Luciole Cold Light Source

Software Manual

09/03/2018

ARINAX

Advanced Research Instrumentation for Neutrons & X-rays

File reference

CLS-MAN-Luciole-SoftManual_01.docx

Product references

Item	Revision
Luciole	

Document Revision

Release N°	Date	Author	Modifications / comments
1	09/03/2018	DLA	Document creation, based on LuciolePythonModuleReleaseNote

Arinax document references

Document title	Revision

External documents references

Document title	Revision
CLS-MAN-Luciole-01-UserManual	6

Abbreviations

SUMMARY

1 Introduction

Luciole Cold Light Source can be remotely controlled via various interfaces. They will be described in more details in the following document.

Arinax provides a small demo application in java to directly control and configure the Luciole Cold Light Source.

For integration purposes, a python and C interfaces are also available, with associated demo source code given “as is”.

2 Default configuration

As described in the user manual, Luciole Cold Light Source can be remotely controlled via Ethernet or a serial RS232 connection. The configuration can be changed but here are the default parameters, they will be used in the examples provided in the following sections.

Serial RS232 :

- Baudrate : 115200
- Parity : None
- Data bits : 8
- Stop bit : 1

Ethernet :

- IP : 169.254.1.1
- Subnet : 255.255.0.0
- Default gateway : 169.254.1.254

3 Requirements

- Luciole Cold Light Source Box
- PC (Windows or Linux, 32 or 64 bits) equipped with one or both
    - Ethernet port for TCP communication
    - Serial RS232 port

4 Enum definitions

The C and Python interface are designed to be quite similar, they share the same enum definitions, such as return error codes. Here is a list and description of the enums that are used by either interface.

4.1 Error code

Error codes	Value	description
SUCCESS	0	No error
ERROR_INVALID_CMD	-20	Unsupported Read/Write access for UserVarId
ERROR_NO_RESP	-1000	No Response
ERROR_COM_NOT_OPEN	-3000	Not open
ERROR_COM_PROBLEM	-2000	Sending/reception problem
ERROR_TCP_NOT_SUPPORTED	-2001	TCP COM not supported for the calling method
ERROR_SER_NOT_SUPPORTED	-2002	SER COM not supported for the calling method
ERROR_COM_NOT_SUPPORTED	-2003	COM not supported for the calling method
ERROR_INVALID_PARAM	-4000	Parameter not valid. E.g. : Chx is not 1or 2

Table 1: Error codes

4.2 Control Mode

Control Mode	Value	description
MANUAL	1	Control with the physical buttons
ANALOG	2	Control with the analog lines
REMOTE	4	Remote control, via ethernet or RS232

Table 2 : Control Mode

4.3 ReadWrite access

Read Write access	Value	description
READ	0	Read a parameter
WRITE	1	Write a value to a parameter

Table 3 : Read Write access

4.4 TCP parameter Id

Read Write access	Value	description
TCP_DHCP	1	DHCP mode (1 for dhcp, 0 for a static configuration)
TCP_IP	2	IP value
TCP_SUBNETMASK	3	Subnet mask
TCP _GATEWAY	4	Default gateway
TCP_SPEED	5	Connection speed (Not used)
TCP_PORT	6	Port (Not used)

Table 4 : TCP parameter Id

4.5 Variable Id

Variable Id	Value	description	R/W access
IP_CONFIG	1	IP address	R/W
ETH_CONFIG	2	Ethernet configuration	R/W
TCP_PORT_NUM	3	TCP port number	R/W
CH1_OFFSET_MAN	4	Channel 1 manual offset(Should not be used)	R/W
CH2_OFFSET_MAN	5	Channel 2 manual offset(Should not be used)	R/W
CH1_OFFSET_REM	6	Channel 1 remote offset(Should not be used)	R/W
CH2_OFFSET_REM	7	Channel 2 remote offset(Should not be used)	R/W
CH1_RATIOMAX_MAN	8	Channel 1 manual maximum ratio(Should not be used)	R/W
CH2_RATIOMAX_MAN	9	Channel 2 manual maximum ratio(Should not be used)	R/W
CH1_RATIOMAX_REM	10	Channel 1 remote maximum ratio(Should not be used)	R/W
CH2_RATIOMAX_REM	11	Channel 2 remote maximum ratio(Should not be used)	R/W
CH1_DIG_COMMAND	101	Channel 1 digital command	R
CH2_DIG_COMMAND	102	Channel 2 digital command	R
LED1_TEMP	103	LED 1 temperature	R
LED21_TEMP	104	LED 2, first one, temperature	R
LED22_TEMP	105	LED 2, second one, temperature	R
OVERTEMP_ERR	106	Temperature overheat error	R
CH1_TRIG_MODE	107	Channel 1 trigger mode	R/W
CH2_TRIG_MODE	108	Channel 2 trigger mode	R/W
CH1_CTRL_MODE	109	Channel 1 control mode	R
CH2_CTRL_MODE	110	Channel 2 control mode	R
CH2_SET_NETWORK_MODE	120	Network mode	W

Table 5 : Variable Id

5 Python Interface

5.1 Introduction

“Luciole.py” is a python module that implements a LucioleClass and its methods to communicate with a Luciole Cold Light Source.

5.2 Requirements

- Python 2.7
- Luciole.py from ARINAX
- Package :pyserial-3.0.1
- Package :crcmod 1.7

5.3 Error codes

5.4 Luciole.py exported methods

User Methods	Description
cls_GetfrmVersion(self)	Returns the firmware version (string)
def cls_IsConnected(self)	Returns True if Luciole is connected to serial or TCP port
cls_OpenInterface(self,strInterfaceDesc)	Open serial or tcp interface giving a description string • serial format: "SER:PortName" ○ example for windows: “SER:COM1” ○ example for linux : " SER:/dev/ttyS0" • TCP format: "TCP:IPAdress:Portnumber" ○ "TCP:169.254.1.2:4528"
cls_CloseInterface(self)	Closes serial and tcp ports
cls_GetLightValue(self,Chx)	Get light value of Chx=1 or 2 return : • Light value>=0 if success • Error code<0 if error
cls_SetLightValue(self,Chx,LightVal)	Set light value (0=<int=<20000) of Chx (int=1 or2) return : • 0→SUCCESS • Error code<0 if error
cls_GetControlMode(self,Chx)	Get control mode of Chx (int=1 or2)

User Methods	Description
	return: • 1→MANUAL ,2→ ANALOG ,4→ REMOTE • Error code<0 if error
cls_ReleaseNetworkMode(self,Chx)	Release Network mode of Chx (int=1 or2) return : • 0→SUCCESS • Error code<0 if error
cls_GetState(self)	Get State (Overtemperature Error) of Luciole Box return: • 1→NO_OverTemp ,2→CH1_OverTemp ,CH2_OverTemp 3,CH1&CH2_OverTemp 4 • Error code<0 if error
cls_GetTrigger(self,Chx)	Get Trigger Mode of Chx (int = 1 or 2) return: • 1 if CHx in trigger Mode • 0 if CHx not in trigger Mode • Error code<0 if error
cls_SetTrigger(self,Chx,Flag)	Enable/Disable (Flag 1 or 0) Trigger Mode of Chx (int = 1 or 2) return: • 0→SUCCESS • Error code<0 if error

Table 6: LucioleClass methods

Admin Methods	Description
cls_OpenSerial(self,portname)	open serial port giving its portname • example for windows: “COM1” • example for linux : "/dev/ttyS0"
cls_CloseSerial(self)	closes serial port
cls_OpenTCPport(self,strIpAdress,port)	Ope tcp port giving :(string) Ip address(example: “169.254.1.2”)and (int) port number (example: 4528)
cls_CloseTCPport(self)	Closes tcp port
cls_ReadWriteValue(self,RWaccess,UserVarId,VarVal)	Read write(RWaccess=0,1) Value of UserVarId(except UserVarId=1) return : • 0→SUCCESS: Write command is well executed • Value of UserVarId: Read Command is well executed • Error Code<0 if error
cls_WriteTCPParams(self,DHCPMode,IpVal, SubNetMaskVal,GatewayVal,Speed,Port)	write TCP param ID : • Speed(1=<int=<6) see TCP Speed Mode ID definitions • Port (int),

Admin Methods	Description

	• DHCPMode(int=0 or1)
	• IpVal,SubNetMaskVal, GatewayVal (16 bit int)
	  ○ examples:
	  ▪ IpVal=0xA9FE0102 for 169.254.1.2
	  ▪ SubNetMaskVal=0xFFFF0102 for 255.255.0.0
	  ▪ GatewayVal=0x00000000 for 0.0.0.0
	return :
	• 0→SUCCESS: Write command is well executed
	• Error Code<0 if error

cls_ReadTCPParams(self, TCPParamId)	Read TCP param ID: 1 to 6 • cls_TCP_DHCP = 1 • cls_TCP_IP = 2 • cls_TCP_SUBNETMASK=3 • cls_CH1_GATEWAY=4 • cls_TCP_SPEED =5 • cls_TCP_PORT = 6 return : • Value of TCPParamId • Error Code<0 if error

Table 7: LucioleClass Admin methods

5.5 LucioleTest.py

"LucioleTest.py" is a test script that implements the most important "LucioleClass" methods. Users may run it to test these methods and deepen their understanding of the "Luciole.py" module.

6 C Interface

6.1 Introduction

Arinax provides a C interface for Windows and Linux on 32 or 64 bits platforms. Through it you can control and configure the Luciole Cold Light Source. Here is a short description of the exported functions and how to use them.

6.2 Exported methods

User Methods	Description
void cls_Close	Closes any open communication.
void cls_CloseCom(HDIAFRAME hDiaFrame)	Close the communication port which handle is hDiaFrame.

void cls_GetComConfig(char **pstrconfig)	Populates pstrconfig with the communication parameters. The format depends on the communication use.
	• serial format: "SER:PortName"
	  ○ example for windows: “SER:COM1”
	  ○ example for linux : " SER:/dev/ttyS0"
	• TCP format: "TCP:IPAdress:Portnumber"
	  ○ "TCP:169.254.1.2:4528"
	If the connection is not established, “NOT CONNECTED” is returned.

int cls_GetControlMode(BYTE Chx)	Get the control mode of a specific channel(Chx = 1 or 2).
	Return :
	• Control mode if > 0.
	  1 : Manual
	  2 : Analog
	  3 : Remote
	• If <=0 see error codes.

int cls_GetLightValue(BYTE Chx)	Get light value of of a specific channel (Chx = 1 or 2) Return : • Light value if >0. • If <=0 see error codes.
void cls_GetState(char **pstrError)	Get the current state of the Cold Light Source. If connected return “Ready” or an error message. Otherwise “”.
int cls_GetTrigger(BYTE Chx)	Get the trigger mode of a specific channel(Chx = 1 or 2). Return : • 1 : Chx in trigger mode. • 0 : Chx not in trigger mode.

User Methods	Description
	• < 0 : see error codes.
BOOL cls_IsConnected()	Return true if the com is open. (cls_OpenCom has been called successfully) False otherwise.
HDIAFRAME cls_OpenCom(char * strPortDesc)	Open a communication port. It can be a serial or a tcp connection depending on strPortDesc. See cls_GetComConfig for a format description. Return : • HDIAFRAME, handle to the open com, if > 0. • If <=0, see error codes.
int cls_ReadChVar(cls_VarID VarId)	Read a variable value. See cls_VarID definition for accepted values. Return : • The variable value if >= 0. • If < 0, see error codes.
long cls_ReadTCPParam(cls_TCPParamID TCPParamId)	See cls_ReadTCPParamEx. Wrapper around an open connection.
long cls_ReadTCPParamEx(HDIAFRAME hDiaFrame, cls_TCPParamID TCPParamId)	Read a TCP parameter on the open com described by hDiaFrame. See cls_TCPParamID definition for accepted values. Return : • TCP parameter if >=0. If < 0, see error codes.
int cls_ReadWriteChVarEx(HDIAFRAME hDiaFrame,cls_RWAccess ReadWrite,cls_VarID VarId, WORD VarVal)	Access a variable VarId on an open com hDiaFrame. See cls_VarID definition for accepted values for VarId. If ReadWrite is set to Read : Return : • The variable value if >= 0. • If < 0, see error codes. If ReadWrite is set to Write : Write VarVal to VarId. See error codes for the return value.
int cls_ReleaseNetworkMode(BYTE Chx)	Release network mode of a specific channel(Chx = 1 or 2). See error code for return value.
int cls_SetEthernetInterface(char * strIpAddr)	Opens an ethernet connection on the IP described in strIpAddr. See error code for return value.
int cls_SetLightValue(BYTE Chx , WORD LightVal)	Set light value of of a specific channel (Chx = 1 or 2). See error code for return value.

User Methods	Description
int cls_SetSerialInterface(const char* strCOM, const char* strBaudRate)	Opens a serial connection on the com port described in strCOM, with the baudrate strBaudRate. See error code for return value.
int cls_SetTrigger(BYTE Chx, int Flag)	Set the trigger mode of a specific channel(Chx = 1 or 2). 1 for trigger mode, 0 otherwise. See error code for return value.
int cls_WriteChVar(cls_VarID VarId, WORD VarVal)	Write VarVal to a variable. See cls_VarID definition for accepted values for VarId. See error code for return value.
long cls_WriteTCPParam(cls_SpeedMode Speed, int Port, BYTE DHCPMode, DWORD IpVal,DWORD SubNetMaskVal,DWORD GatewayVal)	See cls_WriteTCPParamEx. Wrapper around an open connection.
long cls_WriteTCPParamEx(HDIAFRAME hDiaFrame, BYTE DHCPMode , DWORD IpVal,DWORD SubNetMaskVal,DWORD GatewayVal)	Write TCP configuration for the Cold Light Source opened on hDiaFrame. Speed and Port are not used. DHCPMode : • 0 for static configuration. • 1 for dhcp. IpVal : static ip, if needed. (“169.254.1.1”) SubNetMaskVal : subnet mask for static configuration, if needed. (“255.255.0.0”) GatewayVal : gateway ip for static configuration if needed. (“169.254.1.254”) See error code for return value.

Table 8 : C exported functions

7 Demo application

Arinax provides a standalone java application that can connect to the Luciole Cold Light Source and control it.

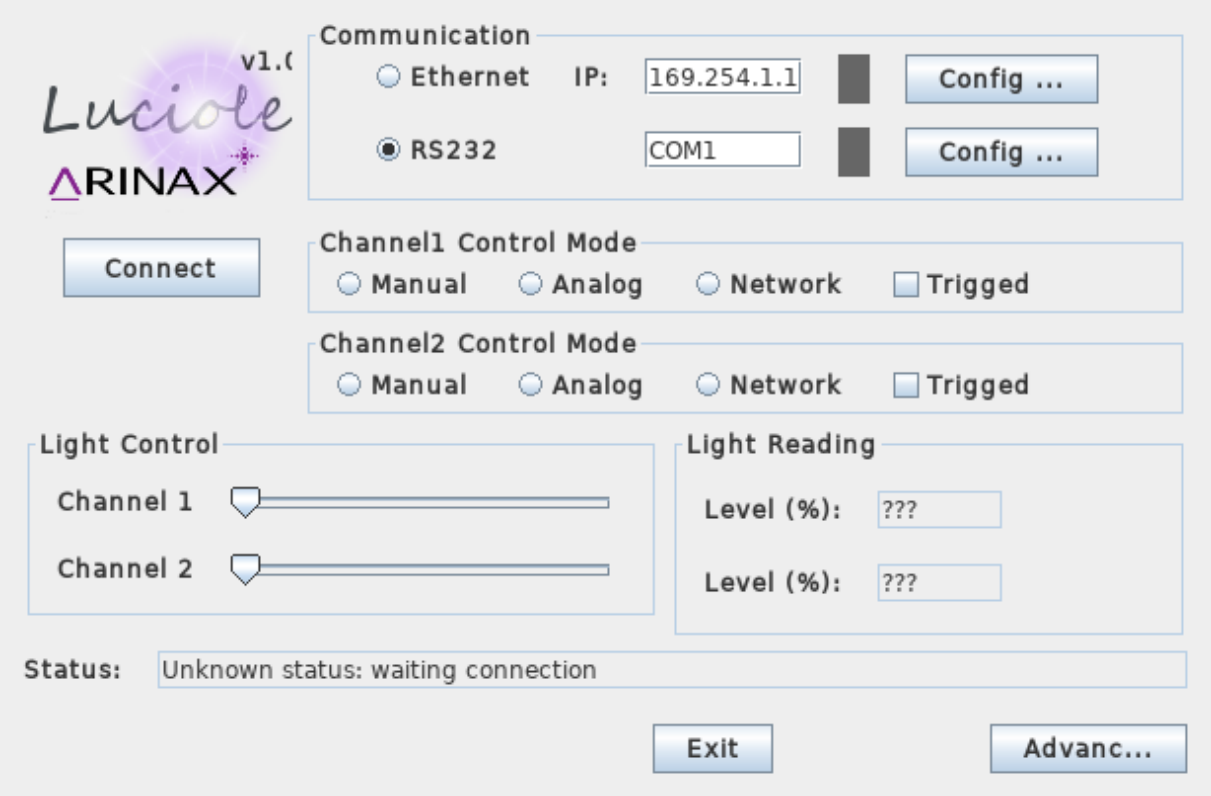

Image 1 : Main interface

This panel allows control and configuration of the Luciole Cold Light Source.

On the Communication panel, you can select to connect by ethernet or RS232. The “Config…” button allow to change the configuration of the Luciole Cold Light Source.

It can be used to change the Ip address of the Luciole Cold Light Source.

When the communication mode has been selected, clicking on the “Connect” button will start a connection to the Luciole Cold Light Source.

Once connected, you can control the lights levels using the sliders in Light Control. The value is read on the Light Reading panel.

The Control Modes panel allow to switch the control mode and to enable the trigger per channel. Note that if it is not on the network control mode, the light control won’t work.

The status panel display the status or any error on the Luciole Cold Light Source.