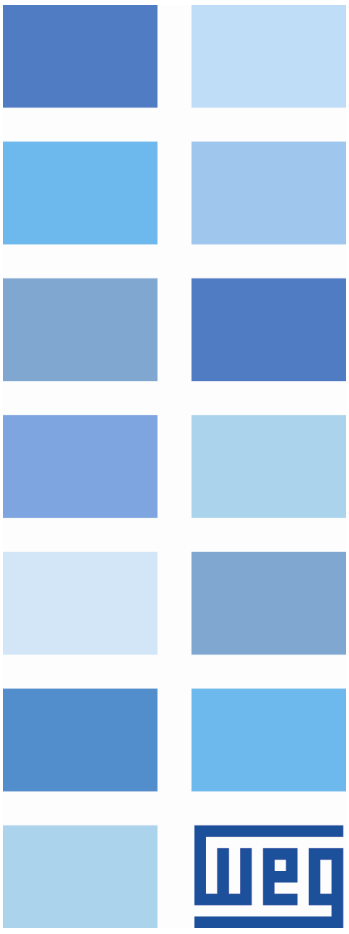
Modbus RTU

MW500

Manual do Usuário



Manual do Usuário Modbus RTU

Série: MW500

Idioma: Português

N º do Documento: 10002253015 / 01

Data da Publicação: 03/2022

SUMÁRIO

SOBRE O MANUAL

Este manual fornece a descrição necessária para a operação do inversor de frequência MW500 utilizando o protocolo Modbus RTU. Este manual deve ser utilizado em conjunto com manual do usuário do MW500.

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, RS485 e USB disponíveis para o produto, bem como a descrição detalhada do protocolo Modbus RTU para utilização destas interfaces.

2 DESCRIÇÃO DAS INTERFACES

As interfaces para comunicação serial RS232, RS485 ou USB disponíveis para o inversor de frequência MW500 dependem do módulo plug-in selecionado para o produto. A seguir são apresentadas informações sobre a conexão e instalação do equipamento em rede de comunicação utilizando diferentes módulos plug-in.

2.1 MÓDULOS PLUG-IN COM INTERFACE RS485

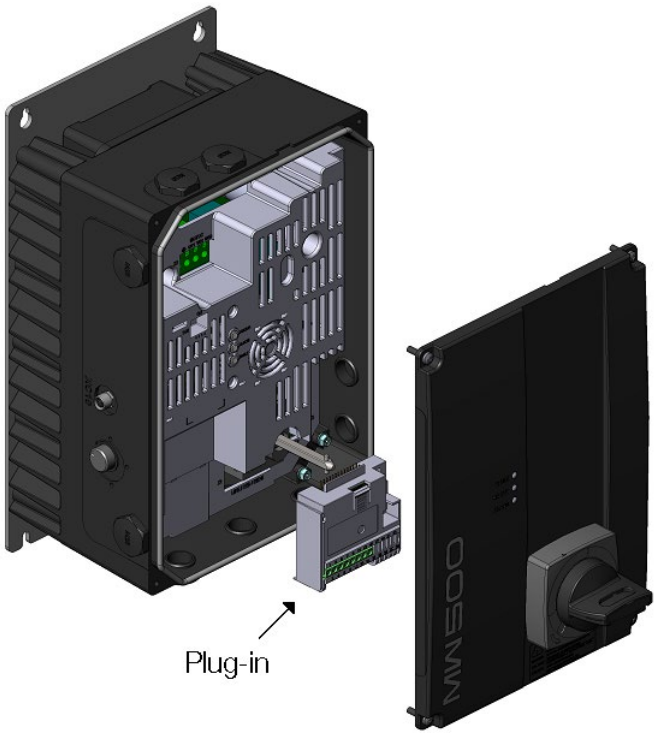


Figura 2.1: *Exemplo de acessório de interface para MW500*

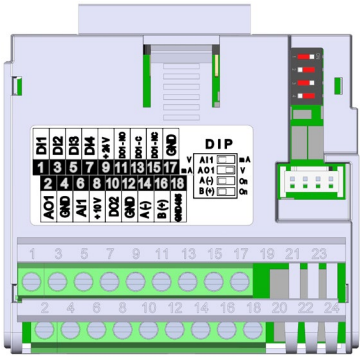
Todos os módulos plug-in para o inversor de frequência MW500 possuem ao menos uma interface RS485 padrão, identificada como Serial (1). Esta interface RS485 padrão possui duas funções:

- Conexão ponto a ponto com HMI remota.
- Conexão via RS485 para operação em rede.

A seleção da função que será utilizada para o produto é feita através do parâmetro P0312.

2.1.1 Conector RS485 do módulo plug-in padrão (CFW500-IOS)

Para o módulo plug-in padrão, a conexão para a interface RS485 está disponível através do borne de controle utilizando a seguinte pinagem:



Pino	Nome	Função
12	A-Line (-)	RxD/TxD negativo
14	B-Line (+)	RxD/TxD positivo
16	Ref.	0V do circuito RS485

Tabela 2.1: *Pinagem do conector RS485 para o módulo plug-in padrão (CFW500-IOS)*

2.2 MÓDULOS PLUG-IN COM INTERFACE RS485 E INTERFACE ADICIONAL

Dependendo do módulo plug-in instalado, o MW500 dispõe de até duas interfaces seriais simultâneas, porém somente uma delas pode ser fonte de comandos ou referências, a outra é obrigatoriamente inativa ou HMI remota, conforme a seleção de P0312.

A interface Serial(1) é a interface padrão do MW500 e está presente em todos os módulos plug-in através dos bornes da porta RS485 padrão. Já a interface Serial(2) está presente somente nos módulos plug-in descritos a seguir:

NOTA!
Não é possível utilizar as interfaces seriais para comunicação com duas redes distintas. A única operação simultânea permitida é utilizar a Serial (1) para conexão com a HMI remota, e outro protocolo de rede programável na Serial (2).

2.2.1 Módulo plug-in com duas interfaces RS485 (CFW500-CRS485-B)

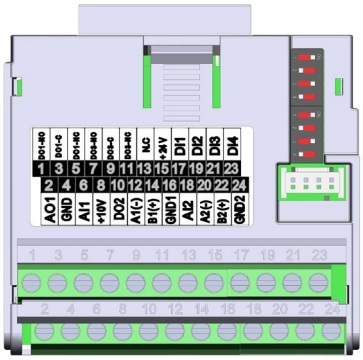


Figura 2.2: *Módulo plug-in com RS485 adicional*

Para este módulo plug-in, além da interface RS485 padrão, uma segunda interface RS485 está disponível. Este acessório permite a ligação simultânea de uma HMI remota na interface RS485 padrão e de uma interface serial programável.

2.2.2 Módulo plug-in com interfaces RS485 e USB (CFW500-CUSB)

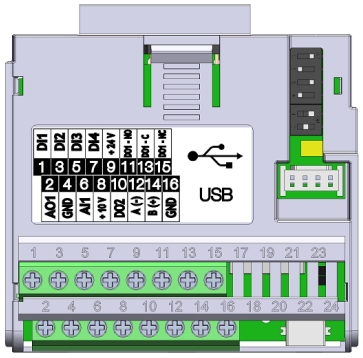


Figura 2.3: *Módulo plug-in com conexão USB*

Para este módulo plug-in, além da interface RS485 padrão, uma interface USB com conector mini-USB está disponível. Ao conectar a interface USB, esta será reconhecida como um conversor USB para serial, e uma porta COM virtual será criada[1]. Portanto a comunicação com o drive é feita através desta porta COM. Desta forma, o acessório USB também disponibiliza a conexão para HMI remota na interface RS485 padrão, mais uma interface serial programável.

2.2.3 Módulo plug-in com interfaces RS485 e RS232 (CFW500-CRS232)

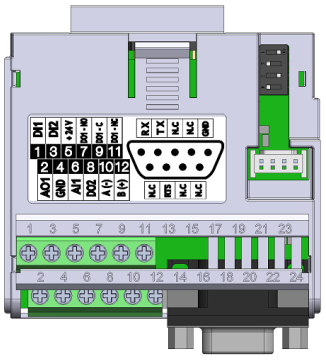


Figura 2.4: *Módulo plug-in com conexão RS232*

Para este módulo plug-in, além da interface RS485 padrão, uma interface RS232 está disponível. Este acessório permite a ligação simultânea de uma HMI remota na interface RS485 padrão e de uma interface serial programável.

2.3 RS485

2.3.1 Características da interface RS485

- Interface segue o padrão EIA/TIA-485.
- Possibilita comunicação utilizando taxas de 9600 até 38400 Kbit/s.

[1] É necessário instalar o driver USB presente no CD-ROM que acompanha o produto. O número da porta COM criada depende da disponibilidade no sistema operacional e, depois de conectado, devem-se consultar os recursos de hardware do sistema para identificar esta porta.

- 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.[2]
- Comprimento máximo do barramento de 1000 metros.

2.3.2 Resistor de terminação

Para cada segmento da rede RS485, é necessário habilitar um resistor de terminação nos pontos extremos do barramento principal. O inversor de frequência MW500 possui chaves que podem ser ativadas para habilitar o resistor de terminação. Consulte o guia de instalação do módulo plug-in para detalhes.

2.3.3 Indicações

As indicações de alarmes, falhas e estados da comunicação são feitas através da HMI e dos parâmetros do produto.

2.3.4 Conexão com a Rede RS485

Para a ligação do inversor de frequência MW500 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.

2.4 RS232

2.4.1 Indicações

As indicações de alarmes, falhas e estados da comunicação são feitas através da HMI e dos parâmetros do produto.

2.4.2 Conexão com a Rede RS232

- Os sinais RX e TX do inversor 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 o inversor e motor.

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

2.4.3 Cabos para Ligação em RS232

Caso seja desejado, estão disponíveis itens dos seguintes cabos para ligação em RS232 entre o inversor e um mestre da rede, como um PC:

Cabo	Item
Cabo RS232 blindado com conectores DB9 Comprimento: 3 metros	10050328
Cabo RS232 blindado com conectores DB9 fêmea Comprimento: 10 metros	10191117

Outros cabos, porém, podem ser encontrados no mercado – em geral denominados null-modem – ou montados de acordo com o desejado para a instalação.

3 PARAMETRIZAÇÃO

A seguir serão apresentados apenas os parâmetros do inversor de frequência MW500 que possuem relação direta com a comunicação Modbus RTU.

3.1 SÍMBOLOS PARA DESCRIÇÃO DAS PROPRIEDADES

RO	Parâmetro somente de leitura
CFG	Parâmetro somente alterado com o motor parado

P0105 – SELEÇÃO 1ª/2ª RAMPA

P0220 – SELEÇÃO FONTE LOCAL/REMOTO

P0221 – SELEÇÃO REFERÊNCIA LOCAL

P0222 – SELEÇÃO REFERÊNCIA REMOTA

P0223 – SELEÇÃO GIRO LOCAL

P0224 – SELEÇÃO GIRA/PARA LOCAL

P0225 – SELEÇÃO JOG LOCAL

P0226 – SELEÇÃO GIRO REMOTO

P0227 – SELEÇÃO GIRA/PARA REMOTO

P0228 – SELEÇÃO JOG REMOTO

Estes parâmetros são utilizados na configuração da fonte de comandos para os modos local e remoto do produto. Para que o equipamento seja controlado através da interface Modbus RTU, deve-se selecionar uma das opções 'serial' disponíveis nos parâmetros.

A descrição detalhada destes parâmetros encontra-se no manual de programação do inversor de frequência MW500.

P0308 – ENDEREÇO SERIAL

Faixa de Valores:	1 a 247	**Padrão:** 1
Propriedades:	CFG	
Grupo de acesso via HMI:	NET	

Descrição:
Permite programar o endereço utilizado para comunicação serial do equipamento. É necessário que cada equipamento da rede possua um endereço diferente dos demais. Os endereços válidos para este parâmetro dependem do protocolo programado no P0312:

- P0312 = 0 (HMI) → não necessita programação de endereço.
- P0312 = 1 (Modbus RTU) → endereços válidos: 1 a 247.
- P0312 = 6 (HMI/Modbus RTU) → interface padrão = HMI (não necessita de endereço).
  → interface adicional = Modbus RTU (endereços válidos: 1 a 247).

P0310 – TAXA DE COMUNICAÇÃO SERIAL

Faixa de Valores:	0 = 9600 bit/s 1 = 19200 bit/s 2 = 38400 bit/s	**Padrão:** 1
Propriedades:	CFG	
Grupo de acesso via HMI:	NET	

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.

NOTA!
Para a utilização da interface RS485 com a HMI remota não é necessário programar a taxa de comunicação. Esta taxa é utilizada apenas com os demais protocolos seriais, tanto pela interface padrão quanto pelas interfaces adicionais.

P0311 – CONFIGURAÇÃO DOS BYTES DA INTERFACE SERIAL

Faixa de Valores:	0 = 8 bits de dados, sem paridade, 1 stop bit 1 = 8 bits de dados, paridade par, 1 stop bit 2 = 8 bits de dados, paridade ímpar, 1 stop bit 3 = 8 bits de dados, sem paridade, 2 stop bits 4 = 8 bits de dados, paridade par, 2 stop bits 5 = 8 bits de dados, paridade ímpar, 2 stop bits	**Padrão:** 1
Propriedades:	CFG	
Grupo de acesso via HMI:	NET	

Descrição:
Permite a configuração do número de bits de dados, paridade e *stop* bits nos bytes da interface serial. Esta configuração deve ser a mesma para todos os equipamentos conectados na rede.

NOTA!
Para a utilização da interface RS485 com a HMI remota não é necessário programar a configuração dos bytes. Esta configuração é utilizada apenas com os demais protocolos seriais, tanto pela interface padrão quanto pelas interfaces adicionais.

P0312 – PROTOCOLO SERIAL

Faixa de Valores:
0 = HMI (1)
1 = SymbiNet (1)
2 = Modbus RTU (1)
3 = BACnet (1)
4 = Reservado
5 = Mestre Modbus RTU (1) e Modbus RTU (2)
6 = HMI (1) e Modbus RTU (2)
7 = Modbus RTU (2)
8 = HMI (1) e BACnet (2)
9 = BACnet (2)
10 = Reservado
11 = Reservado
12 = HMI (1) e Mestre Modbus RTU (2)
13 = Modbust RTU (1) e Mestre Modbus RTU (2)
14 = HMI (1) e SymbiNet (2)
15 = SymbiNet (2)

Padrão: 2

Propiedades: CFG

Grupo de acesso via HMI: NET

Descrição:
Permite selecionar o protocolo desejado para a interface serial.

Tabela 3.1: *Opções para o parâmetro P0312*

Opção	Descrição
0 = HMI (1)	Seleciona, para a interface serial padrão (1), o protocolo de comunicação com a HMI remota.
1 = SymbiNet (1)	Seleciona, para a interface serial padrão (1), o protocolo de comunicação SymbiNet.
2 = Modbus RTU (1)	Seleciona, para a interface serial padrão (1), o protocolo de comunicação Modbus RTU escravo.
3 = BACnet (1)	Seleciona, para a interface serial padrão (1), o protocolo de comunicação BACnet.
4 = Reservado	
5 = Mestre Modbus RTU (1) e Modbus RTU (2)	Para os acessórios de interface do equipamento que possuírem mais de uma interface serial, esta opção permite utilizar a interface padrão (1) como mestre Modbus RTU e, simultaneamente, utilizar a interface adicional (2) como escravo Modbus RTU.
6 = HMI (1) e Modbus RTU (2)	Para os acessórios de interface do equipamento que possuírem mais de uma interface serial (exemplos: CFW500-CUSB, etc.), esta opção permite utilizar HMI do equipamento conectada na interface padrão (1) e, simultaneamente, utilizar o protocolo Modbus RTU na interface adicional (2).
7 = Modbus RTU (2)	Seleciona, para a interface serial adicional (2), o protocolo de comunicação Modbus RTU escravo. A interface serial padrão (1) permanece desabilitada.
8 = HMI (1) e BACnet (2)	Para os acessórios de interface do equipamento que possuírem mais de uma interface serial, esta opção permite utilizar HMI do equipamento conectada na interface padrão (1) e, simultaneamente, utilizar a interface adicional (2) com BACnet.
9 = BACnet (2)	Seleciona, para a interface serial adicional (2), o protocolo de comunicação BACnet. A interface serial padrão (1) permanece desabilitada.
10 ... 11 = Reservado	

12 = HMI (1) e Mestre Modbus RTU (2)	Para os acessórios de interface do equipamento que possuírem mais de uma interface serial, esta opção permite utilizar HMI do equipamento conectada na interface padrão (1) e, simultaneamente, utilizar o drive como mestre Modbus RTU na interface adicional (2).
13 = Modbus RTU (1) e Mestre Modbus RTU (2)	Para os acessórios de interface do equipamento que possuírem mais de uma interface serial, esta opção permite utilizar a interface padrão (1) como escravo Modbus RTU e, simultaneamente, utilizar o drive como mestre Modbus RTU na interface adicional (2).
14 = HMI (1) e SymbiNet (2)	Para os acessórios de interface do equipamento que possuírem mais de uma interface serial, esta opção permite utilizar HMI do equipamento conectada na interface padrão (1) e, simultaneamente, utilizar a interface adicional (2) com Symbinet.
15 = SymbiNet (2)	Seleciona, para a interface serial adicional (2), o protocolo de comunicação SymbiNet. A interface serial padrão (1) permanece desabilitada.

NOTA!
Para mais detalhes sobre o funcionamento do drive como mestre Modbus RTU, deve-se consultar o manual da SoftPLC (documento número 10001499063) e no help do software de programação WLP ou WPS.

P0313 – AÇÃO PARA ERRO DE COMUNICAÇÃO

Faixa de Valores: 0 = Inativo
1 = Para por Rampa
2 = Desabilita Geral
3 = Vai para Local
4 = Vai para Local e mantém comandos e referência
5 = Causa Falha
Padrão: 1

Propriedades: CFG

Grupo de acesso via HMI: NET

Descrição:
Este parâmetro permite selecionar qual a ação deve ser executada pelo equipamento, caso ele seja controlado via rede e um erro de comunicação seja detectado.

Tabela 3.2: *Opções para o parâmetro P0313*

Opção	Descrição
0 = Inativo	Nenhuma ação é tomada, equipamento permanece no estado atual.
1 = Para por Rampa	O comando de parada por rampa é executado, e o motor para de acordo com a rampa de desaceleração programada.
2 = Desabilita Geral	O equipamento é desabilitado geral, e o motor para por inércia.
3 = Vai para Local	O equipamento é comandado para o modo local.
4 = Vai para Local e mantém comandos e referência	O equipamento é comandado para o modo local, mas os comandos de habilitação e a referência de velocidade recebidos via rede são mantidos em modo local, desde que o equipamento seja programado para utilizar, em modo local, comandos via HMI ou Start/Stop a 3 fios, e a referência de velocidade via HMI ou potenciômetro eletrônico.
5 = Causa Falha	No lugar de alarme, um erro de comunicação causa uma falha no equipamento, sendo necessário fazer o reset de falhas do equipamento para o retorno da sua operação normal.

São considerados erros de comunicação os seguintes eventos:

Comunicação Serial (RS485):

- Alarme A128/Falha F228: *timeout* da interface serial.

As ações descritas neste parâmetro são executadas através da escrita automática dos respectivos bits no parâmetro de controle da interface de rede que corresponde à falha detectada. Desta forma, para que os comandos escritos neste parâmetro tenham efeito, é necessário que o equipamento esteja programado para ser controlado pela interface de rede utilizada (com exceção da opção "Causa Falha", que bloqueia o equipamento mesmo que ele não seja controlado via rede). Esta programação é feita através dos parâmetros P0220 até P0228.

P0314 – WATCHDOG SERIAL

Faixa de Valores:	0,0 a 999,0s	**Padrão:** 0,0
Propriedades:	CFG	
Grupo de acesso via HMI:	NET	

Descrição:
Permite programar um tempo para a detecção de erro de comunicação via interface serial. Caso o inversor de frequência fique sem receber telegramas válidos por um tempo maior do que o programado neste parâmetro, será considerado que ocorreu um erro de comunicação, mostrado o alarme A128 na HMI (ou falha F228, dependendo da programação feita no P0313) e a ação programada no P0313 será executada.

Depois de energizado, o inversor de frequência começará a contar este tempo a partir do primeiro telegrama válido recebido. O valor 0,0 desabilita esta função.

P0316 – ESTADO DA INTERFACE SERIAL

Faixa de Valores:	0 = Inativo 1 = Ativo 2 = Erro de Watchdog	**Padrão:** -
Propriedades:	RO	
Grupo de acesso via HMI:	NET	

Descrição:
Permite identificar se o cartão de interface serial RS485 está devidamente instalado, e se a comunicação serial apresenta erros.

Tabela 3.3: *Valores para o parâmetro P0316*

Valores	Descrição
0 = Inativo	Interface serial inativa. Ocorre quando o equipamento não possui cartão de interface RS485 instalado.
1 = Ativo	Cartão de interface RS485 instalado e reconhecido.
2 = Erro de Watchdog	Interface serial ativa, mas detectado erro de comunicação serial – alarme A128/falha F228.

P0680 – ESTADO LÓGICO

Faixa de Valores:	0000h a FFFFh	**Padrão:** -
Propriedades:	RO	
Grupo de acesso via HMI:	NET	

Descrição:
Permite a monitoração do estado do equipamento. Cada bit representa um estado:

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3 a 0
Função	Em Falha	Automático (PID)	Subtensão	LOC/REM	JOG	Sentido de Giro	Habilitado Geral	Motor Girando	Em Alarme	Em modo de configuração	Segunda Rampa	Parada Rápida Ativa	Reservado

Tabela 3.4: *Funções dos bits para o parâmetro P0680*

Bits	Valores
Bits 0 a 3	Reservado.
Bit 4 Parada Rápida Ativa	0: Drive não possui comando de parada rápida ativo. 1: Drive está executando o comando de parada rápida.
Bit 5 Segunda Rampa	0: Drive está configurado para utilizar como rampa de aceleração e desaceleração para o motor a primeira rampa, programada nos parâmetros P0100 e P0101. 1: Drive está configurado para utilizar como rampa de aceleração e desaceleração para o motor a segunda rampa, programada nos parâmetros P0102 e P0103.
Bit 6 Em Modo de Configuração	0: Drive operando normalmente. 1: Drive em modo de configuração. Indica uma condição especial na qual o drive não pode ser habilitado: Executando rotina de auto-ajuste. Executando rotina de start-up orientado. Executando função copy da HMI. Executando rotina auto-guiada do cartão de memória flash. Possui incompatibilidade de parametrização. Sem alimentação no circuito de potência do drive.
Bit 7 Em Alarme	0: Drive não está no estado de alarme. 1: Drive está no estado de alarme. Obs.: o número do alarme pode ser lido através do parâmetro P0048 – Alarme Atual.
Bit 8 Motor Girando	0: Motor está parado. 1: Drive está girando o motor à velocidade de referência, ou executando rampa de aceleração ou desaceleração.
Bit 9 Habilitado Geral	0: Drive está desabilitado geral. 1: Drive está habilitado geral e pronto para girar motor.
Bit 10 Sentido de Giro	0: Motor girando no sentido reverso. 1: Motor girando no sentido direto.
Bit 11 JOG	0: Função JOG inativa. 1: Função JOG ativa.
Bit 12 LOC/REM	0: Drive em modo local. 1: Drive em modo remoto.
Bit 13 Subtensão	0: Sem subtensão. 1: Com subtensão.
Bit 14 Automático (PID)	0: Em modo manual (função PID). 1: Em modo automático (função PID).
Bit 15 Em Falha	0: Drive não está no estado de falha. 1: Alguma falha registrada pelo drive. Obs.: O número da falha pode ser lido através do parâmetro P0049 – Falha Atual.

P0681 – VELOCIDADE DO MOTOR EM 13 BITS

Faixa de Valores: - 32768 a 32767 **Padrão:** -
Propriedades: RO
Grupo de acesso via HMI: NET

Descrição:
Permite monitorar a velocidade do motor. Esta palavra utiliza resolução de 13 bits com sinal para representar a frequência nominal (P0403) do motor:

- P0681 = 0000h (0 decimal) → velocidade do motor = 0
- P0681 = 2000h (8192 decimal) → velocidade do motor = frequência nominal

Valores de velocidade intermediários ou superiores podem ser obtidos utilizando esta escala. Por exemplo, 60 Hz de frequência nominal, caso o valor lido seja 2048 (0800h), para obter o valor em Hz deve-se calcular:

8192 => 60 Hz
2048 => Frequência em Hz

$$\text{Frequência em Hz} = \frac{60 \times 2048}{8192}$$

Frequência em Hz = 30 Hz

Valores negativos para este parâmetro indicam motor girando no sentido reverso de rotação.

NOTA!
Os valores transmitidos via rede apresentam uma limitação na escala utilizada, permitindo que no máximo seja indicada uma velocidade de 4 vezes a velocidade síncrona do motor, saturando em 32767 (ou -32768).

P0682 – PALAVRA DE CONTROLE VIA SERIAL

Faixa de Valores: 0000h a FFFFh **Padrão:** 0000h
Propiedades: -
Grupo de acesso via HMI: NET

Descrição:
Palavra de comando do equipamento via interface Modbus RTU. Este parâmetro somente pode ser alterado via interface serial. Para as demais fontes (HMI, etc.) ele se comporta como um parâmetro somente de leitura.

Para que os comandos escritos neste parâmetro sejam executados, é necessário que o equipamento esteja programado para ser controlado via serial. Esta programação é feita através dos parâmetros P0105 e P0220 até P0228.

Cada bit desta palavra representa um comando que pode ser executado no produto.

Bits	15 a 8	7	6	5	4	3	2	1	0
Função	Reservado	Reset de Falhas	Parada Rápida	Utiliza Segunda Rampa	LOC/REM	JOG	Sentido de Giro	Habilita Geral	Gira/Para

Tabela 3.5: *Funções dos bits para o parâmetro P0682*

Bits	Valores
Bit 0 Gira/Para	0: Para motor por rampa de desaceleração. 1: Gira motor de acordo com a rampa de aceleração até atingir o valor da referência de velocidade.
Bit 1 Habilita Geral	0: Desabilita geral o drive, interrompendo a alimentação para o motor. 1: Habilita geral o drive, permitindo a operação do motor.
Bit 2 Sentido de Giro	0: Sentido de giro do motor oposto ao da referência (sentido reverso). 1: Sentido de giro do motor igual ao da referência (sentido direto).
Bit 3 JOG	0: Desabilita a função JOG. 1: Habilita a função JOG.
Bit 4 LOC/REM	0: Drive vai para o modo local. 1: Drive vai para o modo remoto.
Bit 5 Utiliza Segunda Rampa	0: Drive utiliza como rampa de aceleração e desaceleração do motor os tempos da primeira rampa, programada nos parâmetros P0100 e P0101. 1: Drive utiliza como rampa de aceleração e desaceleração do motor os tempos da segunda rampa, programada nos parâmetros P0102 e P0103.
Bit 6 Parada Rápida	0: Não executa comando de parada rápida. 1: Executa comando de parada rápida. Obs.: quando o tipo de controle (P0202) for V/f ou VVW não se recomenda a utilização desta função.
Bit 7 Reset de Falhas	0: Sem função. 1: Se em estado de falha, executa o reset do drive.
Bits 8 a 15	Reservado.

P0683 – REFERÊNCIA DE VELOCIDADE VIA SERIAL

Faixa de Valores: -32768 a 32767 **Padrão:** 0

Propiedades: -

Grupo de acesso via HMI: NET

Descrição:

Permite programar a referência de velocidade para o motor via interface Modbus RTU. Este parâmetro somente pode ser alterado via serial. Para as demais fontes (HMI, etc.) ele se comporta como um parâmetro somente de leitura.

Para que a referência escrita neste parâmetro seja utilizada, é necessário que o produto esteja programado para utilizar a referência de velocidade via serial. Esta programação é feita através dos parâmetros P0221 e P0222.

Esta palavra utiliza resolução de 13 bits com sinal para representar a frequência nominal (P0403) do motor:

- P0683 = 0000h (0 decimal) → referência de velocidade = 0
- P0683 = 2000h (8192 decimal) → referência de velocidade = frequência nominal (P0403)

Valores de velocidade intermediários ou superiores podem ser obtidos utilizando esta escala. Por exemplo, 60 Hz de frequência nominal, caso deseje-se uma referência de 30 Hz, deve-se calcular:

60 Hz => 8192
30 Hz => Referência em 13 bits

$$\text{Referência em 13 bits} = \frac{30 \times 8192}{60}$$

Referência em 13 bits = 4096 => Valor correspondente a 30 Hz na escala em 13 bits

Este parâmetro também aceita valores negativos para inverter o sentido de rotação do motor. O sentido de rotação da referência, no entanto, depende também do valor do bit 2 da palavra de controle – P0682:

- Bit 2 = 1 e P0683 > 0: referência para o sentido direto
- Bit 2 = 1 e P0683 < 0: referência para o sentido reverso
- Bit 2 = 0 e P0683 > 0: referência para o sentido reverso
- Bit 2 = 0 e P0683 < 0: referência para o sentido direto

NOTA!
Os valores transmitidos via rede apresentam uma limitação devido à escala utilizada, permitindo que no máximo seja programado uma referência de velocidade de 4 vezes a velocidade síncrona do motor.

P0695 – VALOR PARA AS SAÍDAS DIGITAIS

Faixa de Valores: 0000h a 001Fh **Padrão:** 0000h
Propiedades: -
Grupo de acesso via HMI: NET

Descrição:
Possibilita o controle das saídas digitais através das interfaces de rede (Serial, CAN, etc.). Este parâmetro não pode ser alterado através da HMI.

Cada bit deste parâmetro corresponde ao valor desejado para uma saída digital. Para que a saída digital correspondente possa ser controlada de acordo com este conteúdo, é necessário que sua função seja programada para "Conteúdo P0695", nos parâmetros P0275 a P0279.

Bits	15 a 5	4	3	2	1	0
Função	Reservado	Valor para DO5	Valor para DO4	Valor para DO3	Valor para DO2	Valor para DO1

Tabela 3.6: *Funções dos bits para o parâmetro P0695*

Bits	Valores
Bit 0 Valor para DO1	0: saída DO1 aberta. 1: saída DO1 fechada.
Bit 1 Valor para DO2	0: saída DO2 aberta. 1: saída DO2 fechada.
Bit 2 Valor para DO3	0: saída DO3 aberta. 1: saída DO3 fechada.
Bit 3 Valor para DO4	0: saída DO4 aberta. 1: saída DO4 fechada.
Bit 4 Valor para DO5	0: saída DO5 aberta. 1: saída DO5 fechada.
Bits 5 a 15	Reservado.

NOTA!
Algumas saídas digitais podem não estar disponíveis dependendo do módulo plug-in utilizado.

P0696 – VALOR 1 PARA SAÍDAS ANALÓGICAS

P0697 – VALOR 2 PARA SAÍDAS ANALÓGICAS

P0698 – VALOR 3 PARA SAÍDAS ANALÓGICAS

Faixa de Valores:	-32768 a 32767	**Padrão:** 0
Propriedades:	-	
Grupo de acesso via HMI:	NET	

Descrição:
Possibilita o controle das saídas analógicas através das interfaces de rede (Serial, CAN, etc.). Estes parâmetros não podem ser alterados através da HMI.

O valor escrito nestes parâmetros é utilizado como valor para a saída analógica, desde que a função da saída analógica desejada seja programada para “Conteúdo P0696 / P0697 / P0698”, nos parâmetros P0251, P0254, P0257.

O valor deve ser escrito em uma escala de 15 bits (7FFFh = 32767)[3] para representar 100 % do valor desejado para a saída, ou seja:

- P0696 = 0000h (0 decimal) → valor para a saída analógica = 0 %
- P0696 = 7FFFh (32767 decimal) → valor para a saída analógica = 100 %

Neste exemplo foi mostrado o parâmetro P0696, mas a mesma escala é utilizada para o parâmetro P0697 / P0698. Por exemplo, deseja-se controlar o valor da saída analógica 1 através da serial. Neste caso deve fazer a seguinte programação:

- Escolher um dos parâmetros P0696, P0697, P0698 para ser o valor utilizado pela saída analógica 1. Neste exemplo, vamos escolher o P0696.
- Programar, na função da saída analógica 1 (P0254), a opção “Conteúdo P0696”.
- Através da interface de rede, escrever no P0696 o valor desejado para a saída analógica 1, entre 0 e 100 %, de acordo com a escala do parâmetro.

NOTA!
Para o inversor de frequência MW500, a saída analógica 3 representa a saída em frequência (FO).

[3] Para a resolução real da saída, consulte o manual do produto.

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 inversor de frequência MW500 utiliza somente o modo RTU para a transmissão de telegramas. Os bytes são de acordo com a configuração feita através do P0311.

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 inversor de frequência 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 bit/s, os tempos utilizados são os mesmos que para esta taxa. A tabela a seguir nos mostra os tempos para diferentes taxas de comunicação:

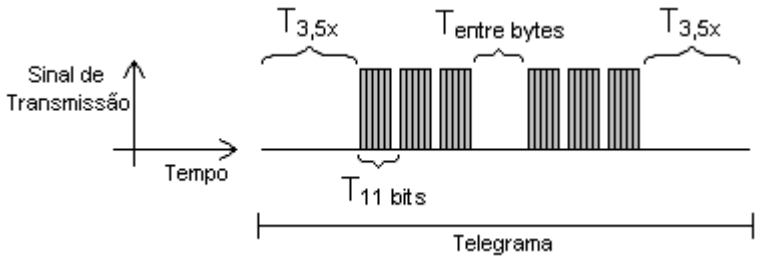


Tabela 4.1: *Taxas de comunicação e tempos envolvidos na transmissão de telegramas*

Taxa de Comunicação	T11 bits	T3,5x
1200 bit/s	9,167 ms	32,083 ms
2400 bit/s	4,583 ms	16,042 ms
4800 bit/s	2,292 ms	8,021 ms
9600 bit/s	1,146 ms	4,010 ms
19200 bit/s	573 µs	2,005 ms
38400 bit/s	573 µs	2,005 ms
57600 bit/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 inversor de frequência MW500 possui as seguintes características:

- Conexão da rede via interface serial RS485.
- Taxa de comunicação, formato dos bytes e endereçamento definidos através de parâmetros.
- Permite a parametrização e controle do inversor de frequência através do acesso a parâmetros.

5.1 FUNÇÕES DISPONÍVEIS E TEMPOS DE RESPOSTA

Na especificação do protocolo Modbus RTU são definidas funções utilizadas para acessar diferentes tipos de dados. No MW500, os parâmetros foram definidos como sendo registradores do tipo *holding*. Para acessar estes dados, foram disponibilizados os seguintes serviços (ou funções):

- Read Coils[4]
  Descrição: leitura de bloco de bits do tipo *coil*.
  Código da função: 01.

- Read Discrete Inputs[4]
  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[4]
  Descrição: leitura de bloco de registradores do tipo *input*.
  Código da função: 04.

- Write Single Coil[4]
  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[4]
  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.

[4] Funções utilizadas para acesso aos dados utilizados pela função SoftPLC.

O tempo de resposta, do final da transmissão do mestre até o início da resposta do escravo, varia de 2 a 10 ms, para qualquer uma das funções acima.

5.2 MAPA DE MEMÓRIA

A comunicação Modbus para o inversor de frequência MW500 é baseada na leitura/escrita de parâmetros do equipamento. Toda a lista de parâmetros do equipamento é disponibilizada como registradores de 16 bits do tipo holding. O endereçamento dos dados é feito com offset igual a zero, o que significa que o número do parâmetro equivale ao número do registrador. A tabela a seguir ilustra o endereçamento dos parâmetros, que podem ser acessados como registradores do tipo holding:

Tabela 5.1: *Mapa de memória para o protocolo Modbus RTU*

Parâmetro	Endereço do dado Modbus	
	Decimal	Hexadecimal
P0000	0	0000h
P0001	1	0001h
⋮	⋮	⋮
P0100	100	0064h
⋮	⋮	⋮

Para a operação do equipamento, é necessário então conhecer a lista de parâmetros do produto. Desta forma pode-se identificar quais dados são necessários para monitoração dos estados e controle das funções. Dentre os principais parâmetros pode-se citar:

Monitoração (leitura):
- P0680 (holding register 680): Palavra de estado
- P0681 (holding register 681): Velocidade do motor

Comando (escrita):
- P0682 (holding register 682): Palavra de comando
- P0683 (holding register 683): Referência de velocidade

Consulte o manual de programação para a lista completa de parâmetros do equipamento.

NOTA!
- Todos os parâmetros são tratados como registradores do tipo holding. Dependendo do mestre utilizado, estes registradores são referenciados a partir do endereço base 40000 ou 4x. Neste caso, o endereço para um parâmetro que deve ser programado no mestre é o endereço mostrado na tabela acima adicionado ao endereço base. Consulte a documentação do mestre para saber como acessar registradores do tipo holding.
- Deve-se observar que parâmetros com a propriedade somente leitura apenas podem ser lidos do equipamento, enquanto que demais parâmetros podem ser lidos e escritos através da rede.
- Além dos parâmetros, outros tipos de dados como marcadores de bit, *word* ou *float* também podem ser acessados utilizando o protocolo Modbus RTU. Estes marcadores são utilizados principalmente pela função SoftPLC disponível para o MW500. Para a descrição destes marcadores, bem como o endereço para acesso via Modbus, deve-se consultar o Manual da SoftPLC.

6 DESCRIÇÃO DETALHADA DAS FUNÇÕES

Neste item é feita uma descrição detalhada das funções disponíveis no inversor de frequência MW500 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.
- Os valores transmitidos são sempre números inteiros, independente de possuírem representação com casa decimal. Desta forma, o valor 9,5 seria transmitido como sendo 95 (5Fh) via serial. Consulte a lista de parâmetros do MW500 para obter a resolução utilizada para cada parâmetro.

6.1 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 da velocidade do motor (P0002) e corrente do motor (P0003) de escravo no endereço 1 (supondo P0002 = 30 Hz e P0003 = 1,5 A).

- Endereço: 1 = 01h (1 byte)
- Endereço do registrador inicial: 2 = 0002h (2 bytes)
- Valor do primeiro parâmetro: 30 = 001Eh (2 bytes)
- Valor do segundo parâmetro: 15 = 000Fh (2 bytes)

Pergunta (Mestre)		**Resposta (Escravo)**	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	03h	Função	03h
Registrador inicial (high)	00h	Byte Count	04h
Registrador inicial (low)	02h	P0002 (high)	00h
Quantidade de registradores (high)	00h	P0002 (low)	1Eh
Quantidade de registradores (low)	02h	P0003 (high)	00h
CRC-	65h	P0003 (low)	0Fh
CRC+	CBh	CRC-	DAh
		CRC+	31h

6.2 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 da referência de velocidade (P0683) em 30 Hz (supondo frequência nominal de 60 Hz), para o escravo no endereço 3.

- Endereço: 3 = 03h (1 byte)
- Endereço do registrador inicial: 683 = 02ABh (2 bytes)
- Valor para o parâmetro: 1000h (2 bytes)

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	03h	Endereço do escravo	03h
Função	06h	Função	06h
Registrador (high)	02h	Registrador (high)	02h
Registrador (low)	ABh	Registrador (low)	ABh
Valor (high)	10h	Valor (high)	10h
Valor (low)	00h	Valor (low)	00h
CRC-	F5h	CRC-	F5h
CRC+	B0h	CRC+	B0h

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

6.3 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 tempo de aceleração (P0100) igual a 1,0s e tempo de desaceleração (P0101) igual a 2,0s, de um escravo no endereço 15.

- Endereço: 15 = 0Fh (1 byte)
- Endereço do registrador inicial: 100 = 0064h (2 bytes)
- Valor para o primeiro parâmetro: 10 = 000Ah (2 bytes)
- Valor para o segundo parâmetro: 20 = 0014h (2 bytes)

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	0Fh	Endereço do escravo	0Fh
Função	10h	Função	10h
Registrador inicial (high)	00h	Registrador inicial (high)	00h
Registrador inicial (low)	64h	Registrador inicial (low)	64h
Quantidade de registradores (high)	00h	Quantidade de registradores (high)	00h
Quantidade de registradores (low)	02h	Quantidade de registradores (low)	02h
Byte Count	04h	CRC-	01h
P0100 (high)	00h	CRC+	39h
P0100 (low)	0Ah		
P0101 (high)	00h		
P0101 (low)	14h		
CRC-	E0h		
CRC+	91h		

6.4 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 inversor de frequência MW500, 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 (MW500), mais a faixa de corrente e tensão do inversor (ex. 'MW500 220 - 230 V 10 A').
- 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 MW500 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').

6.5 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 FALHAS E ALARMES RELACIONADOS COM A COMUNICAÇÃO MODBUS RTU

A128/F228 – TIMEOUT NA RECEPÇÃO DE TELEGRAMAS

Descrição:
Alarme que indica falha na comunicação serial. Indica que o equipamento parou de receber telegramas seriais válidos por um período maior do que o programado no P0314.

Atuação:
O parâmetro P0314 permite programar um tempo dentro do qual o inversor de frequência deverá receber ao menos um telegrama válido via interface serial RS485 – com endereço e campo de checagem de erros corretos – caso contrário será considerado que houve algum problema na comunicação serial. A contagem do tempo é iniciada após a recepção do primeiro telegrama válido. Esta função pode ser utilizada para qualquer protocolo serial suportado pelo inversor de frequência.

Depois de identificado o timeout na comunicação serial, será sinalizada através da HMI a mensagem de alarme A128 – ou falha F228, dependendo da programação feita no P0313. Para alarmes, caso a comunicação seja restabelecida, a indicação do alarme será retirada da HMI.

Possíveis Causas/Correção:
- Verificar instalação da rede, cabo rompido ou falha/mal contato nas conexões com a rede, aterramento.
- Garantir que o mestre envie telegramas para o equipamento sempre em um tempo menor que o programado no P0314.
- Desabilitar esta função no P0314.

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 Drives & Controls - Automação LTDA.
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