The Visual Multi-DC-Motor Control System of Digital Signal Processor Based on Visual Studio 2010

Wenli Huang, Shunxin Liu

school of mechatronics engineering, zhengzhou institute of aeronautical industry management,

Zhengzhou, China

hwl@zzia.edu.cn

Keywords: DSP; VS2010; serial communication; dc motor

Abstract. This paper introduced a kind of digital signal processor (DSP) control system of several (≥2) dc motors, and described the development of dc motor control interface using C# language based on Visual Studio 2010 (VS2010) software platform. The mechanism of serial communication as well as its control on motor behaviors was analyzed in depth between computer (PC) and DSP (client), which was accompanied by program codes corresponding to motor control functions.

Introduction

Dc motor has good starting and speed control performances due to large starting torque, can be speeded smoothly in a wide range, and can withstand impact loads frequently started, braked and reversed, consequently it is widely applied in the field of motor drag [1]. Digital motor control has changed from the traditional "status drive" into "motion control" consisting of the supervisory control of motion parameters such as position, velocity, acceleration, torque, force, et al.. Digital signal processor (DSP) with high performance can satisfy the requirements of high precision, complex strategies and rapid response for motor controls; especially in recent years, DSP, for example TMS320C24x embedded by special motor control chip, is able to bring PWM waveform with dead band time, no spurious signal, low cost and high reliability, greatly simplifying the design for motor motion controller, hence promoting the application of DSP in the field of motor control [2].Visual Studio 2010 (VS2010) supporting mobile application platform of .NET Framework 4.0 edition is the most popular platform environment to develop Windows' application program, and facilitates the development of application programs with very universal and powerful function, using C# language.

According to the working principle, the speed regulation of dc motor can adopt three modes: change the voltage of armature, regulate the circuit resistor of armature or adjust the excitation flux. Because of obvious defects in other two modes, at present people mainly employ the method of changing armature voltage to regulate dc motor's speed smoothly in a wide range [3].

The paper introduced a kind of dc motor's control system using digital signal processor, TMS320F2812, mostly consisting of the development of visual motor control interface, serial communication mechanism and program codes of some main control functions.

The control system hardware platform

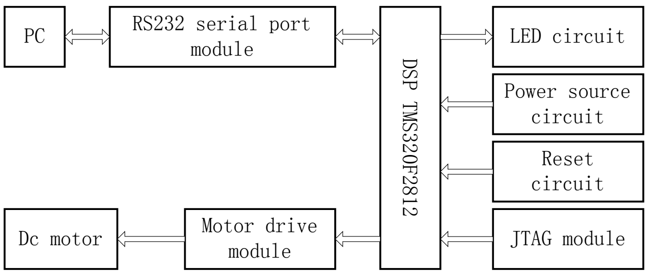

Figure 1. Structure diagram of dc motor control system.

Illustrated in Fig. 1 is the structure diagram of DSP multi-dc-motor control system based on VS2010, which is the three-in-one motor control host system mainly composed of PC, DSP and dc motor, and supplemented by several peripheral circuits: serial port communication module, LED signal circuit, power source circuit, reset circuit, JTAG module and motor drive circuit. They jointly implement behavior control of multi-dc motors. The control system is attributed to host-to-client computer structure among which PC reflects as host computer to execute visual real-time monitoring of motor’s behaviors and transfers specific control commands to the client computer of DSP; then DSP regulates motor’s behaviors according to corresponding algorithms and commands transferred, and feeds back motor running status to the host computer.

Visual motor control interface

A visual motor control interface was developed using C# language based on VS2010 program development tools integrated with .NET Framework 4.0. Created a new project named “SerialPortConnect” within VS2010, and added a series of controls of “ComboBox”, “TextBox”, “Button”, “CheckBox”, et al. on window form of “Winform” built newly, further named and arranged them according to the project’s requirements. The relevant program codes can refer to [4].

Serial communication between PC and DSP would be established, after basic attributes of serial port such as name, parameters (data bits, start-stop bits, polarity bits, etc.), data format (hexadecimal or string) of the serial port are defined. Hence on the premise of correct serial port communication, it is only needed to call data transmition function of serial port for the motor control interface to regulate the motor’s behaviors such as speed measurement, acceleration and deceleration, positive and negative rotation, etc., in accordance with the predetermined information code format (similar to telegraph codes that different codes are indicative of different meanings).

TABLE I. COMMAND CODES OF MOTOR CONTROL

	Control commands	*Command codes*	*Function paraphrases*
Status of motor	Positive /negative rotation	01 FE(FD) 55(AA)	Motor M1 starts forward at set speed. FD for motor M2, AA for negative rotation.
	Stop	02 FE(FD) (FF)	Motor M1 is stopped. FD for motor M2, FF for stopping motor M1 and M2 simultaneously.

Each button has sole command code to execute special regulation for motor, for example showed in Table 1 is motor control command code of positive/negative rotation and stop, without others because of their completely similar format. DSP will adjust motor’s actions - altering duty cycle to increase or decrease motor’s speed, stopping the motor, etc. - as given procedure of the program, when receiving different command codes. Herein it is clear that the command codes of both “FE” on behalf of motor M1 and “55” on behalf of motor’s positive rotation are not unique, and can be completely substituted for other characters, as long as the characters in serial communication program of PC are consistent with those in motor drive program of DSP.

A. Serial Communication Program

The basic settings of serial port such as serial port number, baud rate, data bits, stop bits, parity bits and so on should be completed for serial communication to connect the serial port of PC with that of DSP, then to realize data transfer between PC and DSP. The main several functional codes are listed as below.

```
private void Scanf()
{
       //Acquire all serial ports on PC.
       string[] comScanf = SerialPort.GetPortNames();
       if (comScanf == null)
       {
              MessageBox.Show("No serial ports on the PC");
              return;
       }
       //Erase the context in serial port drop-down box
       cbSerialPort.Items.Clear();
       foreach (string str in comScanf)
       {
           /*Add the serial port number gotten to serial port drop-down box.*/
              cbSerialPort.Items.Add(str);
       }
}
```

Figure 2. Program codes of acquiring serial port.

1) The acquirement of serial port

In C# language, added the quote of “using System.IO.Ports” to the program, and then established the object of class “SerialPort()” which induced the serial port to be operated conveniently in the serial port program. There are two methods to achieve serial list, namely checking registry method and direct scanning method. The latter - the function of “GetPortNames()” in the quote of “System.IO.Ports” - is called to achieve the serial list herein. The relevant program codes are displayed in Fig.2.

2) The settings of serial port parameters

These parameters such as baud rate, data bits, stop bits and polarity bits, should be set firstly for serial communication of UART standard bus RS-232C before the binary data are transmitted between two serial ports. Baud rate and data bits can be simply set as either of them is fixed a certain value; stop bits and polarity bits have a few different options and very similar codes in structure. Consequently, Fig.3 only provided the program codes for the settings of stop bits as an example.

```
switch (cbStopBits.Text)
{
        //SP is the serial port object defined.
       case "1":
              SP.StopBits = StopBits.One;
              break;
         /*The omitted options of “1.5” and “2” below are wholly similar to the option
“1”..*/
              ……;
       default:
       MessageBox.Show("false parameters");
       break;
}
```

Figure 3. Program codes of the settings of stop bits

3) *The opening and closing of serial port*

The opening and closing of serial port, respectively, are the program statements of “SP.Open()” and “SP.Close()” in VS2010 after the serial port object of “SP” is stated.

4) *The data transmission of serial port*

The serial port data can be automatically sent at regular time to satisfy the requirements for some special occasions, which is not discussed in depth here. While the send button on serial port communication interface is clicked, the function of “SendMsgToPort()” will be called to send the unique command codes to PC serial port output buffer, then to complete data transfer forward into DSP serial port. For the hexadecimal data, the format conversion of data is completed first, then to be transmitted into serial port output buffer making use of function of “Write()”; for the string data, the data transmition will be implemented by function of “WriteLine()” immediately. The corresponding program codes are displayed in Fig.4.

5) *The data reception of serial port*

Similar to sending data, the data reception of serial port can also be designed to be implemented automatically at regular time, that is, a new thread will be added by PC to read the data within the PC serial port input buffer after the fixed time is reached. This method reflects large limitation, thus is only suitable for the data source varying slowly. As a

```
private void SendMsgToPort(string msg)
{
    string strSend = msg;
    //For the hexadecimal data.
    if (rdSend16.Checked == true)
    {
        string strBuf, strNoNull, strNoCom, strNoCom1,
               strNoCom2;
        strBuf = strSend;
        //Delete the head and tail spaces.
        strNoNull = strBuf.Trim();
        //Delete the characters of ",", "，", "0x", "0X"..
        strNoCom = strNoNull.Replace(',', ' ');
        strNOCom1 = strNoCom.Replace('，', ' ');
        strNoCom2 = strNoCom1.Replace("0x", "");
        strNoCom2.Replace("0X", "");
        //Insert a space between each two characters
        string[] strArray = strNoCom2.Split(' ');
        //Count the number of characters.
        int byteBufferLength = strArray.Length;
        for (int i = 0; i<strArray.Length; i++)
        {
            if (strArray[i] == "") { byteBufferLength--;}
        }
        /*The string data were transformed into binary date that could be transmitted.*/
        byte[] byteBuffer = new byte[byteBufferLength];
        int j = 0;
        for (int i = 0; i<strArray.Length; i++)
        {
            Byte[] bytesOfStr = Encoding.ASCII
            GetBytes(strArray[i]);
            int decNum = 0;
            if (strArray[i] == "") { continue; }
            else
            {
                decNum = Convert.ToInt32(strArray[i], 16);
            }
            try
             {
                byteBuffer[j] = Convert.ToByte(decNum);
             }
            catch (System.Exception ex)
            {
                MessageBox.Show(ex.Message);
                return;
            }
            j++;
        }
         SP.Write(byteBuffer, 0, byteBuffer.Length);
     }
    else  //For the string data
```

Figure 4. Program codes of sending data to serial port of DSP.

result, the plan of incident response drive data reception is adopted – namely as the value of bytes within serial port input buffer of DSP is larger than the preset value, a serial port interrupt function would be launched to finish the reception of data within DSP serial port input buffer. The string data is read by the function of “ReadLine()”; whereas the hexadecimal data is connected with the function of “Read()” after data format conversion was done. The corresponding program codes are listed in Fig.5.

```
void SP_DataReceived(object sender,
                              SerialDataReceivedEventArgs e)
{
      if (SP.IsOpen) //The serial port has been opened.
      {
            if (rbRcvStr.Checked) //For string data.
            {
                 byte[] byteRead = new byte[SP.BytesToRead];
                 txtReceive.Text += SP.ReadLine() + "\r\n";
                 SP.DiscardInBuffer();
            }
            else if(rbRcv16.Checked) //For hexadecimal data.,
            {
                 try
                  {
                      byte[] receivedData = new byte[SP.BytesToRead];
                      SP.Read(receivedData, 0, receivedData.Length);
                      SP.DiscardInBuffer();
                      stringstrRcv = null;
                      for (inti = 0; i<receivedData.Length; i++)
                      {
                           strRcv += receivedData[i].ToString("X2")+" ";
                        }
                        txtReceive.Text += strRcv +" "+ "\r\n";
                   }
                  catch (System.Exception ex)
                  { MessageBox.Show(ex.Message); }
            }
      }
      else //The serial is closed.
      {
            MessageBox.Show("please open some serial port");
            return;
      }
}
```

Figure 5.   Program codes of serial port receiving data for PC.

B. *Motor Control Program*

```
private void btnInc_Click(object sender, EventArgs e)
{
     if (rdM1.Checked == true) //For motor M1 to be speeded up.
     {
          SendMsgToPort("03" + " " + "FE" + " " + "DE" + " " + "00");
     }
     else if (rdM2.Checked == true) //For motor M2 to be slowed up.
     {
          SendMsgToPort("03" + " " + "FD" + " " + "DE" + " " + "00");
     }
 }
```

Figure 6.   Program codes of motor control.

Based on the serial communication program established, the realization of different motor control commands for motor control window is merely to click the corresponding different buttons to incite the function of “SendMsgToPort()” so as to transmit different command codes to the PC serial port making use of serial communication program, and hence to send the relevant control commands to DSP through serial-connected circuit between PC and DSP, finally to realize different controls of motor. The program codes, as an example for the implementation of increasing speed which called the function of “btnInc_Click()”, is described in Fig.6. Except for concrete command codes for different control commands as indicated in Table 1, the structures of other functions are entirely similar to that of the previous increasing speed function, which is the reason for .the spared discussion herein.

DSP motor drive system

```
/*CPU would launch this interrupt function, as long as the data within serial port input buffer need to be received. */
interrupt void SCIRXINTA_ISR(void)
{
     //Save control commands using an array.
    Sci_VarRx[i] = SciaRegs.SCIRXBUF.all;
    i++;//stated in main function.
    /*According to the    command    code format,    01 is motor    positive/negative rotation identification code.*/
    if(Sci_VarRx[0]==0x0001)
    {
        // the function of motor positive/negative rotation
        Rec_Manage(Sci_VarRx[0]);
    }
    //02 is motor stop identification code.
    else if(Sci_VarRx[0]==0x0002)
    { ……; }
    /*03, 04, 05 are the identification codes of the acceleration/deceleration of motor, the settings of duty cycle and
    the inquiry of motor running state, respectively.*/
    ……;
    if(i==4){ i=0; }
    //Enable other interrupts in one group to be responded.
    PieCtrl.PIEACK.bit.ACK9=1;
    EINT; //Open the global interrupts.
}
```

Figure 7. Program codes of motor control commands to be conducted

DSP is an information hub which guides the motor control strategies to be received, exchanged and implemented; accordingly, its drive program forms a basic platform for the implementation of control strategies. Comprised mostly in DSP drive program are: the initialization function of peripheral devices, main function and input interrupt function. Among these functions, the initialization function fulfills initializing definition of the related devices such as DSP, input/output pins, serial communication ports, event manage and so on; the main function will declare global variables, macros and some applicable functions, and define the initiating state of motor drive system, and enable the interrupt function, finally conduct a boundless cycle to wait for CPU in response to the interrupt requests in order to finish the control commands of motor behaviors in the interrupt function. The motor drive program made up of modular structure executes different motor control commands by calling each functional sub-function, which is clear in logic and easy to expand system function. The DSP in the standby state, would accomplish motor control in accordance with the corresponding command codes, as long as the motor control commands are transmitted to serial port from PC to DSP.

The interrupt mode is employed to send and receive data – namely using interrupt mode to fulfill motor control commands and feed back motor running state to PC. Those in Fig.7 are concrete program codes:

```
void M_Inc(unsigned int num) //The speed-up function
{
       if((num==1)&(M1_Flag!=0)) //For motor M1
       {
              //Duty cycle increment of 2%.
            EvaRegs.CMPR1=EvaRegs.CMPR1+375;
            if(EvaRegs.CMPR1>=0x41EB)
            {
                //The maximum duty cycle of 90%.
               EvaRegs.CMPR1=0x41EB;
            }
       }
       else if((num==2)&(M2_Flag!=0))
       {
            ……;//For other motors.
       }
}
```

Figure 8. Program codes of speed-up regulation of motor.

The other functions such as initialization function, main function and so on can be infered in [2], thus no more discussions here. For sub-function of motor different controls based on the program adopting modular structure, each of them is tied to a particular motor control and has similar structure –namely, firstly judges command object, then conducts a specific command according to different command codes, so Fig.8 only lists program codes of the acceleration function as a sample of other functions of motor controls.

Conclusions

DSP as a motor motion controller to realize motor controls, has become the most popular motor drive scheme at present. The paper utilized VS2010 to develop visual motor control interface, and adopted C# program language to compile serial communication program, then accomplished the behavior regulation and supervision of motor on the basis of data exchange between PC and DSP. In addition, the design has powerful maneuverability and very strong adaptability, so that it can be easily expanded to control more motors and execute more complicated control strategy.

References

[1] F. H. Li and Y. Wang, The motor and drag foundation. Beijing: Tsinghua University press, 2012.

[2] W. G. Gu, Taught you how to learn DSP — Based on TMS320X281x. Beijing: Beihang University press, 2011.

[3] W. Y. Li and X. Li, “The design of DC motor speed control system based on DSP,” Digital Technology & Application, Dec. 2012, pp. 128-130.

[4] X. D. Jiang, The definitive guidlines to C# 4.0. Beijing: China machine press, 2011.