**MODBUS via Serial**

MODBUS fornece uma comunicação cliente/servidor entre dispositivos conectados em diferentes tipos de barramentos ou redes. Somente o cliente pode iniciar uma a transação (querie) para um dispositivo servidor por vez (unicast). Entretanto, o cliente pode enviar uma mensagem para todos os dispositivos servidores (broadcast).

O cliente é o dispositivo mestre e os servidores são os dispositivos escravos.

MODBUS é um protocolo de pedido e resposta e oferece serviços especificados como códigos de funções.

O protocolo MODBUS a nível de transferência de dados define um simples frame de bits chamado de unidade de dados (PDU – Protocol Data Unit) que é independente da camada de comunicação. Dependendo do tipo de barramento de comunicação alguns bits de controle são adicionados a esse frame criando a unidade de dados de aplicação (ADU – Application Data Unit).



O campo de código de função ocupa 1 byte, e os valores válidos para as funções são de 1 a 127, já os valores 128 a 255 são reservados para respostas de exceções. Quando a mensagem é enviada do cliente para o servidor o campo da função de código diz ao servidor que tipo de ação a tomar. O código de função 0 não é válido.

O campo data da mensagem enviados do cliente ao servidor contém informações adicionais, dependendo do tipo de função, para que o servidor possas manipular os dados. Informações tais como: endereços dos registradores, quantidade de dados a ser enviado entre outras informações. O campo data pode ser inexistente, tamanho zero, em certos tipos de pedido, neste caso o servidor não requer nenhuma informação adicional, ou seja, o código da função por si só já especifica a ação.

Se nenhum erro ocorrer no pedido do cliente a resposta do servidor será preenchido com o campo da função com eco da função requisitada e com o campo de dados com os dados requisitados.



Se algum erro ocorrer no processamento da função, o servidor envia o código da exceção no campo da função e código de erro no campo de dados determinar a próxima ação a ser feita pelo cliente.



É recomendado utilizar um timeout no cliente para que o mesmo não fique indefinidamente esperando por uma resposta que nunca pode chegar. Geralmente usamos um tempo de 2 a 3 segundos.

Há dois modos de transmissão, o modo ASCII e modo RTU (Remote Terminal Unit). Nesse documento usaremos somente o modo RTU que o modo de transmissão binário.

**Tamanho ADU**

O tamanho do ADU para as linhas seriais rs232/rs485 é a 256 bytes:

* 1 byte para endereço;
* 253 bytes para PDU onde 1 byte para código da função;
* 2 bytes para CRC.

**Tipos de PDUs**

No protocolo MODBUS é definido três tipos de PDUs:

* PDU de pedido;
* PDU de resposta;
* PDU de reposta de exceção.

MODBUS utiliza a representação de “big-endian” para os itens de endereços e dados, ou seja, quando um dado maior que 1 byte é enviado o byte mais significativo é enviado primeiro.

**Campo de Dados**

O modelo de dados se baseiam em quatro tipos básicos:

* **Discretes Input**: Lê um simples bit;
* **Coils**: Lê e escreve em um simples bit;
* **Input Registers**: Lê um registrador de 16 bits;
* **Holding Registers**: Lê e escreve um registrador de 16 bits;

A distinção entre entrada e saída, e endereçamento entre bit e word itens de dados, não implica em qualquer comportamento do aplicativo. As funções são construindo em base desses quatros acessos básicos.

**Definição de uma transação**

O seguinte diagrama de estado descrever um processamento genérico de uma transação do lado do servidor:



Uma vez o pedido tenha sido processado por um servidor, uma resposta ao cliente é construída. Dependendo do resultado a dois tipos de resposta são construídas:

* **Resposta positiva**: O campo do código da função é preenchido com o código de função da solicitação;
* **Resposta com exceção**: O objetivo é fornecer ao cliente informações de erros detectados durante o processamento do pedido. O campo do código da função é preenchido com o código de função da solicitação + 0x80. O campo data com o código da exceção indicando a razão do erro.

**Código de Funções**

Há três categorias de funções:

* Funções públicas que utilizam endereços de 1 a 64, 72 a 100 e 100 a 127;
* Funções definidas pelos usuários que utilizam endereços de 65 a 71 e 100 a 109;
* Funções reservadas que utilizam endereços acima de 127. São as funções de exceções.

Funções publicas



Para nossos projetos vamos descrever somente algumas funções que utilizamos, para mais informações consulte “*modbus\_application\_protocol\_v1\_1b.pdf*” na secção 6.

**Código de Função 3 – Read Holding Registers**

Esta função é usada para ler um bloco de registradores no dispositivo remoto. A solicitação PDU especifica o endereço do primeiro registrador a ser lido e o número de registradores a serem lidos. Os registros são enumerados de 1 a N, porém são endereçados de 0 a N-1.



Os dados são empacotados na mensagem em dois bytes por registrador.

No servidor podemos processar esse comando de muitas formas, mas recomendo fazer como acesso em uma memória qualquer por ser mais intuitivo como mostrado no desenho a seguir:



Apesar que o protocolo permita, podemos no servidor fragmentar os registradores como mostrado na imagem abaixo, mas não é um método recomendados por ser mais confuso.



**Código de Função 6 – Write Single Register**

Esta função é usada para escrever em um simples registrador no dispositivo remoto. A solicitação PDU especifica o endereço do registrador a ser gravado. Os registradores são endereçados iniciando a partir do endereço 0. A resposta é o eco do pedido logo após o registrador ser atualizado. Os registros são enumerados de 1 a N, porém são endereçados de 0 a N-1.



**Código de Função 16 – Write Multiple Register**

Esta função é usada para escrever em bloco continuo de registradores de 1 a 123 no dispositivo remoto. A solicitação PDU especifica o endereço inicial do registrador a ser gravado. Os registradores são endereçados iniciando a partir do endereço 0. A resposta é o eco do pedido logo após o registrador ser atualizado. Os registros são enumerados de 1 a N, porém são endereçados de 0 a N-1.



**Resposta do servidor**

O servidor responde para o cliente com o campo address com o seu endereço e com o campo CRC com o valor da mensagem atual.

**Códigos de exceções**

Quando um dispositivo cliente envia uma requisição ao servidor ele espera uma reposta normal. Uma das quatro possibilidades e eventos pode ocorrer de uma requisição:

* Se o servidor recebe a requisição sem um erro de comunicação e pode manipular o pedido normalmente. Então o servidor retorna uma resposta normal;
* Se o servidor não recebe o pedido de requisição devido a um erro de comunicação e nenhuma resposta é enviada. O programa cliente deve processar uma condição de timeout para esse caso;
* Se o servidor recebe a requisição sem um erro de comunicação, mas detecta um erro de CRC, LRC ou paridade, nenhuma resposta é enviada ao cliente. O programa cliente deve processar uma condição de timeout para esse caso;
* Se o servidor recebe a requisição sem um erro de comunicação, porém o servidor não conseguir manipular, como por exemplo, acessar endereços inexistentes, o servidor deve retornar com a resposta de exceção para o cliente informando a natureza do erro.

A mensagem de resposta de exceção é composta de dois campos:

* Campo de código de função: Retorna com o código da função recebido do cliente + 0x80 sinalizando uma exceção;
* Campo dado: Aqui retorna com o código da exceção.

Os códigos de exceções que as funções 03 e 06 podem gerar estão na tabela abaixo a seguir:

|  |  |  |
| --- | --- | --- |
| **Código** | **Nome** | **Descrição** |
| 01 | ILLEGAL FUNCTION | O servidor recebeu uma função que não foi implementada ou não foi habilitada.  |
| 02 | ILLEGAL DATA ADDRESS | O servidor precisou acessar um endereço inexistente. |
| 03 | ILLEGAL DATA VALUE | O valor contido no campo de dado não é permitido pelo servidor. Isto indica uma falta de informações na estrutura do campo de dados. |
| 04 | SLAVE DEVICE FAILURE | Um irrecuperável erro ocorreu enquanto o servidor estava tentando executar a ação solicitada. |

**Quadro da Mensagem**

O quadro da mensagem delimita a início e fim de uma transmissão, isto permite que o receptor de início a leitura da mensagem a partir do endereço e quando esta for completa.

No modo RTU a mensagem inicia com um intervalo silencioso de 3.5 vezes o tamanho do caractere.



Um tempo de 3.5 vezes do tamanho de um caractere em uma transmissão de 115200 é de 303uS ((1/115200)\*10\*3.5=303us). Porém, nem todos os dispositivos conseguem detectar esse tempo de silencio seja por implementação simples das bibliotecas ou por serem lentos para isto. Então, o mestre deve utilizar um timeout pela espera de respostas dos escravos bem maiores que 300uS, geralmente são usados timeouts de 1 segundos ou mais.

Um mestre deve pelo menos fazer no mínimo duas tentativas de conexão com o escravo, isto porque na primeira tentativa o escravo pode estar dessincronizado na captura das mensagens. Esse dessincronizo pode ser o fato que o escravo foi ligado ou conectado ao barramento serial durante um processo de comunicação. Isto porque os escravos sempre ficam capturando os bytes do barramento até detecta um silencio no barramento, quando esse silencio é detectado os escravos processam a mensagem capturada e se a mensagem foi pega pela metade a mesma é descartada. Se isso ocorrer em algum momento o dispositivo vai se sincronizar com o barramento, seja na próxima comunicação ao mestre no barramento ou quando o mestre deseja se comunicar com ele. Na pior da hipótese, na primeira tentativa do mestre não vai haver resposta do escravo, pois este está dessincronizado. Então, o mestre vai entrar em timeout e nesse momento todos os dispositivos serão sincronizados para as próximas mensagens e o mestre reinicia a transação com o dispositivo novamente.

Uma mensagem deve ser transmitida em um fluxo continuo. Se um intervalo de silencio for maior que 1.5 o tempo de caractere antes da finalização da transmissão, o receptor deve desconsiderar a transação.



Os endereços válidos são de 0 a 247, onde o valor 0 é reservado para o broadcast. Sempre que um escravo vai enviar uma resposta ao cliente este deve incluir o seu endereço no campo Address.

Representação da word no buffer para dados e endereços:



**Software testador ModBUS**

<http://freemodbus.com/index.html>

vamos criar uma tarefa para comunicação:



Crie uma tarefa começando em adicionar uma conexão:



Entre com a porta serial e suas configurações, também entre com o timeout, depois clique em adicionar a conexão:



Agora vamos adicionar um dispositivo para essa conexão:



Entre com o edereço do escravo:



Agora vamos adicionar uma função para o dispositivo escravo:



Entre com o tipo de função que deseja acessar o escravo:



Clique OK para concluir a contrução da tarefa:



Selecione a tarefa e clique em POLL para iniciar a comunicação com o dispostivo:

