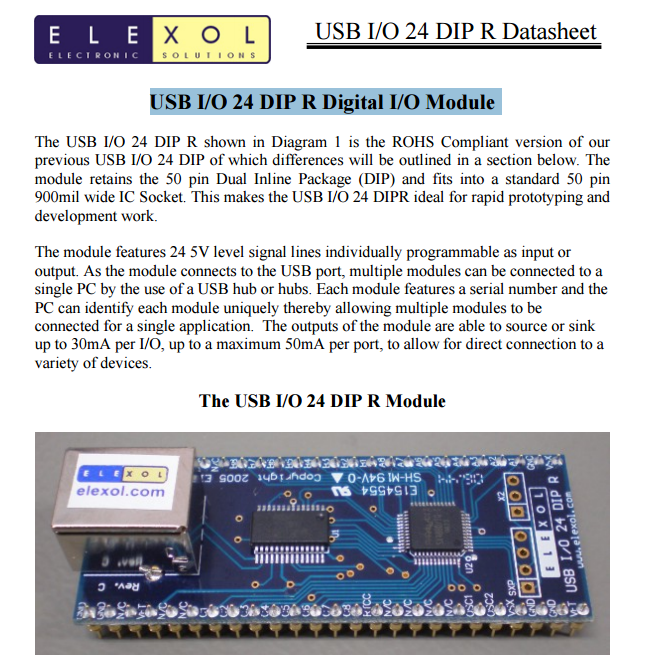
**Writing a C# API for the USIBO24 Elexol**

<https://www.elexol.com/IO_Modules/documents/USBIO24DIPRDS1.pdf>

Following this document, I create a c# class to handle the Elexol USBIO24



The above link for the document explain how to read and write from the Elexol io

There are two options

1. Use serial COM port read and write
2. Use the FTDI driver wrapper in c#

The easy way is to use the Serial Com port read and write as described in the document

|  |
| --- |
| using EleXolIO24RApi;  using System;  using System.Collections.Generic;  using System.IO.Ports;  using System.Linq;  using System.Text;  using System.Threading;  using System.Threading.Tasks;  namespace EleXolIO24RApi  {  public class elexol\_asuart  {  SerialPort \_serialPort = new SerialPort();  byte[] buf = new byte[250];  byte[] m\_direction = new byte[1];  public elexol\_asuart(string COMPORT)  {  \_serialPort.PortName = COMPORT;  \_serialPort.BaudRate = 115200;  \_serialPort.Parity = Parity.None;  \_serialPort.DataBits = 8;  \_serialPort.StopBits = StopBits.One;  //\_serialPort.Handshake = Handshake.XOnXOff;  }  public elexol\_asuart GetBase()  {  return this;  }  public virtual bool Open()  {  \_serialPort.Open();  return \_serialPort.IsOpen;  }  public virtual void Close()  {  if (\_serialPort != null && \_serialPort.IsOpen)  \_serialPort.Close();    }  public virtual void setAllDirection(DIRECTION\_ALL direction)  {  lock (this)  {  string[] strPorts = { "!A", "!B", "!C" };  foreach (string s in strPorts)  {  \_serialPort.Write(s);  if (direction == DIRECTION\_ALL.OUTPUT)  m\_direction[0] = 0x0;  else  m\_direction[0] = 0xFF;  \_serialPort.Write(m\_direction, 0, 1);  }  }  }  public virtual void setDirection(PORT\_NUMBER port, DIRECTION\_ALL direction)  {  lock (this)  {  string portStr = string.Empty;  switch (port)  {  case PORT\_NUMBER.PORTA:  portStr = "!A";  break;  case PORT\_NUMBER.PORTB:  portStr = "!B";  break;  case PORT\_NUMBER.PORTC:  portStr = "!C";  break;  }    if (direction == DIRECTION\_ALL.OUTPUT)  m\_direction[0] = 0x0;  else  m\_direction[0] = 0xFF;  \_serialPort.Write(portStr);  \_serialPort.Write(m\_direction, 0, 1);  }  }  public virtual void setDirection(PORT\_NUMBER port, DIRECTION direction, byte inputPins)  {  lock (this)  {  string portStr = string.Empty;  switch (port)  {  case PORT\_NUMBER.PORTA:  portStr = "!A";  break;  case PORT\_NUMBER.PORTB:  portStr = "!B";  break;  case PORT\_NUMBER.PORTC:  portStr = "!C";  break;  }  if (direction == DIRECTION.OUTPUT)  m\_direction[0] &= inputPins;  else  m\_direction[0] |= inputPins;  \_serialPort.Write(portStr);  \_serialPort.Write(m\_direction, 0, 1);  }  }  public virtual void Write(PORT\_NUMBER port, byte value)  {  string portStr = string.Empty;  switch (port)  {  case PORT\_NUMBER.PORTA:  portStr = "A";  break;  case PORT\_NUMBER.PORTB:  portStr = "B";  break;  case PORT\_NUMBER.PORTC:  portStr = "C";  break;  }  \_serialPort.Write(portStr);  \_serialPort.Write(buf, 0 , 1);  }  public virtual void Write(char port, byte value)  {  \_serialPort.Write(port.ToString());  \_serialPort.Write(buf, 0, 1);  }  public virtual void Write(PORT\_NUMBER port, byte writePins, byte value)  {  setDirection(port, DIRECTION.OUTPUT, writePins);  string portStr = string.Empty;  switch (port)  {  case PORT\_NUMBER.PORTA:  portStr = "A";  break;  case PORT\_NUMBER.PORTB:  portStr = "B";  break;  case PORT\_NUMBER.PORTC:  portStr = "C";  break;  }  \_serialPort.Write(portStr);  \_serialPort.Write(buf, 0, 1);  }  public virtual bool ReadPort(PORT\_NUMBER port, out byte valueRead)  {  valueRead = 0;  string portStr = string.Empty;  switch (port)  {  case PORT\_NUMBER.PORTA:  portStr = "a";  break;  case PORT\_NUMBER.PORTB:  portStr = "a";  break;  case PORT\_NUMBER.PORTC:  portStr = "a";  break;  }  \_serialPort.Write(portStr);  setDirection(port, DIRECTION\_ALL.INPUT);  int timeout = 0;  while (\_serialPort.BytesToRead == 0)  {  Thread.Sleep(10);  if (timeout == 100)  return false;  timeout++;  }  if (\_serialPort.BytesToRead == 1)  {  \_serialPort.Read(buf, 0, 1);  valueRead = buf[0];  Console.WriteLine(buf[0].ToString());  return true;  }  else  {  return false;  }  }  public virtual bool Read(PORT\_NUMBER port, byte InputPins, out byte valueRead)  {  valueRead = 0;  string portStr = string.Empty;  switch (port)  {  case PORT\_NUMBER.PORTA:  portStr = "a";  break;  case PORT\_NUMBER.PORTB:  portStr = "a";  break;  case PORT\_NUMBER.PORTC:  portStr = "a";  break;  }  \_serialPort.Write(portStr);  setDirection(port, DIRECTION.INPUT, InputPins);  int timeout = 0;  while (\_serialPort.BytesToRead == 0)  {  Thread.Sleep(10);  if (timeout == 100)  return false;  timeout++;  }  if (\_serialPort.BytesToRead == 1)  {  \_serialPort.Read(buf, 0, 1);  valueRead = buf[0];  Console.WriteLine(buf[0].ToString());  return true;  }  else  {  return false;  }  }  }  } |

Some notes:

1. The API provides set of overloaded function to read a complete port ( 8 bits)

Or write and read to specific pin

1. Pay attention on the timeout , the read does not work immediately



