PROTOCOLO MODBUS E CICLOS DE MENSAGENS (MODBUS QUERY-RESPONSE)

Leandro Teodoro
Mai/2022

1. INTRODUÇÃO

Em 1979 a fabricante de CLPs Modicon publicou uma nova interface de rede baseada em uma arquitetura Master-Client chamada Modbus. Originalmente o protocolo foi desenvolvido para interfaces RS-232, entretanto foi migrado para o RS-485 algum tempo depois devido às vantagens como redes em longas distâncias e a implementação da conexão Master – Multi Slave.

O protocolo Modbus é de fácil implantação por sua baixa complexidade das mensagens. É um protocolo de nível lógico, dessa forma, não descrevendo rigorosamente o hardware da rede. Assim, as redes Modbus ficaram populares tanto utilizando a tecnologia RS-485 tanto o Ethernet. Com isso, foi uma ótima ferramenta para compatibilidade entre os diversos equipamentos industriais que necessitavam de uma comunicação digital com seu módulo de controle.

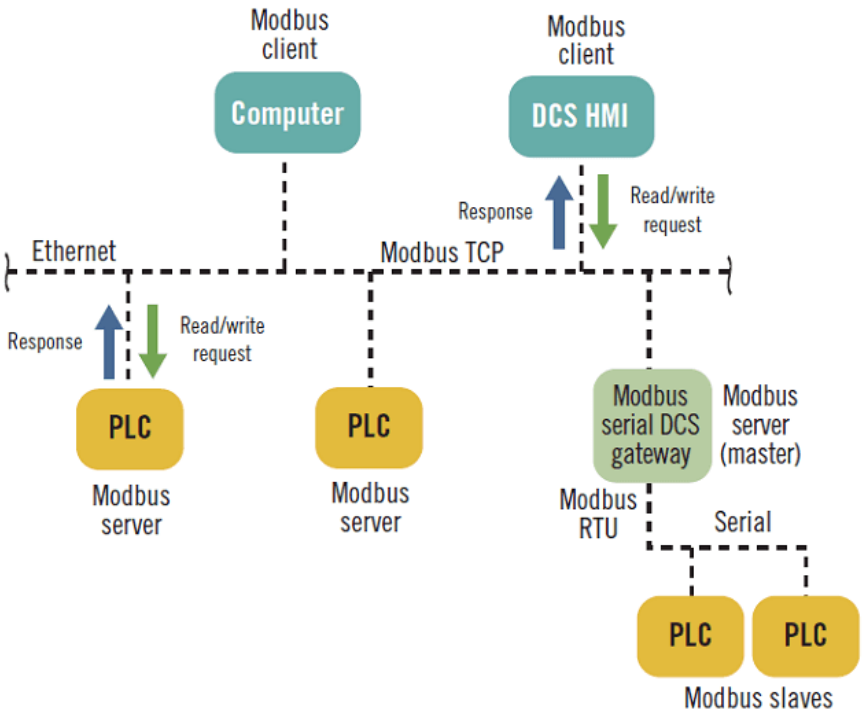


Figura 1 - Exemplo rede Modbus

Inicialmente foram desenvolvidas duas versões lógicas desse protocolo, conhecidas como Modbus/ASCII e Modbus/RTU. O primeiro é baseado no padrão American Standard Code for Information Interchange, sendo menos popular. Dessa forma, discorreremos sobre a versão RTU (Remote Terminal Unit). Um frame RTU enviado pela rede possui o seguinte formato padrão de mensagem: Início da mensagem (4 caracteres de delay) + endereço do dispositivo de destino (8 bits) + código da função (8 bits) + palavras de dados (até 252 bytes) + checagem de erro (16 bits) + Fim da mensagem (4 caracteres de delay).

Start	Slave ID	Funciton Code	Data	CRC Error	Stop
4 char	8 bits	8 bits	Até 252 bytes	2 bytes	4 char

Figura 2 - Modbus RTU Data Frame

2. CILOS DE MENSAGENS

Dentro do protocolo cada função possui um código específico sendo responsável por realizar uma operação distinta. Veremos abaixo algumas das funções mais utilizadas com seus respectivos códigos e um exemplo de um ciclo de requisição do dispositivo master e resposta do slave.

As funções mais utilizadas estão listadas abaixo com seus respectivos códigos. A memória típica do dispositivo Modbus está mapeada em 4 áreas, sendo que o endereço inicial está indicado na respectiva coluna. É muito importante consultar o manual do dispositivo para a correta identificação dos endereços.

Tabela 1 - Funções Modbus

Modbus Function	Code	Device Start Adress (Memory Map)
Read Coil Status	01	00001
Read Input Status	02	10001
Read Holding Registers	03	40001
Read Input Registers	04	30001
Force Single Coil	05	00001
Preset Single Register	06	40001
Exception	128 + Code	----

Agora veremos como é realizada a troca de mensagens em cada uma dessas funções. A estrutura da mensagem Modbus será representada com exemplos da seguinte forma:

[Device Adress] [Function Code] [Data Bytes] [X] [X]

Onde [X] é o byte de checksum gerado pelo algoritmo de erro. Por questões de objetivo esse algoritmo não será discutido nesse artigo.

- **Read Coil Status [Code: 01]:** Esta função é responsável por ler o estado das saídas digitais de um dispositivo slave. A mensagem do master envia quantos bits serão lidos e o endereço inicial de leitura. O slave responde o número de bytes enviados, neste exemplo 02 bytes, e o valor dos bits.

  Master Query: `[20] [01] [00] [29] [00] [10] [X] [X]`

  Slave Response: `[20] [01] [02] [15] [03] [X] [X]`

  Observações: Observe que o master quer ler 10 endereços a partir do endereço 30, ou seja, do 30 ao 39. Entretanto, por padrão, esse endereço na mensagem é subtraído de 1 unidade, o que corresponde a fração [00] [29]. Embora no protocolo Modbus cada saída tenha um endereço específico (vide manual no dispositivo), esta somente ocupa

o espaço de um bit na mensagem. Então, o slave agrupa enviando dois bytes [02] com a informação [15] [03], sendo que o valor menos significativo do primeiro byte é relativo ao endereço 30. Observe:

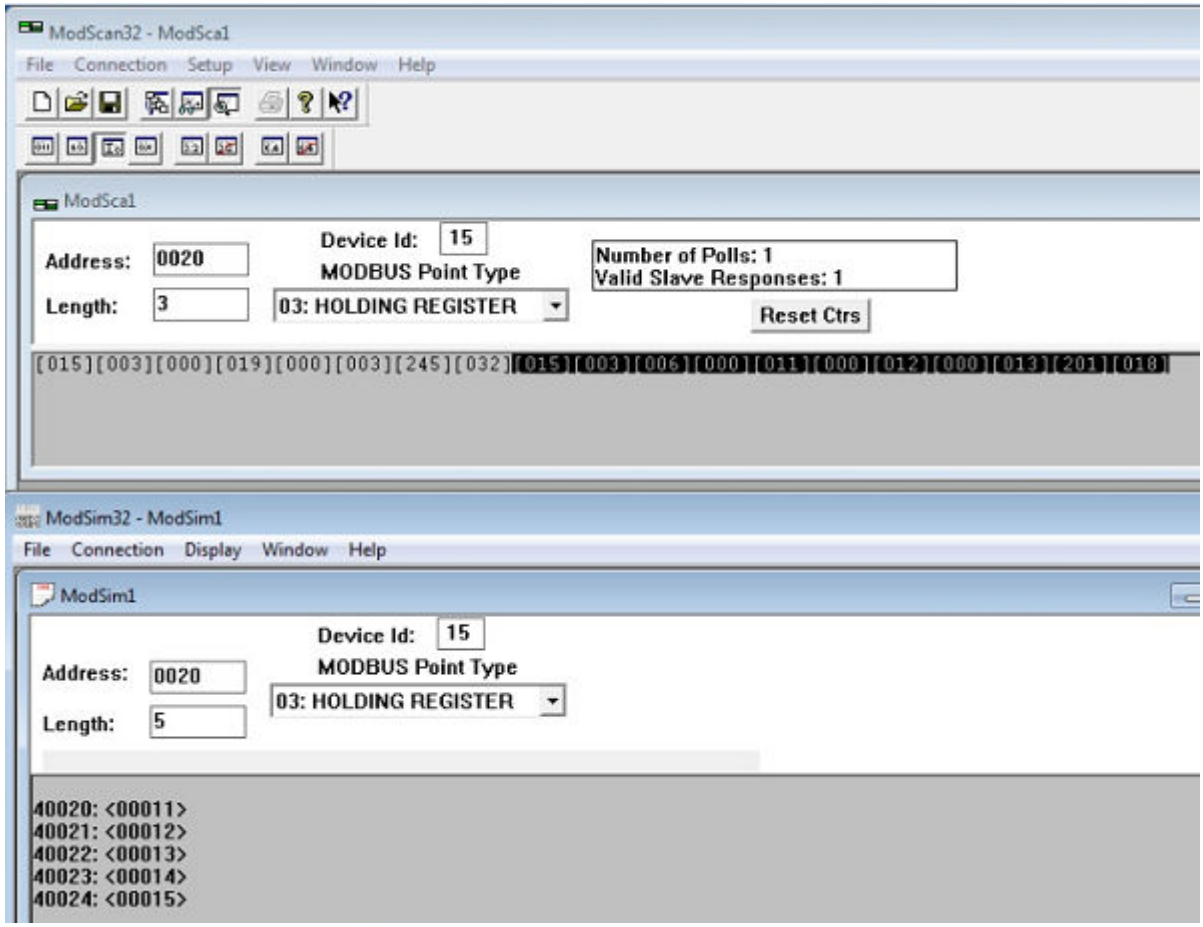
Tabela 2 - Coil Status Slave Response

Data Byte	[15]								[03]							
Binary Relative	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1
Coil Adress	37	36	35	34	33	32	31	30	--	--	--	--	--	--	39	38

- **Read Input Status [Code: 02]:** A função lê um estado de uma ou mais entradas digitais de um dispositivo slave. Segue a mesma ideia da função anterior. A mensagem do master possui o endereço de início de leitura e a quantidade de entradas que serão lidas, ambos no formato de 16bits. O dispositivo slave responde com o número de bytes enviados e os estados digitais das entradas.
  Master Query: `[21] [02] [00] [05] [00] [02] [X] [X]`
  Slave Response: `[21] [02] [01] [03] [X] [X]`
  Observações: Onde [00][05] é a fração do endereço das entradas digitais e [00][02] a quantidade de endereços lidos. Na resposta [03] pode ser convertido para binário onde o bit menos significativo representa o endereço inicial lido.

- **Read Holding Registers [Code: 03]:** Executa a leitura de um ou mais registradores de saída. A mensagem do master possui o endereço inicial e quantos registradores serão lidos. O slave responde a quantidade de registradores enviados e seus respectivos valores. Observe o exemplo:
  Master Query: `[15] [03] [00] [19] [00] [03] [X] [X]`
  Slave Response: `[15] [03] [06] [00] [11] [00] [12] [00] [13] [X] [X]`
  Observações: Na resposta os registradores lidos são de 16 bits. Entretanto, na mensagem são enviados seis bytes para compor a solicitação de leitura de três registradores de 16 bits. Por esse motivo o [06] é encontrado. A fração [00][11] corresponde ao valor do registrador gravado no primeiro endereço de leitura, já o valor [00][13] corresponde ao último.

Figura 3 - Read Holding Registers Example



Observe no software de simulação que o endereço inicial do registrador lido é somado com o endereço inicial do mapeamento de memória que se encontra na tabela 1. Confira no manual do equipamento o endereço atribuído para cada função Modbus.

- **Read Input Registers [Code: 04]:** Realiza a leitura de um ou mais registradores de entrada. Segue a mesma estrutura de mensagens da função anterior.
  Master Query: `[22] [04] [00] [18] [00] [01] [X] [X]`
  Slave Response: `[22] [04] [02] [00] [35] [X] [X]`
  Observações: Da mesma forma que a função anterior o master envia o endereço inicial e a quantidade de registradores que serão lidos no formato de 16 bits.

- **Force Single Coil [Code: 05]:** Agora veremos algumas funções de escrita. A função force single coil é responsável por alterar uma saída digital de um dispositivo, o qual neste exemplo está no endereço da rede [09].
  Master Query: `[09] [05] [00] [05] [FF] [00] [X] [X]`
  Slave Response: `[09] [05] [00] [05] [FF] [00] [X] [X]`
  Observações: A fração [FF][00] são bytes padrões para ligar a saída e [00] [00] utilizado para desligar. Tudo ocorrendo bem o slave responde com a mesma mensagem do master.

- **Preset Single Register [Code: 06]:** Utilizado para escrever em um registrador interno do dispositivo. O master envia duas palavras de 16 bits, uma contendo o endereço do registrador no dispositivo [00][07] e outra contendo o dado que será escrito [00][05].
  Master Query: `[09] [06] [00] [07] [00] [05] [X] [X]`
  Slave Response: `[09] [06] [00] [07] [00] [05] [X] [X]`
  Observações: A mensagem sendo aceita sem erros o slave responde um echo da mensagem do master.

- **Exception Message [128 + Code]:** Durante a comunicação Modbus podem ocorrer problemas e a solicitação do master pode não ser processada. Nesse caso o slave retorna uma mensagem de erro. No exemplo abaixo o master tenta ler um registrador que não existe no dispositivo slave.
  Master Query: `[22] [04] [00] [23] [00] [01] [X] [X]`
  Slave Response: `[22] [132] [02] [X] [X]`
  Observações: A mensagem de erro do slave e composta por [132] que é a soma de 128 mais o código da função que originou a mensagem [04]. O dispositivo também retorna o código [02] que significa o erro *"illegal data adress"*. Existem outros códigos de erros que podem ser retornados, como: 01 – Illegal function, 03 – Illegal data value, entre outros. Para uma descrição completa vide as referências.

3. CONSIDERAÇÕES FINAIS

Para a implementação do protocolo Modbus em microcontroladores ou outros dispositivos microprocessados é primordial que se entenda como funciona o ciclo de mensagens do protocolo. Existem softwares que podem ser acoplados a rede com a finalidade de escutá-la a fim de solucionar problemas, sendo essencial também o conhecimento de como se procede à comunicação da rede para que a ferramenta possa ser utilizada na sua totalidade.

4. REFERÊNCIAS

[1]. MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3

[2]. Simply Modbus. Disponível em: <https://www.simplyModbus.ca/index.html>. Acesso em: Mai/2022