MOX MODBUS

Configuration Guide

0809-901-2903

Preface

Scope of the User Guide

This MOX User Guide provides basic information about the MODBUS protocol, and how to configure MOX equipment to implement it.

The guide has been organized for the operator, and it is expected that the user is an engineer, technician, electrician or similar with an understanding of the operating and programming requirements of the MOX system.

Related Documents

A MOX system contains a collection of MOX equipments and several software packages. For this reason, a number of related documents should be read in conjunction with this user guide.

The related documents are noted below:

- MOX Open Controller User Guide
- MOX Unity Field Controller User Guide
- MOX IoNix Field Controller User Guide
- MoxIDE User Guide
- MoxGRAF User Guide
- Special Function Block Programming Guide

Contents

Figures

Tables

1 Overview

1.1 MODBUS Introduction

MODBUS is an open communication protocol based on master/slave architecture. It defines a message structure often used in industrial automation to establish communication between intelligent devices. It is popular, well established, reliable and relatively easy to implement.

Each slave device that intends to communicate using MODBUS has a unique address. Valid MODBUS node addresses are in the range of 1-247 decimal. Address 0 is reserved for broadcast messages, which all slave devices recognize and useable for writes only. The master device can send out a MODBUS message which contains the MODBUS address of the slave device. The intended slave devices will respond by supplying requested data or taking requested actions. All MODBUS messages, although simple, contain checking information (Cyclic-Redundant Check or Longitudinal-Redundancy Check) to ensure reliability.

1.1.1 How is Data Stored in Standard MODBUS?

MODBUS data are organized into four different types identified by the leading number of the reference MODBUS address as following:

Reference MODBUS Address	Description
00001-**0**9999	Read-Write Coils
10001-**1**9999	Read-Only Discrete Input
30001-**3**9999	Read-Only Input Registers
40001-**4**9999	Read-Write Holding Registers

Table 1 MODBUS Address Allocation

The reference MODBUS addresses are just what the data locations are called and do not appear in the actual messages. The Addresses actually used in the messages are all referenced to 0. The first occurrence of a data item is addressed as item number zero. For example, holding register 40001 is addressed as register 0000 in the data address field of the message. The function code field specifies a holding register operate, so 40001 reference is implicit.

1.1.2 What is MODBUS Message Framing?

There are two types of transmission modes on standard MODBUS networks: ASCII and RTU. Users can select either of these two modes as well as corresponding serial port communication parameters, such as baud rate and parity mode. Each mode determines its own way to pack information into message fields. The following explanation gives specific frame structures of each mode.

ASCII Framing
In ASCII mode, each byte in the message is sent in the form of two ASCII characters. This mode allows up to 1s intervals between characters, not causing an error.

A typical message frame is as following:

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR	2 CHARS	2CHARS	*n* CHARS	2 CHARS	2 CHARS CRLFS

Table 2 ASCII Framing

RTU Framing
Different from ASCII framing, in RTU mode, the message frame must start with a silent interval of at least 3.5 character times. In a sending message of RTU mode, each 8-bit byte contains two 4-bit hexadecimal characters. One of its main advantages is that the message has greater character density than ASCII for the same baud rate.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4	8 BITS	8 BITS	*n* x 8 BITS	16 BITS	T1-T2-T3-T4

Table 3 RTU Framing

1.1.3 What is Function Code?

The function code is sent by the master in the massage to tell the slave what kind of action to perform.

Examples are to read the ON/OFF states of a group of discrete coils or inputs; to read the data contents of a group of registers; to write to designated coils or holding registers. Function code and their corresponding actions are listed below:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
06 (06H)	Write single Holding Register
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers

Table 4 Function Code

1.1.4 What is the Format of a Typical MODBUS Command and Response?

Here is an example of a command to request the content of registers 40005 to 40008 from the Slave with address 1.

Frame Field	Example Value (Hex)	Description
Slave Address	01	The Slave Address
Function	03	Read Analog Holding Registers
Starting Address Hi	00	The Data Address of the first register requested. (4 + 40001 = 40005)
Starting Address Lo	04	
No. of Points Hi	00	The total number of registers requested. (read 4 registers: from 40005 to 40008)
No. of Points Lo	04	
Error Check (LRC or CRC)	-	

Table 5 Example Query

The response message to the above query is shown below:

Frame Field	Example Value (Hex)	Description
Slave Address	01	The Slave Address
Function	03	read Analog Holding Registers
Byte Count	08	08: The number of data bytes to follow (4 registers x 2 bytes each = 8 bytes)
Data	AE 41 56 52 43 40 F4 E6	AE41: The contents of register 40005 5652: The contents of register 40006 4340: The contents of register 40007 F4E6: The contents of register 40008
Error Check (LRC or CRC)	-	-

Table 6 Example Response

1.1.5 What are Data Types?

In MODBUS, one register could be defined as any of these 16-bit data types:

- A 16-bit unsigned integer (a whole number between 0 and 65535)
- A 16-bit signed integer (a whole number between -32767 and 32768)
- A two character ASCII string (2 typed letters)

Two continuous MODBUS registers could be used to form any of these 32-bit data types:

- A 32-bit unsigned integer (a number between 0 and 4,294,967,295)
- A 32-bit signed integer (a number between -2,147,483,647 and 2,147,483,648)
- A four character ASCII string (4 typed letters)
- A 32-bit IEEE floating point number. This is a mathematical notation that allows any real number (a number with decimal points) to be represented by 32 bits with an accuracy of about seven digits.

Longer ASCII strings can also be defined by using more registers with each register being used to store two ASCII characters (two bytes).

1.1.6 What is Byte and Word Ordering?

The MODBUS specification does not define exactly how the data is stored in the registers. Therefore, some manufacturers implement MODBUS in their equipment to store and transmit the higher byte first followed by the lower byte. Alternatively, others store and transmit the lower byte first.

Similarly, when registers are combined to represent 32-bit data types, some devices store the higher 16 bits (high word) in the first register and the remaining low word in the second while others do the opposite.

It does not matter in which order the bytes or words are sent, as long as the receiving device knows which way to expect them.

For example, if the number 29,235,175,522 (AE415652H) was stored in two adjacent registers, e.g. 40005 and 40006, it could be arranged in any of these four ways:

40005	40006	Description	
AE41	5652	high byte/low byte	high word/low word
41AE	5256	low byte/high byte	high word/low word
5652	AE41	high byte/low byte	low word/high word
5256	41AE	low byte/high byte	low word/high word

Table 7 Byte and Word Ordering

1.2 MODBUS TCP/IP Introduction

MODBUS TCP/IP embeds a standard MODBUS data frame inside a TCP/IP frame without the MODBUS checksum. The MODBUS function codes and user data are not modified, and they are just encapsulated into the TCP/IP telegram. The MODBUS TCP/IP protocol enables the MODBUS protocol to be carried over TCP/IP based Ethernet.

MODBUS is a trademark of Modicon, Inc. For more detailed information on MODBUS and MODBUS TCP/IP, you may visit www.modicon.com.

2 MOX Unity Configuration

2.1 MODBUS Master Settings

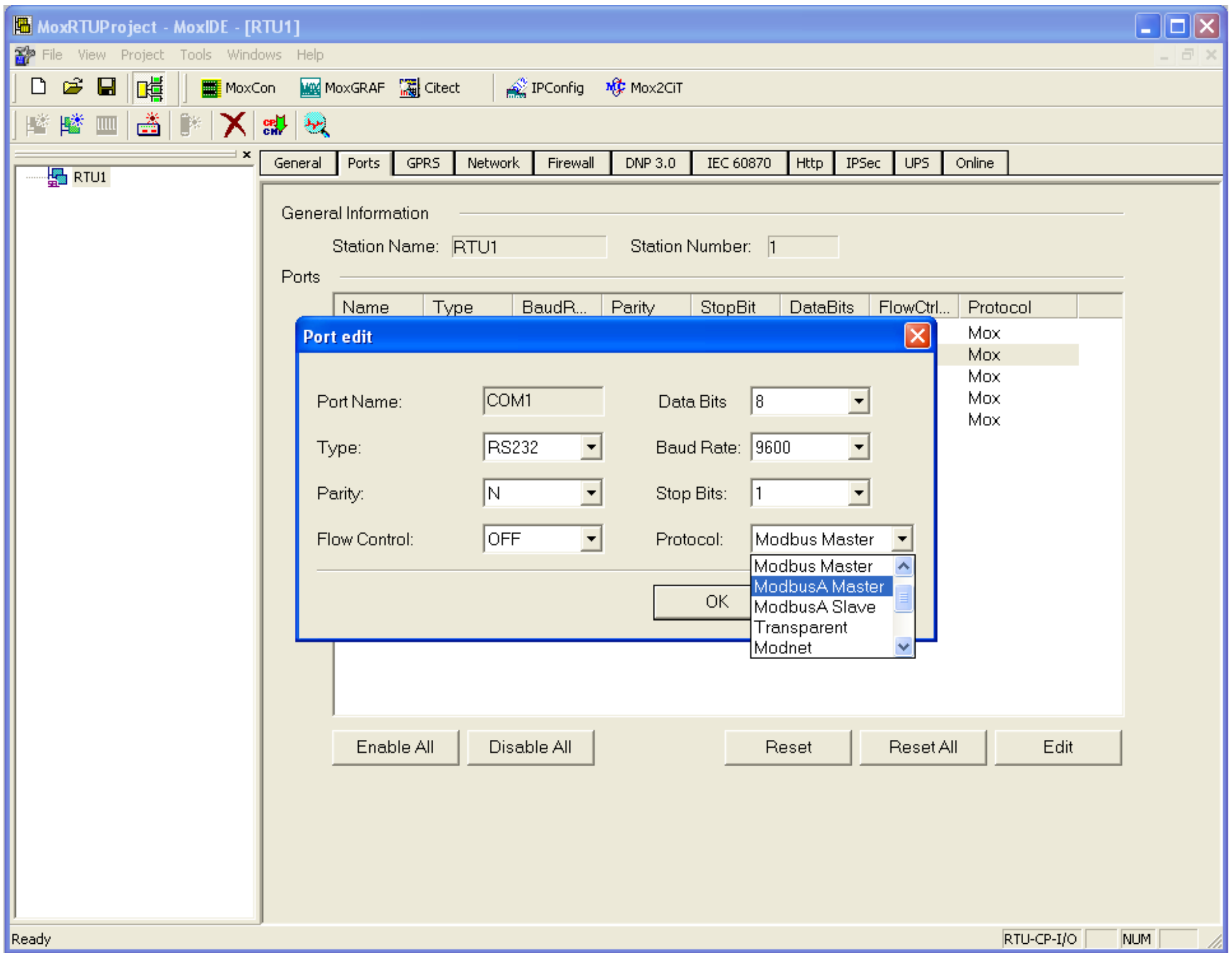
The MOX Unity has the ability to act as a MODBUS master through any of the four serial ports. When it acts as a MODBUS master, following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers

Table 8 Functions Supported by the MOX Unity Acting as a MODBUS Master

2.1.1 Serial Port Allocation

To establish the MOX Unity as a MODBUS master device, you can assign any of the serial ports as the communication port by selecting the "*Modbus Master" (RTU mode)* or *"ModbusA Master" (ASCII mode)* as the protocol using the **MoxIDE** port configuration feature.


MoxRTUProject - MoxIDE - [RTU1]
File View Project Tools Windows Help
MoxCon MoxGRAF Citect IPConfig Mox2CiT
RTU1
General Ports GPRS Network Firewall DNP 3.0 IEC 60870 Http IPSec UPS Online
Enable All Disable All Reset Reset All Edit
Ready RTU-CP-I/O NUM

General Information

Station Name: RTU1 Station Number: 1

Ports

Name	Type	BaudR...	Parity	StopBit	DataBits	FlowCtrl...	Protocol
							Mox
							Mox
							Mox
							Mox
							Mox

Port edit

Port Name:	COM1	Data Bits	8
Type:	RS232	Baud Rate:	9600
Parity:	N	Stop Bits:	1
Flow Control:	OFF	Protocol:	Modbus Master

Protocol list: Modbus Master, ModbusA Master, ModbusA Slave, Transparent, Modnet

OK

Figure 1 the MOX Unity Port Allocation and Configuration

2.1.2 Function Block

To retrieve information from a slave device, the user needs to build relevant code and make use of the ModBusM function block within MoxGRAF. Please refer to “Special Function Block Programming Guide” and “MoxGRAF User Guide” for more information.

2.2 MODBUS Slave Settings

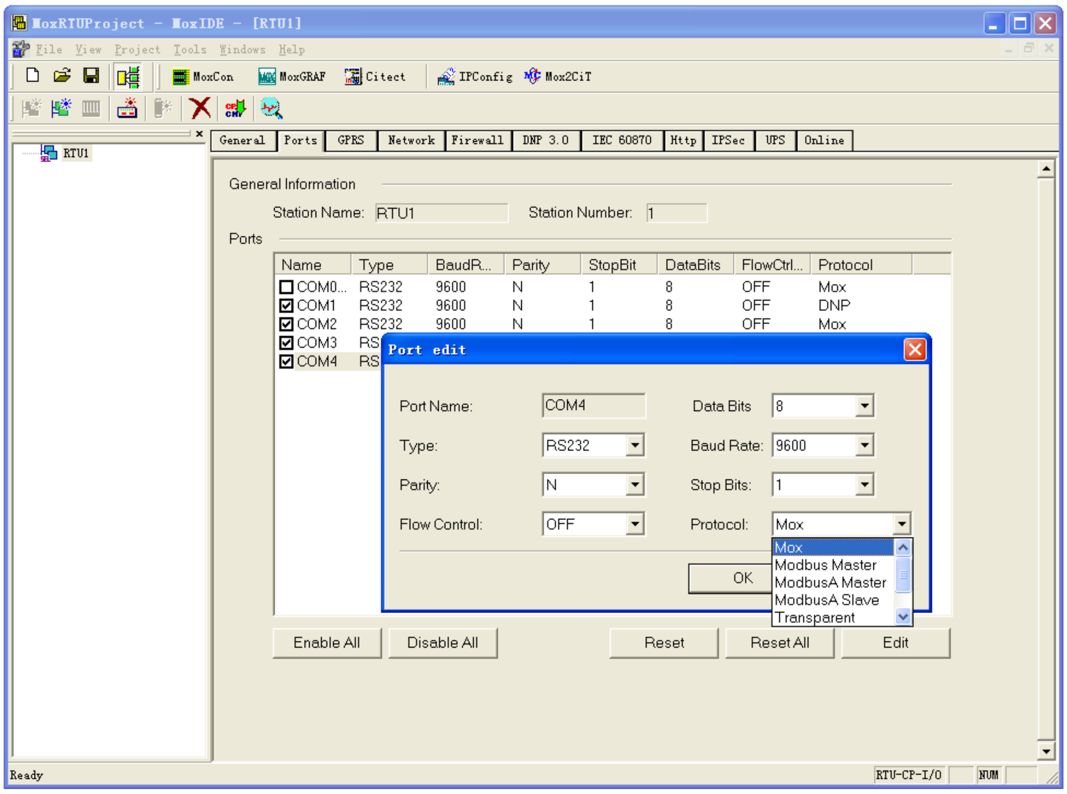
The MOX Unity has the ability to act as a MODBUS slave through any of the four serial ports. The MODBUS slave functionality is a built-in feature. When it acts as a MODBUS slave, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
15 (0FH)	Write multiple Coils
16 (0H)	Write multiple Holding Registers

Table 9 Functions Supported by the MOX Unity Acting as a MODBUS Slave

2.2.1 Serial Port Allocation

To establish the MOX Unity as a MODBUS slave device, you can assign any of the serial ports as a communication port by selecting the *“Mox” (RTU mode)* or *“ModbusA Slave” (ASCII mode)* as the protocol using the **MoxIDE** port configuration feature.



General Information

Station Name: RTU1 Station Number: 1

Ports

Name	Type	BaudR...	Parity	StopBit	DataBits	FlowCtrl...	Protocol
☐ COM0...	RS232	9600	N	1	8	OFF	Mox
☑ COM1	RS232	9600	N	1	8	OFF	DNP
☑ COM2	RS232	9600	N	1	8	OFF	Mox
☑ COM3	RS						
☑ COM4	RS						

Port edit

Port Name:	COM4	Data Bits	8
Type:	RS232	Baud Rate:	9600
Parity:	N	Stop Bits:	1
Flow Control:	OFF	Protocol:	Mox

Protocol list: Mox, Modbus Master, ModbusA Master, ModbusA Slave, Transparent

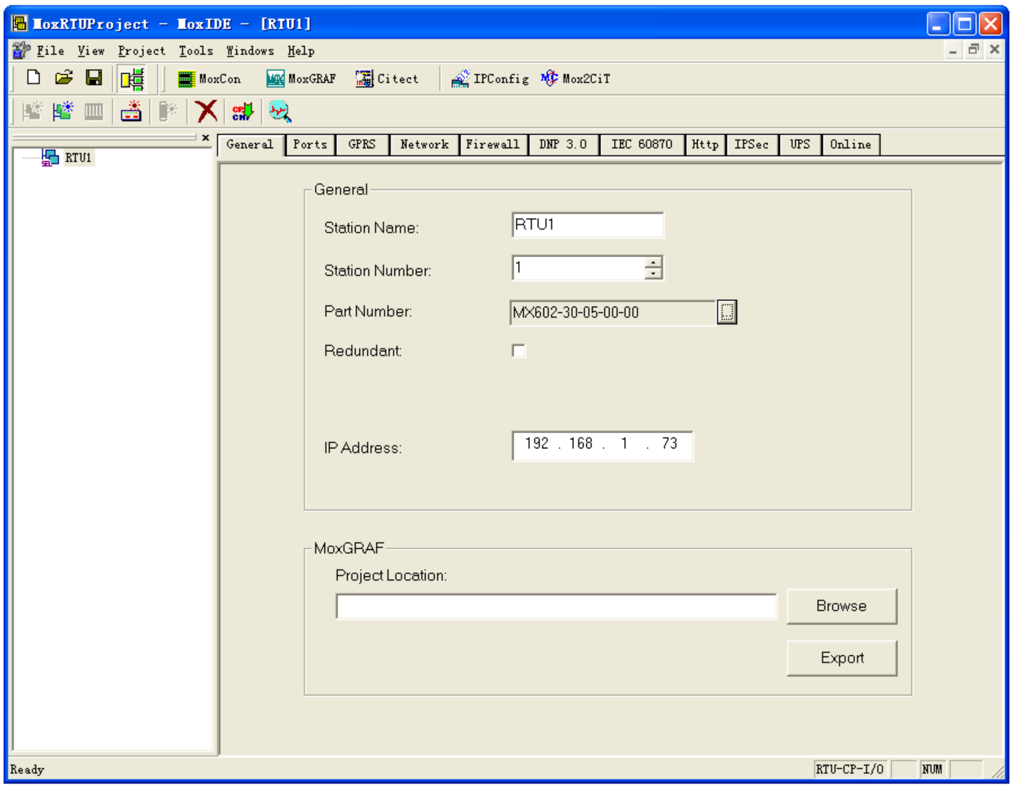
Figure 2 the MOX Unity Port Allocation and Configuration

2.2.2 Slave Address

To establish the MOX Unity as a MODBUS slave device, the MOX Unity is required to be configured with a slave address. This can be achieved using the **MoxIDE** station configuration feature.

The user should assign a **Station Number** for the MOX Unity, please refer to the following figure. The range of the Station Number is 1-250. 0 and 251-255 are reserved for system use. This Station Number is the MODBUS slave address for this MOX Unity.

The user has the option to change the Station Name of the controller. It is recommended you use a unique and meaningful name. Placing the **Station Number** at the end of the new name in this instance is a good way to distinguish it from all other controllers in the network tree in MoxIDE and remind you of the controller's address.


MoxRTUProject - MoxIDE - [RTU1]
File View Project Tools Windows Help
MoxCon MoxGRAF Citect IPConfig Mox2CiT
RTU1
General Ports GPRS Network Firewall DNP 3.0 IEC 60870 Http IPSec UPS Online
MoxGRAF
Project Location:
Browse
Export
Ready
RTU-CP-I/O
NUM

General

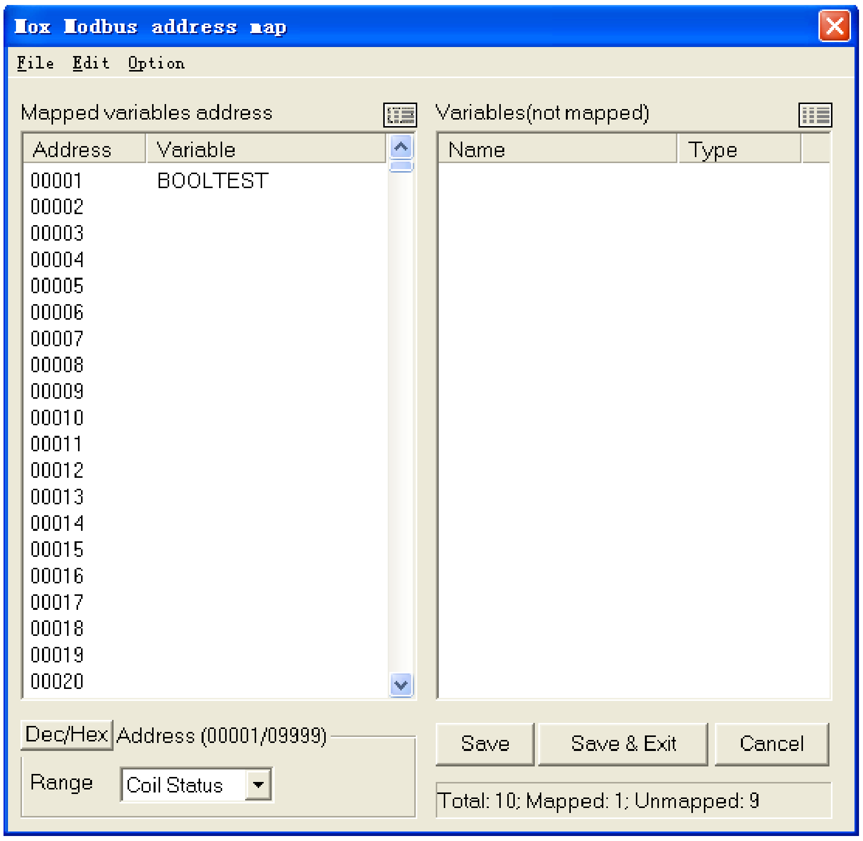
Station Name:	RTU1
Station Number:	1
Part Number:	MX602-30-05-00-00
Redundant:	☐
IP Address:	192 . 168 . 1 . 73

Figure 3 the MOX Unity Station Address Configuration

2.2.3 MODBUS Address Mapping

To establish the MOX Unity as a MODBUS slave device, the user is required to complete the MODBUS address mapping process. The process is aimed at creating a virtual MODBUS map for all MoxGRAF variables that must be controlled or monitored from the SCADA. This can be achieved using the MoxGRAF MODBUS Address Map Utility.

To open the MoxGRAF MODBUS Address Map Utility you can select ***Tools/Mox Modbus Address Map*** from the menu.



Address	Variable
00001	BOOLTEST
00002	
00003	
00004	
00005	
00006	
00007	
00008	
00009	
00010	
00011	
00012	
00013	
00014	
00015	
00016	
00017	
00018	
00019	
00020	

Address (00001/09999)

Range: Coil Status

Total: 10; Mapped: 1; Unmapped: 9

Figure 4 MOX MODBUS Address Map Utility

To start addressing variable tags, select a specified range, e.g. Coil Status. Single click the address from **Mapped Variables address** list to choose an address. Double click the variable that is to be mapped to the selected address from the Variables (not mapped) list. To un-map a mapped variable simply double click on it in the Mapped variable address window and it should move to the un-mapped window.

The MODBUS protocol has a specified address range for each variable type. To make variable address mapping easy, the MODBUS Address Map utility has these bounds set. By selecting the desired register type in "Range" option on the above window, all variables that meet that address range will fill the ***Variables (not mapped)*** window. The following information describes ranges in which variables can be mapped.

Type	MODBUS Address Range	Supports Variable Type
Coil	00001-09999	BOOL
Input Status	10001-19999	BOOL
Input Register	30001-39999	REAL, SINT, DINT, STRING
Holding Register	40001-49999	REAL, SINT, DINT, STRING

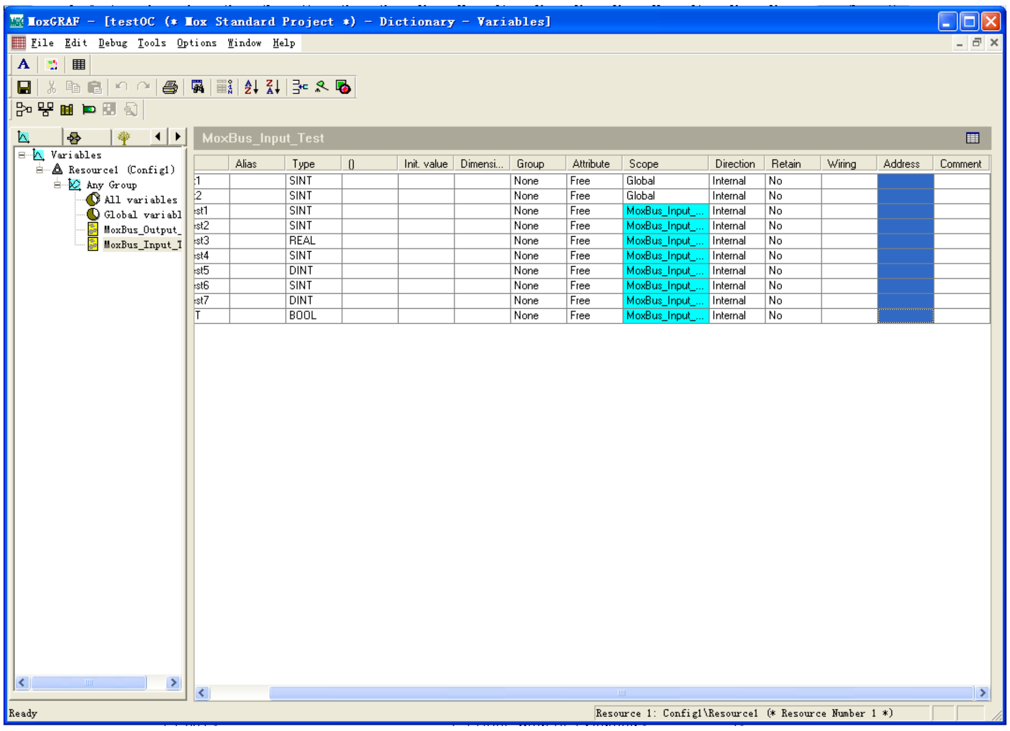
Table 10 MODBUS Address Ranges Variable Support

The table below displays variable data types that are compatible with MODBUS communication protocol and the byte order of each data type:

Variable Type	Byte Allocation	Byte Order
BOOL	1 Byte	-
SINT	1 Byte, map to 1 register	-
DINT	4 Bytes, map to 2 registers	65536 x high register + low register
REAL	4 Bytes, map to 2 registers	Byte order = 1 0 3 2.
STRING	Dependent on string length	Number of bytes = the string length +1(NULL terminal), if it's odd, bytes increase 1. For example, a string with length 4, the bytes should be 5, it's odd, then bytes should be 6, it will be mapped to 3 registers, A register 0 H B register 0 L C register 1 H D register 1 L \0 register 2 H \0 register 2 L

Table 11 Compatible Variable Types for MODBUS Communication

MOX MODBUS Address Map Utility is used to quickly create a MODBUS virtual map for each variable of the application. The tool will also update the variable address displayed in the MoxGRAF Dictionary (Variable Definition) Screen displayed in the figure below.


MoxGRAF - [testOC (* Mox Standard Project *) - Dictionary - Variables]
File Edit Debug Tools Options Window Help
Variables
Resource1 (Config1)
Any Group
All variables
Global variabl
MoxBus_Output_
MoxBus_Input_T
MoxBus_Input_Test
Ready
Resource 1: Config1\Resource1 (* Resource Number 1 *)

	Alias	Type	()	Init. value	Dimensi...	Group	Attribute	Scope	Direction	Retain	Wiring	Address	Comment
:1		SINT				None	Free	Global	Internal	No			
:2		SINT				None	Free	Global	Internal	No			
:st1		SINT				None	Free	MoxBus_Input_...	Internal	No			
:st2		SINT				None	Free	MoxBus_Input_...	Internal	No			
:st3		REAL				None	Free	MoxBus_Input_...	Internal	No			
:st4		SINT				None	Free	MoxBus_Input_...	Internal	No			
:st5		DINT				None	Free	MoxBus_Input_...	Internal	No			
:st6		SINT				None	Free	MoxBus_Input_...	Internal	No			
:st7		DINT				None	Free	MoxBus_Input_...	Internal	No			
T		BOOL				None	Free	MoxBus_Input_...	Internal	No			

Figure 5 MoxGRAF MODBUS Addressing Screen

Within the MoxGRAF Dictionary (Variable Definition) Screen, the MODBUS address assigned to each variable is shown in the form of a 4-bit Hexadecimal number in the **Address** column. You are also able to define a new or change the existing MODBUS address for a variable. If you decided to use this tool be sure to follow the address-mapping standard described above.

After completing the MODBUS address mapping process through using any of the tools described, you should select to build the complete embedded symbol table and download it to the MOX Unity. This can be performed by setting the resource property as shown in the figure below:

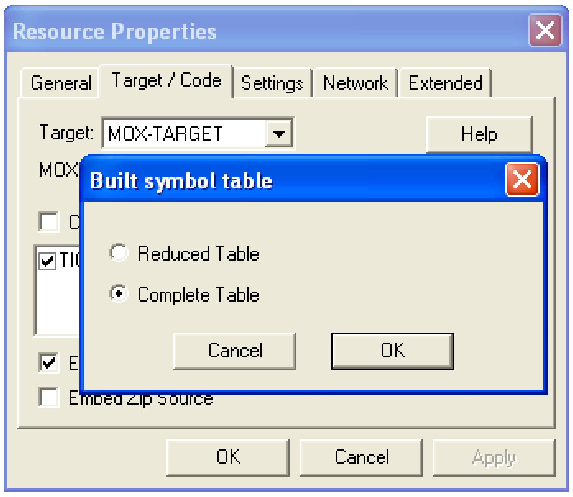


Figure 6 MODBUS Address Download Requirements

2.3 MODBUS TCP/IP Master Settings

The MOX Unity has the ability to act as a MODBUS TCP/IP master through the Ethernet port.

When it acts as a MODBUS TCP/IP master, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
06 (06H)	Write single Holding Register
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers
20 (14H)	Read General Reference
21 (15H)	Write General Reference

Table 12 Functions Supported by the MOX Unity Acting as a MODBUS TCP/IP Master

2.3.1 IP Address

The IP address configuration of the MOX Unity's Ethernet ports can be achieved by selecting **Tools | IPConfig** within MoxIDE. Please refer to "MOX Unity Field Controller User Guide" for detailed **IPConfig** operations.

2.3.2 Function Block

To retrieve information from a slave device, the user needs to build relevant code and make use of the ModNetM function block within MoxGRAF. Please refer to “Special Function Block Programming Guide” and “MoxGRAF User Guide” for more information.

2.4 MODBUS TCP/IP Slave Settings

The MOX Unity has the ability to act as a MODBUS TCP/IP slave through the Ethernet port.

The MODBUS TCP/IP slave functionality is a built-in feature. The MOX Unity MODBUS TCP/IP protocol supports point-to-point communication and multiple-requests. The maximum master number is 9, i.e. a maximum of 9 masters can establish a connection with a MOX Unity at the same time.

When it acts as a MODBUS TCP/IP slave, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
06 (06H)	Write single Holding Register
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers

Table 13 Functions Supported by the MOX Unity Acting as a MODBUS TCP/IP Slave

2.4.1 IP Address

The IP address configuration of the MOX Unity’s Ethernet ports can be achieved by selecting **Tools | IPConfig** within MoxIDE. Please refer to “MOX Unity Field Controller User Guide” for detailed **IPConfig** operations.

2.4.2 MODBUS Address Mapping

To establish the MOX Unity as a MODBUS TCP/IP slave device, the user is required to complete the MODBUS address mapping process. The process is aimed at creating a virtual MODBUS map for all MoxGRAF variables that must be controlled or monitored from the SCADA. The process is the same as MODUBS slave, please refer to 2.2.3 for more information.

3 MOX Open Controller Configuration

3.1 MODBUS TCP/IP Master Settings

The MOX Open Controller (MOX OC) has the ability to act as a MODBUS TCP/IP master through the Ethernet port.

When it acts as a MODBUS TCP/IP master, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
06 (06H)	Write single Holding Register
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers
20 (14H)	Read General Reference
21 (15H)	Write General Reference

Table 14 Functions Supported by the MOX OC Acting as a MODBUS TCP/IP Master

3.1.1 IP Address

The IP address configuration of the MOX Open Controller's Ethernet ports can be achieved by selecting **Tools | IPConfig** within MoxIDE. Please refer to "MOX Open Controller User Guide" for detailed **IPConfig** operations.

3.1.2 Function Block

To retrieve information from a slave device, the user needs to build relevant code and make use of the ModNetM function block within MoxGRAF. Please refer to "Special Function Block Programming Guide" and "MoxGRAF User Guide" for more information.

3.2 MODBUS TCP/IP Slave Settings

The MOX Open Controller has the ability to act as a MODBUS TCP/IP slave through the Ethernet port.

The MODBUS TCP/IP slave functionality is a built-in feature. The MOX Open Controller MODBUS TCP/IP protocol supports point-to-point communication and multiple-requests. The maximum master number is 9, i.e. a maximum of 9 masters can establish a connection with a MOX Open Controller at the same time.

When it acts as a MODBUS TCP/IP slave, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
06 (06H)	Write single Holding Register
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers

Table 15 Functions Supported by the MOX OC Acting as a MODBUS TCP/IP Slave

3.2.1 IP Address

The IP address configuration of the MOX Open Controller's Ethernet ports can be achieved by selecting **Tools | IPConfig** within MoxIDE. Please refer to "MOX Open Controller User Guide" for detailed **IPConfig** operations.

3.2.2 MODBUS Address Mapping

To establish the MOX Open Controller as a MODBUS TCP/IP slave device, the user is required to complete the MODBUS TCP/IP address mapping process. The process is aimed at creating a virtual MODBUS map for all MoxGRAF variables that must be controlled or monitored from the SCADA. The process is the same as the MOX Unity, please refer to section 2.2.3 for more information.

4 MOX IoNix Configuration

4.1 MODBUS Master Settings

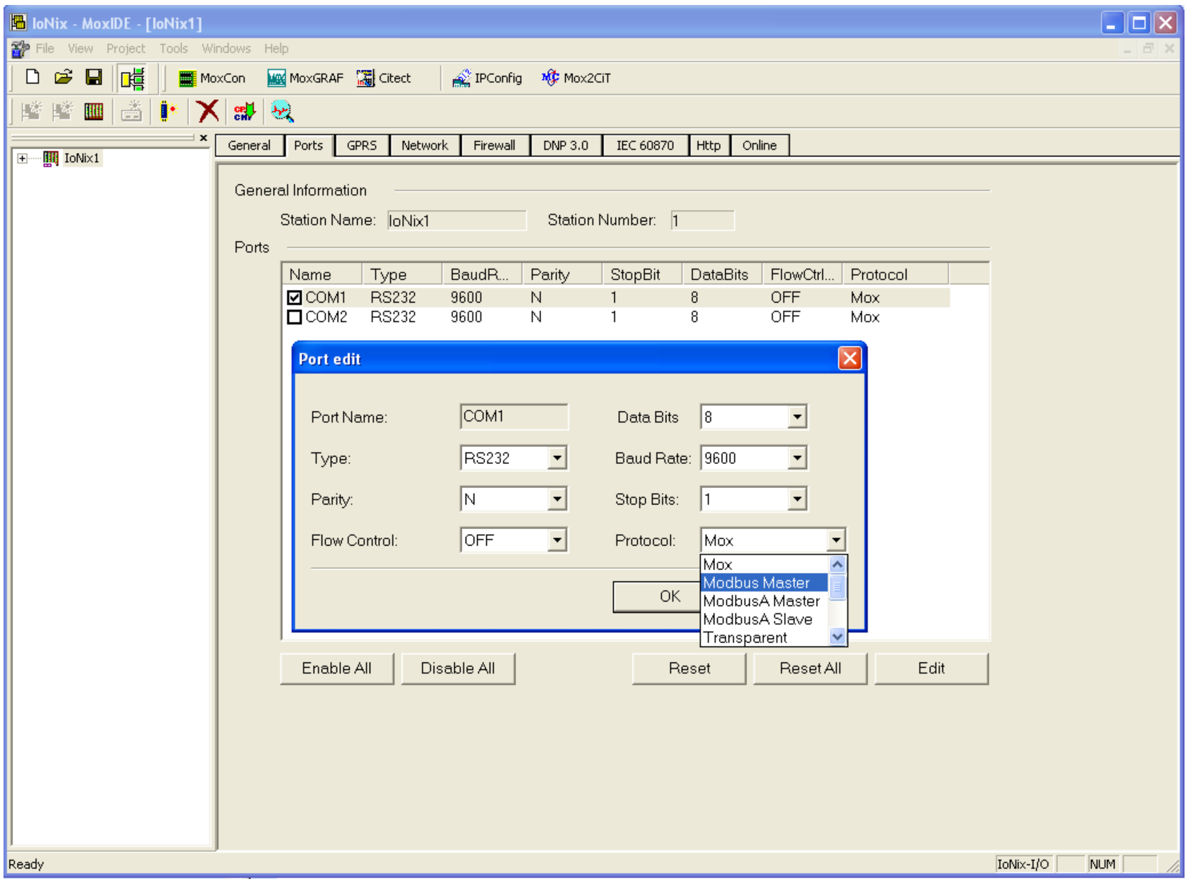
The MOX IoNix has the ability to act as a MODBUS master through either of the two serial ports. When it acts as a MODBUS master, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers

Table 16 Functions Supported by the MOX IoNix Acting as a MODBUS Master

4.1.1 Serial Port Allocation

To establish the MOX IoNix as a MODBUS master device, you can assign any of the serial ports as the communication port by selecting the "*Modbus Master" (RTU mode)* or *"ModbusA Master" (ASCII mode)* as the protocol using the **MoxIDE** port configuration feature.


IoNix - MoxIDE - [IoNix1]
File View Project Tools Windows Help
MoxCon MoxGRAF Citect IPConfig Mox2CiT
IoNix1
General Ports GPRS Network Firewall DNP 3.0 IEC 60870 Http Online
Port edit
OK
Enable All Disable All Reset Reset All Edit
Ready
IoNix-I/O NUM

General Information

Station Name: IoNix1 Station Number: 1

Ports

Name	Type	BaudR...	Parity	StopBit	DataBits	FlowCtrl...	Protocol
☑ COM1	RS232	9600	N	1	8	OFF	Mox
☐ COM2	RS232	9600	N	1	8	OFF	Mox

Port edit

Port Name:	COM1	Data Bits	8
Type:	RS232	Baud Rate:	9600
Parity:	N	Stop Bits:	1
Flow Control:	OFF	Protocol:	Mox

Protocol options: Mox, Modbus Master, ModbusA Master, ModbusA Slave, Transparent

Figure 7 the MOX IoNix Port Allocation and Configuration

4.1.2 Function Block

To retrieve information from a slave device, the user needs to build relevant code and make use of the ModBusM function block within MoxGRAF. Please refer to “Special Function Block Programming Guide” and “MoxGRAF User Guide” for more information.

4.2 MODBUS Slave Settings

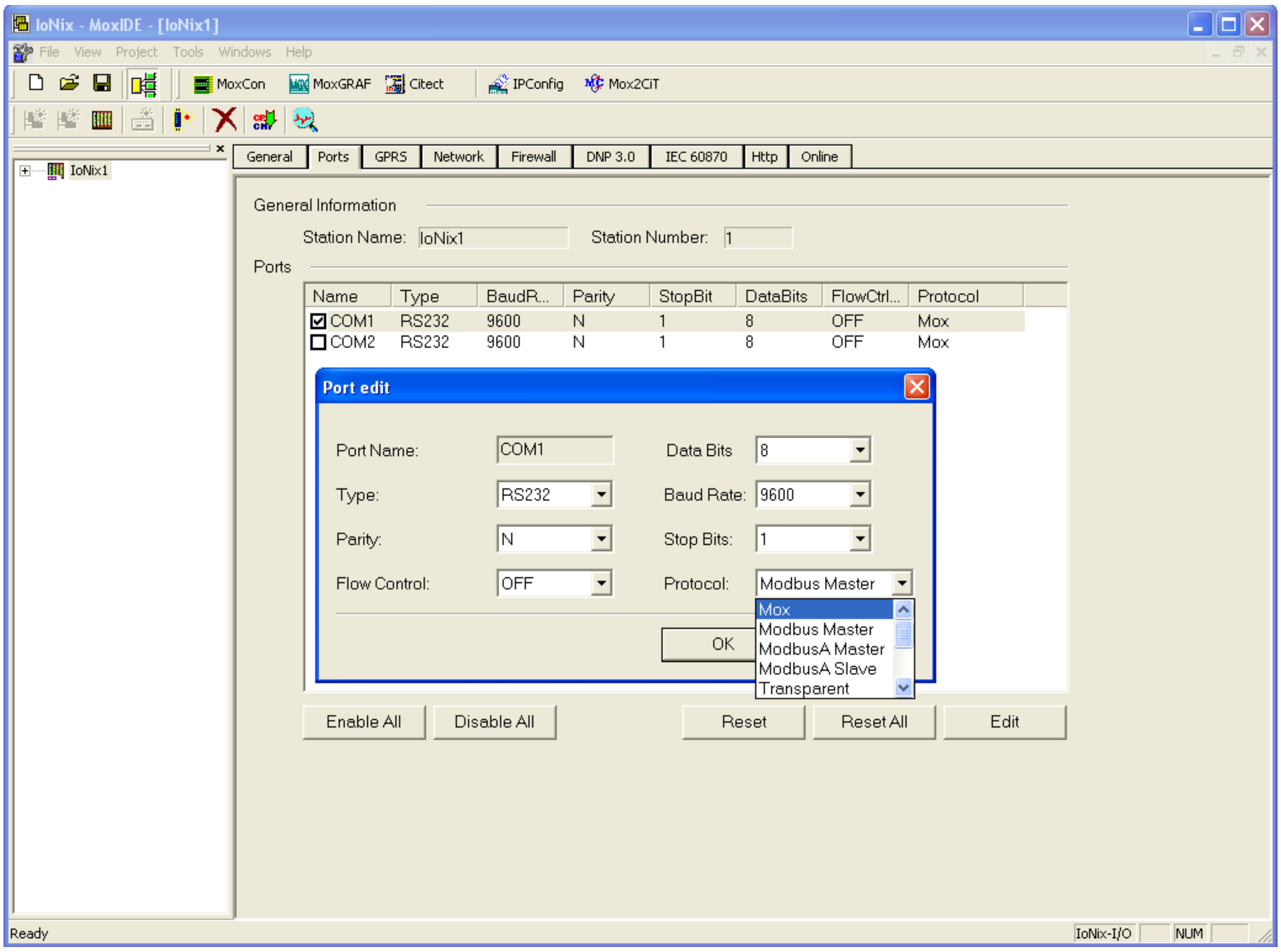
The MOX IoNix has the ability to act as a MODBUS slave through either of the two serial ports. The MODBUS slave functionality is a built-in feature. When it acts as a MODBUS slave, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers

Table 17 Functions Supported by the MOX IoNix Acting as a MODBUS Slave

4.2.1 *Serial Port Allocation*

To establish the MOX IoNix as a MODBUS slave device, you can assign any of the serial ports as a communication port by selecting the *"Mox" (RTU mode)* or *"ModbusA Slave" (ASCII mode)* as the protocol using the **MoxIDE** port configuration feature.


IoNix - MoxIDE - [IoNix1]
File View Project Tools Windows Help
MoxCon MoxGRAF Citect IPConfig Mox2CiT
IoNix1
General Ports GPRS Network Firewall DNP 3.0 IEC 60870 Http Online
Enable All Disable All Reset Reset All Edit
OK
Ready IoNix-I/O NUM

General Information

Station Name: IoNix1 Station Number: 1

Ports

Name	Type	BaudR...	Parity	StopBit	DataBits	FlowCtrl...	Protocol
☑ COM1	RS232	9600	N	1	8	OFF	Mox
☐ COM2	RS232	9600	N	1	8	OFF	Mox

Port edit

Port Name:	COM1	Data Bits	8
Type:	RS232	Baud Rate:	9600
Parity:	N	Stop Bits:	1
Flow Control:	OFF	Protocol:	Modbus Master

Mox
Modbus Master
ModbusA Master
ModbusA Slave
Transparent

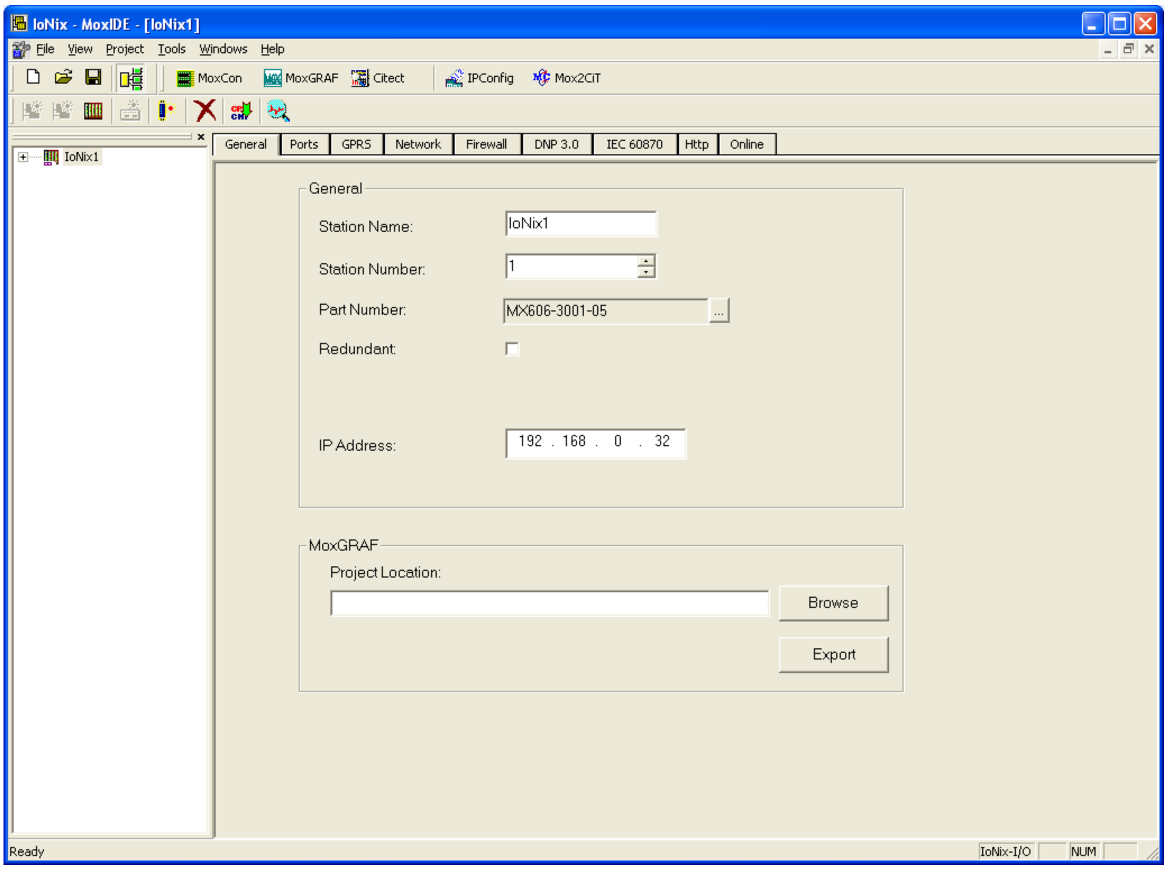
Figure 8 the MOX IoNix Port Allocation and Configuration

4.2.2 *Slave Address*

To establish the MOX IoNix as a MODBUS slave device, the MOX IoNix is required to be configured with a slave address. This can be achieved using the **MoxIDE** station configuration feature.

The user should assign a **Station Number** for the MOX IoNix, please refer to the following figure. The range of the Station Number is 1-250. 0 and 251-255 are reserved for system use. This Station Number is the MODBUS slave address for this MOX IoNix.

The user has the option to change the Station Name of the controller. It is recommended you use a unique and meaningful name. Placing the **Station Number** at the end of the new name in this instance is a good way to distinguish it from all other controllers in the network tree in MoxIDE and remind you of the controller's address.


IoNix - MoxIDE - [IoNix1]

File View Project Tools Windows Help
MoxCon MoxGRAF Citect IPConfig Mox2CiT
IoNix1
General Ports GPRS Network Firewall DNP 3.0 IEC 60870 Http Online

General

Station Name: IoNix1

Station Number: 1

Part Number: MX606-3001-05

Redundant: ☐

IP Address: 192 . 168 . 0 . 32

MoxGRAF

Project Location:

Browse
Export
Ready
IoNix-I/O NUM

Figure 9 the MOX IoNix Station Address Configuration

4.2.3 MODBUS Address Mapping

To establish the MOX IoNix as a MODBUS slave device, the user is required to complete the MODBUS address mapping process. The process is aimed at creating a virtual MODBUS map for all MoxGRAF variables that must be controlled or monitored from the SCADA. The process is the same as the MOX Unity, please refer to section 2.2.3 for more information.

4.3 MODBUS TCP/IP Master Settings

The MOX IoNix has the ability to act as a MODBUS TCP/IP master through the 10/100Mbps Ethernet port.

When it acts as a MODBUS TCP/IP master, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
06 (06H)	Write single Holding Register
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers
20 (14H)	Read General Reference
21 (15H)	Write General Reference

Table 18 Functions Supported by the MOX IoNix Acting as a MODBUS TCP/IP Master

4.3.1 IP Address

The IP address configuration of the Mox IoNix's Ethernet port can be achieved by selecting **Tools | IPConfig** within MoxIDE. Please refer to "MOX IoNix Field Controller User Guide" for detailed **IPConfig** operations.

4.3.2 Function Block

To retrieve information from a slave device, the user needs to build relevant code and make use of the ModNetM function block within MoxGRAF. Please refer to "Special Function Block Programming Guide" and "MoxGRAF User Guide" for more information.

4.4 MODBUS TCP/IP Slave Settings

The MOX IoNix has the ability to act as a MODBUS TCP/IP slave through the 10/100Mbps Ethernet port.

The MODBUS TCP/IP slave functionality is a built-in feature. The MOX IoNix MODBUS TCP/IP protocol supports point-to-point communication and multiple-requests. The maximum master number is 9, i.e. a maximum of 9 masters can establish a connection with a MOX IoNix at the same time.

When it acts as a MODBUS TCP/IP slave, the following functions are supported:

Function Code	Description
01 (01H)	Read Coil Status
02 (02H)	Read Input Status
03 (03H)	Read Holding Registers
04 (04H)	Read Input Registers
05 (05H)	Write single Coil
06 (06H)	Write single Holding Register
15 (0FH)	Write multiple Coils
16 (10H)	Write multiple Holding Registers

Table 19 Functions Supported by the MOX IoNix Acting as a MODBUS TCP/IP Slave

4.4.1 IP Address

The IP address configuration of the Mox IoNix's Ethernet port can be achieved by selecting **Tools | IPConfig** within MoxIDE. Please refer to "MOX IoNix Field Controller User Guide" for detailed **IPConfig** operations.

4.4.2 MODBUS Address Mapping

To establish the MOX IoNix as a MODBUS TCP/IP slave device, the user is required to complete the MODBUS address mapping process. The process is aimed at creating a virtual MODBUS map for all MoxGRAF variables that must be controlled or monitored from the SCADA. The process is the same as the MOX Unity, please refer to section 2.2.3 for more information.

Appendix A Product Support

Warranty Information

All MOX manufactured products are warranted to be free from defects in material and workmanship. Our obligation under this warranty will be limited to repairing or replacing, at our option, the defective parts within 1 year of the date of installation, or within 18 months of the date of shipment from the point of manufacture, whichever is sooner. Products may only be returned under authorization. The purchaser will prepay all freight charges to return any products with a valid return authorization number to the designated repair facility.

This limited warranty does not cover loss or damage that may occur in shipment of the goods or due to improper installation, maintenance, misuse, neglect or any cause other than ordinary commercial or industrial use. Warranty is also void if case is opened without manufacturer's consent. This limited warranty is in lieu of all other warranties whether oral or written, expressed or implied.

Liability associated with all MOX products shall not exceed the price of the individual unit that is the basis of the claim. In no event will there be liability for any loss of profits, loss of use of facilities or equipment or other indirect, incidental or consequential damages.

Contact Details

To obtain support for MOX products, call MOX Group on the following numbers or your designated support provider and ask for MOX Support.

E-mail addresses:

support@mox.com.au

sales@mox.com.au

Visit our web page at:

http://www.mox.com.au

Service Information

If you require service, contact your local MOX Group representative. A trained specialist will help you to quickly determine the source of the problem. Many problems are easily resolved with a single phone call. If it is necessary to return a unit, an RMA (Return Material Authorization) number will be provided.

All returned materials are tracked with our RMA system to ensure speedy service. You must include this RMA number on the outside of the box so that your return can be processed immediately.

Your MOX Group authorized applications engineer will complete an RMA request for you. If the unit has a serial number, we will not need detailed financial information. Otherwise, be sure to have your original purchase order number and date purchased available.

We suggest that you provide a repair purchase order number in case the repair is not covered under our warranty. You will not be billed if the repair is covered under warranty.

Please supply us with as many details about the problem as you can. The information you supply will be written on the RMA form and supplied to the repair department before your unit arrives. This helps us to provide you with the best service, in the fastest manner. Most repairs are completed within two days. During busy periods, there may be a longer delay.

If you need a quicker turnaround, ship the unit to us by airfreight. We give priority service to equipment that arrives by overnight delivery. Many repairs received by midmorning (typical overnight delivery) can be finished the same day and returned immediately.

We apologize for any inconvenience that the need for repair may cause you. We hope that our rapid service meets your needs. If you have any suggestions to help us improve our service, please give us a call. We appreciate your ideas and will respond to them.

For Your Convenience:

Please fill in the following information and keep this manual with your MOX system for future reference:

P.O. #: ___________________ Date Purchased: ____________________

Purchased From: ____

MOX Group

Web: www.mox.com.au
Email: info@mox.com.au

DOC 0809-901-2903 **VER** 1.01.05