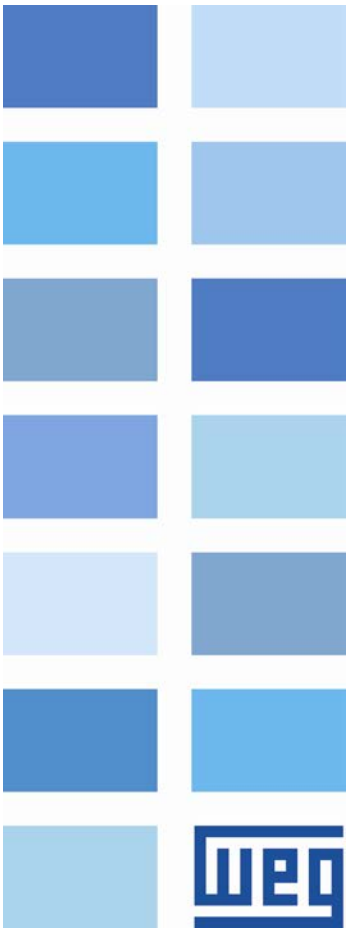
Modbus TCP

PLC300

Manual do Usuário



Manual do Usuário Modbus TCP

Série: PLC300

Idioma: Português

N º do Documento: 10001276962 / 01

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 TCP. Este manual deve ser utilizado em conjunto com manual do usuário do PLC300.

ABREVIAÇÕES E DEFINIÇÕES

ASCII	American Standard Code for Information Interchange
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
IP	Internet Protocol
MAC	Medium Access Control
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

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 TCP foi desenvolvido baseado nas seguintes especificações e documentos:

Documento	Versão	Fonte
MODBUS Application Protocol Specification, December 28th 2006.	V1.1b	MODBUS.ORG
MODBUS Messaging On TCP/IP Implementation Guide, October 24th 2006.	V1.0b	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 DESCRIÇÃO DA INTERFACE

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

1.1 CARACTERÍSTICAS DA INTERFACE ETHERNET

- Interface segue o padrão T-568A / T-568B.
- Pode operar como cliente ou servidor na rede Modbus TCP.
- Possibilita comunicação utilizando taxas de 10 ou 100 Mbps, em modo *half* ou *full duplex*.

1.2 INDICAÇÕES

Além dos marcadores de sistema, que fornecem diversas informações sobre a interface Ethernet, o controlador programável PLC300 possui LEDs no conector RJ45, utilizados para indicação dos estados da interface Ethernet.

Figura 1.1: *LEDs de indicação do estado da interface Ethernet*

O LED SPD(1) indica a taxa de comunicação atual da interface.

Tabela 1.1: *Estados do LED SPD*

Estado	Descrição	Comentário
Apagado	10 Mbps	Utilizando taxa de 10 Mbps
Verde	100 Mbps	Utilizando taxa de 100 Mbps

O LED LINK(2) indica o estado da conexão física da rede, bem como a atividade no barramento.

Tabela 1.2: *Estados do LED LINK*

Estado	Descrição	Comentário
Apagado	Sem Link	Sem conexão, sem atividade
Amarelo	Link	Estabelecido link ethernet, mas sem comunicação
Intermitente amarelo	Atividade no barramento	Indica efetivamente que há comunicação no barramento

1.3 INSTALAÇÃO DA REDE ETHERNET

Para a ligação do controlador programável utilizando a interface Ethernet, os seguintes pontos devem ser observados:

1.3.1 Taxa de Comunicação

A interface Ethernet do controlador programável pode comunicar utilizando as taxas de 10 ou 100 Mbps, em modo *half* ou *full duplex*.

A taxa de comunicação pode ser programada através do menu Setup ou navegador WEB.

1.3.2 MAC Address

Cada controlador programável PLC300 possui um MAC Address único, que é indicado no display (Status de IO).

1.3.3 Endereço IP

Todo equipamento em uma rede Ethernet necessita de um endereço IP e de uma máscara de sub-rede.

O endereçamento IP é único na rede, e cada equipamento deve possuir um endereço IP diferente. A máscara da sub-rede serve para definir quais as faixas de endereço IP que são válidas na rede.

Estes atributos podem ser configurados automaticamente através de um servidor DHCP presente na rede, desde que esta opção esteja habilitada no PLC300.

As configurações do endereço IP, máscara de sub-rede, gateway e DHCP podem ser programados através do menu Setup ou navegador WEB.

1.3.4 Cabos

Para realizar a instalação, recomenda-se a utilização de cabos Ethernet blindados específicos para a utilização em ambiente industrial.

A carcaça metálica do conector Ethernet do PLC300 é aterrada, e realiza o aterramento do cabo caso o conector do cabo possua invólucro metálico.

Normalmente utiliza-se um cabo direto para ligação do PLC300 a um concentrador (*switch*), ou um cabo cruzado (*cross-over*) para ligação direta entre o PLC300 e o PC/CLP. Apesar disto, a interface Ethernet do PLC300 funciona com Auto-MDIX (automatic medium-dependent interface crossover), uma tecnologia que detecta automaticamente o tipo de cabo utilizado e configura a conexão de acordo, tornando desnecessária a utilização dos cabos cruzados.

1.3.5 Recomendações de instalação

- Recomenda-se utilizar equipamentos (cabos, switches) preparados para o ambiente industrial.
- Cada segmento de cabo deve ter no máximo 90m.
- 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 topologia mais comum é em estrela, exatamente como é feito com redes de computadores. Neste caso, todos os equipamentos devem ser conectados a um concentrador (switch).

Figura 1.2: *Topologia estrela*

1.4 ACESSO UTILIZANDO NAVEGADOR WEB

É possível utilizar um navegador WEB para acessar as configurações e estados do controlador programável PLC300. Digitando o endereço IP na barra de endereços do navegador, será apresentada uma página WEB com informações do equipamento.

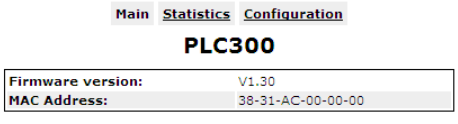


Figura 1.3: *Página WEB inicial*

1.5 CLIENTE SNTP

O controlador programável PLC300 possui um cliente SNTP incorporado. Este cliente utiliza o protocolo SNTP para requisitar de um servidor informações de data e hora, e altera automaticamente estas configurações no PLC300.

Utilizando a interface WEB, é possível configurar o cliente SNTP. Devem ser informados os endereços IP do servidor principal e do servidor redundante aos quais o PLC300 deve se conectar para buscar as informações de data e hora. O servidor redundante será utilizado, quando o servidor principal não estiver acessível na rede. É possível configurar o intervalo de tempo entre as atualizações da data e hora, e o tempo máximo de espera pela resposta do servidor de tempo.

2 PROTOCOLO MODBUS

O protocolo Modbus foi inicialmente desenvolvido em 1979 pela Modicon. Atualmente, é um protocolo aberto amplamente difundido, utilizado por vários fabricantes em diversos equipamentos. É um protocolo da camada de aplicação para comunicação entre dispositivos, principalmente utilizado em sistemas de automação industrial.

2.1 ESTRUTURA DAS MENSAGENS

Modbus é um protocolo baseado em transações, que consistem em uma requisição seguida de uma resposta. Toda comunicação inicia com o cliente (mestre) fazendo uma solicitação a um servidor (escravo), e este responde o que foi solicitado.

A comunicação é baseada em um pacote, denominado PDU (Protocol Data Unit) que é definido pela especificação do protocolo em três tipos:
- PDU de requisição:
  - *Function Code*: código que especifica o tipo de serviço ou função solicitada (1 byte)
  - *Function Data*: dados específicos da função (número de bytes variável)
- PDU de resposta:
  - *Function Code*: código da função correspondente à requisição (1 byte)
  - *Response Data*: dados específicos da função (número de bytes variável)
- PDU de resposta com exceção:
  - *Error Code*: código da função correspondente à requisição com o bit mais significativo em 1 (1 byte)
  - *Exception Code*: código especificando a exceção (1 byte)

Uma transação pode ser visualizada na Figura 2.1.

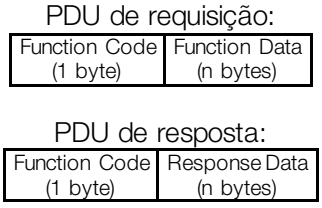


Figura 2.1: *Transação Modbus*

O campo de código da função especifica o tipo de serviço ou função solicitada ao servidor (leitura, escrita, etc.). Para a lista de funções disponíveis para acesso aos dados, e a descrição do campo de dados para cada função, consulte o item 4.

De acordo com o protocolo, cada função é utilizada para acessar um tipo específico de dados. A Tabela 2.1 contém os tipos básicos definidos na especificação.

Tabela 2.1: *Tipos de dados Modbus*

Nome	Tamanho	Acesso
Discrete Input	1 bit	Somente leitura
Discrete Output (Coils)	1 bit	Leitura e escrita
Input Registers	16 bits	Somente leitura
Holding Registers (Registers)	16 bits	Leitura e escrita

Cada implementação do protocolo Modbus pode acrescentar ao PDU dados específicos para o correto processamento das mensagens através da interface utilizada.

2.2 IMPLEMENTAÇÃO MODBUS TCP

Modbus TCP é uma implementação do protocolo Modbus baseado em TCP/IP. Utiliza a pilha TCP/IP para comunicação e adiciona ao PDU Modbus um cabeçalho específico denominado MBAP Header. A associação do cabeçalho ao PDU recebe o nome de ADU (Application Data Unit).

O cabeçalho tem tamanho de 7 bytes, e é composto pelos seguintes campos:
- *Transaction identifier:* usado para identificação da resposta para a transação (2 bytes).

- *Protocol identifier:* 0 (zero) indica Modbus (2 bytes).
- *Length:* contagem de todos os próximos bytes (2 bytes).
- *Unit identifier:* utilizado para identificar o escravo remoto em uma rede Modbus RTU (1 byte).

Figura 2.2: *Modbus TCP ADU*

Modbus TCP não acrescenta ao PDU um campo de checagem de erros, entretanto o frame ethernet já utiliza CRC-32 tornando desnecessário outro campo de checagem.

O cliente Modbus TCP deve iniciar uma conexão TCP com o servidor a fim de enviar as requisições. A porta TCP 502 é a porta padrão para conexão com servidores Modbus TCP.

3 CONFIGURAÇÃO DA INTERFACE

Para realizar a configuração da interface Ethernet, através do Setup do controlador programável PLC300 são disponibilizados os seguintes menus:

3.1 CONFIGURAÇÃO ETHERNET

ENDEREÇO IP

Faixa de Valores:	0.0.0.0 a 255.255.255.255	**Padrão:** 192.168.0.10

Descrição:
Permite programar o valor desejado para o endereço IP utilizado pelo PLC300. O endereçamento IP deve ser único na rede, e cada equipamento deve possuir um endereço IP diferente.

NOTA!
Este atributo pode ser configurado automaticamente através de um servidor DHCP presente na rede, desde que esta opção esteja habilitada.

MÁSCARA DE SUB-REDE

Faixa de Valores:	0.0.0.0 a 255.255.255.255	**Padrão:** 255.255.255.0

Descrição:
Permite programar o valor desejado para a máscara de sub-rede utilizada pelo PLC300. A máscara da sub-rede serve para definir quais as faixas de endereço IP que são válidas na rede.

NOTA!
Este atributo pode ser configurado automaticamente através de um servidor DHCP presente na rede, desde que esta opção esteja habilitada.

GATEWAY PADRÃO

Faixa de Valores:	0.0.0.0 a 255.255.255.255	**Padrão:** 0.0.0.0

Descrição:
Permite programar o valor desejado para o gateway padrão utilizado pelo PLC300. O gateway padrão fornece uma rota para a comunicação com redes remotas.

NOTA!
Este atributo pode ser configurado automaticamente através de um servidor DHCP presente na rede, desde que esta opção esteja habilitada.

DHCP

Faixa de Valores:	Desabilitado Habilitado	**Padrão:** Desabilitado

Descrição:
Permite habilitar ou desabilitar a configuração via servidor DHCP. O servidor DHCP pode atribuir automaticamente endereços IP, máscara de sub-rede, etc. aos equipamentos na rede.

Caso o DHCP seja habilitado, as configurações feitas para endereço IP, máscara de sub-rede e gateway serão desconsideradas. O endereço IP atribuído pelo servidor DHCP pode ser visualizado na tela de Status de I/Os.

TAXA DE COMUNICAÇÃO

Faixa de Valores:	Auto 10MBps Full Duplex 10MBps Half Duplex 100MBps Full Duplex 100MBps Half Duplex	**Padrão:** Auto

Descrição:
A interface Ethernet pode comunicar utilizando as taxas de 10 ou 100 Mbps, em modo *half* ou *full duplex*. Quando selecionada a opção Auto, a detecção automática da taxa de comunicação e do modo de comunicação é realizada.

3.2 CONFIGURAÇÃO MODBUS TCP

PORTA TCP

Faixa de Valores:	0 a 65535	**Padrão:** 502

Descrição:
Configura a porta TCP utilizada para a comunicação com o servidor Modbus TCP.
A porta 502 permanece aberta, mesmo que outra porta esteja configurada. Neste caso é possível conectar ao servidor por qualquer destas portas.

UNIT ID

Faixa de Valores:	1 a 255	**Padrão:** 255

Descrição:
Identificador do protocolo Modbus TCP para este equipamento.
Telegramas recebidos com identificador diferente do configurado, são descartados.
Se o valor configuração for 255, o servidor atuará como um Gateway Modbus TCP/RTU e encaminhará mensagens recebidas que possuam Unit ID entre 1 e 247 para a interface serial RS485, se esta estiver configurada como mestre Modbus RTU.
Desta forma, é possível estabelecer comunicação entre o cliente conectado na interface Ethernet do PLC300 com um equipamento conectado na interface RS485.

AUTENTICAÇÃO DE IP

Faixa de Valores:	0.0.0.0 a 255.255.255.255	**Padrão:** 0.0.0.0

Descrição:
Quando programado para valor diferente de 0.0.0.0, somente o equipamento/PC com este endereço IP poderá estabelecer comunicação com o servidor. Requisições de conexão de outros endereços, serão negadas.

GATEWAY TIMEOUT

Faixa de Valores:	20 a 5000 ms	**Padrão:** 1000 ms

Descrição:
Indica o tempo de timeout da resposta que deve ser utilizado quando uma requisição for encaminhada pelo gateway para um escravo Modbus RTU. Se este tempo se esgotar, e a resposta não for recebida, o gateway retornará um telegrama indicando erro ao cliente Modbus TCP que originou a requisição.

4 OPERAÇÃO NA REDE MODBUS TCP – MODO SERVIDOR

Como servidor da rede Modbus TCP, o controlador programável PLC300 possui as seguintes características:

- Pode operar simultaneamente como cliente e servidor Modbus TCP.
- Porta TCP, Unit ID, etc. definidos através do setup do equipamento.
- Pode atuar como gateway para Modbus RTU via interface RS485.
- Permite acesso a todos os marcadores e dados utilizados para programação em ladder do PLC300.
- Permite a conexão de até 8 clientes Modbus TCP simultaneamente.

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.

4.1 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 cliente até o início da resposta do servidor, pode variar conforme o valor do ciclo de scan do equipamento.

4.2 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.

4.2.1 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 ethernet disponíveis para o PLC300 estão descritos no item 7. Para a descrição de outros marcadores disponíveis e função de cada marcador, consulte o manual do usuário do PLC300.

4.2.2 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 ethernet disponíveis para o PLC300 estão descritos no item 7. Para a descrição de outros marcadores disponíveis e função de cada marcador, consulte o manual do usuário do PLC300.

4.2.3 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.

4.2.4 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.

4.2.5 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.

4.2.6 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.

4.2.7 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.

4.3 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 4.1: *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 4.2: *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:

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

4.4 UTILIZAÇÃO COMO GATEWAY MODBUS RTU

Quando o Unit ID do servidor Modbus TCP está configurado com o valor 255, e a interface RS485 estiver configurada como mestre Modbus RTU, mensagens recebidas pelo servidor que contenham Unit ID com valores entre 1 e 247 serão encaminhadas através do mestre Modbus RTU via RS485 aos escravos desta rede. Mensagens com Unit ID igual a 0 ou 255 serão interpretadas pelo PLC300.

Caso ocorra timeout da resposta do escravo Modbus RTU o gateway retornará um telegrama indicando erro ao cliente Modbus TCP que originou a requisição.

5 DESCRIÇÃO DETALHADA DAS FUNÇÕES MODBUS

Neste item é feita uma descrição detalhada das funções disponíveis no controlador programável PLC300 para comunicação Modbus. 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).

5.1 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	Resposta
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
	etc...

Cada bit da resposta é colocado em uma posição dos bytes de dados enviados. O primeiro byte recebe os 8 primeiros bits a partir do endereço inicial indicado na pergunta. 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 são 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).
- Número do bit inicial: 24000 = 5DC0h
- Número de bits lidos: 8 = 0008h

Pergunta		Resposta	
Campo	*Valor*	*Campo*	*Valor*
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		
Quantidade de bits (low)	08h		

NOTA!
A função 02 – Read Input Discrete – possui exatamente a mesma estrutura da função 1. Somente o código da função e os dados acessíveis são diferentes.

5.2 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	Resposta
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)
	Dado 2 (low)
	etc...

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

Pergunta		Resposta	
Campo	Valor	Campo	Valor
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
		Valor do float (high-low)	80h

NOTA!
A função 04 – Read Input Register – possui exatamente a mesma estrutura da função 3. Somente o código da função e os dados acessíveis são diferentes.

5.3 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). Possui a seguinte estrutura (cada campo representa um byte):

Pergunta	Resposta
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)

Exemplo: escrita do primeiro bit do marcador de saída %QB0, mapeado como coil a partir do endereço 16000.
- Número do bit: 16000 = 3E80h
- Valor para o bit: 1, logo o valor que deve ser escrito é FF00h

Pergunta		Resposta	
Campo	*Valor*	*Campo*	*Valor*
Função	05h	Função	05h
Número do bit (high)	3Eh	Número do bit (high)	1Fh
Número do bit (low)	80h	Número do bit (low)	40h
Valor para o bit (high)	FFh	Valor para o bit (high)	FFh
Valor para o bit (low)	00h	Valor para o bit (low)	00h

Note que para esta função, a resposta é uma cópia idêntica da requisição.

5.4 FUNÇÃO 06 – WRITE SINGLE REGISTER

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

Pergunta	Resposta
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)

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

Pergunta		Resposta	
Campo	*Valor*	*Campo*	*Valor*
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

Note que para esta função, a resposta é uma cópia idêntica da requisição.

5.5 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	Resposta
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)	
Byte 1	
Byte 2	
Byte 3	
etc...	

O valor de cada bit que está sendo escrito é colocado em uma posição dos bytes de dados enviados. O primeiro byte recebe os 8 primeiros bits a partir do endereço inicial indicado. 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.

- 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		Resposta	
Campo	*Valor*	*Campo*	*Valor*
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		
Valor para os bits	0Ah		
Valor para os bits	14h		

5.6 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	Resposta
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)	
Dado 1 (high)	
Dado 1 (low)	
Dado 2 (high)	
Dado 2 (low)	
etc...	

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

Pergunta		Resposta	
Campo	*Valor*	*Campo*	*Valor*
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		
Valor para o inteiro (low-high)	03h		
Valor para o inteiro (low-low)	04h		
Valor para o inteiro (high-high)	01h		
Valor para o inteiro (high-low)	02h		

5.7 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	Resposta
Função	Função
MEI Type	MEI Type
Código de leitura	Conformity Level
Número do Objeto	More Follows
	Próximo objeto
	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...

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:

Pergunta		Resposta	
Campo	*Valor*	*Campo*	*Valor*
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
		More Follows	00h
		Próximo Objeto	00h
		Número de objetos	01h
		Código do Objeto	02h
		Tamanho do Objeto	05h
		Valor do Objeto	'V1.00'

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').

5.8 ERROS DE COMUNICAÇÃO

Erros de comunicação podem ocorrer tanto na transmissão dos telegramas quanto no conteúdo dos telegramas transmitidos. Errros de transmissão e conexão são tratados diretamente pela interface Ethernet e pelo protocolo TCP/IP.

No caso de uma recepção com sucesso, se problemas forem detectados durante o tratamento do telegrama, uma mensagem indicando o tipo de erro ocorrido é retornada:

Código do erro	Descrição
1	Função inválida: a função solicitada não está implementada para o equipamento.
2	Endereço de dado inválido: o endereço do dado (registrador ou bit) não existe.
3	Valor de dado inválido: ▪ Valor está fora da faixa permitida. ▪ Escrita em dado que não pode ser alterado (registrador ou bit somente leitura).
4	Gateway Modbus TCP/RTU não pode encaminhar mensagem porque o endereço do escravo é inválido.
10	Gateway Modbus TCP/RTU desabilitado.
11	Gateway Modbus TCP/RTU identificou timeout aguardando resposta do escravo.

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

As mensagens de erro possuem a seguinte estrutura:

Pergunta	Resposta
Função	Função (com o bit mais significativo em 1)
Dados	Código do erro

Exemplo: solicitação de escrita no registrador 2900 (supondo registrador 2900 como sendo inexistente):

Pergunta		Resposta	
Campo	*Valor*	*Campo*	*Valor*
Função	06h	Função	86h
Registrador (high)	0Bh	Código de erro	02h
Registrador (low)	54h		
Valor (high)	00h		
Valor (low)	00h		

6 OPERAÇÃO NA REDE MODBUS TCP – MODO CLIENTE

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

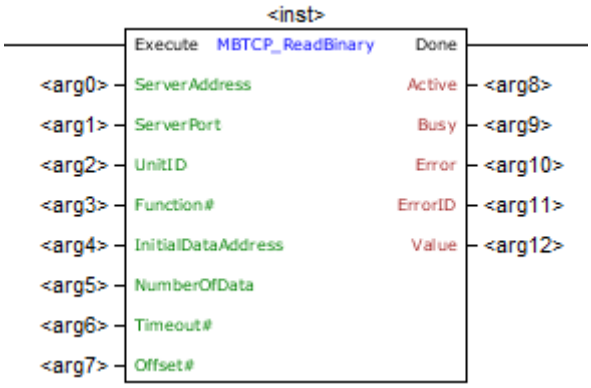
- O envio e recepção de telegramas via interface Ethernet utilizando o protocolo Modbus TCP é programado utilizando blocos em linguagem de programação ladder. É necessário conhecer os blocos disponíveis e o software de programação em ladder para poder programar o cliente da rede.
- As seguintes funções estão disponíveis para envio de requisições pelo cliente Modbus TCP:
  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

6.1 BLOCOS PARA A PROGRAMAÇÃO DO CLIENTE

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

6.1.1 MB TCP Read Binary – Leitura de Bits

Bloco para leitura de bits. Permite fazer a leitura de até 128 bits em sequência do servidor, 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 cliente Modbus TCP quando a conexão estiver livre. Ao término com sucesso da operação – resposta recebida do servidor – 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>: “ServerAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: DWORD
Descrição: Endereço IP do servidor.

<arg1>: “ServerPort” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Porta Modbus TCP do servidor.

<arg2>: “UnitID” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: UnitID do servidor.

<arg3>: “Function#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de leitura: 1= "Read Coils"; 2= "Read Discrete Inputs".

<arg4>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do bit inicial – 0 a 65535.

<arg5>: “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.

<arg6>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do servidor, a partir do início do envio pelo cliente – 20 a 5000 ms.

<arg7>: “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: 0= "Sem Offset"; 1= "Modicon" (Com Offset de 1).

Saídas:

<arg8>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de leitura enviada para o servidor e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

<arg9>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (conexão 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.

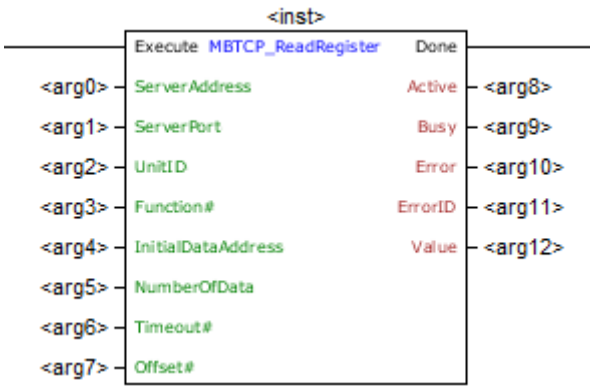
<arg10>: “Error” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Erro na execução da requisição.
Nota: A variável tem que ter permissão de escrita.

<arg11>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE, USINT ou SINT
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= "Cliente não habilitado"; 4= "Timeout na resposta do servidor"; 5= "Servidor retornou erro".
Nota: A variável tem que ter permissão de escrita.

<arg12>: “Value” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL, BOOL[1 … 128]
Descrição: Variável ou array onde serão salvos os dados lidos do servidor.
Nota: A variável tem que ter permissão de escrita.

6.1.2 MB TCP Read Register – Leitura de Registradores

Bloco para leitura de registradores de 16 bits. Permite fazer a leitura de até 8 registradores em sequência do servidor, 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 cliente Modbus TCP quando a conexão estiver livre. Ao término com sucesso da operação – resposta recebida do servidor – 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>: “ServerAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: DWORD
Descrição: Endereço IP do servidor.

<arg1>: “ServerPort” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Porta Modbus TCP do servidor.

<arg2>: “UnitID” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: UnitID do servidor.

<arg3>: “Function#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de leitura: 3= "Read Holding Registers"; 4= "Read Input Registers".

<arg4>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do registrador inicial – 0 a 65535.

<arg5>: “NumberOfData” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Número de registradores lidos a partir do endereço inicial – 1 a 16.

<arg6>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do servidor, a partir do início do envio pelo cliente – 20 a 5000 ms.

<arg7>: “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: 0= "Sem Offset"; 1= "Modicon" (Com Offset de 1).

Saídas:

<arg8>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de leitura enviada para o servidor e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

<arg9>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (conexão 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.

<arg10>: “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.

<arg11>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE, USINT ou SINT
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= "Cliente não habilitado"; 4= "Timeout na resposta do servidor"; 5= "Servidor retornou erro".
Nota: A variável tem que ter permissão de escrita.

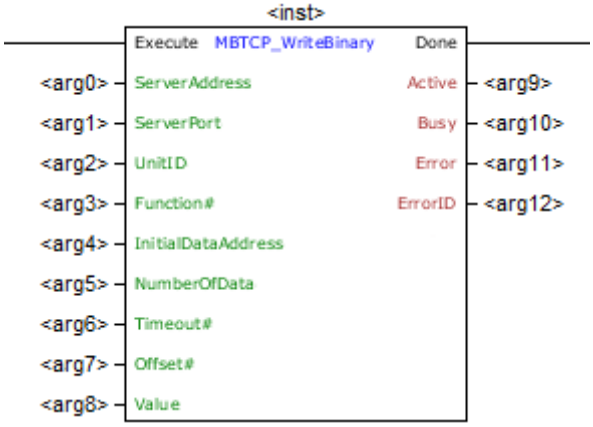
<arg12>: “Value” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE[2 ... 32], SINT[2 ... 32], USINT[2 ... 32], WORD, WORD[1 ... 16], UINT, UINT[1 ... 16], INT, INT[1 ... 16], DWORD, DWORD[1 ... 8], UDINT, UDINT[1 ... 8], DINT, DINT[1 ... 8], REAL ou REAL[1 ... 8]
Descrição: Variável ou array onde serão salvos os dados lidos do servidor.
Nota: A variável tem que ter permissão de escrita.

NOTA!

- O protocolo Modbus, 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.

6.1.3 MB TCP 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 cliente Modbus TCP quando a conexão estiver livre. Ao término com sucesso da operação – resposta recebida do servidor – 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>: “ServerAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: DWORD
Descrição: Endereço IP do servidor.

<arg1>: “ServerPort” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Porta Modbus TCP do servidor.

<arg2>: “UnitID” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: UnitID do servidor.

<arg3>: “Function#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de escrita: 5= "Write Single Coil"; 15= "Write Multiple Coils".

<arg4>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do bit inicial – 0 a 65535.

<arg5>: “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.

<arg6>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do servidor, a partir do início do envio pelo cliente – 20 a 5000 ms.

<arg7>: “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: 0= "Sem Offset"; 1= "Modicon" (Com Offset de 1).

<arg8>: “Value” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BOOL, BOOL[1 ... 128]
Descrição: Variável ou array com os dados que serão escritos no servidor.

Saídas:

<arg9>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de escrita enviada para o servidor e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

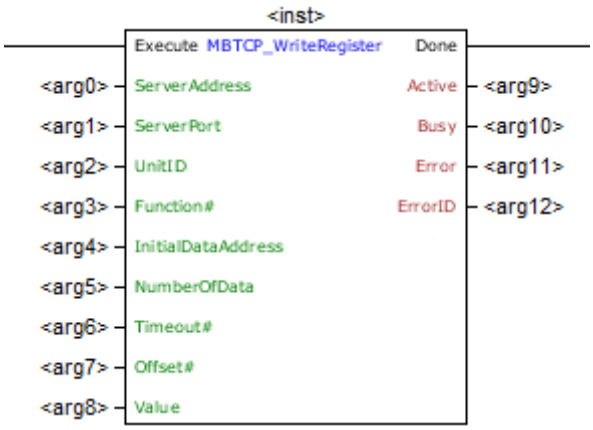
<arg10>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (conexão ocupada com outra requisição), aguardando liberação para que a solicitação seja enviada pelo bloco. Se a entrada de habilitação for retirada enquanto o bloco faz esta indicação, a requisição é descartada.
Nota: A variável tem que ter permissão de escrita.

<arg11>: “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.

<arg12>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE, USINT ou SINT
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= "Cliente não habilitado"; 4= "Timeout na resposta do servidor"; 5= "Servidor retornou erro".
Nota: A variável tem que ter permissão de escrita.

6.1.4 MB TCP Write Register – Escrita de Registradores

Bloco para escrita de registradores. Permite fazer a escrita de um ou mais 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 cliente Modbus TCP quando a conexão estiver livre. Ao término com sucesso da operação – resposta recebida do servidor – 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>: “ServerAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: DWORD
Descrição: Endereço IP do servidor.

<arg1>: “ServerPort” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Porta Modbus TCP do servidor.

<arg2>: “UnitID” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: UnitID do servidor.

<arg3>: “Function#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Código da função de escrita: 6= "Write Single Register"; 16= "Write Multiple Registers".

<arg4>: “InitialDataAddress” – VAR_IN: inserir uma variável (tag).
Tipos de dados: WORD
Descrição: Endereço do registrador inicial – 0 a 65535.

<arg5>: “NumberOfData” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE
Descrição: Número de registradores escritos a partir do endereço inicial – 1 a 16.

<arg6>: “Timeout#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Tempo de espera para chegada da resposta do servidor, a partir do início do envio pelo cliente – 20 a 5000 ms.

<arg7>: “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: 0= "Sem Offset"; 1= "Modicon" (Com Offset de 1).

<arg8>: “Value” – VAR_IN: inserir uma variável (tag).
Tipos de dados: BYTE[2 ... 32], USINT[2 ... 32], SINT[2 ... 32], WORD, WORD[1 ... 16], UINT, UINT[1 ... 16], INT, INT[1 ... 16], DWORD, DWORD[1 ... 8], UDINT, UDINT[1 ... 8], DINT, DINT[1 ... 8], REAL ou REAL[1 ... 8]
Descrição: Variável ou array com os dados que serão escritos no servidor.

Saídas:

<arg9>: “Active” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco ativo, requisição de escrita enviada para o servidor e aguardando resposta.
Nota: A variável tem que ter permissão de escrita.

<arg10>: “Busy” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Bloco habilitado, mas recurso não está disponível (conexão 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.

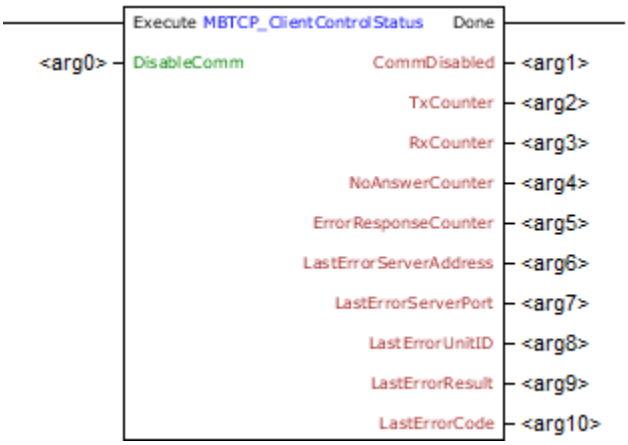
<arg11>: “Error” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Erro na execução da requisição.
Nota: A variável tem que ter permissão de escrita.

<arg12>: “ErrorID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE, USINT ou SINT
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= "Cliente não habilitado"; 4= "Timeout na resposta do servidor"; 5= "Servidor retornou erro".
Nota: A variável tem que ter permissão de escrita.

NOTA!
- O protocolo Modbus, 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.

6.1.5 MB TCP Client Control/Status – Controle e Estado do Modbus TCP

Bloco para controle e monitoração do cliente Modbus TCP. Sempre que uma rede Modbus TCP for montada com o PLC300 como cliente, 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 cliente Modbus TCP. Ao desabilitar o cliente, os contadores e marcadores de status do cliente Modbus TCP também são zerados: 0= "Cliente em execução"; 1= "Desabilita cliente".

Saídas:

<arg1>: “CommDisabled” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Indica se o cliente está ou não desabilitado: 0= "Cliente habilitado"; 1= "Cliente 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 cliente para os servidores. É zerado sempre que o equipamento for desligado ou o cliente 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 cliente. É zerado sempre que o equipamento for desligado ou o cliente 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 cliente que não foram respondidas pelos servidores. É zerado sempre que o equipamento for desligado ou o cliente 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 cliente e que os servidores 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 cliente for desabilitado – 0 a 65535.
Nota: A variável tem que ter permissão de escrita.

<arg6>: “LastErrorServerAddress” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: DWORD
Descrição: Indica o endereço IP do servidor no qual foi detectado o último erro de comunicação. É zerado sempre que o equipamento for desligado ou o cliente for desabilitado.
Nota: A variável tem que ter permissão de escrita.

<arg7>: “LastErrorServerPort” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: WORD ou UINT
Descrição: Indica a porta TCP do servidor no qual foi detectado o último erro de comunicação. É zerado sempre que o equipamento for desligado ou o cliente for desabilitado – 0 a 65535.
Nota: A variável tem que ter permissão de escrita.

<arg8>: “LastErrorUnitID” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Indica o Unit ID do servidor no qual foi detectado o último erro de comunicação. É zerado sempre que o equipamento for desligado ou o cliente for desabilitado – 0 a 255.
Nota: A variável tem que ter permissão de escrita.

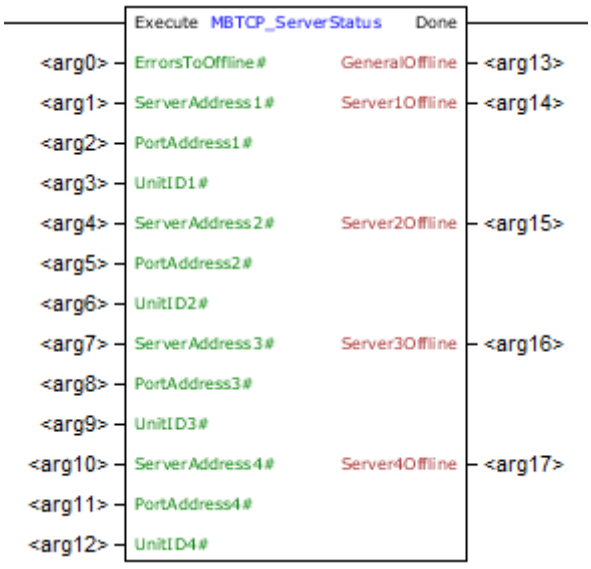
<arg9>: “LastErrorResult” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Indica o resultado da operação – timeout ou resposta de erro, conforme ERROR ID do bloco – para o servidor no qual foi detectado o último erro de comunicação. É zerado sempre que o equipamento for desligado ou o cliente for desabilitado: 0= "Sem erro detectado"; 4= "Timeout na resposta do servidor"; 5= "Servidor retornou erro", 6 = “Falha ao conectar ao servidor”, 7 = “Conexão TCP/IP terminada prematuramente”.
Nota: A variável tem que ter permissão de escrita.

<arg10>: “LastErrorCode” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BYTE ou USINT
Descrição: Indica o código do erro, no caso do cliente receber resposta de erro de algum servidor. É zerado sempre que o equipamento for desligado ou o cliente 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 7.

6.1.6 MB TCP Server Status – Estado dos Servidores da Rede Modbus TCP

Bloco para monitoração dos servidores da rede Modbus TCP. Deve ser utilizado caso seja desejado identificar problemas na comunicação do cliente com algum servidor da rede Modbus TCP.



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 ou USINT
Descrição: Permite programar, para este bloco, a quantidade de erros de comunicação que o cliente deve identificar até que a comunicação com um servidor seja considerada offline. É considerado erro de comunicação toda requisição (leitura ou escrita) que o cliente enviou para um servidor e não recebeu resposta.

<arg1>: “ServerAddress1#” – VAR_IN: inserir uma constante.
<arg4>: “ServerAddress2#” – VAR_IN: inserir uma constante.
<arg7>: “ServerAddress3#” – VAR_IN: inserir uma constante.
<arg10>: “ServerAddress4#” – VAR_IN: inserir uma constante.
Tipos de dados: DWORD
Descrição: Permite programar o endereço IP de até 4 servidores, 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 TCP, 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".

<arg2>: “ServerPort1#” – VAR_IN: inserir uma constante.
<arg5>: “ServerPort2#” – VAR_IN: inserir uma constante.
<arg8>: “ServerPort3#” – VAR_IN: inserir uma constante.
<arg11>: “ServerPort4#” – VAR_IN: inserir uma constante.
Tipos de dados: WORD
Descrição: Permite programar a porta Modbus TCP de até 4 servidores, 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 TCP, atinja o valor programado em “ErrorsToSetOffline”, a saída respectiva é acionada.

<arg3>: “UnitID1#” – VAR_IN: inserir uma constante.
<arg6>: “UnitID2#” – VAR_IN: inserir uma constante.
<arg9>: “UnitID3#” – VAR_IN: inserir uma constante.
<arg12>: “UnitID4#” – VAR_IN: inserir uma constante.
Tipos de dados: BYTE
Descrição: Permite programar o Unit ID de até 4 servidores, 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 TCP, atinja o valor programado em “ErrorsToSetOffline”, a saída respectiva é acionada.

Saídas:

<arg13>: “GeneralOffline#” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Se qualquer uma das saídas dos servidores indicados for acionada, esta saída também será acionada. Funciona como uma lógica OU entre as 4 saídas de indicação dos servidores.
Nota: A variável tem que ter permissão de escrita.

<arg14>: “Server1Offline#” – VAR_OUT: inserir uma variável (tag).
<arg15>: “Server2Offline#” – VAR_OUT: inserir uma variável (tag).
<arg16>: “Server3Offline#” – VAR_OUT: inserir uma variável (tag).
<arg17>: “Server4Offline#” – VAR_OUT: inserir uma variável (tag).
Tipos de dados: BOOL
Descrição: Saída acionada caso a quantidade de erros de comunicação em sequência para os servidores indicados nas respectivas entradas atinja o valor programado em “ErrorsToSetOffline”.
Nota: A variável tem que ter permissão de escrita.

7 MARCADORES DE SISTEMA PARA ETHERNET

Para a interface Ethernet, foram disponibilizados os seguintes marcadores de sistema de leitura (%S) e marcadores de sistema de escrita (%C), para controle e monitoração:

7.1 MARCADORES DE SISTEMA DE LEITURA

Estado da Interface Ethernet: conjunto de marcadores de leitura que indicam o estado da interface Ethernet.	
Marcador	**Descrição**
%SB3492 %SB3493 %SB3494 %SB3495 %SB3496 %SB3497	Endereço físico (MAC).
%SB3498	Modo de comunicação: 0 = Automático (Interface Ethernet está sendo configurada) 1 = 10 MBps Full Duplex 2 = 10 MBps Half Duplex 3 = 100 MBps Full Duplex 4 = 100 MBps Half Duplex.
%SB3499	Reservado.
%SD3500	Endereço IP.
%SD3504	Máscara de sub-rede.
%SD3508	Gateway padrão.

Estado do Servidor Modbus TCP: conjunto de marcadores de leitura que indicam a quantidade de telegramas enviados e recebidos pelo servidor Modbus TCP.	
Marcador	Descrição
%SW3512	Número de telegramas recebidos.
%SW3514	Número de telegramas transmitidos.
%SB3516	Número de conexões ativas.

Estado do Cliente Modbus TCP: conjunto de marcadores de leitura que indicam o estado do cliente Modbus TCP, além de informações para diagnóstico da rede.	
Marcador	Descrição
%SB3520	Estado do cliente Modbus TCP: 0 = Operação normal. 1 = Cliente desabilitado.
%SB3521	Reservado.
%SW3522	Contador de requisições feitas pelo cliente. Contador incrementado toda vez que um novo telegrama é enviado pelo cliente da rede Modbus TCP. É zerado sempre que atingir o limite máximo.
%SW3524	Contador de respostas recebidas com sucesso. Contador incrementado toda vez que o cliente receber uma resposta com sucesso de um servidor da rede. É zerado sempre que atingir o limite máximo.
%SW3526	Contador de requisições sem resposta – timeout. Contador incrementado toda vez que ocorrer timeout para uma requisição feita pelo cliente Modbus TCP para um servidor. É zerado sempre que atingir o limite máximo ou a interface for desabilitada.
%SW3528	Contador de respostas com erro recebidas. Contador incrementado toda vez que o servidor retornar uma resposta de erro para uma requisição feita pelo cliente Modbus TCP. 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 servidor, tipo de erro e código do erro serão salvos nos marcadores %SW3530 até %SB3538.
%SW3530	Último erro ocorrido: porta TCP do servidor.
%SD3532	Último erro ocorrido: endereço IP do servidor.
%SB3536	Último erro ocorrido: Unit ID do servidor.
%SB3537	Último erro ocorrido: tipo de erro. 0 = Sem erro. 4 = Timeout na resposta. 5 = Servidor retornou resposta de erro. 6 = Falha ao conectar ao servidor. 7 = Conexão TCP/IP terminada prematuramente. É zerado sempre que a interface for desabilitada.
%SB3538	Último erro ocorrido: código do erro recebido, caso o tipo seja resposta de erro. É zerado sempre que a interface for desabilitada.

7.2 MARCADORES DE SISTEMA DE ESCRITA

Configuração da Interface Ethernet: conjunto de marcadores de escrita para programar as configurações da interface Ethernet. Também são acessíveis através do menu Setup.	
Marcador	Descrição
%CD3424	Endereço IP.
%CD3428	Máscara de sub-rede.
%CD3432	Gateway padrão.
%CB3436	0 = DHCP desabilitado (padrão) 1 = DHCP habilitado.
%CB3437	Modo de comunicação: 0 = Auto (padrão) 1 = 10 Mb Full Duplex 2 = 10 Mb Half Duplex 3 = 100 Mb Full Duplex 4 = 100 Mb Half Duplex.

Configuração do Servidor Modbus TCP: conjunto de marcadores de escrita para programar as configurações do servidor Modbus TCP. Também são acessíveis através do menu Setup.	
Marcador	Descrição
%CD3440	Autenticação de IP. Se diferente de zero, somente este endereço IP pode se conectar ao servidor Modbus TCP.
%CW3444	Porta TCP (padrão 502).
%CB3446	UnitID (padrão 255).
%CB3447	Reservado.
%CW3448	Timeout de recepção do escravo Modbus RTU (padrão 1000 ms).

Controle do Cliente Modbus TCP: conjunto de marcadores de escrita para controle do cliente Modbus TCP.	
Marcador	Descrição
%CW3452	Controle do cliente Modbus TCP: 0 = Operação normal. 1 = Desabilita interface.

Configuração do Cliente SNTP: conjunto de marcadores de escrita para programar as configurações do cliente SNTP.	
Marcador	Descrição
%CD3456	Endereço IP do servidor SNTP.
%CD3460	Endereço IP do servidor SNTP redundante.
%CW3464	Freqüência de atualização.
%CW3466	Timeout de recepção.

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