PROTOCOLO MODBUS

João Maria Araújo do Nascimento
joao@dca.ufrn.br
Pedro Berretta de Lucena
pedrobdl@yahoo.com.br

LECA-DCA-UFRN
Universidade Federal do Rio Grande do Norte

Resumo: Este artigo descreve as principais características do Protocolo de Comunicação Modbus, apresentando algumas de suas peculiaridades e demonstrando alguns exemplos de implementação.

Palavras Chaves: Redes Industriais, Modbus, mestre e escravo.

Abstract: This paper describes the main characteristics of the Modbus Comunication Protocol, showing some peculiarities and demonstrating some examples of implementation.

Keywords: Industrial Nets, Modbus, master and slave.

1 INTRODUÇÃO

O protocolo Modbus foi desenvolvido pela Modicon Industrial Automation Systems, hoje denominada de Schneider, com o objetivo de comunicar um dispositivo mestre com outros dispositivos escravos, independentemente do tipo de rede utilizada.

Este protocolo define uma estrutura de mensagens compostas por *bytes*, que os mais diversos tipos de dispositivos são capazes de reconhecer (ALFA INSTRUMENTOS, 2000). Embora seja utilizado normalmente sobre conexões seriais padrão RS-232, ele também pode ser usado como um protocolo da camada de aplicação de redes industriais tais como TCP/IP sobre *Ethernet* e MAP (SEIXAS, 200?).

No campo das Redes Industriais, este é talvez o protocolo de mais larga utilização, já que diversos controladores e ferramentas para desenvolvimento de sistemas supervisórios utilizam este protocolo, isto se deve a sua grande simplicidade e facilidade de implementação.

2 MODELO DE COMUNICAÇÃO

O Protocolo Modbus é baseado no modelo de comunicação mestre-escravo, onde apenas o único dispositivo mestre pode inicializar a comunicação, também conhecida como *query*, e os demais dispositivos escravos, respondem enviando os dados solicitados pelo mestre, ou realizam alguma ação solicitada.

O dispositivo mestre pode endereçar cada dispositivo escravo da rede individualmente ou acessar a todos da rede através de mensagens em *broadcast.* Veja os exemplos de comunicação mestre-escravo na Figura 1.

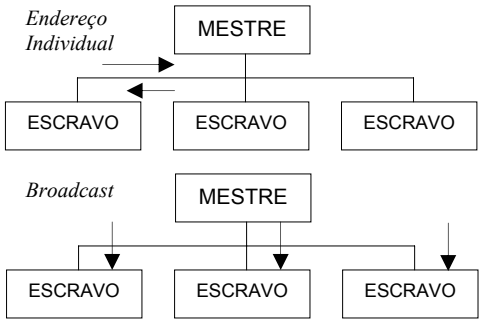

Figura 1 – Modelo mestre-escravo.

Quando o mestre envia uma mensagem endereçada a um escravo, apenas o dispositivo endereçado retorna uma resposta (*response*) a uma *query* e nunca são gerados *responses* quando uma *query* for do tipo *broadcast*.

O formato das mensagens (*query*) definidas pelo protocolo Modbus é estabelecido da seguinte forma:

- Endereço do escravo ou endereço para *broadcast*;
- Código da função que indica qual ação deve ser realizada;
- Parâmetros necessários para as funções;
- Campo *Checksum* para verificar a integridade dos dados.

Já o formato das respostas (*response*) seguem o mesmo modelo de uma *query* porém, são ajustadas obedecendo o formato da função requerida:

- Confirmação da função;
- Parâmetros pertinentes as funções;
- Campo *Checksum*.

Na existência de algum erro de comunicação, ou se o escravo não estiver apto para atender a função requisitada, o dispositivo escravo monta uma mensagem denominada *exception*, a qual justifica o não atendimento da função. Veja uma representação do modelo de comunicação na Figura 2.

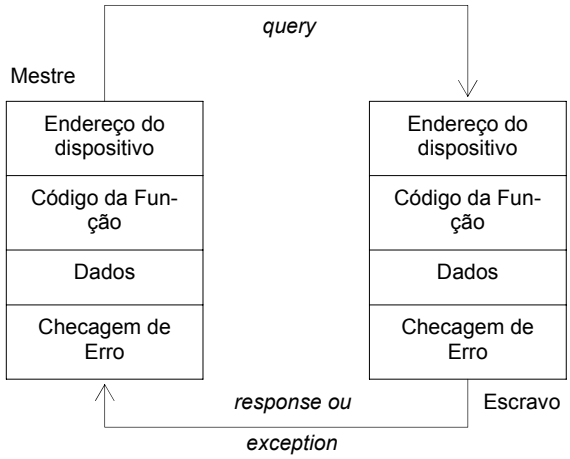

Figura 2 – Modelo das mensagens Modbus.

3 MODOS DE TRANSMISSÃO

O protocolo Modbus pode ser configurado para trabalhar com um dos dois modos de transmissão disponíveis: **ASCII** (*American Code for Informastion Interchange*) ou **RTU** (*Remote Terminal Unit*), os quais definem como os dados serão empacotados na mensagem, estes modos são escolhidos durante a configuração dos parâmetros de comunicação, tais como: *baud rate*, paridade, *stop bits*.

Em uma rede industrial utilizando o protocolo Modbus, todos os dispositivos da rede devem ser configurados com o mesmo modo de transmissão.

3.1 Modo ASCII

Neste modo cada palavra de dado da mensagem são enviados dois caracteres no padrão ASCII. A principal vantagem deste modo de transmissão é a possibilidade de haver grandes intervalos entre o envio de dados de uma mesma mensagem.

O *framing* de dados que é composto por várias palavras de dados apresentará somente valores de **30H** à **39H** e **41H** à **46H**, que correspondem respectivamente aos números de **0** à **9** e **A** à **F** no padrão hexadecimal e **0** à **9** e **10** à **15** no padrão decimal (ALFA INSTRUMENTOS, 2000).

No modo ASCII a quantidade de *bits* por cada palavra de dados do *framing* sempre será igual a **10**, independente da configuração escolhida. Estas são as possíveis configurações:

- 1 *Start bit*, 7 *data bits*, sem paridade e 2 *stop bits*;
- 1 *Start bit*, 7 *data bits*, paridade PAR e 1 *stop bits*;
- 1 *Start bit*, 7 *data bits*, paridade IMPAR e 1 *stop bits*.

No campo de *checksum* é utilizado o método LRC (*Longitudinal Redundancy Check*), o qual será abordado mais adiante.

3.2 Modo RTU

Um dispositivo configurado para este modo, para cada palavra de dados da mensagem é enviado apenas um caracter no padrão HEXADECIMAL. A principal vantagem deste modo RTU em relação ao ASCII é a maior densidade de caracteres que é enviada numa mesma mensagem, aumentando o desempenho da comunicação.

Neste modo de transmissão a palavra de dados sempre será igual a **11**, independente da configuração dos parâmetros.

- 1 *Start bit*, 8 *data bits*, sem paridade e 2 *stop bits*;
- 1 *Start bit*, 8 *data bits*, paridade PAR e 1 *stop bits*;
- 1 *Start bit*, 8 *data bits*, paridade IMPAR e 1 *stop bits*;

O campo *checksum* do *framing* é gerado pelo método **CRC** (*Cyclical Redundancy Check*).

3.3 *Framing* da mensagem

Nos dois modos de transmissão do protocolo Modbus, existem caracteres de indentificação de **início** e **fim** de *framing*, específicos para cada modo. Com esta técnica é possível que os dispoditivos escravos detectem o início de uma mensagem, indentifiquem o endereço do escravo (qual escravo vai responder) ou uma mensagem *broadcast* (todos os escravos recebem, mas não respondem), e finalmente ler todo o conteúdo da mensagem até o seu final.

As mensagens podem ser lidas parcialmente se ocorrcer algum erro, ou existir um período maior que o *time-out* entre o envio de uma palavra de dados e outra. Estes eventos geram as *exceptions*.

3.3.1.1 *Framing* no MODO ASCII

Durante a transmissão das palavras de dados neste modo, intervalos de até **um segundo** entre caracteres são permitidos, sem que a mensagem seja truncada (SEIXAS, 200?). Neste modo o início das mensagens são identificados pelo caracter **:** (dois pontos), o qual corresponde ao valor ASCII 3AH. Já o término das mensagens é composto pelo conjunto de caracteres **Retorno de carro** (*Carriage Return* - CR) e **Avanço de linha** (*line feed* – LF), respectivamente com os correspondetes em ASCII aos valores 0DH e 0AH (ALFA INSTRUMENTOS, 2000).

Os dispositivos escravos verificam constantemente o barramento e quando detectam o caracter **:** (3AH), se preucupam em codificar o campo seguinte que corresponde ao endereço de um dispositivo ou *broadcast*. A seguir é exibido na Figura 3 um *framing* típico no MODO ASCII.

Início	**Endereço**	**Função**	**Dados**	**LRC**	**Fim**
: 3AH	2 *chars*	2 *chars*	N *chars*	2 *chars*	**CRLF**

Figura 3 – *framing* modo ASCII

3.3.1.2 *Framing* modo RTU

O modo RTU não possui bytes que indentificam o início e fim da mensagem. Para indetificar estes campos, não pode existir nenhuma palavra de dados por um mínimo de **3.5 vezes o tamanho da palavra de dados**, este técnica é conhecida como *silent*.

Para uma taxa de 4800 bps, o tempo total de envio de uma palavra é **2291,6 us** (11 x (1/4800)), ou seja, para identificar o

início e fim de um *framing*, não deve ocorrer nenhuma transmissão por um período de **6,785ms** (3,5 x 2291,6 us).

Neste modo, os dispositivos ficam observando os intervalos de tempo *silent* que, após detectado, dá se início ao recebimento da mensagem. No final da mensagem o mestre deve gerar outro intervalo de tempo, equivalente ao início, para caracterizar o final da mensagem.

Toda mensagem no modo RTU deve ser transmitida continuiamente, pois se um intervalo de **1,5 x tamanho da palavra** for detectado pelo escravo, ele descarta todos os dados que já recebeu e assume que o novo caracter que recebeu é o campo de endereço de uma nova mensagem, da mesma forma se uma mensagem for recebida em um tempo menor que o *silent* ocorrerá um erro. Abaixo é mostrado na Figura 4, um framing no modo RTU.

Início	Endereço	Função	Dados	CRC	Fim
silent	1 *char*	1 *char*	N *chars*	2 *chars*	***silent***

Figura 4 – *framing* modo RTU

3.3.1.3 Campo de endereço

No campo de endereço de uma mensagem Modbus, podem existir **dois caracteres** no modo ASCII, ou **8 *bits***, no modo RTU. Os endereços válidos correspondem a faixa de endereço de **0 à 247**, porém o endereço **0** é utilizado para *broadcast*, que é o único endereço, que todos os escravos reconhecem além do prório (ALFA INSTRUMENTOS, 2000).

Quando um escravo retorna um *response* para uma máquina mestre, ele coloca seu próprio endereço na mensagem para indentificá-lo.

3.3.1.4 Campo de funções

No campo de funções também são utilizados **dois caracteres** no modo ASCII, ou **8 *bits***, no modo RTU. As funções válidas estão presentes na faixa de **1 à 255**, porém nem todas estão implementadas e muitas são comuns para diversos tipos de controladores.

O dispositivo mestre deve ter a responsabilidade de verificar o campo de função das respostas dos escravos, pois este campo informa se houve problemas com a função solicitada, ou seja, se não houve nenhum problema, o escravo retorna no campo de funções o mesmo valor da função solicitada pelo mestre, mas se houve problemas o escravo devolve o mesmo valor da função, porém com o seu *bit* mais significativo em 1 (nível alto).

3.3.1.5 Campo de dados

Este campo é formado por dois caracteres ASCII no modo ASCII, ou 1 *byte* no modo RTU. Ele pode variar de **00H à FFH**.

Neste campo existem informações relacionadas com o código da função no campo de funções, como por exemplo, o número de variáveis discretas a serem lidas ou ativadas.

3.3.1.6 Comandos com erro

Podem ocorrer os seguintes erros na transmissão de uma mensagem:

- Se o escravo não recebeu a mensagem do mestre por problemas de comunicação, o escravo não retorna nenhuma mensagem, e o mestre irá verificar o *time-out*;
- Se o escravo recebeu a mensagem, porém detectou problemas de comunicação, ele também não irá retorrnar uma mensagem ao mestre, e o mestre verifica o *time-out*;
- Se o escravo recebeu uma mensagem sem erros de comunicação, mas ele não foi capaz de atender a solicitação, ele irá retornar ao mestre uma *exception* informando ao mestre a natureza do erro.

Abaixo na Tabela 1, alguns códigos de *exceptions* e seus significados:

Código	Descrição
1	Função inválida
2	Registrador inválido
3	Valor de dado inválido
4	Falha no dispositivo
5	Estado de espera
6	Dispositivo ocupado
7	Não reconhecimento
8	Erro de paridade

Tabela 1 – *exceptions*

3.3.1.7 Campo *checksum*

Este campo é verificado ora pelo dispositivo mestre ora pelo dispositivo escravo. No modo ASCII utiliza o método LRC, e no modo RTU o CRC.

3.3.1.7.1 Checagem do *framing* – LRC (*Longitudinal Redundancy Check*)

O cálculo deste campo é realizado utilizando todos os campos da mensagem exceto os caracteres de início **: 3A** e fim de mensagem **CRLF**. O valor gerado por este método é de 8 *bits*, e portanto possui dois caracteres ASCII.

O LRC é calculado pelo escravo logo no recebimento da mensagem e em seguida comparadado com o valor do LRC recebido. O calculado é feito a partir da adição sucessiva dos 8 *bits* dos campos da mensagem, descartando possíveis *bits* de estouro, e submetendo o resultado final a lógica de complemento de dois. Assim como o resultado tem que necessariamente ser um valor de 8 *bits* e o resultado das adições sucessivas provavelmente excederá 255 o valor máximo permitido, simplesmente é descartado o nono *bit*.

Um exemplo do cálculo de LRC pode ser visto no exemplo abaixo:

3AH | 30H 43H | 30H 33H | 30H 30H 30H 41H | 30H 30H 30H 31H | 45H 35H | 0DH 0AH

Este *framing* corresponde em decimal a:

: | 12 | 3 | 10 | 1 | 230 | CLRF

Em hexadecimal a:

: | 0CH | 03H | 000AH | 0001H | **E6H** | CLRF

O somatório destes valores é equivalente a: 12 + 3 + 10 + 1 = 26, o que equivale em hexadecimal a 1AH.

Realizando o complemento de 1 temos: FFH – 1AH = E5H.

Em sequida com o complemento de 2: E5H + 01H = E6H.

Como o modo de transmissão é o ASCII, o valor E equivale a 45H e o valor 6 a 36H.

3.3.1.7.2 Checagem do *framing* – CRC (*Cyclical Redundancy Check*)

Com este método é gerado um valor de 16 *bits*, onde os 8 *bits* menos significativos são enviados primeiros e os 8 *bits* mais significativos após. Este método exige uma maior complexidade para o seu desenvolvimento, porém é mais confiável.

Foge da idéia deste artigo a explicação do desenvolvimento deste método.

4 PRINCIPAIS FUNÇÕES DO PROTOCOLO MODBUS

As funções em Modbus variam de 1 a 255 (01H a FFH), mas apenas a faixa de um a 127 (01H a 7FH) é utilizada, já que o bit mais significativo é reservado para indicar respostas de exceção (SEIXAS, 200?). A maioria das funções funcionam em diversos dispositivos e outras ainda devem ser implementadas.

4.1 Leitura de entradas e saídas discretas

- ***Read Coil Status***: leitura de saídas discretas – Código 01. Formato: endereço incial 2 *bytes* e número de saídas 2 *bytes*;
- ***Read Input Status***: leitura de entradas discretas – código 02. Formato: endereço incial 2 *bytes* e número de saídas 2 *bytes.*

4.2 Leitura de Registradores

- ***Read Holding Registers***: leitura de registradores do dispositivo escravo – código 03. Formato: endereço incial 2 *bytes* e número de registradores 2 *bytes*;
- ***Read Input Registers***: leitura dos valores das entradas dos registros – cógigo 04. Formato: endereço incial 2 *bytes* e número de registradores 2 *bytes;*
- ***Preset Single Register***: escrita de um valor em registrador – código 06. Formato: endereço 2 *bytes* e valor 2 *bytes.*

5 EXEMPLOS DE IMPLEMENTAÇÃO

Hoje pode se encontrar aplicações para utilização do protocolo Modbus em diversas áreas, desde o controle de motores, inversores inteligentes a controle de poços de petróleo, ou seja, a implementação deste protocolo é confiável e principalmente fácil. Abaixo na Figura 5, um exemplo de um programa simples que gera *framing* para o protocolo Modbus modelo ASCII e que pode ser usado em computadores, para comunicação com qualquer dispositivo que utilize Modbus.

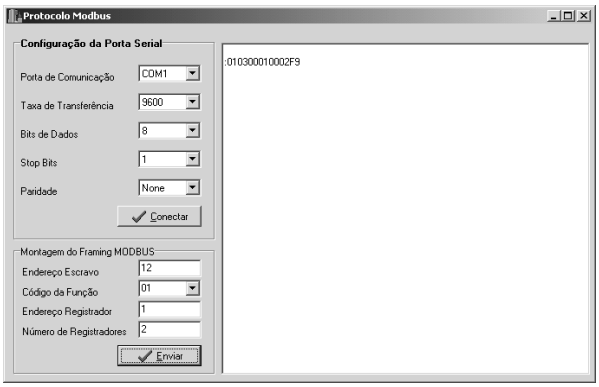

Figura 5 – Aplicativo Modbus ASCII

6 CONCLUSÃO

Atualmente a utilização de redes industriais é um fator indispensável quando se quer obter informações e gerenciar processos, evitando perdas de produtividade e falhas no processo. Com a utilização do protocolo Modbus, o qual é conhecido como um protocolo orientado a caracter, é possível de forma clara e objetiva desenvolver soluções em redes industriais capazes de atender uma grande faixa de necessidades, a partir de uma estrutura de mensagens de fácil aprendizagem e simples implementação.

REFERÊNCIA BIBLIOGRÁFICA

ALFA INSTRUMENTOS. **Protocolo de Comunicação Modbus RTU/ASCII**, Alfa Instrumentos, 2000.

SEIXAS, Constantino. **Protocolos Orientados a Caracter**. UFMG – Departamento de Engenharia Eletrônica, Minas Gerais, 200?.

MICHEL, J. C.; STRINGARI, S. **Protótipo de Rede Industrial Utilizando o Padrão Serial RS485 e Protocolo Modbus**, I Congresso Brasileiro de Computação - CBComp 2001, Blumenau – SC, 2001.

NATALE, Ferdinando. **Automação industrial**. São Paulo: Érica, 2000.