Connecting UniOP Using Generic Modbus RTU

This Technical Note contains the information needed to connect UniOP to control devices using the Modbus RTU standard protocol over a serial communication line.

Contents

1. Introduction 1
2. Setting up UniOP for Communication 1
  2.1 Controller Setup 1
  2.2 Node Override 3
3. Modbus RTU Data Types 5
Appendix A. Communication Error Codes 6
Appendix B. Implementation Details 7

1. Introduction

The Modbus RTU generic driver is included in the Designer file UPLC166.DLL.
UniOP operator panels can be connected to a Modbus network as the network master using this generic driver.

2. Setting up UniOP for Communication

To create a UniOP application for a generic Modbus application, select the driver “Modbus RTU” from the list of available communication drivers in the Change Controller Driver… dialog box.

2.1 Controller Setup

Figure 1 below shows the "Controller Setup..." dialog box for the Modbus RTU protocol.
The Slave ID address is the address of the Modbus device on the network.

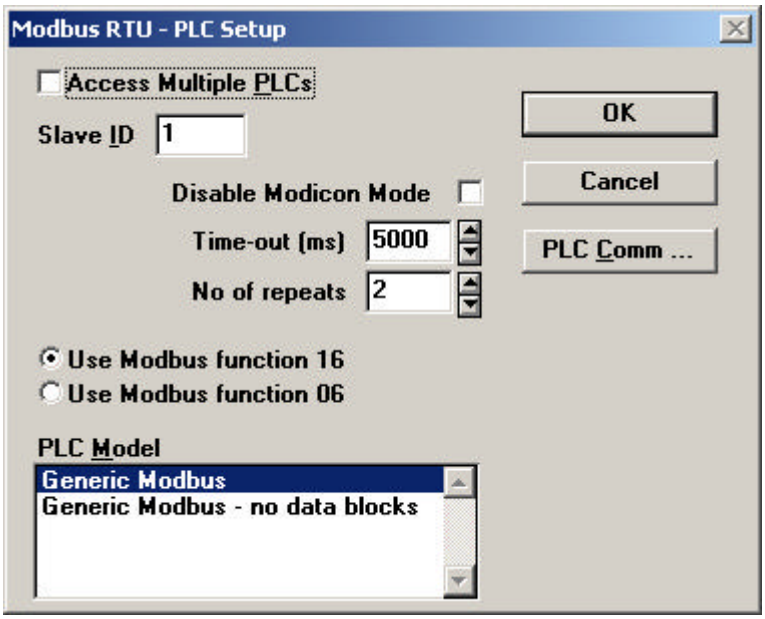

Figure 1 - Controller Setup dialog box

The "**Disable Modicon Mode**" checkbox defines if the addressing mode should follow the original Modicon standard (first address is 1) or a more generic approach where the address for the variables starts from index zero.
If Modicon Mode is enabled, the valid range for each data type is: 1…65536.
If Modicon Mode is disabled, the valid range for each data type is: 0…65535.
Please note that the permissible addressing range does not mean that the controller connected to the panel is actually containing that amount of variables.

The "**No of repeats**" parameter defines the number of times a certain message will be sent to the controller before reporting the communication error status.
A value of 1 for the parameter "No of repeats" means that the panel will eventually report the communication error status if the response to the first request packet is not correct.

The "**Time-out"** parameter defines the time inserted by the protocol between two retries of the same message. It is expressed in milliseconds.

Two radio buttons are available to specify what Modbus Function will be used for write operations to the Holding Registers data type. The user can select between the function 06 (preset single register) and function 16 (preset multiple registers).
If the Modbus function 06 is selected, the protocol will always use function 06 to write to the controller, even to write multiple consecutive registers.
If the Modbus function 16 is selected, the protocol will always use function 16 to write to the controller, even for a single register write request and the "Byte Count" parameter of the query is set to two. Using function 16 may result in higher communication performance
By default the Modbus Function 16 is used and should be disabled only for those controller that to dot support it.

The "**Generic Modbus – no data blocks**" model forces the creation of Designer projects where the panel will make only communications requests including only one register. For each Function Code which supports multiple items access, the "No. of Points" parameter of the Modbus request frame is always fixed to one.

This option will result in lower communication performance and should be used only when connecting to controllers that only can exchange one register at a time.

The protocol allows the connection to multiple Modbus slave devices connected to one UniOP panel. In that case, the "**Access Multiple PLCs**" option must be used.
When Access Multiple PLCs is checked, the dialog box will change and will appear as shown in Figure 2 below.

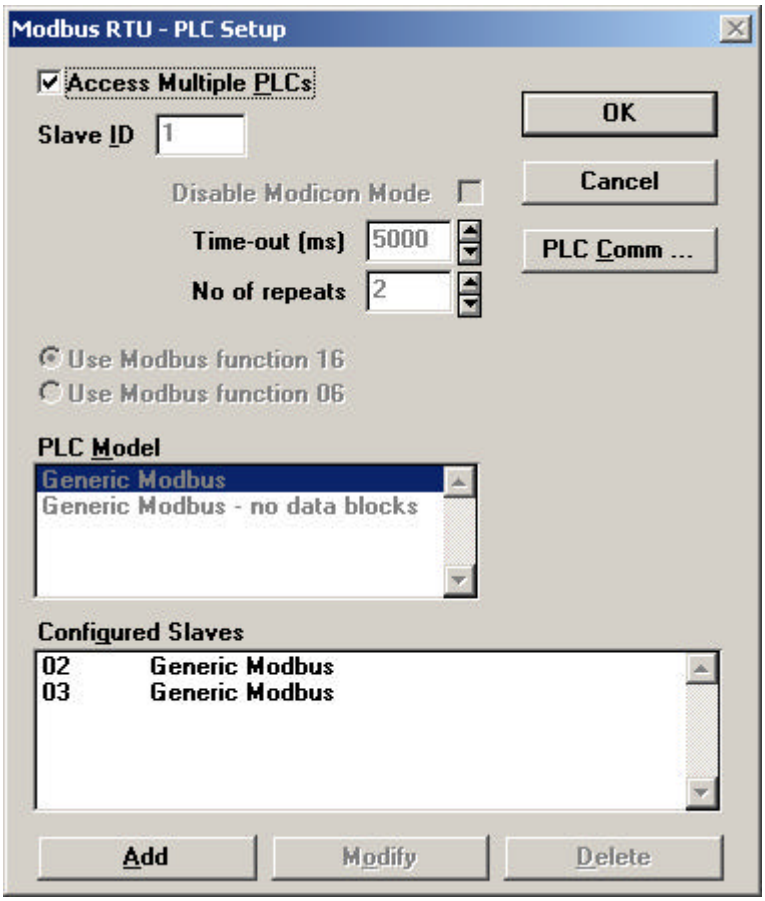

Figure 2 - Controller Setup Dialog Box for multiple controllers

Press the "Add" button to add new Modbus devices to the network and enter their address.

A typical situation in which UniOP is connected to several slave devices and a project page contains variables coming from all the partners, may become critical from the point of view of the data refresh time, when one or more of the slaves is missing or not responding.
A proper combination of the parameters "Time-out" and "No of repeats" allows to fine tune the application. The Modbus master, in fact, can be configured to skip very quickly over the nodes that are not responding.

2.2 Node Override

The Slave ID parameter you have entered in "Controller Setup" can be changed at run-time using the special Data Type called "Node Override".

Note: *Node Override requires the Modbus RTU driver version 5.07 – 4.04 or above and firmware version V4.50S or above*

Figure 3 shows the Modbus RTU Define Field dialog box when the special "Node Override" data type is selected.
Node Override is an internal UniOP variable that makes possible changing the settings for the Slave ID parameter specified at programming time in Controller Setup.
Node Override can be placed in a page and the Slave ID can be changed with a simple data entry operation over this variable.

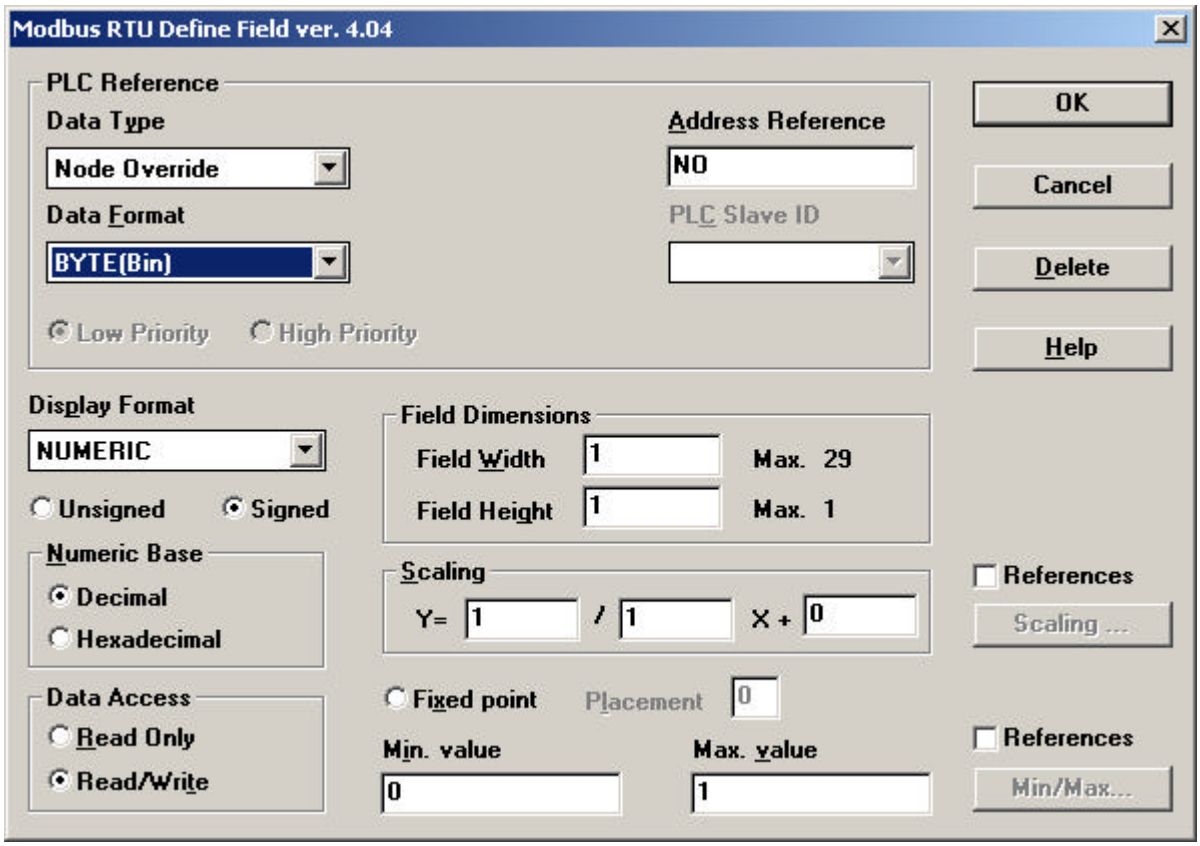

Figure 3 – Node Override

Changes to "Node Override" will require a UniOP power cycle to be effective.

When the "Controller Setup" is configured for "Access to Multiple Controller" there is one "Node Override" variable for each of the nodes included in the configuration. In the case of Multiple Controllers the dialog changes as shown in Figure 4.
The PLC Slave ID list box will define which Slave ID will be affected.

Note: *When "Access Multiple Controllers" is enabled, UniOP can store* ***up to 8*** *different Node Override variables; that is, if Controller Setup includes more than 8 slaves, only 8 of them can be re-mapped using the Node Override option.*

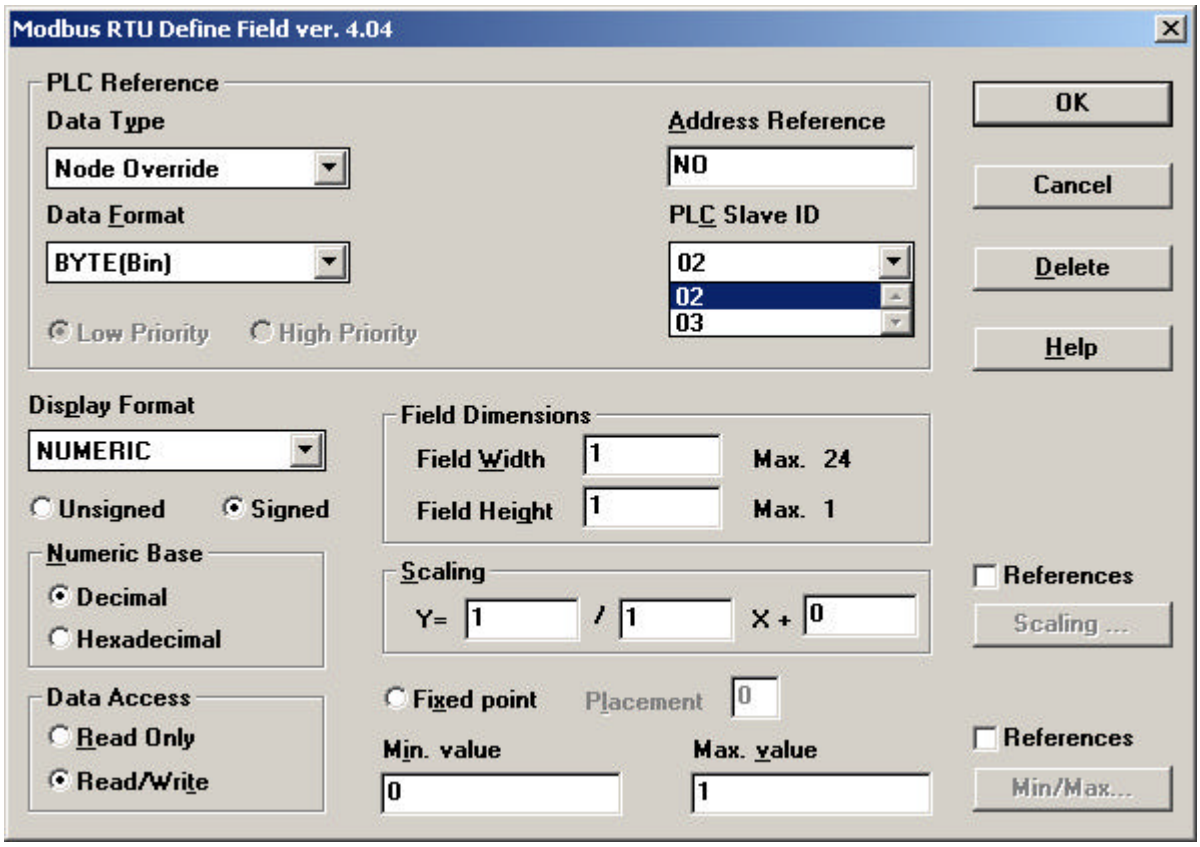

Figure 4 – Node Override for Multiple Controller access

3. Modbus RTU Data Types

The following standard Modbus data types can be accessed using the UniOP implementation of the Modbus RTU protocol:

Data type	Access mode
Input bits	read only
Output coils	read/write
Holding registers	read/write
Input registers	read only

Up to 65535 variables of each data type can be addressed.

Note that the address values actually used in the Modbus packet frame are: “Reference Address – 1”. When the ‘Disable Modicon Mode’ option is checked in Controller Setup, the address used in the Modbus frames on the serial line are the same as the reference entered in the Define Field dialog box.

Appendix A. Communication Error Codes

Current communication status is displayed in the System Menu of the UniOP.
A message and a numeric error code describe the error status.
The message reports the current communication status. The number shows the code of the current communication error or, if the communication is correct, the code of the last error encountered. When the error code 0 is shown, it means there have been no communication errors since this system start-up.

Code	Description	Notes
00	**No error**	There are no communication errors and there have been no errors since start-up.
04	**Negative ACK**	NAK returned by the controller
05	**Time out (receiving)**	No response received from the controller within the timeout period.
06	**Response error**	Error code field (BCC) received from controller is wrong.
07	**General communication error**	General un-known communication error

Appendix B. Implementation Details

This Modbus RTU implementation supports only a subset of the standard Modbus Function Codes. The supported Function Codes are listed in the table below.

Code	Function	Description
01	Read Coil Status	Reads multiple bits
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave
03	Read Holding Registers	Read multiple Registers
04	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave
05	Force Single Coil	Forces a single Coil to either ON or OFF
06	Preset Single Register	Presets a value in a Register
16	Preset Multiple Registers	Presets value in multiple Registers

Communication speed with controllers is supported up to 57600 baud.

Floating point data format is compliant to the IEEE standard.

Note: *Baud rate greater than 38400 are only supported starting for firmware version V4.22. Speed is anyway still limited to 9600 for UniOP models without PC/Printer port.*