JNIOR Modbus Protocol Implementation

Dated November 8, 2010
(v4.0 and later)

Notice Every effort was made to make this document as accurate and useful as practical at the time of writing. INTEG process groups reserves the right to alter the information presented herein as well as the function of the JNIOR product at any time without prior notice. All information is subject to change.

Use Restrictions This document, all related documents and the software contained in the **JNIOR** are copyrighted by INTEG process group and may not be copied or reproduced without prior consent from INTEG process group, inc.

INTEG process group, inc.
2919 E. Hardies Rd, First Floor
Gibsonia, PA 15044

www.integpg.com

JNIORsales@integpg.com

PH (724) 933-9350
FAX (724) 443-3553

Contents

Revision History

Date	OS Version	Change Description
12/22/2005	v2.03.225	Added login extension supporting an encoded login.
2/1/2006	v2.11.221	Remapped Counter and Usage addressing. Bit and Register space now overlap and are immutable (non-volatile). All functions now address the same memory space.
7/19/2006	v2.11.395	Added 104 (0x68) Pulse Multiple Coils with Mask and 105 (0x69) Write Multiple Coils with Mask functions. Corrected address references throughout the document. Added example transactions and additional Registry, numeric format and application information.
8/23/2006	v2.12.38	Added read/write access to external Sensor Port modules.
4/9/2008	v2.14.17	Added support for the external RTD Analog Module and the 4ROUT Digital Module with pulse capabilities.
7/21/2009	V3.3	Modbus Server is now ’disabled’ by default
11/16/2009	V3.4	Modbus Server is back to being ’enabled’ by default
11/8/10	V4.0	Added Digital Inputs and Relay Outputs 9-16 to the address space

This defines the JNIOR Modbus/TCPIP protocol implementation available through the Ethernet network.

Modbus Server Enable/Disable

By default the Modbus protocol server is enabled and ready to accept connections when the JNIOR is received from the factory. (Please note that login is also enabled by default.) The server may be configured to not run in the future by defining and setting the following Registry key. Note that this key affects the server status after boot and does not enable or disable the server immediately.

```
ModbusServer/Server = enabled (default)
ModbusServer/Server = disabled
```

Modbus Port

The default Modbus Port is 502. You may elect to use a custom port number compatible with your Modbus application software. Using a unique port number offers additional security against those that might be searching for Modbus servers on the network by making the Modbus server difficult to find. The following Registry key may be defined. This selects the port when the Modbus server is started at boot.

```
ModbusServer/Port = 502 (default)
```

Login Requirement

Unless it is specifically disabled by setting the following Registry Key to “disabled” this protocol implementation will require the successful completion of the **100 (0x64) User Login** Function before any of the remaining functions will be recognized. The Login will remain valid until the connection is broken or until another User Login is attempted.

```
ModbusServer/Login = enabled (default)
ModbusServer/Login = disabled
```

*jr*310 Modbus Addressing

The JNIOR Modbus address space consists of a single 8Kb non-volatile (immutable) memory block. JNIOR I/O registers overlay the low end of the address space. The remainder of the defined address space is available as Modbus scratch memory. Data written to this memory will retain its content after power is removed from the JNIOR. Optionally I/O expansion modules connected via the external Sensor Port may be mapped into the data space.

The standard Modbus memory areas are all one and the same. This means that Discrete Inputs, Coils, Input Registers and Holding Registers are all one and the same. Therefore the individual bits of any given register many be addressed using Discrete Input or Coil commands and likewise blocks of Discrete Input or Coil bits may be transferred as registers.

Addresses are zero-based in the protocol although application level programs may refer to them using a one-based approach. Care should be taken to avoid any confusion resulting in an offset of one.

The Register addressing is word (16-bit) based. For 32-bit and 64-bit values the lower address word should be read or written first. This makes a difference when reading the 32-bit counters for instance as JNIOR will sample the entire 32-bit counter value when the first (lower address) word is read. This insures that the value does not change should the counter advance while the words are being read. Writing occurs when the high address word is written.

Modbus Data Store		
Word Address Range	**Bit Address Range**	**Description**
0000	00000 – 00015	Digital Inputs/Relay Outputs (1-8) **
0001 – 0016	00016 – 00271	32-bit Digital Input Counters **
0017 – 0080	00272 – 01295	64-bit Input/Output Usage Meters **
0081	01296 – 01311	Digital Inputs/Relay Outputs (9-16) **
0082 – 0097	01312 – 01567	32-bit Digital Input Counters **
0098 – 0161	01568 – 02591	64-bit Input/Output Usage Meters **
0162 – 0255	02592 – 04095	- Reserved -
0256 – 4095	04096 – 65535	Immutable Scratch Memory

** The JNIOR I/O overlays the memory and the following addresses are predefined.

Discrete Inputs (bit addressing)	
Address	**Description**
bit 00000	Digital Input 1
bit 00001	Digital Input 2
bit 00002	Digital Input 3
bit 00003	Digital Input 4
bit 00004	Digital Input 5
bit 00005	Digital Input 6
bit 00006	Digital Input 7
bit 00007	Digital Input 8
bit 01296	Digital Input 9
bit 01297	Digital Input 10
bit 01298	Digital Input 11
bit 01299	Digital Input 12
bit 01300	Digital Input 13
bit 01301	Digital Input 14

bit 01302	Digital Input 15
bit 01303	Digital Input 16

Coils (bit addressing)	
Address	**Description**
bit 00008	Relay Output 1
bit 00009	Relay Output 2
bit 00010	Relay Output 3
bit 00011	Relay Output 4
bit 00012	Relay Output 5
bit 00013	Relay Output 6
bit 00014	Relay Output 7
bit 00015	Relay Output 8
bit 01304	Relay Output 9
bit 01305	Relay Output 10
bit 01306	Relay Output 11
bit 01307	Relay Output 12
bit 01308	Relay Output 13
bit 01309	Relay Output 14
bit 01310	Relay Output 15
bit 01311	Relay Output 16

Input Registers (32-bit, 2 words each)	
Address	**Description**
word 0001	Digital Input Counter 1
word 0003	Digital Input Counter 2
word 0005	Digital Input Counter 3
word 0007	Digital Input Counter 4
word 0009	Digital Input Counter 5
word 0011	Digital Input Counter 6
word 0013	Digital Input Counter 7
word 0015	Digital Input Counter 8
word 0082	Digital Input Counter 9
word 0084	Digital Input Counter 10
word 0086	Digital Input Counter 11
word 0088	Digital Input Counter 12
word 0090	Digital Input Counter 13
word 0092	Digital Input Counter 14
word 0094	Digital Input Counter 15
word 0096	Digital Input Counter 16

Input Registers (64-bit, 4 words each)	
Address	**Description**
word 0017	Digital Input 1 Usage Meter
word 0021	Digital Input 2 Usage Meter
word 0025	Digital Input 3 Usage Meter
word 0029	Digital Input 4 Usage Meter
word 0033	Digital Input 5 Usage Meter
word 0037	Digital Input 6 Usage Meter
word 0041	Digital Input 7 Usage Meter
word 0045	Digital Input 8 Usage Meter
word 0098	Digital Input 9 Usage Meter
word 0102	Digital Input 10 Usage Meter
word 0106	Digital Input 11 Usage Meter

word 0110	Digital Input 12 Usage Meter
word 0114	Digital Input 13 Usage Meter
word 0118	Digital Input 14 Usage Meter
word 0122	Digital Input 15 Usage Meter
word 0126	Digital Input 16 Usage Meter
word 0049	Relay Output 1 Usage Meter
word 0053	Relay Output 2 Usage Meter
word 0057	Relay Output 3 Usage Meter
word 0061	Relay Output 4 Usage Meter
word 0065	Relay Output 5 Usage Meter
word 0069	Relay Output 6 Usage Meter
word 0073	Relay Output 7 Usage Meter
word 0077	Relay Output 8 Usage Meter
word 0130	Relay Output 9 Usage Meter
word 0134	Relay Output 10 Usage Meter
word 0138	Relay Output 11 Usage Meter
word 0142	Relay Output 12 Usage Meter
word 0146	Relay Output 13 Usage Meter
word 0150	Relay Output 14 Usage Meter
word 0154	Relay Output 15 Usage Meter
word 0158	Relay Output 16 Usage Meter

You may toggle bits anywhere in the address space although the addressing gets a little difficult. Multiply the word address by 16 bits to get the starting bit address for that word. For instance word address 1,000 contains bit addresses 16,000 thru 16,015.

Mapping I/O Expansion Modules

I/O Expansion Modules of various types are available for connection to the JNIOR Sensor Port. These may include temperature probes or modules. A module may contain one or more individual 1-Wire devices. These devices provide additional input and output capabilities and access to these capabilities through the Modbus Protocol may be desired.

Each external device has its own unique 8-byte 64-bit device identification code (DeviceID). The JNIOR Protocol provides for read/write access to these devices and even defines pseudo-IDs to address the internal digital I/O. The JNIOR Modbus implementation maps the internal I/O by default to the fixed addresses described in the prior section. The external devices, where DeviceIDs and types are not known until connected, are not mapped by default. An entry in the JNIOR Registry is required to make the device available through this protocol.

Each device when read reports a ReadBlock containing the status of inputs and the current setting of outputs. The format of this ReadBlock varies by device type. Through a Registry key entry the entire content of a device's ReadBlock may be mapped into the Modbus memory space. Care must be taken to insure that address selections do not interfere with other devices or with Modbus addresses already in use for other purposes.

Locating the Device in the Registry

Devices may be connected or removed from the Sensor Port while the JNIOR remains powered. A new device will not appear in the Registry and therefore cannot be mapped until it is first discovered. This occurs during the boot process. When a new device is connected for the first time it is important to reboot the JNIOR. This insures that a registry entry for the device is created in the `OneWire/` registry folder.

After connecting the device and rebooting the JNIOR, use the Registry Editor or the Browser interface to examine the Registry. There should be a `OneWire/` folder containing one or more device addresses. The `OneWire/Devices` registry key lists the DeviceIDs of those devices connected during the discovery process that ran at boot-up. There will be a separate folder for each device listed and there may be other folders if other devices had been connected at a prior time. These folders are named by the hexadecimal representation of their DeviceID.

If you cannot determine which of the listed DeviceIDs belongs to the device that you want to map, disconnect all of the other devices and reboot. The `OneWire/Devices` key would then only contain addresses associated with the module or device you have connected. You can then open the Registry folder(s) associated with that device(s). You may edit `OneWire/<DeviceID>/Desc` key content as desired to identify the device.

Selecting a Mapping Address

Each device type has its own formatted data block. The entire block will be mapped into the Modbus address space. The block formats for typical devices will be described in the next section. These can be from 1 to dozens of bytes. You must select a Modbus address to insure that the entire block fits in the memory space and does not overlap any other devices.

Addresses 0 thru 255 are reserved for internal I/O. While addresses in this range that have not been outlined in the prior table may be used, an OS update may later introduce a conflict. Your address assignment should be at address 256 or above.

Modbus addresses refer to 16-bit words (2 bytes each). If you are mapping a 4-20ma Analog Expansion module for instance the associated ReadBlock is 12 bytes in length and contains 6 16-bit values (4 analog input followed by 2 analog output raw values). This would occupy 6 Modbus addresses. The highest address that you could use is then 4090 which would place the block at the very end of the

Modbus space. Recalling that the highest available address is 4095, using that address or others above 4090 would cause the block to overlap the end of memory and the error would lead to unpredictable results. Mapping this module to address 1000 would create the following assignments:

Address	Read/Write Status	Content
1000	Read Only	Ch 1 Current Input
1001	Read Only	Ch 2 Current Input
1002	Read Only	Ch 3 Current Input
1003	Read Only	Ch 4 Current Input
1004	Read/Write	Ch 1 Current Output
1005	Read/Write	Ch 2 Current Output

You can find out more about the formatting for this module in a subsequent section.

Mapping the Device

One you have selected a Modbus Address for the device you must create an entry in the Registry in that device's folder. For example, to map the analog module discussed in the prior section at address 1000 you would add the following key:

```
OneWire/<DeviceID>/ModbusAddress = 1000
```

where the DeviceID is the hexadecimal representation that appears in the Registry for the device. It is easiest to navigate to the folder for the device and create a `ModbusAddress` key. You need not type the DeviceID address. Note that Registry entries are case-sensitive.

Once this key appears in the Registry the device if connected will be available at that address with the next successful Modbus connection. Remember that the Modbus addressing as defined in this document is represented as zero-based (0-4095). Some applications may use one-based addressing (1-4096) and this may lead to some confusion if you are not aware of it.

Device ReadBlock Formats

Each device will have its own Read Block format and not all of the content may be written (if any). The blocks for some typical devices are shown here. Note that each Modbus address references 2 bytes. When reading or writing a mapped Modbus address one may need to apply masking and shifts to properly read/write the desired fields/values. Refer to the JNIOR Protocol document for current device read blocks, write blocks and descriptions.

External Sensor Type 10 – Temperature Probe Read Block

Item	**# of Bytes**	**Content**	**Defined Values**
1	8	Temperature (double) *Read Only*	Current temperature in degrees Celsius. Conversions take from 500 to 750 milliseconds. Resolution is 0.0625 degrees Celsius.
8 bytes Total Length (4 Modbus Addresses)			

External Sensor Type 12 – Dual Addressable Switch Read Block

Item	**# of Bytes**	**Content**	**Defined Values**
1	1	Channel A Level (byte) *Read/Write*	0=low, 1=high
2	1	Channel B Level (byte) *Read/Write*	0=low, 1=high
2 bytes Total Length (1 Modbus Address)			

External Sensor Type 1D – 4kbit RAM with Counter Read Block

Item	**# of Bytes**	**Content**	**Defined Values**
1	4	Channel A Count (unsigned int) *Read Only*	
2	4	Channel B Count (unsigned int) *Read Only*	
8 bytes Total Length (4 Modbus Addresses – 2 per counter)			

Note: RAM access not available.

External Sensor Type 20 – Quad A/D Converter Read Block

Item	**# of Bytes**	**Content**	**Defined Values**
1	2	Channel A (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFFF full scale.
2	2	Channel B (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFFF full scale.
3	2	Channel C (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFFF full scale.
4	2	Channel D (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFFF full scale.
8 bytes Total Length (4 Modbus Addresses)			

External Sensor Type 28 – Temperature Probe Read Block

Item	**# of Bytes**	**Content**	**Defined Values**
1	8	Temperature (double) *Read Only*	Current temperature in degrees Celsius. Conversions take from 500 to 750 milliseconds. Resolution is 0.0625 degrees Celsius.
8 bytes Total Length (4 Modbus Addresses)			

External Sensor Type 2C – Digital Potentiometer Read Block

Item	# of Bytes	Content	Defined Values
1	1	Feature Register (byte) *Read Only*	(see below)
2	1	Control Register (byte) *Read/Write*	(see below)
3	1	Wiper Position (byte) *Read/Write*	0-255
3 bytes Total Length (2 Modbus Addresses – position in MSB of second)			

Feature Register

D7	D6	D5	D4	D3	D2	D1	D0
PR		NWP		NP		WSV	PC

PC 0: logarithmic potentiometer element(s)
1: linear potentiometer element(s)

WSV 0: wiper position(s) non-volatile
1: wiper position(s) volatile

NP 00: device contains 1 potentiometer
01: device contains 2 potentiometers
10: device contains 3 potentiometers
11: device contains 4 potentiometers

NWP 00: 5-bit (32 positions)
01: 6-bit (64 positions)
10: 7-bit (128 positions)
11: 8-bit (256 positions)

PR 00: 5K Ohm resistance
01: 10K Ohm resistance
10: 50K Ohm resistance
11: 100K Ohm resistance

Control Register

D7	D6	D5	D4	D3	D2	D1	D0
X	CPC	X	X	IWN		WN	

WN 00: select potentiometer 1
01: select potentiometer 2
10: select potentiometer 3
11: select potentiometer 4

IWN 1’s complement of WN

CPC 0: charge pump OFF
1: charge pump ON

X don’t care

External Sensor Type FB – 4ROUT Digital Module

Item	# of Bytes	Content	Defined Values
1	1	Last Used Channel Select Mask	Bits D0 thru D3 correspond to Relay Outputs A thru D. (0 = No Change, 1 = Change)
2	1	Relay State	Bits D0 thru D3 correspond to Relay Outputs A thru D. (0 = Open, 1 = Closed)
3	2	Relay A Pulse Time Remaining. (unsigned short)	1 to 65535 milliseconds. (0 = static)
4	2	Relay B Pulse Time Remaining. (unsigned short)	1 to 65535 milliseconds. (0 = static)
5	2	Relay C Pulse Time Remaining. (unsigned short)	1 to 65535 milliseconds. (0 = static)
6	2	Relay D Pulse Time Remaining. (unsigned short)	1 to 65535 milliseconds. (0 = static)
10 bytes Total Length			

External Sensor Type FC – RTD Temperature Module

Item	# of Bytes	Content	Defined Values
1	2	Input Channel 1 (signed short)	Signed integer temperature in degrees Celsius X10.
2	2	Input Channel 2 (signed short)	Signed integer temperature in degrees Celsius X10.
3	2	Input Channel 3 (signed short)	Signed integer temperature in degrees Celsius X10.
4	2	Input Channel 4 (signed short)	Signed integer temperature in degrees Celsius X10.
8 bytes Total Length			

Note: Each channel requires the connection of a 2-wire or 3-wire PT100 RTD device. The value of 32767 (0x7FFF) is returned for unwired channels.

External Sensor Type FD – 10V Analog Module

Item	# of Bytes	Content	Defined Values
1	2	Input Channel 1 (short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
2	2	Input Channel 2 (short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
3	2	Input Channel 3 (short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
4	2	Input Channel 4 (short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
5	2	Output Channel 1 (unsigned short) *Read/Write*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
6	2	Output Channel 2 (unsigned short) *Read/Write*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
12 bytes Total Length (6 Modbus Addresses)			

Note: Inputs are +/- 10 volts. 0x0000 represents a -10V input and 0xFFF0 a +10V input. 0x8000 therefore represents the 0V input level. Outputs are 0 to 10 volts only. A setting of 0x0000 results in a 0V output and 0xFFF0 in +10V out. Only the most significant 12 bits are used for the output channels.

External Sensor Type FE – 4-20ma Analog Module

Item	# of Bytes	Content	Defined Values
1	2	Input Channel 1 (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
2	2	Input Channel 2 (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
3	2	Input Channel 3 (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
4	2	Input Channel 4 (unsigned short) *Read Only*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
5	2	Output Channel 1 (unsigned short) *Read/Write*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
6	2	Output Channel 2 (unsigned short) *Read/Write*	Raw 16-bit A/D reading 0x0000 to 0xFFF0 full scale.
12 bytes Total Length (6 modbus Addresses)			

Note: 0x0000 represents a 4 milliamp loop current and 0xFFF0 (thru 0xFFFF) represents 20 milliamps. Scaling is linear. Only the most significant 12 bits are used for the output channels.

For more information concerning ReadBlock content refer to the JNIOR Protocol documentation.

TCP/IP Message Structure

MBAP Header

The following header is used with each Request, Response or Error message.

Transaction ID	2 Bytes	0x0000 to 0xFFFF
Protocol ID	2 Bytes	**0x0000**
Length	2 Bytes	Count of all Bytes to follow including Unit ID
Unit ID	1 Byte	Ignored
Request/Response/Error	Variable	As Required

Transaction ID – User defined. JNIOR retransmits this ID number with the associated Response or Error message.

Protocol ID – Must be 0x0000.

Length – This is the total count of remaining bytes in the message including the Unit ID byte.

Unit ID – Specifies the Unit ID number. JNIOR represents a single unit and this field is ignored.

Byte Ordering

When acquiring a multiple-byte numeric value the byte order as transmitted across the network is important. These numeric values must be properly assembled from the data being received byte-by-byte through the network connection. The Modbus Protocol uses big-endian byte ordering.

big-endian: adj.
Describes a computer architecture in which, within a given multi-byte numeric representation, the most significant byte has the lowest address (the word is stored ‘big-end-first’). Most processors, including the IBM 370 family, the PDP-10, the Motorola microprocessor families, and most of the various RISC designs are big-endian. Big-endian byte order is also sometimes called network order.

A format for storage or transmission of binary data in which the most significant bit (or byte) comes first. The term comes from "Gulliver’s Travels" by Jonathan Swift. The Lilliputians, being very small, had correspondingly small political problems. The Big-Endian and Little-Endian parties debated over whether soft-boiled eggs should be opened at the big end or the little end. [Source: The Network Working Group Internet Glossary RFC 1392]

Java data streams format these numeric values by writing the most significant byte first using big-endian as is common in the network world. Java applications and applets can easily manipulate values passed through the Modbus Protocol. C/C++, Visual Basic and similar language programmers need to take extra care in constructing values as these languages may use the little-endian byte order employed by Intel processors and the standard Personal Computer.

Public Function Codes

01 (0x01) Read Coils

This function obtains the status of the Digital Inputs (Discrete Inputs), Relay Outputs (Coils) and generally the state of any bit address within the MODBUS data store. Its function is identical to Function Code 02 (0x02) Read Discrete Inputs.

Request

Function Code	1 Byte	**0x01**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Coils	2 Bytes	1 to 2000 (0x7D0)

Response

Function Code	1 Byte	**0x01**
Byte Count	1 Byte	**N***
Coil Status	**n** Bytes	n = N or N+1

***N** = Quantity of Outputs / 8, if the remainder is different than 0 then N = N+1

Error

Function Code	1 Byte	**Function Code + 0x80**
Exception Code	1 Byte	01 or 02 or 03

Example

The following reads the status of the eight Relay Outputs using the Read Coils function. Here we see that relays 1, 3, 4 and 6 are closed corresponding to the bit set in the returned 0x2D.

The following is the complete exchange. All bytes are in hexadecimal.

```
Transmit: (12 Bytes)                              Receive: (10 Bytes)
 00 04         MBAP: Trans ID                      00 04        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 06         MBAP: Msg Length (6)                00 04        MBAP: Msg length (4)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 01            Read Coils                          01           Read Coils
 00 08         Starting addr 0008                  01           Byte Count (1)
 00 08         All 8 relays                        2D           1, 3, 4 & 6 are closed
```

02 (0x02) Read Discrete Inputs

This function obtains the status of the Digital Inputs (Discrete Inputs), Relay Outputs (Coils) and generally the state of any bit address within the MODBUS data store. Its function is identical to Function Code 01 (0x01) Read Coils.

Request

Function Code	1 Byte	**0x02**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Inputs	2 Bytes	1 to 2000 (0x7D0)

Response

Function Code	1 Byte	**0x02**
Byte Count	1 Byte	**N***
Input Status	**n** Bytes	n = N or N+1

***N** = Quantity of Outputs / 8, if the remainder is different than 0 then N = N+1

Error

Function Code	1 Byte	**0x82**
Exception Code	1 Byte	01 or 02 or 03

Example

The following reads the status of the eight Relay Outputs using the Read Discrete Inputs function. Here we see that relays 1, 3, 4 and 6 are closed corresponding to the bit set in the returned 0x2D.

The following is the complete exchange. All bytes are in hexadecimal.

```
Transmit: (12 Bytes)                              Receive: (10 Bytes)
 00 05         MBAP: Trans ID                      00 05        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 06         MBAP: Msg Length (6)                00 04        MBAP: Msg length (4)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 02            Read Discrete                       02           Read Discrete
 00 08         Starting addr 0008                  01           Byte Count (1)
 00 08         All 8 relays                        2D           1, 3, 4 & 6 are closed
```

03 (0x03) Read Holding Registers

This function obtains the contents of the Digital Input Counters, Usage Meters and generally any 16-bit word value stored within the MODBUS Data Store. It is equivalent to Function Code 04 (0x04) Read Input Registers.

For 32-bit or 64-bit values multiple MODBUS Registers are required. The first (lowest address word) represents the high-order or most significant 16-bits of the value and the highest address word the least significant 16-bits.

The 32-bit counters and 64-bit usage meters are read one word at a time and the lower address word should be read first. The JNIOR will sample the entire multi-word value when the lowest address word is read. This insures that the returned value is accurate should the content advance during the process.

Request

Function Code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function Code	1 Byte	**0x03**
Byte Count	1 Byte	**2 x N***
Register Value	**N*** x 2 Bytes	

***N** = Quantity of Registers

Error

Function Code	1 Byte	**0x83**
Exception Code	1 Byte	01 or 02 or 03

Example

The following reads the 32-bit counters for Digital Input 1 and 2 using Read holding Registers. Note that we must read two Holding Registers for each counter.

This is the complete exchange. All bytes are in hexadecimal.

```
Transmit: (12 Bytes)                              Receive: (17 Bytes)
 00 06         MBAP: Trans ID                      00 06        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 06         MBAP: Msg length (6)                00 0B        MBAP: Msg length (11)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 03            Read Holding Regs                   03           Read holding Regs
 00 01         Starting addr 0001                  08           Byte Count (8)
 00 04         Register Count (4)                  00 00 04 6E  din1 count = 1,134
                                                   00 01 DC B9  din2 count = 122,041
```

The register at word address 0000 may be used to access the states of the Digital Inputs and Relay Outputs with one single read. More information may be found under the description for 06 (0x06) Write Single Register.

04 (0x04) Read Input Registers

This function obtains the contents of the Digital Input Counters, Usage Meters and generally any 16-bit word value stored within the MODBUS Data Store. It is equivalent to Function Code 03 (0x03) Read Holding Registers.

For 32-bit or 64-bit values multiple MODBUS Registers are required. The first (lowest address word) represents the high-order or most significant 16-bits of the value and the highest address word the least significant 16-bits.

The 32-bit counters and 64-bit usage meters are read one word at a time and the lower address word should be read first. The JNIOR will sample the entire multi-word value when the lowest address word is read. This insures that the returned value is accurate should the content advance during the process.

Request

Function Code	1 Byte	**0x04**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Input Registers	2 Bytes	1 to 125 (0x7D)

Response

Function Code	1 Byte	**0x04**
Byte Count	1 Byte	**2 x N***
Input Register Value	**N*** x n Bytes	

***N** = Quantity of Registers

Error

Function Code	1 Byte	**0x84**
Exception Code	1 Byte	01 or 02 or 03

Example

The following reads the 32-bit counters for Digital Input 1 and 2 using Read Input Registers. Note that we must read two Input Registers for each counter.

This is the complete exchange. All bytes are in hexadecimal.

```
Transmit: (12 Bytes)                              Receive: (17 Bytes)
 00 07         MBAP: Trans ID                      00 07        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 06         MBAP: Msg length (6)                00 0B        MBAP: Msg length (11)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 04            Read Input Regs                     04           Read Input Regs
 00 01         Starting addr 0001                  08           Byte Count (8)
 00 04         Register Count (4)                  00 00 04 6E  din1 count = 1,134
                                                   00 01 DC B9  din2 count = 122,041
```

The register at word address 0000 may be used to access the states of the Digital Inputs and Relay Outputs with one single read. More information may be found under the description for 06 (0x06) Write Single Register.

05 (0x05) Write Single Coil

This function opens or closes a single Relay Output and generally sets the state of any bit within the MODBUS Data Store.

Request

Function Code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Response

Function Code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Error

Function Code	1 Byte	**0x85**
Exception Code	1 Byte	01 or 02 or 03

Example

The following closes Relay Output 5 (bit address 0012). Note that an output value of 0x0000 would open the relay and 0xFF00 closes it.

This is the complete exchange. All bytes are in hexadecimal.

```
Transmit: (12 Bytes)                           Receive: (12 Bytes)
 00 08         MBAP: Trans ID                   00 08       MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)            00 00       MBAP: Protocol ID (0)
 00 06         MBAP: Msg Length (6)             00 06       MBAP: Msg Length (6)
 00            MBAP: Unit ID (X)                00          MBAP: Unit ID (X)

 05            Write Single                     05          Write Single
 00 0C         Address 0012                     00 0C       Address 0012
 FF 00         Close Relay (set bit)            FF 00       Close Relay
```

06 (0x06) Write Single Register

This provides access to the 32-bit Counters, Usage Meters and generally any 16-bit word within the MODBUS Data Store.

For 32-bit or 64-bit values multiple MODBUS Registers are required. The first (lowest address word) represents the high-order or most significant 16-bits of the value and the highest address word the least significant 16-bits.

The 32-bit counters and 64-bit usage meters are written one word at a time and the lower address word should be written first. The JNIOR will write the entire multi-word value when the highest address word is written. This insures that the value is properly set and does not change should the content advance during the process.

Request

Function Code	1 Byte	**0x06**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Response

Function Code	1 Byte	**0x06**
Output Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Error

Function Code	1 Byte	**0x86**
Exception Code	1 Byte	01 or 02 or 03

Examples

Here we write the value 2,169 to Modbus word address 1000. This is the complete exchange. Byte values are in hexadecimal.

```
Transmit: (12 Bytes)                            Receive: (12 Bytes)
 00 09         MBAP: Trans ID                    00 09        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)             00 00        MBAP: Protocol ID (0)
 00 06         MBAP: Msg Length (6)              00 06        MBAP: Msg Length (6)
 00            MBAP: Unit ID (X)                 00           MBAP: Unit ID (X)

 06            Write Single Reg                  06           Write Single Reg
 03 E8         Address 1000                      03 E8        Address Written 1000
 08 79         Value 2169                        08 79        Value Written 2169
```

Fast Digital Input and Relay Output Status

Modbus word address 0000 contains both the JNIOR Digital Input status and the Relay Output status packed as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rout8	rout7	rout6	rout5	rout4	rout3	rout2	rout1	din8	din7	din6	din5	din4	din3	din2	din1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R

When read, the address returns the full discrete status covering all of the JNIOR digital inputs and relay outputs.

Controlling 8 Relays Simultaneously

When word address 0000 is read, it returns the full discrete status covering all of the JNIOR digital inputs and relay outputs. Note carefully that the digital input bits are Read Only. When this address is written the entire set of eight Relay Outputs may therefore be commanded all at once. The relays change state simultaneously.

For example this exchange simultaneously opens the odd-numbered relays and closes the even-numbered relays. All relays change state at exactly the same moment.

```
Transmit: (12 Bytes)                              Receive: (12 Bytes)
 00 0A         MBAP: Trans ID                      00 0A        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 06         MBAP: Msg Length (6)                00 06        MBAP: Msg Length (6)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 06            Write Single Reg                    06           Write Single Reg
 00 00         Address 0000                        00 00        Address Written 0000
 AA 00         New relay states                    AA 00        Value Written
```

The 22 (0x16) Mask Write Register function may be used to command selected sets of relays simultaneously. This function’s use of logical AND and OR masks provides for very flexible “block” control of selected relays.

14 (0x0E) Read Device Identification

This returns basic device identification. The user-defined 102 (0x66) Read Registry Keys function should be used to obtain detailed product configuration information. Refer to Function Code 43 (0x2B) for description.

15 (0x0F) Write Multiple Coils

This provides access to the Relay Outputs and generally any bit within the MODBUS Data Store. Note that the Relay Outputs affected by this command DO NOT change state simultaneously.

Request

Function Code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0000 to 0x07B0
Byte Count	1 Byte	**N* (Must be 0x01)**
Outputs Value	**N*** x 1 Byte	

***N** = Quantity of Outputs / 8, if the remainder is different from 0 then N = N + 1

Response

Function Code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0000 to 0x07B0

Error

Function Code	1 Byte	**0x8F**
Exception Code	1 Byte	01 or 02 or 03

Example

The following opens Relay Outputs 1 through 4. The complete exchange is shown. All bytes are in hexadecimal.

```
Transmit: (14 Bytes)                              Receive: (12 Bytes)
 00 0B          MBAP: Trans ID                     00 0B        MBAP: Trans ID
 00 00          MBAP: Protocol ID (0)              00 00        MBAP: Protocol ID (0)
 00 08          MBAP: Msg Length (8)               00 06        MBAP: Msg length (8)
 00             MBAP: Unit ID (X)                  00           MBAP: Unit ID (X)

 0F             Write Multi Coils                  0F           Write Multi Coils
 00 08          Starting addr 0008                 00 08        Starting addr 0008
 00 04          Coil Count (4)                     00 04        Coil Count (4)
 01             Byte Count (1)
 00             Coil states (all open)
```

16 (0x10) Write Multiple Registers

This provides access to the 32-bit Counters, Usage Meters and generally any 16-bit word within the MODBUS Data Store.

For 32-bit or 64-bit values multiple MODBUS Registers are required. The first (lowest address word) represents the high-order or most significant 16-bits of the value and the highest address word the least significant 16-bits.

The 32-bit counters and 64-bit usage meters are written one word at a time and the lower address word should be written first. The JNIOR will write the entire multi-word value when the highest address word is written. This insures that the value is properly set and does not change should the content advance during the process.

Request

Function Code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	0x0000 to 0x0078
Byte Count	1 Byte	2 x N*
Registers Value	N* x 2 Bytes	

*N = Quantity of Registers

Response

Function Code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 123 (0x7B)

Error

Function Code	1 Byte	**0x90**
Exception Code	1 Byte	01 or 02 or 03

Example

This exchange stores the large value 534,210,927 in as a 64-bit value at address 1002. In order to do this we must write 4 individual 16-bit registers. The Write Multiple Registers function serves that purpose.

The following is the complete exchange. All bytes are in hexadecimal.

```
Transmit: (21 Bytes)                              Receive: (12 Bytes)
 00 0C          MBAP: Trans ID                     00 0C        MBAP: Trans ID
 00 00          MBAP: Protocol ID (0)              00 00        MBAP: Protocol ID (0)
 00 0F          MBAP: Msg Length (15)              00 06        MBAP: Msg length (6)
 00             MBAP: Unit ID (X)                  00           MBAP: Unit ID (X)

 10             Write Multi Regs                   10           Write Multi Regs
 03 EA          Starting Addr 1002                 03 EA        Addr 1002
 00 04          Write 4 Regs                       00 04        4 Regs Written
 08             Byte Count (8)
 00 00 00 00    Value 534210927
 1F D7 69 6F
```

22 (0x16) Mask Write Register

This provides access to the 32-bit Counters, Usage Meters and generally any 16-bit word within the MODBUS Data Store.

For 32-bit or 64-bit values multiple MODBUS Registers are required. The first (lowest address word) represents the high-order or most significant 16-bits of the value and the highest address word the least significant 16-bits.

Request

Function Code	1 Byte	**0x16**
Reference Address	2 Bytes	0x0000 to 0xFFFF
And_Mask	2 Bytes	0x0000 to 0xFFFF
Or_Mask	2 Bytes	0x0000 to 0xFFFF

Response

Function Code	1 Byte	**0x16**
Reference Address	2 Bytes	0x0000 to 0xFFFF
And_Mask	2 Bytes	0x0000 to 0xFFFF
Or_Mask	2 Bytes	0x0000 to 0xFFFF

Error

Function Code	1 Byte	**0x96**
Exception Code	1 Byte	01 or 02 or 03

Example

When used to access word address 0000 and thus modify the relay output states, the Mask Write Register function can be used to simultaneous change selected relays without affecting the state of others. The AND MASK is used to select the unaffected relays and the OR MASK is used to define the new state of the remaining relays.

More information regarding the use of word address 0000 to access both the Digital Inputs and Relay Outputs can be found under the description for the function 06 (0x06) Write Single Register.

The following example closes Relay Output 1 and Relay Output 7 without affecting the other relays. The two selected relays close simultaneously.

```
Transmit: (14 Bytes)                            Receive: (14 Bytes)
 00 03         MBAP: Trans ID                    00 03        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)             00 00        MBAP: Protocol ID (0)
 00 08         MBAP: Msg Length (8)              00 08        MBAP: Msg Length (8)
 00            MBAP: Unit ID (X)                 00           MBAP: Unit ID (X)

 16            Mask Write                        16           Mask Write successful
 00 00         Address (0000)                    00 00        Address modified
 BE 00         AND MASK (0xBE00)                 BE 00        AND MASK used
 41 00         OR MASK (0x4100)                  41 00        Or MASK used
```

With other word addresses the Mask Write Register function is useful in toggling individual bits that may be used as logical flags without affecting the settings of other flags within the word.

23 (0x17) Read/Write Multiple Registers

This provides access to the 32-bit Counters, Usage Meters and generally any 16-bit word within the MODBUS Data Store.

For 32-bit or 64-bit values multiple MODBUS Registers are required. The first (lowest address word) represents the high-order or most significant 16-bits of the value and the highest address word the least significant 16-bits.

The 32-bit counters and 64-bit usage meters are written one word at a time and the lower address word should be read or written first. The JNIOR will sample the entire multi-word value when the lowest address word is written and write the entire multi-word value when the highest address word is written. This insures that the value is properly returned and set should the content advance during the process.

Request

Function Code	1 Byte	**0x17**
Read Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity to Read	2 Bytes	0x0001 to approx. 0x0076
Write Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity to Write	2 Bytes	0x0001 to approx. 0x0076
Write Byte Count	1 Byte	2 x **N***
Write Registers Value	**N*** x 2 Bytes	

***N** = Quantity to Write

Response

Function Code	1 Byte	**0x17**
Byte Count	1 Byte	2 x **N'***
Read Registers Value	**N'*** x 2 Bytes	

***N'** = Quantity to Read

Error

Function Code	1 Byte	**0x97**
Exception Code	1 Byte	01 or 02 or 03

Example

This exchange writes the value 1,519 to word address 1000 returning the prior content of that address which in this case was 2,169.

```
Transmit: (19 Bytes)                              Receive: (11 Bytes)
 00 04         MBAP: Trans ID                      00 04        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 0D         MBAP: Msg Length (13)               00 05        MBAP: Msg Length (5)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 17            Read/Write Multi                    17           Read/Write Multi
 03 E8         Read Addr (1000)                    02           Byte Count (2)
 00 01         Read Count (1)                      08 79        Value 2169
 03 E8         Write Addr (1000)
 00 01         Write Count (1)
 02            Byte Count (2)
 05 EF         Value 1519
```

43 (0x2B) Read Device Identification

This returns basic device identification. The user-defined 102 (0x66) Read Registry Keys function should be used to obtain detailed product configuration information. This implementation is also available as Function Code 14 (0x0E) for compatibility.

Request

Function Code	1 Byte	**0x2B**
MEI Type	1 Byte	0x0E
Read Device ID Code	1 Byte	01 / 02 / 03 / 04
Object ID	1 Byte	0x00 to 0xFF

Response

Function Code	1 Byte	**0x2B**
MEI Type	1 Byte	0x0E
Read Device ID Code	1 Byte	01 / 02 / 03 / 04
Conformity Level	1 Byte	
More Follows	1 Byte	00 / FF
Next Object ID	1 Byte	
Number of Objects	1 Byte	
List Of		
Object ID	1 Byte	
Object length	1 Byte	
Object Value	Object length	String

Error

Function Code	1 Byte	**0xAB**
MEI Type	1 Byte	0x0E
Exception Code	1 Byte	01 or 02 or 03

Object Table

Object ID	Object Description	Type
0x00	VendorName	ASCII String
0x01	ProductCode	ASCII String
0x02	MajorMinorRevision	ASCII String

Example

This reads the JNIOR OS version string (MajorMinorRevision).

```
Transmit: (11 Bytes)                              Receive: (24 Bytes)
 00 05         MBAP: Trans ID                      00 05        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 05         MBAP: Msg Length (5)                00 12        MBAP: Msg Length (13)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 2B            Read Device Ident                   2B           Read Device Ident
 0E            MEI 0x0E                            0E           MEI 0x0E
 01            Read Device (1)                     01           Read Device (1)
 02            Object ID (2)                       81           Conformity
                                                   00           No More Follows
                                                   00           Next Object ID (was last)
                                                   01           Count of Objects (1)

                                                   02           MajorMinorRevision Object
                                                   08           String Length (8)
                                                   32 2E 31 31  “2.11.395”
                                                   2E 33 39 35
```

It is recommended that the special function 102 (0x66) Read Registry Keys be used to obtain detailed device information in preference to this method.

User-Defined Functions

The following are Function Codes specific to the JNIOR Modbus implementation.

100 (0x64) User Login

By default a successful User Login is required to activate the balance of this Modbus implementation. This Login requirement may be disabled through the Registry.

Request

Function Code	1 Byte	**0x64**
UserName Length	1 Byte	1 to 255
UserName	Length	String
Password Length	1 Byte	0 to 255
Password	Length	String

Response

Function Code	1 Byte	**0x64**
Login ID	1 Byte	0 to 255

Error

Function Code	1 Byte	**0xE4**
Exception Code	1 Byte	01 or 02 or 03

If the User Login is unsuccessful then the Login ID of 255 (0xFF) is returned. The Modus implementation will remain inactive. Login ID values from 127 to 254 indicate administrator status (Registry can be written).

The User Login may be disabled through the Registry by setting the M`odbusServer/Login` key to `disabled`.

The User Login may take a few seconds to complete. It is therefore recommended that the Modbus connection not be opened and closed for each command. This will result in slow response. Instead an open connection can be maintained for faster command response.

Encoded Password Transfer

The UserName and Password in the above transaction are transferred in clear text. This means that someone able to monitor network traffic may view packet content and will be able to see your login information. This may be of concern when communicating with JNIOR over public networks.

Optionally one may encode the combined username:password string (for instance “jdoe:mypass”) using Base64 encoding as defined by IEC RFC 1521. This renders the login information in a format that is not easily read by humans. The base64 encoded login string is transferred as the Password and its use is signified by supplying a blank (zero length) Username string. Note that this a minimal step and by no means represents true security. It will however minimize the temptation associated with accidentally discovering a user’s password.

Example

The following transaction successfully logs into the connection using the default administrator’s account. This is a clear text transmission and it is recommended that the login be performed using the encoded method for increased security.

This is the complete transaction. All bytes are in hexadecimal.

```
Transmit: (20 Bytes)                              Receive: (9 Bytes)
 00 01         MBAP: Trans ID                      00 01       MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00       MBAP: Protocol ID (0)
 00 0E         MBAP: Msg Length (14)               00 03       MBAP: Msg Length (3)
 00            MBAP: Unit ID (X)                   00          MBAP: Unit ID (X)

 64            User Login                          64          User Login
 05            Username Length (5)                 80          Success – ID (128)
 6A 6E 69 6F   “jnior”
 72
 05            Password Length (5)
 6A 6E 69 6F   “jnior”
 72
```

101 (0x65) Pulse Multiple Coils

This provides access to the JNIOR Relay Outputs through MODBUS address assignments 00008 thru 000015 wherein temporary output states can be obtained for selected coils of given durations. Pulse durations (T) are defined in milliseconds.

Request

Function Code	1 Byte	**0x65**
Pulse Count	2 Bytes	
List Of		
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0000 to 0x07B0
Byte Count	1 Byte	**Must be 0x01**
Outputs Value	1 Byte	
Pulse Duration	4 Bytes	Integer Milliseconds

Response

Function Code	1 Byte	**0x65**
Pulse Count	2 Bytes	

Error

Function Code	1 Byte	**0xE5**
Exception Code	1 Byte	01 or 02 or 03

A Pulse Count of zero (0) causes immediate cancellation of all queued pulse requests and termination of the currently executing pulse.

This defines a list of pulses that are executed sequentially. All outputs defined within a single pulse change states simultaneously.

Example

This transaction opens Relay Output 1 and Closes Relay Outputs 2 and 3 simultaneously for 5 seconds. This is the complete exchange. All byte are in hexadecimal.

```
Transmit: (20 Bytes)                              Receive: (10 Bytes)
 00 02          MBAP: Trans ID                     00 02         MBAP: Trans ID
 00 00          MBAP: Protocol ID (0)              00 00         MBAP: Protocol ID (0)
 00 0E          MBAP: Msg Length (14)              00 04         MBAP: Msg Length (4)
 00             MBAP: Unit ID (0)                  00            MBAP: Unit Id (X)

 65             Pulse Multi                        65            Pulse Multi
 00 01          Pulse Count (1)                    00 01         Number Queued

 00 08          Starting Addr (0008)
 00 03          Quantity of Outputs (3)
 01             Byte Count (1)
 06             Relay 1 off, 2&3 on
 00 00 13 88    5000 milliseconds
```

102 (0x66) Read Registry Keys

This provides access to the JNIOR Registry. The response contains the unparsed Registry content.

Request

Function Code	1 Byte	**0x66**
Number of Keys to Read	2 Bytes	0x0001 to 0xFFFF
List of		
User Assigned ID	2 Bytes	
Key Length	1 Byte	1 to 255
Key String	Key Length	String

Response

Function Code	1 Byte	**0x66**
Number of Keys Read	2 Bytes	0x0000 to 0xFFFF
List of		
User Assigned ID	2 Bytes	From request
Content Length	1 Byte	0 to 255
Content String	Content Length	String

Error

Function Code	1 Byte	**0xE6**
Exception Code	1 Byte	01 or 02 or 03

Example

Here we read the `$Version` Key from the Registry. The assignment of IDs is arbitrary and there to support parsing only. This is the complete exchange. All byte values are hexadecimal.

```
Transmit: (21 Bytes)                              Receive: (21 Bytes)
 00 02         MBAP: Trans ID                      00 02        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 0F         MBAP: Msg Length (15)               00 0F        MBAP: Msg Length (15)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 66            Read Registry                       66           Read Registry
 00 01         Key Count (1)                       00 01        Key Count (1)

 00 01         Arbitrary ID (1)                    00 01        Assigned ID (1)
 08            String Length (8)                   08           String length (8)
 24 56 65 72   "$Version"                          32 2E 31 31  "2.11.395"
 73 69 6F 6E                                       2E 33 39 35
```

103 (0x67) Write Registry Keys

This provides access to the JNIOR Registry. The response indicates how many keys were successfully written. If Login is required then this function will only be successful if the login is for an administrator account.

Request

Function Code	1 Byte	**0x67**
Number of Keys to Write	2 Bytes	0x0001 to 0xFFFF
List of		
Key Length	1 Byte	1 to 255
Key String	Key Length	String
Content Length	1 Byte	0 to 255
New Content String	Content Length	String

Response

Function Code	1 Byte	**0x67**
Number of Keys Written	2 Bytes	0x0000 to 0xFFFF

Error

Function Code	1 Byte	**0xE7**
Exception Code	1 Byte	01 or 02 or 03

Example

The following defines the `Test/NewKey` Registry Key and assigns the string “written” to it. This is the complete exchange. All byte values are hexadecimal.

```
Transmit: (30 Bytes)                            Receive: (10 Bytes)
 00 03         MBAP: Trans ID                    00 03        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)             00 00        MBAP: Protocol ID (0)
 00 18         MBAP: Msg Length (24)             00 04        MBAP: Msg Length (4)
 00            MBAP: Unit ID (X)                 00           MBAP: Unit ID (X)

 67            Write Registry                    67           Write Registry
 00 01         Key Count (1)                     00 01        Number Written

 0B            Key Length (11)
 54 65 73 74   “Test/NewKey”
 2F 4E 65 77
 4B 65 79
 07            Value Length
 77 72 69 74   “written”
 74 65 6E
```

The new Registry Key has been written (or updated) in the JNIOR Registry. This can be verified through a Telnet connection and the Registry editor as shown below. Note that there are other means by which you may work with the Registry. For instance, the standard applets that are supplied with the JNIOR may be used with an Internet browser to view and edit the Registry.

The following is part of a Telnet session in which we view the Registry Key just created using the 103 (0x67) Write Registry Keys function.

```
TINI /> registry
JNIOR Registry Editor
Copyright (C) 2005 INTEG process group, inc.  All Rights Reserved.
See Help for more information.


Content of /..
   1  $BootTime = Wed Jul 19 14:07:54 GMT 2006
   2  $Model = 310
   3  $SerialNumber = 123456
   4  $Version = 2.11.395
   5  IO/..
   6  IpConfig/..
   7  JniorServer/..
   8  Test/..
   9  <exit>

Key (or ## selection) to Add/Edit/Remove? 8

Content of Test/..
   1  NewKey = written
   2  <previous>

Key (or ## selection) to Add/Edit/Remove?

TINI />
```

104 (0x68) Pulse Multiple Coils with Mask

This provides access to the JNIOR Relay Outputs through MODBUS address assignments 00008 thru 000015 wherein temporary output states can be obtained for selected coils of given durations. Pulse durations (T) are defined in milliseconds. Only those outputs with Mask value of 1 will be affected by the command.

Request

Function Code	1 Byte	**0x68**
Pulse Count	2 Bytes	
List Of		
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0000 to 0x07B0
Byte Count	1 Byte	**Must be 0x01**
Mask Output Select	1 Byte	
Outputs Value	1 Byte	
Pulse Duration	4 Bytes	Integer Milliseconds

Response

Function Code	1 Byte	**0x68**
Pulse Count	2 Bytes	

Error

Function Code	1 Byte	**0xE8**
Exception Code	1 Byte	01 or 02 or 03

This defines a list of pulses that are executed sequentially. All outputs defined within a single pulse change states simultaneously. A Pulse Count of zero (0) causes immediate cancellation of all queued pulse requests and termination of the currently executing pulse.

Example

The following pulses relay outputs 1 and 3 for 5 seconds while not affecting relay output 2. Note that the output state for relay 2 in the command is ignored as the corresponding mask bit is not set. The return is immediate. The pulse completes 5 seconds later.

The complete exchange is shown below. All bytes are shown in hexadecimal.

```
Transmit: (21 Bytes)                              Receive: (10 Bytes)
 00 02         MBAP: Trans ID                      00 02        MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00        MBAP: Protocol ID (0)
 00 0F         MBAP: Msg Length (15)               00 04        MBAP: Msg Length (4)
 00            MBAP: Unit ID (X)                   00           MBAP: Unit ID (X)

 68            Pulse Multi w/Mask                  68           Pulse Multi w/Mask
 00 01         1 Pulse Requested                   00 01        1 Pulse Queued

 00 08         Starting addr 0008
 00 03         relay count is 3
 01            must be 1
 05            Addr 8 and 10 selected
 07            Set 8, 9 and 10
 00 00 13 88   5000 milliseconds
```

105 (0x69) Write Multiple Coils with Mask

This provides access to the JNIOR Relay Outputs through bit address assignments 00008 thru 000015 as well as bit states throughout the MODBUS Data Store. Only outputs with Mask value of 1 are affected by the command.

Request

Function Code	1 Byte	**0x69**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0000 to 0x07B0
Byte Count	1 Byte	**N***
Mask Output Select	**N*** x 1 Byte	
Outputs Value	**N*** x 1 Byte	

***N** = Quantity of Outputs / 8, if the remainder is different from 0 then N = N + 1

Response

Function Code	1 Byte	**0x69**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0000 to 0x07B0

Error

Function Code	1 Byte	**0xE9**
Exception Code	1 Byte	01 or 02 or 03

Note that the outputs affected by this command DO NOT change state simultaneously. If a simultaneous state change is required then 23 (0x17) Read/Write Multiple Registers must be used.

Example

The following closes relay outputs 4 and 6 while not affecting relay output 5. Note that the output state for relay 5 is ignored as the corresponding mask bit is not set.

The complete exchange is shown below. All bytes are in hexadecimal.

```
Transmit: (15 Bytes)                              Receive: (12 Bytes)
 00 03         MBAP: Trans ID                      00 03         MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)               00 00         MBAP: Protocol ID (0)
 00 09         MBAP: Msg length (9)                00 06         MBAP: Msg Length (6)
 00            MBAP: Unit ID (X)                   00            MBAP: Unit ID

 69            Write Multi w/Mask                  69            Write Multi w/Mask
 00 0B         Starting addr 0011                  00 0B         Starting Addr (11)
 00 03         Coil Count (3)                      00 03         Coil Count (3)
 01            byte count (1)
 05            Addr 11 & 13 selected
 07            Setr 11, 12, and 13
```

Modbus Exception Responses

Should there be a problem with a Request an Exception Response will be returned. The following describes the possible codes.

Code	Name	Meaning
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server at this time. This may be because a Login is required before the function may be used. It may be because the function is not part of this implementation.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server. More specifically, the combination of reference number and transfer length is invalid.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for the server. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect.

Example

The following transaction attempts to read a Holding Register outside of the available Modbus Data Store. This results in an expected Modbus Exception Response.

```
Transmit: (12 Bytes)                           Receive: (9 Bytes)
 00 02         MBAP: Trans ID                   00 02       MBAP: Trans ID
 00 00         MBAP: Protocol ID (0)            00 00       MBAP: Protocol ID (0)
 00 06         MBAP: Msg Length (6)             00 03       MBAP: Msg Length (3)
 00            MBAP: Unit ID (X)                00          MBAP: Unit ID (X)

 03            Read Holding Register            83          Func Code OR’d with 0x80
 1F 40         Address 8000 (bad)               02          Illegal Data Address
 00 01         Register Count (1)
```

Refer to the example for 03 (0x03) Read holding Registers to see the difference in a successful transaction.