C# and .NET Framework for uC communication protocol implementation

C.D. CĂLEANU, V. TIPONUȚ, I. BOGDANOV, S. IONEL, I. LIE
Applied Electronics Department, Faculty of Electronics and Telecommunications
University "POLITEHNICA" Timişoara
V. Pârvan Av, no. 2, 300223 Timişoara
ROMÂNIA
http://www.etc.upt.ro

Abstract: - This paper describes an integrating hardware and software ensemble for the development of microcontroller-based computer numerical control system. The discuss is focused on design and implementation of the communication between the PC and a numerical control machine. Among topics covered are: the C#/.NET serial port programming, and further, how to create a Windows application for sending, trough a COM port, the Computer Aided Design/Manufacturing data needed by a numerical control machine. A PIC microcontroller-based communication hardware module was developed and tested.

Key-Words: C# serial port programming, PIC microcontroller, CNC machine.

1 Introduction

The parts that make up a Computer Numerical Control (CNC) system could be classified as [1]:

a) ***The software components*** represented by the Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) programs that allow the users to design their products and further convert the design into code instructions, e.g. G code, programs that interpret the instructions and operate the machine.

b) ***The hardware components*** usually represented by a Personal Computer (PC), the afferent communication ports (parallel, serial, USB) that send the signals to the machine's controller. The controller that reads the signals generated by the PC via the program and the communication port operates stepper motors in a controlled manner. As a result, the machine moves each axis (usually two or three-axis machines) individually or simultaneously.

As a practical example, let's suppose that a CNC machine is required to create the geometric entity depicted by fig. 1.

The two axis controller must be fed with at least two kind of information: two words, one for X axis and the other for the Y axis, which quantify the displacement measured in motor's encoder steps. In our project these words offers a resolution of 12 bits each. Another 8 bits are used for coding additional information. For example, on path A-B the cutting tool must be off then, in B point, it must be activated. Therefore, the axis controller receives, one time, a 32 bits word.

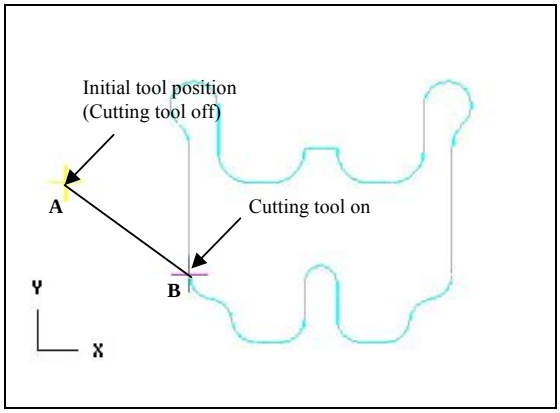

Fig. 1. The CNC machine tool must produce a formed surface on a workpiece.

Integrating software and hardware designed to manipulate streams of bytes, between a PC and a power block commanding axis motors is the concern of this paper.

2 Programming the Serial Port using C# and .NET framework

2.1 The selection of the communication interface

In order to exchange data between the application software running on the PC and the microcontroller-based hardware which command the CNC machine

the parallel and the serial port represents good candidates. Although the newly USB interface offers higher data transfer rates, it is not present on all PC motherboards, especially older models lack this functionality.

We decide to make use of the serial port for our communication purpose although is harder to interface than the parallel port.

In most cases, any device you connect to the serial port will need the serial transmission converted back to parallel so that it can be used. On the software side of things, there are many more registers that you have to attend to than on a Standard Parallel Port (SPP).

But there are also other advantages of using serial data transfer rather than parallel [2]:

a) The serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables as they are for parallel. As a consequence, serial cables can be longer than parallel cables and safer in a noisily environment.

b) If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper that running 19 or 25 core cable.

c) Many microcontrollers have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial Communication reduces the pin count of these MPU's. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use an 8 bit Parallel method.

2.2 Serial Communication using C#

In June 2000, Microsoft announced both the .NET (pronounced DOT NET) platform and a new programming language called C# (pronounced C Sharp) [3] – [5]. The programming language of choice for .NET platform is C#. The language is powerful, productive, type safe, has a rich and clear syntax and, most importantly, provides a conceptually appealing implementation of the object-oriented paradigm.

Serial communication could be done in a simple and straightforward manner using the specifications of C# and .NET framework [6], [7].

The System.IO.Ports namespace contains classes for controlling serial ports. The most important class is the *SerialPort* class. It provides a framework for synchronous and event-driven I/O, access to pin and break states, and access to serial driver properties. It can be used to wrap stream objects, allowing the serial port to be accessed by classes that use streams.

Programmatically it could be created by one of the seven public constructors. Initializes a new instance of the *SerialPort* class using the specified port name, baud rate, parity bit, data bits and stop bit is the most complex form of these constructors.

In order to memorize the program's session COM port settings we will create a serialized object which will be easily load/saved form/to disk. Serialization is the process of converting the state of an object into a form that can be persisted or transported. During this process, the public and private fields of the object and the name of the class, including the assembly containing the class, are converted to a stream of bytes, which is then written to a data stream. The easiest way to make a class serializable is to mark it with the *Serializable* attribute as follows.

```
[Serializable]
public class MyPortSerialized
{
public int PortName;
        public int BaudRate;
        public int DataBits;
        public int StopBits;
        public int Parity;
        public int Handshake;
}
```

The following example uses a binary formatter to do the serialization. It is create an instance of the stream and the formatte and then call the *Serialize* method on the formatter. The stream and the object to serialize are provided as parameters to this call.

```
private void defaultButton_Click(object sender, EventArgs e)
{
        MyPortSerialized PortSerialized = new
MyPortSerialized();

        PortSerialized.PortName                 =
this.portComboBox.SelectedIndex;
        PortSerialized.BaudRate                 =
this.transferSpeedComboBox.SelectedIndex;
        PortSerialized.DataBits                 =
this.dataBitsComboBox.SelectedIndex;
        PortSerialized.StopBits                 =
this.stopBitscomboBox.SelectedIndex;
        PortSerialized.Parity                   =
this.parityControlComboBox.SelectedIndex;
        PortSerialized.Handshake                =
this.flowControlComboBox.SelectedIndex;

        FileStream fs = new FileStream("default.cfg",
FileMode.OpenOrCreate);
        BinaryFormatter bf = new BinaryFormatter();
        bf.Serialize(fs, PortSerialized);
        fs.Close();
}
```

The BinaryFormatter used above is very efficient and produces a compact byte stream. All objects serialized with this formatter can also be deserialized with it, which makes it an ideal tool for serializing objects that will be deserialized on the .NET Framework. The complement of serialization is deserialization, which converts a stream into an object.

A TextBox control is used for displaying the codes to be sent/received via RS232. When data received event is trigger, *ReadExisting() SerialPort*'s public method is invoked. It reads all immediately available bytes, based on the encoding, in both the stream and the input buffer of the *SerialPort* object and put them in the Receive Textbox.

Through software XON/XOFF control protocol the hardware device request more bytes to read. XOFF is a software control sent to stop the transmission of data and the XON control is sent to resume the transmission. These controls are used instead of Request to Send (RTS) and Clear to Send (CTS) hardware controls.

Then *Write() SerialPort*'s public method is used for send a specified count of bytes (for our application only 4 bytes at once) to an output buffer at the specified offset.

All the above mentioned operations were implemented by the form presented in fig. 2.

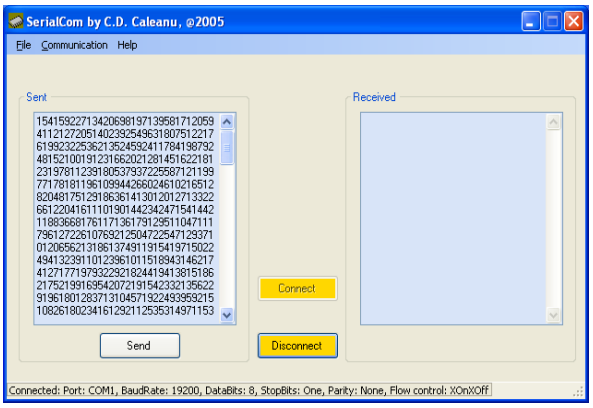

Fig. 2. The main application form.

3 The Hardware

In this chapter, a PIC microcontroller-based system allowing serial data transferred is presented.

Basically, this circuit intermediate between the PC serial port and the power block providing servo control of CNC machine's DC motors and incremental encoders.

3.1 Schematics

At the design core stands Microchip's PIC16F877A [8]-[10], a 40-pin enhanced flash microcontroller.
Its main role is to communicate with PC COM port via the Universal Synchronous Asynchronous Receiver Transmitter (USART) module, also known as a Serial Communications Interface or (SCI).

The USART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers, or it can be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

Also, the hardware system implements an 80 bytes software FIFO buffer. At the beginning of the transmission, the buffer is filled. Then, it receives the next 4 bytes from the PC only when an interrupt is raised (RA0/INT pin) by motors power block.

The SN74AHCT573 devices are octal transparent D-type latches. When the latch-enable (LE) input is high, the Q outputs follow the data (D) inputs. When LE is low, the Q outputs are latched at the logic levels of the D inputs. A buffered output-enable (OE) input can be used to place the eight outputs in either a normal logic state (high or low) or the high-impedance state. In the high-impedance state, the outputs neither load nor drive the bus lines significantly. The high-impedance state and increased drive provide the capability to drive bus lines without interface or pull-up components.

Using RA0-RA3 uC's lines programmed as outputs, the latches are selected for receiving a byte from uC's FIFO. When all four latches are loaded, using RE0 line connected to all buffers output-enable pin, the four bytes word is available for reading at 2x20 pins HDD IDE type connector.

In order to interface the PC COM port with the digital hardware, a MAX232 level converter, transforming RS-232 levels back into 0 and 5 Volts, has been used. It includes a Charge Pump, which generates +10V and -10V from a single 5v supply. This I.C. also includes two receivers and two transmitters in the same package.

A LM340T- 5.0 monolithic 3-terminal positive voltage regulator assures a smooth and noiseless 5V power supply for the entire circuit.

The schematic of the communication ensemble is shown in fig. 3.

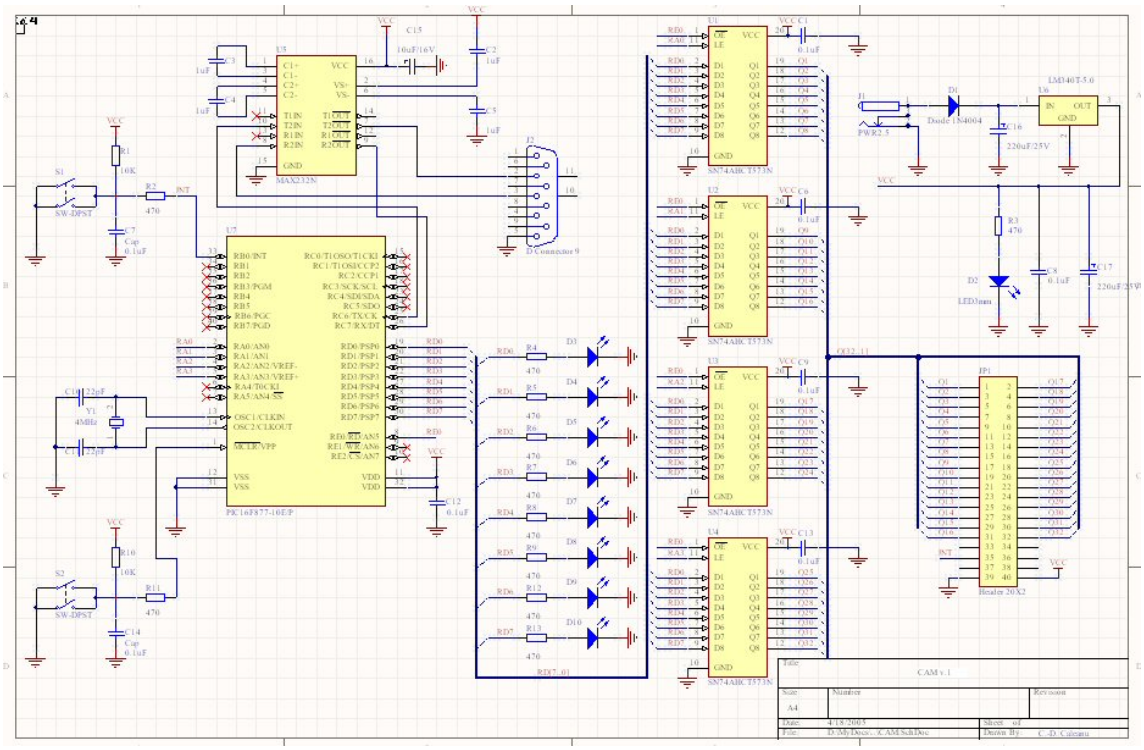

Fig. 3. Using a PIC16F877 uC, 32 bits words are transferred via RS232 from the PC to the CNC machine.

3.2 Programming the PIC16F877 uC

Both the sender (for example, the PC) and the receiver (for example, the CNC milling machine) must be set up to use the same communication parameters. For example, PC COM1 port is set at 19.2Kb/s communication speed (see fig. 4). uC SPBRG register is responsible for the USART communication speed, according to the relation: Baud Rate = FOSC/(16 (*SPBRG value (decimal)* + 1)). Taking into account the 4Mhz oscillator crystal (see fig. 3) SPBRG will be loaded with decimal value 12 in order to have the same 19.2K baud rate as the PC serial port.

4 Conclusion

A software and hardware ensemble were designed and tested towards communicating between PC and a CNC machine trough a microcontroller-based board. Emphasis was placed on the following communication aspects: how to develop an application and program a COM port using Microsoft newest tools, C# and .NET framework, how to design a PIC microcontroller circuit for using the build-in USART module. Schematics and uC programming key issues are also provided.

References:

[1] A.H.G. Al-Dhaher, Integrating hardware and software for the development of microcontroller-based system, *Microprocessors and Microsystems*, no. 25, Elsevier 2001, pp. 317-328.

[2] Jan Axelson, *Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks*, Lakeview Research, 2000.

[3] Tom Archer, *Inside C#*, 2nd Edition, Microsoft Press, 2002.

[4] David S. Platt, *Introducing Microsoft .NET*, 2nd Edition, Microsoft Press, 2002.

[5] Larry O'Brien and Bruce Eckel, *Thinking in C#*, Prentice Hall, 2003.

[6] Jesse Liberty, *Programming C#*, 4th Edition, O'Reilly, Feb. 2005.

[7] *** .NET Framework SDK Documentation, Microsoft, 2006.

[8] www.microchip.com

[9] *** PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers, Microchip Technology Inc., 2003.

[10] Sid Katzen, *The Quintessential PIC Microcontroller*, Springer-Verlag, 2000.