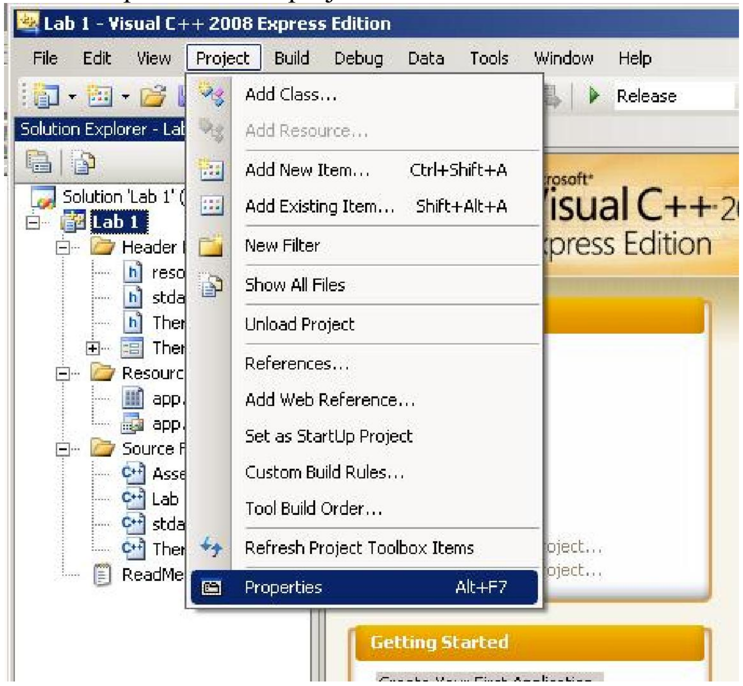
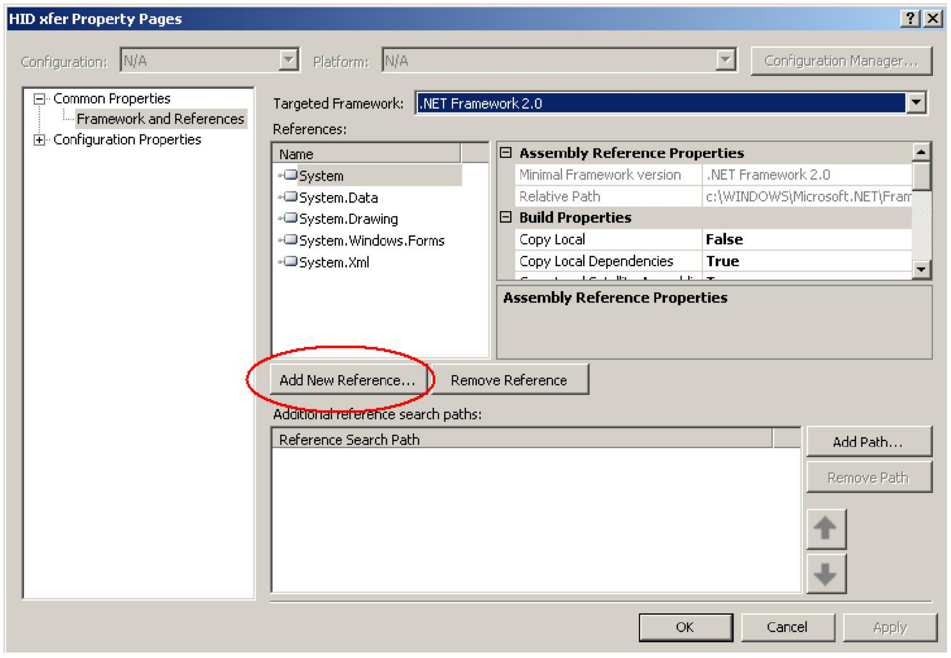
SimpleIO DLL Documentation

Using the SimpleIO DLL with Visual Studio

1. Associate DLL with the project.
   a. Select Properties on the project menu.



   b. Select the Add New Reference button on the Property Pages dialog box



Targeted Framework: .NET Framework 2.0

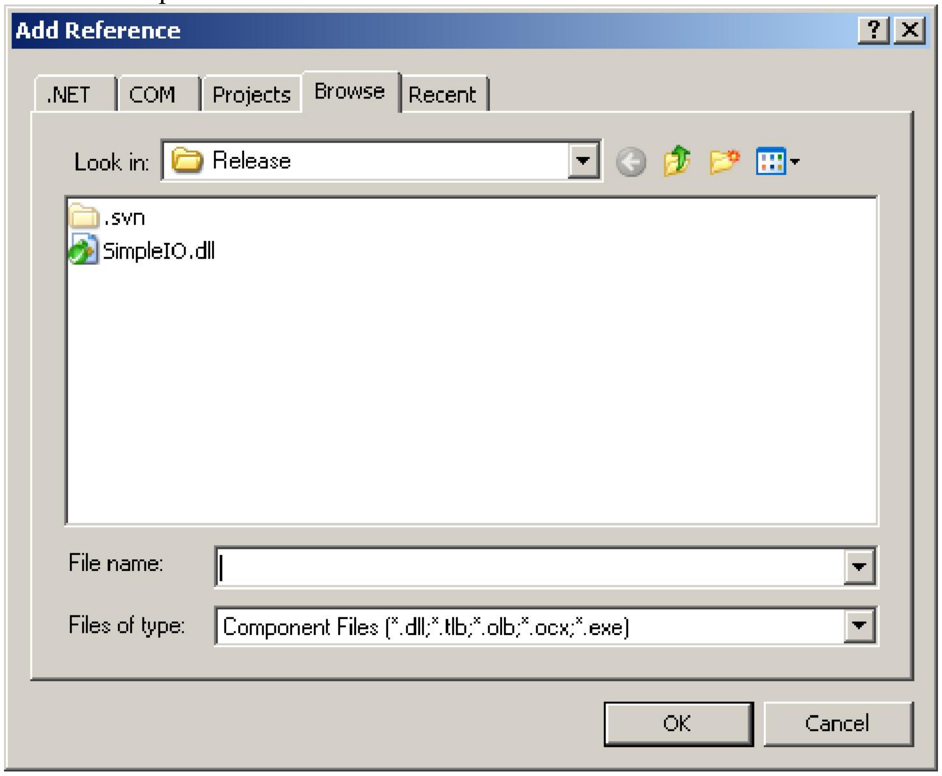
References:

Name
System
System.Data
System.Drawing
System.Windows.Forms
System.Xml

Assembly Reference Properties	
Minimal Framework version	.NET Framework 2.0
Relative Path	c:\WINDOWS\Microsoft.NET\Fram
Build Properties	
Copy Local	**False**
Copy Local Dependencies	**True**

Assembly Reference Properties

c. Select the Browse tab. If necessary navigate to the current \DLL directory. Double click on SimpleIO.dll file.



2. SimpeIO will now be shown in the references pane of the Property Pages dialog box. Click on OK to close the dialog box.

Simple IO API

Summary:

```
SimpleIOClass::InitMCP2200(unsigned int VendorID, unsigned int
ProductID)
bool SimpleIOClass::ConfigureMCP2200(unsigned char IOMap,
                                     unsigned long BaudRateParam,
                                     unsigned int RxLEDMode,
                                     unsigned int TxLEDMode,
                                     bool FLOW,
                                     bool ULOAD,
                                     bool SSPND)
bool SimpleIOClass::SetPin(unsigned int pin)
bool SimpleIOClass::ClearPin(unsigned int pin)
int SimpleIOClass::ReadPinValue(unsigned int pin)
bool SimpleIOClass::ReadPin(unsigned int pin, unsigned int
*returnvalue)
bool SimpleIOClass::WritePort(unsigned int portValue)
bool SimpleIOClass::ReadPort(unsigned int *returnvalue)
int SimpleIOClass::ReadPortValue()
```

While ConfigureMCP2200 configures the device with one call, it may also be configured one parameter at a time:

```
bool SimpleIOClass::fnRxLED(unsigned int mode)
bool SimpleIOClass::fnTxLED(unsigned int mode)
bool SimpleIOClass::fnHardwareFlowControl(unsigned int onOff)
bool SimpleIOClass::fnULoad(unsigned int onOff)
bool SimpleIOClass::fnSuspend(unsigned int onOff)
bool SimpleIOClass::fnSetBaudRate(unsigned long BaudRateParam)
bool SimpleIOClass::ConfigureIO(unsigned char IOMap)
bool SimpleIOClass::ConfigureIoDefaultOutput(unsigned char ucIoMap,
unsigned char ucDefValue)

Constants:
const unsigned int OFF = 0;
const unsigned int ON = 1;
const unsigned int TOGGLE = 3;
const unsigned int BLINKSLOW = 4;
const unsigned int BLINKFAST = 5;
```

---

1. InitMCP2200

```
Function:
      SimpleIOClass::InitMCP2200 (unsigned int VendorID, unsigned int
                                  ProductID)

Summary:
      Configures the Simple IO class for a specific Vendor and product
      ID.

Description:
      Sets the Vendor and Product ID used for the project.

Precondition:
      None

Parameters:
      Vendor ID - Assigned by USB IF (www.usb.org)
      Product ID - Assigned by the Vendor ID Holder

Returns:
      none

Example:
      InitMCP2200 (0x4D8, 0x00DF);

Remarks:
      Call this function before any other calls to set the Vendor and
Product IDs.
```

---

2. ConfigureMCP2200

```
Function:
      bool SimpleIOClass::ConfigureMCP2200 (unsigned char IOMap,
                                            unsigned long BaudRateParam,
                                            unsigned int RxLEDMode,
                                            int TxLEDMode,
                                            bool FLOW,
                                            bool ULOAD,
                                            bool SSPND)

Summary:
      Configures the device.

Description:
      Sets the default GPIO designation, baudrate, TX/RX Led modes,
      flow control

Precondition:
      The Vendor and Product ID must have been specified by
      SimpleIOInit.

Parameters:
      IOMap - A byte which represents the input/output state of the
              pins (each bit may be either a 1 for input, and 0 for
              output.

      BaudRateParam - the default communication baudrate

      RxLEDMode - can take one of the constant values (OFF, ON, TOGGLE,
                  BLINKSLOW, BLINKFAST) in order to define the behavior
                  of the RX Led

                        OFF = 0;
                        ON = 1;
                        TOGGLE = 3;
```

```
                            BLINKSLOW = 4;
                            BLINKFAST = 5;

   TxLEDMode  - can take one of the defined values (OFF, ON, TOGGLE,
                BLINKSLOW, BLINKFAST) in order to define the behavior
                of the TX Led

    FLOW - this parameter establishes the default flow control method
           (false - no HW flow control, true - RTS/CTS flow control)

    ULOAD - this parameter establishes if the pin is configured as
USBCFG status.

    SSPND - this parameter establishes if the pin is configured as
SSPND status.

Returns:
      Function returns true if the transmission is successful and
      returns False if the transmission fails.

Example:

      if (SimpleIOClass::ConfigureMCP2200(0x43, 9600, BLINKSLOW,
                                         BLINKFAST, false, false,
                                         false) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command "

Remarks:
      None
```

3. fnRxLED

```
Function:
      bool SimpleIOClass::fnRxLED (unsigned int mode)

Summary:
      Configures the Rx LED mode. Rx LED configuration will be stored
      in NVRAM

Description:
      Sets the Rx Led mode to one of the possible values and it also
      sets the remaining of the relavant parameters (GPIO designation,
      baudrate, flow control, Tx Led) with the default values as
      they're assigned either at the call to the ConfigureMCP2200()or
      with the default values read back from the device itself

Precondition:
      The Vendor and Product ID must have been specified by
      InitMCP2200()

Parameters:
      mode (constant): OFF, TOGGLE, BLINKSLOW, BLINKFAST
```

```
Returns:
      returns False if the transmission fails.

Example:
      if (SimpleIOClass::fnRxLED (BLINKFAST) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

4. fnTxLED

```
Function:
      bool SimpleIOClass::fnTxLED (unsigned int onOff, unsigned int
      mode)

Summary:
      Configures the Tx LED mode. Tx LED configuration will be stored
      NVRAM

Description:
Sets the Tx Led mode to one of the possible values and it also sets the
      remaining of the relavant parameters (GPIO designation, baudrate,
      flow control, Tx Led) with the default values as they're assigned
      either at the call to the ConfigureMCP2200()or with the default
      values read back from the device itself

Precondition:
      The Vendor and Product ID must have been specified by
      InitMCP2200()

Parameters:
      mode (constant): OFF, TOGGLE, BLINKSLOW, BLINKFAST

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:
      if (SimpleIOClass::fnTxLED (BLINKSLOW) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

5. fnHardwareFlowControl

```
Function:
      bool SimpleIOClass::fnHardwareFlowControl (unsigned int onOff)

Summary:
      Configures the flow control of the MCP2200. The flow control
      configuration will be stored in NVRAM

Description:
      Sets the flow control to HW flow control (RTS/CTS) or No flow
      control

Precondition:
      The Vendor and Product ID must have been specified by
      InitMCP2200()

Parameters:
      onOff -     1 - if Hw flow control needed
                  0 - if No flow control needed

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:
      if (SimpleIOClass::fnHardwareFlowControl(1) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

6. fnULoad

```
Function:
      bool SimpleIOClass::fnULoad(unsigned int onOff)

Summary:
      Configures the GP1 pin of the MCP2200 to show the status of the
      USB configuration

Description:
      When the GP1 is designated to show the USB configuration status,
      the pin will start low (during power-up or after reset) and it
      will go high after the MCP2200 is successfully configured by the
      host

Precondition:
      The Vendor and Product ID must have been specified by
      InitMCP2200()

Parameters:
      onOff -     1 - GP1 will reflect the USB configuration status
```

```
                  0 - GP1 will not reflect the USB configuration status
                  (can be used as GPIO)

Returns:
      Function returns true if the transmission is successful returns
            False if the transmission fails.

Example:
      if (SimpleIOClass::fnULoad(1) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
      lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

7. fnSuspend

```
Function:
      bool SimpleIOClass::fnSuspend(unsigned int onOff)

Summary:
      Configures the GP0 pin of the MCP2200 to show the status of
      Suspend/Resume USB states

Description:
      When the GP0 is designated to show the USB Suspend/Resume states,
      the pin will go low when the Suspend state is issued or will go
      high when the Resume state is on

Precondition:
      The Vendor and Product ID must have been specified by
      InitMCP2200()

Parameters:
      onOff -     1 - GP0 will reflect the USB Suspend/Resume states
                  0 - GP0 will not reflect the USB Suspend/Resume
                  states (can be used as GPIO)

Returns:
      Function returns true if the transmission is successful returns
            False if the transmission fails.

Example:
      if (SimpleIOClass::fnSuspend(1) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

8. fnSetBaudRate

```
Function:
      bool SimpleIOClass::fnSetBaudRate (unsigned long BaudRateParam)

Summary:
      Configures the device's default baudrate. The baudrate value will
      be stored in NVRAM

Description:
      Sets the desired baudrate and it will store it into device's
      NVRAM

Precondition:
      The Vendor and Product ID must have been specified by
      SimpleIOInit.

Parameters:
      BaudRateParam - the desired baudrate value

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:
      if (SimpleIOClass::fnSetBaudRate(9600) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

9. ConfigureIO

```
Function:
      bool SimpleIOClass::ConfigureIO (unsigned char IOMap)

Summary:
      Configures the GPIO pins for Digital Input, Digital Output

Description:
      GPIO Pins can be configured as Digital Input, Digital Output

Precondition:
      The Vendor and Product ID must have been specified by
      SimpleIOInit.

Parameters:
      IOMap - a byte which represents a bitmap of the GPIO
               configuration
               a bit set to '1' will be a digital input
               a bit set to '0' will be a digital output
               MSB  -   -   -   -   -   -  LSB
               GP7 GP6 GP5 GP4 GP3 GP2 GP1 GP0

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:
      if (SimpleIOClass::ConfigureIO(0xA5) == SUCCESS)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

10. ConfigureIoDefaultOutput

```
Function:
      bool SimpleIOClass::ConfigureIoDefaultOutput(unsigned char
      ucIoMap, unsigned char ucDefValue)

Summary:
      Configures the IO pins for Digital Input, Digital Output and also
      the default output latch value

Description:
      IO Pins can be configured as Digital Input, Digital Output
      The default output latch value is received as a parameter

Precondition:
      The Vendor and Product ID must have been specified by
      SimpleIOInit.

Parameters:
      ucIoMap - a byte containing a bit-map used to set the GPIOs as
                either input or output
                1 - GPIO configured as input
                0 - GPIO configured as output
                MSB   -    -    -    -    -    -   LSB
                GP7  GP6  GP5  GP4  GP3  GP2  GP1  GP0

      ucDefValue - the default value that will be loaded to the output
                   latch (effect only on the pins configured as
                   outputs)

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:

      if (SimpleIOClass::ConfigureIoDefaultOutput(IoMap, DefValue) ==
      SUCCESS)
```

```
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;
```

Remarks:
      Error code is returned in LastError

11. SetPin

Function:
      bool SimpleIOClass::SetPin(unsigned int pin)

Summary:
      Sets the specified pin.

Description:
      Sets the specified pin to logic '1'.

Precondition:
      Must have previously been configured as an output via a
      ConfigureIO or ConfigureIoDefaultOutput call.

Parameters:
      pin - The pin number to set (0-7)

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:
```
      if (SimpleIOClass::SetPin (2))
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;
```

Remarks:
      Error code is returned in LastError

12. ClearPin

Function:
      bool SimpleIOClass::ClearPin(unsigned int pin)

Summary:
      Clears the specified pin.

Description:
      Clears the specified pin to logic '0'.

Precondition:

```
      Must have previously been configured as an output via a
      ConfigureIO or ConfigureIoDefaultOutput call.

Parameters:
      pin - The pin number to set (0-7)

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:
      if (SimpleIOClass::ClearPin (2))
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

13. ReadPin

```
Function:
      bool SimpleIOClass::ReadPin(unsigned int pin, unsigned int
      *returnvalue)

Summary:
      Reads the specified pin.

Description:
      Reads the specified pin and returns the value in returnvalue.  If
      the pin has been configured as Digital Input, the return value
      will be either 0 or 1.

Precondition:
      Must have previously been configured as an input via a
      ConfigureIO or ConfigureIoDefaultOutput call.

Parameters:
      pin - The pin number to set (0-7)
      returnvalue - the value read on the pin (0 or 1)

Returns:
      Function returns true if the transmission is successful
      returns False if the transmission fails.

Example:
      unsigned int rv;
      if (SimpleIOClass::ReadGPIOn (0, &rv))
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;
```

```
Remarks:
      Error code is returned in LastError
```

---

14. ReadPinValue

```
Function:
      int SimpleIOClass::ReadPinValue(unsigned int pin)

Summary:
      Reads the specified pin.

Description:
      Reads the specified pin and returns the value as the return
      value.  If the pin has been configured as Digital Input, the
      return value will be either 0 or 1.
      if an error occurs, the function will return a value of 0x8000

Precondition:
      Must have previously been configured as an input via a
      ConfigureIO or ConfigureIoDefaultOutput call.

Parameters:
      pin - The pin number to set (0-7)

Returns:
      Function returns the read value of the pin
      returns a value of 0x8000 if an error occurs

Example:
      unsigned int rv;
      if (SimpleIOClass::ReadPinValue(0) != 0x8000)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Error code is returned in LastError
```

---

15. WritePort

```
Function:
      bool SimpleIOClass::WritePort(unsigned int portValue)

Summary:
      Writes a value to the GPIO port.

Description:
      Writes the GPIO port.  This provides a means to write all pins at
      once instead of one-at-a-time.

Precondition:
      Must have previously been configured as an input via a
      ConfigureIO or ConfigureIoDefaultOutput call.

Parameters:
      portValue - Byte value to set on the port.

Returns:
      Function returns true if the transmission is successful returns
      False if the transmission fails.

Example:
      if (SimpleIOClass::WritePort (0x5A))
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Pins configured for output returns the current state of the port.
      Pins configured as input read as zero.
```

---

16. ReadPort

```
Function:
      bool SimpleIOClass::ReadPort(unsigned int *returnvalue)

Summary:
      Reads the GPIO port as digital input.

Description:
      Reads the GPIO port and returns the value in returnvalue.  This
      provides a means to read all pins at once instead of one-at-a-
      time.

Precondition:
      Must have previously been configured as an input via a
      ConfigureIO or ConfigureIoDefaultOutput call.

Parameters:
      pin - The pin number to set (0-7)
      returnvalue - the value read on the pin (0 or 1)

Returns:
      Function returns true if the read is successful
      returns False if there the transmission fails.

Example:
      unsigned int rv;
      if (SimpleIOClass::ReadPort (&rv))
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;
```

```
Remarks:
      Pins configured for output returns the current state of the port.
      Pins configured as input read as zero.
```

---

17. ReadPortValue

```
Function:
      int SimpleIOClass::ReadPortValue()

Summary:
      Reads the GPIO port as digital input.

Description:
      Reads the GPIO port and returns the value of the port.  This
      provides a mean to read all pins at once instead of one-at-a
      time. In case of an error the returned value will be 0x8000

Precondition:
      Must have previously been configured as an input via a
      ConfigureIO or ConfigureIoDefaultOutput call.

Parameters:
      None

Returns:
      Function returns true if the read is successful
      returns False if the transmission fails.

Example:
      int rv;
      rv = SimpleIOClass::ReadPortValue()
      if (rv != 0x8000)
            lblStatusBar->Text = "Success";
      else
            lblStatusBar->Text = "Invalid command " +
      SimpleIOClass::LastError;

Remarks:
      Pins configured for output returns the current state of the port.
      Pins configured as input read as zero.
```
