UnityPro Communication Driver

This document has the specific information related to the driver configuration. For a generic explanation on Devices, Channels, Nodes and Points configuration, please refer to the reference guide.

Contents

Section 1 – Summary Information 2
Section 2 – Channel Configuration 3
Protocol Options 3
Settings 3
Section 3 – Node Configuration 4
Station Configuration 4
Section 4 – Point Configuration 5
Section 5 – Troubleshoot 5
Revision History 6

Section 1 – Summary Information

Communication Driver Name: UnityPro

Current Version: 1.1

Implementation DLL: T.ProtocolDriver.UnityPro.dll

Protocol: MODBUS RTU, ASCII and TCP

Interface: TCP/IP and Serial

Description: Modbus driver implements communication with PLC and IO devices compatibles with Modbus Open Standard protocol. It operates as a Master on TCP/IP or serial networks. The communications blocks are dynamically created according the pooling cycle defined on the AccessType for each Device Point.

PLC types supported: PLC's Quantum, Momentum M340 and M580 - Schneider-Electric

Communication block size: user configurable, default is 250

Protocol Options: Message Format (ASCII, RTU or RTU TCP), use block write command (0x0F or 0x10) or single write command (0x05 or 0x06) for single writings

Multi-threading: user configurable, default is five threads to each network node

Max number of nodes: user defined

PC Hardware requirements: Standard PC Ethernet interface board, RS485 or RS232 port

Supported Operands:

Operand	Read	Write	Data Type	Address size
0 – %Q or %M - Coils	✓	✓	Bit	1 bit
1 – %I - Input Status	✓	-	Bit	1 bit
3 – %IW - Input Registers	✓	-	Word	2 bytes
4 – %MW - Holding Registers	✓	✓	Word	2 bytes

Table 1

Section 2 – Channel Configuration

Protocol Options

BlockSize: Defines the maximum amount of items per group, the default value is **250**.

If the communication points are configured in sequence and the BlockSize equals to 250, the driver can create the internal groups with 125 Registers or 2000 Coils.

Encoding: Determines how information will be packed into the message fields and decoded. The options are:

- **RTU**: Remote Terminal Unit mode, where each 8–bit byte in a message contains two 4–bit hexadecimal characters
- **ASCII**: The message is encoded in ASCII mode, where each 8–bit byte in a message is sent as two ASCII characters
- **RTU TCP**: The default transmission mode when the message is carried on a MODBUS TCP/IP network. It contains information to allow the recipient to recognize message boundaries even if the message has been split into multiple packets

SingleWrite: Indicates the driver behavior for the writings with only one item:

- **Use block write** : The driver uses the 0x0F command for Coils, or the 0x10 command for Holding Registers
- **Use single write:** The driver uses the 0x05 command for Coils, or the 0x06 command for Holding Registers

Settings

Serial and MultiSerial channels:

- Default configuration for ASCII mode :
  DataBits: 7
  StopBits: 1 if parity is used, 2 if no parity

- Default configuration for RTU mode :
  DataBits: 8
  StopBits: 1 if parity is used, 2 if no parity

  Set the other fields according to your Serial or MultiSerial port configuration

TCP/IP channels:

- **NodeConnections**: Defines the maximum number of parallel requests that will be sent to each node (asynchronous communication)

Section 3 – Node Configuration

Station Configuration

SlaveId: Set this field with the address of the slave device in the network. They can be addressed from 1 to 247 for serial nodes, or 0 to 255 for TCP/IP nodes. The address 0 is used for the broadcast.

Serial channels:

- Station syntax: <SlaveId>

  Ex: 1

MultiSerial channels:

- Station syntax: <Com Port> ; <SlaveId>

  Where: <Com Port> = the serial port number

- Ex: com1 ; 1

TCP/IP channels:

- Station syntax: <IP address> ; <Port number> ; <SlaveId>

  Where : <IP address> = IP address of the slave device in the modbus network

  < Port number > = TCP port where the slave device is listening (default is 502)

  Ex: 192.168.1.101 ; 502 ; 1

Section 4 – Point Configuration

The syntax for the UnityPro communication points is: <Operand><Address>

Where: <Operand> indicates the memory area, the valid values are:

0 (%Q or %M) for Coils

1 (%I) for Input Status

3 (%IW) for Input Registers

4 (%MW) for Holding Registers

For more information about the valid operands, see the Table 1:

<Address> indicates the data address in the memory area, from 1 to 65535

Ex: 400001 (Operand = 4 (%MW), Address = 1)

Section 5 – Troubleshoot

The status of the driver execution can be observed through the diagnostic tools, which are:

- Trace window
- Property Watch
- Module Information

The above tools indicate if the operations have succeeded or have failed where the status 0 (zero) means success. Negative values are internal error codes and positive values are protocol error codes.

Modbus protocol error codes:

Error	Name	Description
1	ILLEGAL FUNCTION	The function code received in the query is not allowable.

2	ILLEGAL DATA ADDRESS	The data address received in the query is not allowable.
3	ILLEGAL DATA VALUE	A value contained in the query data field is not allowable.
4	SLAVE DEVICE FAILURE	Error while attempting to perform the requested action.
5	ACKNOWLEDGE	Request accepted, but a long duration of time will be required.
6	SLAVE DEVICE BUSY	The slave is engaged in a long–duration program command.
7	NEGATIVE ACKNOWLEDGE	Cannot perform the program function received in the query.
8	MEMORY PARITY ERROR	Parity error in the extended memory.

Revision History

Revision	Description	Date
A	Initial Revision	Nov,04th 2015