HAIDAR TECHNOLOGY, LLC.

Industrial Control Innovations Begin Here

WWW.haidartechnology.com
(614) 389-3022
Sales@haidartechnology.com

ModBus Modules
RMC-xxxx-x Series

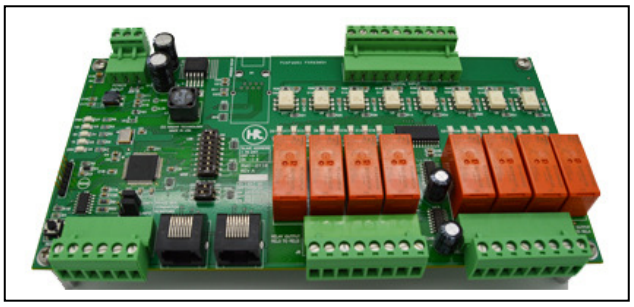
Reference Manual
REV 1.00

Revision 1.00
Issue Date: 6/29/2015

Important Notice:
Haidar Technology products are not designed, intended, authorized, or warranted to be suitable for use in life-support applications, devices, or systems, or in other critical applications. Haidar Technology and the buyer agree that Haidar Technology will not be liable for incidental or consequential damages arising from the use of Haidar Technology products. It is the user's responsibility to protect life and property against incidental failure. Haidar Technology reserves the right to make changes and improvements to its products without providing notice.

Table of Contents

1. Overview



Haidar offer complete stand-alone data acquisition modules which are broadly used in IoT or other industrial applications, such as facility monitoring, environment monitoring, and industrial process control. Haidar distributed remote monitor and control IO modules are categorized into two families, Ethernet IO (RMC-xxxx-E series) and Full-Duplex RS-485 IO (RMC-xxxx-S series), which are subdivided into Analog IO and Digital IO modules. All of them support Modbus RTU communication protocol.
Available Modules:

- RMC-D100-x: 8 Digital input & 8 Digital output
- RMC-D110-x: 8 Digital input & 8 Relay output
- RMC-D120-x: 16 Digital input
- RMC-D130-x: 16 Digital output
- RMC-D140-x: 16 Relay output
- RMC-A200-x: 8 Analog input, 4 analog output and 4 4-20ma CLT
- RMC-T300-x: 4 Thermocouple, 4 Relay output and 4 PWM output
- RMC-C800: RS232/USB To RS422 interface module
- RMC-C810: ModBus TCP To RS422 interface module

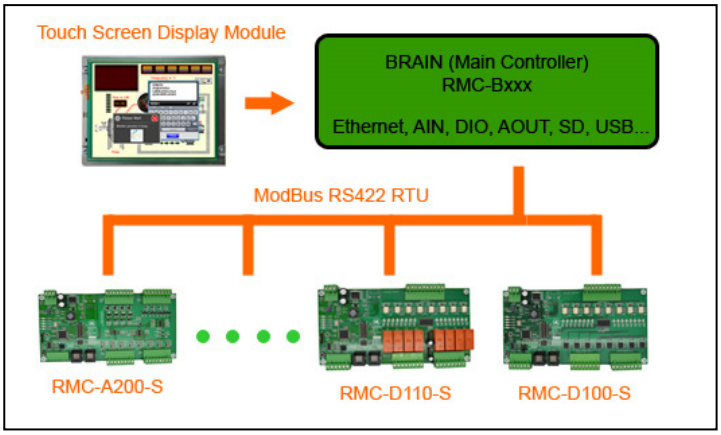
2. Features

Haidar's ModBus Modules are low cost yet very powerful modules that can be used for any application requires remote control and monitoring capability. Haidar's ModBus modules are designed with embedded control in mind. Most industrial process systems require user interface and control, simple microcontroller board can be used as a bridge between Haidar's user interface touch screens and Haidar's ModBus modules, forming robust and flexible control and monitoring architecture that can be used across wide range of industrial applications. Below are the main features of Haidar's ModBus Modules:

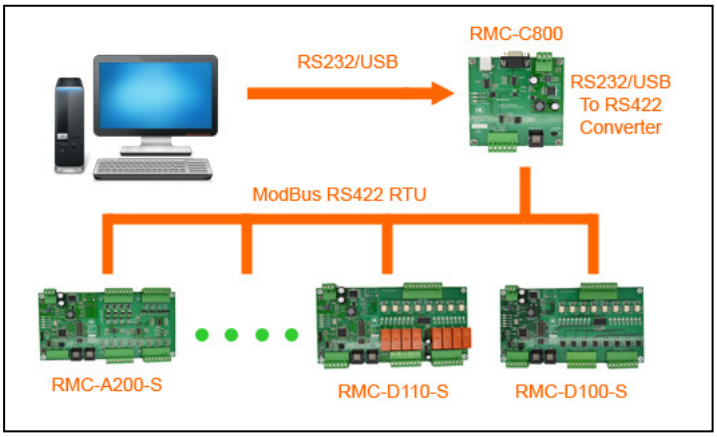
- Low Cost
- Wide selection of modules
- Serial Full-Duplex RS485 (4-wire RS485) or Ethernet TCP/IP interface
- Multiple mounting options
- Multiple network options
- Powerful 32-bit microcontroller on board
- Single 24V DC power input
- Pluggable screw terminals
- Dual RJ45 plugs for RS485 network

3. Network configuration options

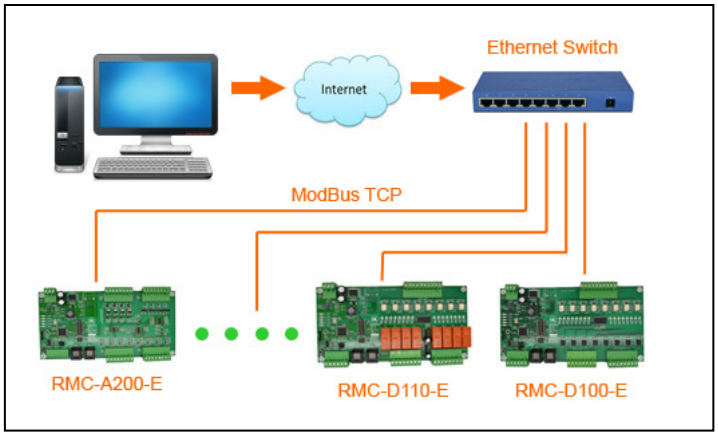
- **Embedded Control**
  In this configuration, a main controller board is used to bridge between a touch screen interface and the Modbus modules.



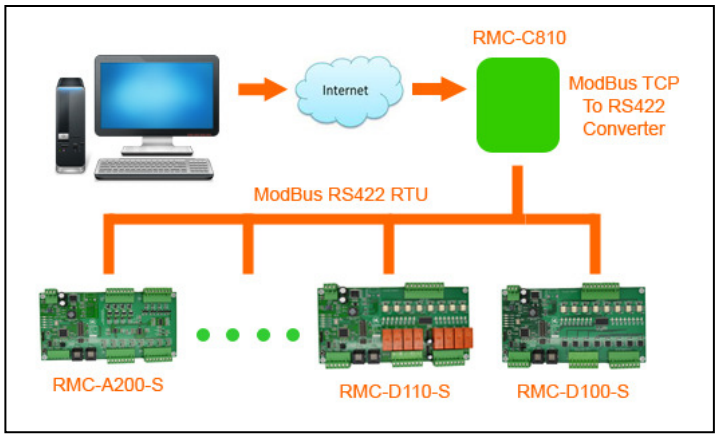
- **Full-Duplex RS485 bus**
  In this configuration, RMC-C800 interface module is used to convert the master RS232/USB to Slave Full-Duplex RS485 bus. Up to 247 slaves can be daisy-chained on the bus.



- **ModBus TCP**
  In this configuration, all modules are set with ModBus TCP option. An Ethernet switch or router is used to communicate with each module over local Ethernet network or the internet.



- **Master ModBus TCP To Slave RS485**
  In this configuration, RMC-C810 interface module is used to convert the master Modbus TCP to Slave Full-Duplex RS485 bus. Up to 247 slaves can be daisy-chained on the bus. This is the recommended configuration for ModBus TCP network as only one IP address is required to communicate with a group of slaves over a local Ethernet network or the internet.



4. Full-Duplex vs Half-Duplex RS485 Bus

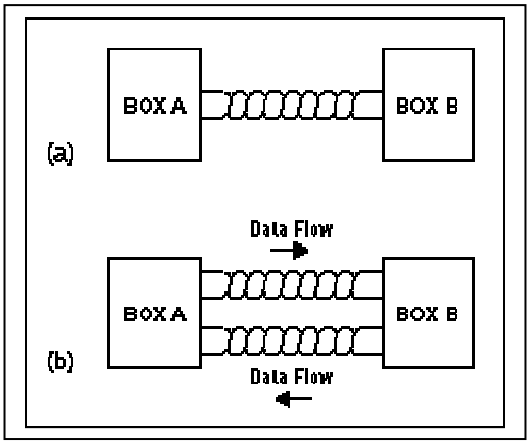
Full-Duplex RS485 is also called RS422

RS-485 bus standard is one of the most widely used physical layer bus designs in Industrial and instrumentation (I&I) applications.

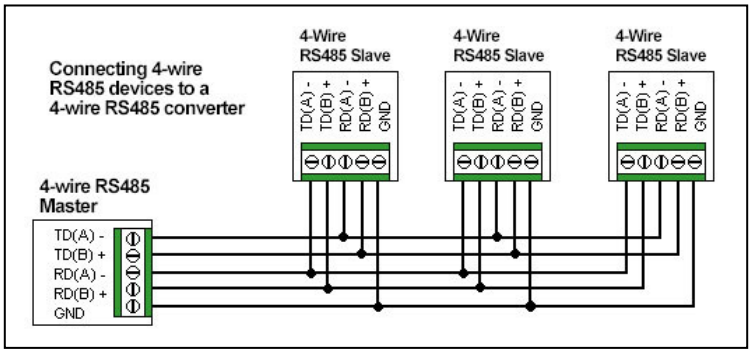
The key features of RS-485 that make it ideal for use in I&I communications applications are

- Long distance links—up to 4000 feet.
- Bidirectional communications possible over a single pair of twisted cables.
- Differential transmission increases noise immunity and decreases noise emissions.
- Multiple drivers and receivers can be connected on the same bus.
- Wide common-mode range allows for differences in ground potential between the driver and receiver.

RS-485 is designed in such a way that only one transmitter on a twisted pair can be active at a time. With this constraint, in the system shown in the figure below, Box A can transmit data to Box B *or* Box B can transmit data to Box A, but *both* can't transmit data at the same time. This is referred to as "half duplex." A full-duplex system, on the other hand, would allow communications in both directions simultaneously. Full-duplex systems can be designed using RS-485, but require two twisted-pair cables running between nodes, as shown in figure below. One twisted pair is dedicated to transmitting information in one direction, and the other twisted pair is dedicated to transmitting data in the opposite direction.



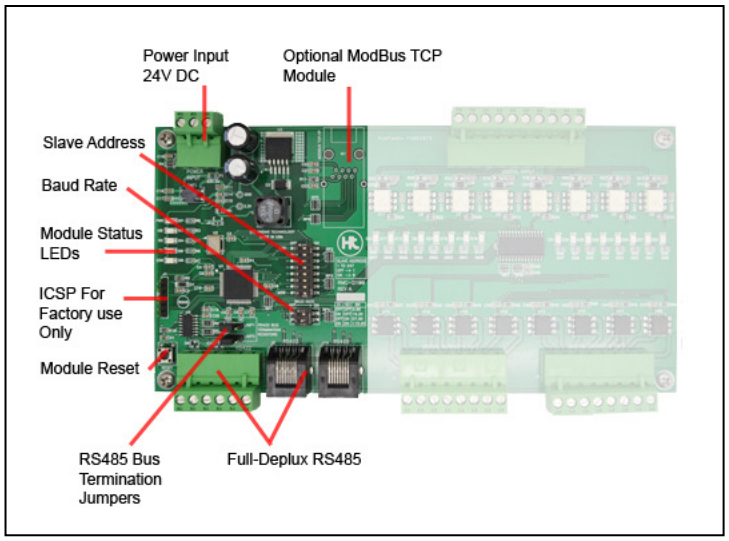
The figure below shows Full-Duplex RS485 (also known as a 4-wire RS-485) network connected in a multipoint master/slave configuration. Full-duplex RS-485 allows for simultaneous communication in both directions between master and slave nodes



Haidar RMC series uses Full-Duplex RS485 (4W RS485) bus to link multiple slaves with one master. The data on the master pair (TD(A)- and TD(B)+) are only received by the slaves; the data on the slave pair (RD(A)- and RD(B)+) are only received by the only master. At any time, one slave has the right to send data to the master. Fifth wire must also be used as common which connect the GND of all slaves on the bus to the master GND.

5. RMC Series Common Hardware

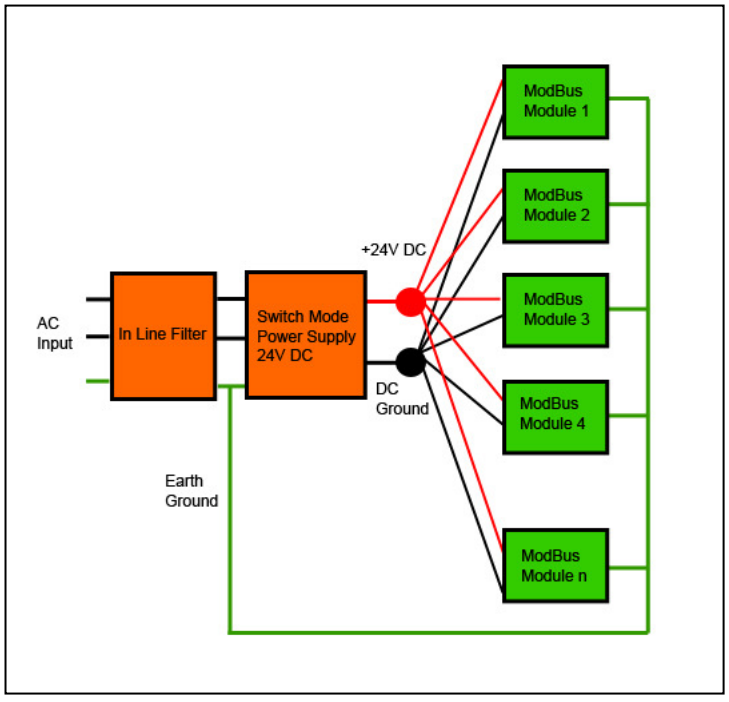
All Haidar RMC series modules share the same controller section as shown in the figure below. This common section include the main 32-bit controller, power filter and converter, Modbus TCP module, Status LEDs, RS485 interface and slave address and baud rate switches.



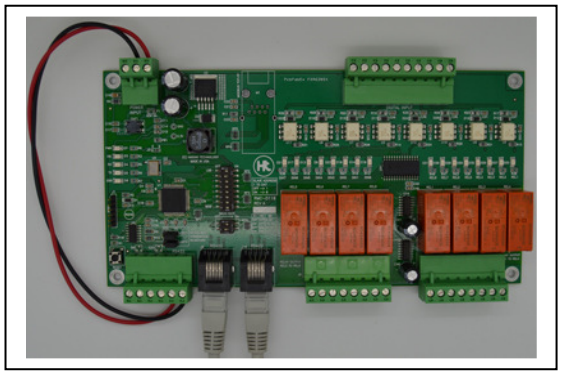
6. RMC Series Power Input and Grounding

As shown in the figure below, each module is powered from the main 24V DC power supply via a twisted pair of wires, one for the power GND (Black) and the other for the +24V (Red). AC filter is recommended to reduce the AC noise. Grounding the modules is also recommended. An earth ground wire (Green) is used to connect all the modules earth ground pins to the AC ground.

The Four mounting holes are connected directly to earth ground and connected to power ground via a 10 Ohm (R4) resistor and 0.01uF (C18) capacitor.



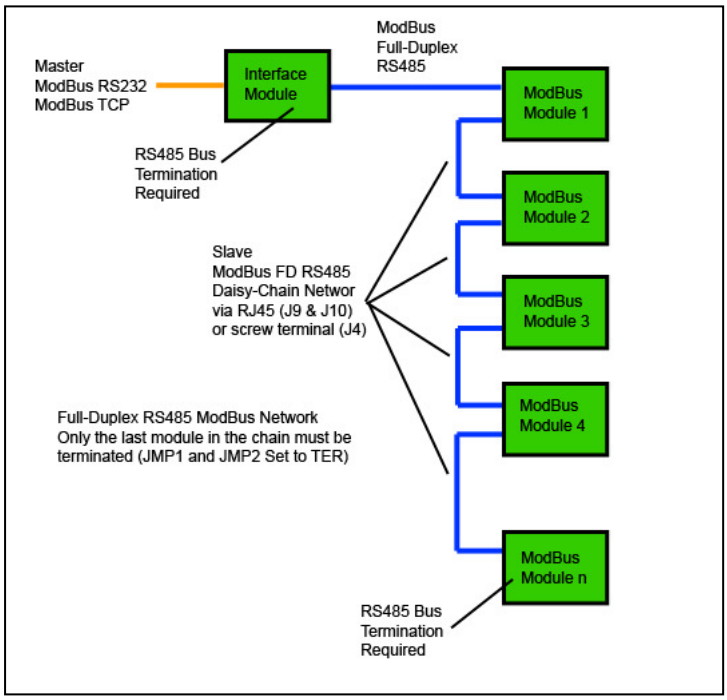
Modules can be also powered directly from the Full-Duplex RS485 cable. As shown in the figure below, two wires are required to connect J4 VBUS and GND to the module main power input J2.



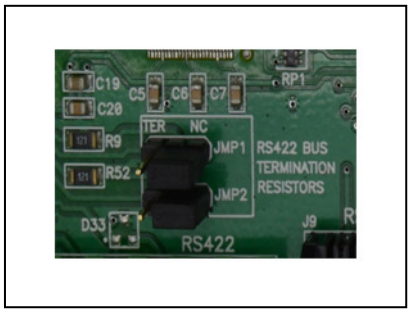
This power scheme is recommended for testing purposes and is not recommended for real system operation.

7. RMC Series Multipoint Serial Bus

As shown in the figure below, slaves are daisy-chained using 5 wire cable. An interface module is used to convert the master RS232, USB or Ethernet TCP/IP to Full-Duplex RS485. Each module (slave) has two RJ45 (J9 & J10) connectors which can be used to link the slaves to the master using standard Ethernet cables. A pluggable 6-pin screw terminal (J4) is also can be used to daisy-chain the slaves using standard 5-wire cable.

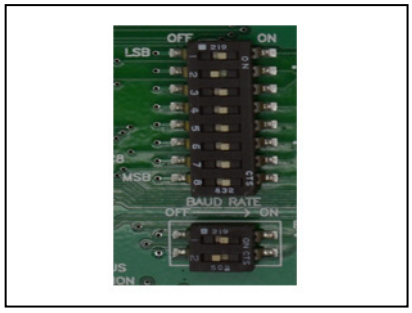


Lines termination is required at each extremity of the bus. On board JMP1 and JUMP2 are used to terminate the bus using 120 ohm resistors.



8. Slave Address and Baud Rate

According to ModBus protocol, each slave must have unique address called Slave Address. The individual slave modules are assigned addresses in the range of 1 – 247. Address 0 is reserved for the Broadcast mode where the master can send request to all salves. On board dip-switch is used to set the individual slave address.



S1(LSB)	S2	S3	S4	S5	S6	S7	S8 (MSB)	Address
1 (OFF)	0 (ON)	0 (ON)	0 (ON)	0 (ON)	0 (ON)	0 (ON)	0 (ON)	1
0	1	0	0	0	0	0	0	2
1	1	0	0	0	0	0	0	3
0	1	1	0	1	1	1	1	246
1	1	1	0	1	1	1	1	247

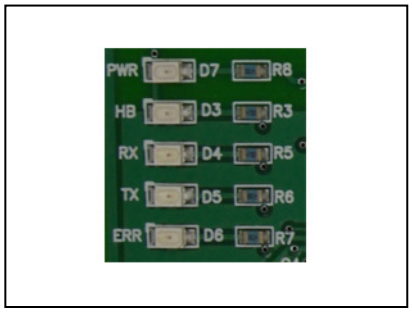
The Default slave address is 1.

Communication speed or baud rate is set by the on-board baud rate dip-switch. All slaves must have the same baud rate and it must be equal to the master baud rate. Default is 19.2 kb/sec.

S1	S2	Baud Rate
OFF	OFF	9.6 kb/sec
ON	OFF	19.2 kb/sec (Default)
OFF	ON	57.6 kb/sec
ON	ON	115.2 kb/sec

9. Status LED's

Five LED's are used to show the status of the module.



LED	Color	Status Description
PWR	Green	ON when power is applied
HB	Green	Heart Beat. Normally flashes 10 times per second. If the HB LED is OFF or Solid ON then the main controller is not working.
RX	Orange	ON when data is being received
TX	Orange	ON when data is being transmitted
ERR	Red	ON when ModBus communication error occurred. • Slave Address is out of range • Illegal Function (Error code 1) • Illegal Data Address (Error code 2) • Illegal Data Value (Error code 3) • Slave Device Fail (Error code 4)

10. ModBus TCP Option

Ethernet ModBus TCP option is available for all slave modules. Each slave device can be equipped with intelligent Modbus TCP module which converts the serial ModBus RTU device into an Ethernet ModBus TCP device.

The XPORT-MBTCP module form Lantronix is an intelligent RJ45 connector with ModBus RTU/ASCII Master or Slave software built-in. It supports both 10/100M Ethernet connections.
Each device connected to the TCP/IP network must have a unique IP address. When multiple Modbus devices share a single IP, then Modbus TCP includes an additional address called the Unit ID. When the XPORT Server is receiving Modbus TCP messages from remote masters, the Unit ID is converted to use in the Modbus/RTU message as the slave address.

DeviceInstaller configuration utility from Lantronix is used to setup the XPORT module. XPORT must be configured as shown in the table below:

XPORT Setting Name	Setting Value
IP Address	User defined
Default Gateway	User defined
Netmask	User defined
Protocol	RTU, Slave (See note 1)
Serial Interface	19200, 8, N, 2, RS232 (See note 2)
RTS/CTS Mode	Fixed (See note 3)
Slave Address	ModBus TCP Header
Broadcast	Disabled
Pipline	Enabled
TCP Exception codes	Disabled
Char, Message Time Out	50msec, 500msec
CP1 Function	Unused
CP2 Function	Unused
CP3 Function	Unused

Notes:

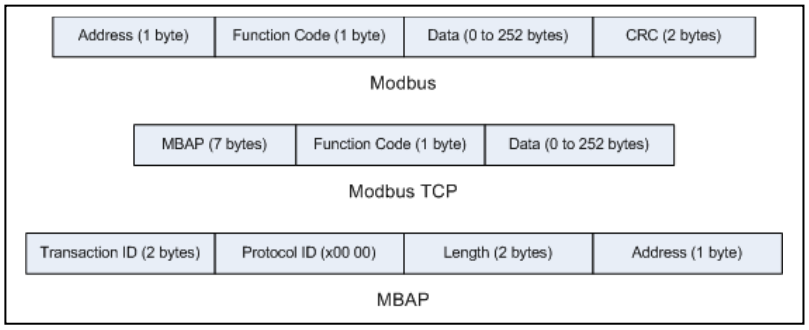
1) Only ModBus RTU is supported.
2) The Serial interface settings are fixed to
    - 19200 baud rate
    - 8-bit data
    - No parity
    - Two Stop bits
3) No modem handshaking

11. ModBus Overview

The Modbus protocol was developed in 1979 by Modicon, Incorporated, for industrial automation systems and Modicon programmable controllers. It has since become an industry standard method for the transfer of discrete/ analog I/O information and register data between industrial control and monitoring devices. Modbus is now a widely-accepted, open, public-domain protocol that requires a license, but does not require royalty payment to its owner.

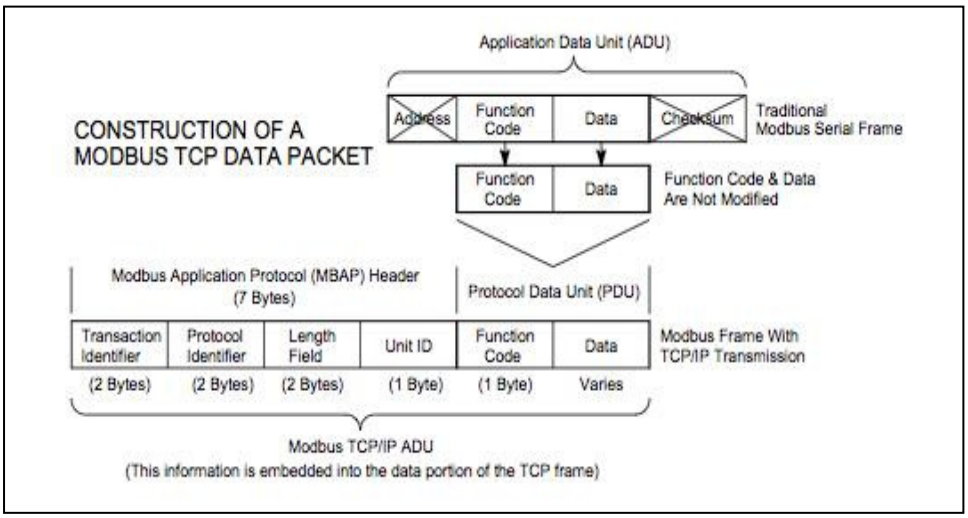
Modbus devices communicate using a master-slave (client-server) technique in which only one device (the master/client) can initiate transactions (called queries). The other devices (slaves/servers) respond by supplying the requested data to the master, or by taking the action requested in the query. A slave is any peripheral device (I/O transducer, valve, network drive, or other measuring device) which processes information and sends its output to the master using Modbus.

The Haidar I/O Modules form slave/server devices, while a typical master device is a host computer running appropriate application software.
Masters can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a response to all queries addressed to them individually, but do not respond to broadcast queries. Slaves do not initiate messages on their own, they only respond to queries from the master. A master's query will consist of a slave address (or broadcast address), a function code defining the requested action, any required data, and an error checking field. A slave's response consists of fields confirming the action taken, any data to be returned, and an error checking field. Note that the query and response both include a device address, a function code, plus applicable data, and an error checking field. If no error occurs, the slave's response contains the data as requested. If an error occurs in the query received, or if the slave is unable to perform the action requested, the slave will return an exception message as its response (see Modbus Exceptions). The error check field of the slave's message frame allows the master to confirm that the contents of the message are valid. Traditional Modbus messages are transmitted serially and parity checking is also applied to each transmitted character in its data frame.



Modbus TCP/IP (also Modbus-TCP) is simply the Modbus RTU protocol with a TCP interface that runs on Ethernet. The Modbus messaging structure is the *application protocol* that defines the rules for organizing and interpreting the data independent of the data transmission medium.
TCP/IP refers to the Transmission Control Protocol and Internet Protocol, which provides the transmission medium for Modbus TCP/IP messaging.
Simply stated, TCP/IP allows blocks of binary data to be exchanged between computers. It is also a world-wide standard that serves as the foundation for the World Wide Web. The primary function of TCP is to ensure that all packets of data are received correctly, while IP makes sure that messages are correctly addressed and routed. Note that the TCP/IP combination is merely a *transport protocol, a*nd does not define what the data means or how the data is to be interpreted (this is the job of the application protocol, Modbus in this case).
So in summary, Modbus TCP/IP uses TCP/IP and Ethernet to carry the data of the Modbus message structure between compatible devices. That is, Modbus TCP/IP combines a physical network (Ethernet), with a networking standard (TCP/IP), and a standard method of representing data (Modbus as the application protocol). Essentially, the Modbus TCP/IP message is simply a Modbus communication encapsulated in an Ethernet TCP/IP wrapper.

In practice, Modbus TCP embeds a standard Modbus data frame into a TCP frame, without the Modbus checksum, as shown in the following diagram.



12. RMC Series ModBus Protocol

- **ModBus Mode:** RMC series Only supports RTU mode

- **Serial Interface Settings:** The table below shows the format for each byte (11 bits) in RTU mode.

Data	8-bit, least significant bit sent first
Start bits	1 bits
Parity bits	None
Stop bits	2 bits
Baud Rate	9600, 19200 (Default), 57600 and 115200 b/sec

- **Supported Functions:** The Modbus specification defines a set of function codes that specify how to read and write bits or words (16 bits). Modbus uses the term "coil" to refer to a bit, and "register" to refer to a word.
  The RMC Series supports the following function codes:

Function Code (Decimal)	**Function**	**Description**
1	Read Coil Status	Read one or more bits
2	Read Discrete Inputs	Read one or more bits
3	Read Holding Registers	Read one or more holding registers
4	Read Input Registers	Read one or more input registers
5	Write Single Coil	Set one bit
6	Write Single Output Register	Set one output register
15	Write Multiple Coils	Set multiple bits
16	Write Multiple Output Registers	Set multiple output registers
17	Report Slave ID	Read Slave specific Information

In RMC, there is no difference between function 1 and 2, and there no difference between Function 3 and 4.

- **ModBus Exceptions:** The RMC Series supports the following exception codes:

Exception Code (Decimal)	Exception	Description
1	Illegal Function	The function code received is not allowable action for the slave.
2	Illegal Data Address	The data address received is not allowable address.
3	Illegal Data Value	The data value received is not allowable value.
4	Slave Device Fail	An error occurred while the slave (server) was attempting to perform requested action.

- **Message Error Checking:** The RTU mode includes an error-checking field that is based on a Cyclical Redundancy Checking (CRC) method performed on the message contents. The CRC field contains a 16-bit value implemented as two 8-byts.
  The CRC field is appended to the message as the last filed in the message. When this is done, the Low-Order (LSB) of the field is appended first, followed by the High-Order (MSB). The CRC high-order byte is the last byte to be sent in the message.

- **Exception Response:** In a normal response, the slave simply echoes the function code of the original query in the function field of the response. All function codes have their most-significant bit (msb) set to 0 (their values are below 80H). In an exception response, the slave sets the msb of the function code to 1 in the returned response (i.e. exactly 80H higher than normal) and returns the exception code in the data field. This is used by the client/master application to actually recognize an exception response and to direct an examination of the data field for the applicable exception code.

  ModBus Error PDU Response

Error Code	Function Code + 80H
Exception Code	1, 2, 3, or 4

- **Supported Data Types (RMC Modules):** All I/O values are accessed via 16-bit Input Registers or 16-bit Output Registers (see Register Map). Input registers contain information that is read-only. For example, the current input value read from a channel, or the states of a group of digital inputs. Output registers contain read/write information that may be configuration data or output data. For example, the high limit value of an alarm function operating at an input, or an output value or an output channel. I/O values of RMC modules take the following common forms of data to represent temperature and discrete on/off, as required. This is not a Modbus standard and will vary between devices.

Data Type	Description
Count Value (Signed Integer)	A 16-bit signed integer value representing an A/D count, a DAC count, time value, or frequency with a range of –32768 to +32767.
Count Value (Unsigned Integer)	A 16-bit unsigned integer value representing an A/D count, a DAC count, time value, or frequency with a range of 0 to 65535.
Temperature (Signed Integer)	A 16-bit signed integer value with resolution of 0.1°C/lsb. For example, a value of 12059 is equivalent to 1205.9°C, a value of –187 equals –18.7°C. The maximum possible temperature range is –3276.8°C to +3276.7°C.
Discrete (One Bit)	A discrete value is generally indicated by a single bit of a 16-bit word. The bit number/position typically corresponds to the discrete channel number for this model. Unless otherwise defined for outputs, a 1 bit means the corresponding output is closed or ON, a 0 bit means the output is open or OFF.

	For inputs, a value of 1 means the input is in its high state (usually >> 0V), while a value of 0 specifies the input is in its low state (near 0V).
Unsigned Long	A 32-bit unsigned value with a range from 0 to (2^32 – 1).
Signed Long	A 32-bit signed value with a range from –2^31 to (2^31 – 1).

- **ModBus Registers:** The Modbus data model has a simple structure that only differentiates between four basic data types:
  - Discrete Inputs
  - Discrete Outputs (Coils)
  - Input Registers (Input Data)
  - Holding (Output) Registers (Output Data)

  The Modbus registers of a device are organized around the four basic data reference types noted above and this data type is further identified by the leading number of the reference address as follows:

Reference	Description
0x0000	Read/Write Discrete Outputs or Coils. A 0x reference address is used to drive output data to a digital output channel.
1x0000	Read Discrete Inputs. The ON/OFF status of a 1x reference address is controlled by the corresponding digital input channel.
3x0000	Read Input Registers. A 3x reference register contains a 16-bit number received from an external source—e.g. an analog signal.
4x0000	Read/Write Output or Holding Registers. A 4x register is used to store 16-bits of numerical data (binary or decimal), or to send the data from the CPU to an output channel.

IMPORTANT NOTE: The reference addresses noted in the memory map are not explicit hard-coded memory addresses. Internally, all Modbus devices use a zero-based memory offset computed from the reference address. However, the system interface of Modbus systems (software) will vary in this regard and may require you to enter the actual reference address, drop the leading number, or enter an absolute memory offset from 1, or a memory address offset from 0. This is system dependent and a common source of programming errors. Be wary of this when writing higher-level application programs to access these registers.
Note that not all Modbus functions operate on register map registers. All data addresses in Modbus messages are referenced to 0, with the first occurrence of a data item addressed as item number zero. Further, a function code field already specifies which register group it operates on (i.e. 0x, 1x, 3x, or 4x reference addresses). For example, holding register 40001 is addressed as register 0000 in the data address field of the message. The function code that operates on this register specifies a “holding register” operation and the “4xxxx” reference group is implied. Thus, holding register 40108 is actually addressed as register 006BH (107 decimal).

13. Supported Functions

- **Function Code 01/02: Read Coil/Input Status**

  In the RMC series, function codes 1 and 2 are identical. These two functions read bit values starting at a given address.

  This example reads the output channel status coils 0 to 3.

  Modbus PDU Request

Field Name	Example Decimal (Hexadecimal)
Function code	1 (01)
Starting Address High	0 (00)
Starting Address Low	0 (00)
Length High	0 (00)
Length Low	0 (04)

  Modbus PDU Response

Field Name	Example Decimal (Hexadecimal)
Function code	1 (01)
Byte Count	1 (01)
Data (Coils 0-3)	10 (0A)

  The *start address* is the address of the first bit to read. *Length* is the number of bits to read, however, the response will always be in multiples of 8 bits. For example, if *length* is 5, the response will contain 8 bits. If *length* is 14, the response will be 16 bits.

- **Function Code 03/04: Read Holding/Input Register**

  In the RMC series, function codes 3 and 4 are identical. These two functions read registers values starting at a given address.

  This example reads Holding or Input Registers 6 to 8.

  Modbus PDU Request

Field Name	Example Decimal (Hexadecimal)
Function code	3 (03)
Starting Address High	0 (00)
Starting Address Low	5 (05)
Length High	0 (00)
Length Low	3 (03)

  Modbus PDU Response

Field Name	Example Decimal (Hexadecimal)
Function code	3 (03)
Byte Count	6 (06)
Data High (Register 6)	0 (00)
Data Low (Register 6)	255 (FF)
Data High (Register 7)	1 (01)
Data Low (Register 7)	15 (0F)
Data High (Register 8)	10 (0A)
Data Low (Register 8)	16 (10)

  Note that registers are addressed starting at 0 (registers 1-16 addressed as 0-15).

- **Function Code 05: Write Single Coil**
  This function sets the value of a single bit at a given address.

  This example forces discrete output 3 ON.

  Modbus PDU Request

Field Name	**Example Decimal (Hexadecimal)**
Function code	5 (05)
Starting Address High	0 (00)
Starting Address Low	3 (03)
Data High	255 (FF)
Data Low	0 (00)

  Modbus PDU Response

Field Name	**Example Decimal (Hexadecimal)**
Function code	5 (05)
Starting Address High	0 (00)
Starting Address Low	3 (03)
Data High	255 (FF)
Data Low	0 (00)

  Start address is the address of the bit to set, and *data* is the value to set the bit to. To turn a bit ON, *data* must be 0xFF00. To turn a bit OFF, *data* must be 0x0000.

- **Function Code 06: Write Single Holding (Output) Register**
  This function sets the value of a single output register at a given address.

  This example set the value of output register 0 to FFFF.

  Modbus PDU Request

Field Name	**Example Decimal (Hexadecimal)**
Function code	6 (06)
Starting Address High	0 (00)
Starting Address Low	0 (00)
Data High	255 (FF)
Data Low	255 (FF)

  Modbus PDU Response

Field Name	**Example Decimal (Hexadecimal)**
Function code	6 (06)
Starting Address High	0 (00)
Starting Address Low	0 (00)
Data High	255 (FF)
Data Low	255 (FF)

- **Function Code 15: Write Multiple Coils**
  This function sets multiple discrete outputs (Coils) at given address.

  This example forces outputs 1&3 OFF and 0&2 ON for coils 0-3

  Modbus PDU Request

Field Name	**Example Decimal (Hexadecimal)**
Function code	15 (0F)
Starting Address High	0 (00)
Starting Address Low	0 (00)
Length High	0 (00)
Length Low	4 (04)
Byte Count	1 (01)
Data High	5 (05)

  Modbus PDU Response

Field Name	**Example Decimal (Hexadecimal)**
Function code	15 (0F)
Starting Address High	0 (00)
Starting Address Low	0 (00)
Length High	0 (00)
Length Low	4 (04)

  The *start address* is the address of the first bit to write. *Length* is the number of bits to write,
  Byte Count = N * 1 Byte
  Where, N = Length / 8, if the remainder is different of 0 => N = N + 1

- **Function Code 16: Write Multiple Holding (Output) Registers**
  This function sets multiple output registers at given address.

  This example sets the output registers 0-2

  Modbus PDU Request

Field Name	**Example Decimal (Hexadecimal)**
Function code	16 (10)
Starting Address High	0 (00)
Starting Address Low	0 (00)
Length High	0 (00)
Length Low	3 (03)
Data High (Register 0)	0 (00)
Data Low (Register 0)	0 (00)
Data High (Register 1)	255 (FF)
Data Low (Register 1)	255 (FF)
Data High (Register 2)	0 (00)
Data Low (Register 2)	5 (05)

  Modbus PDU Response

Field Name	**Example Decimal (Hexadecimal)**
Function code	16 (10)
Starting Address High	0 (00)
Starting Address Low	0 (00)
Length High	0 (00)
Length Low	3 (03)

- **Function Code 17: Report Slave ID**

  This command returns the model, serial, and firmware number for slave/server device, the status of the Run indicator, and any other information specific to the device. This command does not address Register Map.

Modbus PDU Request

Field Name	**Example Decimal (Hexadecimal)**
Function code	17 (11)

Modbus PDU Response

Field Name	**Example Decimal (Hexadecimal)**
Function code	17 (11)
Byte Count	18 (12)
Slave ID	Device Specific For example: 10 = RMC-D100 11 = RMC-D110 20 = RMC-A200 30 = RMC-T300
Run Indicator	0 (00) = OFF, 255 (FF) = ON
Vendor Name	"H"
Firmware REV Major	Device Specific For example: 1
Firmware REV Minor	Device Specific For example: 0
Hardware REV	Device Specific For example: "A"
Connection Type	0 = RS485, 1 = TCP/IP
Baud Rate	0 = 115.2 kb/Sec 1 = 57.6 kb/sec 2 = 19.2 kb/sec 3 = 9.6 kb/sec
Not Used	0
Input Register Count	Device Specific For example: 8
Holding (Output) Register Count	Device Specific For example: 16
Discrete Input Count	Device Specific For example: 8
Discrete Output Count	Device Specific For example: 8
Not Used	0
Not Used	0
Not Used	0
Not Used	0
Not Used	0

14. Manual Change History

Date	Revision	Change
6/29/2015	REV1.00	Initial version of this manual

15. Hardware Limited Warrnty

Haidar Technology, LLC. Warrants its hardware products to be free from manufacturing defects in materials and workmanship under normal use for a period of one (1) year from the date of purchase from HAIDAR. This warranty extends to products purchased directly from HAIDAR or an authorized HAIDAR distributor. Purchasers should inquire of the distributor regarding the nature and extent of the distributor's warranty, if any. HAIDAR shall not be liable to honor the terms of this warranty if the product has been used in any application other than that for which it was intended, or if it has been subjected to misuse, accidental damage, modification, or improper installation procedures. Furthermore, this warranty does not cover any product that has had the serial number altered, defaced, or removed. This warranty shall be the sole and exclusive remedy to the original purchaser. In no event shall HAIDAR be liable for incidental or consequential damages of any kind (property or economic damages inclusive) arising from the sale or use of this equipment. HAIDAR is not liable for any claim made by a third party or made by the purchaser for a third party. HAIDAR shall, at its option, repair or replace any product found defective, without charge for parts or labor. Repaired or replaced equipment and parts supplied under this warranty shall be covered only by the unexpired portion of the warranty. Except as expressly set forth in this warranty, HAIDAR makes no other warranties, expressed or implied, nor authorizes any other party to offer any warranty, including any implied warranties of merchantability or fitness for a particular purpose. Any implied warranties that may be imposed by law are limited to the terms of this limited warranty. This warranty statement supercedes all previous warranties, and covers only the HAIDAR hardware.

16. Returns and Repair Policy

No merchandise may be returned for credit, exchange, or service without prior authorization from. To obtain warranty service, contact the factory and request an RMA (Return Merchandise Authorization) number. Enclose a note specifying the nature of the problem, name and phone number of contact person, RMA number, and return address. Authorized returns must be shipped freight prepaid to HAIDAR Technology with the RMA number clearly marked on the outside of all cartons. Shipments arriving freight collect or without an RMA number shall be subject to refusal. HAIDAR reserves the right in its sole and absolute discretion to charge a 15% restocking fee, plus shipping costs, on any products returned with an RMA.

Return freight charges following repair of items under warranty shall be paid by HAIDAR, shipping by standard ground carrier. In the event repairs are found to be non-warranty, return freight costs shall be paid by the purchaser.