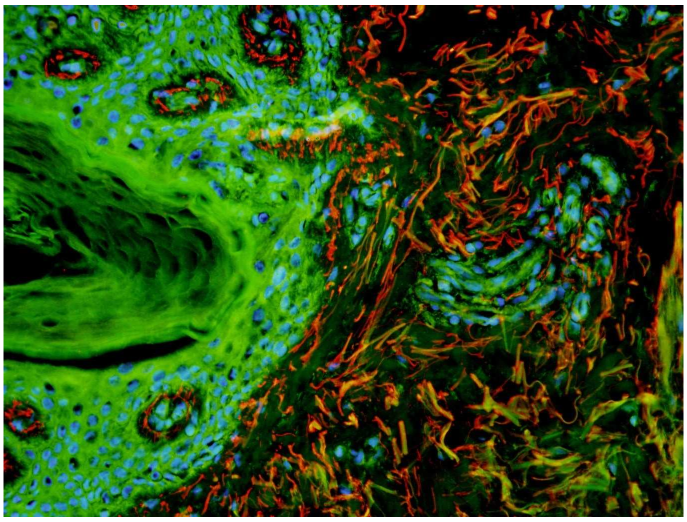
Life Sciences And Analytical



Essential Commands

***p*E-4000**, ***p*E-300** and ***p*E-2**

fluorescence excitation systems

2014/07/28

Author: gordon   Last Changed: 2014-07-29 12:09:57 +0000 (Tue, 29 Jul 2014)   svn Rev.: 242

Contents

1 Introduction **3**

2 Hardware **3**

2.1 TTL interface 3

2.2 Analogue interface 3

2.3 Data connection options USB or TCP/IP 3

3 Software **4**

3.1 Control Text Strings 4

3.2 Using the Character String Interface . . 4

3.3 ***p*E** Commands 5

4 Setting up the Interface **8**

4.1 Setting up the USB Interface 8

4.2 Setting Up USB on Windows Vista and 7 8

4.3 Setting Up USB on Windows 8 8

4.4 Setting Up USB on Linux 8

List of Tables

1 ***p*E** Control Commands 5

2 Example pE Responses 5

List of Figures

1 ***p*E** Example Command Operation . . . 6

2 ***p*E** Example Lambda Query 7

Release Notes

This is version initial release of this document.

1.0.0 Initial Release;

1 Introduction

COOLLED’s ***p*E** 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 from conventional mercury and metal halide sources which have to be switched on and controlled though mechanical shutters and irises. 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 ***p*E** system has been designed to provide the user with a choice of ways of controlling the LED’s functions:

- Manually via switches and an intensity control on the remote control “Pod“.
- From a PC using a choice of interfaces and software.

This documentation describes the ***p*E** commands that may be used over either ***p*E**’s USB/CDC virtual serial port (***p*E-4000**,***p*E-300** & ***p*E-2**), or over a TCP/IP connection (***p*E-2** only).

2 Hardware

On the reverse panel of the ***p*E** light source, there are a set of three connectors which provide a choice for remote interfacing and control.

- TTL interface(s) for hardware remote control
- Analogue port interfaces for remote hardware control (***p*E-4000** only)
- USB Virtual Serial Port(s)
- TCP/IP (Ethernet) data connection (***p*E-2** only)

2.1 TTL interface

This is the simplest method for remote control of the LED arrays. There is an individual TTL input for each physical light channel, 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 ***p*E** light source.

2.2 Analogue interface

The ***p*E-4000** light source offers four 0V to 10V analogue inputs, using which external equipment may control the intensity of the light output. It also offers four 0V to 10V analogue outputs that may be used to control external equipment.

2.3 Data connection options USB or TCP/IP

The ***p*E-2**, ***p*E-300** & ***p*E-4000** systems can be controlled via a USB connection, by which they look and behave like serial ports.

The ***p*E-2** also has a built-in TCP/IP connection like the Internet, where it takes a serial data stream rather like a simple telnet.

A TCP/IP to USB interface program is available for PCs if TCP/IP control is wanted for ***p*E-300** or ***p*E-4000**, however it does require that a PC is present (Windows or Linux/BSD, check with **CoolLED** for Mac.).

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

- The USB connection will usually be the choice as the ***p*E** is then connected directly to your PC.

3 Software

3.1 Control Text Strings

The base interface between a controlling computer and pE, is a simple set of text commands and responses usually sent via 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.

3.2 Using the Character String Interface

This is the most fundamental method by which ***p*E** 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 functions.

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

The command have reasonable mnemonics for easier learning.

The following commands are a subset of those available and will be sufficient for most purposes. An additional document is available detailing legacy, little-used and other commands. (***p*E-300** and ***p*E-4000** do not support the legacy Arm, Sequence, Queue or CSM commands).

3.3 *p*E Commands

In the following tables, the letter x defines the channel (A to H), n refers to N (On) or F (Off). s refers to S (Selected) or X (Not selected).

Command	Function	Description.
CSS?		Elicit a status map response for all channels. See details below.
CSS.	Level Map	Sets channel intensities *(Since R1.8.3)* See details below.
LAMBDAS	LAMBDAs enquiry	Elicits information about the available wavelengths.
LAMS	LAM enquiry	Elicits information about the wavelengths ready for use.
LOAD:	LOAD:470	Asks a ***p*E-4000** to load an LED. .. responds when loaded, e.g. LAM:B:470
ANxn	ANAN ANAF	changes the current wavelength on channel x to use or to NOT use the analogue control input.
CSn	CSN	Turns On or Off the **selected** channels.
C[+-]	CS+ CS-	Changes intensity of all channels
XVER	Version enquiry	Elicits various information about the software and hardware versions, etc..

XLIVE	XLIVE=YES
	XLIVE=NO
		Reports channel state regularly.
		Reports only as a response to channel commands.

Table 1: ***p*E** Control Commands

The CSS command and response have the form: CSS[xsn]*
e.g.: CSSA[S|X][N|F]nnnB[S|X][N|F]nnnC[S|X][N|F]nnnD[S|X][N|F]nnn

nnn is the intensity in integer percent (for pE-2 this must be exactly three digits).
channels may be in any order and any realistic number of channels.
Example: CSSAXF000
Example: CSSEXF000FSN050GSN075HSF100AXF000BSN050CSN075DSF100
Our response will always be alphabetic order, but may use more or fewer channels.
Example: CSSAXF000BSN050CSN075DSF100
Example: CSSAXF000ESN070
Example: CSSEXF000FSN050GSN075HSF100

(***p*E-4000** offers additional channels E to H for TTL+Analogue outputs), where On/Off controls the TTL and intensity controls the analogue output.

Response	Description.
XVER=XXXXX	The firmware version of the ***p*E** as a string.
XHEAD_VER=XXXXX	The firmware version of the ***p*E-2** LED head as a string.
XHW_POD=XXXXX	The hardware version of the ***p*E** Pod as a string.
XFW_POD=XXXXX	The firmware version of the ***p*E** Pod as a string.
XDATA_VER=XXXXX	The non-volatile data version as a string.
XHW_VER=XXXXX	The hardware version of the ***p*E** as a string.
XFW_BAK=XXXXX	The firmware version of the ***p*E-4000** backend processor(s) as a string.
LAMBDA:A0=XXXXX	Channel+LED A label, e.g., 405. Chan+LED are A0..A3, B0..B3, C0..C3, D0..D3
LAM:A:XXXXX	LAM/Channel A label, e.g., 400nm. Ditto LAM:B: LAM:C: etc.

Table 2: Example pE Responses

Note that labels are usually numeric but may sometimes be a string, e.g., WH1 or GYR.

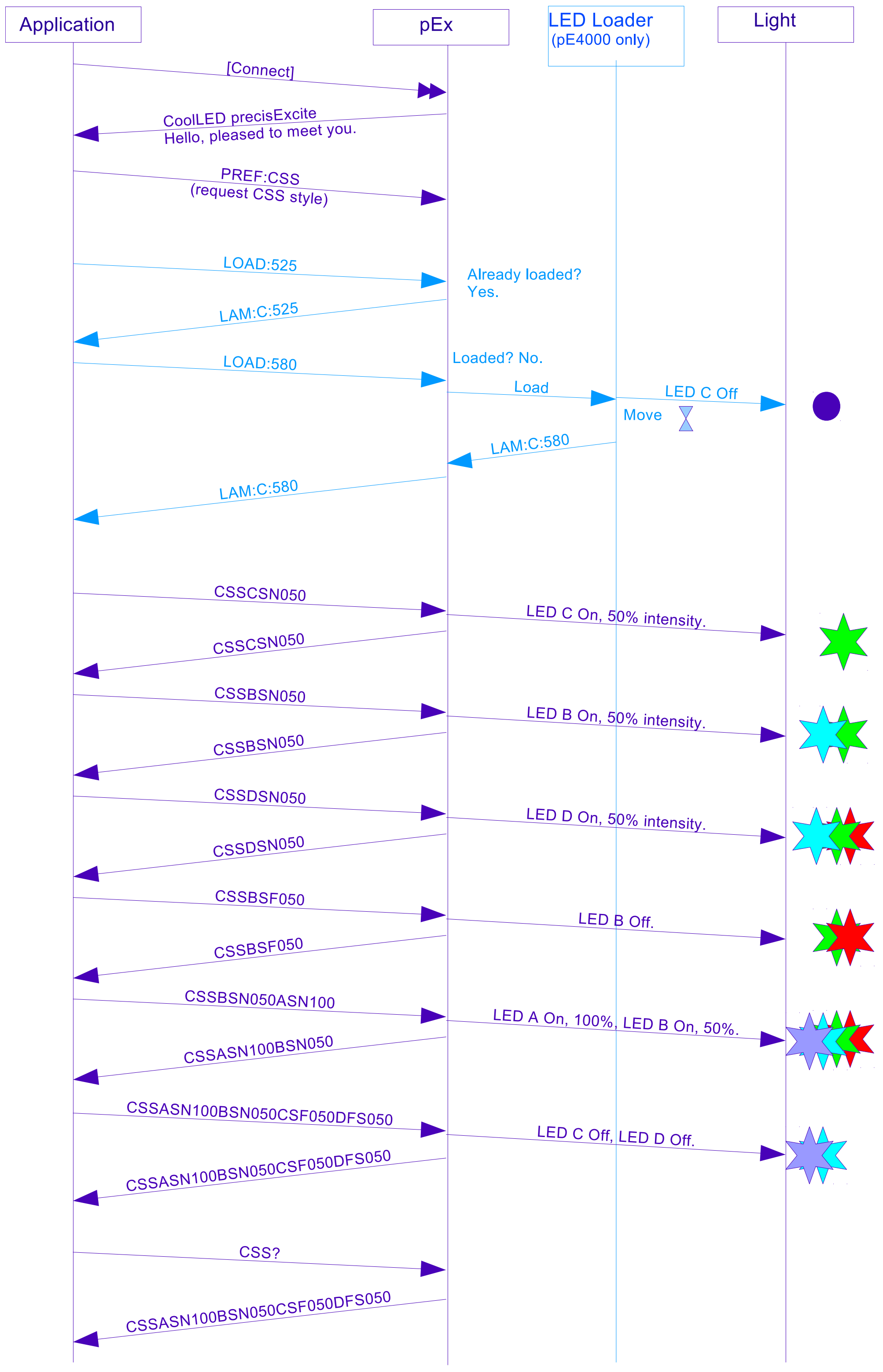


Figure 1: ***pE*** Example Command Operation

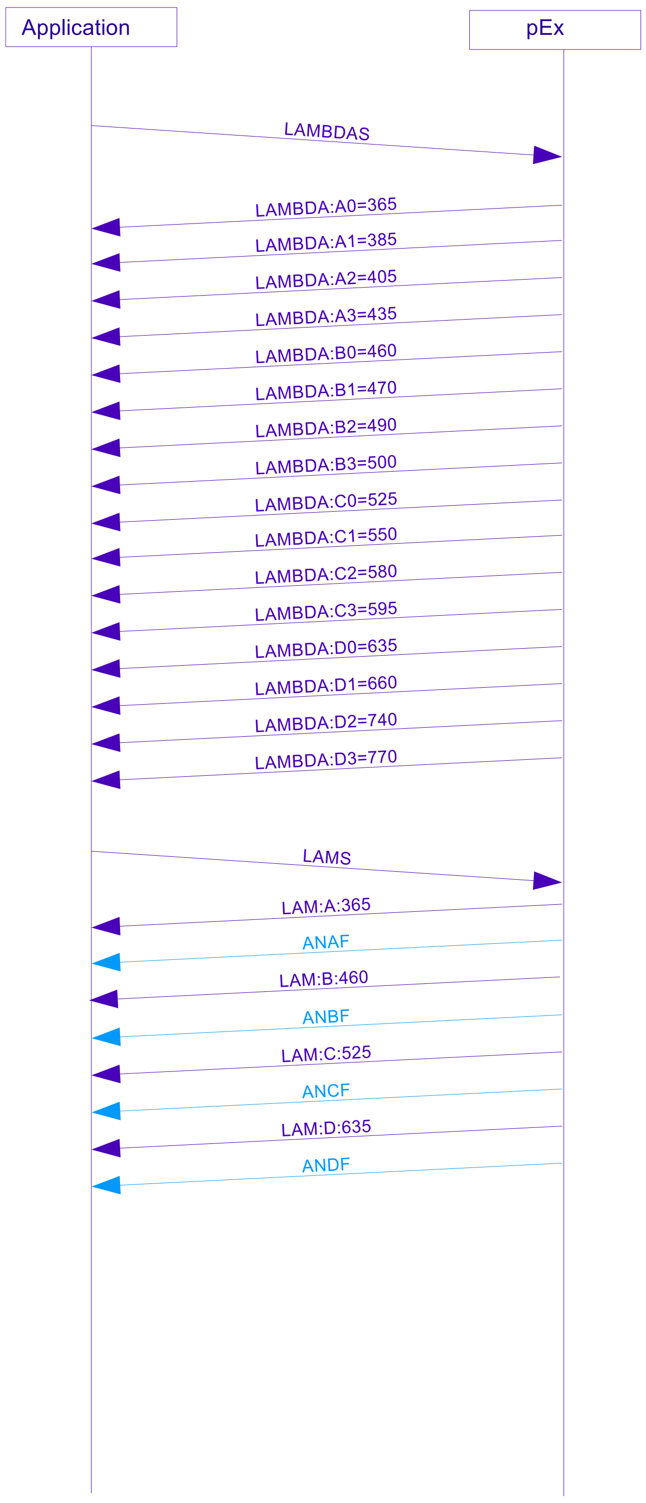


Figure 2: ***p*E** Example Lambda Query

4 Setting up the Interface

4.1 Setting up the USB Interface

The USB interface uses Microsoft's own built-in drivers for Virtual Serial Ports. This means that ***p*E** 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 ***p*E**, telling Windows that ***p*E** uses a Virtual COM Port. This is a file named `CoolLED-pE.inf`, which ids available from the CoolLRD website.

When you first plug ***p*E** 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. Tell Windows to look in the location where you have the .inf file.

Before you can start communicating with the ***p*E** via the USB interface, you must configure it to match the virtual COM port that Windows assigns.

From version 1.6 of our PC application, there is a **Finder** tab, which searches the network and COM ports for ***p*E**s . . . note that COM ports are a historic record. Double-click the entry to connect to the ***p*E**. Use Menu File->Save Config to permanently save the setting.

Alternatively.....

4.2 Setting Up USB on Windows Vista and 7

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

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

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

```
CoolLED USB Virtual Serial Port (Com4)
```

4.3 Setting Up USB on Windows 8

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

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

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

```
CoolLED USB Virtual Serial Port (Com4)
```

4.4 Setting Up USB on Linux

The ***p*E** port will be found in /dev/serial/by-id

```
/dev/serial/by-id/CoolLED....
```