TOTUS Modbus Integration Application Note

Author: David Luca

Table of Contents

[Document History 1](#_Toc398904043)

[Introduction 2](#_Toc398904044)

[Totus Modbus Register map 2](#_Toc398904045)

[1. Integrating in C++ applications 3](#_Toc398904046)

[2. Integrating in C# applications 3](#_Toc398904047)

[2.1 Resources 4](#_Toc398904048)

[2.2 Connecting via TCP 5](#_Toc398904049)

[2.3 Connecting via Serial port 5](#_Toc398904050)

[2.4 Reading values from Totus unit 5](#_Toc398904051)

[2.4.1 Reading Alarms 6](#_Toc398904052)

[2.4.2 Reading temperatures 6](#_Toc398904053)

[2.4.3 Reading DGA values 6](#_Toc398904054)

[3. Integrating in Java applications 7](#_Toc398904055)

[3.2 Connecting via TCP 8](#_Toc398904056)

[3.3 Connecting via Serial 9](#_Toc398904057)

[3.4 Reading values from Totus unit 9](#_Toc398904058)

[3.4.1 Reading temperatures 9](#_Toc398904059)

[3.4.2 Reading alarms 9](#_Toc398904060)

[3.4.3 Reading DGA values 10](#_Toc398904061)

[4. Integrating in Python applications 11](#_Toc398904062)

[Conclusions 11](#_Toc398904063)

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author | Comments |
| 0.1 | 16/09/2014 | 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.

# Totus Modbus Register map

In TOTUS dashboard interface press **Settings** menu and select **MODBUS** menu or type in your browser http://<Totus-IP>/#modbus-settings link. There is a link “Download Register Map” for this device (or http://<Totus-IP>/modbus-map) where you will see a table of MODBUS protocol description, meaning of the fields and register transfer requirements to read and/or write into the dictionary.

I.e.:

Please note that this specification is still in draft status and may change prior to formal release.

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. |

To access Modbus via serial cable you need to press “Add New MODBUS Interface” and select the Serial0, 1 or 2, 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.



To access Modbus via TCP you need to 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.



# Integrating in C++ applications

The code examples provided have been built using Visual Studio C++ Express 2013 and latest version of libmodbus binaries ([libmodbus-3.0.6.tar.gz](http://libmodbus.org/releases/libmodbus-3.0.6.tar.gz)) zip archive from <http://libmodbus.org/download/>

# Integrating in C# applications

## 2.1 Resources

The code examples provided have been built using Visual Studio C# Express 2013 and latest version of NModbus ([NModbus\_net-3.5\_1.11.0.0-source.zip](https://code.google.com/p/nmodbus/downloads/detail?name=NModbus_net-3.5_1.11.0.0-source.zip&can=2&q=)) zip archive from: <https://code.google.com/p/nmodbus/downloads/list>.

Alternatively use SVN to checkout read-only sources for NModbus into an empty folder (i.e. E:\SVN):

svn checkout **http**://nmodbus.googlecode.com/svn/trunk/ nmodbus-read-only

Additional documentation for NModbus API: <http://ftp.icpdas.com/pub/cd/8000cd/napdos/modbus/nmodbus/nmodbus_api_manual_v1.2_en.pdf>

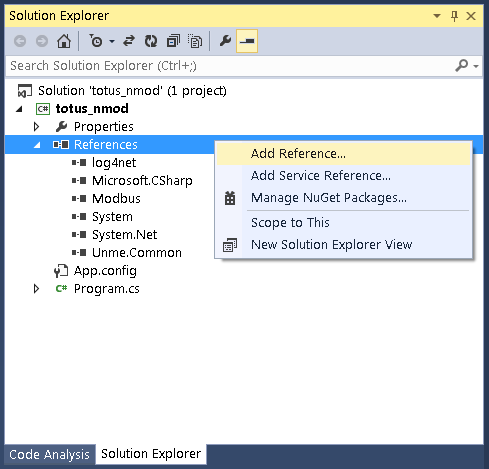
<http://www.icpdas.com/products/PAC/i-8000/modbus.htm>

https://code.google.com/p/nmodbus/

Example code is available in MySample folder inside the downloaded sources folder.

1. Create new project in Visual Studio C# Express 2013 (i.e. Totus\_nmod).

2. In Solution Explorer expand the project and right-click References line, select Add References… and browse for **Modbus.dll, Log4net.dll and Unme.Common.dll** extracted from binary archive or SVN checkout folder.

****

## 2.2 Connecting via TCP

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

using System.Net;//for tcp client

using System.Net.Sockets;

using Modbus.Device;//for modbus master

…..

/\*

\* Connecting via TCP

\*/

TcpClient client = new TcpClient("192.168.42.37", 502); //IP and port of the TOTUS unit

Console.WriteLine("Client connected:{0}", client.Connected.ToString());

ModbusIpMaster master = ModbusIpMaster.CreateIp(client);

master.Transport.ReadTimeout = 1000; //ms

## 2.3 Connecting via Serial port

The following example shows how to connect to Totus unit via Serial port:

using Modbus.Device;//for modbus master

using System.IO.Ports;//for access to PC port

…..

/\*

\* Connecting via Serial

\*/

SerialPort serialPort = new SerialPort(); //Create a new SerialPort object.

serialPort.PortName = "COM1"; //PC port

serialPort.BaudRate = 115200; //baud rate

serialPort.DataBits = 8;

serialPort.Parity = Parity.None;

serialPort.StopBits = StopBits.One;

serialPort.RtsEnable = false; //false for RS232, true for RS485

serialPort.Open();

ModbusSerialMaster master = ModbusSerialMaster.CreateRtu(serialPort);//or .CreateAscii(serialPort)

master.Transport.ReadTimeout = 1000;//ms

The settings for baudrate, data and stop bits need to match the configured settings on Totus Settings->MODBUS page. Depending on ASCII or RTU selection use ModbusSerialMaster.CreateAscii or ModbusSerialMaster.CreateRtu functions in your code.

## 2.4 Reading values from Totus unit

Reading is done via ReadInputs function from library that returns array of values depending on the type of the dictionary:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

…..

### 2.4.1 Reading Alarms

//read alarms

byte slaveID = 1;

ushort numInputs = 2;

ushort startAddress = 100;

bool[] alarms = master.ReadInputs(slaveID, startAddress, numInputs);

string[] totusAlarms = {

"ALARM/System/HL/State",

"ALARM/System/HHLL/State"

};

for (int i = 0; i < numInputs; i++)

{

Console.WriteLine("Alarm {0} {1} = {2}", startAddress + i, totusAlarms[i], alarms[i] ? 1 : 0);

}

### 2.4.2 Reading temperatures

//read int16 temperatures

ushort numInputs = 10;

byte slaveID = 1;

ushort startAddress = 1000; //select address from Totus Modbus table

ushort[] temps = master.ReadInputRegisters(slaveID, startAddress, numInputs);//\*2 because we are reading 2byte unsigned short that needs converted to 4 byte floats

for (int i = 0; i < numInputs; i++)

{

Console.WriteLine("Temp16 {0} {1} = {2}°C", startAddress + i, totusTemps[i], (float)temps[i] / 10); // divide by 10 as specified in Scaling column

}

### 2.4.3 Reading DGA values

//read DGA float32 gases

string[] 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"

};

byte slaveID = 1;

ushort numInputs = 12;

ushort startAddress = 2200; //select address from Totus Modbus table

ushort[] inputsdga = master.ReadInputRegisters(slaveID, startAddress, (ushort)(numInputs \* 2));//\*2 because we are reading 2byte unsigned short that needs converted to 4 byte floats

for (int i = 0; i < numInputs; i++)

{

Console.WriteLine("Float32 {0} {1} = {2} ppm", startAddress + i \* 2, totusDGA[i], Convert2Float(inputsdga[i \* 2], inputsdga[i \* 2 + 1]));

}

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

static float Convert2Float(ushort high, ushort low)

{

//Convert ushort array to Float

ushort[] data = new ushort[2] { low, high }; //Big endian

float[] floatData = new float[data.Length / 2];

Buffer.BlockCopy(data, 0, floatData, 0, data.Length \* 2);

return floatData[0];

}

# Integrating in Java applications

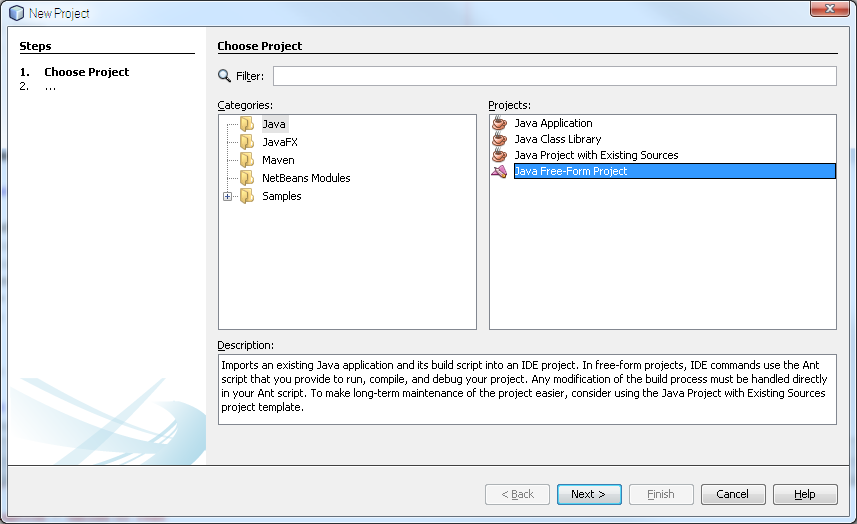
The code examples provided have been built using Netbeans IDE and Oracle Java 8 platform using latest version of JaMod sources from <http://sourceforge.net/projects/jamod/files/jamod/1.2/>.

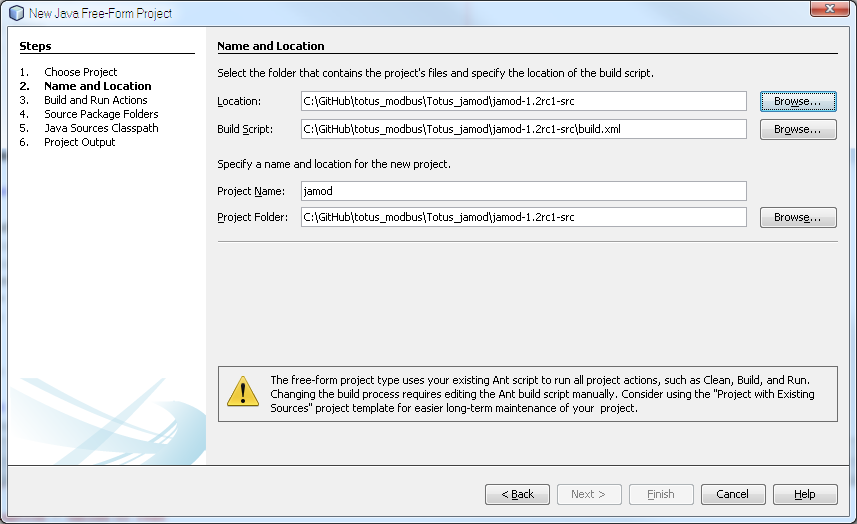
You also need Java Communications API 3.0 from <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-misc-419423.html#java_comm_api-30u1> if building for Solaris/Linux or for Windows you can use <http://smslib.org/download/> that provides links to different platforms for serial communication libraries.

The project Totus\_jamod described below includes the Jamod 1.2rc1 built against RXTX library from Mfizz Inc. since Java Communications API doesn’t support x64 bit Windows platforms.

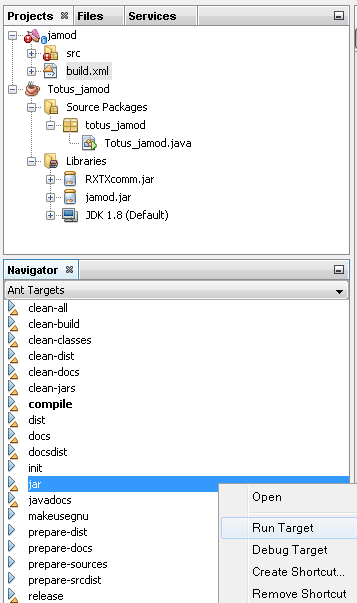
From <http://mfizz.com/oss/rxtx-for-java> download <https://bitbucket.org/jlauer/mfz-cdn/downloads/mfz-rxtx-2.2-20081207-win-x64.zip> which contains RXTXcomm.jar that needs copied into jamod-1.2rc1-src\lib folder.

To build Jamod using NetBeans 8 extract the sources to an empty folder and select File -> New Project menu, then in **Categories** pick **Java** and then in **Projects** select **Java Free-Form Project** and hit **Next** button. After that you will configure the project by selecting the location of the project (jamod-1.2rc1-src/build.xml) and then associate targets in the build.xml with actions in the IDE and click Finish. Edit jamod-1.2rc1-src/build.properties file to enable build.serial.gnu=true for building against RXTX (Jamod website by default uses false so it’s built for comm.jar which is Java Communications API).

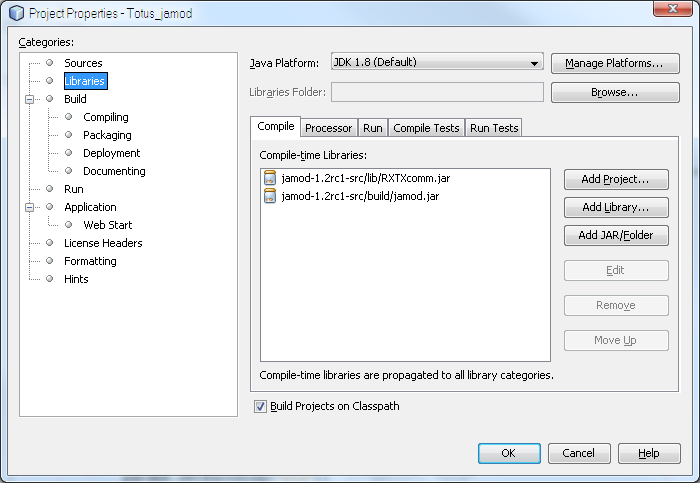




Click on build.xml and then right click on **“compile”** target, click on Run Target to compile classes, then click on **“jar”** and Run Target, which builds jamod.jar.



Create a new project (i.e. totus\_jamod) then add jamod-1.2rc1-src/build/jamod.jar built in previous step and jamod-1.2rc1-src/lib/RXTXcomm.jar (or the jamod-1.2-SNAPSHOT.jar and comm.jar from project website if building for Linux/Solaris using Java Communications API) to project library by accessing Project Properties->Library category and **Add JAR/Folder** button:



Imports required:

package totus\_jamod;

import java.net.\*;

import java.io.\*;

import net.wimpi.modbus.\*;

import net.wimpi.modbus.msg.\*;

import net.wimpi.modbus.io.\*;

import net.wimpi.modbus.net.\*;

import net.wimpi.modbus.util.\*;

import java.nio.\*;

import gnu.io.\*; //for RXTX library

//import javax.comm.\*; //for Java Communications API;

## 3.2 Connecting via TCP

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

TCPMasterConnection con = new TCPMasterConnection(InetAddress.getByName("192.168.42.37"));

con.setPort(502); //port as configured on the unit

con.connect(); //connect to unit

ModbusTCPTransaction trans = new ModbusTCPTransaction(con);

## 3.3 Connecting via Serial

For Windows platforms you need to have the proper 32bit/64bit rxtxSerial.dll file in Totus\_jamod application folder or in your Windows\System32 folder.

//Setup serial parameters

SerialParameters params = new SerialParameters();

params.setPortName("COM1");//PC COM port

params.setBaudRate(115200);//baudrate set in Totus unit

params.setDatabits(SerialPort.DATABITS\_8);

params.setParity(SerialPort.PARITY\_NONE);

params.setStopbits(SerialPort.STOPBITS\_1);

params.setEncoding("rtu"); //"ascii", "rtu"

params.setEcho(false);

//params.setReceiveTimeout(3000);//not available in RXTX

params.setFlowControlIn(SerialPort.FLOWCONTROL\_NONE);//FLOWCONTROL\_NONE for RS232, FLOWCONTROL\_RTSCTS\_IN for RS485

params.setFlowControlOut(SerialPort.FLOWCONTROL\_NONE); ////FLOWCONTROL\_NONE for RS232, FLOWCONTROL\_RTSCTS\_OUT for RS485

SerialConnection con = new SerialConnection(params);

ModbusSerialTransaction trans = new ModbusSerialTransaction(con);

## 3.4 Reading values from Totus unit

Reading is done via trans.setRequest() method that receive requests of type ReadInputRegistersRequest/ReadInputDiscretesRequest and calling trans.execute() then trans.getResponse() returns ReadInputRegistersResponse/ReadInputDiscretesResponse objects with array of values depending on the type of the dictionary.

### 3.4.1 Reading temperatures

int startAddress = 1000;

int numInputs = 10;

ReadInputRegistersRequest req = new ReadInputRegistersRequest(startAddress, numInputs);

req.setUnitID(1);

trans.setRequest(req);

trans.execute();

ReadInputRegistersResponse res = (ReadInputRegistersResponse)trans.getResponse();

String 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"

};

for (int i = 0; i < numInputs; i++)

{

float temp = res.getRegister(i).getValue();

System.out.println("Temp16 " + (startAddress + i) + " " + totusTemps[i] + " = " + temp / 10.0 + "°C");

}

### 3.4.2 Reading alarms

int startAddress = 100; //register map

int numInputs = 2;

ReadInputDiscretesRequest req = new ReadInputDiscretesRequest(startAddress, numInputs);

req.setUnitID(1); //slave ID of the unit

trans.setRequest(req);

trans.execute();

ReadInputDiscretesResponse res = (ReadInputDiscretesResponse)trans.getResponse();

String totusAlarms[] = {

"ALARM/System/HL/State",

"ALARM/System/HHLL/State"

};

for (int i = 0; i < numInputs; i++)

{

boolean bit = res.getDiscretes().getBit(i);

System.out.println("Alarm " + (startAddress + i) + " " + totusAlarms[i] + " = " + bit);

}

### 3.4.3 Reading DGA values

int startAddress = 2200;

int numInputs = 12;

ReadInputRegistersRequest req = new ReadInputRegistersRequest(startAddress, numInputs \* 2);//float=2\*ints16

req.setUnitID(1);

trans.setRequest(req);

trans.execute();

ReadInputRegistersResponse res = (ReadInputRegistersResponse)trans.getResponse();

String 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"

};

for (int i = 0; i < numInputs; i++)

{

float ppm = Convert2Float(res.getRegister(i \* 2).toBytes(), res.getRegister((i \* 2) + 1).toBytes());

System.out.println("Float32 " + (startAddress + i \* 2) + " " + totusDGA[i] + " = " + ppm + " ppm");

}

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

public static float Convert2Float(byte[] a, byte[] b)

{

ByteBuffer bbuffer = ByteBuffer.allocate(a.length + b.length);

bbuffer.put(a);

bbuffer.put(b);

bbuffer.compact(); // no need if backing array is sized appropriately to begin with

float result = ByteBuffer.wrap(bbuffer.array()).order(ByteOrder.BIG\_ENDIAN).getFloat();

return result;

}

# Integrating in Python applications

The code examples provided have been built using Python 2.7 and latest version of PyModBus library ([v1.3.0](https://github.com/bashwork/pymodbus/archive/master.zip)) from https://github.com/bashwork/pymodbus

# 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):

****