SAPHIR

SIGNAL PHYSIQUE & INSTRUMENTATION

GModBus over TCP

USER GUIDE

ModBus +

Over TCP

COMPATIBLE WITH NATIONAL INSTRUMENTS LabVIEW SILVER PRODUCT

Table of contents

1. Foreword

TCP/IP ModBus protocol is a communication protocol based on a client – server structure. The network can be composed of several clients and several servers connected with a RJ45 link. One or several ModBus networks can be implemented on a same platform.

Through this protocol, only the client can prompt the exchange with the server by sending a request and waiting for an answer. The validity of the communication is controlled by the TCP/IP layer.

GModBus over TCP driver encapsulates all these layers in order to make it easy for the developer to insert a computer, as client or server, within such a network.

The following functionalities, hidden to the user, are managed:

- Link and Network low layers of ModBus procedure
- Encoding/decoding ModBus frames
- Frames control through identification and timeout
- TCP/IP communication management

2. Configuration

2.1 Software configuration

"GModBus over TCP" driver runs under the following LabVIEW:

- 2010 and later

And on the following platforms:

- PC under Windows XP and later
- RT system (Real Time)

2.2 Hardware : Network connection

There are 2 ways to connect a computer to an Ethernet ModBus network:

- **Point to point link:**
  The computer and the equipment are networked through a « crossed » Ethernet wire.
- **Ethernet network:**
  The computer is connected to an existing network through a straight Ethernet wire.

3. GModBus

3.1 GModBus over TCP within LabVIEW

"GModBus over TCP" driver installation adds the *"GModBus over TCP"* palette to LabVIEW functions palette.

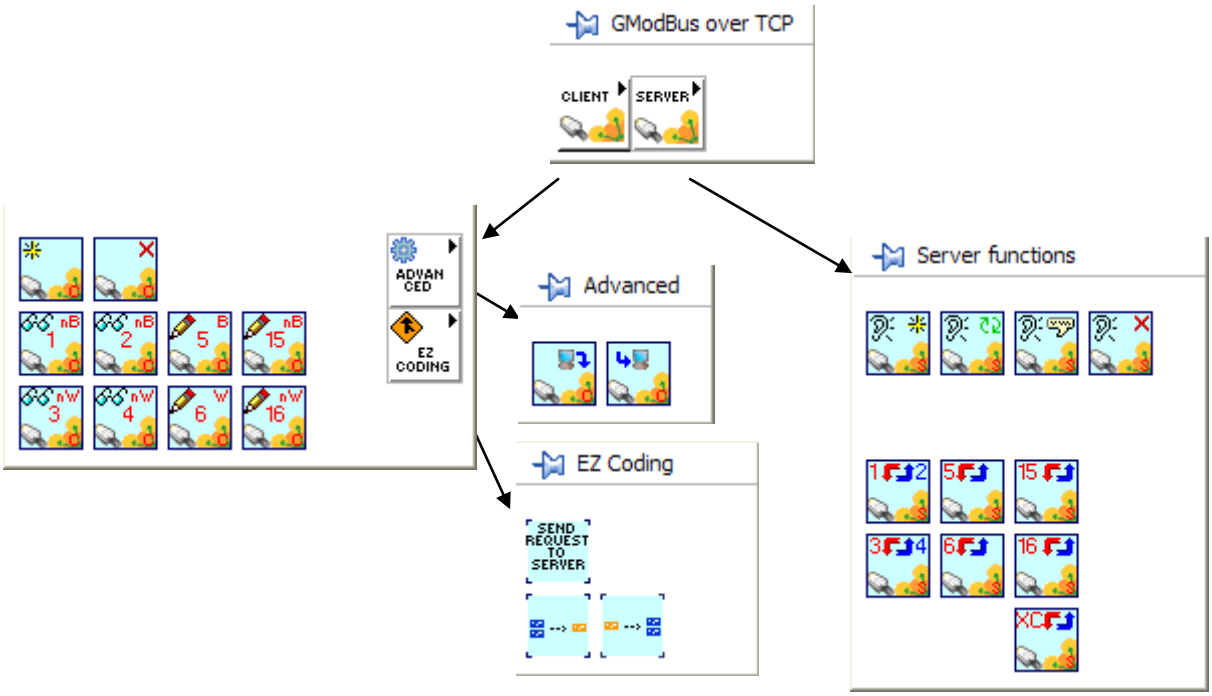


Figure 1 : "GModBus over TCP" within LabVIEW functions palette

4. GModBus over TCP components

4.1 Foreword and writing conventions

The set of VIs that composes *"GModBus over TCP"* driver follows the connector model below:

Figure 2 : Connector model of "GModBus over TCP" driver

- ***error in:*** describes the error conditions found before the VI. The default value corresponds to "no error". If an error is transmitted to the ***error in*** input of the VI, this error is relayed to ***error out*** without the VI executing its function. If an error occurs when the function is running, this error is automatically relayed to ***error out***.

The connectors respect LabVIEW conventions as follows:

The label *connectorname (x)* means *x* is the default value associated with this connector if no other value is conveyed to its Input.

The connectors which names appear in bold in the context help of a VI must be wired. Otherwise the caller VI will not be able to run (broken arrow).

4.2 Client tools

This chapter describes the VIs to use to realise a client for a ModBus network. These VIs are found in the Functions palette by selecting Functions > SAPHIR > "GModBus over TCP" > Client_Tools.

Figure 3: Clients VIs palette

4.2.1 Open

Figure 4: MBVTCP_open.vi

This VI initializes the TCP/IP communication described by ***Network. NetRefTCP out***, unique reference to the network, is required by the other *"GModBus over TCP"* VIs managing the same client.

- ***Network:*** input is defined by the two following elements:
  - ***Server address:*** IP address of the server concerned by the requests (ex :196.168.25.42).
  - ***Remote port (502)***: port used for the TCP/IP communication.

If the connection is not opened within the time defined by ***timeout ms (60000)***, the VI generates an error.

4.2.2 Close

Figure 5: MBVTCP_close.vi

This VI closes the TCP/IP communication of the Network associated with ***NetRef in***.

It is imperative for the release process to be done properly to free the memory resources of the computer.

4.2.3 Request VIs

4.2.3.a Connector model

The set of VIs that composes the *Client* part of *"GModBus over TCP"* driver follows the connector model below:

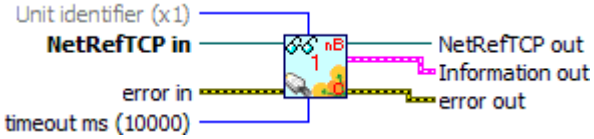


Figure 6: Connector model of Clients VIs

- ***NetRefTCP in / NetRefTCP out***: ***NetRefTCP in*** is the reference to the ModBus network obtained at the opening of the communication (cf § 4.2.1-) ***NetRefTCP out*** is a copy of ***NetRefTCP in***.

- ***error in / error out*** : ***error in*** describes the errors conditions found before the VI. The default value corresponds to "no error". If an error is transmitted to the ***error in*** input of the VI, this error is relayed to ***error out*** without the VI executing its function. If an error occurs when the function is running, this error is automatically relayed to ***error out***.
- ***Timeout :*** If the client request does not get any response within the time defined by ***timeout ms (10000)***, the VI exits with an error.
- ***Unit identifier (x1):*** is used for the communication with a serial network through a gateway.
- ***Information out:*** is a cluster that contains the following information:
- ***Unit Identifier:*** Identical to ***Unit identifier (x1)***.
- ***exceptionCode:*** Code refering to ModBus protocol exceptions (cf. § 7.2). The default value is 0: no exception occurred.
- ***functionCode:*** Number of the used request
- ***sendFrame:*** String sent to the server equipment. (This data is given for information, *"GModBus over TCP"* driver deals with the sending of the frame by itself).
- ***receivedFrame:*** String received by the client (sent by the server as an answer to the request)

4.2.3.b Request 1: Reading of N output bits

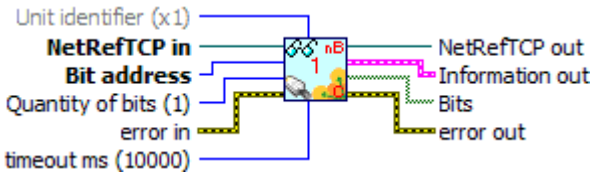


Figure 7: MBVTCP_lecNBitsSortie(1).vi

This VI is used to read consecutive output ***Bits*** defined in the memory of the destination server.

- ***Bit address***: address of the first bit
- ***Quantity of bits (1):*** number of bits to read
- ***Bits:*** value of the read bits

4.2.3.c Request 2: reading of N input bits

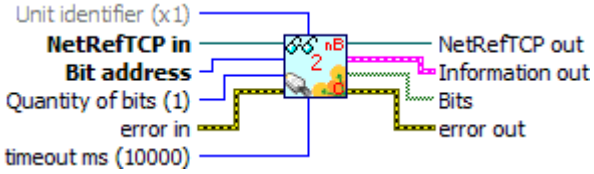


Figure 8: MBVTCP_lecNBitsEntree(2).vi

This VI is used to read consecutive input ***Bits*** defined in the memory of the destination server.

- ***Bit address:*** address of the first bit
- ***Quantity of bits (1)***: the number of bits to read

- ***Bits:*** value of the read bits

4.2.3.d Request 3: reading of N output words

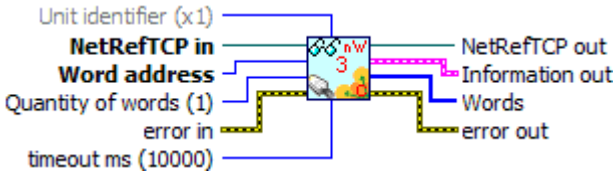


Figure 9: MBVTCP_lecNMotsSortie(3).vi

This VI is used to read consecutive output ***Words*** defined in the memory of the destination server.

- ***Word address:*** address of the first word
- ***Quantity of Words (1):*** number of words to read
- ***Words*** value of the read words

4.2.3.e Request 4: reading of N Input words

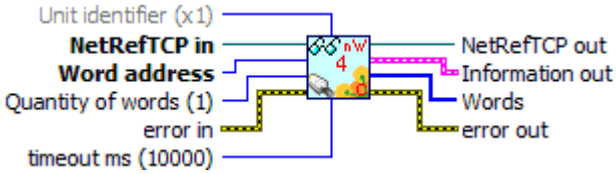


Figure 10: MBVTCP_lecNMotsEntree(4).vi

This VI is used to read consecutive input ***Words*** defined in the memory of the destination server.

- ***Word address:*** address of the first word
- ***Quantity of Words (1):*** number of words to read
- ***Words:*** value of the read words

4.2.3.f Request 5: writing of an output bit

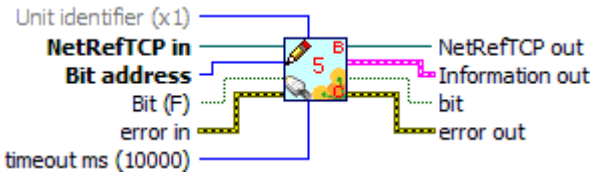


Figure 11: MBVTCP_ecrBitSortie(5).vi

This VI is used to write (at 0 or at 1) an output Bit in the memory of the destination server.

- ***Bit address***: address of the bit to write
- ***Bit (F):*** value of the bit to write

4.2.3.g Request 6: writing of an output word

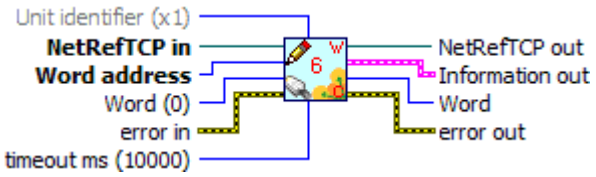


Figure 12: MBVTCP_ecrMotSortie(6).vi

This VI is used to write an output Word in the memory of the destination server.

- ***Word address:*** address of the word to write
- ***Word (0)***: value of the word to write

4.2.3.h Request 15: writing of N output bits

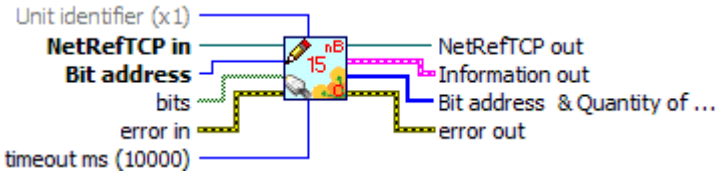


Figure 13: MBVTCP_ecrNBitsSortie(15).vi

This VI is used to write (at 0 or at 1) a group of consecutive output bits in the memory of the destination server.

- ***Bit address***: address of the first bit to write
- ***Bits***: number of bits to write and their values

The driver sends groups of 8 bits. If the number of bits written is not a multiple of 8, the driver fills the missing bits as FALSE. Depending on the equipment of destination, these bits can be interpreted or not.

4.2.3.i Request 16: writing of N output words

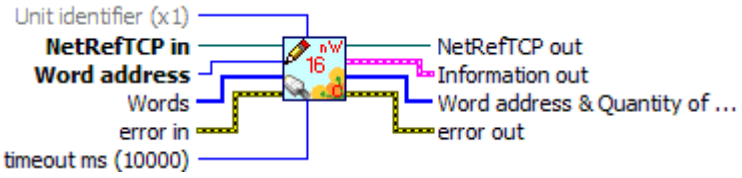


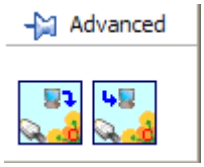
Figure 14: MBVTCP_ecrNmotsSortie(16).vi

This VI is used to write a group of consecutive output words (16 bits) in the memory of the destination server.

- ***Word address***: address of the first word to write
- ***Words***: number of words to write and their values

4.2.4 Advanced palette

These VIs manage basics communication functions and make it possible to gain performance in comparison with the requests VIs that are more high level.



The MBVTCP_SendRequest.vi sends a request to a ModBus server and returns a reference to this request through the *Expected answer* field. This reference is used to get the associate answer with MBVTCP_ReceiveResponse.vi.

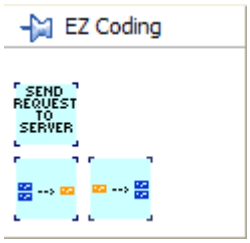
The *Send frame* is given just for information.

According to the request type the type of data received will be different (some data in the dataReceived frame may be empty).

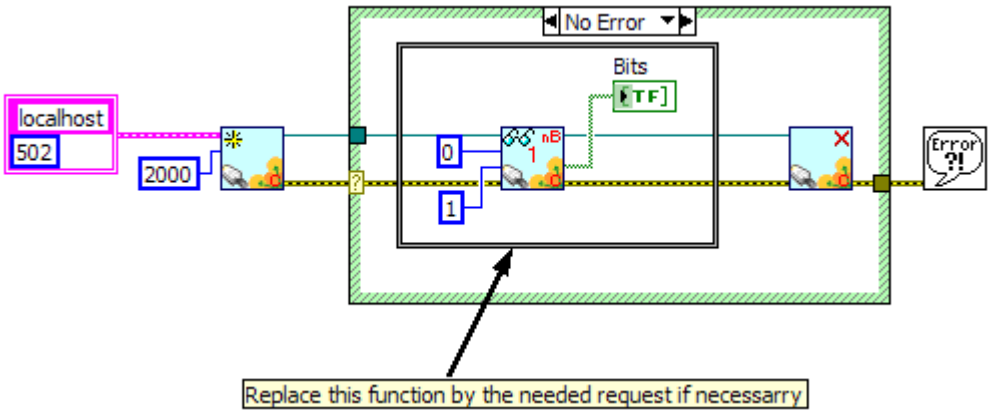
These functions could be useful in some specific use case performance or "advanced synchronisation" are needed.

4.2.5 EZ Coding VIs

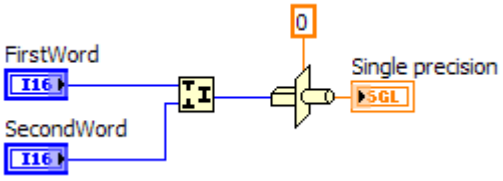
EZ Coding VIs are to be dropped on an existing VI. They propose a starting architecture to the implementation of a ModBus client.



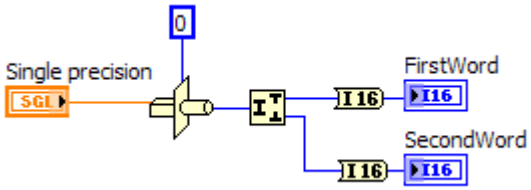
Drop this VI into the block diagram to place his content and customized it.



This code implements a simple request to a ModBus server.



This code implements a way to interpret 2 words to obtain a single.



This code implements a way to interpret a single to obtain 2 words.

4.3 Server tools

This chapter describes the VIs to use to realise a Server for ModBus network. These VIs are found in the Functions palette by selecting Functions > SAPHIR > "GmodBus over TCP" > Server_Tools.

Figure 15: Server VIs palette

In *Server* mode, the computer never initiates the communication. It carries out three different tasks:

- Supervise clients' connections and disconnections.
- Receive the requests sent by the clients.
- Answer to the clients' requests.

4.3.1 Connections scanning

The scanning of the clients' connections and the managing of the communications with the clients must be carried out independently.

The three following VIs are used to implement the scanning of the different connections to the server.

4.3.1.a Initialization of the scanning

Figure 16: MBVTCP_initializeListener.vi

- *port (502):* number of the port to scan.
- *listener ID out:* reference needed for the scanning of the connections.

4.3.1.b Listening of the connections

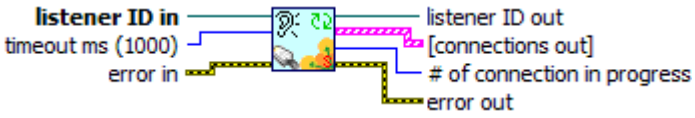


Figure 17: MBVTCP_listenConnections.vi

- *listener ID in:* reference obtained during the initialization of the listening.
- *TimeOut ms (1000):* maximum time to wait for a connection.
- *listener ID out:* copy of *listener ID in*.
- *[connections out]:* array containing the TCP/IP communication reference (NetRefTCP out) and the IP address (remote address) of each client connected to the server.
- *# of connections in progress:* number of clients connected to the server.

4.3.1.c Stop of the scanning

When you stop the server, you must close the *listener ID* reference with the above VI.

This process makes it possible to release properly the memory resources of the computer.

Figure 18: MBVTCP_closeListener.vi

- *listener ID in:* reference to close.

4.3.2 Requests management

4.3.2.a Requests reception

The reception of the requests sent by the client(s) is carried out with the following VI:

Figure 19: MBVTCP_listenRequest.vi

- *[received request]* returns a table of which each element describes the request received :
- *Connection ID:* TCP/IP reference of the client who sent the request.
- *remote address:* IP address of the client who sent the request.
- *receivedFrame*: String written in the request.
- *Transaction Identifier:* Specific number of the request sent by the client allowing identifying the answer to send.
- *Unit Identifier:* identifies a client located on a serial network and communicating through the Ethernet network.
- *exceptionCode:* Code refering to ModBus protocol exceptions (cf. §7.2). 0 by default, no exception occured.
- *functionCode:* Number of the request sent.
- *data :* Data contained in the request.
- *# of received request:* number of requests received.

After receiving a request, the answer must be sent as soon as possible to avoid timeout errors on the Client side.

4.3.2.b Connector model of answer to the requests VIs

The set of VIs that composes the answer request part of *"GModBus over TCP" driver* follows the connector model below:

Figure 20: Connector model of answer to requests VIs

- *Received request:* contains data about the processed request.
- *Error in:* describes the errors conditions found before the VI. The default value corresponds to "no error". If an error is transmitted to the *error in* input of the VI, this error is relayed to *error out* without the VI executing its function. If an error occurs when the function is running, this error is automatically relayed to *error out*.

- *Send frame:* describes the frame sent by the server to the client. (This data is given just for information)

4.3.2.c Answer to requests 1 and 2 (reading N input or output bits)

Figure 21: MBVTCP_repLectureNBits(1_2).vi

This VI permits to answer to the requests 1 or 2 sent by the client through ModBus network.

- *Bits*: array containing the values of all the registers of the server

4.3.2.d Answer to requests 3 and 4 (reading N input or output words)

Figure 22: MBVTCP_repLectureNMots(3_4).vi

This VI permits to answer to the requests 3 or 4 sent by the client through ModBus network

- *Words*: array containing the values of all the registers of the server

4.3.2.e Answer to request 5 (writing an output bit)

Figure 23: MBV_repEcritureBit(5).vi

This VI permits to answer to request 5 sent by the client through ModBus network.

- *Bits in:* array containing the values of all the registers of the server
- *Bits out:* values of the server registers after the processing of the request

4.3.2.f Answer to request 6 (writing an output word)

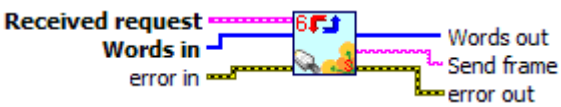


Figure 24: MBVTCP_repEcritureMot(6).vi

This VI is used to write an output Word in the memory of the destination server.

- *Word address:* address of the word to write
- *Word (0)*: value of the word to write

4.3.2.g Answer to request 15 (writing N output bits)

Figure 25: MBVTCP_repEcritureNBits(15).vi

This VI permits to answer to request 15 sent by the client through ModBus network.

- *Bits in*: array containing the values of all the registers of the server
- *Bits out*: values of the server registers after the processing of the request

4.3.2.h Answer to request 16 (writing of N output words)

Figure 26: MBV_repEcritureNMots(16).vi

This VI permits to answer to request 16 sent by the client through ModBus network.

- *Words in:* array containing the values of all the registers of the server
- *Words out*: values of the server registers after the processing of the request

4.3.2.i Generation of an exception code

Figure 27: MBV_repException.vi

This VI returns an exception to a ModBus network client in case its request is not supported by the server.

- *Exception Code:* Code of the exception to send (cf § 7.2).

4.4 Tools

This chapter describes tools to quickly simulate a ModBus client or server. You will find them in the LabVIEW Tools menu bar.

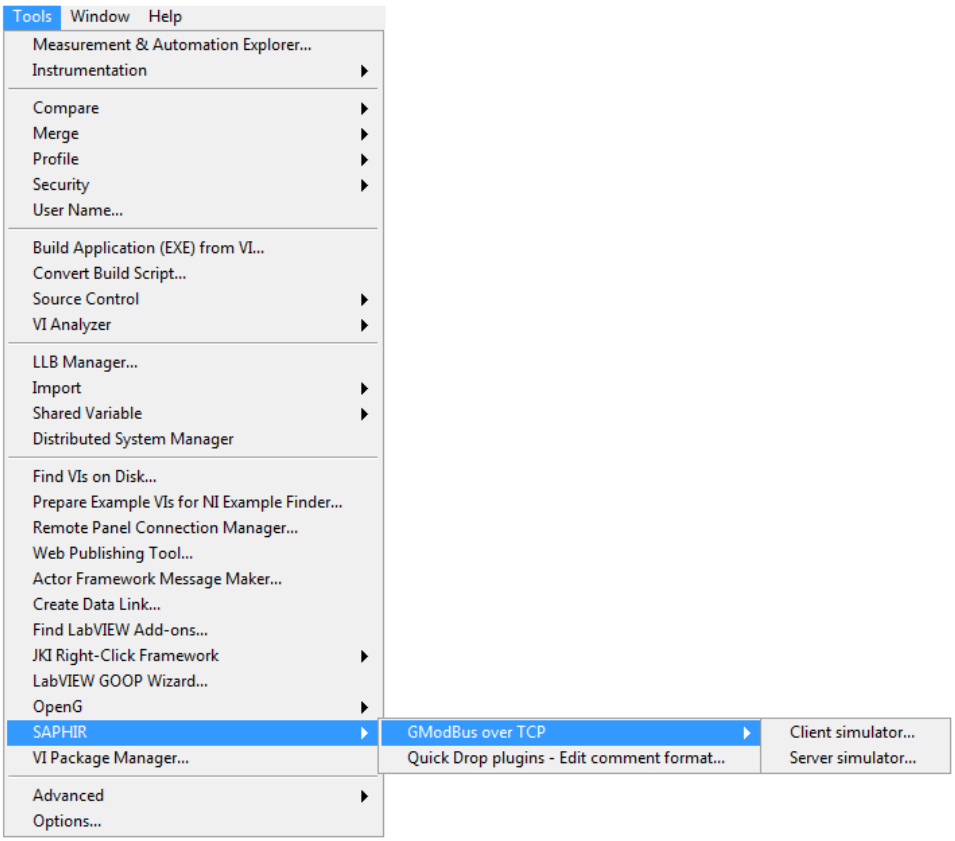


Figure 28: Tools to simulate ModBus network elements

4.4.1 GModBus

« GModBus over TCP » will quickly test the communication with a server through ModBus network.

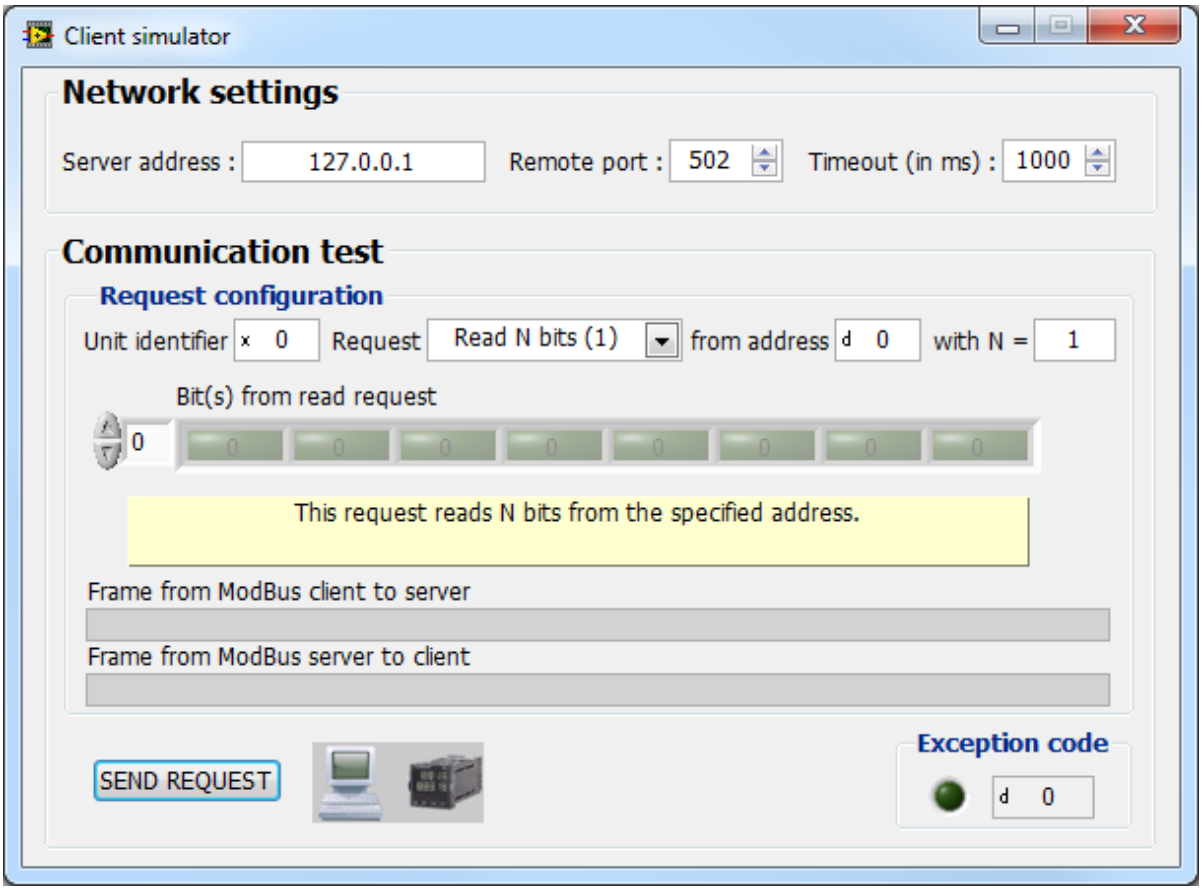


Figure 29: Client window

The interface falls into two sections:

- **Network settings:** defines the TCP/IP parameters of the ModBus network:
- **Server address:** IP address of the server to test.
- **Remote port:** Number of the port used for the TCP/IP communication.
- **Timeout (in ms):** maximum time to wait to receive the server answer.

Communication test

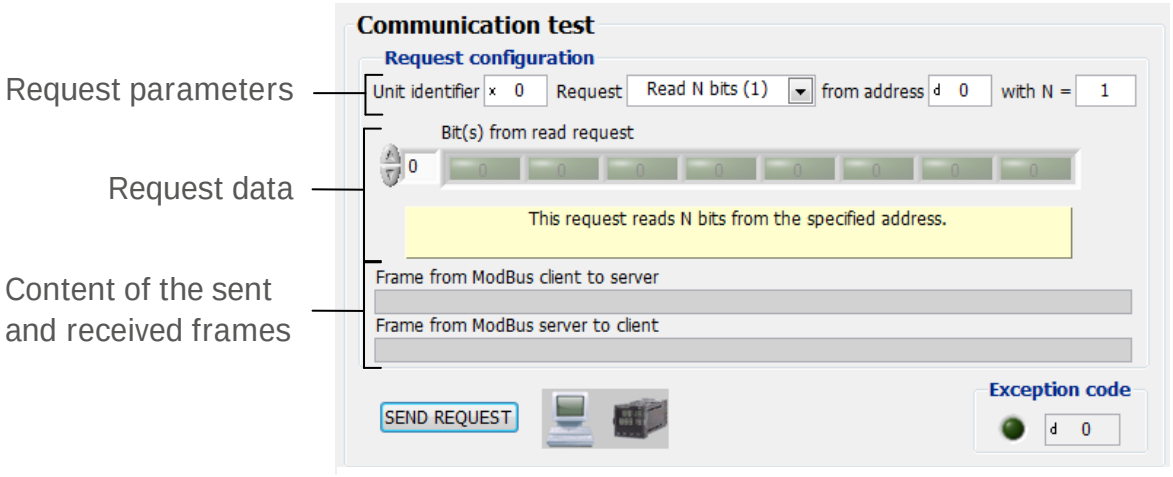


Figure 30: Counication test interface in Client mode

All the types of *Request* of GModBus over TCP driver are managed:

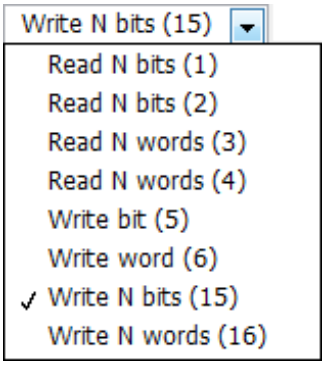


Figure 31: Requests choice

Unit Identifier identifies an equipment located on a serial network and communicating through the Ethernet network. If the contacted server is on the Ethernet network, keep the FF hexadecimal value (256 in decimal).

The first register to read or write is defined by *From address.*

The field *With N =* is only available for requests 1, 2, 3 and 4. It represents the number of bit(s) or word(s) to read or to write.

The data zone, located below the request parameters, permits to determine the values to write during the use of writing requests.

When all the settings are done, click on *SEND REQUEST* button to start the communication with the server. The content of the frames sent and received by the client is displayed below the data zone.

The *Exception code* refers to ModBus protocol exceptions (cf. §7.2).

4.4.2 GModBus over TCP server

« GModBus over TCP Server» application simulates a server of which registers are represented with an array of bits and an array of words.

A ModBus network client can read or write these tables.

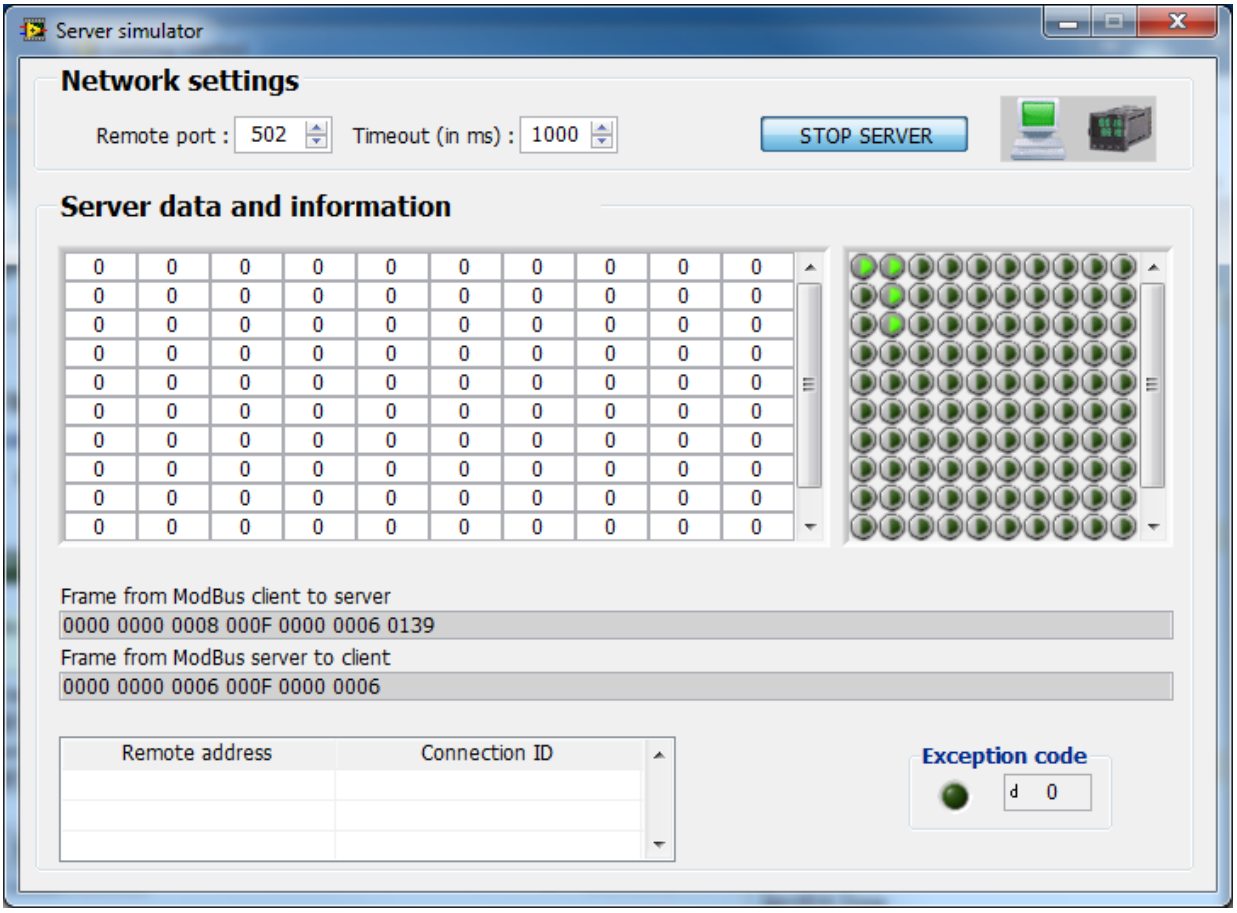


Figure 32: Server window

Three sections compose this interface:

The **Network settings** permits to configure the ModBus server parameters:

Remote port (502): Ethernet communication port of the server

Timeout (in ms): time left for each scanning of the Ethernet port.

When clicking on the *RUN SERVER* button the server begins to scan the client's connections (listening state).

This change of state is notified by the computer icon which becomes green. The *address* and the *Connection ID* of the connected clients are displayed in the array besides.

The settings of ModBus network can't be modified when the server is listening.

The received and sent frames are displayed in hexadecimal code below the register tables.

The server doesn't stop instantaneously; the *timeout* determines the maximum time it could take.

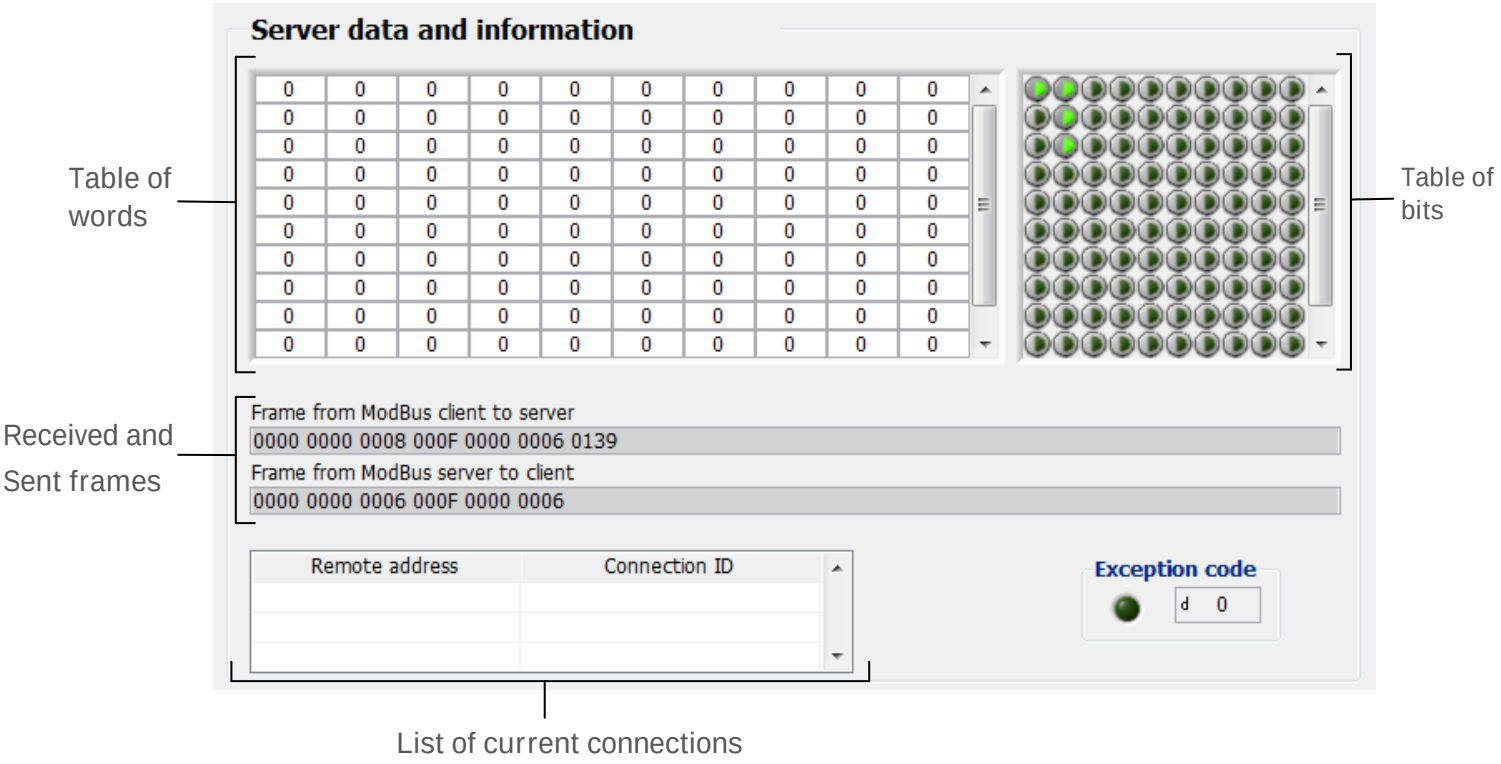


Figure 33: Communication test interface of the server

The *Exception cod* refers to ModBus protocol exceptions (cf. §7.2).

5. GModBus over TCP activation

After the download and installation of GModBus over TCP toolkit, an activation window will pop up at LabVIEW launching. Follow the steps of the add-on activation as shown in the picture below.

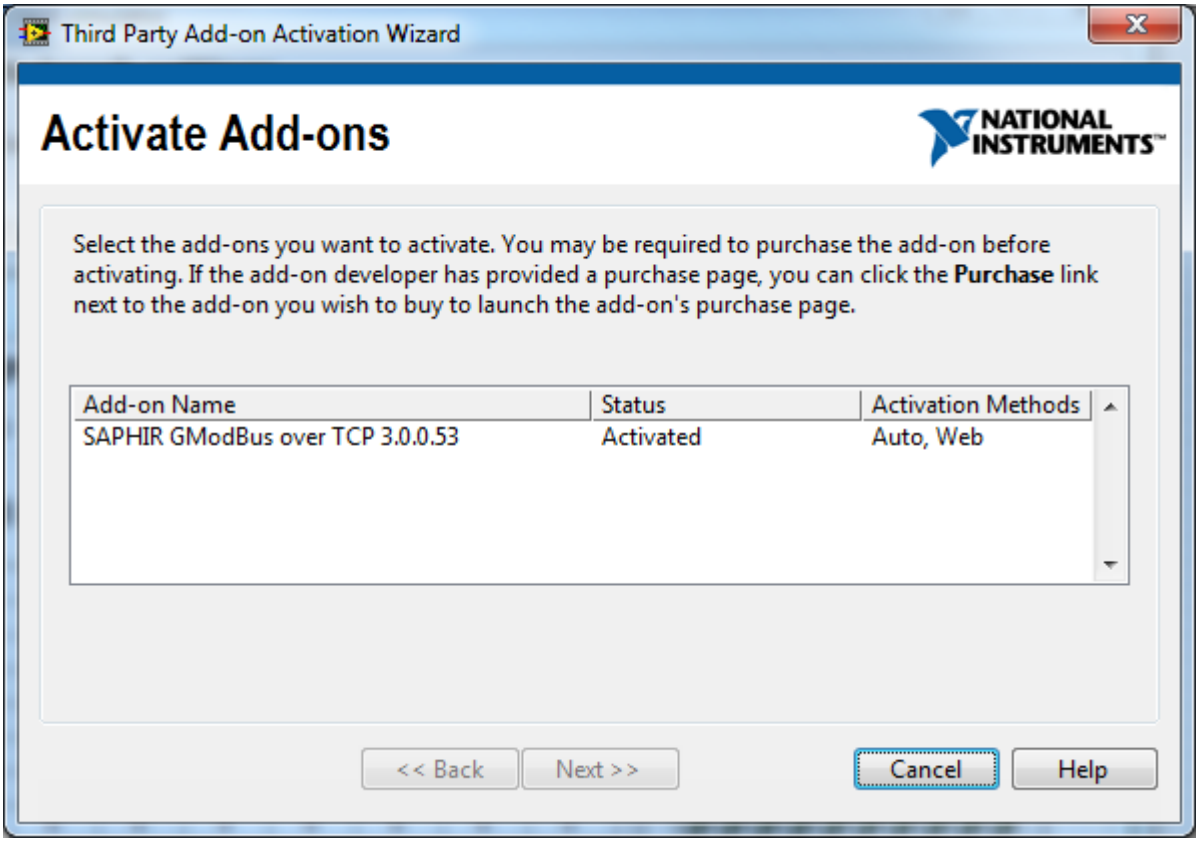


Figure 34: Third Party Add-on

You can try *"GModBus over TCP"* during 30 days. After this period toolkit's VIs will become broken. To activate the toolkit after this period, simply go to Help menu and select Activate Add-ons...

6. GModBus over TCP support

The "Online Support & Resources" menu opens the following SAPHIR community page: http://decibel.ni.com/content/groups/saphir-toolkit

On these pages, you can find documentation or start discussions with other users of the toolkit.

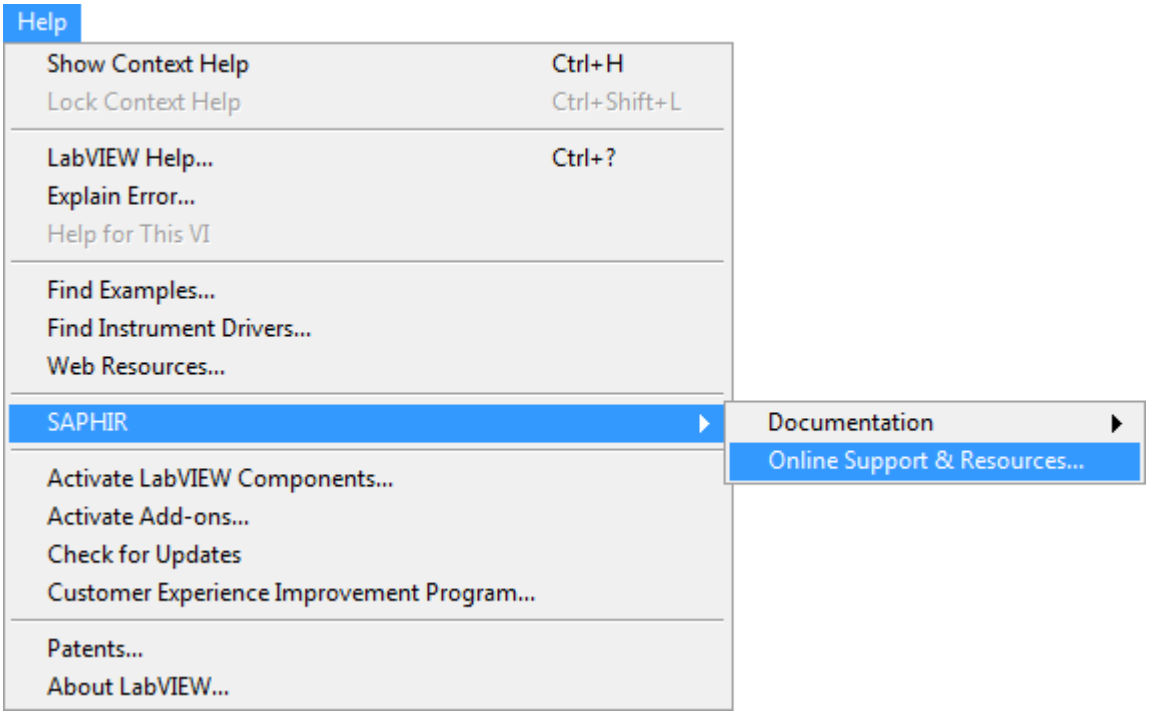


Figure 35: Online support

For direct contact with SAPHIR support team, send us an e-mail at support@saphir.fr.

7. GModBus over TCP driver errors

7.1 Specific errors

Following errors can be generate by *"GModBus over TCP"* functions

Error	Explanation
6101	TimeOut error.
6102	Unable to connect server.
6103	Connection rejected by the server.
6104	The function used to answer the request is not adapted to the request.

Figure 36: Specific errors

7.2 Exception codes

Following exception codes are specific to ModBus protocol.

Decimal Codes	Explanation
1	Not implemented function
2	Out of limits address
3	Out of limits data
4	Defective equipment
5	Acquit/release.
6	Busy equipment
7	Impossible to release
8	Memory error

Figure 37: Exception codes

When an exception code occurs a warning is fired on "error out" output.

8. Frequently asked questions

The list below collects the most common problems encountered during the implementation of *"GModBus over TCP"* driver:

Network setting:

The IP address keyboarded must not contains any non-significant zero (ex : 192.168.011.002 → 192.168.11.2).

The values returned by the server do not correspond to the expected values:

To return a float number (point) 32 or 64 bits, the server uses respectively 2 or 4 words. The Figure below shows the most common way to interpret 2 words to make a 32 bits float (number).

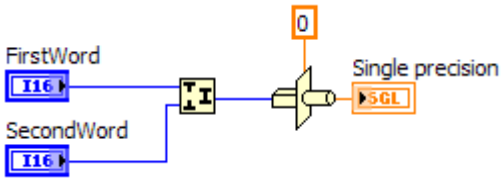
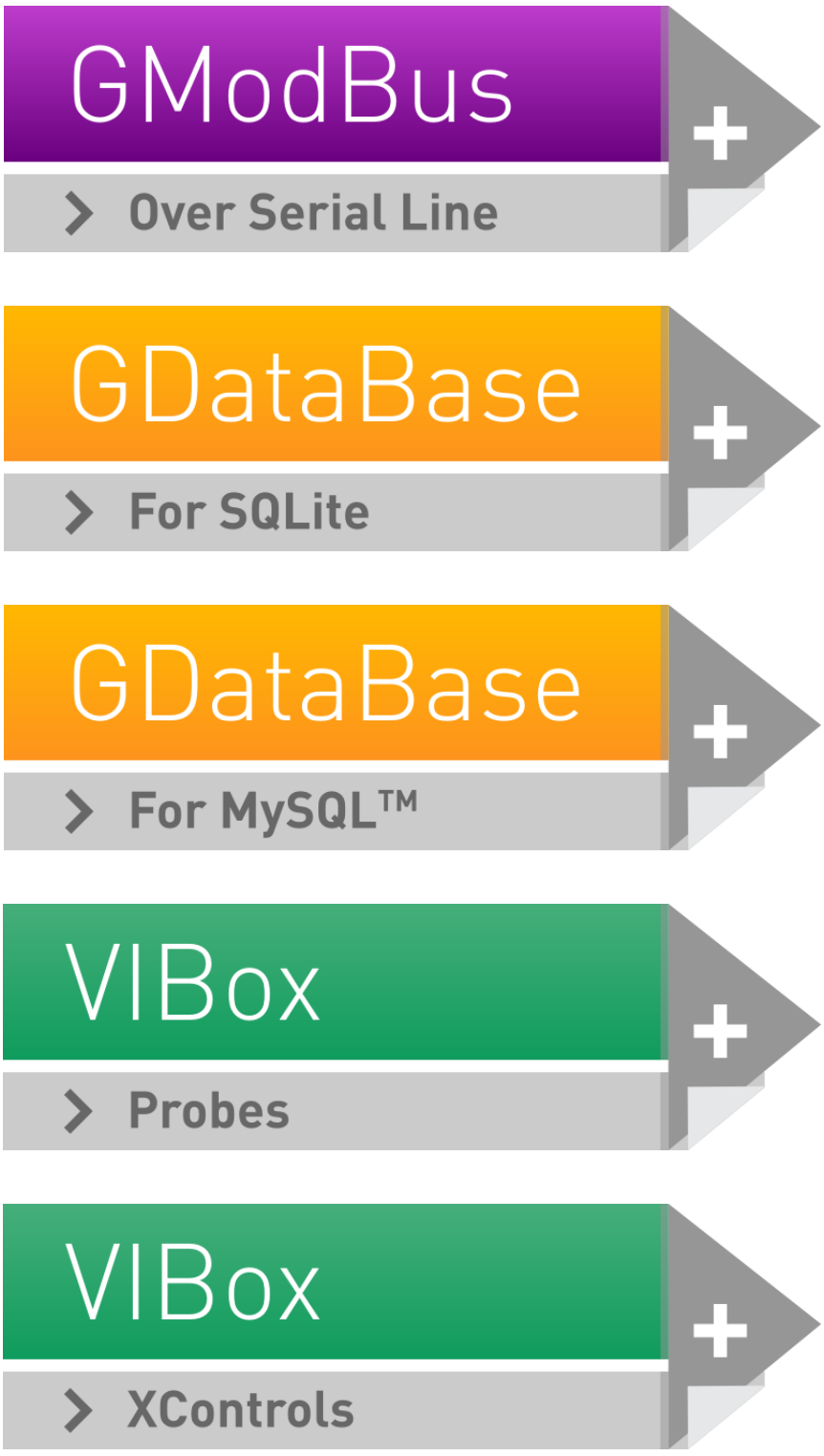


Figure 38: Interpretation of 2 words in 32 bits float

The way to interpret words can change according to the type of server. For more information refer to the manufacturer's documentation.

INDEX

Other add-ons that could be helpful



contact@saphir.fr
+33 (0)4 38 92 15 50
www.saphir.fr