TANGO
Device
Server

Modbus
User's Guide

Modbus Class

Revision: release_3_6_1 - Author: buteau
Implemented in C++

Introduction:

A Class to handle the modbus protocol over TCP/IP or Serial (RTU).

Class Inheritance:

- Tango::Device_3Impl
  - Modbus

Properties:

Device Properties		
Property name	**Property type**	**Description**
Protocol	Tango::DEV_STRING	RTU'' : Binary serial communication. ''TCP'' : Communication over ethernet.
Iphost	Tango::DEV_STRING	The host IP address used with the TCP protocol in the form aa.bb.cc.dd.
Serialline	Tango::DEV_STRING	The name of the serial line device used with RTU protocol. This can be any device name of a Serial Class object in the Tango system.
Address	Tango::DEV_SHORT	Node index used with the RTU or TCP protocol
CacheConfig	Array of string	Describe which data has to be cached. Each set of cached data is described by 3 parameters which are: 1 - Command to be used to read data (ReadHoldingRegisters, ReadInputStatus ReadInutRegisters or ReadMultipleCoilStatus) 2 - First address to be read 3 - Number of data to read
CacheSleep	Tango::DEV_LONG	Cache update thread main loop sleeping time (in ms)
SocketConnectionSleep	Tango::DEV_LONG	The necessary sleep time between closing a connection (Socket) and opening a new connection. To avoid hang-ups a non blocking socket is used to check the availability on the network. Afterwards the non blocking socket is closed and a blocking socket will be opened. The SocketConnectionSleep time specified the wait time in ms between these two connections.
TCPTimeout	Tango::DEV_LONG	Timeout used when the TCP protocol is used (in sec)

Device Properties Default Values:

Property Name	**Default Values**
Protocol	RTU
Iphost	No default value
Serialline	No default value
Address	1
CacheConfig	No default value
CacheSleep	1000
SocketConnectionSleep	200
TCPTimeout	1

There is no Class properties.

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
ForceSingleCoil	DEVVAR_SHORTARRAY	DEV_VOID
ReadCoilStatus	DEV_SHORT	DEV_SHORT
ReadInputStatus	DEVVAR_SHORTARRAY	DEVVAR_CHARARRAY
ReadHoldingRegisters	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY
ReadInputRegisters	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY
PresetSingleRegister	DEVVAR_SHORTARRAY	DEV_VOID
ReadExceptionStatus	DEV_VOID	DEV_SHORT
FetchCommEventCtr	DEV_VOID	DEVVAR_SHORTARRAY
ForceMultipleCoils	DEVVAR_SHORTARRAY	DEV_VOID
ReadMultipleCoilsStatus	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY
PresetMultipleRegisters	DEVVAR_SHORTARRAY	DEV_VOID
MaskWriteRegister	DEVVAR_SHORTARRAY	DEV_VOID
ReadWriteRegister	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY

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 - ForceSingleCoil

- **Description:** Write single coil (digital I/O) state.

- **Argin:**
  DEVVAR_SHORTARRAY : coil address, 0/1

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

5 - ReadCoilStatus

- **Description:** Read coil (digital I/O) status.
- **Argin:**
  DEV_SHORT : coil address
- **Argout:**
  DEV_SHORT : Coil status
- **Command allowed for:**

6 - ReadInputStatus

- **Description:** Read discrete input status. Return one boolean per array element.
- **Argin:**
  DEVVAR_SHORTARRAY : input address, no. of inputs
- **Argout:**
  DEVVAR_CHARARRAY : Input status.
- **Command allowed for:**

7 - ReadHoldingRegisters

- **Description:** Read multiple 16bits registers.
- **Argin:**
  DEVVAR_SHORTARRAY : register address, no. of registers
- **Argout:**
  DEVVAR_SHORTARRAY : Holding 16bits register.
- **Command allowed for:**

8 - ReadInputRegisters

- **Description:** Read Multiple 16bits input registers.
- **Argin:**
  DEVVAR_SHORTARRAY : register address, no. of registers
- **Argout:**
  DEVVAR_SHORTARRAY : Input 16bits registers

- **Command allowed for:**

9 - PresetSingleRegister

- **Description:** Write single 16bits register.

- **Argin:**
  DEVVAR_SHORTARRAY : Register address, register value.

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

10 - ReadExceptionStatus

- **Description:** Read exception status (usually a predefined range of 8 bits

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_SHORT : exception status

- **Command allowed for:**

11 - FetchCommEventCtr

- **Description:** Fetch communications event counter.

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEVVAR_SHORTARRAY : status, event count

- **Command allowed for:**

12 - ForceMultipleCoils

- **Description:** Write multiple coils (digital I/O) state. argin[0] = coil_address argin[1] = number of coils argin[2] = 1st coil state argin[3] = 2nd coil state ...

- **Argin:**
  DEVVAR_SHORTARRAY : coil address, nb of coils, coil states

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

13 - ReadMultipleCoilsStatus

- **Description:** Read multiple coil (digital I/O) status. argin[0] = register address argin[1] = number of registers

- **Argin:**
  DEVVAR_SHORTARRAY : coil address, nb of coils

- **Argout:**
  DEVVAR_SHORTARRAY : Status of coils

- **Command allowed for:**

14 - PresetMultipleRegisters

- **Description:** Write multiple 16bits registers. argin[0] = register address argin[1] = number of registers argin[2] = 1st register argin[3] = 2nd register ...

- **Argin:**
  DEVVAR_SHORTARRAY : register address, nb of registers, register data

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

15 - MaskWriteRegister

- **Description:** Mask write a 16bits register.

- **Argin:**
  DEVVAR_SHORTARRAY : register address, AND mask, OR mask

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

16 - ReadWriteRegister

- **Description:** Read and Write multiple 16bits registers. argin[0] = read address argin[1] = nb of registers to read argin[2] = write address, argin[3] = nb of registers to write, argin[4] = 1st register value to write argin[5] = 2nd register value to write ...

- **Argin:**
  DEVVAR_SHORTARRAY : read address, no. to read, write address, nb.of write, write data

- **Argout:**
  DEVVAR_SHORTARRAY : read registers

- **Command allowed for:**

ESRF - Software Engineering Group

TANGO
Device
Server

Modbus
User's Guide

Modbus Class

Revision: release_3_6_1 - Author: buteau
Implemented in C++

Introduction:

A Class to handle the modbus protocol over TCP/IP or Serial (RTU).

Class Inheritance:

- Tango::Device_3Impl
  - Modbus

Properties:

Device Properties		
Property name	**Property type**	**Description**
Protocol	Tango::DEV_STRING	RTU'' : Binary serial communication. ''TCP'' : Communication over ethernet.
Iphost	Tango::DEV_STRING	The host IP address used with the TCP protocol in the form aa.bb.cc.dd.
Serialline	Tango::DEV_STRING	The name of the serial line device used with RTU protocol. This can be any device name of a Serial Class object in the Tango system.
Address	Tango::DEV_SHORT	Node index used with the RTU or TCP protocol
CacheConfig	Array of string	Describe which data has to be cached. Each set of cached data is described by 3 parameters which are: 1 - Command to be used to read data (ReadHoldingRegisters, ReadInputStatus ReadInutRegisters or ReadMultipleCoilStatus) 2 - First address to be read 3 - Number of data to read
CacheSleep	Tango::DEV_LONG	Cache update thread main loop sleeping time (in ms)
SocketConnectionSleep	Tango::DEV_LONG	The necessary sleep time between closing a connection (Socket) and opening a new connection. To avoid hang-ups a non blocking socket is used to check the availability on the network. Afterwards the non blocking socket is closed and a blocking socket will be opened. The SocketConnectionSleep time specified the wait time in ms between these two connections.
TCPTimeout	Tango::DEV_LONG	Timeout used when the TCP protocol is used (in sec)

Device Properties Default Values:

Property Name	**Default Values**
Protocol	RTU
Iphost	No default value
Serialline	No default value
Address	1
CacheConfig	No default value
CacheSleep	1000
SocketConnectionSleep	200
TCPTimeout	1

There is no Class properties.

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
ForceSingleCoil	DEVVAR_SHORTARRAY	DEV_VOID
ReadCoilStatus	DEV_SHORT	DEV_SHORT
ReadInputStatus	DEVVAR_SHORTARRAY	DEVVAR_CHARARRAY
ReadHoldingRegisters	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY
ReadInputRegisters	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY
PresetSingleRegister	DEVVAR_SHORTARRAY	DEV_VOID
ReadExceptionStatus	DEV_VOID	DEV_SHORT
FetchCommEventCtr	DEV_VOID	DEVVAR_SHORTARRAY
ForceMultipleCoils	DEVVAR_SHORTARRAY	DEV_VOID
ReadMultipleCoilsStatus	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY
PresetMultipleRegisters	DEVVAR_SHORTARRAY	DEV_VOID
MaskWriteRegister	DEVVAR_SHORTARRAY	DEV_VOID
ReadWriteRegister	DEVVAR_SHORTARRAY	DEVVAR_SHORTARRAY

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 - ForceSingleCoil

- **Description:** Write single coil (digital I/O) state.

- **Argin:**
  DEVVAR_SHORTARRAY : coil address, 0/1

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

5 - ReadCoilStatus

- **Description:** Read coil (digital I/O) status.
- **Argin:**
  DEV_SHORT : coil address
- **Argout:**
  DEV_SHORT : Coil status
- **Command allowed for:**

6 - ReadInputStatus

- **Description:** Read discrete input status. Return one boolean per array element.
- **Argin:**
  DEVVAR_SHORTARRAY : input address, no. of inputs
- **Argout:**
  DEVVAR_CHARARRAY : Input status.
- **Command allowed for:**

7 - ReadHoldingRegisters

- **Description:** Read multiple 16bits registers.
- **Argin:**
  DEVVAR_SHORTARRAY : register address, no. of registers
- **Argout:**
  DEVVAR_SHORTARRAY : Holding 16bits register.
- **Command allowed for:**

8 - ReadInputRegisters

- **Description:** Read Multiple 16bits input registers.
- **Argin:**
  DEVVAR_SHORTARRAY : register address, no. of registers
- **Argout:**
  DEVVAR_SHORTARRAY : Input 16bits registers

- **Command allowed for:**

9 - PresetSingleRegister

- **Description:** Write single 16bits register.

- **Argin:**
  DEVVAR_SHORTARRAY : Register address, register value.

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

10 - ReadExceptionStatus

- **Description:** Read exception status (usually a predefined range of 8 bits

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_SHORT : exception status

- **Command allowed for:**

11 - FetchCommEventCtr

- **Description:** Fetch communications event counter.

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEVVAR_SHORTARRAY : status, event count

- **Command allowed for:**

12 - ForceMultipleCoils

- **Description:** Write multiple coils (digital I/O) state. argin[0] = coil_address argin[1] = number of coils argin[2] = 1st coil state argin[3] = 2nd coil state ...

- **Argin:**
  DEVVAR_SHORTARRAY : coil address, nb of coils, coil states

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

13 - ReadMultipleCoilsStatus

- **Description:** Read multiple coil (digital I/O) status. argin[0] = register address argin[1] = number of registers

- **Argin:**
  DEVVAR_SHORTARRAY : coil address, nb of coils

- **Argout:**
  DEVVAR_SHORTARRAY : Status of coils

- **Command allowed for:**

14 - PresetMultipleRegisters

- **Description:** Write multiple 16bits registers. argin[0] = register address argin[1] = number of registers argin[2] = 1st register argin[3] = 2nd register ...

- **Argin:**
  DEVVAR_SHORTARRAY : register address, nb of registers, register data

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

15 - MaskWriteRegister

- **Description:** Mask write a 16bits register.

- **Argin:**
  DEVVAR_SHORTARRAY : register address, AND mask, OR mask

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

16 - ReadWriteRegister

- **Description:** Read and Write multiple 16bits registers. argin[0] = read address argin[1] = nb of registers to read argin[2] = write address, argin[3] = nb of registers to write, argin[4] = 1st register value to write argin[5] = 2nd register value to write ...

- **Argin:**
  DEVVAR_SHORTARRAY : read address, no. to read, write address, nb.of write, write data

- **Argout:**
  DEVVAR_SHORTARRAY : read registers

- **Command allowed for:**

ESRF - Software Engineering Group

Frame Alert

This document is designed to be viewed using the frames feature. If you see this message, you are using a non-frame-capable web client.
Link to Non-frame version.

TANGO
Device
Server

Modbus
Device Commands Description
Modbus Class

Revision: release_3_6_1 - Author: buteau

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 - ForceSingleCoil

- **Description:** Write single coil (digital I/O) state.

- **Argin:**
  DEVVAR_SHORTARRAY : coil address, 0/1

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

5 - ReadCoilStatus

- **Description:** Read coil (digital I/O) status.

- **Argin:**
  DEV_SHORT : coil address

- **Argout:**
  DEV_SHORT : Coil status

- **Command allowed for:**

6 - ReadInputStatus

- **Description:** Read discrete input status. Return one boolean per array element.

- **Argin:**
  DEVVAR_SHORTARRAY : input address, no. of inputs

- **Argout:**

DEVVAR_CHARARRAY : Input status.

- **Command allowed for:**

7 - ReadHoldingRegisters

- **Description:** Read multiple 16bits registers.
- **Argin:**
  DEVVAR_SHORTARRAY : register address, no. of registers
- **Argout:**
  DEVVAR_SHORTARRAY : Holding 16bits register.
- **Command allowed for:**

8 - ReadInputRegisters

- **Description:** Read Multiple 16bits input registers.
- **Argin:**
  DEVVAR_SHORTARRAY : register address, no. of registers
- **Argout:**
  DEVVAR_SHORTARRAY : Input 16bits registers
- **Command allowed for:**

9 - PresetSingleRegister

- **Description:** Write single 16bits register.
- **Argin:**
  DEVVAR_SHORTARRAY : Register address, register value.
- **Argout:**
  DEV_VOID :
- **Command allowed for:**

10 - ReadExceptionStatus

- **Description:** Read exception status (usually a predefined range of 8 bits
- **Argin:**
  DEV_VOID :
- **Argout:**
  DEV_SHORT : exception status
- **Command allowed for:**

11 - FetchCommEventCtr

- **Description:** Fetch communications event counter.
- **Argin:**
  DEV_VOID :
- **Argout:**
  DEVVAR_SHORTARRAY : status, event count
- **Command allowed for:**

12 - ForceMultipleCoils

- **Description:** Write multiple coils (digital I/O) state. argin[0] = coil_address argin[1] = number of coils argin[2] = 1st coil state argin[3] = 2nd coil state ...
- **Argin:**
  DEVVAR_SHORTARRAY : coil address, nb of coils, coil states
- **Argout:**
  DEV_VOID :
- **Command allowed for:**

13 - ReadMultipleCoilsStatus

- **Description:** Read multiple coil (digital I/O) status. argin[0] = register address argin[1] = number of registers
- **Argin:**
  DEVVAR_SHORTARRAY : coil address, nb of coils

- **Argout:**
  DEVVAR_SHORTARRAY : Status of coils

- **Command allowed for:**

14 - PresetMultipleRegisters

- **Description:** Write multiple 16bits registers. argin[0] = register address argin[1] = number of registers argin[2] = 1st register argin[3] = 2nd register ...

- **Argin:**
  DEVVAR_SHORTARRAY : register address, nb of registers, register data

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

15 - MaskWriteRegister

- **Description:** Mask write a 16bits register.

- **Argin:**
  DEVVAR_SHORTARRAY : register address, AND mask, OR mask

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

16 - ReadWriteRegister

- **Description:** Read and Write multiple 16bits registers. argin[0] = read address argin[1] = nb of registers to read argin[2] = write address, argin[3] = nb of registers to write, argin[4] = 1st register value to write argin[5] = 2nd register value to write ...

- **Argin:**
  DEVVAR_SHORTARRAY : read address, no. to read, write address, nb.of write, write data

- **Argout:**
  DEVVAR_SHORTARRAY : read registers

- **Command allowed for:**