

LOW LEVEL PROTOCOLE Single DSPIC 33F using RS232

To control the chassis you need to send via RS232 at 19200 bauds some special char that will be received by the DSPIC 33F that controls the motors. To get the data from the chassis it is the same you need to receive from the DSPIC some char to read the IR sensors value, the battery level, the robot consumed current, the speed (in tics) and the accumulated Odometry (in tics).

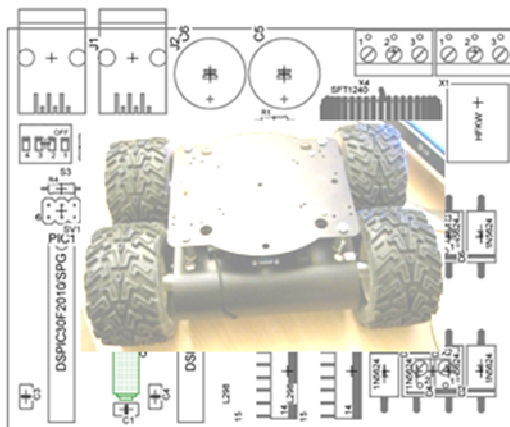
Sending or receiving data is only a matter of sending and receiving chars on a RS232 serial port. This task is achieved on the embedded CPU running Linux or windows or any CPU that have one available com port. And you can use this protocol on any software that can open a serial port (Matlab, RTMAPS, C++, C#, java etc ...).



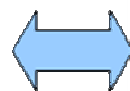
X86 CPU
WinXPe or Linux



Wifibot Lab
(RS232)



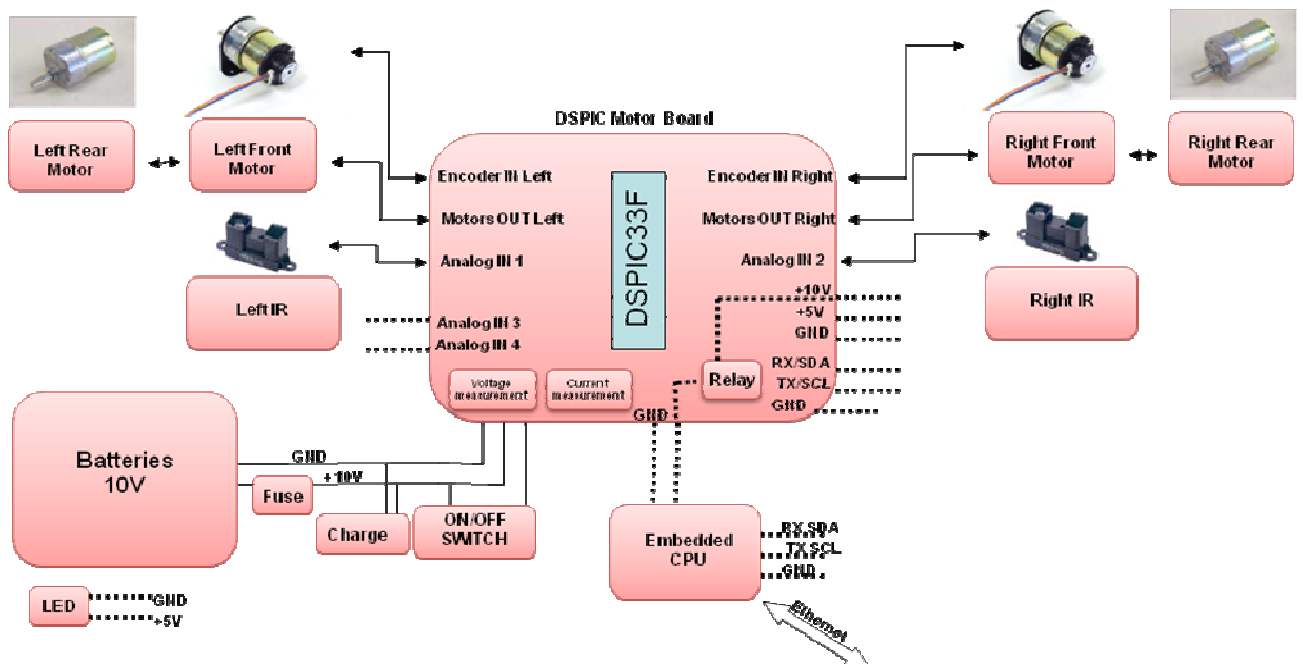
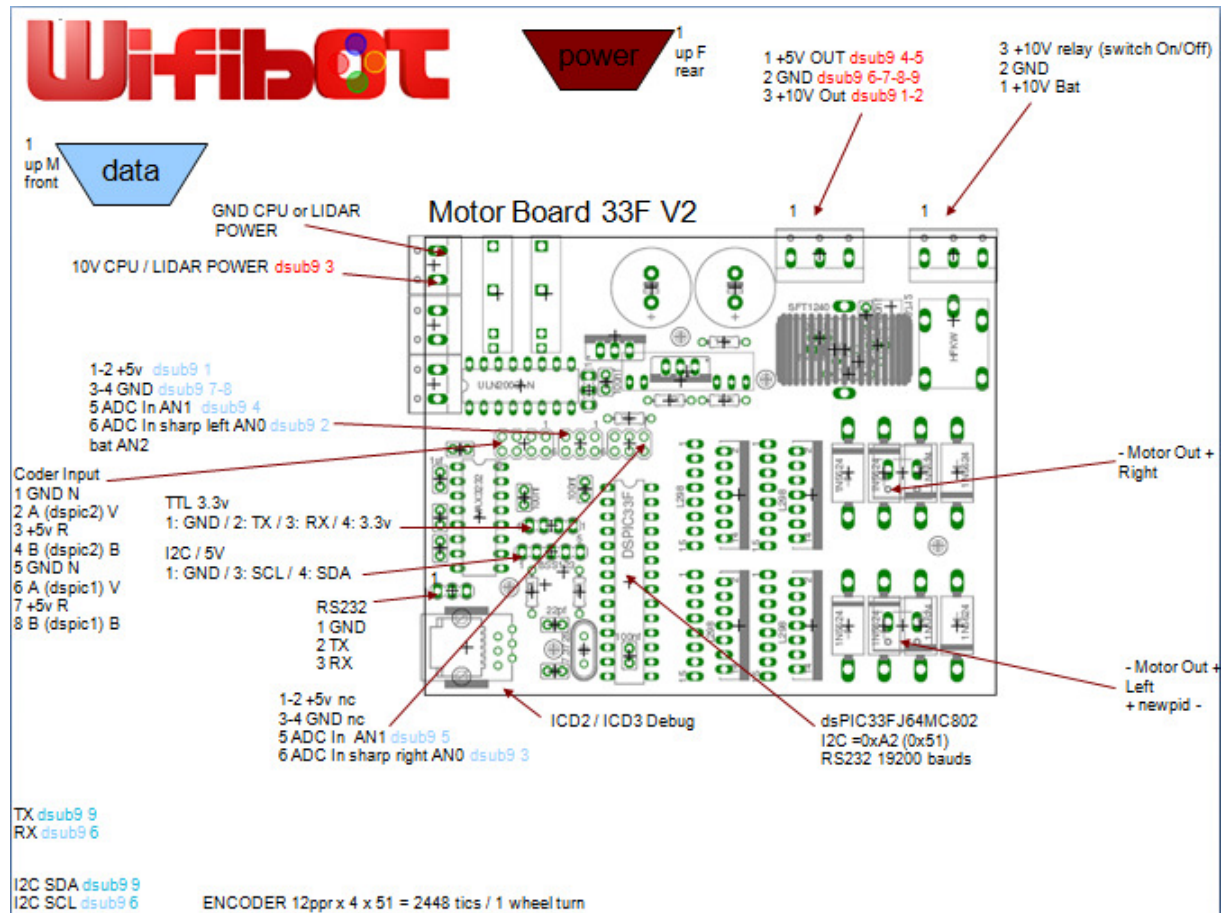
DSPIC Motor Board



Motor + Hall Coders

Controlling the motors:

DSPIC 33F RS232



SET SPEED to move (Send periodically at max 250ms, else Timeout occurs and speed is set to 0):

You need to send 9 char at 19200 bauds.

Char 1 is 255

Char2 is size (here is 0x07)

Char 3-4 is the left speed 0 -> 240 tics max

Char 5-6 is the right speed 0 -> 240 tics max

Char 7 is the Left / Right speed command flag : Forward / Backward and speed control left & right ON / OFF.

Char 7 is decomposed as follow (1 byte char -> 8 bites):

(128) Bite 7 Left Side Closed Loop Speed control :: 1 -> ON / 0 -> OFF

(64) Bite 6 Left Side Forward / Backward speed flag :: 1 -> Forward / 0 -> Reverse

(32) Bite 5 Right Side Closed Loop Speed control :: 1 -> ON / 0 -> OFF

(16) Bite 4 Right Side Forward / Backward speed flag :: 1 -> Forward / 0 -> Reverse

(8) Bite 3 PID speed 0 is 50 ms 1 is 10 ms (50 ms is recommended)

(4) Bite 2 Not Used (Relay 3 On/Off (future option))

(2) Bite 1 Not Used (Relay 2 On/Off (future option))

(1) Bite 0 Not Used Relay 1 for CPU or sensors. On/Off: 0 is OFF 1 is ON (DSUB9 POWER Pin 3)

Char 8-9 is the CRC 16 bits (char 7 low char 8 high, see end of document for details)

So to control for example the left side and the right side (Left & Right 2 motors, 2 encoders):

The speed is between 0-240

If we want the left side & right side to move at speed 120 forward without motor control we send:

Char 1 is 255

Char2 is 0x07

Char 3 is 0

Char 4 is 120

Char 5 is 0

Char 6 is 120

Char 7 is 80 (0 + 64 + 0 + 16)

Char 8 – Char 9 = CRC16(data)

```
SetMotorRS23233f(hUSB,120,120,80);
```

```
//hUSB is the serial port handle opened at 19200 baud
```

```
int SetMotorRS23233f(HANDLE hUSB, short speed1,short speed2,BYTE SpeedFlag)
{
```

```
    DWORD n;
    BYTE sbuf[10];
    sbuf[0] = 255;
    sbuf[1] = 0x07;
    sbuf[2] = (BYTE) speed1;
    sbuf[3] = (BYTE) (speed1 >> 8);
```

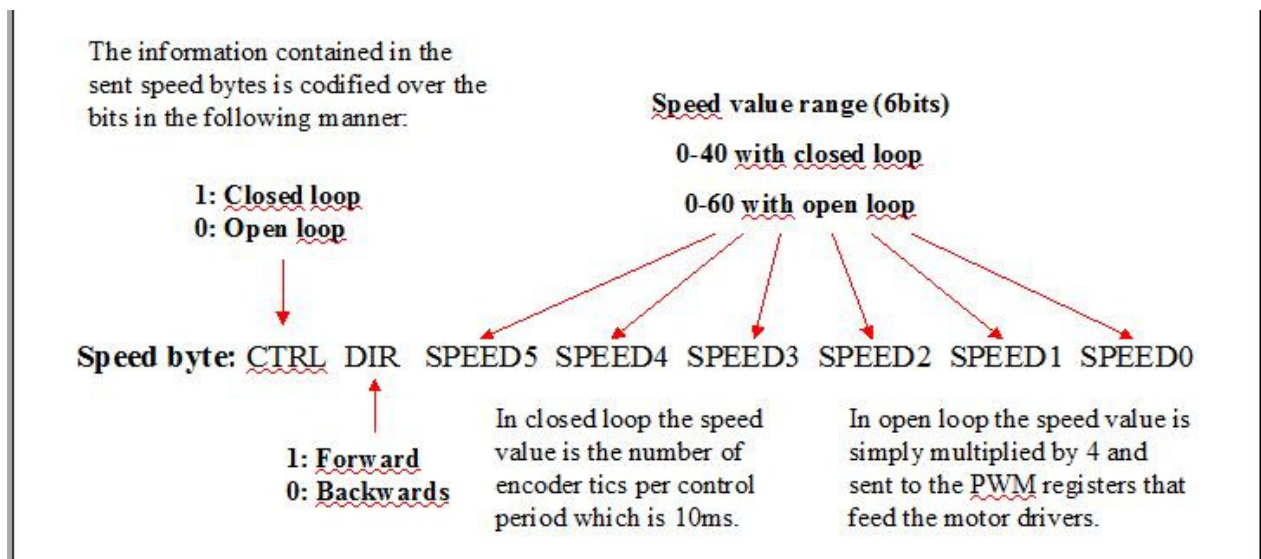
```

sbuf[4] = (BYTE)speed2;
sbuf[5] = (BYTE)(speed2 >> 8);
sbuf[6] = SpeedFlag+1;
short mycrcsend = Crc16(sbuf+1,6);
sbuf[7] = (BYTE)mycrcsend;
sbuf[8] = (BYTE)(mycrcsend >> 8);

bool res = WriteFile(hUSB, &sbuf, 9,&n, NULL);
return res;
}

```

To use the old WIFIBOT protocol with only 2 char and less resolution (0 to 60):
(The wifibot socket protocol use this for the moment)



If we want the left side & right side to move at speed 20 forward without motor control we send:

$$128*0 + 64*1 + 20*1 = 84$$

```
SetMotorRS23233f_low_res(hUSB,84,84);
```

If we want the left side to move at speed 20 backward with motor control we send:

$$128*1 + 64*0 + 20*1 = 148$$

```
SetMotorRS23233f_low_res(hUSB,148,148);
```

```

int SetMotorRS23233f_low_res(HANDLE hUSB, BYTE speed1,BYTE speed2)
{
    DWORD n;
    BYTE sbuf[30];
    BYTE tt=0;
    sbuf[0] = 255;
    sbuf[1] = 0x07;

    int tmp1 = 8*(speed1&0x3F);
    int tmp2 = 8*(speed2&0x3F);
    if (speed2&0x80) tt=tt+32;
    if (speed2&0x40) tt=tt+16;
    sbuf[2] = (BYTE)tmp1;
    sbuf[3] = (BYTE)(tmp1 >> 8);
}

```

```

    sbuf[4] = (BYTE)tmp2;
    sbuf[5] = (BYTE)(tmp2 >> 8);
    sbuf[6] = (speed1&0x80) + (speed1&0x40) + tt +1; //+1 Relay ON +8 10ms
    pid mode ;
    short mycrcsend = Crc16(sbuf+1,6);
    sbuf[7] = (BYTE)mycrcsend;
    sbuf[8] = (BYTE)(mycrcsend >> 8);
    bool res = WriteFile(hUSB, &sbuf, 9,&n, NULL);
    return res;
}

```

SET PID:

If we want to set a new PID on the left & right side at P=0.77 I=0.01 D=0.30

```
SetMotorPIDRS23233f(hUSB,0x00,0x00,77,1,30,360);
```

```

int SetMotorPIDRS23233f(HANDLE hUSB,BYTE speed1,BYTE speed2,BYTE pp,BYTE
ii,BYTE dd,short maxspeed)
{
    DWORD n;
    BYTE sbuf[30];

    sbuf[0] = 255;
    sbuf[1] = 0x09;
    sbuf[2] = speed1; //Always 0
    sbuf[3] = speed2; //Always 0
    sbuf[4] = pp;
    sbuf[5] = ii;
    sbuf[6] = dd;
    sbuf[7] = (BYTE)maxspeed;
    sbuf[8] = (BYTE)(maxspeed >> 8);
    short mycrcsend = Crc16(sbuf+1,8);
    sbuf[9] = (BYTE)mycrcsend;
    sbuf[10] = (BYTE)(mycrcsend >> 8);
    WriteFile(hUSB, &sbuf, 11, &n, NULL);
    return sbuf[0];
}

```

Maxspeed is the max command possible – 20 : **set it to 360.**

To GET DATA from the chassis:

The DSPIC sends you 21 chars in continuous at 10ms:

```

//DSPIC C code to show you the sending part to the PC:
bufsend[0]=(-speedlab);
bufsend[1]=(-speedlab >> 8); //speed is a short and it is tics / 50 ms
bufsend[2]=(unsigned char)(tmpadc2 >> 2); //Bat Volt:10.1V 1.28V 404/4->101
bufsend[3]=(unsigned char)(tmpadc4 >> 2); //3.3v->255 2v-> 624/4 -> 156
bufsend[4]=(unsigned char)(tmpadc3 >> 2); //3.3v->255 2v-> 624/4 -> 156
bufsend[5]=bufposition[0]; //Acumulated odometrie is a float
bufsend[6]=bufposition[1]; //12ppr x 4 x 51 gear box = 2448 tics/wheel turn
bufsend[7]=bufposition[2];
bufsend[8]=bufposition[3];
bufsend[9]=(speedlab2);

```

```

bufsend[10]=(speedlab2 >> 8);
bufsend[11]=(unsigned char)(tmpadc0 >> 2);
bufsend[12]=(unsigned char)(tmpadc1 >> 2);
bufsend[13]=bufposition2[0];
bufsend[14]=bufposition2[1];
bufsend[15]=bufposition2[2];
bufsend[16]=bufposition2[3];
bufsend[17]=0;//robot current // *0.194-37.5 = I in Amp / * 10 for the GUI
: 10 -> 1 A (ACS712 30Amp chip) // *0.129-25 =I (for ACS712 20A chip)
bufsend[18]=14; //firmware version
bufsend[19]=crc16 low;
bufsend[20]=crc16 high;

```

You receive by using:

```

int GetMotorRS23233f(HANDLE hUSB, SensorData *dataL, SensorData *dataR)
{
    DWORD n;
    BYTE sbuf[30];
    bool res=false;

    do {
        ReadFile(hUSB, &sbuf, 1, &n, NULL);
    }while(sbuf[0]!=255);

    res = ReadFile(hUSB, &sbuf, 21, &n, NULL);

    short mycrcrcv = (short)((sbuf[20] << 8) + sbuf[19]);
    short mycrcsend = Crc16(sbuf,19);

    if (mycrcrcv!=mycrcsend)
    {
        do {
            ReadFile(hUSB, &sbuf, 1, &n, NULL);
        }while(sbuf[0]!=255);
    }
    else {
        dataL->SpeedFront=(int)((sbuf[1] << 8) + sbuf[0]);
        if (dataL->SpeedFront > 32767) dataL->SpeedFront=dataL->SpeedFront-65536;
        dataL->BatLevel=sbuf[2];
        dataL->IR=sbuf[3];
        dataL->IR2=sbuf[4];
        dataL->odometry=((((long)sbuf[8] << 24))+(((long)sbuf[7] << 16))+(((long)sbuf[6] << 8))+((long)sbuf[5]));

        dataR->SpeedFront=(int)(sbuf[10] << 8) + sbuf[9];
        if (dataR->SpeedFront > 32767) dataR->SpeedFront=dataR->SpeedFront-65536;
        dataR->BatLevel=0;
        dataR->IR=sbuf[11];
        dataR->IR2=sbuf[12];
        dataR->odometry=((((long)sbuf[16] << 24))+(((long)sbuf[15] << 16))+(((long)sbuf[14] << 8))+((long)sbuf[13]));
        dataL->Current=sbuf[17];
        dataR->Current=sbuf[17];
        dataL->Version=sbuf[18];
        dataR->Version=sbuf[18];
    }
    return res;
}

```

Serial Port Open (Windows):

```
HANDLE SetupRS232CommPort( LPCSTR comport)
{
    HANDLE hCom;
    DCB dcb;
    COMMTIMEOUTS ct;

    hCom = CreateFileA( comport, GENERIC_READ | GENERIC_WRITE, 0, 0,
        OPEN_EXISTING, 0, 0);
    GetCommState(hCom, &dcb);
    dcb.BaudRate = RS232_SPEED;//19200 bauds
    dcb.fParity = FALSE;
    dcb.fOutxCtsFlow = FALSE;
    dcb.fOutxDsrFlow = FALSE;
    dcb.fDtrControl = DTR_CONTROL_DISABLE;
    dcb.fDsrSensitivity = FALSE;
    dcb.fOutX = FALSE;
    dcb.fInX = FALSE;
    dcb.fRtsControl = RTS_CONTROL_DISABLE;
    dcb.fAbortOnError = FALSE;
    dcb.ByteSize = 8;
    dcb.Parity = NOPARITY;
    dcb.StopBits = 0;
    SetCommState(hCom, &dcb);

    GetCommTimeouts(hCom, &ct);
    ct.ReadIntervalTimeout = 500;
    ct.ReadTotalTimeoutMultiplier = 500;
    ct.ReadTotalTimeoutConstant = 500;
    SetCommTimeouts(hCom, &ct);
    SetCommMask(hCom, EV_RXCHAR);
    return hCom;
}
```

CRC 16 bits:

```
short Crc16(unsigned char *Adresse_tab , unsigned char Taille_max)
{
    unsigned int Crc = 0xFFFF;
    unsigned int Polynome = 0xA001;
    unsigned int CptOctet = 0;
    unsigned int CptBit = 0;
    unsigned int Parity= 0;

    Crc = 0xFFFF;
    Polynome = 0xA001;
    for ( CptOctet= 0 ; CptOctet < Taille_max ; CptOctet++)
    {
        Crc ^= *( Adresse_tab + CptOctet);

        for ( CptBit = 0; CptBit <= 7 ; CptBit++)
        {
            Parity= Crc;
            Crc >>= 1;
            if (Parity%2 == true) Crc ^= Polynome;
        }
    }
    return(Crc);
}
```