X-Cite®
Fluorescence Illumination • In Control

Software Development Kit

Updated: November 26, 2014

For Products:

X-Cite 120PC series of Fluorescence Illumination Systems
X-Cite *exacte* Fluorescence Illumination System
X-Cite 120LED Illumination System
X-Cite 110LED Illumination System
X-Cite XR2100 Optical Power Meter

Table of Contents

1.0 Overview

This software development kit describes the information required to communicate with X-Cite devices. This document will describe the communication protocols for the various X-Cite devices. All X-Cite devices are either communicated with via the COM port or can be communicated with via a virtual COM port. As there are numerous ways to program a serial port, they will not all be described in this document. Microsoft has a number of examples in C++, Basic and .NET on their website. A basic outline of serial communications within Windows is found in the last section of this document.

1.1 Important Notes

It is important to note that when the virtual COM port is assigned a port number, this port number remains constant for the serial number of the device attached. For example, when an X-Cite *exacte* is connected, it may be assigned COM port 4. The COM port 4 will always be assigned to the X-Cite *exacte* with the same serial number. If another X-Cite *exacte* is connected with a different serial number, a new COM port number will be assigned.

2.0 X-Cite 120PC, X-Cite *exacte*, X-Cite 110LED and X-Cite 120LED

The command sets for these devices are all structured the same; however as the products have progressed, the command set has been expanded. The command set is based on the 120PC command set and has expanded from there. The 110LED and 120LED support the commands that exist in the eXacte, however since the products do not support CLF or calibration, the responses are simulated for compatibility purposes. The table in section 2.3 lists all commands in the command set, and which products they are supported by in functionality. The 110LED and 120LED will respond to all *exacte* commands however ones that are not marked supported, the responses are simulated.

2.1 Connection Settings

The X-Cite *exacte*, 110LED and 120LED are connected to the computer via a USB cable. The X-Cite 120PC is connected to a computer via a RS-232 cable. The drivers that are included with the *exacte*, 110LED and 120LED include a virtual COM port. The 110LED and 120LED COM Port can be set to any baud rate, 8 data bits, no parity and 1 stop bit. The 110LED and 120LED communicate at full USB speed of 12Mbps. The virtual or actual COM Port for the 120PC and *exacte*, as the case dictates, is to be set at 9,600 baud, 8 data bits, no parity and 1 stop bit. All X-Cite devices do not support hardware handshaking.

2.2 Command Set Format and Responses

This command list shows all commands, and which products the command is supported by. It is important to note that the commands after the identify command, only function once the identify command has been sent. It is important to note that even though the 110LED and 120LED supports all 120PC and *exacte* commands, not all commands are functional in the 110LED or 120LED, such as calibration and CLF.

All of the commands listed in this section result in an acknowledgement being sent from the X-Cite unit when the command has been received successfully. Otherwise, an error message is sent by the X-Cite unit.

An acknowledgement has a packet structure as defined in the following table.

Byte	Hexadecimal Value	ASCII Value
0	0x0D	'\r'

An acknowledgement with data has a packet structure as defined in the following table.

Byte	Hexadecimal Value	ASCII Value
0-X where X will be from 1 to less than 11	0x30 – 0x39, 0x2C, 0x2E, 0x2F	'0-9' , ',' , '/' , '.'
X + 1	0x0D	'\r'

An error message has a packet structure as defined in the following table.

Byte	Hexadecimal Value	ASCII Value
0	0x65	'e'
1	0x0D	'\r'

2.3 X-Cite *eXacte*, 120PC, 110LED & 120LED Command Set Listing

Note: Commands not listed that were in the X-Cite 120PC command set have been discontinued.

Description	Command	Response	Device for which command is supported			
			120PC	eXacte	110LED	120LED
Connect, enables PC control of the unit.	“tt\r”	“\r”	✓	✓	✓	✓
Get Intensity Level	“ii\r”	“x\r” Where if x is: 0 - intensity level of 0% 1 - intensity level of 12% 2 - intensity level of 25% 3 - intensity level of 50% 4 - intensity level of 100% For eXacte, if intensity is not at one of the above levels, an “e\r” is returned. For 110LED & 120LED, where if x is: 0 – intensity level less than 12% 1 – intensity level 12 to 24% 2 – intensity level 25 to 49% 3 – intensity level 50 to 99% 4 – intensity level of 100%	✓	✓	✓	✓
Get Lamp Hours	“hh\r”	“xxxx\r” where xxxx is the number of lamp hours. For the 110LED and 120LED, multiply this number by 10 for the hour count.	✓	✓	✓	✓
Get Software Version	“vv\r”	“xx\r” where xx is the version number of the software multiplied by 10.	✓	✓	✓	✓

Get Unit Status	“uu\r”	“xxx\r” where xxx is the status of the unit. The number returned is bitwise and is decoded as follows:
		bit 5 - Lock Bit: 1 = front panel locked, 0 = front panel unlocked;
		bit 4 - Lamp Ready Bit: 1 = lamp is ready, 0 = lamp is not ready;
		bit 3 - Home Bit: 1 = fault, 0 = pass;
		bit 2 - Shutter Bit: 1 = shutter is opened, 0 = shutter is closed;
		bit 1 - Lamp Bit: 1 = lamp is ON, 0 = lamp is OFF;
		bit 0 - Alarm Bit: 1 = alarm is ON, 0 = alarm is OFF.
			✓	✓	✓	✓

Description	Command	Response	Device for which command is supported			
			120PC	eXacte	110LED	120LED
Set Intensity Level	"ix\r" where x is the desired intensity level. If x is: 0 – Intensity is 0% 1 – Intensity is 12% 2 – Intensity is 25% 3 – Intensity is 50% 4 – Intensity is 100% Note: For the 110LED and 120LED, 0 – Intensity is 5% and 1% respectively.	"\r"	✓	✓	✓	✓
Close Shutter	"zz\r"	"\r"	✓	✓	✓	✓
Open Shutter	"mm\r"	"\r"	✓	✓	✓	✓
Turn Lamp Off	"ss\r"	"\r"	✓	✓	✓	✓
Turn Lamp On	"bb\r"	"\r"	✓	✓	✓	✓
Clear Alarm	"aa\r"	"\r"	✓	✓	✓	✓
Lock Front Panel	"ll\r"	"\r"	✓	✓	✓	✓
UnLock Front Panel	"nn\r"	"\r"	✓	✓	✓	✓
ALL COMMANDS BELOW THIS POINT ARE ENABLED BY IDENTIFY COMMAND						
Identify, enables the extended command set of the X-Cite *exacte*, 110LED & 120LED. The extended command set is the commands listed in this table below this point. This command must be issued before any of the commands listed beyond this point will function.	"jj\r"	"\r"		✓	✓	✓
Enable PC Shutter Control, enables PC control of the internal shutter via serial port commands.	"cc\r"	"\r"		✓	✓	✓
Disable PC Shutter Control, disables PC control of the internal	"yy\r"	"\r"		✓	✓	✓

Description	Command	Response	Device for which command is supported			
			120PC	eXacte	110LED	120LED
shutter via serial port commands.						
Disconnect PC, disconnects all control from the PC for the X-Cite exacte.	“xx\r”	“\r”		✓	✓	✓

Get Unit Status, same as above command, however adds bits 7 to 15 in the response.	“uu\r”	“xxx\r” where xxx is the status of the unit. The number returned is bitwise and is decoded as follows:
		Bit 15 – Iris Moving: 1 = Iris movement complete, 0 = Iris Moving.
		bit 14 – CLF Engaged
		bit 10 – Light guide inserted
		bit 8 – X-Cite exacte communication mode.
		Bit 7 – Power or Intensity Mode : 1 = Power mode, 0 = Intensity Mode
		bit 5 - Lock Bit: 1 = front panel locked, 0 = front panel unlocked;
		bit 4 - Lamp Ready Bit: 1 = lamp is ready, 0 = lamp is not ready;
		bit 3 - Home Bit: 1 = fault, 0 = pass;
		bit 2 - Shutter Bit: 1 = shutter is opened, 0 = shutter is closed;
		bit 1 - Lamp Bit: 1 = lamp is ON, 0 = lamp is OFF;
		bit 0 - Alarm Bit: 1 = alarm is ON, 0 = alarm is OFF.
		Note: The 110LED and 120LED will respond to CLF commands and Calibration commands, however since these features are not enabled on these products, the responses are simulated for compatibility purposes only.
				✓	✓	✓

Increment Iris Setting	“++\r”	“\r”		✓	✓	✓
Decrement Iris Setting	“--\r”	“\r”		✓	✓	✓
Change Power Mode, changes between intensity and power display if calibrated.	“qq\r”	“\r” (For 110LED & 120LED, always returns error as if unit not calibrated)		✓		
Get Calibration Time	“ee\r”	“xxx\r” where xxx is the number of calibration hours remaining in ASCII format. (110LED & 120LED, always returns 0)		✓		

Description	Command	Response	Device for which command is supported			
			120PC	eXacte	110LED	120LED
Clear Calibration	“ff\r”	“\r”		✓		
Turn CLF On	“kk\r”	“\r” (110LED & 120LED, changes status bit, however CLF not supported)		✓		
Turn CLF Off	“gg\r”	“\r” (110LED & 120LED, changes status bit, however CLF not supported)		✓		
Get Unit Serial Number	“GSN\r”	“xxxx\r” where xxxx is the serial number of the unit in ASCII format.		✓	✓	✓
Set Intensity Level	“dxxx\r” where xxx is the desired intensity percentage from 0 to 100. Note: For 110LED & 120LED, the intensity can only be set from minimum allowed intensity to 100%.	“\r”		✓	✓	✓
Get Intensity Level	“dd\r”	“xxx\r” where xxx is the intensity percentage in ASCII format.		✓	✓	✓
Change Output Power	pxxxxx\r”	“\r”		✓		
	where xxxxx is the desired output power setting, converted to an integer that can be interpreted by the unit: **xxxxx = Desired Power in watts X Power Factor X 100** The unit will display power in W, mW, or uW as appropriate. See “Get Power Factor” command description for obtaining power factor. Notes: 1. Any decimal places remaining in xxxxx after calculation will be truncated. 2. Adding a sixth character to xxxxx will return an error. 3. Requiring decimals or extra characters in xxxxx is an indication of going beyond the calibrated range of the system. For 110LED & 120LED, always returns an error as if unit not calibrated.			✓		
Get Output Power	“pp\r”	“xxxxx\r” where xxxxx is a number in ASCII format that can be used to calculate power output in watts: Power in watts = xxxxx / [Power Factor X 100] See Get Power Factor command description for obtaining power factor. For 110LED & 120LED, always returns error as if unit not calibrated.		✓		

Description	Command	Response	Device for which command is supported			
			120PC	eXacte	110LED	120LED
Get Power Factor	“??\r”	“xxxxx\r” where xxxxx is the power factor to be applied to “Output Power” values to convert them to watts, or to convert a desired setting into an integer that can be interpreted by the unit. The power factor will be returned as either 100, 1000, or 10,000. See “Get Output Power” and “Change Output Power” commands for proper use of this number. The 110LED & 120LED always returns 100. Power factor is redefined during each calibration process and should never be assumed to be a constant.		✓		
Get temperature	“gt\r”	“xxx,yyy\r” where xxx is the temperature of LED1 in °C and yyy is the temperature of LED2 in °C. In the 120LED this is the temperature of LED 1 and LED 2, in the 110LED this is the minimum then maximum LED temperatures.			✓	✓
Get speedDIAL software version	“pv\r”	“x.y.z\r” where x is the major software version, y is the minor software version and z is the bug fix level software version.			✓	✓
Get Unit Serial Number	“sn\r”	“xxxx\r” where xxxx is the serial number of the unit in ASCII format.			✓	✓
Get TTL State	“tl\r”	“x\r” where if x is 0, TTL is disabled, if x is 1, TTL is enabled in ASCII format.			✓	✓
Set TTL State	“tl=x\r” where if x is 0 TTL is disabled, if x is 1, TTL is enabled in ASCII format.	“\r”			✓	✓
Get TTL Timeout	“ti\r”	“xxx\r” where xxx is the current TTL timeout in minutes. 0 minutes is that the TTL will never timeout. The number format is ASCII.			✓	✓
Set TTL Timeout	“ti=xxx\r” where xxx is the desired TTL timeout from 0 to 999 minutes, 0 will set it to never. The number format is ASCII.	“\r”			✓	✓
Get Manufacturing Date	“md\r”	“mm/dd/yy\r” in month/day/year format as to when the unit was manufactured.			✓	
Get Minimum Allowed Intensity	“ni\r”	“xxx\r” where xxx is the minimum allowed intensity in %.			✓	

Description	Command	Response	Device for which command is supported			
			120PC	eXacte	110LED	120LED
Terminal Mode (See Section 3.0 for details)	"tm?\r"	Replies with a 1 if terminal mode is enabled, 0 otherwise.			✓	
	"tm=x\r" where x is the mode of the terminal communications port. If x is 1, it is active, if x is 0, is not active. A reboot is required to make the change active.	"\r" if the command is accepted.			✓	

3.0 110LED Terminal Mode

Terminal mode activates a second COM port that is to be used for debugging and development purposes supported only by the 110LED. Once activated the 110LED will request a different USB driver, which is located in the “dual” folder of the installation package. The 110LED will appear as 2 COM ports, the first one called “Communications” will use this command set as the single COM port mode, the second COM port called “Terminal” accepts no commands, however gives a running display in plain English as to the operating state of the 110LED. Using a program such as HyperTerminal, while running your software using the communication COM port, you can see via Hyperterminal all the status and informational changes in the 110LED to verify your programs functionality. Once completed, it is recommended that you turn off Terminal mode for the end user. The Dual mode driver is not certified by Microsoft and is only meant for debugging and development purposes.

4.0 X-Cite XR2100 Power Meter

The X-Cite XR2100 Optical Power Meter is connected to the computer via a USB cable. The drivers that are included with the XR2100 include a virtual COM port. This virtual COM Port is to be set at 19,200 baud, 8 data bits, no parity and 1 stop bit. The XR2100 has no hardware handshaking. This section has two table sets for the XR2100 Power Meter. The first is for the standard meter and the second section is the XR2100 OEM Power Meter. The XR2100 OEM Power Meter loses a number of the standard commands and has one additional command. The XR2100 OEM Power Meter only supports Hexadecimal communication mode, the decimal mode has been removed.

4.1 Message Format and Protocol

In decimal mode, all commands sent to the XR2100 must be terminated with a carriage return (“\r” in C or Chr$(13) in Basic). Line feed characters are ignored. Every response is also terminated with a line feed carriage return sequence (“\n\r” in C or Chr$(10) Chr$(13) in Basic).

In hex mode, all commands sent to the XR2100 must be terminated with an eight bit CRC (i.e. “FE” or “1A”), followed by carriage return (“\r” in C or Chr$(13) in Basic). Line feed characters are ignored. Every response is also terminated with an eight bit CRC and line feed carriage return sequence (“\n\r” in C or Chr$(10) Chr$(13) in Basic).

When the carriage return is received the XR2100 will respond with an error message if there is a problem. If there are no errors, the radiometer will respond as indicated below.

The maximum number of characters in any one command sent to the XR2100 is 18, including the carriage return and CRC.

4.2 Error Response

If there is a problem with the command the XR2100 will respond with the following message:

Err

For example:

Pwe?	Will result in	Err
Hlp=45	Will result in	Err

4.3 Numeric Formats

When sending a number to the XR2100 there are three possible formats that can be used.

For all numeric inputs, the number can be entered as a hex string. For example:

byte/char	Can be	xF3
word/int	Can be	x1A50
long/float/single	Can be	x003ACD6E

Example: Inp=xF1

Notice that every hex number begins with the lowercase x.
When this format is used, the XR2100 will respond with a hex numeric also.

The text can be upper or lower case.

Integer numbers such as:

byte/char	Can be	0 to 255
word/int	Can be	0 to 65535

Example: USN=63924

Real number such as:

float/single	Can be	[-]d.dddde[-]ddd 0 ≤ d ≤ 9, and [-] represents an optional sign.

Example: Set=3.9264e-003

4.4 CRC8 (8 bit CRC)

The following code written in C is used to calculate the 8 bit CRC. With slight modification, the same functionality can be written in Basic.
*Data is a pointer to the null terminated string that is to have a CRC calculation.

```
unsigned char CalcCRC8(unsigned char *Data)
{
  unsigned char LoopCntr;
  unsigned char CRC8;
  unsigned char A;

  CRC8 = 0;                                                  // reset CRC8
  while (*Data){                                             // loop until null char
    A = *Data++;                                             // get first data byte
    for (LoopCntr = 0; LoopCntr < 8; LoopCntr++, A >>= 1){   // 8 bit loop
      if ((A ^ CRC8) & 0x01){                                // test bit 0 of (Data XOR CRC8)
        CRC8 ^= 0x18;                                        // toggle bits 3 and 4 of CRC8
        CRC8 >>= 1;                                          // rotate right CRC8, 1 time
        CRC8 |= 0x80;                                        // set bit 7 of CRC8
      }else
        CRC8 >>= 1;                                          // rotate right CRC8, 1 time
    }
  }
  return CRC8;                                               // return CRC8
}
```

4.5 Response Format Commands

Description	Command To XR2100	Response
Set Hex Mode	Hex	Base=Hex $\underset{\text{CRC8}}{\boxed{\text{XX}}}$ X = Hex digit. Outlines and spaces provided for documentation purposes only. In hex mode all responses are terminated with an 8 bit CRC, and all numerical data is sent as a hex string (i.e. Set=x3FF5A188). This is the power up default mode.
Set Decimal Mode	Dec	Base=Dec In decimal mode all responses are **NOT** terminated with an 8 bit CRC, and all numerical data is sent as a string (i.e. Set=1.475e-2)

4.6 Read Command Descriptions & Listings

Description	Command To XR2100	Response
Get current power in W	Pwr?	Pwr**=**d.dddde[-]ddd 0 ≤ d ≤ 9, and [-] represents an optional sign. This output format only applies in Decimal Mode. Pwr=x $\underset{\text{Power}}{\boxed{\text{XXXXXXXX}}}$ $\underset{\text{CRC8}}{\boxed{\text{XX}}}$ X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode.
Get current analog input	Rad?	Rad= n 0 ≤ n ≤ 16777215 This output format only applies in Decimal Mode. Rad=x $\underset{\text{A/D Count}}{\boxed{\text{XXXXXX}}}$ $\underset{\text{CRC8}}{\boxed{\text{XX}}}$ X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode.
Get current temperature input	Tmp?	Tmp**=**n 0 ≤ n ≤ 16777215 This output format only applies in Decimal Mode. Tmp=x $\underset{\text{A/D Count}}{\boxed{\text{XXXXXX}}}$ $\underset{\text{CRC8}}{\boxed{\text{XX}}}$ X = Hex digit. Outlines and spaces provided for documentation purposes only.

		This output format only applies in Hex Mode.
Get current input channel	Inp?	Inp=n 0 ≤ n ≤ 255 This output format only applies in Decimal Mode. Inp=x $\boxed{XX}$ $\boxed{XX}$ Channel CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode.
Get current number of stored readings	Num?	Num=n 0 ≤ n ≤ 65535 This output format only applies in Decimal Mode. Num=x $\boxed{XXXX}$ $\boxed{XX}$ Count CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode.
Get current reading number	Rec?	Rec=n 0 ≤ n ≤ 65535 This output format only applies in Decimal Mode. Rec=x $\boxed{XXXX}$ $\boxed{XX}$ Count CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode.
Get data log record	Dat?	Dat=x $\boxed{XXXX}$ $\boxed{XXXX}$ $\boxed{XXXX}$ $\boxed{XXXXXXXX}$ $\boxed{XXXXXXXX}$ $\boxed{XX}$ $\boxed{XX}$ RTD RTC SN Power Irradiance Input Unused This output format only applies in Decimal Mode. Dat=x $\boxed{XXXX}$ $\boxed{XXXX}$ $\boxed{XXXX}$ $\boxed{XXXXXXXX}$ $\boxed{XXXXXXXX}$ $\boxed{XX}$ $\boxed{XX}$ $\boxed{XX}$ RTD RTC SN Power Irradiance Input Unused CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. word RTD = Real Time Date Bits 15-9 = Year – 2000 Bits 8-5 = Month Bits 4-0 = Day

		word RTC = Real Time Clock Bits 15-11 = Hours Bits 10-5 = Minutes Bits 4-0 = Seconds / 2 word SN = Serial Number (zero if eXacte not connected) float Power = Power in W float Irradiance = Irradiance in W/cm^2 byte Input = Input channel byte Unused = Reserved for future use
Move to next data log record	Next	Response is the same as Num? This command only moves the read pointer of the data log buffer to the next record if a Dat? command had been previously executed.
Get XR2100 serial number	S/N?	S/N=n 0 ≤ n ≤ 65535 This output format only applies in Decimal Mode. S/N=x $\boxed{XXXX}$ $\boxed{XX}$ Number CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode.
Get mode	Mod?	Mod=n 0 ≤ n ≤ 65535 This output format only applies in Decimal Mode. Mod=x $\boxed{XXXX}$ $\boxed{XX}$ Mode CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. Bit 0 = Calibration Required Bit 1 = Hex Mode Bit 2 = Default Setup Bit 3 = Exclude S/N from data logs Bit 4 = Do not log readings during X-CITE CAL Bit 5 = Reserved for future use Bit 6 = Reserved for future use Bit 7 = Reserved for future use Bit 8 = Relative/Absolute Lockout Bit 9 = Power/Irrad. Lockout Bit 10 = External Lockout Bit 11 = Set Lockout Bit 12 = Store Lockout Bit 13 = Irradiance Mode (read only when Bit 15 = 1) Bit 14 = Absolute Mode Bit 15 = External Input (read only)
Get real time date	RTD?	RTD=MM/dd/yy This output format only applies in Decimal Mode. MM is Month dd is Day yy is Year – 2000
Get real time clock	RTC?	RTC=hh:mm:ss This output format only applies in Decimal Mode.

		hh is Hour mm is Minute ss is Seconds
Get calibration due date	Cal?	Cal=MM/dd/yy This output format only applies in Decimal Mode. Cal=x $\boxed{\text{XX}}$ $\boxed{\text{XX}}$ $\boxed{\text{XX}}$ $\boxed{\text{XX}}$ MM dd yy CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. MM is Month dd is Day yy is Year – 2000
Get identity	Who?	Who=Stargate1.3a This output format only applies in Decimal Mode. Who=$\boxed{\text{XR2100 1.00}}$ $\boxed{\text{XX}}$ Identity CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. Note: Text starting with the first numeric character is the beginning of the firmware revision i.e. -- 1.00
Get currently pressed keys	Key?	Key=n 0 ≤ n ≤ 255 This output format only applies in Decimal Mode. Key=x $\boxed{\text{XX}}$ $\boxed{\text{XX}}$ Keys CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. Bit 0 = On Key Bit 1 = Store Key Bit 2 = External Key Bit 3 = Pwr/Irr Key Bit 4 = Set Key Bit 5 = Unused – Reserved for future use Bit 6 = Rel/Abs Key Bit 7 = Unused – Reserved for future use
Get current status	Sta?	Sta=n 0 ≤ n ≤ 255 This output format only applies in Decimal Mode. Sta=x $\boxed{\text{XX}}$ $\boxed{\text{XX}}$ Status CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. Bit 0 = No Lightguide Bit 1 = Low Battery

		Bit 2 = Unused, always active. Bit 3 = Unused, always off. Bit 4 = Set Failed Bit 5 = Unused – always off Bit 6 = Unused – Reserved for future use Bit 7 = Unused – Reserved for future use
Read 1-Wire bus	R1B	R1B=n,e,x $\underbrace{\boxed{XXXXXXXX}}_{ID}$,f,l,x $\underbrace{\boxed{XXXXXXXX}}_{ID}$,f,l … 0 ≤ n ≤ 255 : n = Number of 1-Wire ICs on bus 0 ≤ e ≤ 255 : e = Number of external inputs found. 0 ≤ f ≤ 1 : f = First input in chip. 0 ≤ l ≤ 44 : l = Last input in chip. This output format only applies in Decimal Mode. R1B=x $\underbrace{\boxed{XX}}_{n}$,x $\underbrace{\boxed{XX}}_{e}$,x $\underbrace{\boxed{XXXXXXXX}}_{ID}$,x $\underbrace{\boxed{XX}}_{f}$,x $\underbrace{\boxed{XX}}_{l}$,x $\underbrace{\boxed{XXXXXXXX}}_{ID}$,x $\underbrace{\boxed{XX}}_{f}$,x $\underbrace{\boxed{XX}}_{l}$ $\underbrace{\boxed{XX}}_{CRC8}$ X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. The multiplexer chip will have zeros for f and l. There will be one set of ID, f, and l for each chip on the bus.
Get LCD contrast	Con?	Con=n 0 ≤ n ≤ 15 This output format only applies in Decimal Mode. Con=x $\underbrace{\boxed{XX}}_{Contrast}$ $\underbrace{\boxed{XX}}_{CRC8}$ X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode.
Get Wavelength Selected	SWL?	SWL=n 320 ≤ n ≤ 750 where n is the wavelength selected
Get calibration date	SCD?	SCD=x $\underbrace{\boxed{XX}}_{MM}$ $\underbrace{\boxed{XX}}_{dd}$ $\underbrace{\boxed{XX}}_{yy}$ $\underbrace{\boxed{XX}}_{CRC8}$ X = Hex digit. Outlines and spaces provided for documentation purposes only. This output format only applies in Hex Mode. MM is Month dd is Day yy is Year – 2000
Get exacte serial number	USN?	USN=n where n is the serial number of the XR2100.
Get favorite wavelength	Fav?	Fav=xXXYYYY where XX is the wavelength number to be set and YYYY is the desired wavelength.

4.7 Write Command Descriptions & Listings

Note: All numeric data may be in entered in the formats described in section 4.3

Description	Command To XR2100	Response
Set current input	Inp=n	Inp=n

channel	0 ≤ n ≤ 1 0 = internal input 1 = external input	Response format is similar to read format. If n is out of range, Err is returned.
Set irradiance set point	Set=d.dddde[-]ddd 0 ≤ d ≤ 9, [-] represents an optional sign	Set=d.dddde[-]ddd Response format is similar to read format.
Set reference for relative mode	Ref=d.dddde[-]ddd 0 ≤ d ≤ 9, [-] represents an optional sign	Ref=d.dddde[-]ddd Response format is similar to read format.
Add Current Reading To Data log	DLog	Num=n Response is similar to Num? command **The XR2100 will acquire the X-Cite S/N (if enabled for data logs), via the IR Port before logging the readings.**
Set LCD contrast	Con=n 0 ≤ n ≤ 15	Con=n Response format is similar to read format. If n > 15, Err is returned.
Turn off XR2100	Off	Powering Down Power Down After executing this command, the XR2100 will not wake up via the RS232 port. Subsequent auto shutdowns will continue using this mode.
Put XR2100 to Sleep	Sleep	Sleep Mode Power Down **After executing this command the radiometer will wakeup if a 55 ms BREAK signal is sent via the RS232 port.** **Sending a BREAK signal is equivalent to pressing the ON keypad button.** **See the Microsoft Windows Communications API (Comm API) for information on how to send a BREAK signal.** **Subsequent auto shutdowns will continue using this mode.**
Calibrate eXacte	DoCal	Calibrating Passed or Failed **The XR2100 sends "Calibrating" before switching to the calibration Port and starting the calibration procedure. When finished, the XR2100 switches back to the USB port and sends "Passed" or "Failed".** **This command is equivalent to pressing the X-Cite CAL keypad button.**
Set Wavelegnth	SWL=n 320 ≤ n ≤ 750 where n is the set wavelength	Response format is similar to read format.
Set favorite wavelength	Fav=xXXYYYY where XX is the wavelength number to be set and YYYY is the desired wavelength.	Fav=xXXYYYY where XX is the wavelength number to be set and YYYY is the desired wavelength.

4.8 XR2100-OEM Read Command Descriptions & Listings

Description	Command To XR2100-OEM	Response

Get current power in W	Pwr?	Pwr=x $\underbrace{\boxed{XXXXXXXX}}_{\text{Power}}$ $\underbrace{\boxed{XX}}_{\text{CRC8}}$
		
		X = Hex digit.
		Outlines and spaces provided for documentation purposes only.

Get current analog input	Rad?	Rad=x $\underbrace{\boxed{XXXXXX}}_{\text{A/D Count}}$ $\underbrace{\boxed{XX}}_{\text{CRC8}}$
		
		X = Hex digit.
		Outlines and spaces provided for documentation purposes only.

Get current temperature input	Tmp?	Tmp=x $\underbrace{\boxed{XXXXXX}}_{\text{A/D Count}}$ $\underbrace{\boxed{XX}}_{\text{CRC8}}$
		
		X = Hex digit.
		Outlines and spaces provided for documentation purposes only.

Get current input channel	Inp?	Inp=x $\underbrace{\boxed{XX}}_{\text{Channel}}$ $\underbrace{\boxed{XX}}_{\text{CRC8}}$
		
		X = Hex digit.
		Outlines and spaces provided for documentation purposes only.

Get XR2100 serial number	S/N?	S/N=x $\underbrace{\boxed{XXXX}}_{\text{Number}}$ $\underbrace{\boxed{XX}}_{\text{CRC8}}$ X = Hex digit. Outlines and spaces provided for documentation purposes only.
Get mode	Mod?	Mod=x $\underbrace{\boxed{XXXX}}_{\text{Mode}}$ $\underbrace{\boxed{XX}}_{\text{CRC8}}$ X = Hex digit. Outlines and spaces provided for documentation purposes only. Bit 0 = Calibration Required Bit 1 = Hex Mode Bit 2 = Default Setup Bit 3 = Exclude S/N from data logs Bit 4 = Do not log readings during X-CITE CAL Bit 5 = Reserved for future use Bit 6 = Reserved for future use Bit 7 = Reserved for future use Bit 8 = Relative/Absolute Lockout Bit 9 = Power/Irrad. Lockout Bit 10 = External Lockout Bit 11 = Set Lockout Bit 12 = Store Lockout Bit 13 = Irradiance Mode (read only when Bit 15 = 1) Bit 14 = Absolute Mode Bit 15 = External Input (read only)
Get identity	Who?	Who=$\underbrace{\boxed{\text{XR2100 1.00}}}_{\text{Identity}}$ $\underbrace{\boxed{XX}}_{\text{CRC8}}$

		X = Hex digit. Outlines and spaces provided for documentation purposes only. Note: Text starting with the first numeric character is the beginning of the firmware revision i.e. -- 1.00
Get current status	Sta?	Sta=x $\boxed{XX}$ $\boxed{XX}$ Status CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. Bit 0 = No Lightguide Bit 1 = Low Battery Bit 2 = Unused, always active. Bit 3 = Unused, always off. Bit 4 = Set Failed Bit 5 = Unused – always off Bit 6 = Unused – Reserved for future use Bit 7 = Unused – Reserved for future use
Read 1-Wire bus	R1B	R1B=x $\boxed{XX}$,x $\boxed{XX}$,x $\boxed{XXXXXXXX}$,x $\boxed{XX}$,x $\boxed{XX}$,x $\boxed{XXXXXXXX}$,x $\boxed{XX}$,x $\boxed{XX}$ $\boxed{XX}$ n e ID f l ID f l CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. The multiplexer chip will have zeros for f and l. There will be one set of ID, f, and l for each chip on the bus.
Get Wavelength Selected	SWL?	SWL=n 320 ≤ n ≤ 750 where n is the wavelength selected
Get calibration date	SCD?	SCD=x $\boxed{XX}$ $\boxed{XX}$ $\boxed{XX}$ $\boxed{XX}$ MM dd yy CRC8 X = Hex digit. Outlines and spaces provided for documentation purposes only. MM is Month dd is Day yy is Year – 2000
Get exacte serial number	USN?	USN=n where n is the serial number of the XR2100.
Get favorite wavelength	Fav?	Fav=xXXYYYY where XX is the wavelength number to be set and YYYY is the desired wavelength.

4.9 XR2100-OEM Write Command Descriptions & Listings

Note: All numeric data may be in entered in the formats described in section 4.3

Description	Command To XR2100-OEM	Response
Set current input channel	Inp**=**n 0 ≤ n ≤ 1 0 = internal input 1 = external input	Inp**=**n Response format is similar to read format. If n is out of range, Err is returned.
Set irradiance set point	Set=d.dddde[-]ddd 0 ≤ d ≤ 9, [-] represents an optional sign	Set=d.dddde[-]ddd Response format is similar to read format.
Set reference for relative mode	Ref=d.dddde[-]ddd 0 ≤ d ≤ 9, [-] represents an optional sign	Ref=d.dddde[-]ddd Response format is similar to read format.

Set LCD contrast	Con=n 0 ≤ n ≤ 15	Con=n Response format is similar to read format. If n > 15, Err is returned.
Calibrate eXacte	DoCal	Calibrating Passed or Failed **The XR2100 sends "Calibrating" before switching to the calibration Port and starting the calibration procedure. When finished, the XR2100 switches back to the USB port and sends "Passed" or "Failed".** **This command is equivalent to pressing the X-Cite CAL keypad button.**
Set Wavelegnth	SWL=n 320 ≤ n ≤ 750 where n is the set wavelength	Response format is similar to read format.
Set favorite wavelength	Fav=xXXYYYY where XX is the wavelength number to be set and YYYY is the desired wavelength.	Fav=xXXYYYY where XX is the wavelength number to be set and YYYY is the desired wavelength.
Save settings to NVM	Save	Saved

5.0 Windows Serial Port Programming Overview

Programming serial ports in Windows is done through the file interface. The examples listed below are using Microsoft Visual C++. To open a serial port in windows it is the same as creating a file. The filename however is the system path plus the COM port name you wish to open. For example:

```
sprintf(szComPort,"\\\\.\\COM%d",cpPort);
```

Where cpPort is the port number you wish to open.

Then to open the communications port, open a file, for example:

```
if((ComPort=CreateFile(szComPort,GENERIC_READ|GENERIC_WRITE,0,NULL,OPEN_EXIST
ING, 0, 0)) == INVALID_HANDLE_VALUE) {
      return(FALSE);
} else {
      return(TRUE);
}
```

Then, set the communication timeouts, use the COMMTIMEOUTS data structure and call the SetCommTimeouts function. Once the timeouts are set, set the communication timing parameters using the BuildCommDCB function. For example:

```
sprintf(szBuf,"COM%d: baud=%d parity=%c data=%d stop=%d to=on xon=off dsr=off
octs=off dtr=off rts=off idsr=off",cpPort,nBaud,cParity,nDataBit,nStopBit);

if(!BuildCommDCB(szBuf, &dcb)) {
      return(FALSE);
}
```

Once the communication parameters have been set, you may want to set the communication state and event triggers using the SetCommState and SetCommMask functions. These functions are not required; however since X-Cite devices use CR at the end of all communication strings you may want to trigger a

function on this byte. One other useful function is PurgeComm, to clear out the transmit and receiving buffers.
To transmit data, it is the same as writing to a file, just call the WriteFile function, example:

```
WriteFile(ComPort,&szBuf,nLength,&dwCount,NULL);
```

To receive data, first check to see how much data has been received, you can use the ClearCommError function to obtain the current status and how many bytes have been received. For example:

```
ClearCommError(ComPort,&dwEvent,&ComStat);
```

The ComStat structure has a cbInQue member that contains the number of received bytes. You can then use the ReadFile function to obtain the data, for example:

```
ReadFile(ComPort,&c, 1, &dwCount, NULL);
```

Where the created pointer to c is a character buffer and the 1 specifies the length of c. You may want to do this for byte by byte processing or you can specify a longer buffer and read more of the data in one function call.

When you are done with communicating, it is best to close the serial port by closing the handle, for example:

```
CloseHandle(ComPort);
```

The examples provided are enough to perform basic windows serial port communication. There are numerous methods for using these functions and implementing a serial port class. It is best to use the Microsoft developer network resources for serial port programming and to obtain specific examples for the language you choose to use.

6.0 X-Cite Utilities DLL

A windows utilities library DLL is also included in this SDK for use with the 110LED and 120LED, the DLL will aid in the finding of the windows COM port, and resetting the Microsoft Windows CDC driver in case of a device reset. A known bug in the Microsoft Windows CDC driver does not properly reset itself if the USB device has a short interruption in communication, either due to a quick power cycle or an ESD shock. This defect has existed in the Microsoft Windows CDC driver since Windows XP, and an ETA for a fix has not been provided by Microsoft. The occurrence of this defect is very low; however this library does include a way to correct if required. To detect when the CDC driver has not properly reset, sending commands to the X-Cite unit and no response is received is the only way. When this occurs it is best to disconnect from the X-Cite device and use the reset function provided in the SDK. Then re-find the device and reconnect to it. An example application has also been provided demonstrating the usage of this DLL.
The DLL includes functions for:

a) Resetting the Microsoft Windows CDC driver
b) Finding the COM port of the 110LED or 120LED
c) Connecting to the 110LED or 120LED
d) Sending commands to the 110LED or 120LED
e) Receiving data from the 110LED or 120LED
f) Disconnecting from the 110LED or 120LED

The utilities DLL comes compiled in 32 bit and 64 bit versions, it is important to run the correct DLL on the correct version of windows. The 32 bit DLL will run on 32 and 64 bit OS versions, with the exception of resetting the COM port. The reset COM port function requires the correct DLL running on the correct OS.

The DLL functions available in this DLL are (with examples in Visual C++):

Function Name and Description	Function Call	Calling Parameters	Return Values
Find Device – Returns a true or false if a device of type specified is found attached to the system.	FindDevice	**unsigned char cType** Set cType to 1 to find 120LED devices, 2 to find 110LED devices.	TRUE if found, FALSE if not found.

	To load function:
	`lpfnDllFindDevice = (LPFNDLLFINDDEVICE)GetProcAddress(hXCLEDUtil,"FindDevice");`
	
	To call function:
	`if(lpfnDllFindDevice(nSelect)) {`		

Get COM Port – Returns the COM port for the device type specified	GetDeviceCOMPort	**unsigned char cType** Set cType to 1 for 120LED devices, 2 for 110LED devices. **char *pBuf** Pointer to a character buffer to return the COM port, example, if on COM7, will return COM7. This buffer should be at least 7 bytes long. If multiple devices are found, this will be a string of COM ports comma separated. **unsigned short int nBufLength** The length of the buffer given by *pBuf.	TRUE if the COM port was found and loaded into the buffer. FALSE if the COM port was either not found or the buffer size given was not of sufficient size.

	To load function:
	`lpfnDllGetDeviceCOMPort = (LPFNDLLGETDEVICECOMPORT)GetProcAddress(hXCLEDUtil, "GetDeviceCOMPort");`
	
	To call function:
	`if(lpfnDllGetDeviceCOMPort(TYPE_600SERIES, &szBuf[0], sizeof(szBuf))) {`		

Reset Device COM Port, resets the Microsoft Windows CDC driver by detaching and reattaching the USB cable (soft disconnect). Must use the correct 32 bit or 64	ResetDeviceCOMPort	**char *pBuf** A pointer to a buffer containing the COM port to reset. Must be in format of COMX where X is the port number.	TRUE if the port was reset, FALSE if not. If it returns a FALSE, most likely cause is 32 bit DLL run on 64 bit system or vice versa.

bit DLL to match OS for this to function.			

	To load function:
	`lpfnDllResetDeviceCOMPort = (LPFNDLLRESETDEVICECOMPORT)GetProcAddress(hXCLEDUtil, "ResetDeviceCOMPort");`
	
	To call function:
	`if(lpfnDllResetDeviceCOMPort(&szBuf[0])) {`		

Connect To Device, opens the serial port to the device type specified.	ConnectToDevice	**unsigned char cType** Set cType to 1 for 120LED devices, 2 for 110LED devices.	TRUE if the device was found and the port was opened successfully. FALSE if either device not found or port could not be opened. Function does not support overlapped I/O.

	To load function:
	`lpfnDllConnectToDevice = (LPFNDLLCONNECTTODEVICE)GetProcAddress(hXCLEDUtil,"ConnectToDevice");`
	
	To call function:
	`if(lpfnDllConnectToDevice(cType)) {`		

Device is Connected, returns the current connection state of the device type specified.	DeviceConnected	**unsigned char cType** Set cType to 1 for 120LED devices, 2 for 110LED devices.	TRUE if the device is connected and the serial port is open. FALSE if the serial port is not opened. A FALSE does not mean the device is not connected, just the serial port is not open.

	To load function:
	`lpfnDllDeviceConnected = (LPFNDLLDEVICECONNECTED)GetProcAddress(hXCLEDUtil,"DeviceConnected");`
	
	To call function:
	`if(lpfnDllDeviceConnected(cType)) {`		

Write to Device, writes data out to the device specified.	WriteToDevice	**unsigned char cType** Set cType to 1 for 120LED devices, 2 for 110LED devices. **char *pBuf** Pointer to a buffer containing the command to write out. Must end with a carriage return and then a NULL. Do not terminate with a line feed. **unsigned char cLengthToWrite**	TRUE of the command was written out. FALSE if the command write failed.

		The length of the command to write out.	

	To load function:
	`lpfnDllWriteToDevice =`
	`(LPFNDLLWRITETODEVICE)GetProcAddress(hXCLEDUtil,"WriteToDevice");`
	
	To call function:
	`if(lpfnDllWriteToDevice(cType,&szBuf[0],(unsigned char)strlen(szBuf))) {`		

Data Received From Device, Checks the serial port buffer for a complete command response, waits for the carriage return at the end of the response before returning a TRUE.	DataReceivedFromDevice	**Unsigned char cType** Set cType to 1 for 120LED devices, 2 for 110LED devices.	TRUE if a complete command response has been received. FALSE if a complete command response has not been received.

	To load function:
	`lpfnDllDataReceivedFromDevice =`
	`(LPFNDLLDATARECEIVEDFROMDEVICE)GetProcAddress(hXCLEDUtil,"DataReceivedFromDevice");`
	
	To call function:
	`if(lpfnDllDataReceivedFromDevice(cType)) {`		

Read from Device, loads into the buffer provided the response from the 110LED or 120LED.	ReadFromDevice	**unsigned char cType** Set cType to 1 for 120LED devices, 2 for 110LED devices. **char *pBuf** Pointer to a buffer in which to load the response into. **unsigned char cBufferLength** The length of the buffer receiving the data.	TRUE if loading the buffer was successful, FALSE if unable to load buffer, most common reason is response buffer was too small. Recommend response buffer be at least 50 bytes long.

	To load function:
	`lpfnDllReadFromDevice =`
	`(LPFNDLLREADFROMDEVICE)GetProcAddress(hXCLEDUtil,"ReadFromDevice");`
	
	To call function:
	`if(lpfnDllReadFromDevice(cType,&szBuf[0],sizeof(szBuf))) {`		

Disconnect from Device, closes the	DisConnectFromDevice	**unsigned char cType** Set cType to 1 for 120LED devices, 2 for 110LED devices.	TRUE if the serial port was closed, FALSE if the port could not be closed.

COM port for the device specified.			

	To load function:
	`lpfnDllDisConnectFromDevice = (LPFNDLLDISCONNECTFROMDEVICE)GetProcAddress(hXCLEDUtil,"DisConnectFromDevice");`
	
	To call function:
	`lpfnDllDisConnectFromDevice(cType);`		

6.1 Loading the X-Cite Utilities DLL

To load the DLL in Visual C++, use the loadlibrary function and included header file “XCLEDUtilLib.h” in your main application C file. This header should only be included once during the compile. All functions are the names declared with a preceeding descriptor of lpfnDll to indicate a long function pointer to a dynamically linked library. The following sample code can be used to load the library,

```
BOOL CXCLEDUtilitiesDLLTestApplicationDlg::LoadXCDLL(void)
{
   DWORD dwInfo;

   // Nullify the handle to the X-Cite Utilities DLL library,
   // hXCLEDUtil is of type HINSTANCE
   hXCLEDUtil = NULL;
   // Load the X-Cite Utilities DLL library.
   // Change the define to the path of the library file in your application
   hXCLEDUtil = LoadLibrary(XCLEDUTILDLLFILE);

   // Check to make sure the handle is valid.
   if(!hXCLEDUtil) {
      CString csBuf;
      // If the handle is not valid, get the last system error message that was generated.
      dwInfo = GetLastError();
      // If the error message says the module was not found, display error message.
      if(dwInfo == MODULE_NOT_FOUND) {
         // Adjust fixed text strings if not using unicode
         MessageBox(TEXT("DLL Module not Found!"),TEXT("XCLEDUtil.DLL Module load failure"),MB_OK);
         // Otherwise display a generic DLL not loaded error message
      } else {
         // Adjust fixed text strings if not using unicode
         csBuf.Format(L"DLL Module not Loaded! Error Code: %ld",dwInfo);
         MessageBox(csBuf,TEXT("XCLEDUtil.DLL Module load failure"),MB_OK);
      }
      // Return that the function failed to load the library
      return(FALSE);
   }

   // Load all the library functions
   lpfnDllFindDevice = (LPFNDLLFINDDEVICE)GetProcAddress(hXCLEDUtil,"FindDevice");
   lpfnDllResetDeviceCOMPort = (LPFNDLLRESETDEVICECOMPORT)GetProcAddress(hXCLEDUtil,
"ResetDeviceCOMPort");
   lpfnDllGetDeviceCOMPort = (LPFNDLLGETDEVICECOMPORT)GetProcAddress(hXCLEDUtil, "GetDeviceCOMPort");
   lpfnDllFindXT600DeviceType =
(LPFNDLLFINDXT600DEVICETYPE)GetProcAddress(hXCLEDUtil,"FindXT600DeviceType");
   lpfnDllConnectToDevice = (LPFNDLLCONNECTTODEVICE)GetProcAddress(hXCLEDUtil,"ConnectToDevice");
   lpfnDllDeviceConnected = (LPFNDLLDEVICECONNECTED)GetProcAddress(hXCLEDUtil,"DeviceConnected");
   lpfnDllWriteToDevice = (LPFNDLLWRITETODEVICE)GetProcAddress(hXCLEDUtil,"WriteToDevice");
   lpfnDllDataReceivedFromDevice =
(LPFNDLLDATARECEIVEDFROMDEVICE)GetProcAddress(hXCLEDUtil,"DataReceivedFromDevice");
```

```
   lpfnDllReadFromDevice = (LPFNDLLREADFROMDEVICE)GetProcAddress(hXCLEDUtil,"ReadFromDevice");
   lpfnDllDisConnectFromDevice =
(LPFNDLLDISCONNECTFROMDEVICE)GetProcAddress(hXCLEDUtil,"DisConnectFromDevice");

   // Return that the library load was successful
   return(TRUE);
}
```

Once the application is done with the library, ensure that the COM port is closed first on exit, and then use the FreeLibrary function to unload the DLL (If the file handle is valid). It is good practice to verify that the function pointers are valid (Not NULL) prior to calling them.

6.2 Notes on Usage of the X-Cite Utilities DLL

The X-Cite utilities DLL includes a header file “XCLEDUtilLib.h”, which defines the functions and the function names to be used. It also declares the directory path for the library location to be used at runtime. In addition to this it also defines the device types that can be used with the library, currently only TYPE_120LED and TYPE_600SERIES are valid for use.