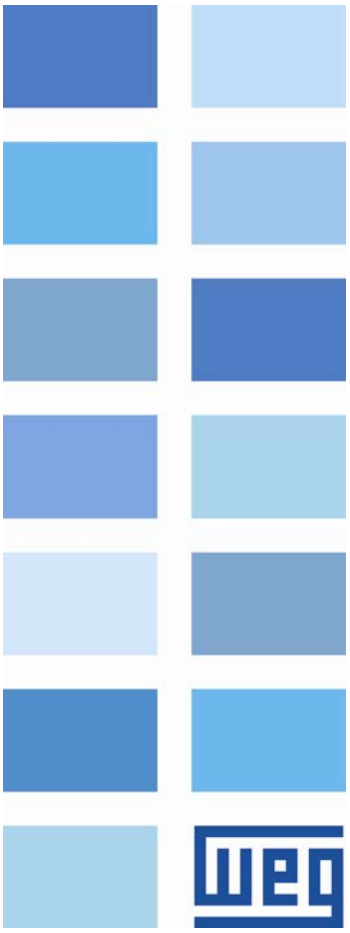
Modbus RTU

PLC300

Manual do Usuário



Manual do Usuário Modbus RTU

Série: PLC300

Idioma: Português

N ° do Documento: 10000850708 / 03

Data da Publicação: 04/2013

SUMÁRIO

SOBRE O MANUAL

Este manual fornece a descrição necessária para a operação do controlador programável PLC300 utilizando o protocolo Modbus RTU. Este manual deve ser utilizado em conjunto com manual do usuário do PLC300.

ABREVIAÇÕES E DEFINIÇÕES

ASCII	American Standard Code for Information Interchange
CRC	Cycling Redundancy Check
EIA	Electronic Industries Alliance
TIA	Telecommunications Industry Association
RTU	Remote Terminal Unit

REPRESENTAÇÃO NUMÉRICA

Números decimais são representados através de dígitos sem sufixo. Números hexadecimais são representados com a letra 'h' depois do número. Números binários são representados com a letra 'b' depois do número.

DOCUMENTOS

O protocolo Modbus RTU foi desenvolvido baseado nas seguintes especificações e documentos:

Documento	Versão	Fonte
MODBUS Application Protocol Specification, December 28th 2006.	V1.1b	MODBUS.ORG
MODBUS Protocol Reference Guide, June 1996.	Rev. J	MODICON
MODBUS over Serial Line, December 20th 2006.	V1.02	MODBUS.ORG

Para obter esta documentação, deve-se consultar a MODBUS.ORG, que atualmente é a organização que mantém, divulga e atualiza as informações relativas ao protocolo Modbus.

1 INTRODUÇÃO À COMUNICAÇÃO SERIAL

Em uma interface serial os bits de dados são enviados sequencialmente através de um canal de comunicação ou barramento. Diversas tecnologias utilizam comunicação serial para transferência de dados, incluindo as interfaces RS232 e RS485.

As normas que especificam os padrões RS232 e RS485, no entanto, não especificam o formato nem a sequência de caracteres para a transmissão e recepção de dados. Neste sentido, além da interface, é necessário identificar também o protocolo utilizado para comunicação. Dentre os diversos protocolos existentes, um protocolo muito utilizado na indústria é o protocolo Modbus RTU.

A seguir serão apresentadas características das interfaces seriais RS232 e RS485 disponíveis para o produto, bem como a descrição detalhada do protocolo Modbus RTU para utilização destas interfaces.

2 DESCRIÇÃO DAS INTERFACES

O controlador programável PLC300 possui uma interface RS232 e uma interface RS485 padrão no produto. A seguir são apresentadas informações sobre a conexão e instalação do equipamento em rede.

2.1 RS232

2.1.1 Características da interface RS232

- Interface segue o padrão EIA/TIA-232.
- Opera com o protocolo Modbus RTU no modo escravo apenas, configurado para o endereço de rede 1.
- Possibilita comunicação utilizando taxas de 1200 até 57600 Kbit/s.
- Permite a conexão entre o equipamento e o mestre da rede (ponto-a-ponto).
- Distância máxima para ligação entre os dispositivos: 10 metros.

2.1.2 Pinagem do Conector

A conexão para a interface RS232 está disponível através do conector XC3 utilizando a seguinte pinagem:

Tabela 2.1: *Pinagem do conector para RS232*

Pino	Nome	Função
9	TX	Transmissão de dados (ligado ao RX do mestre)
10	RX	Recepção de dados (ligado ao TX do mestre)
11	GND	Referência para circuito RS232

2.1.3 Conexão com a Rede RS232

- Os sinais RX e TX do escravo devem ser ligados respectivamente aos sinais TX e RX do mestre, além da conexão do sinal de referência (GND).
- A interface RS232 é muito susceptível a interferências. Por este motivo, o cabo utilizado para comunicação deve ser o mais curto possível – sempre menor que 10 metros – e deve ser colocado em separado da fiação de potência que alimenta outros equipamentos.

2.2 RS485

2.2.1 Características da interface RS485

- Interface segue o padrão EIA/TIA-485.
- Pode operar como escravo ou mestre da rede Modbus RTU.
- Possibilita comunicação utilizando taxas de 1200 até 57600 Kbit/s.
- Interface isolada galvanicamente e com sinal diferencial, conferindo maior robustez contra interferência eletromagnética.
- Permite a conexão de até 32 dispositivos no mesmo segmento. Uma quantidade maior de dispositivos pode ser conectada com o uso de repetidores[1].
- Comprimento máximo do barramento de 1000 metros.

2.2.2 Pinagem do Conector

A conexão para a interface RS485 está disponível através do conector XC3 utilizando a seguinte pinagem:

Tabela 2.2: *Pinagem do conector para RS485*

Pino	Nome	Função
12	A-Line (-)	RxD/TxD negativo
13	B-Line (+)	RxD/TxD positivo
14	GND	0V isolado do circuito RS485

[1] O número limite de equipamentos que podem ser conectados na rede também depende do protocolo utilizado.

2.2.3 Indicações

Além dos marcadores de sistema, que fornecem diversas informações sobre a interface RS485, o controlador programável PLC300 possui um LED bicolor – verde e vermelho – na parte frontal do produto utilizado para indicação da interface Serial.

Figura 2.1: *LED de indicação da interface Serial*

Durante a inicialização do equipamento, ambos os LEDs são acesos para teste por um período de aproximadamente 500 ms alternadamente. Após este período, para a interface RS485, eles farão as seguintes indicações.

- LED Verde: acende sempre que um telegrama é transmitido pela interface RS485.
- LED Vermelho: acende sempre que um byte é recebido incorretamente (erro de paridade ou frame) ou detectado erro de CRC no telegrama recebido pela interface RS485.

2.2.4 Chaves para Habilitação do Resistor de Terminação

Para cada segmento da rede RS485, é necessário habilitar um resistor de terminação nos pontos extremos do barramento principal. O PLC300 possui chaves (S1) que podem ser ativadas (colocando ambas as chaves na posição ON) para habilitar o resistor de terminação.

2.2.5 Conexão com a Rede RS485

Para a ligação do equipamento utilizando a interface RS485, os seguintes pontos devem ser observados:

- É recomendado o uso de um cabo com par trançado blindado.
- Recomenda-se também que o cabo possua mais um fio para ligação do sinal de referência (GND). Caso o cabo não possua o fio adicional, deve-se deixar o sinal GND desconectado.
- A passagem do cabo deve ser feita separadamente (e se possível distante) dos cabos para alimentação de potência.
- Todos os dispositivos da rede devem estar devidamente aterrados, preferencialmente na mesma ligação com o terra. A blindagem do cabo também deve ser aterrada.
- Habilitar os resistores de terminação apenas em dois pontos, nos extremos do barramento principal, mesmo que existam derivações a partir do barramento.

3 CONFIGURAÇÃO DAS INTERFACES

Para realizar a configuração das interfaces RS232 e RS485, através do Setup do controlador programável PLC300 são disponibilizados os seguintes menus:

3.1 CONFIGURAÇÃO RS232

BAUD RATE

Faixa de Valores:	0 = 1200 bit/s 1 = 2400 bit/s 2 = 4800 bit/s 3 = 9600 bit/s 4 = 19200 bit/s 5 = 38400 bit/s 6 = 57600 bit/s	**Padrão:** 4

Descrição:
Permite programar o valor desejado para a taxa de comunicação da interface serial, em bits por segundo. Esta taxa deve ser a mesma para todos os equipamentos conectados na rede.

PARIDADE

Faixa de Valores:	0 = sem paridade 1 = paridade ímpar 2 = paridade par	**Padrão:** 2

Descrição:
Permite a configuração da paridade nos bytes da interface serial. Esta configuração deve ser a mesma para todos os equipamentos conectados na rede.

STOP BITS

Faixa de Valores:	0 = 1 stop bit 1 = 2 stop bits	**Padrão:** 0

Descrição:
Permite a configuração dos stop bits nos bytes da interface serial. Esta configuração deve ser a mesma para todos os equipamentos conectados na rede.

NOTA!
O endereço do escravo Modbus RTU via interface RS232 é fixo em 1.

3.2 CONFIGURAÇÃO RS485

BAUD RATE

Faixa de Valores:	0 = 1200 bit/s 1 = 2400 bit/s 2 = 4800 bit/s 3 = 9600 bit/s 4 = 19200 bit/s 5 = 38400 bit/s 6 = 57600 bit/s	**Padrão:** 4

Descrição:
Permite programar o valor desejado para a taxa de comunicação da interface serial, em bits por segundo. Esta taxa deve ser a mesma para todos os equipamentos conectados na rede.

PARIDADE

Faixa de Valores:	0 = sem paridade 1 = paridade ímpar 2 = paridade par	**Padrão:** 2

Descrição:
Permite a configuração da paridade nos bytes da interface serial. Esta configuração deve ser a mesma para todos os equipamentos conectados na rede.

STOP BITS

Faixa de Valores:	0 = 1 stop bit 1 = 2 stop bits	**Padrão:** 0

Descrição:
Permite a configuração dos stop bits nos bytes da interface serial. Esta configuração deve ser a mesma para todos os equipamentos conectados na rede.

MODO DE OPERAÇÃO

Faixa de Valores:	0 = escravo 1 = mestre	**Padrão:** 1

Descrição:
Via interface RS485, o PLC300 possui dois modos de operação na rede Modbus RTU:

- *Escravo:* como escravo da rede, ele disponibiliza funções para leitura e escrita dos dados e marcadores utilizados na configuração e programação em ladder do produto. Para maiores informações sobre este modo de operação, consulte o item 5.
- *Mestre:* como mestre da rede, o PLC300 disponibiliza blocos em ladder para envio de comandos para os escravos da rede, conforme configurado nestes blocos. Neste modo, não será possível acessar os dados e configurações do PLC300 via interface RS485. Apenas um mestre pode estar configurado para operar no barramento RS485. Para maiores informações sobre este modo de operação, consulte o item 7 e a documentação do software de programação WPS.

ENDEREÇO DO ESCRAVO

Faixa de Valores:	1 a 247	**Padrão:** 1

Descrição:
Permite configurar o endereço do PLC300 como escravo na rede Modbus RTU via interface RS485. Este endereço é utilizado apenas se a interface estiver programada no modo escravo, não possui função se o PLC300 for programado como mestre da rede.

4 PROTOCOLO MODBUS RTU

O protocolo Modbus foi inicialmente desenvolvido em 1979. Atualmente, é um protocolo aberto amplamente difundido, utilizado por vários fabricantes em diversos equipamentos.

4.1 MODOS DE TRANSMISSÃO

Na especificação do protocolo estão definidos dois modos de transmissão: ASCII e RTU. Os modos definem a forma como são transmitidos os bytes da mensagem. Não é possível utilizar os dois modos de transmissão na mesma rede.

O controlador programável PLC300 utiliza somente o modo RTU para a transmissão de telegramas. Os bytes são de acordo com a configuração feita através do setup do equipamento.

4.2 ESTRUTURA DAS MENSAGENS NO MODO RTU

A rede Modbus RTU utiliza o sistema mestre-escravo para a troca de mensagens. Permite até 247 escravos, mas somente um mestre. Toda comunicação inicia com o mestre fazendo uma solicitação a um escravo, e este responde ao mestre o que foi solicitado. Em ambos os telegramas (pergunta e resposta), a estrutura utilizada é a mesma: Endereço, Código da Função, Dados e CRC. Apenas o campo de dados poderá ter tamanho variável, dependendo do que está sendo solicitado.

Mestre (telegrama de requisição):

Endereço (1 byte)	Função (1 byte)	Dados da requisição (n bytes)	CRC (2 bytes)

Escravo (telegrama de resposta):

Endereço (1 byte)	Função (1 byte)	Dados da resposta (n bytes)	CRC (2 bytes)

4.2.1 Endereço

O mestre inicia a comunicação enviando um byte com o endereço do escravo para o qual se destina a mensagem. Ao enviar a resposta, o escravo também inicia o telegrama com o seu próprio endereço. O mestre também pode enviar uma mensagem destinada ao endereço 0 (zero), o que significa que a mensagem é destinada a todos os escravos da rede (broadcast). Neste caso, nenhum escravo irá responder ao mestre.

4.2.2 Código da Função

Este campo também contém um único byte, onde o mestre especifica o tipo de serviço ou função solicitada ao escravo (leitura, escrita, etc.). De acordo com o protocolo, cada função é utilizada para acessar um tipo específico de dado.

Para a lista de funções disponíveis para acesso aos dados, consulte o item 5.

4.2.3 Campo de Dados

Campo com tamanho variável. O formato e conteúdo deste campo dependem da função utilizada e dos valores transmitidos. Este campo está descrito juntamente com a descrição das funções (ver item 5).

4.2.4 CRC

A última parte do telegrama é o campo para checagem de erros de transmissão. O método utilizado é o CRC-16 (Cycling Redundancy Check). Este campo é formado por dois bytes, onde primeiro é transmitido o byte menos significativo (CRC-), e depois o mais significativo (CRC+). A forma de cálculo do CRC é descrita na especificação do protocolo, porém informações para sua implementação também são fornecidas no Apêndice B.

4.2.5 Tempo entre Mensagens

No modo RTU não existe um caracter específico que indique o início ou o fim de um telegrama. A indicação de quando uma nova mensagem começa ou quando ela termina é feita pela ausência de transmissão de dados na rede, por um tempo mínimo de 3,5 vezes o tempo de transmissão de um byte de dados (11 bits). Sendo assim, caso um telegrama tenha iniciado após a decorrência deste tempo mínimo, os elementos da rede irão assumir que o primeiro caracter recebido representa o início de um novo telegrama. E da mesma forma, os elementos da rede irão assumir que o telegrama chegou ao fim quando, recebidos os bytes do telegrama, este tempo decorra novamente.

Se durante a transmissão de um telegrama, o tempo entre os bytes for maior que este tempo mínimo, o telegrama será considerado inválido, pois o controlador programável irá descartar os bytes já recebidos e montará um novo telegrama com os bytes que estiverem sendo transmitidos.

Para taxas de comunicação superiores a 19200 bits/s, os tempos utilizados são os mesmos que para esta taxa. A tabela a seguir nos mostra os tempos para diferentes taxas de comunicação:

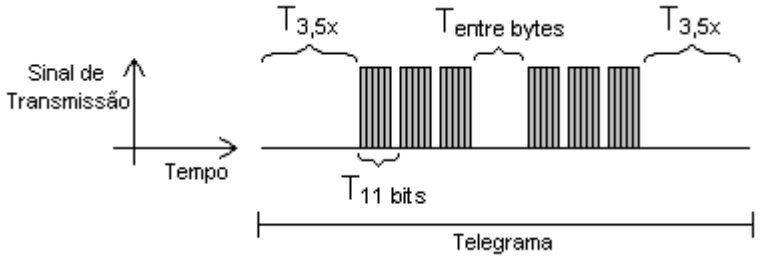


Tabela 4.1: *Taxas de comunicação e tempos envolvidos na transmissão de telegramas*

Taxa de Comunicação	T11 bits	T3,5x
1200 bits/s	9,167 ms	32,083 ms
2400 bits/s	4,583 ms	16,042 ms
4800 bits/s	2,292 ms	8,021 ms
9600 bits/s	1,146 ms	4,010 ms
19200 bits/s	573 µs	2,005 ms
38400 bits/s	573 µs	2,005 ms
57600 bits/s	573 µs	2,005 ms

- $T_{11\ bits}$ = Tempo para transmitir uma palavra do telegrama.
- $T_{entre\ bytes}$ = Tempo entre bytes.
- $T_{3,5x}$ = Intervalo mínimo para indicar começo e fim de telegrama (3,5 x T_{11bits}).

5 OPERAÇÃO NA REDE MODBUS RTU – MODO ESCRAVO

Como escravo da rede Modbus RTU, o controlador programável PLC300 possui as seguintes características:

- Conexão da rede via interface serial RS232 ou RS485.
- Taxa de comunicação, formato dos bytes e endereçamento[2] definidos através do setup do equipamento.
- Permite acesso a todos os marcadores e dados utilizados para programação em ladder do PLC300.

NOTA!
As interfaces RS232, RS485 (no modo escravo), USB e Ethernet, pelo fato de utilizarem as mesmas funções para acesso aos dados e programação do equipamento, não devem ser utilizadas simultaneamente para realizar funções de download de programa ou monitoração online do controlador programável PLC300, pois podem ocorrer conflitos durante o acesso simultâneo aos dados.

5.1 FUNÇÕES DISPONÍVEIS E TEMPOS DE RESPOSTA

Na especificação do protocolo Modbus são definidas funções utilizadas para acessar diferentes tipos de dados. No PLC300, para acessar estes dados, foram disponibilizados os seguintes serviços (ou funções):

- Read Coils
  Descrição: leitura de bloco de bits do tipo *coil*.
  Código da função: 01.

- Read Discrete Inputs
  Descrição: leitura de bloco de bits do tipo entradas discretas.
  Código da função: 02.

- Read Holding Registers
  Descrição: leitura de bloco de registradores do tipo *holding*.
  Código da função: 03.

- Read Input Registers
  Descrição: leitura de bloco de registradores do tipo *input*.
  Código da função: 04.

- Write Single Coil
  Descrição: escrita em um único bit do tipo *coil*.
  Código da função: 05.

- Write Single Register
  Descrição: escrita em um único registrador do tipo *holding*.
  Código da função: 06.

- Write Multiple Coils
  Descrição: escrita em bloco de bits do tipo *coil*.
  Código da função: 15.

- Write Multiple Registers
  Descrição: escrita em bloco de registradores do tipo *holding*.
  Código da função: 16.

- Read Device Identification
  Descrição: identificação do modelo do dispositivo.
  Código da função: 43.

O tempo de resposta, do final da transmissão do mestre até o início da resposta do escravo, varia desde o tempo mínimo entre bytes na comunicação Modbus RTU até o valor do ciclo de scan do equipamento.

[2] Endereço programável apenas para interface RS485, para interface RS232 o endereço é fixo em 1.

5.2 MAPA DE MEMÓRIA

O controlador programável PLC300 possui diferentes tipos de dados acessíveis através da comunicação Modbus. Estes dados são mapeados em endereços de dados e funções de acesso conforme descrito nos itens seguintes.

NOTA!
O software de programação WPS possui listas que permitem a visualização de todos os tipos de marcadores disponíveis para o PLC300. Nestas listas, existe um campo para indicação do endereço do registrador Modbus para acesso ao marcador.

5.2.1 Marcadores de Sistema de Leitura – %SB / %SW / %SD

Os marcadores de sistema de leitura representam os dados do PLC300 utilizados para indicações de estado e monitoração de funções do equipamento.

- Acesso: somente leitura.
- Tipo de dado: *input register* ou *input discrete*.
- Funções de acesso Modbus: 02 e 04.
- Faixa de endereço Modbus para acesso via *input register*: 3000 ... 4999.
- Faixa de endereço Modbus para acesso via *input discrete*: 0 ... 15999.

Os marcadores de sistema relacionados com a comunicação serial disponíveis para o PLC300 estão descritos no item 8. Para a descrição de outros marcadores disponíveis e função de cada marcador, consulte o manual do usuário do PLC300.

5.2.2 Marcadores de Sistema de Escrita – %CB / %CW / %CD

Os marcadores de sistema de escrita representam os dados do PLC300 utilizados para configuração e controle das funções do equipamento.

- Acesso: leitura/escrita.
- Tipo de dado: *holding register* ou *coil*.
- Funções de acesso Modbus: 01, 03, 05, 06, 15 e 16.
- Faixa de endereço Modbus para acesso via *holding register*: 3000 ... 4999.
- Faixa de endereço Modbus para acesso via *coil*: 0 ... 15999.

Os marcadores de sistema relacionados com a comunicação serial disponíveis para o PLC300 estão descritos no item 8. Para a descrição de outros marcadores disponíveis e função de cada marcador, consulte o manual do usuário do PLC300.

5.2.3 Inputs – %IB / %IW / %ID

Marcadores que representam dados relativos a entradas digitais e analógicas físicas, disponíveis no hardware do PLC300.

- Acesso: somente leitura.
- Tipo de dado: *input register* ou *input discrete*.
- Funções de acesso Modbus: 02 e 04.
- Faixa de endereço Modbus para acesso via *input register*: 5000 ... 5999.
- Faixa de endereço Modbus para acesso via *input discrete*: 16000 ... 23999.

Para a descrição exata de quais marcadores estão disponíveis e função de cada marcador, consulte o manual do usuário do PLC300.

5.2.4 Outputs – %QB / %QW / %QD

Marcadores que representam dados relativos a saídas digitais e analógicas físicas, disponíveis no hardware do PLC300.

- Acesso: leitura/escrita.

- Tipo de dado: *holding register* ou *coil*.
- Funções de acesso Modbus: 01, 03, 05, 06, 15 e 16.
- Faixa de endereço Modbus para acesso via *holding register*: 5000 ... 5999.
- Faixa de endereço Modbus para acesso via *coil*: 16000 ... 23999.

Para a descrição exata de quais marcadores estão disponíveis e função de cada marcador, consulte o manual do usuário do PLC300.

5.2.5 Inputs de rede – %IB / %IW / %ID

Marcadores que representam dados relativos a valores recebidos através das interfaces de rede do PLC300. Possuem a mesma nomenclatura das entradas físicas, mas sua numeração inicia a partir do marcador 2000 (exemplo: %IB2000).

- Acesso: somente leitura.
- Tipo de dado: *input register* ou *input discrete*.
- Funções de acesso Modbus: 02 e 04.
- Faixa de endereço Modbus para acesso via *input register*: 6000 ... 7999.
- Faixa de endereço Modbus para acesso via *input discrete*: 24000 ... 39999.

5.2.6 Outputs de rede – %QB / %QW / %QD

Marcadores que representam dados relativos a valores transmitidos através das interfaces de rede do PLC300. Possuem a mesma nomenclatura das saídas físicas, mas sua numeração inicia a partir do marcador 2000 (exemplo: %QB2000).

- Acesso: leitura/escrita.
- Tipo de dado: *holding register* ou *coil*.
- Funções de acesso Modbus: 01, 03, 05, 06, 15 e 16.
- Faixa de endereço Modbus para acesso via *holding register*: 6000 ... 7999.
- Faixa de endereço Modbus para acesso via *coil*: 24000 ... 39999.

5.2.7 Marcadores em Memória – %MB / %MW / %MD

Marcadores de uso geral para programação em ladder do PLC300. Representam as variáveis globais, criadas dinamicamente durante a elaboração do programa no software WPS.

- Acesso: leitura/escrita.
- Tipo de dado: *holding register* ou *coil*.
- Funções de acesso Modbus: 01, 03, 05, 06, 15 e 16.
- Marcadores voláteis:
  - Faixa de endereço Modbus para acesso via holding register: 8000 ... 27999.
  - Faixa de endereço Modbus para acesso via coil: 40000 ... 49999.
- Marcadores retentivos
  - Faixa de endereço Modbus para acesso via holding register: 28000 ... 47999.
  - Faixa de endereço Modbus para acesso via coil: 50000 ... 59999.

A quantidade de marcadores disponíveis nesta área é dependente dos marcadores criados no software de programação do PLC300. Para que seja possível acessar o marcador desejado, primeiramente é necessário criar este marcador e fazer o download do programa do usuário utilizando o software de programação.

NOTA!
A quantidade de dados acessíveis via coils e input discretes não corresponde a toda a área em memória acessível via registradores. Por exemplo, caso seja criada uma quantidade de marcadores em memória maior que a quantidade acessível via coil (10000 bits = 1250 bytes), os marcadores adicionais somente poderão ser acessados via holding registers.

5.3 ACESSO AOS DADOS

Cada uma das regiões de memória descritas anteriormente é distribuída em bytes. O protocolo Modbus, porém, permite que o acesso seja feito apenas por bits ou por registradores de 16 bits. Para acessar estas

regiões de memória, é necessário então fazer a relação entre o tipo e a numeração do dado no PLC300 com o tipo e o endereço Modbus. As tabelas a seguir mostram como é feita a relação entre a numeração do dado no PLC300 e o endereço dos registradores Modbus que acessam estes dados.

Marcadores de Sistema de Leitura		
Numeração do Marcador PLC300		**Endereço do Registrador (input register) Modbus**
%SB3001	%SB3000	3000
%SB3003	%SB3002	3001
⋮		⋮
%SB3101	%SB3100	3050
⋮		⋮

Marcadores de Sistema de Escrita		
Numeração do Marcador PLC300		**Endereço do Registrador (holding register) Modbus**
%CB3001	%CB3000	3000
%CB3003	%CB3002	3001
⋮		⋮
%CB3101	%CB3100	3050
⋮		⋮

Inputs		
Numeração do Marcador PLC300		**Endereço do Registrador (input register) Modbus**
%IB1	%IB0	5000
%IB3	%IB2	5001
⋮		⋮
%IB2001	%IB2000	6000
⋮		⋮

Outputs		
Numeração do Marcador PLC300		**Endereço do Registrador (holding register) Modbus**
%QB1	%QB0	5000
%QB3	%QB2	5001
⋮		⋮
%QB2001	%QB2000	6000
⋮		⋮

Marcadores (voláteis e retentivos)		
Numeração do Marcador PLC300		**Endereço do Registrador (holding register) Modbus**
%MB1	%MB0	8000
%MB3	%MB2	8001
⋮		⋮
%MB40001	%MB40000	28000
⋮		⋮

A tabela a seguir exemplifica como é calculado o endereço Modbus com acesso via registradores, para diferentes tipos de dados disponíveis para o PLC300:

Dado	Descrição	Tipo do dado	Endereço base	Offset a partir do endereço base	Endereço Modbus
%SW3002	Marcador de sistema de leitura, que representa o tempo de ciclo de scan.	Input Register	3000	2 bytes (1 word)	3001
%CW3030	Marcador de sistema de escrita, para ajuste da hora do RTC.	Holding Register	3000	30 bytes (15 words)	3015
%IB0	Inputs físicos, representando as entradas digitais 1 até 8.	Input Register	5000	0 bytes (0 words)	5000 byte baixo
%MB11	Marcador em memória volátil, representando uma variável global criada pelo usuário com tamanho de um byte.	Holding Register	8000	11 bytes (5 words)	8005 byte alto
%MD40004	Marcador em memória retentivo, representando uma variável global criada pelo usuário com tamanho de quatro bytes.	Holding Register	28000	4 bytes (2 words)	28002 e 28003

De forma semelhante, o acesso via dados binários (coils ou input discretes) também utiliza um endereço base mais o offset dados pelo número do marcador. No entanto, como cada byte possui oito bits, para cada byte a partir do endereço base devem ser adicionados oito bits no endereço para acesso via dados binários.

O formato e a função do dado na área de memória acessada, no entanto, não é pré-definido, e depende da programação feita no software WPS. Por exemplo, para o marcador de memória %M_0 é possível criar as seguintes variáveis no software WPS:

- *%MB0:* marcador de byte, ocupa apenas um byte de memória, podendo representar um inteiro de 8 bits com ou sem sinal. No acesso via registradores, como o protocolo Modbus permite o acesso de leitura ou escrita de pelo menos 16 bits, sempre que este marcador for lido ou escrito, os bytes %MB0 e %MB1 serão acessados.
- *%MW0:* marcador de word, ocupa dois bytes de memória, podendo representar um inteiro de 16 bits com ou sem sinal. Neste caso, os bytes %MB0 e %MB1 serão reservados para este marcador.
- *%MD0:* marcador de double, ocupa quatro bytes de memória, podendo representar um inteiro de 32 bits com ou sem sinal, ou então uma variável do tipo float. Neste caso, os bytes %MB0 até %MB3 serão reservados para este marcador. No acesso por registradores, é necessário fazer a leitura ou escrita de dois registradores em sequência, com o valor menos significativo no primeiro registrador, para que os quatro bytes sejam acessados.

Tabela 5.1: *Exemplo de endereçamento de dados para marcadores voláteis no PLC300*

Tipo de Marcador / End. Modbus Registrador (bit)	Byte (%MB)	Word (%MW)	Double (%MD)
8000 (40000 ... 40015)	X	X	X
	X		
8001 (40016 ... 40031)	X	X	
	X		
8002 (40032 ... 40047)	X	X	X
	X		
8003 (40048 ... 40063)	X	X	
	X		

De forma similar, é possível fazer o acesso aos dados utilizando as funções de acesso a bits. Neste caso, pode-se fazer acesso a um bit individualmente, ou a um grupo de bits que representa um marcador. Por exemplo, se for definido no software WPS um marcador do tipo word no endereço 8000 – %MW0 – é possível acessar este marcador, utilizando as funções de leitura ou escrita múltipla de coils, utilizando os bits 40000 até 40015.

Nos endereços de memória do PLC300, variáveis com tamanho superior a um byte são armazenadas sempre com o byte menos significativo primeiro. Desta fora, a disposição em memória para valores de Byte, Word ou Double segue o descrito pela tabela a seguir.

Tabela 5.2: *Exemplo de endereçamento de dados para marcadores voláteis no PLC300*

Tipo de Marcador / End. Modbus Registrador (bit)	Byte (%MB)		Word (%MW)		Double (%MD)	
8000 (40000 ... 40015)	%MB0	Valor único	%MW0	Valor -signf. Valor +signf.	%MD0	Valor -signf. ... Valor +signf
	%MB1	Valor único				
8001 (40016 ... 40031)	%MB2	Valor único	%MW2	Valor -signf. Valor +signf.		
	%MB3	Valor único				
8002 (40032 ... 40047)	%MB4	Valor único	%MW4	Valor -signf. Valor +signf.	%MD4	Valor -signf. ... Valor +signf
	%MB5	Valor único				
8003 (40048 ... 40063)	%MB6	Valor único	%MW6	Valor -signf. Valor +signf.		
	%MB7	Valor único				

Como o protocolo Modbus define que, para transmitir um registrador de 16 bits, deve-se transmitir sempre o byte mais significativo primeiro, ao acessar qualquer registrador, o endereço seguinte de memória é transmitido primeiro. Desta forma, caso sejam lidos 4 registradores em sequência, a partir do registrador 8000, o conteúdo de cada registrador será transmitido da seguinte forma:

1º Registrador – 8000		2º Registrador – 8001		3º Registrador – 8002		4º Registrador – 8003	
%MB1	%MB0	%MB3	%MB2	%MB5	%MB4	%MB7	%MB6

NOTA!
Para o controlador programável PLC300, o tamanho máximo de cada telegrama, incluindo endereço, função, campo de dados e CRC, não deve ultrapassar 67 bytes.

6 DESCRIÇÃO DETALHADA DAS FUNÇÕES

Neste item é feita uma descrição detalhada das funções disponíveis no controlador programável PLC300 para comunicação Modbus RTU. Para a elaboração dos telegramas, é importante observar o seguinte:

- Os valores são sempre mostrados em hexadecimal.
- O endereço de um dado, o número de dados e o valor de registradores são sempre representados em 16 bits. Por isso, é necessário transmitir estes campos utilizando dois bytes – superior (high) e inferior (low).
- Os telegramas, tanto para pergunta quanto para resposta, não podem ultrapassar 64 bytes.

6.1 FUNÇÃO 01 – READ COILS

Lê o conteúdo de um grupo de bits internos que necessariamente devem estar em sequência numérica. Esta função possui a seguinte estrutura para os telegramas de leitura e resposta (cada campo representa um byte):

Pergunta (Mestre)	**Resposta (Escravo)**
Endereço do escravo	Endereço do escravo
Função	Função
Endereço do bit inicial (byte high)	Campo Byte Count (no. de bytes de dados)
Endereço do bit inicial (byte low)	Byte 1
Quantidade de bits (byte high)	Byte 2
Quantidade de bits (byte low)	Byte 3
CRC-	etc...
CRC+	CRC-
	CRC+

Cada bit da resposta é colocado em uma posição dos bytes de dados enviados pelo escravo. O primeiro byte recebe os 8 primeiros bits a partir do endereço inicial indicado pelo mestre. Os demais bytes continuam a sequência, caso o número de bits de leitura seja maior que 8. Caso o número de bits lidos não seja múltiplo de 8, os bits restantes do último byte devem ser preenchidos com 0 (zero).

Exemplo: leitura dos 8 bits do marcador de saída 2000, mapeado como coil a partir do endereço 24000, supondo este marcador com o valor 100 (64h).

- Endereço: 1 = 01h
- Número do bit inicial: 24000 = 5DC0h
- Número de bits lidos: 8 = 0008h

Pergunta (Mestre)		**Resposta (Escravo)**	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	01h	Função	01h
Bit inicial (high)	5Dh	Byte Count	01h
Bit inicial (low)	C0h	Estado dos bits 1 até 8	64h
Quantidade de bits (high)	00h	CRC-	50h
Quantidade de bits (low)	08h	CRC+	63h
CRC-	2Eh		
CRC+	5Ch		

6.2 FUNÇÃO 02 – READ INPUT DISCRETE

NOTA!
A função 02 – Read Input Discrete – possui exatamente a mesma estrutura da função 01. Somente o código da função e os dados acessíveis são diferentes.

6.3 FUNÇÃO 03 – READ HOLDING REGISTER

Lê o conteúdo de um grupo de registradores, que necessariamente devem estar em sequência numérica. Esta função possui a seguinte estrutura para os telegramas de leitura e resposta (cada campo representa um byte):

Pergunta (Mestre)	Resposta (Escravo)
Endereço do escravo	Endereço do escravo
Função	Função
Endereço do registrador inicial (byte high)	Campo Byte Count
Endereço do registrador inicial (byte low)	Dado 1 (high)
Quantidade de registradores (byte high)	Dado 1 (low)
Quantidade de registradores (byte low)	Dado 2 (high)
CRC-	Dado 2 (low)
CRC+	etc...
	CRC-
	CRC+

Exemplo: leitura do marcador em memória %MD0, representando um float IEEE que ocupa 4 bytes em memória. Supondo o valor do float igual à 1,0 (3F800000h em representação de float IEEE).

- Endereço: 1 = 01h
- Endereço do registrador inicial: 8000 = 1F40h
- Quantidade de registradores lidos: 2 = 0002h

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	03h	Função	03h
Registrador inicial (high)	1Fh	Byte Count	04h
Registrador inicial (low)	40h	Valor do float (low-high)	00h
Quantidade de registradores (high)	00h	Valor do float (low-low)	00h
Quantidade de registradores (low)	02h	Valor do float (high-high)	3Fh
CRC-	02h	Valor do float (high-low)	80h
CRC+	08h	CRC-	F7h
		CRC+	CFh

6.4 FUNÇÃO 04 – READ INPUT REGISTER

NOTA!
A função 04 – Read Input Register – possui exatamente a mesma estrutura da função 03. Somente o código da função e os dados acessíveis são diferentes.

6.5 FUNÇÃO 05 – WRITE SINGLE COIL

Esta função é utilizada para escrever um valor para um único bit (coil). O valor para o bit é representado utilizando dois bytes, onde o valor FF00h representa o bit igual a 1, e o valor 0000h representa o bit igual a 0 (zero). Esta função possui a seguinte estrutura (cada campo representa um byte):

Pergunta (Mestre)	Resposta (Escravo)
Endereço do escravo	Endereço do escravo
Função	Função
Endereço do bit (byte high)	Endereço do bit (byte high)
Endereço do bit (byte low)	Endereço do bit (byte low)
Valor para o bit (byte high)	Valor para o bit (byte high)
Valor para o bit (byte low)	Valor para o bit (byte low)
CRC-	CRC-
CRC+	CRC+

Exemplo: escrita do primeiro bit do marcador de saída %QB0, mapeado como coil a partir do endereço 16000.

- Endereço: 1 = 01h
- Número do bit: 16000 = 3E80h
- Valor para o bit: 1, logo o valor que deve ser escrito é FF00h

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	05h	Função	05h
Número do bit (high)	3Eh	Número do bit (high)	3Eh
Número do bit (low)	80h	Número do bit (low)	80h
Valor para o bit (high)	FFh	Valor para o bit (high)	FFh
Valor para o bit (low)	00h	Valor para o bit (low)	00h
CRC-	80h	CRC-	8Ah
CRC+	3Ah	CRC+	3Ah

Note que para esta função, a resposta do escravo é uma cópia idêntica da requisição feita pelo mestre.

6.6 FUNÇÃO 06 – WRITE SINGLE REGISTER

Esta função é utilizada para escrever um valor para um único registrador. Esta função possui a seguinte estrutura (cada campo representa um byte):

Pergunta (Mestre)	Resposta (Escravo)
Endereço do escravo	Endereço do escravo
Função	Função
Endereço do registrador (byte high)	Endereço do registrador (byte high)
Endereço do registrador (byte low)	Endereço do registrador (byte low)
Valor para o registrador (byte high)	Valor para o registrador (byte high)
Valor para o registrador (byte low)	Valor para o registrador (byte low)
CRC-	CRC-
CRC+	CRC+

Exemplo: escrita do marcador de sistema de escrita %CB3000. Como a escrita é feita sempre enviando um registrador de 16 bits, os bytes mapeados nos endereços %CB3000 e %CB3001 serão escritos.

- Endereço: 1 = 01h
- Endereço do registrador inicial: 3000 = 0BB8h
- Valor para o marcador: 50 = 0032h

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	06h	Função	06h
Registrador (high)	0Bh	Registrador (high)	0Bh
Registrador (low)	B8h	Registrador (low)	B8h
Valor (high – equivale ao valor para %CB3001)	00h	Valor (high – equivale ao valor para %CB3001)	00h
Valor (low – equivale ao valor para %CB3000)	32h	Valor (low – equivale ao valor para %CB3000)	32h
CRC-	8Ah	CRC-	8Ah
CRC+	1Eh	CRC+	1Eh

Note que para esta função, a resposta do escravo é uma cópia idêntica da requisição feita pelo mestre.

6.7 FUNÇÃO 15 – WRITE MULTIPLE COILS

Esta função permite escrever valores para um grupo de bits (coils), que devem estar em sequência numérica. Também pode ser usada para escrever um único bit (cada campo representa um byte).

Pergunta (Mestre)	Resposta (Escravo)
Endereço do escravo	Endereço do escravo
Função	Função
Endereço do bit inicial (byte high)	Endereço do bit inicial (byte high)
Endereço do bit inicial (byte low)	Endereço do bit inicial (byte low)
Quantidade de bits (byte high)	Quantidade de bits (byte high)
Quantidade de bits (byte low)	Quantidade de bits (byte low)
Campo Byte Count (no. de bytes de dados)	CRC-
Byte 1	CRC+
Byte 2	
Byte 3	
etc...	
CRC-	
CRC+	

O valor de cada bit que está sendo escrito é colocado em uma posição dos bytes de dados enviados pelo mestre. O primeiro byte recebe os 8 primeiros bits a partir do endereço inicial indicado pelo mestre. Os demais bytes (se o número de bits escritos for maior que 8) continuam a sequência. Caso o número de bits escritos não seja múltiplo de 8, os bits restantes do último byte devem ser preenchidos com 0 (zero).

Exemplo: escrita de 16 bits a partir do marcador de saída %QW0, mapeado como coil a partir do endereço 16000.

- Endereço: 1 = 01h
- Número do primeiro bit: 16000 = 3E80h
- Quantidade de bits: 16 = 0010h
- Valor para os bits 0 até 7: 10 = 0Ah
- Valor para os bits 8 até 15: 20 = 14h

Pergunta (Mestre)		**Resposta (Escravo)**	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	0Fh	Função	0Fh
Bit inicial (byte high)	3Eh	Bit inicial (byte high)	1Fh
Bit inicial (byte low)	80h	Bit inicial (byte low)	40h
Quantidade de bits (byte high)	00h	Quantidade de bits (byte high)	00h
Quantidade de bits (byte low)	10h	Quantidade de bits (byte low)	10h
Byte Count	02h	CRC-	52h
Valor para os bits	0Ah	CRC+	07h
Valor para os bits	14h		
CRC-	24h		
CRC+	8Ch		

6.8 FUNÇÃO 16 – WRITE MULTIPLE REGISTERS

Esta função permite escrever valores para um grupo de registradores, que devem estar em sequência numérica. Também pode ser usada para escrever um único registrador (cada campo representa um byte).

Pergunta (Mestre)	**Resposta (Escravo)**
Endereço do escravo	Endereço do escravo
Função	Função
Endereço do registrador inicial (byte high)	Endereço do registrador inicial (byte high)
Endereço do registrador inicial (byte low)	Endereço do registrador inicial (byte low)
Quantidade de registradores (byte high)	Quantidade de registradores (byte high)
Quantidade de registradores (byte low)	Quantidade de registradores (byte low)
Campo Byte Count (nº de bytes de dados)	CRC-
Dado 1 (high)	CRC+
Dado 1 (low)	
Dado 2 (high)	
Dado 2 (low)	
etc...	
CRC-	
CRC+	

Exemplo: escrita do marcador de memória de escrita %MD0, representando um valor inteiro de 32 bits – 4 bytes em memória. Supondo o valor a ser escrito igual a 16909060 decimal (01020304h)

- Endereço: 1 = 01h
- Endereço do registrador inicial: 8000 = 1F40h
- Quantidade de registradores escritos: 2 = 0002h

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	10h	Função	10h
Registrador inicial (high)	1Fh	Registrador inicial (high)	1Fh
Registrador inicial (low)	40h	Registrador inicial (low)	40h
Quantidade de registradores (high)	00h	Quantidade de registradores (high)	00h
Quantidade de registradores (low)	02h	Quantidade de registradores (low)	02h
Byte Count	04h	CRC-	47h
Valor para o inteiro (low-high)	03h	CRC+	C8h
Valor para o inteiro (low-low)	04h		
Valor para o inteiro (high-high)	01h		
Valor para o inteiro (high-low)	02h		
CRC-	BAh		
CRC+	7Bh		

6.9 FUNÇÃO 43 – READ DEVICE IDENTIFICATION

Função auxiliar, que permite a leitura do fabricante, modelo e versão de firmware do produto. Possui a seguinte estrutura:

Pergunta (Mestre)	Resposta (Escravo)
Endereço do escravo	Endereço do escravo
Função	Função
MEI Type	MEI Type
Código de leitura	Conformity Level
Número do Objeto	More Follows
CRC-	Próximo objeto
CRC+	Número de objetos
	Código do primeiro objeto
	Tamanho do primeiro objeto
	Valor do primeiro objeto (n bytes)
	Código do segundo objeto
	Tamanho do segundo objeto
	Valor do segundo objeto (n bytes)
	etc...
	CRC-
	CRC+

Esta função permite a leitura de três categorias de informações: Básica, Regular e Estendida, e cada categoria é formada por um grupo de objetos. Cada objeto é formado por uma sequência de caracteres ASCII. Para o controlador programável PLC300, apenas informações básicas estão disponíveis, formadas por três objetos:

- Objeto 00h – VendorName: representa o nome do fabricante do produto.
- Objeto 01h – ProductCode: formado pelo código do produto (PLC300).
- Objeto 02h – MajorMinorRevision: indica a versão de firmware do produto, no formato 'VX.XX'.

O código de leitura indica quais as categorias de informações são lidas, e se os objetos são acessados em sequência ou individualmente. No caso, o PLC300 suporta os códigos 01 (informações básicas em sequência), e 04 (acesso individual aos objetos). Os demais campos são especificados pelo protocolo e possuem valores fixos.

Exemplo: leitura das informações básicas em sequência, a partir do objeto 02h, de um equipamento no endereço 1:

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	2Bh	Função	2Bh
MEI Type	0Eh	MEI Type	0Eh
Código de leitura	01h	Código de leitura	01h
Número do Objeto	02h	Conformity Level	81h
CRC-	70h	More Follows	00h
CRC+	77h	Próximo Objeto	00h
		Número de objetos	01h
		Código do Objeto	02h
		Tamanho do Objeto	05h
		Valor do Objeto	'V1.00'
		CRC-	3Ch
		CRC+	53h

Neste exemplo, o valor dos objetos não foi representado em hexadecimal, mas sim utilizando os caracteres ASCII correspondentes. Por exemplo, para o objeto 02h, o valor 'V1.00' foi transmitido como sendo cinco caracteres ASCII, que em hexadecimal possuem os valores 56h ('V'), 31h ('1'), 2Eh ('.'), 30h ('0') e 30h ('0').

6.10 ERROS DE COMUNICAÇÃO

Erros de comunicação podem ocorrer tanto na transmissão dos telegramas quanto no conteúdo dos telegramas transmitidos. De acordo com o tipo de erro, o escravo poderá ou não enviar resposta para o mestre.

Quando o mestre envia uma mensagem para um escravo configurado em um determinado endereço da rede, este não irá responder ao mestre caso ocorra um dos seguintes eventos:

- Erro no bit de paridade.
- Erro no CRC.
- *Timeout* entre os bytes transmitidos (3,5 vezes o tempo de transmissão de um byte).

Nestes casos, o mestre deverá detectar a ocorrência do erro pelo *timeout* na espera da resposta do escravo. No caso de uma recepção com sucesso, durante o tratamento do telegrama, o escravo pode detectar problemas e enviar uma mensagem de erro, indicando o tipo de problema encontrado:

- Função inválida (código do erro = 1): a função solicitada não está implementada para o equipamento.
- Endereço de dado inválido (código do erro = 2): o endereço do dado (registrador ou bit) não existe.
- Valor de dado inválido (código do erro = 3): ocorre nas seguintes situações:
  - Valor está fora da faixa permitida.
  - Escrita em dado que não pode ser alterado (registrador ou bit somente leitura).

NOTA!
É importante que seja possível identificar no mestre qual o tipo de erro ocorrido para poder diagnosticar problemas durante a comunicação.

No caso da ocorrência de algum destes erros, o escravo deve retornar uma mensagem para o mestre que indica o tipo de erro ocorrido. As mensagens de erro enviadas pelo escravo possuem a seguinte estrutura:

Pergunta (Mestre)	Resposta (Escravo)
Endereço do escravo	Endereço do escravo
Função	Função (com o bit mais significativo em 1)
Dados	Código do erro
CRC-	CRC-
CRC+	CRC+

Exemplo: mestre solicita para o escravo no endereço 1 a escrita no registrador 2900 (supondo registrador 2900 como sendo inexistente):

Pergunta (Mestre)		**Resposta (Escravo)**	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	06h	Função	86h
Registrador (high)	0Bh	Código de erro	02h
Registrador (low)	54h	CRC-	C3h
Valor (high)	00h	CRC+	A1h
Valor (low)	00h		
CRC-	CAh		
CRC+	3Eh		

7 OPERAÇÃO NA REDE MODBUS RTU – MODO MESTRE

Além da operação como escravo, o controlador programável PLC300 também permite a operação como mestre da rede Modbus RTU. Para esta operação, é necessário observar os seguintes pontos:

- Somente a interface RS485 permite operação como mestre da rede.
- É necessário programar, nas configurações do produto, o modo de operação como "Mestre", além da taxa de comunicação, paridade e stop bits, que devem ser as mesmas para todos os equipamentos da rede.
- O mestre da rede Modbus RTU não possui endereço, logo o endereço configurado no PLC300 não é utilizado.
- O envio e recepção de telegramas via interface RS485 utilizando o protocolo Modbus RTU é programado utilizando blocos em linguagem de programação ladder. É necessário conhecer os blocos disponíveis e o software de programação em ladder para poder programar o mestre da rede.
- As seguintes funções estão disponíveis para envio de requisições pelo mestre Modbus:
  Função 01: Read Coils
  Função 02: Read Discrete Inputs
  Função 03: Read Holding Registers
  Função 04: Read Input Registers
  Função 05: Write Single Coil
  Função 06: Write Single Register
  Função 15: Write Multiple Coils
  Função 16: Write Multiple Registers

7.1 BLOCOS PARA A PROGRAMAÇÃO DO MESTRE

Para o controle e monitoração da comunicação Modbus RTU utilizando o controlador programável PLC300, foram desenvolvidos os seguintes blocos, que devem ser utilizados durante a programação em ladder.

7.1.1 MB Read Binary – Leitura de Bits

Bloco para leitura de bits. Permite fazer a leitura de até 128 bits em sequência do escravo destino, utilizando as funções 1 (Read Coils) e 2 (Read Discrete Inputs) do Modbus.

Possui uma entrada de habilitação do bloco "Execute" e uma saída "Done", que é ativada após o término da execução com sucesso da função. Após a transição positiva de "Execute" um novo telegrama é enviado pelo mestre Modbus RTU quando a interface serial RS485 estiver livre. Ao término com sucesso da operação – resposta recebida do escravo – a saída "Done" é ativada, permanecendo ativa enquanto a entrada estiver ativa, e os dados recebidos são copiados para "Value". Em caso de erro na execução da requisição, a saída "Error" é ativada, e o código do erro é colocado em "ErrorID".

Entradas:

<arg0>: "SlaveAddress" – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Endereço do escravo destino – 1 a 247.

<arg1>: "Function#" – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de leitura: 1= "Read Coils"; 2= "Read Discrete Inputs".

<arg2>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do bit inicial – 0 a 65535.

<arg3>: “NumberOfData” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Número de bits lidos em sequência a partir do endereço inicial – 1 a 128.

<arg4>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do escravo, a partir do início do envio pelo mestre – 20 a 5000 ms.

<arg5>: “Offset#” – VAR_IN: inserir uma constante.
Tipos de dados: BOOL
Descrição: Indica se o endereço do dado programado em “InitialDataAddress#” possui offset, ou seja, se o endereço do dado programado no bloco deve ser subtraído de 1 para enviar pela rede Modbus: FALSE= "Sem Offset"; TRUE= "Com Offset de 1".

Saídas:

<arg6>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de leitura enviada para o escravo e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

<arg7>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (interface RS485 ocupada com outra requisição), aguardando liberação para que a solicitação seja enviada pelo bloco. Se a entrada de habilitação for retirada enquanto o bloco faz esta indicação, a requisição é descartada.
Nota: A variável tem que ter permissão de escrita.

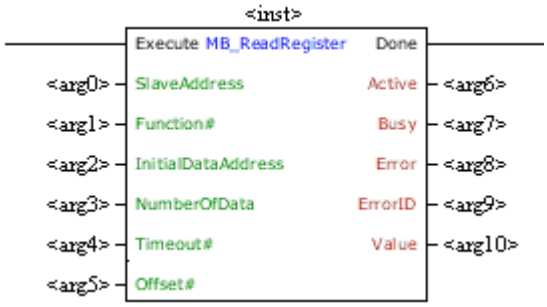
<arg8>: “Error” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Erro na execução da requisição.
Nota: A variável tem que ter permissão de escrita.

<arg9>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Em caso de erro na requisição, indica o tipo de erro ocorrido. Resultados possíveis: 0= "Executado com sucesso"; 1= "Algum dado de entrada inválido"; 2= "Mestre não habilitado"; 4= "Timeout na resposta do escravo"; 5= "Escravo retornou erro".
Nota: A variável tem que ter permissão de escrita.

<arg10>: “Value” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL[1 … 128]
Descrição: Variável ou array onde serão salvos os dados lidos do escravo.
Nota: A variável tem que ter permissão de escrita.

7.1.2 MB Read Register – Leitura de Registradores

Bloco para leitura de registradores de 16 bits. Permite fazer a leitura de até 16 registradores em sequência do escravo destino, utilizando as funções 3 (Read Holding Registers) e 4 (Read Input Registers) do Modbus.



Possui uma entrada de habilitação do bloco “Execute” e uma saída “Done”, que é ativada após o término da execução com sucesso da função. Após a transição positiva de “Execute” um novo telegrama é enviado pelo mestre Modbus RTU quando a interface serial RS485 estiver livre. Ao término com sucesso da operação – resposta recebida do escravo – a saída “Done” é ativada, permanecendo ativa enquanto a entrada estiver ativa, e os dados recebidos são copiados para “Value”. Em caso de erro na execução da requisição, a saída “Error” é ativada, e o código do erro é colocado em “ErrorID”.

Entradas:

<arg0>: “SlaveAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Endereço do escravo destino – 1 a 247.

<arg1>: “Function#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de leitura: 3= "Read Holding Registers"; 4= "Read Input Registers".

<arg2>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do registrador inicial – 0 a 65535.

<arg3>: “NumberOfData” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Número de registradores lidos a partir do endereço inicial – 1 a 16.

<arg4>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do escravo, a partir do início do envio pelo mestre – 20 a 5000 ms.

<arg5>: “Offset#” – VAR_IN: inserir uma constante.
Tipos de dados: BOOL
Descrição: Indica se o endereço do dado programado em “InitialDataAddress#” possui offset, ou seja, se o endereço do dado programado no bloco deve ser subtraído de 1 para enviar pela rede Modbus: FALSE= "Sem Offset"; TRUE= "Com Offset de 1".

Saídas:

<arg6>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de leitura enviada para o escravo e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

<arg7>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (interface RS485 ocupada com outra requisição), aguardando liberação para que a solicitação seja enviada pelo bloco. Se a entrada de habilitação for retirada enquanto o bloco faz esta indicação, a requisição é descartada.
Nota: A variável tem que ter permissão de escrita.

<arg8>: “Error” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Erro na execução da requisição.
Nota: A variável tem que ter permissão de escrita.

<arg9>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Em caso de erro na requisição, indica o tipo de erro ocorrido. Resultados possíveis: 0= "Executado com sucesso"; 1= "Algum dado de entrada inválido"; 2= "Mestre não habilitado"; 4= "Timeout na resposta do escravo"; 5= "Escravo retornou erro".
Nota: A variável tem que ter permissão de escrita.

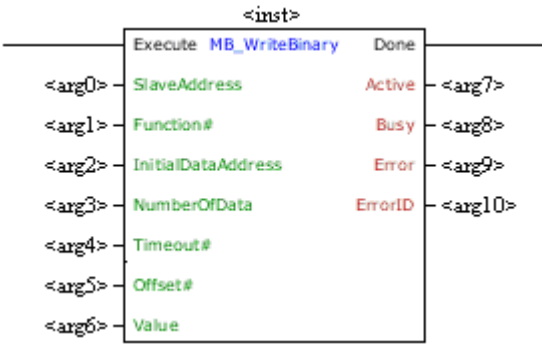
<arg10>: “Value” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE[1 ... 32], SINT[1 ... 32], USINT[1 ... 32], WORD[1 ... 16], UINT[1 ... 16], INT[1 ... 16], DWORD[1 ... 8], UDINT[1 ... 8], DINT[1 ... 8] ou REAL[1 ... 8]
Descrição: Variável ou array onde serão salvos os dados lidos do escravo.
Nota: A variável tem que ter permissão de escrita.

NOTA!

- O protocolo Modbus RTU, utilizando as funções 3 e 4, permite a leitura de registradores de 16 bits apenas. Para leitura de dados com mais de 16 bits (um REAL, por exemplo), é possível fazer a leitura de múltiplos registradores, e salvar o valor em uma variável com tamanho maior que 16 bits.
- É importante que a quantidade de registradores lidos seja compatível com o tamanho da variável ou do array onde os dados serão salvos.

7.1.3 MB Write Binary – Escrita de Bits

Bloco para escrita de bits. Permite fazer a escrita de até 128 bits utilizando as funções 5 (Write Single Coil) e 15 (Write Multiple Coils) do Modbus.



Possui uma entrada de habilitação do bloco “Execute” e uma saída “Done”, que é ativada após o término da execução com sucesso da função. Após a transição positiva de “Execute” um novo telegrama é enviado pelo mestre Modbus RTU quando a interface serial RS485 estiver livre. Ao término com sucesso da operação – resposta recebida do escravo – a saída “Done” é ativada, permanecendo ativa enquanto a entrada estiver ativa. Em caso de erro na execução da requisição, a saída “Error” é ativada, e o código do erro é colocado em “ErrorID”.

Entradas:

<arg0>: “SlaveAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Endereço do escravo destino – 1 a 247.

<arg1>: “Function#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de escrita: 5= "Write Single Coil"; 15= "Write Multiple Coils".

<arg2>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do bit inicial – 0 a 65535.

<arg3>: “NumberOfData” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Número de bits escritos em sequência a partir do endereço inicial – 1 a 128.

<arg4>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do escravo, a partir do início do envio pelo mestre – 20 a 5000 ms.

<arg5>: “Offset#” – VAR_IN: inserir uma constante.
Tipos de dados: BOOL
Descrição: Indica se o endereço do dado programado em “InitialDataAddress#” possui offset, ou seja, se o endereço do dado programado no bloco deve ser subtraído de 1 para enviar pela rede Modbus: FALSE= "Sem Offset"; TRUE= "Com Offset de 1".

<arg6>: “Value” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BOOL[1 ... 128]
Descrição: Variável ou array com os dados que serão escritos no escravo.

Saídas:

<arg7>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de escrita enviada para o escravo e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

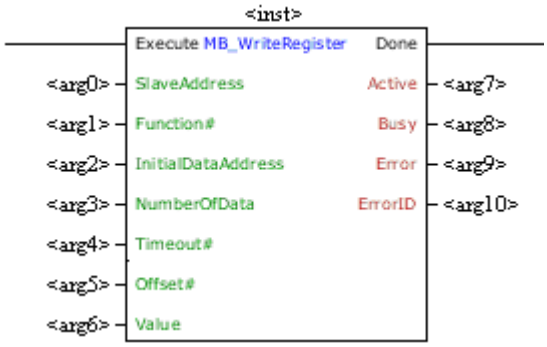
<arg8>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (interface RS485 ocupada com outra requisição), aguardando liberação para que a solicitação seja enviada pelo bloco. Se a entrada de habilitação for retirada enquanto o bloco faz esta indicação, a requisição é descartada.
Nota: A variável tem que ter permissão de escrita.

<arg9>: “Error” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Erro na execução da requisição.
Nota: A variável tem que ter permissão de escrita.

<arg10>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Em caso de erro na requisição, indica o tipo de erro ocorrido. Resultados possíveis: 0= "Executado com sucesso"; 1= "Algum dado de entrada inválido"; 2= "Mestre não habilitado"; 4= "Timeout na resposta do escravo"; 5= "Escravo retornou erro".
Nota: A variável tem que ter permissão de escrita.

7.1.4 MB Write Register – Escrita de Registradores

Bloco para escrita de registradores. Permite fazer a escrita de até 16 registradores utilizando a função 6 (Write Holding Register) ou 16 (Write Multiple Registers) do Modbus.



Possui uma entrada de habilitação do bloco “Execute” e uma saída “Done”, que é ativada após o término da execução com sucesso da função. Após a transição positiva de “Execute” um novo telegrama é enviado pelo mestre Modbus RTU quando a interface serial RS485 estiver livre. Ao término com sucesso da operação – resposta recebida do escravo – a saída “Done” é ativada, permanecendo ativa enquanto a entrada estiver ativa. Em caso de erro na execução da requisição, a saída “Error” é ativada, e o código do erro é colocado em “ErrorID”.

Entradas:

<arg0>: “SlaveAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Endereço do escravo destino – 1 a 247.

<arg1>: “Function#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de escrita: 6= "Write Single Register"; 16= "Write Multiple Registers".

<arg2>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do registrador inicial – 0 a 65535.

<arg3>: “NumberOfData” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Número de registradores escritos a partir do endereço inicial – 1 a 16.

<arg4>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do escravo, a partir do início do envio pelo mestre – 20 a 5000 ms.

<arg5>: “Offset#” – VAR_IN: inserir uma constante.
Tipos de dados: BOOL
Descrição: Indica se o endereço do dado programado em “InitialDataAddress#” possui offset, ou seja, se o endereço do dado programado no bloco deve ser subtraído de 1 para enviar pela rede Modbus: FALSE= "Sem Offset"; TRUE= "Com Offset de 1".

<arg6>: “Value” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE[1 ... 32], USINT[1 ... 32], SINT[1 ... 32], WORD[1 ... 16], UINT[1 ... 16], INT[1 ... 16], DWORD[1 ... 8], UDINT[1 ... 8], DINT[1 ... 8] ou REAL[1 ... 8]
Descrição: Variável ou array com os dados que serão escritos no escravo.

Saídas:

<arg7>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de escrita enviada para o escravo e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

<arg8>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (interface RS485 ocupada com outra requisição), aguardando liberação para que a solicitação seja enviada pelo bloco. Se a entrada de habilitação for retirada enquanto o bloco faz esta indicação, a requisição é descartada.
Nota: A variável tem que ter permissão de escrita.

<arg9>: “Error” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Erro na execução da requisição.
Nota: A variável tem que ter permissão de escrita.

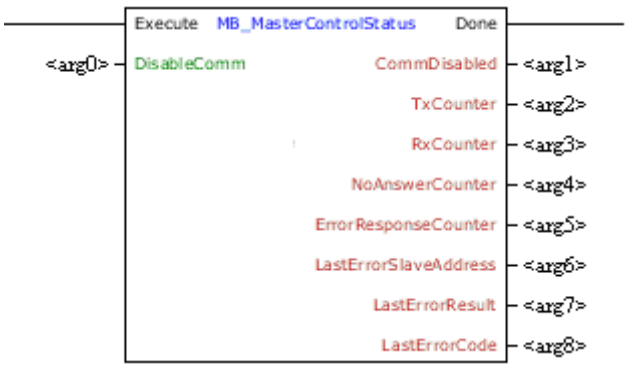
<arg10>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Em caso de erro na requisição, indica o tipo de erro ocorrido. Resultados possíveis: 0= "Executado com sucesso"; 1= "Algum dado de entrada inválido"; 2= "Mestre não habilitado"; 4= "Timeout na resposta do escravo"; 5= "Escravo retornou erro".
Nota: A variável tem que ter permissão de escrita.

NOTA!

- O protocolo Modbus RTU, utilizando a função 16, permite a escrita de registradores de 16 bits apenas. Para escrita de dados com mais de 16 bits (um REAL, por exemplo), é possível fazer a escrita de múltiplos registradores, e utilizar como fonte dos dados uma variável com tamanho maior que 16 bits.
- É importante que a quantidade de registradores escritos seja compatível com o tamanho da variável ou do array de onde os dados serão utilizados.

7.1.5 MB Master Control/Status – Controle e Estado do Modbus RTU

Bloco para controle e monitoração do mestre da rede Modbus RTU. Sempre que uma rede Modbus RTU for montada com o PLC300 como mestre da rede, recomenda-se utilizar este bloco para obter informações sobre o estado da comunicação.



Possui uma entrada de habilitação do bloco “Execute” e uma saída “Done” que é ativada após o término da execução da função. Enquanto a entrada de habilitação “Execute” estiver ativa, os dados de entrada são utilizados e os dados de saída são atualizados. Caso a entrada seja zerada, os valores de entrada são desconsiderados e os argumentos de saída são zerados. A saída “Done” reflete o valor da entrada.

Entradas:

<arg0>: “DisableComm” – VAR_IN: inserir uma constante ou uma variável (tag).
Tipos de dados: BOOL
Descrição: Permite desabilitar o mestre Modbus. Ao desabilitar o mestre, os contadores e marcadores de status do mestre Modbus RTU também são zerados: 0= "Mestre em execução"; 1= "Desabilita mestre".

Saídas:

<arg1>: “CommDisabled” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Indica se o mestre está ou não desabilitado. Pode ocorrer por solicitação do usuário ou caso a interface esteja programada para operar como escravo da rede: 0= "Mestre habilitado"; 1= "Mestre desabilitado".
Nota: A variável tem que ter permissão de escrita.

<arg2>: “TxCounter” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: WORD ou UINT
Descrição: Contador de requisições enviadas pelo mestre da rede para os escravos. É zerado sempre que o equipamento for desligado ou o mestre for desabilitado – 0 a 65535.
Nota: A variável tem que ter permissão de escrita.

<arg3>: “RxCounter” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: WORD ou UINT
Descrição: Contador de telegramas recebidos pelo mestre da rede. É zerado sempre que o equipamento for desligado ou o mestre for desabilitado – 0 a 65535.
Nota: A variável tem que ter permissão de escrita.

<arg4>: “NoAnswerCounter” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: WORD ou UINT
Descrição: Contador de requisições do mestre que não foram respondidas pelos escravos. É zerado sempre que o equipamento for desligado ou o mestre for desabilitado – 0 a 65535.
Nota: A variável tem que ter permissão de escrita.

<arg5>: “ErrorResponseCounter” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: WORD ou UINT
Descrição: Contador de requisições do mestre e que os escravos responderam com alguma resposta de erro. O código do erro pode ser obtido no marcador que indica o código do último erro detectado. É zerado sempre que o equipamento for desligado ou o mestre for desabilitado – 0 a 65535.
Nota: A variável tem que ter permissão de escrita.

<arg6>: “LastErrorSlaveAddress” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Indica o endereço do escravo no qual foi detectado o último erro de comunicação. É zerado sempre que o equipamento for desligado ou o mestre for desabilitado – 0 a 247.
Nota: A variável tem que ter permissão de escrita.

<arg7>: “LastErrorResult” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Indica o resultado da operação – timeout ou resposta de erro, conforme ERROR ID do bloco – para o escravo no qual foi detectado o último erro de comunicação. É zerado sempre que o equipamento for desligado ou o mestre for desabilitado: 0= "Sem erro detectado"; 4= "Timeout na resposta do escravo"; 5= "Escravo retornou erro".
Nota: A variável tem que ter permissão de escrita.

<arg8>: “LastErrorCode” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Indica o código do erro, no caso do mestre receber resposta de erro de algum escravo. É zerado sempre que o equipamento for desligado ou o mestre for desabilitado – 0 a 255.
Nota: A variável tem que ter permissão de escrita.

NOTA!
Os dados acessados utilizando este bloco também estão disponíveis através de marcadores de sistema de leitura e escrita, conforme descrito no item 8.

7.1.6 MB Slave Status – Estado dos Escravos da Rede Modbus RTU

Bloco para monitoração dos escravos da rede Modbus RTU. Deve ser utilizado caso seja desejado identificar problemas na comunicação do mestre com algum escravo da rede Modbus RTU.

Possui uma entrada de habilitação do bloco “Execute” e uma saída “Done” que é ativada após o término da execução da função. Enquanto a entrada de habilitação “Execute” estiver ativa os dados de entrada são utilizados e os dados de saída são atualizados a cada execução do bloco. A saída “Done” reflete o valor da entrada.

Entradas:

<arg0>: “ErrorsToSetOffline#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Permite programar, para este bloco, a quantidade de erros de comunicação que o mestre deve identificar até que a comunicação com um escravo da rede seja considerada offline. É considerado erro de comunicação toda requisição (leitura ou escrita) que o mestre enviou para um escravo e não recebeu resposta ou a resposta recebida possuía erro de CRC.

<arg1>: “AddressSlave1#” – VAR_IN: inserir uma constante.
<arg2>: “AddressSlave2#” – VAR_IN: inserir uma constante.
<arg3>: “AddressSlave3#” – VAR_IN: inserir uma constante.
<arg4>: “AddressSlave4#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Permite programar o endereço de até 4 escravos, cuja quantidade de erros de comunicação serão monitorados para saber se estão online ou offline. Caso a quantidade de erros de comunicação em sequência, detectados nos blocos de leitura e escrita via Modbus, atinja o valor programado em “ErrorsToSetOffline”, a saída respectiva é acionada. Caso deseje-se monitorar um número menor de escravos, pode-se deixar qualquer das entradas em zero: 0= "Ignora entrada"; 1 a 247.

Saídas:

<arg5>: “GeneralOffline#” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Se qualquer uma das saídas dos escravos indicados for acionada, esta saída também será acionada. Funciona como uma lógica OU entre as 4 saídas de indicação dos escravos.
Nota: A variável tem que ter permissão de escrita.

<arg6>: “Slave1Offline#” – VAR_OUT: inserir uma variável (tag).
<arg7>: “Slave2Offline#” – VAR_OUT: inserir uma variável (tag).
<arg8>: “Slave3Offline#” – VAR_OUT: inserir uma variável (tag).
<arg9>: “Slave4Offline#” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Saída acionada caso a quantidade de erros de comunicação em sequência para os escravos indicados nas respectivas entradas atinja o valor programado em “ErrorsToSetOffline”.
Nota: A variável tem que ter permissão de escrita.

8 MARCADORES DE SISTEMA PARA RS232 E RS485

Para interfaces seriais RS232 e RS485, foram disponibilizados os seguintes marcadores de sistema de leitura (%S) e marcadores de sistema de escrita (%C), para controle e monitoração:

8.1 MARCADORES DE SISTEMA DE LEITURA

Estado do Mestre Modbus (RS485): conjunto de marcadores de leitura que indicam o estado do mestre Modbus, além de informações para diagnóstico da rede.	
Marcador	Descrição
%SB3100	Estado do mestre Modbus: 0 = Operação normal. 1 = Mestre desabilitado.
%SB3101	Reservado.
%SW3102	Contador de requisições feitas pelo mestre. Contador incrementado toda vez que um novo telegrama é enviado pelo mestre da rede Modbus RTU. É zerado sempre que atingir o limite máximo.
%SW3104	Contador de respostas recebidas com sucesso. Contador incrementado toda vez que o mestre receber uma resposta com sucesso de um escravo da rede. É zerado sempre que atingir o limite máximo.
%SW3106	Contador de requisições sem resposta – timeout. Contador incrementado toda vez que ocorrer timeout para uma requisição feita pelo mestre da rede Modbus RTU para um escravo. É zerado sempre que atingir o limite máximo ou a interface for desabilitada.
%SW3108	Contador de respostas com erro recebidas. Contador incrementado toda vez que o escravo retornar uma resposta de erro para uma requisição feita pelo mestre Modbus RTU. É zerado sempre que atingir o limite máximo ou a interface for desabilitada. Sempre que este erro for detectado, os dados para o endereço do escravo, tipo de erro e código do erro serão salvos nos marcadores %SB3110 até %SB3112.
%SB3110	Último erro ocorrido: endereço do escravo.
%SB3111	Último erro ocorrido: tipo de erro. 0 = Sem erro. 4 = Timeout na resposta. 5 = Escravo retornou resposta de erro. É zerado sempre que a interface for desabilitada
%SB3112	Último erro ocorrido: código do erro recebido, caso o tipo seja resposta de erro. É zerado sempre que a interface for desabilitada
%SB3113	Reservado.

Estado do escravo Modbus (RS485): conjunto de marcadores de leitura que indicam a quantidade de telegramas enviados e recebidos pelo escravo Modbus RTU.	
Marcador	Descrição
%SW3120	Número de telegramas recebidos. Especifico para o modo escravo.
%SW3122	Número de telegramas transmitidos. Especifico para o modo escravo.

8.2 MARCADORES DE SISTEMA DE ESCRITA

Configuração da Interface RS232: conjunto de marcadores de escrita para programar as configurações da interface RS232. Também são acessíveis através do menu Setup.	
Marcador	Descrição
%CB3060	Reservado.
%CB3061	Reservado.
%CB3062	Formato dos bytes: 0 = sem paridade, 1 stop bit 1 = paridade ímpar, 1 stop bit 2 = paridade par, 1 stop bit 3 = reservado 4 = sem paridade, 2 stop bits 5 = paridade ímpar, 2 stop bits 6 = paridade par, 2 stop bits
%CB3063	Taxa de comunicação para RS232: 0 = 1200 bit/s 1 = 2400 bit/s 2 = 4800 bit/s 3 = 9600 bit/s 4 = 19200 bit/s 5 = 38400 bit/s 6 = 57600 bit/s

Configuração da Interface RS485: conjunto de marcadores de escrita para programar as configurações da interface RS485. Também são acessíveis através do menu Setup.	
Marcador	Descrição
%CB3068	Endereço serial (modo escravo) 1 ... 247.
%CB3069	Modo de operação: 0 = Escravo Modbus RTU. 1 = Mestre Modbus RTU.
%CB3070	Formato dos bytes: 0 = sem paridade, 1 stop bit 1 = paridade ímpar, 1 stop bit 2 = paridade par, 1 stop bit 3 = reservado 4 = sem paridade, 2 stop bits 5 = paridade ímpar, 2 stop bits 6 = paridade par, 2 stop bits
%CB3071	Taxa de comunicação para RS485: 0 = 1200 bit/s 1 = 2400 bit/s 2 = 4800 bit/s 3 = 9600 bit/s 4 = 19200 bit/s 5 = 38400 bit/s 6 = 57600 bit/s

Controle do Mestre Modbus (RS485): conjunto de marcadores de escrita para controle do mestre Modbus.	
Marcador	Descrição
%CW3100	Controle do mestre Modbus: 0 = Operação normal. 1 = Desabilita interface.

I. APÊNDICES

APÊNDICE A. TABELA ASCII

Tabela I.1: *Caracteres ASCII*

Dec	Hex	Chr		Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr
0	00	NUL	(Null char.)	32	20	Sp	64	40	@	96	60	`
1	01	SOH	(Start of Header)	33	21	!	65	41	A	97	61	a
2	02	STX	(Start of Text)	34	22	"	66	42	B	98	62	b
3	03	ETX	(End of Text)	35	23	#	67	43	C	99	63	c
4	04	EOT	(End of Transmission)	36	24	$	68	44	D	100	64	d
5	05	ENQ	(Enquiry)	37	25	%	69	45	E	101	65	e
6	06	ACK	(Acknowledgment)	38	26	&	70	46	F	102	66	f
7	07	BEL	(Bell)	39	27	'	71	47	G	103	67	g
8	08	BS	(Backspace)	40	28	(	72	48	H	104	68	h
9	09	HT	(Horizontal Tab)	41	29	)	73	49	I	105	69	i
10	0A	LF	(Line Feed)	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	(Vertical Tab)	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	(Form Feed)	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	(Carriage Return)	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	(Shift Out)	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	(Shift In)	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	(Data Link Escape)	48	30	0	80	50	P	112	70	p
17	11	DC1	(Device Control 1)	49	31	1	81	51	Q	113	71	q
18	12	DC2	(Device Control 2)	50	32	2	82	52	R	114	72	r
19	13	DC3	(Device Control 3)	51	33	3	83	53	S	115	73	s
20	14	DC4	(Device Control 4)	52	34	4	84	54	T	116	74	t
21	15	NAK	(Negative Acknowledgement)	53	35	5	85	55	U	117	75	u
22	16	SYN	(Synchronous Idle)	54	36	6	86	56	V	118	76	v
23	17	ETB	(End of Trans. Block)	55	37	7	87	57	W	119	77	w
24	18	CAN	(Cancel)	56	38	8	88	58	X	120	78	x
25	19	EM	(End of Medium)	57	39	9	89	59	Y	121	79	y
26	1A	SUB	(Substitute)	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	(Escape)	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	(File Separator)	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	(Group Separator)	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	(Record Separator)	62	3E	>	94	5E	^	126	7E	~
31	1F	US	(Unit Separator)	63	3F	?	95	5F	_	127	7F	DEL

APÊNDICE B. CÁLCULO DO CRC UTILIZANDO TABELAS

A seguir é apresentada uma função, utilizando linguagem de programação "C", que implementa o cálculo do CRC para o protocolo Modbus RTU. O cálculo utiliza duas tabelas para fornecer valores pré-calculados dos deslocamentos necessários para a realização do cálculo.

```
/* 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, 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,
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 };


/* 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, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40 };


/* The function returns the CRC as a unsigned short type */
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 byte of CRC initialized     */
    unsigned char uchCRCLo = 0xFF;      /* low byte of CRC initialized      */
    unsigned uIndex;                    /* will index into CRC lookup table */
    while (usDataLen--)                 /* pass through message buffer      */
    {
        uIndex = uchCRCLo ^ *puchMsgg++; /* calculate the CRC               */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex};
        uchCRCHi = auchCRCLo[uIndex];
    }

    return (uchCRCHi << 8 | uchCRCLo);
}
```

WEG Equipamentos Elétricos S.A.
Jaraguá do Sul – SC – Brasil
Fone 55 (47) 3276-4000 – Fax 55 (47) 3276-4020
São Paulo – SP – Brasil
Fone 55 (11) 5053-2300 – Fax 55 (11) 5052-4212
automacao@weg.net
www.weg.net