TANGO
Device
Server

CellCouette
User’s Guide

CellCouette Class

Revision: release_1_1_7 - Author: vince_soleil
Implemented in C++ - CVS repository: tango-ds

Introduction:

this device is used to control a specific sample environnement called Cell Couette. This equipement is consisted of 3 parts: - the motor controller - the torque controller - temperature controller the multiplexage is ensured by a multipoint link RS-485. but sometime theses controllers can be in concurrency. To improve the communication, THE STATE COMMAND MUST BE POLLED BY THE USER (under JIVE application). The other attributes which read on the hardware are polled by default. Moreover, to ensure the communication , the elapsed time between 2 RS232 requests is at least 500 ms The device communicates with controllers by using the Serial Device (RS232 communication) the communication parameters of the associated proxy device server called Serial are: Baudrate = 19200; Charlength = 8; Newline = none; Serialline = COMx; stopbits = 0; Timeout = 2000 3 modes are available: CONTINUOUS: the rotor rotates continuously OSCILLATION: the rotor oscilates RAMP: the rotor rotates continuously by speed steady period

Class Inheritance:

- Tango::Device_4Impl
  - CellCouette

Properties:

Device Properties		
Property name	**Property type**	**Description**
SerialProxyName	Tango::DEV_STRING	name of the serial device proxy
TorqueCalibrationCoefficient	Tango::DEV_DOUBLE	TorqueCalibrationCoefficient

Device Properties Default Values:

Property Name	**Default Values**
SerialProxyName	No default value
TorqueCalibrationCoefficient	1

There is no Class properties.

States:

States	
Names	**Descriptions**
MOVING	the rotor is moving
STANDBY	the rotor is stopped
FAULT	serial communication problem

Attributes:

Scalar Attributes			
Attribute name	**Data Type**	**R/W Type**	**Expert**
mode: mode = 0 means CONTINUOUS mode = 1 means OSCILLATION	DEV_USHORT	WRITE	No
cellRadius: is used to calculate: - the distortion in oscillation mode - the stress torque - the shear rate in continuous mode	DEV_DOUBLE	WRITE	No
cellGap: used to calculate the distortion in oscillation mode	DEV_DOUBLE	WRITE	No
positionUnity: used to select the position unity (discret values) 0: STEP unity 1: DEGREE unity the conversion law is: 10000 step <=> 360ï¿½	DEV_USHORT	WRITE	No
position: current rotor position, unity is based on position unity	DEV_DOUBLE	READ_WRITE	No
speedUnity: used to select the speed unity (discret values) 0: STEP_PER_SECOND 1: HERTZ 2: ROTATION_PER_MINUTE 3: RADIAN_PER_SECOND 4: SHEAR_RATE	DEV_USHORT	WRITE	No
speedMax: prevent too high speed value. unity is based on the speed unity attribute	DEV_DOUBLE	READ_WRITE	No
isPositiveRotation: to select either positive or negative rotation	DEV_BOOLEAN	READ_WRITE	No
speed: write part: speed preset if resoution = 1 => 0.01Hz < range speed < 2 Hz) if resoution = 64 => 2 Hz < range speed <128 Hz) this value can’t exceed speedMax attribute value read part: estimated value by calculation (delta_position/delta_t)	DEV_DOUBLE	READ_WRITE	No
halfAngularAmplitude: defines the half angular amplitude in oscillation mode as: theta_min = - theta and theta_max = + theta the unity is based on position unity attribute	DEV_LONG	READ_WRITE	No
frequency: used in WT (motor time ramp) and WH(high hard motor speed) calculations	DEV_DOUBLE	WRITE	No
deformation: deformation = theta * (R+dR)/dR	DEV_DOUBLE	READ	No
timeRamp: - in continuous mode: - write part : WT preset - read part = write part - in oscillation mode: - read part : calculated value as WT = period x (0.5 - 1/Pi) x 1000 (in ms) - write part : no available to apply the preset value call the SetMotorParam Command	DEV_DOUBLE	READ_WRITE	No
speedMaxRamp: - in continuous mode: - write part : WH preset - read part = write part - in oscillation mode: - read part : calculated value as WH = (2*Pi*theta)/T - write part : no available	DEV_LONG	READ_WRITE	No
weightOffset: capteur offset in gramms	DEV_DOUBLE	WRITE	No
weight: F sensor measurement	DEV_DOUBLE	READ	No
cellHeight: cell Height	DEV_DOUBLE	WRITE	No
torque: torque measurement in micro Nm	DEV_DOUBLE	READ	No
stress: stress = M/(2*Pi*R^2*H)	DEV_DOUBLE	READ	No
temperature: sample temperature	DEV_DOUBLE	READ	No

motorResolution: set motor resolution: possible values are: 1, 2, 4, 8, 16, 32, 64. the resolution has an effect on the rotor speed	DEV_USHORT	WRITE	No
viscosity: viscosity = (1000*M/2*Pi*speed(Rad*sec-1)))* (1/ (H*R^2(R+DeltaR)/DeltaR + 4R^4/(8.465*16-H))	DEV_DOUBLE	READ	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
Start	DEV_VOID	DEV_VOID
Stop	DEV_VOID	DEV_VOID
PowerOFF	DEV_VOID	DEV_VOID
Reset	DEV_VOID	DEV_VOID
SetMotorParam	DEV_VOID	DEV_VOID
GetMotorParam	DEV_VOID	DEV_STRING
GetMotorState	DEV_VOID	DEV_STRING

Device Commands for Expert Level Only

Command name	Argument In	Argument Out
SendCommand	DEV_STRING	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:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

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:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

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:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

4 - Start

- **Description:** start the motor motion according to the operating mode selected (either CONTINUOUS or OSCILLATIONS) - CONTINOUS MODE: this is the write part of speed attribute which define the rotor speed once you push on start comand - OSCILLATIONS MODE: execute a sequence memorized in the controller (already programmed in the firmware)

- **Argin:**
  DEV_VOID : nothing

- **Argout:**
  DEV_VOID : nothing

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

5 - Stop

- **Description:** stop the motor motion

- **Argin:**
  DEV_VOID : nothing

- **Argout:**
  DEV_VOID : nothing

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

6 - PowerOFF

- **Description:** cut the motor power

- **Argin:**
  DEV_VOID : nothing

- **Argout:**
  DEV_VOID : nothing

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY

- Tango::FAULT

7 - Reset

- **Description:** the rotor returns to the default state (note that the encoder position is set to 0)

- **Argin:**
  DEV_VOID : nothing

- **Argout:**
  DEV_VOID : nothing

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

8 - SendCommand (for expert only)

- **Description:** send a specifc command to the specified controller. You must specify the controller adress in the string to send. adress list: 15: motor controller 03: temperature controller 02: torque controller ex: "15QX" where: 15 : the adress of the motor controller (2 bytes) QX: is the command (N bytes)

- **Argin:**
  DEV_STRING : command to send

- **Argout:**
  DEV_STRING : controller response

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

9 - SetMotorParam

- **Description:** Once you have written new values of speedMax, motorResolution, timeRamp attributes you must call setMotorParam(), thus new motor parameters are sent to the motorController

- **Argin:**
  DEV_VOID : nothing

- **Argout:**
  DEV_VOID : nothing

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

10 - GetMotorParam

- **Description:** get motor parameters

- **Argin:**
  DEV_VOID : nothing

- **Argout:**
  DEV_STRING : motor parameters

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT

11 - GetMotorState

- **Description:** return a message which inform you on the motor state

- **Argin:**
  DEV_VOID : nothing

- **Argout:**
  DEV_STRING : motor state

- **Command allowed for:**
  - Tango::MOVING
  - Tango::STANDBY
  - Tango::FAULT