# ModBus-Link

## User's Guide

ModBus-Linkmanual version 2.0 revised 000427 LKa

## CARDIAC

CARDIAC · Vipeveien 51 · N-3917 Porsgrunn, Norway·
Telephone: (+47) 35 93 06 00, Telefax: (+47) 35 93 06 66. E-mail: cardiac@cardiac.no. Web: http://www.cardiac.no .


## Important Warning

Because of the variety of uses of this product, and the impossibility of our anticipating every use, those responsible for using and applying this product must satisfy them that it is appropriate and acceptable for their application. That responsibility includes using this product in accordance with all sound engineering, programming and safety practices. The diagrams and layout examples shown in this manual are presented solely to illustrate the text of the manual. CARDIAC assumes no responsibility for actual uses based on illustrated uses or applications.

## License

You have a non-exclusive license to use the ModBus-Link software. You may use this product on one computer at a time. For backup purposes only, you may copy ModBus-Link software to other disks, which you must retain in your possession and control. You may not copy ModBus-Link software to any other computer by any means for any other purpose. The ModBus-Link software, the related documentation and their look and feel are copyrighted. You may not modify, disassemble or translate ModBus-Link software or related documentation without the written consent from CARDIAC.

## Warranty

CARDIAC warrants the ModBus-Link program disks to be free from defects for one year from the date of purchase. If, during this warranty period, you find that the ModBus-Link disks contain physical defects, return the defective item along with proof of purchase and a description of the defect. You will receive a replacement at no charge. CARDIAC makes no warranty or representation, express or implied, with respect to this software, its performance or its fitness for a particular purpose. As a result, the software is sold "as is" and you, the purchaser, assume the entire risk as to quality and performance. In no event shall CARDIAC be liable for direct or indirect, special, incidental or consequential damages resulting from any defect in the software or its documentation. Such damages include but are not limited to loss of profit, opportunity, programs or data. In no case will CARDIAC be liable for damages due to action or inaction of governments or quasi-governments, acts of riot, civil unrest, natural disasters or force majeure. This warranty excludes all other warranties, oral or written, express or implied.
The warranty shall be governed by and construed in accordance with Norwegian law. The parties accept Skien and Porsgrunn City Court as the proper legal venue for settlement of any dispute or controversy arising that can't be amicably resolved by the parties.

## Intended Audience

This manual assumes you are familiar with Windows software and basic Windows operations such as pointing and clicking, launching an application and moving files. Because ModBus-Link works only with LabVIEW software and ModBus hardware, familiarity with LabVIEW and ModBus controllers is assumed. If you have just purchased LabVIEW or ModBus controllers, you should familiarize yourself with those products before you attempt to install or use ModBus-Link.

# TABLE OF CONTENTS

# 1 INTRODUCTION

ModBus-Link allows National Instruments' LabVIEW software to communicate with ModBus slave and Master devices. You can read and write data to the devices and thereby monitor and control instruments, machines or processes. Because ModBus-Link works with LabVIEW, you can quickly create operating panels to control and monitor your devices. You can record and analyze data in real time or after the fact. You can communicate to a single controller or to a network of controllers.

## *1.1 Requirements*

You can use ModBus-Link on any machine that can run LabVIEW 5. The system requirements are the same as for LabVIEW 5.

You must have LabVIEW version 5.0 or later. You should be familiar with LabVIEW before attempting to install or use ModBus-Link.

You will also need additional serial cables to hook up to the PC`s serial interface (RS-232).

## *1.2 If You Need Help.*

If you need help CARDIAC can be reached by calling +47 35930600 or by sending an e-mail to cardiac@cardiac.no.

# 2 INSTALLATION

## *2.1 Hardware*

**Warning!**
When installing or removing interface cables, always power down the ModBus device and the PC. Otherwise, you may damage your equipment.

You connect your PC to ModBus devices with an interface cable. One end of the cable connects to a COM port of your PC. The other end connects to the ModBus port of ModBus Slave / Master.

The pinouts for cables are diagrammed in Section 6.

**NOTE!**
You must set the ModBus device's port to use the serial line configuration and slave address you desire. These values must match those used by the ModBus-Link software. You must set the framing method to RTU.

## *2.2 Software*

Before the program is installed, the CD or download file and the computer should be checked for virus. Some virus scanning programs may have problems during installation of this software. Any virus checking programs should be turned off before installation and back on afterwards.

Take a backup of the installation disks before use.

### 2.2.1 Windows

Run SETUP.EXE:
The installation program let you install the program, or cancel the installation. If you choose to follow the installation, please follow the instructions on the screen.

## *2.3 Configuration*

### 2.3.1 Modbus Master

The "Initialize ModBus.vi" allow you to configure serial communication between the PC and the ModBus devices. The configuration settings in the "Initialize ModBus.vi" must match those on the controller in order for ModBus-Link to provide error free communication.

**Slave Address** is the current device address.

**Timeout (ms)** sets the maximum amount of time that ModBus-Link will wait for the controller to respond to a query. If the timeout limit is exceeded, ModBus-Link will display the error: "Timeout for ModBus request". If the PC and the controller settings are not compatible, response timeouts will always occur.

**Number of Retries** sets the number the ModBus-Link will re-transmit a query to a controller.

**Serial port** selects the appropriate COM port.

**Data bits**, **stop bits**, **parity**, **Baud Rate**, and **buffer size** specify the transmission characteristics of each byte transmitted.

The ModBus-Link VI's will return **error** messages through the error clusters. These errors can be used in the application either to stop the execution or to warn the user of the error.

### 2.3.2 ModBus slave

The "CARDIAC ModBus Slave Init Comm.vi" allow you to configure serial communication between the Master and the ModBus Slave devices (your virtual Modbus device programmed in LabVIEW). The configuration settings in the "CARDIAC Modbus Slave Init Comm.vi" must match those on the Master in order for Modbus-Slave to provide error free communication.

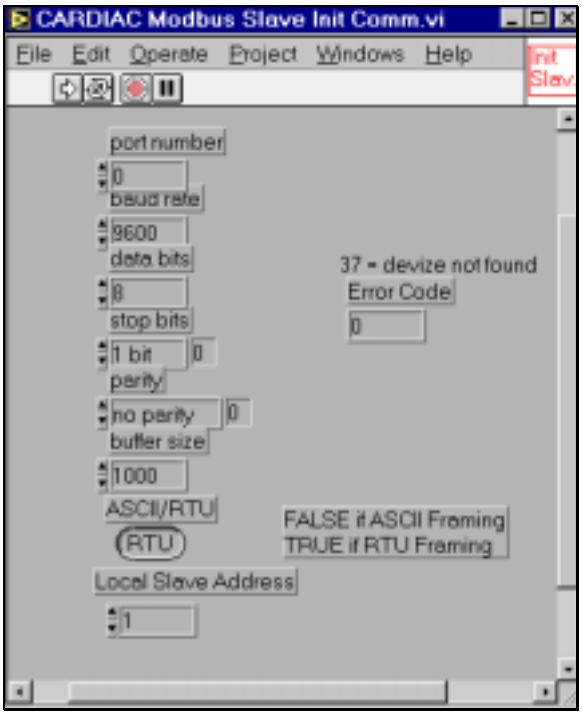



Figure 1 shows the CARDIAC Modbus Slave Init Comm.vi that always must be run before the main polling loop is started, or if any communication parameters are changed.

**Local Slave Address** is the current device address, your PC with your LabVIEW application.
**Serial port** selects the appropriate COM port.
**Data bits**, **stop bits**, **parity**, **Baud Rate**, and **buffer size** specify the transmission characteristics of each byte transmitted.
It is recommended to use the RTU mode, which responds to the Modbus binary framing protocol.

# 3 OVERVIEW

## *3.1 Modbus Master*

ModBus-Link allows National Instruments' LabVIEW software to communicate with ModBus devices. You can read and write data to the device and thereby monitor and control instruments, machines or processes. Because ModBus-Link works with LabVIEW, you can quickly create operating panels to control and monitor your devices. You can record and analyze data in real time or after the fact. You can communicate to a single controller or to a network of controllers.

Typical applications of ModBus-Link include operator panels, statistical process control, and coordination of controller activity and data collection.
This release: ModBus-Link contains a "driver", a LabVIEW Virtual Instrument (VI) library. The driver handles the low level network communications between the Windows and the ModBus device. The VI library provides LabVIEW icons that you connect in a LabVIEW block diagram.

## *3.2 Modbus Slave*

CARDIAC ModBus-Slave allows National Instruments' LabVIEW 5.x software to communicate with a Modbus Master. You can read and write data to the Modbus Master device and thereby monitor and control a virtual Modbus Slave Instrument (PC running LabVIEW 5.x).
Typical applications of ModBus-Slave are when you need to transfer your LabVIEW application to act like a virtual Modbus Slave instrument (device). The product is used to develop virtual (software) instruments that will be hooked up in a Modbus network. As a reference the software is used in equipment for measuring Multiphase flow offshore, for Multifluid International and Framo Engineering AS.
This release: ModBus-Slave contains a "driver", a LabVIEW Virtual Instrument (VI) library. The driver handles the low level network communications between the Windows and the ModBus device. The VI library provides LabVIEW icons that you connect in a LabVIEW block diagram. You only need to use three main VIs to set up and run the Modbus slave.

# 4 USING MODBUS-LINK

## *4.1 ModBus Master*

### 4.1.1 ModBus Demo

A demo program is included. This program read from one holding register and writes the output to a graph as well as to a new holding register.

### 4.1.2 ModBus-Link VIs

The ModBus-Link VIs are grouped into three categories:

Read: VIs that read from the ModBus devices.
Utility: VIs that is used by the read and write VIs. You can use them to override ModBus-Link's default error actions.
Write: VIs that writes to the ModBus devices.

Chapter 4 shows more detailed VI information.
The numbers before the names refer to the ModBus command.

#### 4.1.2.1 Read VIs

01 Read Coil Status.vi
02 Read Input Status.vi
03 Read Holding Registers.vi
04 Read Input Registers.vi
07 Read Exception Status.vi
11 Fetch Comm Event Ctr.vi
12 Fetch Comm Event Log.vi
17 Report Slave ID.vi
20 Read General Reference.vi
24 Read FIFO Queue.vi

Use these VIs to read information from the ModBus devices.

#### 4.1.2.2 Utility VIs

08 Diagonstic.vi
Calculate CRC.VI
Initialize ModBus.vi
Process ModBus command.vi
Serial Port, Flush.vi
Serial Port, Wait for Bytes.vi

Use these VIs to Initialize ModBus-Link and for diagnostic loop-back for the controller.

#### 4.1.2.3 Write VIs

05 Force Single Coil.vi

06 Preset single Register.vi
15 Force Multiple Coils.vi
16 Preset Multiple Regs.vi
21 Write General Reference.vi
22 Mask Write 4X Register.vi
23 Read/Write 4X Registers.vi

Use these VIs to write information to the ModBus devices.

## *4.2 ModBus slave*

### 4.2.1 CARDIAC ModBus-Slave VIs

The CARDIAC Modbus Slave software includes a set of 31 VIs, including an example of how to use the software in your application. The name of the example is “Example Slave Message Processor.vi”

The VIs included in the CARDIAC Modbus Slave 98 folder is shown in figure 2.

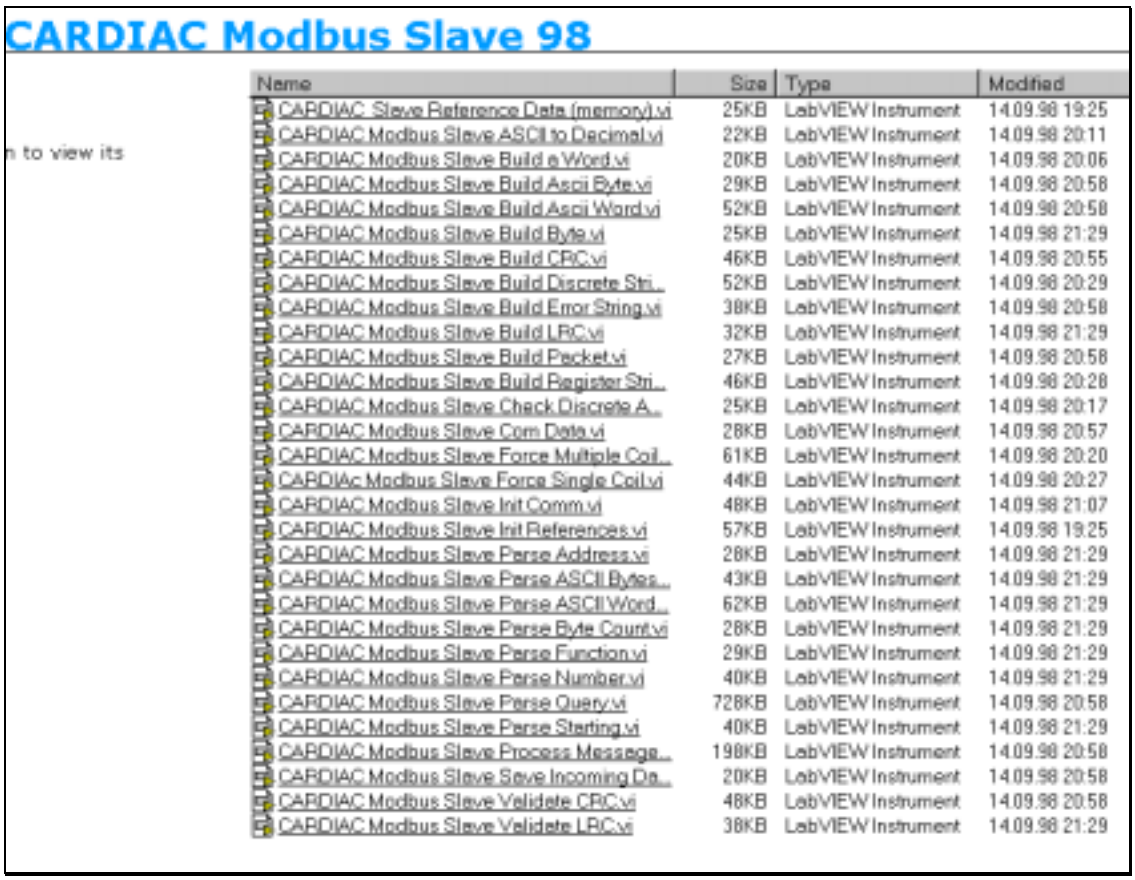


Figure 2. The VIs included in the CARDIAC Modbus Slave 98 folder, in addition the product are shipped with an example of how to use the software.

### 4.2.2 VIs the users need to know.

The only five VIs the user need to know are:

- Example Slave Message processor.vi
- CARDIAC Modbus Slave Init.vi
- CARDIAC Modbus Slave Init References.vi
- CARDIAC Modbus Slave Process Message.vi
- CARDIAC Slave Reference Data (memory).vi

The Slave uses all other VIs when needed, and should not be used by the software developer.
The following ModBus commands are supported in the Virtual Device:
Support for ModBus command 01, 02, 03, 04, 05, 06, 15 and 16. Other command can be added on request. The 08, Diagonstic.vi is not included.

### 4.2.3 Example Slave Message processor.vi

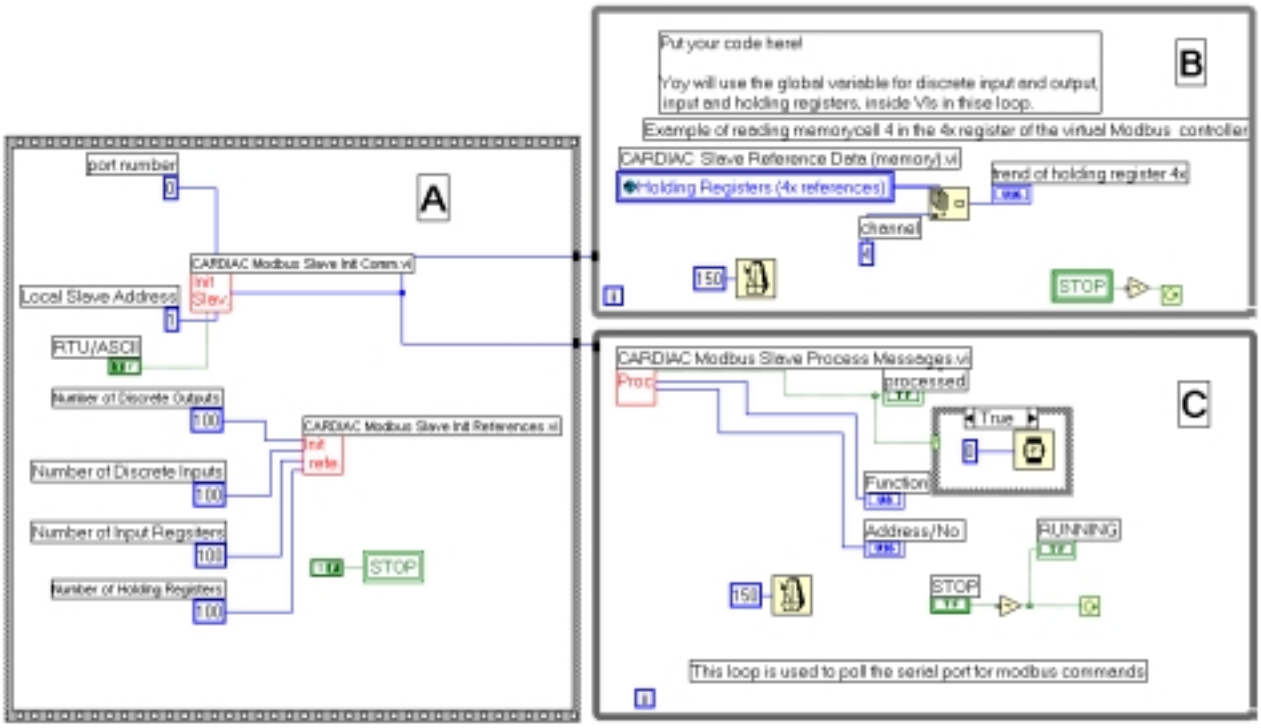


Figure 4, example of use of the Modbus Slave application.

The “Example Slave Message Processor.vi” give a simple example how to use the software as a ModBus Slave device.
The “A” sequence shows the “Modbus Slave Init Comm.vi”. This VI is used to set up all the communications parameters for the serial device, and the slave address and Modbus ASCII or RTU frame. In the example only the port number and the Local Slave Address are connected. In your real

application it may be a need also to add input for the other input terminal as well. In this example this values are used as default values.
In the “A”, you also need to set up the CARDIAC Modbus Slave Init References.vi. This VI is used to set up global memory variable for all the discrete input and output, and for the controllers input and holding registers. This registers and coils are the memory cells, the Modbus Master used to access the values in the Modbus Slave device. All the values are 16 bit registers, and if you want to build float values, it is normal to scale the registers with a constant value, for example 10, 100 or 1000. If you want to use sgl you can add two 16 bit registers and use the type casting tool in LabVIEW. This method is not common, but possible.
The loop “C”, is the main loop that is used for the Modbus Slave to check if there are messages from the Modbus Master, which the ModBus Link device (Slave number) shall respond to. The wait 150 msec is the polling rate for new data from the Master. If the master need faster response, then you may need to set a shorter polling rate. If you recognize any problem with the polling rate, please start this VI as an own thread in LabVIEW.
The actual read and write between the Mater and the Slave is done in the loop “C”, but when your software need to update or read data from the registers, this is done in the loop “B”.
All the registers use the LabVIEW global variable, “CARDIAC Slave Reference Data (memory)”. This global variable contains the input registers, holding registers and the coils (Discrete references) used by the Slave. This entire global variable will be written or read by the Master, and when you update variables inside your application.

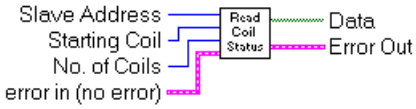
# 5 VI REFERENCE

## *5.1 ModBus Master*

Each ModBus-Link VI is described with a graphic summary of inputs (left) and outputs (right), followed by a description and usage notes. The graphic shows the LabVIEW icon and connector pane.

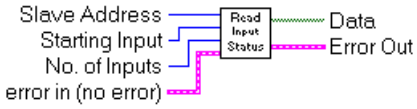
### 5.1.1 Read Vis

#### 5.1.1.1 Read Coil Status.vi



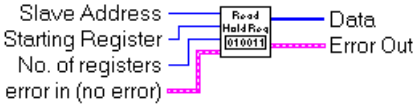

Reads the ON/OFF status of discrete coils (0X references, coils) in the slave. Coils are addressed starting at zero. The coil status in the response message is packed as one coil per bit of the data field. The LSB of the first data byte contains the coil addressed in the query. ModBus command 01.

#### 5.1.1.2 Read Input Status.vi



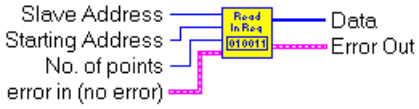

Reads the ON/OFF status of discrete inputs (1X references, inputs) in the slave. This VI is similar as the Read Coil Status.vi but for input instead of output from the slave. ModBus command 02.

#### 5.1.1.3 Read Holding Registers.vi



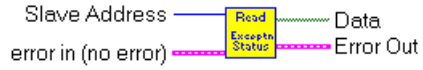

Reads the binary contents of holding registers (4X references) in the slave. The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. ModBus command 03.

#### 5.1.1.4 Read Input Registers.vi



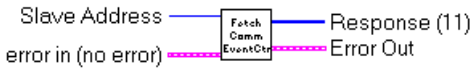

Reads the binary contents of input registers (4X references) in the slave. This VI is similar to the Read Holding Register.vi. ModBus command 04.

#### 5.1.1.5 Read Exception Status.vi



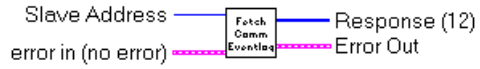

Reads the contents of eight Exception Status coils within the slave controller.. Certain coils have predefined assignments in the various controllers. Other coils can be programmed by the user to hold information about the controller's status. The function provides a simple method for accessing this information. ModBus command 07

#### 5.1.1.6 Fetch Comm Event Ctr.vi



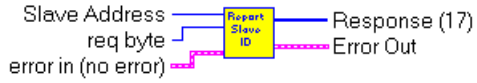

Returns a status word and an event count from the slave's communications event counter. By fetching the current count before and after a series of messages, a master can determine whether the messages were handled normally by the slave. The controller's event counter is incremented once for each successful message completion. The normal response contains a two-byte event count, and a two-byte event count. ModBus command 11.

#### 5.1.1.7 Fetch Comm Event Log.vi



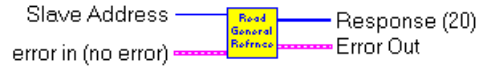

Returns a status word, event count and a field of event bytes from the slave. The status word and event count are identical to that returned by the Fetch Comm Event Ctr.vi. The message counter contains the quantity of messages processed by the slave since its last restart, clear counters operation, or power-up. The normal response contains a two-byte event count, a two-byte event count, a two-byte message count field, and a field containing 0-64 bytes of events. ModBus command 12.

#### 5.1.1.8 Report Slave ID.vi



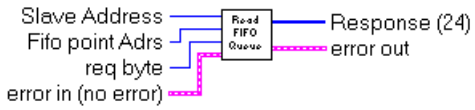

Returns a description of the type of controller present at the slave address, the current status of the slave Run indicator, and other information specific to the slave device.. The data contents in the response are specific to each type of controller. ModBus command 17

#### 5.1.1.9 Read General Reference.vi




Returns the contents of registers in Extended Memory file (6xxxx) references. The function can read multiple groups of references. The groups can be separate, but the references within each group must be sequential. ModBus command 20.

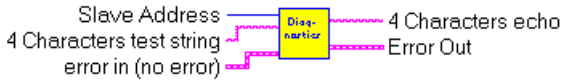
#### 5.1.1.10 Read FIFO Queue.vi




Read the contents of a First-In-First-Out queue of 4xxxx registers. The function returns a count of the registers in the queue, followed by the queued data. Up to 32 registers can be read: The count plus up to 31 queued data registers. ModBus command 24.

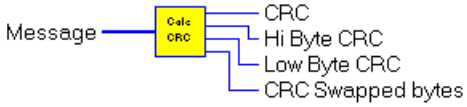
### 5.1.2 Utility

#### 5.1.2.1 Diagonstic.vi



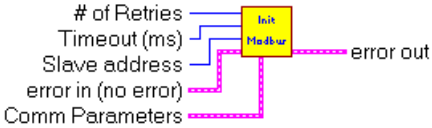

Test the Diagnostic loop-back for the controller. Use a four-character string. The VI will response a echo of the string if the communication is OK. If you use less than four character, the VI will stop after # of retries with a timeout error. ModBus command 08.

#### 5.1.2.2 Calculate CRC.VI



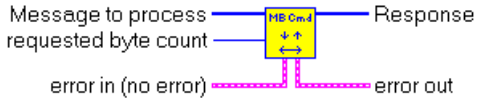

Calculate 16 bit CRC of the message using CRC lookup table.

#### 5.1.2.3 Initialize ModBus.vi



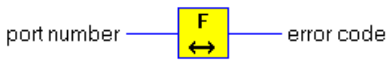

Set communication parameters and other necesarry parameters.

#### 5.1.2.4 Process ModBUS command.vi



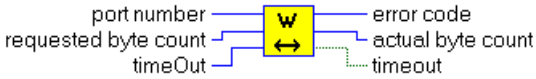

Process Modbus command to the ModBus device.

#### 5.1.2.5 Serial Port, Flush.vi




Empties the buffer at the specified port.

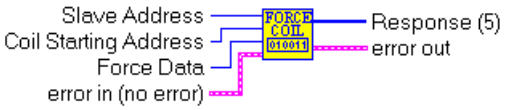
#### 5.1.2.6 Serial Port, Wait for Bytes.vi




Wait for the requested number of bytes at the specified port. Quits when the number is reached or when timeout.

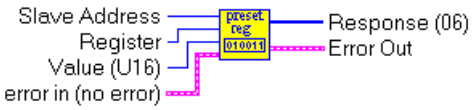
### 5.1.3 Write Vis

#### 5.1.3.1 Force Single Coil.vi



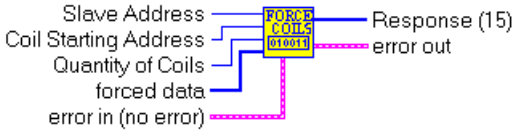

Force a single coil, 0X reference to either ON or OFF. When broadcast, the function forces the same coil reference in all attached slaves. The function will override the controller's memory protect state and the coil's disable state. The forced state will remain valid until the controller's logic next solves the coil. The coil will remain forced if it is not programmed in the controller's logic. The normal response is an echo of the qurry, returned after the coil state has been forced. ModBus command 05.

#### 5.1.3.2 Preset single Register.vi



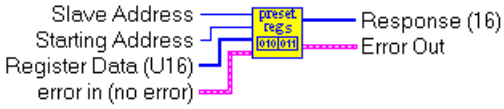

Preset a value into a single holding register (4X reference). When broadcast, the function presets the same register reference in all attached slaves. The function will override the controller's memory protect state. The preset value will remain valid until the controller's logic next solves the registers contents. The registers value will remain if it is not programmed in the controller's logic. The normal response is an echo of the query, returned after the registers contents has been preset. ModBus command 06.

#### 5.1.3.3 Force Multiple Coils.vi



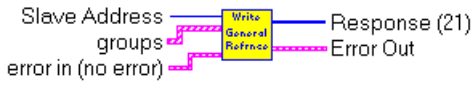

Forces each coil (0X reference) in a sequence of coils to either ON or OFF. When broadcast, the function forces the same coil reference in all attached slaves. The function will override the controller's memory protect state and a coils disable state. The forced state will remain valid until the controller's logic next solves the coil. Coils will remain forced if it is not programmed in the controller's logic. The normal response returns the slave address, function code, starting address, and the quantity of coils forced. ModBus command 15.

### 5.1.3.4 Preset Multiple Regs.vi



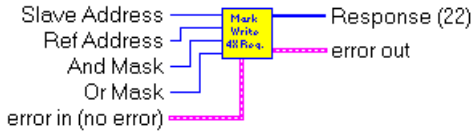

Preset values into a sequence of holding register (4X references). When broadcast, the function presets the same register reference n all attached slaves. The function will override the controller's memory protect state. The preset values will remain valid in the registers until the controller's logic next solves the registers contents. The registers values will remain if they are not programmed in the controller's logic The normal response returns the slave address, function code, starting address, and the quantity of registers forced. ModBus command 16.

### 5.1.3.5 Write General Reference.vi



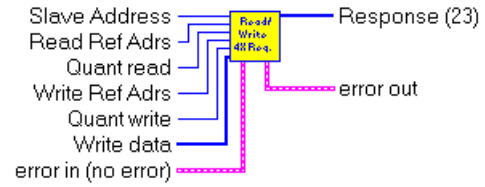

Writes the contents of registers in Extended Memory file (6xxxx) references. The function can write multiple groups of references. The groups can be separate, but the references within each group must be sequential. The normal response ia an echo of the query. ModBus command 21.

### 5.1.3.6 Mask Write 4X Register.vi




Modifies the contents of a specified 4xxxx register using a combination of an AND mask and OR mask, and the registers current contents. The function can be used to set or clear individual bits in the register. The normal response ia an echo of the query. The rsponse is written after the register has been written. ModBus command 22.

### 5.1.3.7 Read/Write 4X Registers.vi




Performs a combination of one read and one write operation in a single ModBus transaction. The function can write new contents to a group of 4xxxx registers and then return the contents of another group of 4xxxx registers. Function not supported in all controllers. The query specifies the starting address and quantity of registers of the group to be read. I also specifies the starting address, quantity of registers, and data for the group to be written. The normal response contains the data from the group that were read. ModBus command 23.

## *5.2 ModBus-Slave*

### 5.2.1 CARDIAC Modbus Slave Init.vi

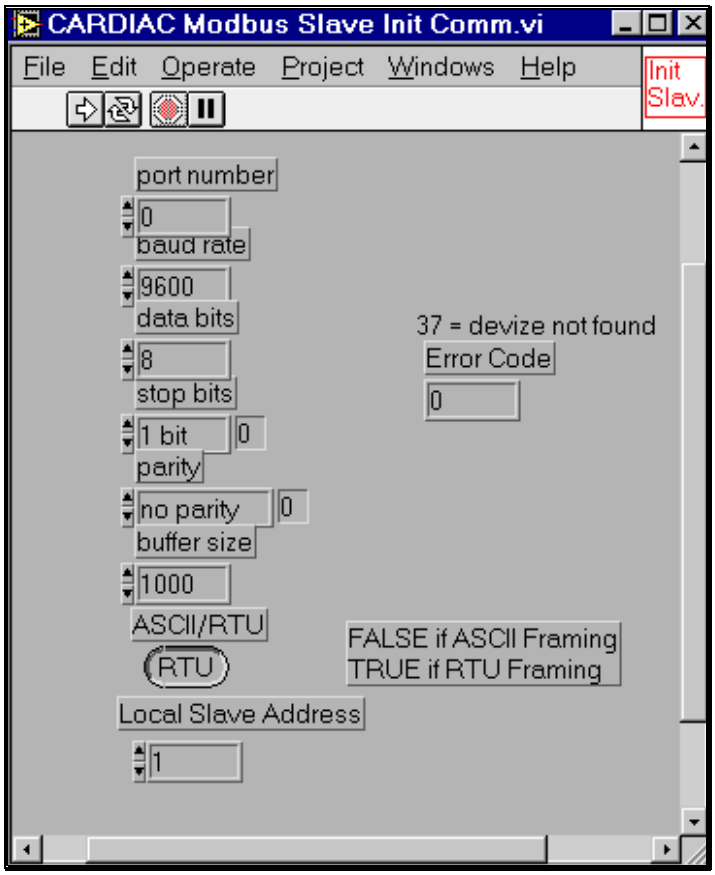
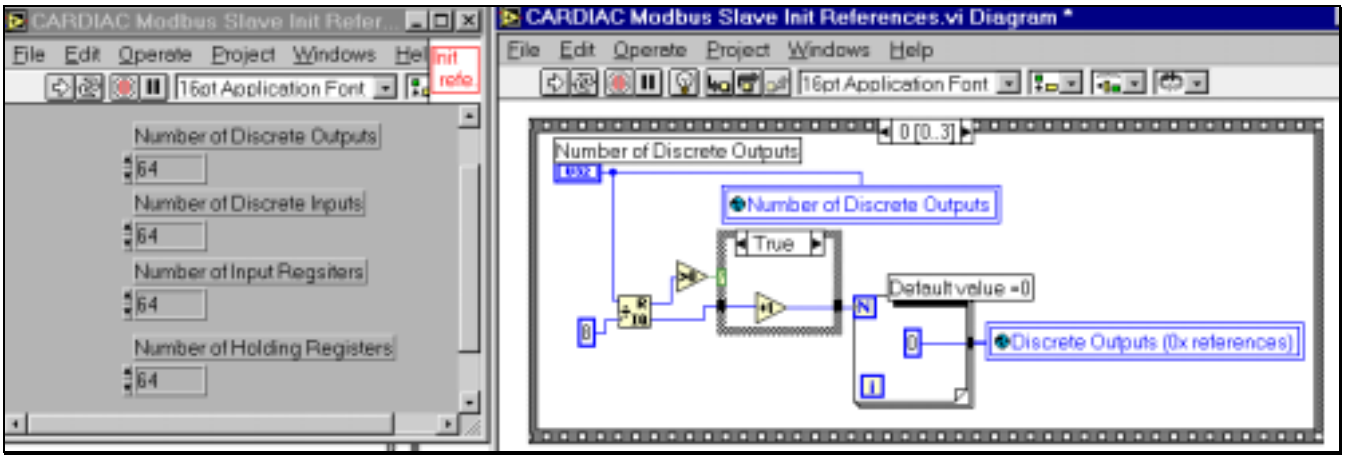


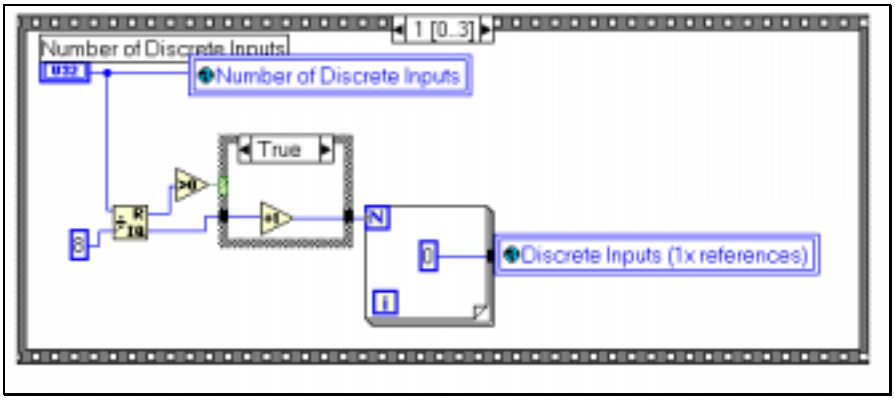

Figure 5, Init setup

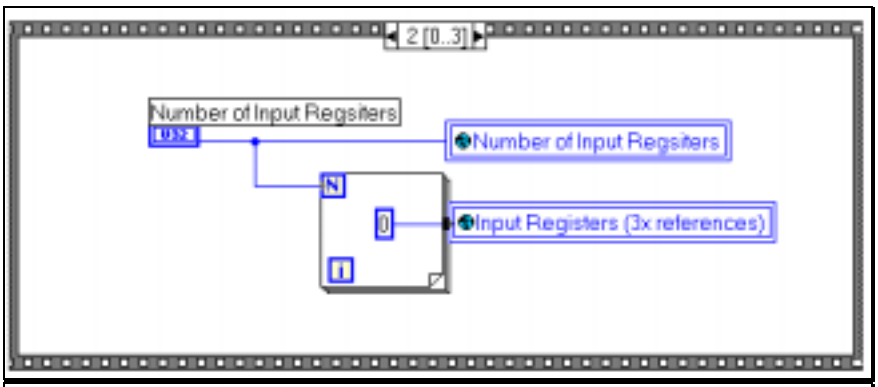
The VI is used to set up the communication parameters between the Master and the Slave.

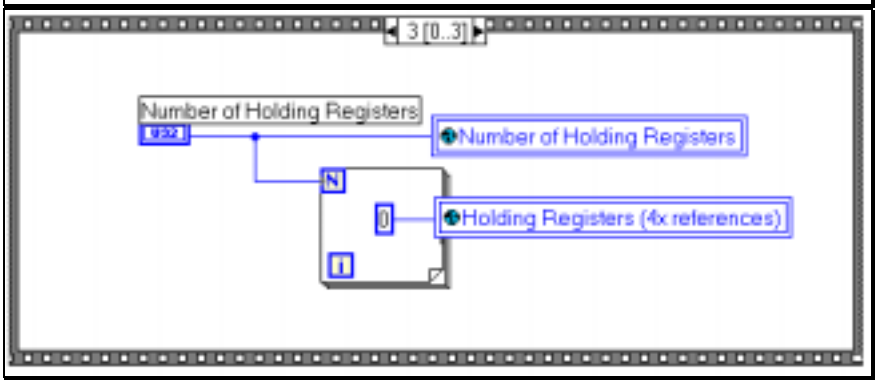
### 5.2.2 CARDIAC ModBus Slave Init References.vi

This VI us used to set up the number of 0x, 1x, 3x and 4x registers. All the values is set to default value as zero. Your main loop "B" and/or the Modbus Master can change values in the registers. You must at least set up the same number of registers as the Modbus Master asks for. The registers as started on index 0, and that will in normal condition reference to the first register or coil in the Master.

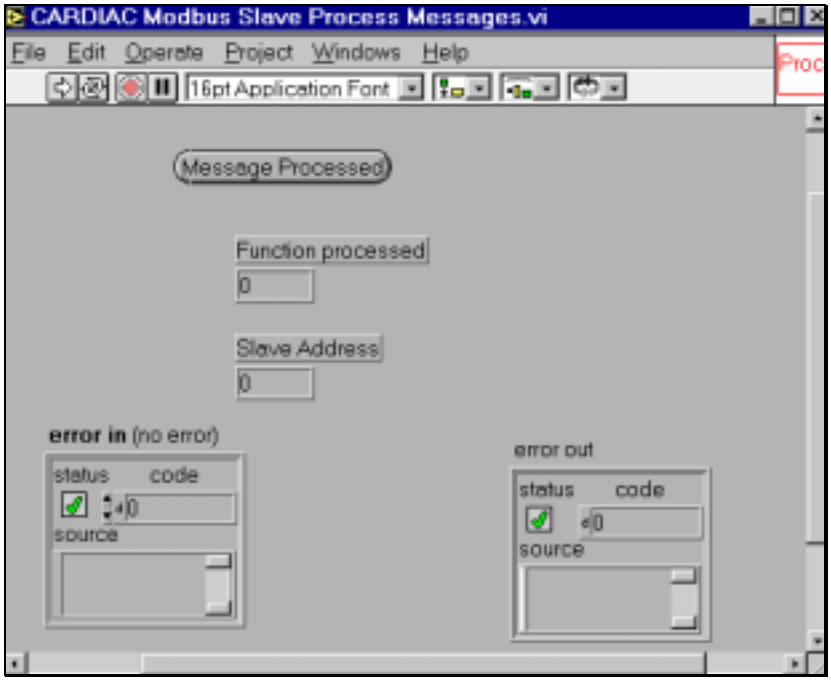

CARDIAC Modbus Slave Init Refer...
File Edit Operate Project Windows Hel
16pt Application Font
Number of Discrete Outputs
64
Number of Discrete Inputs
64
Number of Input Registers
64
Number of Holding Registers
64
CARDIAC Modbus Slave Init References.vi Diagram *
File Edit Operate Project Windows Help
16pt Application Font
0 [0..3]
Number of Discrete Outputs
Number of Discrete Outputs
True
Default value =0
Discrete Outputs (0x references)


1 [0..3]
Number of Discrete Inputs
Number of Discrete Inputs
True
Discrete Inputs (1x references)


2 [0..3]
Number of Input Registers
Number of Input Registers
Input Registers (3x references)

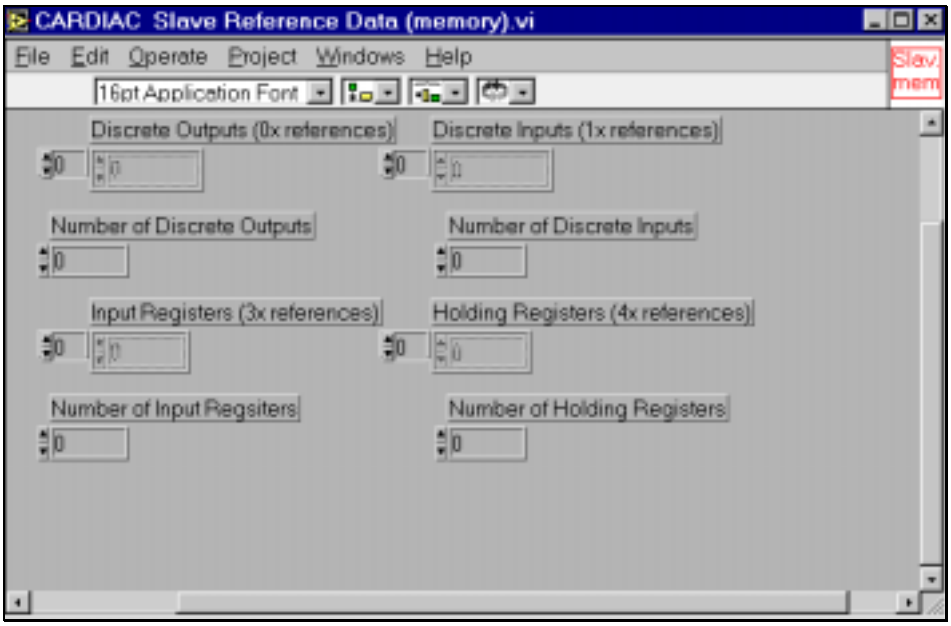

3 [0..3]
Number of Holding Registers
Number of Holding Registers
Holding Registers (4x references)

### 5.2.3 CARDIAC Modbus Slave Process Message.vi




This VI needs to be run all the time, because it polls the serial Modbus interface for Modbus Master request. If the VI is not running, the Master will not get any answer from the slave device. If the delay of the loop is to long, the Master can get timeout from the slave. The frequency of the loop needs to be as fast as the Master poll rate.

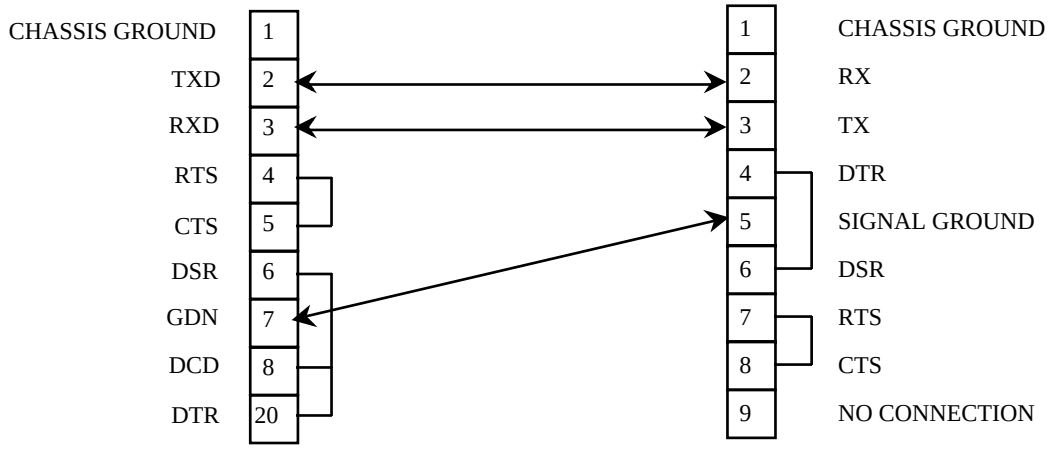
### 5.2.4 CARDIAC Slave Reference Data (memory).vi



The global is used as a virtual memory bank for the Modbus Slave device. These variables are used by your program to read and write information.

# 6 CABLE PINOUTS

## *6.1 PC DB-25 to ModBus Device*




## *6.2 PC DB-9 to ModBus Device*