFF domain. The XDATA pointer type is incremented, the value on the one location is read, the second number within domain.

```
.
.

define XBYTE ((unsigned char *) 0x020000h)
x data unsigned char Nr_aleator;
x data unsigned char * Gen_Random;
Gen_Random=0x200000L;
Nr_aleator=*Gen_Random;
Gen_Random++;

void Serial_INT() interrupt 5
{
 if(RI==1){    // tratare receptie in general
    if((Stare_Trans_Rec==1)&&(Contor_Octeti<2)){
        // se fol. cind receptorul este transm. (transmisia se autoasculta)
        // pentru codurile de control
        // Contor_Octeti++;
        // incrementare Contor_Octeti // numai la transmisie
   if(Contor_Octeti==0){
                    // deoarece acesti 2 octeti sunt si transmisi si receptionati
        if(SBUF!=0x5a)
                        // s-a detectat o coliziune
```

```
                        // in acelasi timp
                        Flag_Ocupare_Mag=0;  // pierdere drept de transmisie
                            }
                           if(Contor_Octeti==1){
                        if(SBUF!=0xa5)         // s-a detectat o coliziune
                        Flag_Ocupare_Mag=0;  // pierdere drept de transmisie
                        }
                }
                    }
   if(Stare_Trans_Rec==2){
        // se fol. cind receptorul este receptor ( numai asculta magistrala )
                        Flag_Ocupare_Mag=0;
                                // start delay 3*T pe Magistrala TH0=166;   //
                        reset WDT Magistrala
                        if(Contor_Octeti==2)
                                Buff_Trans[2]=SBUF;
                        if(Contor_Octeti==3)
                                Buff_Trans[3]=SBUF;
                        if(Contor_Octeti==4){
                                if(SBUF==OWN_ADR){
                     Contor_Octeti=0;       // reset contor octeti receptionati
                    Flag_Ocupare_Mag=1;// cistigarea dreptului la magistrala
                                                        }
                                }
        Contor_Octeti++;                        // incrementare Contor_Octeti
                        }
}
if(TI==1){    // tratare transmisie
        if(Stare_Trans_Rec==1){        // se fol. cind transm. este transm.
        Contor_Octeti++;       // index folosit la transmisie si este si nr.
                                // de octeti la receptie doar pentru
                                // primii 2 octeti de test
          }
        if(Stare_Trans_Rec==2){
                        // se fol. cind transm. este receptor ( cind se transmite
                        // "acknoledge" catre transmitter )
          }
       }
 RI=0;
 TI=0;                      // achitare port serial
```

```
}

void initser(void){
        SCON=0x50;  // 01010000b  >> configurare port serial in Modul 1
                    // ( 10 biti ):
                    // 1 bit de start; 8 biti de date; 1 bit stop; Timer1
                    // generator de frecventa
        TMOD|=0x20; // 00100000b  >> configurare Timer1 in Modul 2
        TH1=0xFD;   // 9600 bauds in conditiile in care XTAL =
                    // 11.0592MHz
        PCON&=0x7F // SMOD = PCON.7 = 0 ( implicit la Reset este init.
                    // cu "0" )
        ET1=0;      // ET1 = IE.3 = 0  disable Timer1 overflow interrupt
                    //       ( implicit la Reset este init. cu "0" )
        TR1=1;      // ON Timer1; TR1 = TCON.6
          }

void delay(int x)
{
   int i,j;
   for (i=0;i<x;i++)
         for (j=0;j<1000;j++);
}
void Rutina_TIMER_0() interrupt 2 {

                Flag_Ocupare_Mag=1;     // achitare WDT Magistrala
                TH0=166;                // reinit. divizr HARD cu 166
                                        // => const. de
                                           divizare este 256-166=90
}
```


References

[***1] www.semiconductors.philips.com
[***2] www.siteplayer.com/docs/001212/ 80C51_FAM_HARDWARE_1.pdf
[***3] www.ashling.com/pdf_datasheets/ PhilipsSupportMatrix.PDF
[***4] www.ibr.cs.tu-bs.de/lehre/ws0304/ ist-prak