MODBUS TCP/RTU for 8 * DI & 8 * DO

1. Introduction

MODBUS TCP Command & Response format

A basic MODBUS command always takes two main parameters：

1. ID Address：Designated to receive this Command ID address。
2. Function Code：This Command function。

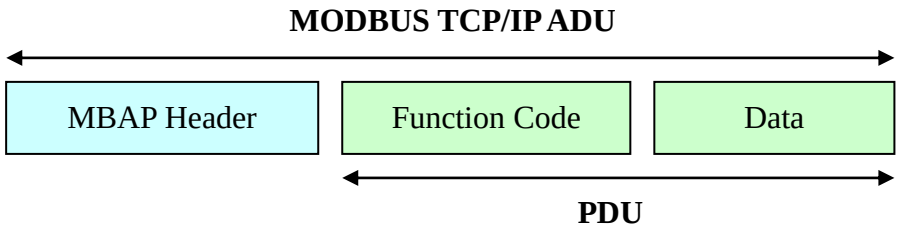
The reception to the Command element will return a Response inform the completion of action in response to the remote or the value returned by the read, Response Command format in the format is basically the same, but also have ID Address and Function code in order to master identification, the following table for common Function code order.

2. MODBUS TCP Protocol

The following MODBUS functions are supported.

Function code	Description
0x01	Read Coils – Read Digital Output
0x02	Read Discrete Inputs
0x05	Write Single Coils – Preset single DO

MODBUS On TCP/IP Application Data Unit



The MBAP Header contains the following fields:

Fields	Length	Description	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

The header is 7 bytes long:

Transaction Identifier - It is used for transaction pairing, the MODBUS server copies in the response the transaction identifier of the request.

Protocol Identifier – It is used for intra-system multiplexing. The MODBUS protocol is identified by the value 0.

Length - The length field is a byte count of the following fields, including the Unit Identifier and data fields.

Unit Identifier – This field is used for intra-system routing purpose. It is typically used to communicate to a MODBUS+ or a MODBUS serial line slave through a gateway between an Ethernet TCP-IP network and a MODBUS serial line. This field is set by the MODBUS Client in the request and must be returned with the same value in the response by the server.

All MODBUS/TCP ADU are sent via TCP to registered port 502.

MODBUS Protocol Address Map

MODBUS Register	HEX	Function	Despcription	Action
00001	0000	Read/Write DO	Starting Address 0000	R/W
10001	0000	Read DI	Starting Address 0000	R

Read DO status：

Example for read Digital Output

Request		Response	
Field Name	(Hex)	Field Name	(Hex)
Transaction Identifier Hi	00	Transaction Identifier Hi	00
Transaction Identifier Lo	00	Transaction Identifier Lo	00
Protocol Identifier Hi	00	Protocol Identifier Hi	00
Protocol Identifier Lo	00	Protocol Identifier Lo	00
Length Hi	00	Length Hi	00
Length Lo	06	Length Lo	04
MODBUS Address	01	MODBUS Address	01
Function	01	Function	01
Starting Address Hi	00	Byte Count	01
Starting Address Lo	00	Outputs status 03-00	8A
Quantity of Outputs Hi	00		
Quantity of Outputs Lo	08		

DO1 is OFF
DO2 is ON
DO3 is OFF
DO4 is ON
DO5 is OFF
DO6 is OFF
DO7 is OFF
DO8 is ON

Read DI status：

Example for read Digital Input

Request		Response	
Field Name	(Hex)	Field Name	(Hex)
Transaction Identifier Hi	00	Transaction Identifier Hi	00
Transaction Identifier Lo	00	Transaction Identifier Lo	00
Protocol Identifier Hi	00	Protocol Identifier Hi	00
Protocol Identifier Lo	00	Protocol Identifier Lo	00
Length Hi	00	Length Hi	00
Length Lo	06	Length Lo	04
MODBUS Address	01	MODBUS Address	01
Function	02	Function	02
Starting Address Hi	00	Byte Count	01
Starting Address Lo	00	Inputs status 07-00	12
Quantity of Inputs Hi	00		
Quantity of Inputs Lo	08		

DI1 is OFF
DI2 is ON
DI3 is OFF
DI4 is OFF
DI5 is ON
DI6 is OFF
DI7 is OFF
DI8 is OFF

Write DO status：

Here is example of a request to write DO1 ON：

Request		Response	
Field Name	(Hex)	Field Name	(Hex)
Transaction Identifier Hi	00	Transaction Identifier Hi	00
Transaction Identifier Lo	00	Transaction Identifier Lo	00
Protocol Identifier Hi	00	Protocol Identifier Hi	00
Protocol Identifier Lo	00	Protocol Identifier Lo	00
Length Hi	00	Length Hi	00
Length Lo	06	Length Lo	06
MODBUS Address	01	MODBUS Address	01
Function	05	Function	05
Output Address Hi	00	Output Address Hi	00
Output Address Lo	00	Output Address Lo	00
Output value Hi	FF	Output value Hi	FF
Output value Lo	00	Output value Lo	00

For MODBUS RTU

CRC-16

/* The function returns the CRC as a type unsigned short. */

/* CRC Generation Function */

unsigned short CRC16 (puchMSG, usDataLen)

unsigned char *puchMsg： /* message to calculate CRC upon */

unsigned short usDataLen： /* quantity of bytes in message */

```
{
    unsigned char uchCRCHi = 0xFF; /* high CRC byte initialized */
    unsigned char uchCRCLo = 0xFF; /* low CRC byte initialized */
    unsigned uIndex; /* will index index into CRC lookup */
    while (usDataLen--) /* pass through message buffer */
    {
        uIndex=uchCRCHi ^ *puchMsgg++； */ calculate the CRC */
        uchCRCHi = uchCRCLo ^auchCRCHi〔uIndex〕;
        uchCRCLo = auchCRCLo〔uIndex〕;
    }
    return（uchCRCHi<< 8 | uchCRCLo）;
}
```

High Order Byte Table

```
/* Table of CRC values for high - order byte */
static unsigned char auchCRCHi [ ] = {
0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x01 , 0xC0 ,
0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 ,
0x00 , 0xC1 , 0x81 , 0x40 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 ,
0x80 , 0x41 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 ,
0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x00 , 0xC1 ,
0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x01 , 0xC0 , 0x80 , 0x41 ,
0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x00 , 0xC1 ,
0x81 , 0x40 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 ,
0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x01 , 0xC0 ,
0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x00 , 0xC1 , 0x81 , 0x40 ,
0x01 , 0xC0 , 0x80 , 0x41 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x00 , 0xC1 ,
0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 ,
0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x01 , 0xC0 ,
0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 ,
0x00 , 0xC1 , 0x81 , 0x40 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 ,
0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 ,
0x01 , 0xC0 , 0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 ,
0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x00 , 0xC1 , 0x81 , 0x40 ,
0x01 , 0xC0 , 0x80 , 0x41 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x00 , 0xC1 ,
0x81 , 0x40 , 0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 ,
0x00 , 0xC1 , 0x81 , 0x40 , 0x01 , 0xC0 , 0x80 , 0x41 , 0x01 , 0xC0 ,
0x80 , 0x41 , 0x00 , 0xC1 , 0x81 , 0x40
} ;
```

Low Order Byte Table

```
/* Table of CRC values for low - order byte */
static char auchCRCLo [ ] = {
0x00 , 0xC0 , 0xC1 , 0x01 , 0xC3 , 0x03 , 0x02 , 0xC2 , 0xC6 , 0x06 ,
0x07 , 0xC7 , 0x05 , 0xC5 , 0xC4 , 0x04 , 0xCC , 0x0C , 0x0D , 0xCD ,
0x0F , 0xCF , 0xCE , 0x0E , 0x0A , 0xCA , 0xCB , 0x0B , 0xC9 ,
0x09 ,
0x08 , 0xC8 , 0xD8 , 0x18 , 0x19 , 0xD9 , 0x1B , 0xDB , 0xDA , 0x1A ,
0x1E , 0xDE , 0xDF , 0x1F , 0xDD , 0x1D , 0x1C , 0xDC , 0x14 ,
0xD4 ,
0xD5 , 0x15 , 0xD7 , 0x17 , 0x16 , 0xD6 , 0xD2 , 0x12 , 0x13 , 0xD3 ,
0x11 , 0xD1 , 0xD0 , 0x10 , 0xF0 , 0x30 , 0x31 , 0xF1 , 0x33 , 0xF3 ,
0xF2 , 0x32 , 0x36 , 0xF6 , 0xF7 , 0x37 , 0xF5 , 0x35 , 0x34 , 0xF4 ,
0x3C , 0xFC , 0xFD , 0x3D , 0xFF , 0x3F , 0x3E , 0xFE , 0xFA ,
0x3A ,
0x3B , 0xFB , 0x39 , 0xF9 , 0xF8 , 0x38 , 0x28 , 0xE8 , 0xE9 , 0x29 ,
0xEB , 0x2B , 0x2A , 0xEA , 0xEE , 0x2E , 0x2F , 0xEF , 0x2D ,
0xED ,
0xEC , 0x2C , 0xE4 , 0x24 , 0x25 , 0xE5 , 0x27 , 0xE7 , 0xE6 , 0x26 ,
0x22 , 0xE2 , 0xE3 , 0x23 , 0xE1 , 0x21 , 0x20 , 0xE0 , 0xA0 , 0x60 ,
0x61 , 0xA1 , 0x63 , 0xA3 , 0xA2 , 0x62 , 0x66 , 0xA6 , 0xA7 , 0x67 ,
0xA5 , 0x65 , 0x64 , 0xA4 , 0x6C , 0xAC , 0xAD , 0x6D , 0xAF ,
0x6F ,
0x6E , 0xAE , 0xAA , 0x6A , 0x6B , 0xAB , 0x69 , 0xA9 , 0xA8 ,
0x68 ,
0x78 , 0xB8 , 0xB9 , 0x79 , 0xBB , 0x7B , 0x7A , 0xBA , 0xBE , 0x7E ,
0x7F , 0xBF , 0x7D , 0xBD , 0xBC , 0x7C , 0xB4 , 0x74 , 0x75 , 0xB5 ,
0x77 , 0xB7 , 0xB6 , 0x76 , 0x72 , 0xB2 , 0xB3 , 0x73 , 0xB1 , 0x71 ,
0x70 , 0xB0 , 0x50 , 0x90 , 0x91 , 0x51 , 0x93 , 0x53 , 0x52 , 0x92 ,
0x96 , 0x56 , 0x57 , 0x97 , 0x55 , 0x95 , 0x94 , 0x54 , 0x9C , 0x5C ,
0x5D , 0x9D , 0x5F , 0x9F , 0x9E , 0x5E , 0x5A , 0x9A , 0x9B , 0x5B ,
0x99 , 0x59 , 0x58 , 0x98 , 0x88 , 0x48 , 0x49 , 0x89 , 0x4B , 0x8B ,
0x8A , 0x4A , 0x4E , 0x8E , 0x8F , 0x4F , 0x8D , 0x4D , 0x4C , 0x8C ,
0x44 , 0x84 , 0x85 , 0x45 , 0x87 , 0x47 , 0a46 , 0x86 , 0x82 , 0x42 ,
0x43 , 0x83 , 0x41 , 0x81 , 0x80 , 0x40
} ;
```

Abstract Reference

Modbus TCP

Function 01 Read Coil

Send:010000000006010100000004
Recv:01000000000401010100

Function 02 Read Input Status

Send:010000000006010200000008
Recv:01000000000401020103

Function 05 Write Single Coil

Set DO1 ON

Send:010000000006010500000FF00
Recv:010000000006010500000FF00

Set DO1 OFF

Send:010000000006010500000000
Recv:010000000006010500000000

Set DO2 ON

Send:010000000006010500001FF00
Recv:010000000006010500001FF00

Set DO2 OFF

Send:010000000006010500010000
Recv:010000000006010500010000

Set DO3 ON

Send:010000000006010500002FF00

Recv:010000000006010500002FF00

Set DO3 OFF

Send:010000000006010500020000
Recv:010000000006010500020000

Set DO4 ON

Send:010000000006010500003FF00
Recv:010000000006010500003FF00

Set DO4 OFF

Send:010000000006010500030000
Recv:010000000006010500030000