TOTUS Modbus Python Integration Application Note

Author: David Luca

Table of Contents

[Document History 1](#_Toc399836944)

[Introduction 1](#_Toc399836945)

[Configure MODBUS on TOTUS 2](#_Toc399836946)

[1. Configure Modbus via TCP 2](#_Toc399836947)

[2. Configure Modbus via Serial 3](#_Toc399836948)

[Totus Modbus Register map 3](#_Toc399836949)

[Pre-requisites 4](#_Toc399836950)

[Project Setup 5](#_Toc399836951)

[1 Connecting to Totus 7](#_Toc399836952)

[1.1 Connecting via TCP 7](#_Toc399836953)

[1.2 Connecting via Serial 8](#_Toc399836954)

[2. Reading values from Totus unit 8](#_Toc399836955)

[2.1 Reading temperatures 8](#_Toc399836956)

[2.2 Reading alarms 8](#_Toc399836957)

[2.3 Reading DGA values 9](#_Toc399836958)

[Conclusions 9](#_Toc399836959)

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author | Comments |
| 0.1 | 29/09/2014 | David Luca | First draft |

# Introduction

The purpose of the document is to introduce customers into using available MODBUS software libraries to connect their software to TOTUS instruments.

This document assumes you can access Totus interface by typing in your browser the IP address of the unit (in this document we will use <Totus-IP>). You may have to setup VPN connection, for which you need to contact Camlin Technologies.

# Configure MODBUS on TOTUS

## Configure Modbus via TCP

On TOTUS web interface navigate to **Settings**->**MODBUS** page. Press “Add New MODBUS Interface”, select “New TCP interface” then type in the form SlaveID (default 1) and the port to be used (i.e. 502, 1502) then press **Submit** button.



## Configure Modbus via Serial

On TOTUS web interface navigate to **Settings**->**MODBUS** page. Press “Add New MODBUS Interface” and select the Serial0, Serial1 or Serial2, specify on the form Baudrate, SlaveID (usually 1), data and stop bits, parity control, MODBUS protocol (ASCII/RTU) and flow control (usually “none” for RS232 or “hardware” for RS485 to enable direction via RTS line). For this application note we’ll be using 115200bps, 8 data bits, no parity, 1 stop bit then press **Submit** button.



# Totus Modbus Register map

On TOTUS web interface navigate to **Settings**->**MODBUS** page. Click on the link “Download Register Map” for this device to download the MODBUS map for this device. This is a table of MODBUS protocol description, meaning of the fields and register transfer requirements to read and/or write into the dictionary.

e.g.:

0000\_Common

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Topic** | **Address** | **Register** | **Number of Registers** | **Access** | **Format** | **Scaling** | **Offset** | **Notes** |
| UTC Time | 0 | 30001 | 2 | Read only | UINT32 |  |  | Clock time in UNIX time format; number of seconds from 1 Jan 1970. |
| ALARM/System/HL/State | 100 | 10101 | 1 | Read only | BOOLEAN |  |  | Combined system alarm for any non severe (H or L) alarm. |
| ALARM/System/HHLL/State | 101 | 10102 | 1 | Read only | BOOLEAN |  |  | Combined system alarm for any severe (HH or LL) alarm. |

# Pre-requisites

The code examples provided have been built using LiClipse IDE and Python 2.7 platform using latest version of Pymodbus from <https://code.google.com/p/pymodbus/>.

Dependencies needed: Setuptools (<https://pypi.python.org/packages/source/s/setuptools/setuptools-6.0.1.zip#md5=f35cd3a424145c68c235dcb7aef89c48>) then extract into an empty folder, open a command prompt into that folder and type: python setup.py install

For serial communications Pyserial library: <https://pypi.python.org/pypi/pyserial> ([pyserial-2.7.win32.exe](https://pypi.python.org/packages/any/p/pyserial/pyserial-2.7.win32.exe#md5=21555387937eeb79126cde25abee4b35))

If building under Windows install the compiler and set of system headers necessary for producing binary wheels for Python 2.7 packages from <http://aka.ms/vcpython27>

Download <https://code.google.com/p/pymodbus/downloads/detail?name=pymodbus-0.9.0.zip> and extract it into an empty folder then open a command prompt into that folder and type:

python setup.py install

Alternatively use SVN to checkout latest version of the library:

svn checkout http://pymodbus.googlecode.com/svn/trunk/ pymodbus-read-only  
cd pymodbus-read-only  
python setup.py install

Or from GitHub:

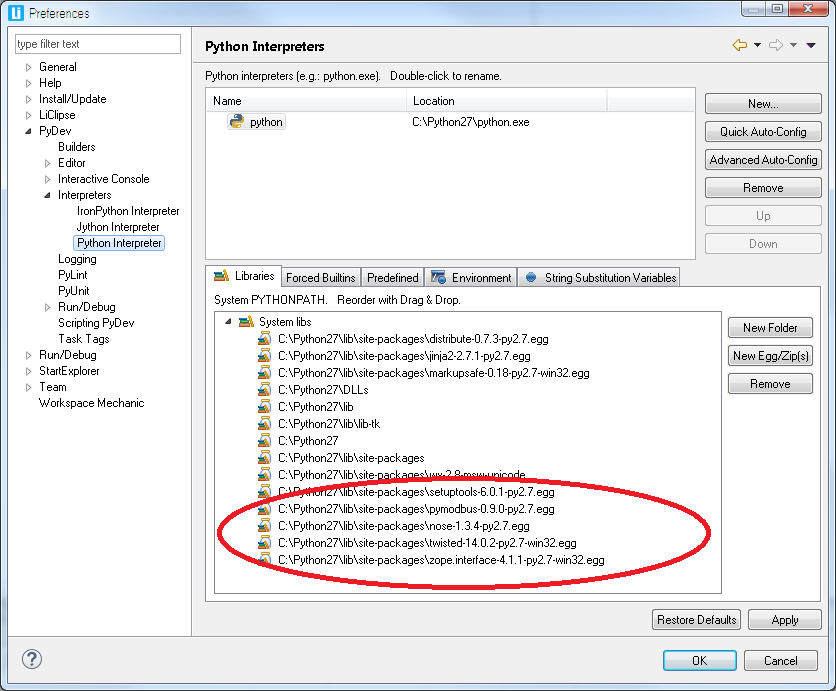
git clone <https://github.com/bashwork/pymodbus.git>

cd pymodbus

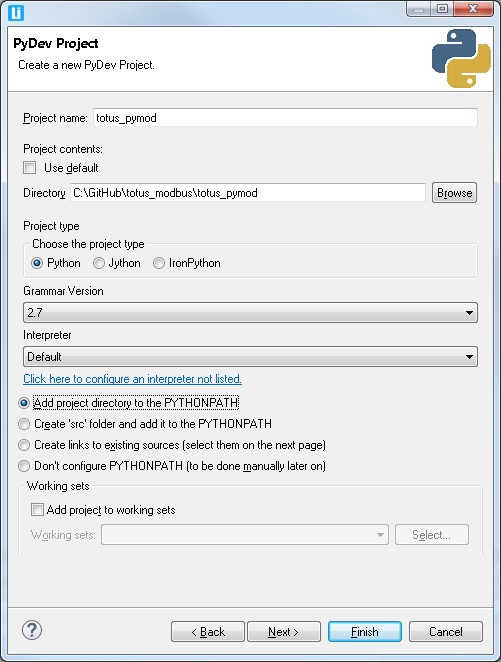
python setup.py install

# Project Setup

LiClipse needs Python 2.7 configured as interpreter so specify the path to Python environment using Window->Preferences menu, select PyDev->Interpreters->Python Interpreters in the left tree:

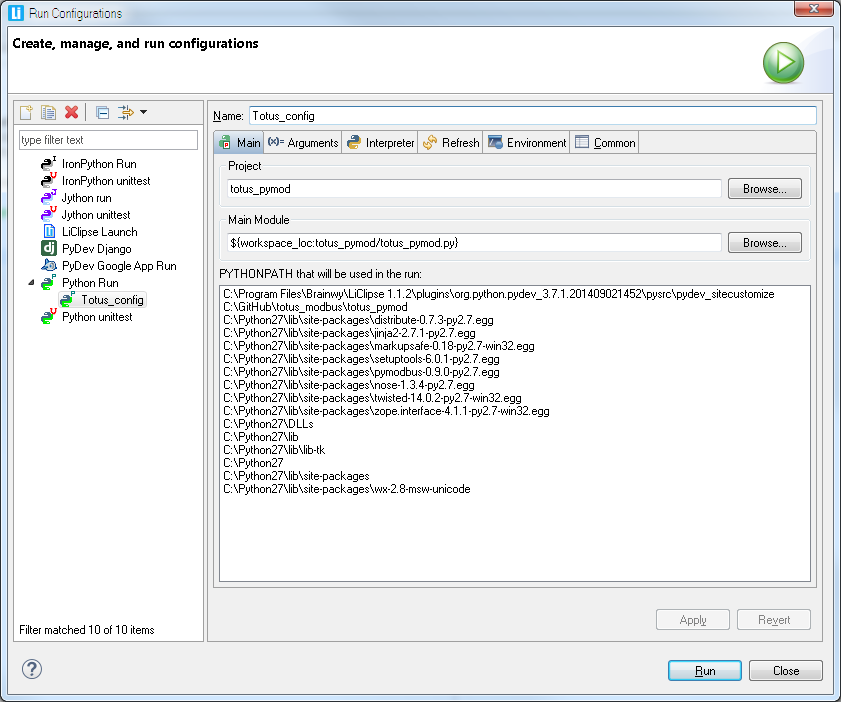


Create a new project using File->New->Project then select General->PyDev->PyDev Project in wizard window, press **Next** button, type the project name (e.g. totus\_pymod) and select a folder then press **Finish**.



Add a new file to the project by using File->New menu, select File wizard and specify “totus\_pymod.py” and press **Finish** button.

Press Run button on the toolbar  and press right-click on PyDev Run to add e new configuration and name it “Totus\_config”. Click on **Apply** button, then click on “Totus\_config”, select Main tab, press **Browse** button and select totus\_pymod project. Also near Main Module press **Browse** button and select totus\_pymod.py as startup module project:



## 1 Connecting to Totus

Jump to section 1.1 if connecting via TCP or section 1.2 if connecting via Serial connection.

### 1.1 Connecting via TCP

The following example shows how to initiate a connection to Totus unit via TCP port specified in the settings form (502).

Imports required:

import sys, traceback

import struct

from pymodbus.constants import Endian

from pymodbus.client.sync import ModbusTcpClient as ModbusClient

…

client = ModbusClient(*'192.168.46.83'*)

client.connect()

### 1.2 Connecting via Serial

Imports required:

import sys, traceback

import struct

from pymodbus.constants import Endian

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

…

client = ModbusClient(method=*'rtu'*, port=*'COM5'*, timeout=1, bytesize = 8, baudrate = 115200, stopbits = 1, parity = *'N'*) # parity can be 'E'ven, 'O'dd

client.connect()

## 2. Reading values from Totus unit

### 2.1 Reading temperatures

# read multiple int16 values

totusTemps = [

*"Thermal/AmbientTemp"*,

*"Thermal/AmbientTemp/1hAvg"*,

*"Thermal/AmbientHumidity"*,

*"Thermal/AmbientHumidity/1hAvg"*,

*"Thermal/TopOilTemp"*,

*"Thermal/TopOilTemp/1hAvg"*,

*"Thermal/BottomOilTemp"*,

*"Thermal/BottomOilTemp/1hAvg"*,

*"Thermal/TapChangerTemp"*,

*"Thermal/TapChangerTemp/1hAvg"*

]

numInputs = 10

startAddress = 1000

slaveID = 1

result = client.read\_input\_registers(startAddress, numInputs, slaveID)

for i in range(0, len(totusTemps)):

print *"Temp16 "* + str(startAddress + i) + *" "* + totusTemps[i] + *" = "* + str(result.getRegister(i)/10.0) + *"\xb0C"*# scaling is 10

### 2.2 Reading alarms

# read alarms

totusAlarms = [

*"ALARM/System/HL/State"*,

*"ALARM/System/HHLL/State"*

]

numInputs = 2

startAddress = 100

slaveID = 1

result = client.read\_discrete\_inputs(startAddress, numInputs, slaveID)

for i in range(0, len(totusAlarms)):

print *"Alarm "* + str(startAddress + i) + *" "* + totusAlarms[i] + *" = "* + str(result.getBit(i))

### 2.3 Reading DGA values

# read DGA float32 gases

totusDGA = [

*"DGA/SourceA/CH4"*,

*"DGA/SourceA/C2H6"*,

*"DGA/SourceA/C2H4"*,

*"DGA/SourceA/C2H2"*,

*"DGA/SourceA/CO"*,

*"DGA/SourceA/CO2"*,

*"DGA/SourceA/O2"*,

*"DGA/SourceA/N2"*,

*"DGA/SourceA/H2"*,

*"DGA/SourceA/H2O"*,

*"DGA/SourceA/TDCG"*,

*"DGA/SourceA/THC"*

]

numInputs = 12

startAddress = 2200

slaveID = 1

result = client.read\_input\_registers(startAddress, numInputs \* 2, slaveID)

for i in range(0, len(totusDGA)):

val = Convert2Float(result.getRegister(i\*2), result.getRegister((i\*2) + 1))

print *"Float32 "* + str(startAddress + i\*2) + *" "* + totusDGA[i] + *" = "* + str(val) + *" ppm"*

The floating point value are read as 16bit big endian values and assembled in a float using Convert2Float utility function:

def **Convert2Float**(a, b):

raw = struct.pack(*'>HH'*, a, b)

value = struct.unpack(*'>f'*, raw)[0]

return value

# Conclusions

This document exemplified Modbus interfacing using Open-Source libraries for various programming platforms which allows customers to implement their own Human Machine Interface to extract information from Totus units.

Example output (values with 3276.7 are for not connected sensors):

****