Micronix C# Overview

This guide will assist you in creating C# code to use in conjunction with the Micronix Motion Controller (MMC). Included are C# test program “SOFT_MMC-TEST_V1.0.1” and this guide to set up basic serial connection, sending commands and receiving outputs from the MMC device.

1. Establishing a Serial Connection to the MMC Device via USB

This guide and the C# test program will be discussing on using the classes found in the System.IO.Ports Namespace to connect to the MMC Device.

*For more information about the System.IO.Ports, please refer to Microsoft Documentation: https://docs.microsoft.com/en-us/dotnet/api/system.io.ports

a. Virtual RS-232 configuration settings:

Software Parameter	Setting
Data Bits	8
Stop Bits	1
Parity	No
Handshake	No
Baud rate	38400

This translates to the following C# program:

```
using System.IO.Port;

private void Open()
{
   serialPort1.PortName = “COM3”;
   serialPort1.DataBits = 8;
   serialPort1.StopBits = StopBits.One;
   serialPort1.Parity = Parity.None;
   serialPort1.Handshake = None;
   serialPort1.BaudRate = 38400;

   try{ serialPort1.Open(); }
   catch{ Console.WriteLine(“COM3 is not valid”); }
}
```

The MMC device’s COM port number can be found in Window’s Device Manager under “Ports (COM & LPT)” with the name “USB Serial Port (COMx)”

If the MMC device is not automatically recognized by your computer, you will have to first install the FTDI interface drivers before communicating with the controller. Drivers can be downloaded from:

http://www.ftdichip.com/Drivers/VCP.htm

2. Sending commands to the MMC Device

Commands are sent in ASCII text format. It is important when sending commands to the MMC device is to follow the command line syntax and to send commands ending with the following terminating characters “\n\r” (\n is new line with a hexadecimal value of 0X0A and \r is carriage return with a hexadecimal value of 0X0D)

More information about the Command Line Syntax can be found in our MMC Reference Manual in Section 5. Commands

C# Example Program

```
using System.IO.Port;

private void SerialWrite( string msg )
{
   if (serialPort1.IsOpen)
   {
      if (serialPort1.BytesToRead <= 0)
      {
         System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();
         byte[ ] data = encoder.GetBytes( msg )
         try
         {
            serialPort1.Write(data, 0, data.Length);
            serialPort1.Write( “\n\r” );
         }
      }
   }
}
```

In the example above, if the user would like to send a move relative command for 5 mm to Axis 1, then the user would send “ SerialWrite(“1MVR5”); “ in their program.

3. Receiving output responses from the MMC Device

Outputs from received from the MMC device using query commands (Commands that typically ends with “?” [ie. VER? would query version number of the device and outputs a response in return])

Be careful with sending and receiving commands on the MMC device because the communication protocol RS-485 is half-duplex for our MMC controller, meaning that both send and write are on the same line so read and write cannot occur at the same time. Doing so would cause bus contention. Ways to have proper sending and receiving scheduling would be to add a time delay after sending a query command (typically 20 milliseconds) or by knowing how much bytes to read in the receive buffer [SerialPort.BytesToRead()].

C# Example Program

```
using System.IO.Port;

private void SerialRead( object sender, System.IO.PortsSerialDataReceivedEventArgs e )
{
   if (serialPort1.IsOpen)
   {
      if (serialPort1.BytesToWrite <= 0)
      {
         while (serialPort1.BytesToRead > 0)
         {
            try { Console.WriteLine( serialPort1.ReadLine() ); }
         }
      }
   }
}
```