TANGO Device Server

CYBERSTARx1000 User's Guide

CYBERSTARx1000 Class

Revision: release_1_3_10 - Author: jean_coquet
Implemented in C++ - CVS repository: tango-ds

Introduction:

This device controls the pulse processing unit (PPU) of a fast scintillation detector (SD). This PPU is called CYBERSTAR. This product come from OXFORD DANFYSIK. The device controls the 3 parts of the PPU thanks to a RS232 link. 1- the high voltage module. (Voltage applied to the photomultplier of SD) 2- the preamplifier and shaping Amplifier module.(pulses shaping) 3- the Single Channel Annalyser.(energy selector).

Class Inheritance:

- Tango::Device_4Impl
  - CYBERSTARx1000

Properties:

Device Properties		
Property name	**Property type**	**Description**
SerialProxyName	Tango::DEV_STRING	name of the Serial Line device Proxy
MaxThreshold	Tango::DEV_FLOAT	Maximum voltage for upper threshold attribute

Device Properties Default Values:

Property Name	**Default Values**
SerialProxyName	No default value
MaxThreshold	No default value

There is no Class properties.

Attributes:

Scalar Attributes			
Attribute name	**Data Type**	**R/W Type**	**Expert**
voltage: sets the voltage to apply to the detector. It essentially determines the photomultiplier amplification.For large countrates, it is necessary to operate at lower voltages since the large number of electrons that would be created could damage the photomultiplier. High Voltage	DEV_DOUBLE	READ_WRITE	No
forcedRemoteMode: set forced remote control ON/OFF. In local mode the program stored in the miniature controller continuously polls the status of the local/remote button and checks if a force remote mode	DEV_BOOLEAN	READ_WRITE	No
amplifierGain: adjusts the amplitude of the shaped pulses available on the signal out of the BNC socket. Usually pulse heights are between 1 and 10 V.	DEV_DOUBLE	READ_WRITE	No
peakingTime: Elapsed time between the arrival of an X or gamma photon inside the scintillator and the peak value of the signal out pulse. It is the an Integration time. A small value of the peaking time allows high counting rate but poor energy linearity. A high value allows good energy linearity but the low counting rate	DEV_DOUBLE	READ_WRITE	No
scaLowerThreshold: defines the lower voltage threshold which generates TTL pulses. Commonly an SCA is operated by setting an upper and lower levels. it is adjustable between 50 mV to 10 V, precision +/- 10 mV.	DEV_DOUBLE	READ_WRITE	No
scaUpperThreshold: defines the upper voltage threshold which generates output TTL pulses. Commonly an SCA is operated by setting an upper and lower levels. it is adjustable between 50 mV to 10 V, precision +/- 10 mV.	DEV_DOUBLE	READ_WRITE	No
windowWidth: the window width is defined as: dV = high threshold - low threshold. where high threshold = window center position + dV/2 and high threshold = window center position - dV/2	DEV_DOUBLE	READ_WRITE	No
windowCenterPosition: defines the center of the SCA voltage window	DEV_DOUBLE	READ_WRITE	No

Commands:

More Details on commands....

Device Commands for Operator Level		
Command name	**Argument In**	**Argument Out**
Init	DEV_VOID	DEV_VOID
State	DEV_VOID	DEV_STATE
Status	DEV_VOID	CONST_DEV_STRING

1 - Init

- **Description:** This commands re-initialise a device keeping the same network connection.
After an Init command executed on a device, it is not necessary for client to re-connect to the device.
This command first calls the device *delete_device()* method and then execute its *init_device()* method.
For C++ device server, all the memory allocated in the *nit_device()* method must be freed in the *delete_device()* method.
The language device desctructor automatically calls the *delete_device()* method.

- **Argin:**
DEV_VOID : none.

- **Argout:**
DEV_VOID : none.

- **Command allowed for:**

2 - State

- **Description:** This command gets the device state (stored in its *device_state* data member) and returns it to the caller.

- **Argin:**
DEV_VOID : none.

- **Argout:**
DEV_STATE : State Code

- **Command allowed for:**

3 - Status

- **Description:** This command gets the device status (stored in its *device_status* data member) and returns it to the caller.

- **Argin:**
DEV_VOID : none.

- **Argout:**
CONST_DEV_STRING : Status description

- **Command allowed for:**