precisExcite

Fluorescence LED system

Software

USER MANUAL

Release 1.4.3

2008/07/10

a **COOL**LED LED product

PrecisExcite - Simply Better Control

Contents

1 Introduction

COOLLED's precisExcite fluorescence excitation system uses high powered LED arrays to generate separate peaks of illumination for use in fluorescence microscopy.

Being solid state light sources, it is easy to switch them on and off quickly in submilliseconds and to vary the intensity in 1% steps. This is very different to conventional mercury and metal halide sources which have to be switched on and controlled though mechanical shutters and iris's. LED's have the additional advantages of being very stable and having such a long life that the need to replace the light source is unlikely to be necessary.

The ***precisExcite*** system has been designed to provide the user with a choice of ways of controlling the LED's functions:

a. Manually via switches and variable intensity controls on the remote pod

b. Remotely from a PC using a choice of interfaces and software

This documentation describes the ***precisExcite*** software options and the software development kit (SDK) which should enable most users to integrate the light source into their fluorescence microscopy applications.

1.1 Release Notes

Version 1.4 of the SDK:

- **Remains fully compatible with existing hardware and software.**
- **Adds support for additional channels**. The present precisExcite hardware has three channels, A, B and C, however in the future we anticipate a version or versions with a greater number of channels.
- **Adds support for channel state reporting**. The original precisExcite and SDK reported every change to channel state and also reported after any 5 second period of no data. From version 1.4 firmware, this reporting is turned Off. It may now be controlled by the `XLIVE` command and channels may be queried by commands of the form `C?` (all channels) or `CA?, CB?, etc.` (individual channel).
- **Adds some additional informational properties.** Notably software, firmware and hardware version information, channel labels (normally wavelength) and LAM identifiers.

Version 1.4.1 of the manual:

- Adds documentation on further commands that may be useful to implementers
- Corrects some document layout problems.

Version 1.4.2 of the manual

- Added omitted command AX

Version 1.4.3 of the manual

- Corrected XLIVE options.

2 Hardware Overview

On the reverse panel of the ***precisExcite*** controller, there are a set of three connectors which provide a choice for remote interfacing and control.

- Parallel port TTL interface for remote hardware control
- USB data connection
- TCP/IP (Ethernet) data connection

2.1 Parallel port TTL interface

This is the simplest method for remote control of the LED arrays. On the 15 way D-type connector there is an individual TTL input pin which, when set at a high, will switch on the specific array, overriding any manual settings on the remote pod or other remote control signals. If rapid switching of the LEDs is required, this is the recommended method as it is independent of the state of the program cycle within the ***precisExcite*** main controller.

Full interconnection details are given later in this document.

2.2 Data connection options USB or TCP/IP

The ***precisExcite*** can be controlled either via a USB connection, when it looks and behaves like a serial port, or via a TCP/IP connection like the Internet, where it takes a serial data stream rather like a simple telnet.

Which connection method you choose to use depends upon your needs.

- For many users, the USB connection will probably be the easiest choice as the ***precisExcite*** is connected directly to your PC via a USB port
- For other users TCP/IP may be preferable, particularly where control is needed from a remote location.

3 Software Overview

The ***precisExcite*** interface uses text-based signalling over the USB or Ethernet connection. The commands used are brief text strings terminated with a return or newline character. The commands used are generally simple mnemonics which aids understanding and memorability by the user.

User's applications programmes (e.g. Visual Basic, C++) can interface with the ***precisExcite*** to control the wavelength selection, intensity and LED 'on' times (exposure). For fast switching applications where intensity control is also required, it is recommended that the user's programme uses a combination of software control of intensity and TTL switching of 'on' times.

As an example of a software controller and a useful standard alone program, the ***precisExcite*** PC Software Control Package demonstrates how a PC can communicate with the ***precisExcite*** via a graphical software interface. See below for a detailed description of this package.

As part of the Software Development Kit, a Windows .dll file and a DDE interface are included alternative interface methods from the text-based signalling.

More details of the software structure, the Command Queue and the Interface References are given in the Software Interface section further down this document. There are also examples of simple textual instructions to help demonstrate the requirements from the system software.

4 The *precisExcite* PC Software Control Package

This package provides a graphical software interface and drives the main unit through software instructions either via Ethernet or USB. The power levels for the different wavelengths are set by graphical sliders and switched on and off by checking the appropriate 'On' box. External control of the LED switching can be achieved via the 15 pin D-type parallel TTL port, which offers an individual control signal for each channel.

On time (exposure time) may be controlled using the Pulse feature. The pulse width can be preset from milliseconds to hundreds of hours. The pulse can be triggered by checking the **Pulse** CheckBox or from the Trigger signal on the TTL port.

An additional option is to run a pre-defined script which runs a sequence of wavelengths and power settings. There are also interactive stop/start and loop options.

For a more complete description of the PC application, see the later section of this document **"The PC Application"**.

Installation instructions for this software are given in section "Installing the PC Software", following the important information on the setting up of the USB and Ethernet (TCP/IP) interfacing in the next section.

5 Setting up the USB Interface

Note: We strongly recommend that you install the *precisExcite* software before connecting the USB cable. This is because Windows tries to associate a driver with new hardware it has seen and could, under some circumstances, associate an incorrect driver.

The USB interface uses Microsoft's own built-in drivers for Virtual Serial Ports. This means that ***precisExcite*** behaves very much as an ordinary COM port on your PC and can be controlled by any application that uses Windows COM ports correctly. While this means that no special drivers are required, there is a requirement to tell Windows about our hardware via a `driver .INF file'.

Although this file is often described as a 'driver file', in this case the information it contains tells Windows to use its own internal drivers for ***precisExcite***, telling Windows that ***precisExcite*** uses a Virtual COM Port. This is a file named `precisExcite.inf` and is on the CD. The same file is also copied into the installation folder (usually `c:\Program Files\precisExcite`) so that it is available whenever needed.

When you first plug ***precisExcite*** into your PC via the USB interface, Windows will tell you it has found new hardware. It will ask you for the location of the driver file and advise you to insert the CD that came with your hardware. As the file is on the CD, this is the easiest method to use. If for any reason the CD is not conveniently available, you can instead ask Windows to look in a location you define (as defined above).

Before you can start communicating with the ***precisExcite*** via the USB interface, you must configure it to match the virtual COM port that Windows assigns. Windows Device Manager gives this information and the port is clearly identified as ***precisExcite***.

Getting to Windows Device Manager can be a bit rather long process. If you have a command prompt window or Start>Run, the command `devmgmt.msc` will get you there quickly. Otherwise follow the instructions below.

When you know the COM port, type the port name, e.g., COM6 into the box on the Setting page of the ***precisExcite*** PC application, then use Menu **File->Save Config** to save the change to the application's startup configuration file.

5.1 Setting Up USB on Windows 2000 or XP

Start > settings > Control Panel

Double-click System > Hardware > Device Manager

Click the [+] by ports to see the list of COM ports.

The ***precisExcite*** port will be identified with a string in the form

```
precisExcite USB COM Port (Com4)
```

From that string use the part COM4 only

5.2 Setting Up USB on Windows Vista

Start > Control Panel > System Management > System > Hardware > Device Manager > Continue

Click the [+] by Ports to see the list of COM ports.

The ***precisExcite*** port will be identified with a string in the form

`PrecisExcite USB COM Port (COM).`

6 Setting up the TCP/IP Interface

If you are already on a network, the easy answer is to ask your network administrator to set up ***precisExcite*** for you, however the easy answer isn't always the one available, so we'll go into detail here to guide you.

If you are not already on a network, just accept the numbers that are preconfigured on ***precisExcite*** and set your computer's IP address and NetMask to match, however you will not need to restore your computer's IP address at the later .. you can just leave it as it is.

If you are not already on this network, you will have to change your setting and may have to set up a separate small local network whilst you configure ***precisExcite.***

If you have no knowledge at all of TCP/IP networking, it may be helpful to read the subsection later in this document **"A Simple Guide to TCP, IP Address and NetMask"** before you start, to help you understand what is important about them and why it's important.

6.1 How to do it

If you have the USB connection set up, the settings on ***precisExcite***, it may be easiest to set up the IP address via USB, to avoid having to change the network setting on your PC. ***precisExcite*** allows the use of either a fixed IP address or a DHCP (automatically) allocated IP address and this, too, may best be configured via USB. These settings are on the *Settings* tab of the PC application.

If you are not familiar with this procedure, it is best to read right through this section before you start to make any changes. A little extra time spent now may save some time later.

First set up your computer so that you can can communicate with ***precisExcite***. Your computer needs to be on subnet 192.168.0.xxx with a NetMask 155.255.255.0. To check this...

- on Windows, open a Command Prompt and type `ipconfig`.

  Usually Start->run and type CMD then the enter key,
  or Start->Accessories->Command Prompt,
  or Start->All Programs->Accessories->Command prompt.

- on unix, in any terminal, type `ifconfig`.

In each case find entry described as ``IP Address" or ``IPv4 Address".

Note: You may need Administrator rights to change the TCP/IP settings

Right click on Local Area Connection. If there is more than one, you need to choose the one that applies to your normal network connection.

On the General tab, find and select the Internet Protocol (TCP/IP) entry, then clink Properties

Write down the present settings .. you will need them later.

Enter suitable values for your PC to connect to ***precisExcite***, for example, set your IP Address to 192.168.0.253 and the Subnet mask to 255.255.255.0

Note that all computers on a subnet must have a unique address, so please ensure that you do not try to use an address that someone else already has. This is unlikely to do any damage, but as doing so can cause odd behaviour on your network that my take a little while to explain.

Press OK then OK to change the settings. You may need to restart Windows before it will start using the new settings. Typically Windows will tell you if you need to do this.

Start ***precisExcite***, click the Settings tab, click Connect and you should see a Connected message in the status line at the bottom of the application. If you do not, then something has gone wrong either with the setup, or with the physical network connection to between your PC and the ***precisExcite*** main unit. Check your setup and try again.

Once Connected, enter the new settings you require, check that the values are correct, then press Send. The ***precisExcite*** unit will disconnect and is now provisionally on its new address, but you will later need to commit this data to non-volatile storage for it to remain permanent.

On the ***precisExcite*** Menu, choose Save Config (or use the Ctrl-S shortcut).

If you had to change your PC's network settings to get this far, now restore them to their original values using the same method as before.

Run ***precisExcite*** again, Press Connect and you should again be Connected. Go to the settings tab and press Commit to store the new settings permanently.

IP Address

The IP address of the ***precisExcite*** main unit in the form 192.168.0.252

Netmask

Network subnet mask in the form 255.255.255.0

Gateway

Not Used at present.

TCP Port

TCP port on ***precisExcite***. Use 18259; you are very unlikely to need any other value

Send

Send data to ***precisExcite*** main unit

Commit

Commit the data to non-volatile storage **remember to do this to make the settings in the *precisExcite* main unit permanent**

Restart

Restart the ***precisExcite*** main unit.

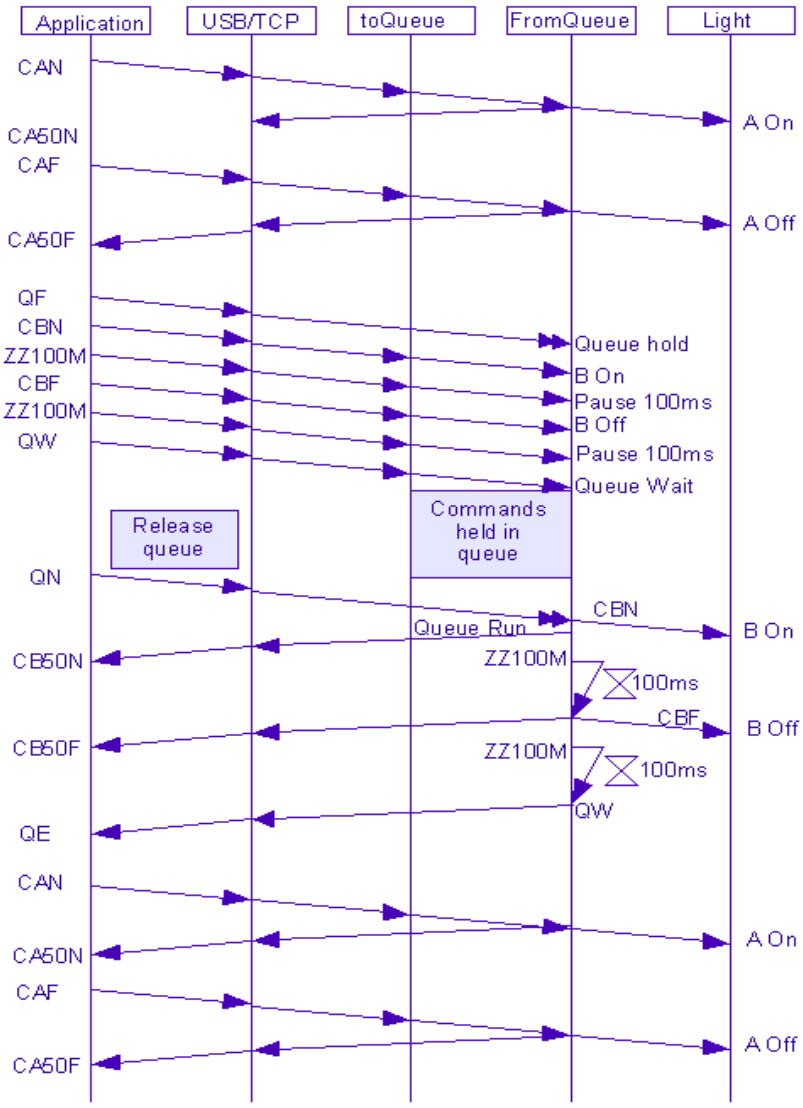
7 Software Interface

7.1 The Command Queue

precisExcite holds almost all commands that are sent to it in a single first-in first-out queue. The exceptions are two queue control commands used for synchronisation and clearing.

The main program loop within ***precisExcite*** takes these commands from the queue and actions them. Some of those commands will have a direct effect on the light sources, for example the On, Off and Intensity commands, whilst some will synchronise with triggers or will cause the main loop to pause for a time before continuing.

This simple structure allows great control over the behaviour and timing of ***precisExcite***'s actions, but does come with some timing constraints that may require consideration. In particular, timing granularity tends to be around 10ms using only queue-based time management.



7.2 Control Text Strings

The base interface between a controlling computer and *precisExcite*, is a simple set of text commands and responses sent via either a TCP/IP interface or a USB "Virtual Serial Port" interface. These command strings are typically just a few characters long, comprise standard ASCII characters and are terminated by a new-line character.

Immediate commands (not queued)

Command	Function	Description
QN	Queue On:	Starts the actions being taken from the queue output.
QC	Queue Clear:	Clears the queue

Queued commands

Command	Function	Description
QF	Queue Off:	Stops the actions being taken from the queue output
QW	Queue Wait:	Queues a mark for return when reached Returns QE

The following commands apply to **each channel**, where the second letter defines the channel, e.g., CAN, CBN, CCN, CDN, CEN, CFN.

CAI100	Channel	Sets the intensity in percent on channel A
CAN	Channel On:	Turns on channel A
CAF	Channel Off:	Turns off channel A
CAPM1000	Pulse Channel:	Turns on channel A for [time] (Not presently implemented) Does not affect intensity setting Time S is integer seconds Time M is integer milliseconds Time U integer microseconds
AA+	Arm Channel:	Arm channel A to fire on TTL rising edges
AA-	Arm Channel:	Arm channel A to fire on TTL falling edges
AA*	Arm Channel:	Arm channel A to fire on TTL either/both edges
AA#	Arm Channel:	Arm channel A to follow pulse on TTL input
AX	Arm Cancel	Revert to TTL direct controls
CA?	Chan Query	Elicits a channel state message from precisExcite

The following commands contain no channel data

AP+	Arm Pulse:	Arm present channel for precise pulse on TTL rising edge
AQ+	Arm Queue:	Continue the queue on TTL rising edge
ZZM100	Pause:	Pause until time has passed Time as before, but excluding microseconds
LAMS	LAM enquiry	Elicits information about the installed LAMs/Channels
XVER	Version Enq.	Elicits information about the software and hardware versions. (Also LAM-type data from existing precisExcite.)
XLIVE	Reporting	XLIVE=YES reports channel state regularly XLIVE=NO reports only as a response to channel
C?		Elicit a state response for all all channel.
IP		Reports IP address.

Response data	
XVER=XXXXX	The firmware version of the precisExcite as a string.
XHEAD_VER=XXXXX	The firmware version of the precisExcite LED head as a string.
XPOD_VER=XXXXX	The firmware version of the precisExcite Pod as a string.
XDATA_VER=XXXXX	The non-volatile data version as a string.
XHW_VER=XXXXX	The hardware version of the precisExcite as a string.
XCPU=XXXXX	The CPU version of the precisExcite as a string.
XLAM_L=X	The Left-hand LAM as a digit
XLAM_R=X	The Right-hand LAM as a digit
LAM:A:XXXXX	LAM/Channel A label, e.g., 400nm. Ditto LAM:B: LAM:C: etc.
LAMA=XXXXX	The LAM A type identifier as a string. Ditto LAMB, LAMC, etc.
IP=192.168.0.252	(Not yet implemented on hardware) The IP address in use by the precisExcite unit.

7.3 Miscellaneous Commands

The following commands generally have the form XABCD as an enquiry and XABCD=Something to set the value. Exceptions are indicated. NV is non-volatile data (see XCOMMIT)

Command	NV	Data	Comment
RESET			Forces a reset of precisExcite and control Pod.
XREBOOT			Ditto
XIP	Y	IP-Address	Sets the IP address to use, e.g.: XIP=192.168.0.252
XNM	Y	Netmask	Sets IP netmask; not presently used
XGW	Y	Gateway IP addr.	Sets gateway IP; not presently used
XNS	Y	DNS IP addr.	Sets DNS IP; not presently used
XSTATIC	Y	YES/NO	Tells to use static(Yes) or DHCP(No) IP.
XDHCP	Y	YES/NO	Tells to get IP address from DHCP server.
XPORT	Y	IP Port number	Normally use 18259
XMAC			Read-only. Ethernet MAC address
XUSE	Y		Configuration settings. **Do Not Change!** **Operational and safety implications.**
XDBG	Y		Debug setting (only useful with special probe)
XTE			Report Peltiėr PWM.
XRADCX			Report ADCX reading (X is A..L)
XVIRGIN	Y		Set to everything to factory defaults, including serial number. **Auto-COMMIT!**
XCOMMIT	Y		**Writes NV-data to the NV store to make the changes permanent.**

7.4 Programming Library Interfaces

The PC Interface toolkit also supplies some programming library options, in particular a Windows™ DLL file with both .NET (managed code) and COM-Interop (unmanaged code) presentations.

8 Programming examples

The software installation includes a number of examples that control ***precisExcite*** by a number of methods. These include an example controlling via a USB Virtual COM port from the Visual Basic for Applications (VBA) that is built into Microsoft Word, an example using the same VBA in Word to control via the .dll. You'll need to **Enable Macros** to use these. Windows will pop up a box asking you if you want to do that. There is an example using a DDE connection and example scripts that the ***precisExcite*** PC application can run.

8.1 Using the Character String Interface

This is the most fundamental method by which ***precisExcite*** may be controlled. The method may be used via either the TCP interface or the USB interface. In both cases the text commands are processed by the same internal queuing functions.

Commands always consist of a string of characters terminated by an end of line character, either <cr> or <nl>.

The list of supported command is given in tabular form in the section **"Interface Reference/Text Commands"**.

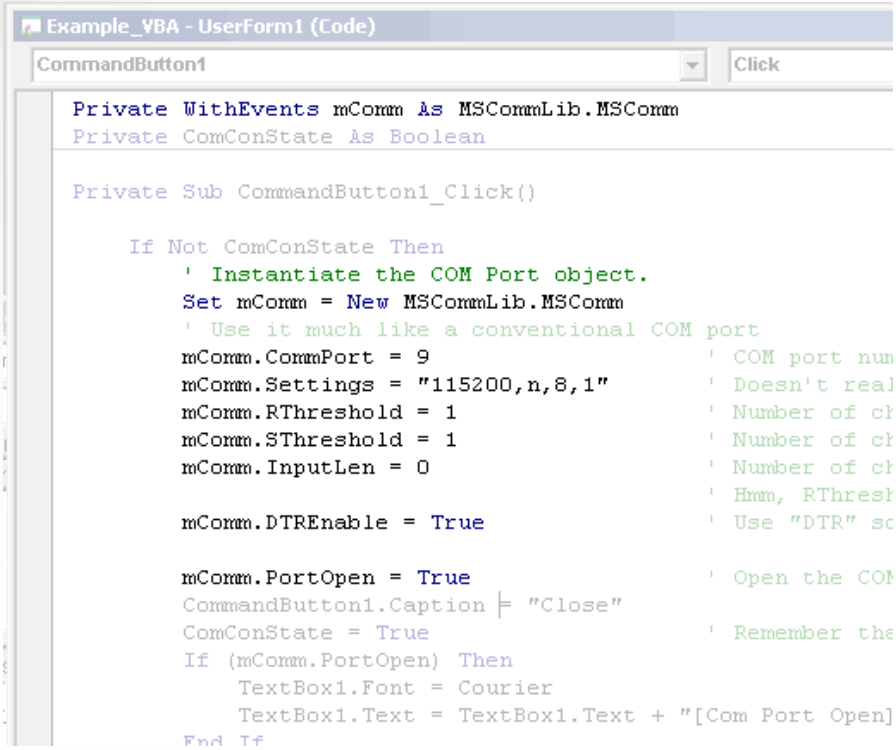
The command have reasonable mnemonics for easier learning. The general structure of a command is to begin with a C for a channel or a Q for a queue administration command. So to turn on channel A, our command is "Channel A oN", command CAN<cr>, to turn it off is "Channel A ofF", command CAF<cr>. "Channel B Intensity 75%" is CBI70<cr>. "Channel C Pulse for 100 ms" is CCP100M.

The Queue is administered by "Queue oN", command QN<cr>, which allows the queue to run, "Queue ofF", command QF<cr>, which holds the queue until it sees a QN<cr>, "Queue Wait", QW<cr> which command is placed in the queue and will signal to us when it is reached using a QE<cr> or "Queue End" message. Finally "Queue Clear", QC<cr> tells ***precisExcite*** to discard everything in its queue.

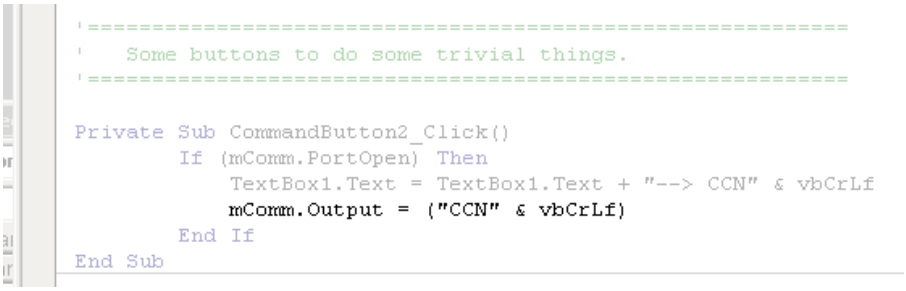
"ZZ" commands cause precisExcite to sleep for the specified time, so ZZ100M<cr> means sleep for 100 Milliseconds.

The VBA example uses a Windows standard ActiveX control MSCOMM32.OCX. This file must be selecyed via the **References** menu option (under **Tools** in VBA in Word 2003, sometimes elsewhere). Select Browse, go to the C:\WINDOWS\System32 folder, look for files of type .ocx and select MSCOMM32.OCX, then press Open. In the check-box list in the References-Project window, check the box for Microsoft Comm Control 6.0 and press OK. You may have to close and open the editor to find this entry and later to find the Comm icon (a telephone) on the Toolbox bar. Once you have this, drag and drop a copy of the icon onto your application's form.

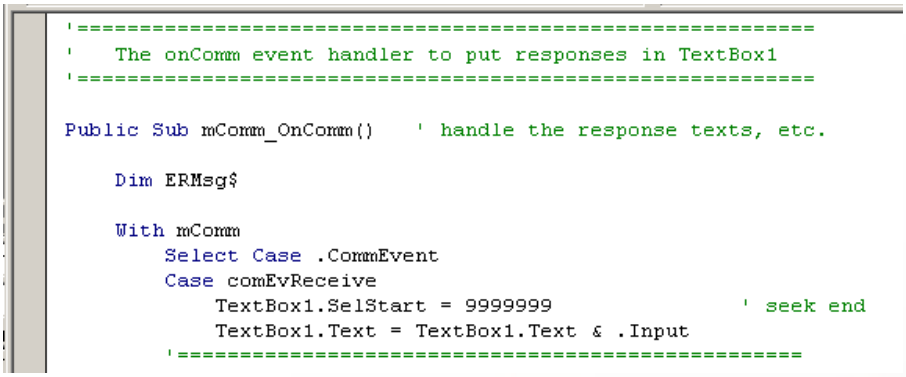
In the code you will need to declare an object, instantiate it and you'll need some setup-code in a function, e.g.:



Some more code in that subroutine later closes the port in a similar way and makes it's reference `Nothing'. You'll need the `write to precisExcite' line itself:



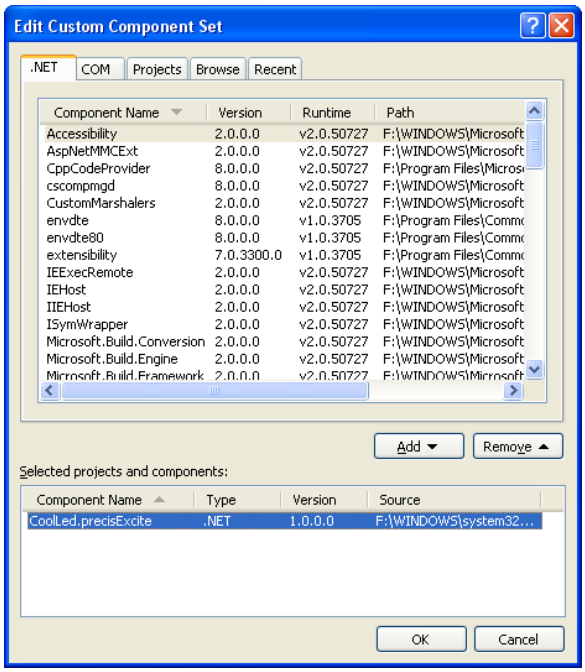
You may also want or need a delegate routine to handle the received data:



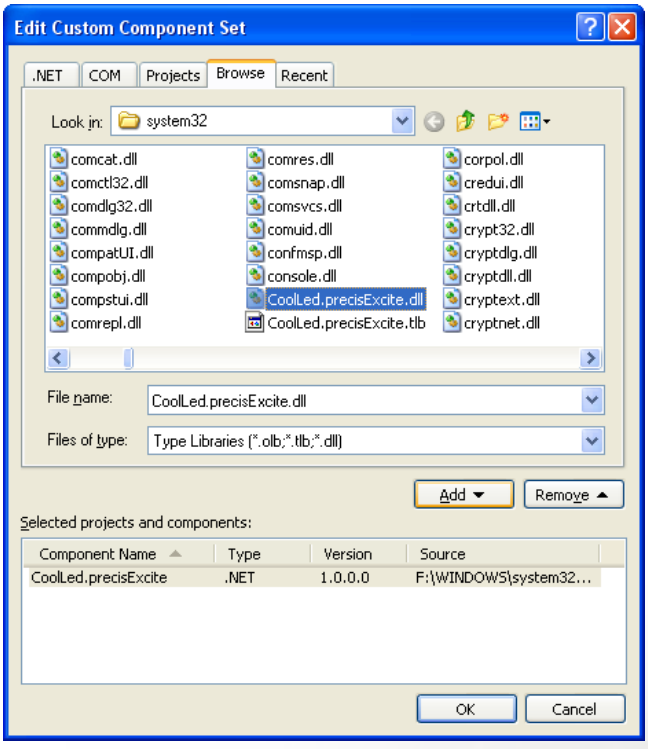
8.2 Using the DLL Library Interface

The VB2005 example uses a .NET `managed code' library, which is also made to the COM-Interop standard for use with the pre-.NET version of VB and VBA.

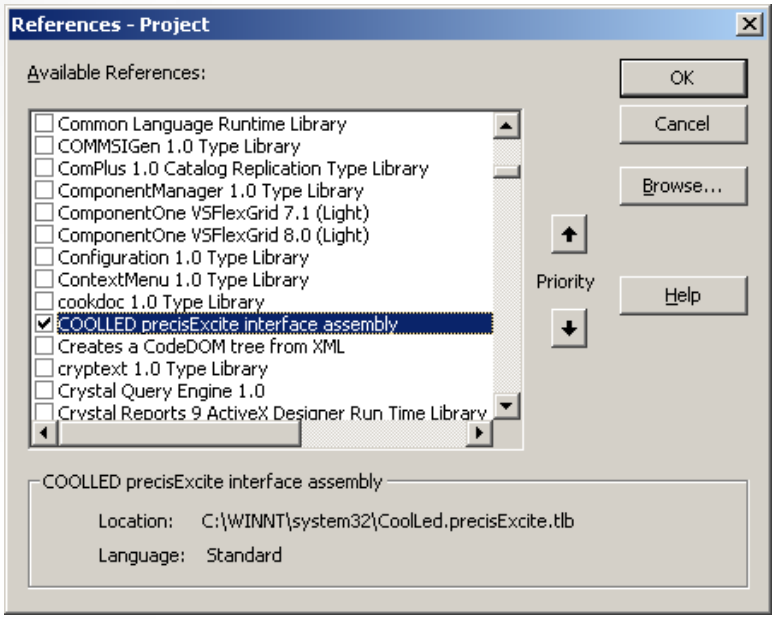
From the Object Browser, select Custom Component Set, then the "...", then from the .NET components select CoolLed.precisExcite and press "Add" then press "OK".



If the component set is not shown, you may have to locate it using the Browse tab, the "Look In" entry, locate the file: C:\WINDOWS\System32\CoolLed.precisExcite.dll, press "Add", then press "OK".



In some tools, you may need menu "Tools"-> "Add Reference" or "Project"->"Add Reference", then find and check the CoolLed.precisExcite entry.



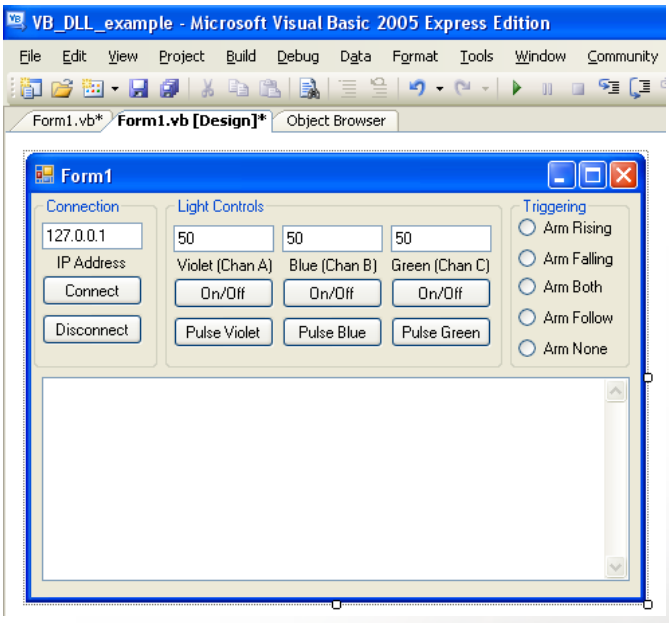
On a system without a .NET framework, you may have to use the "Custom Component Set" as above, locate and add the file C:\WINDOWS\System32\CoolLed.precisExcite.tlb.

Whichever of the above methods you use, the Objects that result will look similar.

8.2.1 The example application.

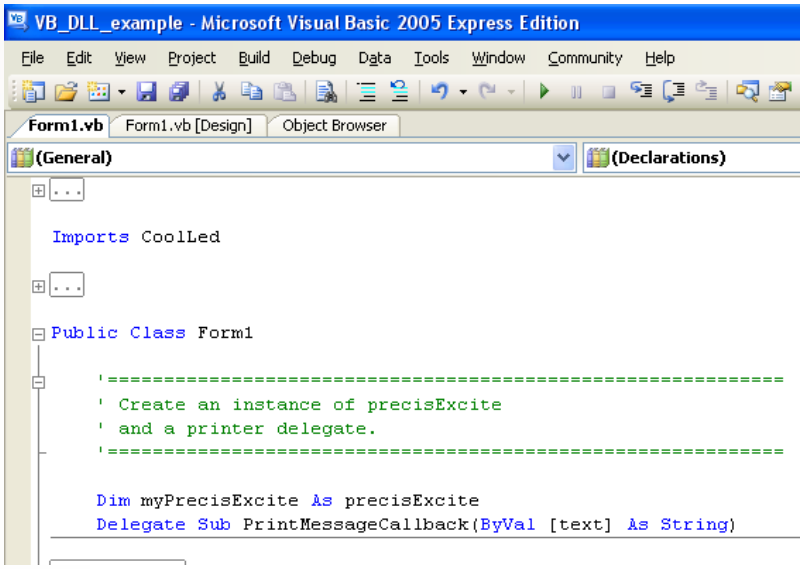
The example is in Visual Basic 2005. Other IDEs and languages differ in detail, but the essence is similar.

The example has a Windows Form that comprises a number of buttons and text boxes that allow us to demonstrate most of the essential function, without producing a huge and complex application.



The application needs to Import the library for ***precisExcite***, then instantiate a precisExcite object and a delegate subroutine to handle the text back from ***precisExcite***. The we need the various bits of code that will control things.

The Import and Instantiation code right at the beginning of the example looks like this. The Import line ensures the application gets the Namespace and interface definitions.



Our example has Connect and Disconnect buttons, in this case for the TCP/IP method. When we press the Connect button, it runs the following code to set up a connection to ***precisExcite***, via Ethernet. In our example, the IP address is taken from a text box and is 127.0.0.1, which is a self-reference address for the machine on which the application will run. That address will connect to the ***precisExcite*** simulator. To connect to a real ***precisExcite***, enter its IP address in the text box.

```
' Connect to the IP address on the form
Private Sub ButtonConnect_Click(ByVal sender As System.Object, ByVa
    myPrecisExcite.Address = TextBoxAddress.Text
    myPrecisExcite.Connect()
End Sub
```

The Disconnect code is very similar.

Once the application has connected, we can use the other controls. The various On/Off buttons have self-toggling structure as below.

```
' Send the various messages to precisExcite to make it do
' things.
' The code here just flips the state between On and Off
Private Sub ButtonChanA_Click(ByVal sender As System.Object, ByVa
    If (myPrecisExcite.ChanAState = True) Then
        myPrecisExcite.ChanAState = False
    Else
        myPrecisExcite.ChanAState = True
    End If
End Sub
```

We have also an example setting the light intensity. Note that although intensity is in integer percent, for compatibility between .NET and COM code presentations, we have use an Unsigned Integer for this data.

```
    '============================================================
    ' The code here send the new intenisty (in percent) when we
    ' leave a text box (tab, mouse click elsewhere, etc.)
    '============================================================
    Private Sub TextBoxChanA_TextChanged(ByVal sender As System.Object, ByVal e
        myPrecisExcite.ChanALevel = UInteger.Parse(TextBoxChanA.Text)
    End Sub
```

The Pulse example uses a fixed time of one second, expressed as 1000 milliseconds.

```
    '============================================================
    ' This code asks for a timed pulse. The time is hard-coded here,
    ' but a real application might want it configurable.
    ' Time is in milliseconds, though precisExcite presently has
    ' about a 10ms granularity.
    '============================================================
    Private Sub ButtonPulseA_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        myPrecisExcite.ChanPulse(precisExcite.Channels.A, 1000) ' time is in milliseconds
    End Sub
```

Management of the Trigger functions is a little more complex as there are several variations in the way triggering may be used. The example code shows all of these, but for the following example, we show the general structure and highlight just one trigger method. In the example, Channel A will light for 100 milliseconds in response to a rising edge on the Trigger TTL input.

```
    '============================================================
    ' This requests a trigger method from whichever radio button we just clicked.
    '============================================================

    Private Sub RBArmRising_CheckedChanged(ByVal sender As System.Object, _
                                           ByVal e As System.EventArgs) _
                                           Handles RBArmRising.Click, _
                                                   RBArmFalling.Click, _
                                                   RBArmBoth.Click, _
                                                   RBArmFollow.Click, _
                                                   RBArmNone.Click

        If RBArmRising.Checked Then
            myPrecisExcite.ChanArm(precisExcite.Channels.A, _
                    precisExcite.TriggerTypes.RisingEdge, _
                    100)
        ElseIf RBArmFalling.Checked Then
            myPrecisExcite.ChanArm(precisExcite.Channels.A, _
                    precisExcite.TriggerTypes.FallingEdge, _
                    100)
        ElseIf RBArmBoth.Checked Then
```

Properties.

Property	Type	Value	Description
Address	String	IP Address	Example: 192.168.0.252
Port	Long	Port Number	Default 18259
COMPortName	Enum	COMx	Virtual COM port to use
ConnectionType	Enum	.TCP	Use TCP for connection
	Long	.Serial	Use USB for connection
ReceivedData	String		Message strings returned from precisExcite
ChanAState	Boolean	True	Channel A On (all other channels off)
ChanAState	Boolean	False	Channel A Off
ChanALevel	Long	0..100	Light level setting of channel A in percent
ChanBLevel etc. ditto			
ChanATime	Long	$0..2^{31}$	Pulse duration for channel A in milliseconds
ChanBTime etc., ditto.			
ChanALabel	String	(readonly)	Normally the wavelength, e.g., "400nm"
ChanBLabel etc., ditto.			
LAM_L/LAM_R	String	(readonly)	Lam ID codes for left/right LAMs.
LAMA	String	(readonly)	Lam IDs for LAMs A to F.
pExVersion	String	(readonly)	PrecisExcite firmware version
CPU	String	(readonly)	PrecisExcite CPU version
PodVersion	String	(readonly)	PrecisExcite Pod firmware version
HeadVersion	String	(readonly)	PrecisExcite LED head firmware version
HW_Version	String	(readonly)	PrecisExcite hardware version
ConfigDataVersion	String	(readonly)	PrecisExcite non-volatile data version
SerialNumber	String	(readonly)	PrecisExcite serial number
LiveData	Boolean	True	Report all state/level changes
	Boolean	False	Respond only to change and enquiry commands (default condition from firmware version R1.4)
DelMessageHandler	Delegate		Delegate function to handle text response strings.

```
Public Delegate Sub DelegateMessageHandler(ByVal msg As String)
```

Constructors

Method	Description
precisExcite	Class Constructor
IprecisExcite	Interface Definition

Methods

Method	Arg	Type	Description
Connect	None		Connect to precisExcite on Address
Disconnect	None		Disconnect from precisExcite
Poll	None		Collect return text strings
ChanOn	Channel	Enum	Turn the named Channel ON
ChanOff	Channel	Enum	Turn the named Channel Off
ChanPulse	Channel	Enum	Piulse the named Channel
	Time	Long	Pulse duration in milliseconds (0..2^{31})
ChanArm	Channel	Enum	Arm the named Channel (for TTL trigger)
	Time	Long	Pulse duration in milliseconds (0..2^{31})

The final pieces of our application are the Poll and Delegate routines.

The Poll routine invites the .dll to send strings to us. This should be run in a BackgroundWorker thread or equivalent, to avoid blocking of the main thread. Blocking is where the routine does not return until some data or event occurs and we cannot tolerate that wait.

```
'============================================================
' The background worker thread.
' The Poll routine presently blocks, so we definitely _don't_
' want this code on the main thread.
'============================================================
Private Sub BackgroundWorker1_DoWork(ByVal sender As System.Object,
    While True
        myPrecisExcite.Poll()
    End While
End Sub
```

The delegate routine is called by the .dll when it has received text from preecisExcite. This text includes the reports of the present On and Off state and intensity levels of the channels and may also contain other string like the ``Hello" message at connect time. These string may be ignored if you wish. The example just displays them in the text window. The strings have already been used by the .dll to update properties.

```
#Region "Background worker thread and Calbacks"

    '============================================================
    ' This is the callback to handle printing to the text window.
    ' There code within checks to see whether we can print
    ' immediately, or whether we should "Invoke" a callback.
    '============================================================
    Private Sub PrintMessage(ByVal [Text] As String)
        Console.WriteLine([Text])

        ' InvokeRequired required compares the thread ID of the
        ' calling thread to the thread ID of the creating thread.
        ' If these threads are different, it returns true.
        If Me.TextBoxTrace.InvokeRequired Then
            Dim d As New PrintMessageCallback(AddressOf PrintMessage)
            Me.Invoke(d, New Object() {[Text]})
        Else
            Me.TextBoxTrace.AppendText([Text])
        End If
    End Sub
```

8.3 Using the DDE Interface

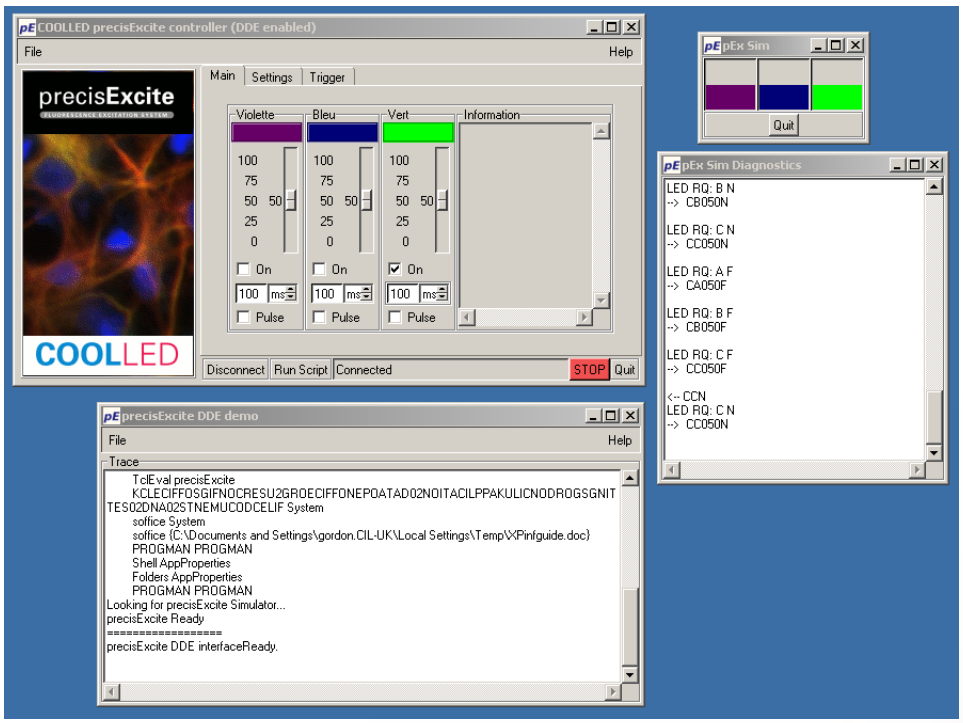
The DDE interface is an interface to the *precisExcite* PC application, rather than direct to the *precisExcite* unit itself.

This interface uses the **Scripting** engine within the PC Application to control both that application and in turn to ***precisExcite*** itself. This may be useful, particularly in circumstances where it will be helpful to see a local display of the ***precisExcite*** behaviour from a remote location. As an example, the PC application will show the light levels and on-off state of the unit, even if the connection between them is over the Internet and the unit a on the other side of the world.

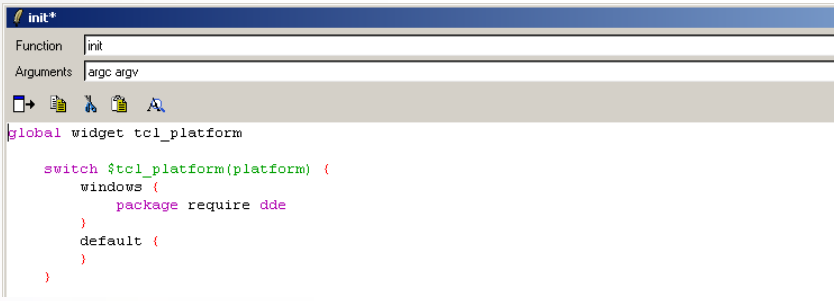
The DDE example was written using the Tcl/Tk scripting language (www.tcl.tk) and the VisualTcl GUI builder (vtcl.sourceforge.net) that we used to build the PC application and compiled with ActiveState's TclDevKit.

The example searches for a ***precisExcite*** PC application, connects to it, sets the IP address to 127.0.0.1, asks the application to Connect and runs a sequence of commands. The application connects to the Simulator, sends the commands and the simulator's levels and on/off states will change to match.

To run the example, start first the simulator pEx-sim.exe, then the ***precisExcite*** application precisExcite.exe and finally the DDE Demo pExDDE-demo.exe, which runs immediately.



The Tcl application needs a library:



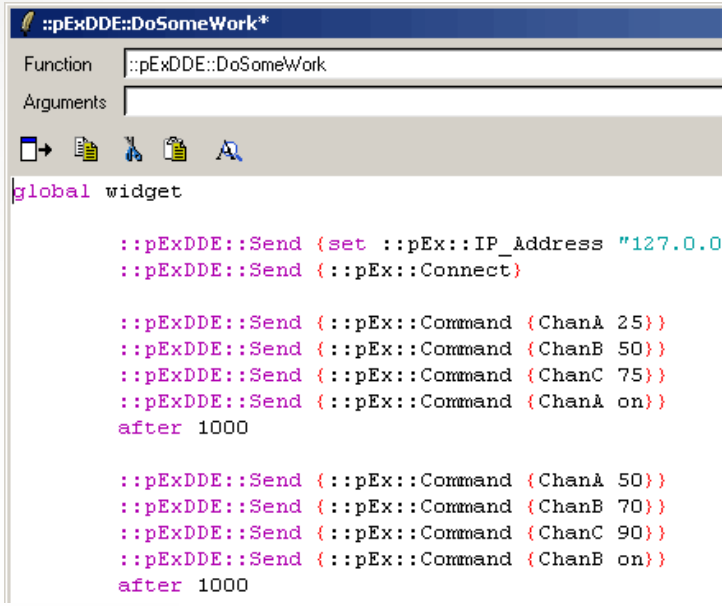
It then searches for a precisExcite. The server announces itself as TclEval, the name by the the inner Tcl interpreter and offers the Topic precisExcite.

```
set ::pExDDE::ddeserv {}

switch $::pExDDE::DDE_Application {
TclEval {
        ::pExDDE::Trace "Looking for precisExcite Simulator..."
        set ::pExDDE::ddeserv [dde services TclEval precisExcite]
        #::pExDDE::Trace "\t\t...precisExcite?: <$::pExDDE::ddeserv>"
    }
default {
        ::pExDDE::Trace "Looking for precisExcite system..."
        set ::pExDDE::ddeserv [dde services $::pExDDE::DDE_Application
```

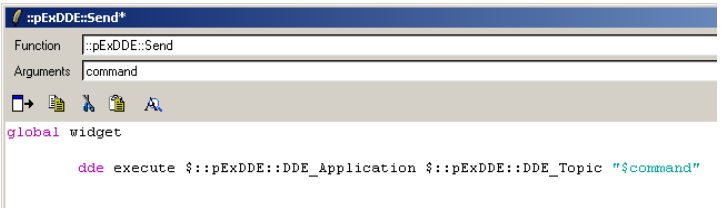
Once the application has found the serverand connected, it asks for some work to be done by another routine:

```
        if { $::pExDDE::ddeserv != {} } {
            ::pExDDE::Trace "precisExcite Ready"
            ::pExDDE::Trace "==================="

            #    ========================
            #    OK, lets go do some stuff
            #    ========================
            after 1000 {::pExDDE::DoSomeWork}

        } else {
            ::pExDDE::Trace "precisExcite system not found"
        }
```

The worker routine sends a number of DDE commands.to the PC application. These are executed by the application's function "::pEx::Command" and use the supplied parameter.



Finally, we have a wrapper to make the lines in DoSomeWork more programmer friendly:



9 Troubleshooting

1. There is a known problem with the TCP connection if ***precisExcite*** is physically disconnected from the Ethernet connection without first disconnecting the TCP in software, where ***precisExcite*** will not subsequently reconnect. Please turn off the power to the unit for around 10 seconds, then repower. The unit should now connect normally.

2. Occasionally and particularly if precisExcite was plugged into a USB port before the drivers were installed, Windows fails correctly identify the ***precisExcite*** hardware. If this happens, use Device Manager to reinstall the drivers. On the list of devices you will see a device with an exclamation mark, usually named ***precisExcite***, but sometimes just Unknown Device. Double click this icon and follow the instructions to reinstall the drivers. The file `precisExcite.inf` on the CD contains the information for Windows to use for this task and it simply tells Windows to use its own built-in driver. This problem can sometimes be caused by a poor USB connection or an over-long USB cable.

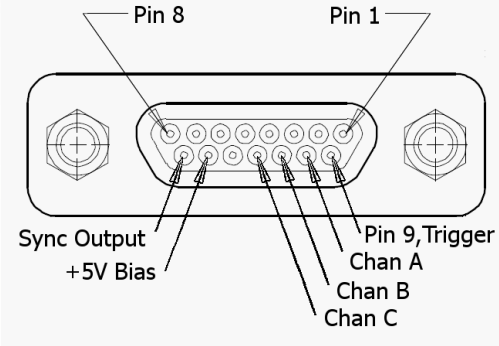
3. **Interlock Shutdown**: precisExcite has an electrical interlock system so that the LED lights cannot be turned on when the unit is not completely installed. This is primarily to comply with European Laser regulations, under which LED light sources are classified, however it is also a useful safety feature. PrecisExcite will give this message if any of the light guides, interlock lead or collimator head are not installed correctly. A few early model ***precisExcite*** units had an oxidation problem on some electrical contacts. If everything seems connected OK but you still have an Interlock Shutdown error, please call us for further advice.

4. On **laptop computers running on battery power**, we have seen a number of cases when using TCP/IP, where the laptop de-powers the network card despite being set to **not** de-power on battery. The only solution appears to be to use a power lead.

5. **COM ports "In Use" and failure to use ports above COM15.** This seems to be a bug in the Windows driver. Windows periodically finds precisExcite `again' and allocates a new virtual COM port. This has a couple of effects. Firstly the COM port number changes and secondly, COM ports above COM15 are often not accessible to applications. Open Device Manager, go to Ports (COM & LPT) and click the [+] expander. Right-click the precisExcite entry. Select Properties, then Port Settings, then Advanced. Use the COM Port Number spinbox to select a sensible port number. Typically Windows protests that the port is "In Use", however normally it will accept the new entry anyway.

6. We have had an instance with TCP/IP where a user reported failure to connect when the Ethernet was set to "10Mbit full", but where it worked fine when set to "Auto". We don't presently know why that might be.

10 TTL Interface Connections



Pin	Signal	
1	0V	
2	0V	
3	0V	
4	0V	
5	0V	
6	0V	
7	0V	
8	0V	
9	Trigger Input	See description
10	Channel A	+ve turns light On
11	Channel B	+ve turns light On
12	Channel C	+ve turns light On
13	Channel D	+ve turns light On
14	+5V/1k bias	Available for switching
15	Sync Output	+ve indicates On

Channel Controls

TTL signals have some specified voltages between 0 and +5 Volts. Provided the signals you have are "TTL compatible", the connections should work together sensibly. AS an example, the Parallel or Printer port on a PC is TTL Compatible.

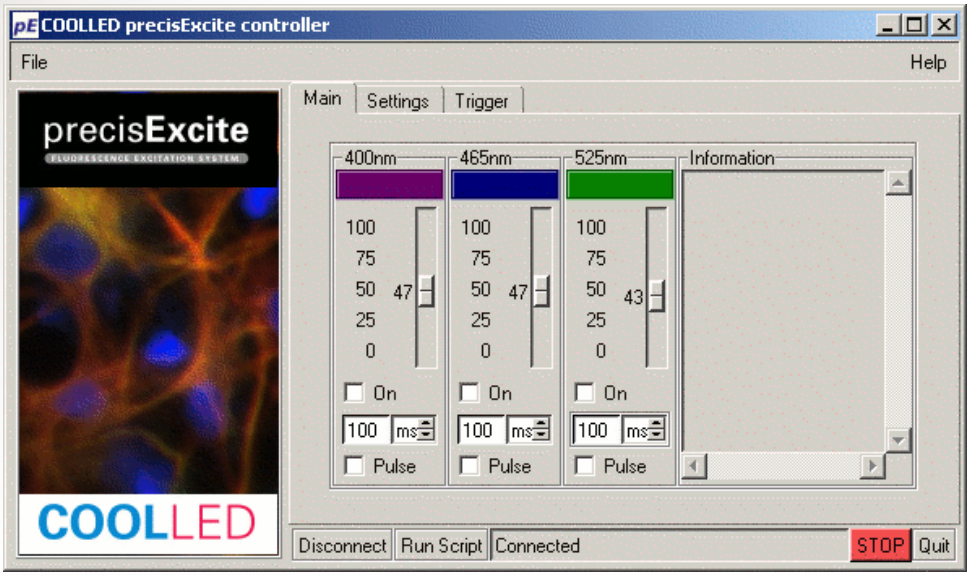
Channels A, B and C each control one of the output colours. When the TTL signal is "high", which is a value between around 1.5V and 5V, each of these lines turn their respective light On. Note that at present only one channel may be on at one time and that Channel A takes precedence over Channel B, which in turn takes precedence over Channel C.

The **Sync Output** indicates that a channel is On. It makes no attempt to indicate which channel that is.

The **Trigger Input** may be used in various ways including pulsing a channel upon rising TTL edge, falling TTL edge or both edges. It may also make a *precisExcite*s script continue execution. The section on Triggering gives information about how to use this input.

The **+5V/1k bias** supplies a fixed 5 Volts out via a 1k resistor and is available as a pull-up to bias switches and open drain/collector semiconductors.

11 The PC Application



File Menu

There are options on this menu allow you that allow you to load or save configuration files. ***precisExcite*** loads a file `precisExcite.rc` at startup.

There is an option that allows you to load a script file, see Automation.

Help Menu

Offers the usual About and Help windows. The former gives information about hardware and software revisions, the latter should take you to this file.

Main Tab

Selects the normal operations window. This is where you will normally work when using ***precisExcite***.

Settings Tab

Selects the set-up window. This is where you will set up the network parameters. ***precisExcite*** the first time or in a new environment. See Setting up ***precisExcite***.

Trigger Tab

Where you will set up the behaviour of the TTL-compatible external trigger inputs.

Lamp On/Off indicators

Indicate the ON/OFF state of the illuminator's lights, but do not attempt to indicate the light intensity.

Light Level Controls

Control and show the intensity setting of the illuminator's lights, but do not attempt to show the ON/OFF state.

Light On/Off switches

Turn each light on and off.

Note that only one colour at a time can be used, so turning on one colour will immediately turn off any other colour that is on.

Exposure time settings

Set the exposure time in seconds, minutes or hours. Resolution is presently to one decimal place. Much shorter times and higher precision are planned but not yet ready for release.

Please contact us if you want shorter times.

Light Pulse switches

Begin a light pulse (timed exposure).

Information window

A window for additional information about the system.

Connect to or Disconnect from *precisExcite* unit

Connect to or disconnect from the ***precisExcite*** main unit. Note that you may have to configure some settings before you can do this.

See Setting up ***precisExcite***.

Run a script

Run a script to control tedious or complex sequences. If a script is already running, then this button will end it.

See Automation.

Status information

Shows some status information about the system, mostly the connection state.

Stop lights and scripts

"Emergency Stop" button. This will immediately stop any light, pulse or script that is currently in progress.

Exit the program

If connected to a ***precisExcite*** main unit, will disconnect before terminating.

External Control Signals

precisExcite has several external TTL-compatible inputs that allow the light output to be controlled by external equipment. It also has a TTL-compatible output that indicates when the light is on, which can then be used to daisy-chain control to other external equipment.

11.1 Automation

precisExcite has an ability to run simple scripts that allow you to automate a sequence of events.

The ***precisExcite*** main unit uses an internal queuing method to sequence events. The queue can contain a mixture of lighting control commands, pause commands and simple operator-interface commands. These commands allow you to turn colours on or off, set the levels of colours, pause for a time to control an exposure or pause to interact with the machine when controlling groups of events.

precisExcite may be controlled via a USB data connection or a TCP/IP (Ethernet) data connection that allow control of colour, intensity, on and off times, pulse durations and the behaviour of the triggering input.

precisExcite has TTL-compatible input trigger and output synchronisation signals, via which it may be synchronised accurately with external events and systems.

11.1.1 Scripting

The ***precisExcite*** **PC Application** contains a simple scripting engine that interprets human-friendly commands into the briefer and less readable machine commands we send over the USB and TCP/IP connections.

These script commands are used by the DDE interface method.

Some example scripts are provided to give an idea how this works. Load them using the File::Load Script menu and try running them. They are simple text files (with a suffix .scr) that you can change with any plain-text editor like Notepad or Vim.

Within these scripts, the '#' character defines the start of a comment, which continues until the end of the line. Comments are completely ignored by the software.

Individual colours are controlled by their name followed by a percentage brightness level or the words On or Off. Case is not significant.

Pauses may be in milliseconds(ms), seconds(s), minutes(m) or hours(h), however short duration pauses of 100ms or less are of only modest accuracy, as time granularity in the software is presently around 10ms.

Operator interactions are controlled by the UserPause or Repeat commands, which take arguments of: a label (only in the case of the repeat), an interaction specifier and a message string delimited by the { and } characters.

Labels are defined by an alphanumeric string [_A-Za-z0-9] terminated by a colon. The first character of the string must not be a digit.

Loops cannot be nested.

The following interaction parameters are available for the interactive commands:

ok	displays a dialog box offering just an OK button.
okcancel	displays a dialog box offering OK and Cancel buttons. Ok contnues the script and Cancel ends it.
yesno	displays a dialog box offering Yes and No buttons. Yes contnues the script and No ends it.
always	continues, usually into a loop, without any message. No message text is required

There are some queue controls:

Queue Run	releases the queue from any stopped state.
Queue Stop	halts the queue, whilst commands are loaded to it. You may not need this.
Queue Wait	waits at this point in the script for the precisExcite main unit to complete any work in progress. The various userpause and repeat

The following script commands are available. These include those above in a briefer form. The commands and arguments have also some synonyms for convenience.

Immediately executed commands, not queued

Command	Args	Args	Dewscription
Queue	Run		Allows the queue to run and the commands in it to execute.
Queue	Clear		Clears/empties the queue of commands.

Queued Commands

Command	Args	Args	Description
Queue	Off		Stops the queue and stops the actions being taken from the queue output
	Stop		(synonym)
	Wait		Waits until the queue has caught up with the script before sending further commands.
ChanA ChanB ChanC	On/Off		Turns On/Off the Channel A light at the present intensity
ChanA ChanB ChanC	Pulse 100	Ms/s/hm/h	Pulses the Channel A light at the present intensity for the specified time in milliseconds, seconds, minutes or hours.
ChanA ChanB ChanC	100		Sets the Channel A intensity to [number] percent. Does not affect On/Off status
PulseArm	100	Ms/s/hm/h	Arm the trigger for a pulse of [time] in milliseconds, seconds, minutes or hours.
ScriptArm			Arm the trigger to continue the script.
Pause	100	Ms/s/hm/h	Pause here for [time] in milliseconds, seconds, minutes or hours.
UserPause	ok	{A message}	Wait for user response, then continue.
	yesno	{A message}	Wait for user response, then continue or abort.
	okcancel	{A message}	Wait for user response, then continue or abort.
Repeat	ok	{A message}	Repeats from preceding label on OK user response.
	yesno	{A message}	Repeats from preceding label on OK user response, or abort.
	okcancel	{A message}	Repeats from preceding label on OK user response, or abort.
	retrycancel	{A message}	Repeats from preceding label on OK user response, or abort.
	yesnocancel	{A message}	Repeats from preceding label on OK user response, or aborts.
	always		Repeats from preceding label unconditionally.

12 Installing the PC software

The software for ***precisExcite***'s PC applications and PC Interface are supplied on CD.

Depending upon the CD supplied, it may be self-booting or require you to double-click the installation program. Again depending on the CD this will have either a ***pE*** or a Windows setup icon.

Follow the instructions on the installer. Mostly you will simply want to press **Next** or **Finish**.

13 A Simple Guide TCP, IP Address and NetMask.

You will almost certainly have *some* understanding of networking, even if you don't realise it. In the following text, we try to explain the concepts in as computer-beginner terms as we can. Inevitably for some, that may appear patronising. Please forgive us, we don't mean it to be so.

TCP/IP is the main method by which The Internet communicates. TCP and IP are both protocols that underlie most of those other protocols like HTTP for the World Wide Web and SMTP & POP3 for email and so on. They comprises the main `wrapper' method for most other data on the Internet. If TCP/IP doesn't work, neither does anything much else. But the basics of TCP/IP are fairly simple, so getting it to work isn't so hard.

Computers connected to a TCP/IP network must have an address, much like your postal address. We see this in two forms, the "reader friendly" names like www.precisExcite.com and the rather less friendly numerical form like 192.168.0.252. The former is like your street address, the latter like a PO Box number or telephone number. This number is the "IP Address" and much like your telephone number, it has a portion that identifies your computer and a portion that is, in effect, an area code.

Another strange number, the NetMask, defines what part of the IP address is the area code and what part is your computer.

As with telephone numbers in most places (less so in the USA), big cities tend to have short area codes and many `local' digits, while small towns tend to have long area codes and fewer local digits.

Those `dotted quad' numbers 192.168.0.252 and 255.255.255.0 are decimal representations the of the binary numbers with which computers are most happy, but that are strange to most people. Binary uses only the digits 0 and 1, though computer people normally organise them into larger groups of 8, 16, 32 or 64 `bits'. Each of the numbers between the dots represents an "unsigned eight-bit number". Unsigned means never negative and such eight-bit numbers can have values only between zero and 255. That 255 means that all of the eight bits are 1s.

Now let's consider that NetMask again. Notice that all the left-hand numbers are 255, meaning all 1s, an all the right-hand numbers are zeros. I guess you've already realised that the 1s define which part of the IP address is the `area code' and the zeros which part is the local number. Usually NetMasks have each number either 255 or 0, but occasionally the final non-zero number may be different as the area-code doesn't *have* to have a multiple of eight bits.

So our example IP address, 192.168.0.252 has an "area code" of 192.168.0 and a local number of 252. We call that area code a *Subnet Address*.

Just a couple more things to go. Within each Subnet, the local numbers that comprise all zeros and all ones, so in our example 0 and 255, are reserved for *broadcast* messages. If we try to use those for a single computer, the other computers on our subnet will get confused. The broadcast messages are used to find out, for example, what computer 252 actually is. One computer broadcasts "Hey, who is 252, please" and our machine only should respond "It's Me!". Of course, if two machine both respond "It's Me!", again the computers get confused.

Please try not to confuse computers. They're not as clever as they think they are!

14 Glossary

COM	Windows COM(unications) port. A serial port on a PC.
COM	Common Object Model. A Windows legacy library interface method
COM-Interop	COM Interoperability method (see COM and .NET)
DHCP	Dynamic Host Configuration Protocol (A protocol for automating the configuration of computers that use TCP/IP)
DLL	Dynamic Linked Library. Microsoft-defined pre-compiled library module for use within application programs.
IP	Inter-network Protocol. The fundamental data transport protocol of the Internet upon which all other Internet protocols are based
LAM	LED Array Module. The high-density LED array blocks used within precisExcite
LAN	Local Area network. A typical on-site or on-campus computer network.
LED	Light Emitting Diode. The electronic light devices used in precisExcite and elsewhere.
Managed Code	Windows programming term for post .NET programming standards where libraries publicly declare and enforce their interface.
.NET	Windows library interface method.
TCP	Transport Control Protocol. The usual basic protocol for transferring data in the Internet and local area networks.
TCP/IP	The TCP protocol as supported by the IP protocol; typical for most Internet and LAN traffic.
Telnet	A simple text-based inter-machine terminal session over a TCP/IP connection.
TTL	A physical electronic interface standard where signals are defined having levels within the 0V to 5V range. The acronym is "Transistor-Transistor Logic".
Unmanaged Code	Windows programming term for pre .NET programming standards where libraries did not publicly declare and enforce their interface.
USB	Universal Serial Bus. A serial communications interface standard.