



TANGO Device Server

BT500
User's Guide

BT500 Class

Revision: release_1_0_9 - Author: vince_soleil
Implemented in C++ - CVS repository: tango-ds

Introduction:

handles generic BT500 cryogeenic temperature controller from Air Liquide (through serial line)

Class Inheritance:

- Tango::Device_4Impl
  - BT500

Properties:

Device Properties		
Property name	**Property type**	**Description**
SerialLineUrl	Tango::DEV_STRING	Tango name of the Serial Device for Communication
Period	Tango::DEV_ULONG	periodic exec period in ms
ReadTimeOut	Tango::DEV_LONG	timeout for reading
TimeInDeadband	Tango::DEV_LONG	time in deadband is used for State : when the temperature is inside deadband (abs (setpoint - temperature) < deadband) more than TimeInDeadband, the setpoint is considered as reached : state STANDBY otherwise RUNNING default : 60
ChannelsInUse	Tango::DEV_STRING	temperature channel used The device will read only the channels really used for example the BT500 comes with 2 temperature chann and on option with 2 more temperature channels for 2 channels : set property to 12 for 3 channels : 123 or 124 for 4 channels : 1234

Device Properties Default Values:

Property Name	**Default Values**
SerialLineUrl	No default value
Period	No default value
ReadTimeOut	No default value
TimeInDeadband	No default value
ChannelsInUse	1234

There is no Class properties.

Attributes:

Scalar Attributes			
Attribute name	**Data Type**	**R/W Type**	**Expert**
setpoint: setpoint and setpoint read back	DEV_DOUBLE	READ_WRITE	No
TC1: Temperature 1	DEV_DOUBLE	READ	No
TC2: Temperature 2	DEV_DOUBLE	READ	No
TC3: Temperature 3	DEV_DOUBLE	READ	No
TC4: Temperature 4	DEV_DOUBLE	READ	No
deadband: used for state management * setpoint is reached if temperature is stabilised in the range (setpoint +/-deadband) => state is STANDBY otherwise state is RUNNING	DEV_DOUBLE	READ_WRITE	No
pumpModeAutomatic: displays the control mode of the Pump [MANUAL|AUTOMATIC] true : automatic mode false : manual mode	DEV_BOOLEAN	READ_WRITE	Yes
pumpManualSpeed: Displays the speed of the pump (0 to 32, no unit) in manual mode, use it to set the speed of the pump	DEV_LONG	READ_WRITE	Yes
regulationChannel: displays the channel used as input for regulation sets the channel for regulation [1|2|3|4]	DEV_LONG	READ_WRITE	Yes
ovenMaxPower: oven maximum power the standard hardware does not support more than 100W	DEV_SHORT	READ_WRITE	Yes
ovenPowerPercent: the current value of the percentage of maximum power in the oven	DEV_SHORT	READ	Yes

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
ExecLowLevelCommand	DEV_STRING	DEV_STRING
ModePAP	DEV_VOID	DEV_VOID
ModeREG	DEV_VOID	DEV_VOID
RET	DEV_VOID	DEV_VOID
Local	DEV_VOID	DEV_VOID
Stop	DEV_VOID	DEV_VOID
Reset	DEV_VOID	DEV_VOID
ModeRGU	DEV_VOID	DEV_VOID

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:**

4 - ExecLowLevelCommand

- **Description:**

- **Argin:**
  DEV_STRING : command to be sent (exact syntax)

- **Argout:**
  DEV_STRING : response of the controller

- **Command allowed for:**

5 - ModePAP

- **Description:** switches the BT500 from MAIN Menu to PAP Menu (PAS A PAS, STEP BY STEP Menu) current Mode must be REMOTE, and Menu must be MAIN Menu

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

6 - ModeREG

- **Description:** switches the BT500 from PAP Menu to REG Menu (REGulation Menu) current Mode must be REMOTE, and Menu must be PAP Menu

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

7 - RET

- **Description:** If in REG Menu returns to PAP Menu if in PAP Menu returns to Main Menu Must be in REMOTE Mode

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

8 - Local

- **Description:** Returns to Local Command Mode (the touch screen on the front panel of the BT500) Must be in REMOTE Mode To switch to Remote Mode again the only way is to use the on the front panel of the BT500

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

9 - Stop

- **Description:** The device returns to STANDBY State

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

10 - Reset

- **Description:** resets after a fault

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

11 - ModeRGU

- **Description:** sends the BT500 directly to RGU mode (Soleil working page)

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**