Manual da Comunicação Serial RS232 / RS485

Inversor de Freqüência

Série: CFW-11

Idioma: Português

Documento: 0899.5740 / 02

04/2008

Sumário

Sobre o Manual

Este manual fornece a descrição necessária para a operação do inversor de freqüência CFW-11 utilizando as interfaces seriais RS232 ou RS485. Este manual deve ser utilizado em conjunto com manual do usuário do CFW-11.

Abreviações e Definições

ASCII	American Standard Code for Information Interchange
CRC	Cycling Redundancy Check
EIA	Electronic Industries Alliance
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.

1 Introdução à Comunicação Serial

Em uma interface serial os bits de dados são enviados seqüencialmente 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 seqüência de caracteres para a transmissão e recepção de dados. Neste sentido, além da interface, é necessário identificar também o protocolo utilizado para comunicação. Dentre os diversos protocolos existentes, um protocolo muito utilizado na indústria é o protocolo Modbus-RTU.

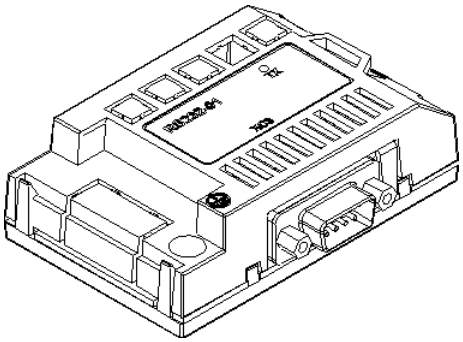
A seguir serão apresentadas características das interfaces seriais RS232 e RS485 disponíveis para o inversor CFW-11, bem como os protocolos para utilização destas interfaces.

2 Kits Acessórios

Para disponibilizar uma interface serial para o inversor de freqüência CFW-11 é necessário utilizar um dos kits para comunicação RS232 ou RS485 descritos a seguir. Informações sobre a instalação destes módulos no inversor podem ser obtidas na bula que acompanha o kit.

2.1 RS232

2.1.1 Kit RS232-01



- ☑ Item WEG: 10051958.
- ☑ Composto pelo módulo de comunicação RS232 (figura ao lado), bula de montagem e parafuso de fixação.
- ☑ Interface segue o padrão EIA RS232C.
- ☑ Permite a conexão entre o CFW-11 e o mestre da rede (ponto-a-ponto).
- ☑ Distância máxima para ligação dos dispositivos de 10 metros.

2.1.2 Pinagem do Conector

O módulo para comunicação RS232 possui um conector DB9 macho (XC8) com a seguinte pinagem:

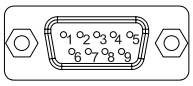


Tabela 2.1 - Pinagem do conector DB9 para RS232

Pino	Nome	Função
1	Não conectado	-
2	RX	Recepção de dados
3	TX	Transmissão de dados
4	Não conectado	-
5	GND	Referência para circuito RS232
6	Não conectado	-
7	Não conectado	-
8	Não conectado	-
9	Não conectado	-

2.1.3 Indicações e Chaves

☑ LED TX: LED para indicação de transmissão de dados pelo inversor, na cor verde.

2.1.4 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.

2.1.5 Cabos para Ligação em RS232

Caso seja desejado, a WEG pode fornecer os seguintes cabos para ligação em RS232 entre o inversor CFW-11 e um mestre da rede, como um PC:

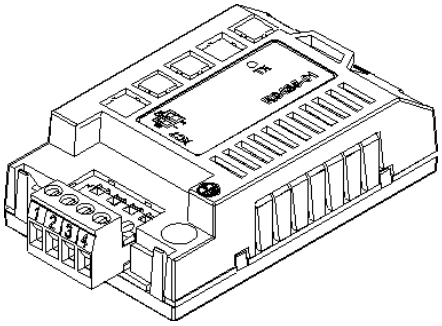
Cabo	Item WEG
Cabo RS232 blindado com conectores DB9 fêmea Comprimento: 3 metros	10050328
Cabo RS232 blindado com conectores DB9 fêmea Comprimento: 10 metros	10191117

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

2.2 RS485

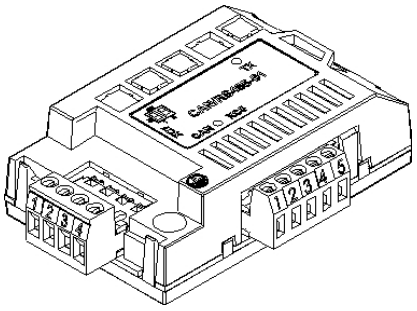
O CFW-11 possui duas opções para utilizar a interface RS485, descritas a seguir.

2.2.1 Kit RS485-01



- ☑ Item WEG: 10051957.
- ☑ Composto pelo módulo de comunicação RS485 (figura ao lado), bula de montagem e parafuso de fixação.
- ☑ Interface segue o padrão EIA-485.
- ☑ Interface isolada galvanicamente e com sinal diferencial, conferindo maior robustez contra interferência eletromagnética.
- ☑ Permite a conexão de até 32 dispositivos no mesmo segmento. Uma quantidade maior de dispositivos pode ser conectada com o uso de repetidores.[1]
- ☑ Comprimento máximo do barramento de 1000 metros.

2.2.2 Kit CAN/RS485-01



- ☑ Item WEG: 10051960.
- ☑ Composto pelo módulo de comunicação CAN/RS485-01 (figura ao lado), bula de montagem e parafuso de fixação.
- ☑ Possuem as mesmas características da interface RS485-01, mais uma interface CAN, para aplicações onde seja necessária a operação em conjunto de ambas as interfaces.

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

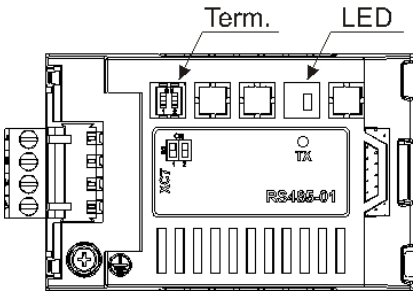
2.2.3 Pinagem do Conector

O módulo para comunicação RS485 possui um conector *plug-in* de 4 vias (XC7) com a seguinte pinagem:

Tabela 2.2 - Pinagem do conector de 4 vias para RS485

Pino	Nome	Função
1	B-Line (+)	RxD/TxD positivo
2	A-Line (-)	RxD/TxD negativo
3	GND	0V isolado do circuito RS485
4	Ground	Terra (blindagem)

2.2.4 Indicações e Chaves



- ☑ LED TX: LED para indicação de transmissão de dados pelo inversor, na cor verde.
- ☑ Resistor de terminação (S1): chave para habilitar o resistor de terminação necessário para a interface RS485. Este resistor deve ser habilitado (posição *ON*) somente nos dois dispositivos localizados nos extremos do barramento principal.

2.2.5 Conexão com a Rede RS485

Para a ligação do inversor 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.

2.3 Anybus-CC

As interfaces RS232 e RS485 também podem ser disponibilizadas utilizando os kits Anybus-CC passivos para RS232 ou RS485. Consulte o Manual da Comunicação Anybus-CC para obter informações sobre estes kits.

3 Parametrização do Inversor

A seguir serão apresentados apenas os parâmetros do inversor de freqüência CFW-11 que possuem relação com a comunicação serial.

3.1 Símbolos para Descrição das Propriedades

RO	Parâmetro somente de leitura
CFG	Parâmetro somente alterado com o motor parado
Net	Parâmetro visível através da HMI se o inversor possuir interface de rede instalada – RS232, RS485, CAN, Anybus-CC, Profibus – ou se a interface USB for conectada
Serial	Parâmetro visível através da HMI se o inversor possuir interface RS232 ou RS485 instalada
USB	Parâmetro visível através da HMI se a interface USB do inversor for conectada

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/Pára Local

P0225 – Seleção Jog Local

P0226 – Seleção Giro Remoto

P0227 – Seleção Gira/Pára Remoto

P0228 – Seleção Jog Remoto

Estes parâmetros são utilizados na configuração da fonte de comandos para os modos local e remoto do inversor CFW-11. Para que o inversor seja controlado através da interface Serial, deve-se selecionar uma das opções 'Serial/USB' disponíveis nos parâmetros.

A descrição detalhada deste parâmetros encontra-se no Manual de Programação do CFW-11.

P0308 – Endereço Serial

Faixa de Valores:	1 a 247	Padrão: 1
Propriedades:	CFG, Serial	
Grupos de acesso via HMI:		

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 113 Serial RS232 / 485

Descrição:
Permite programar o endereço utilizado para comunicação serial do inversor. 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 = 1 (TP) → endereços válidos: 1 a 30.
☑ P0312 = 2 (Modbus-RTU) → endereços válidos: 1 a 247.

P0310 – Taxa de Comunicação Serial

Faixa de Valores: 0 = 9600 bits/s
1 = 19200 bits/s
2 = 38400 bits/s
3 = 57600 bits/s

Padrão: 0

Propriedades: CFG, Serial

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 113 Serial RS232 / 485

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.

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: 0

Propriedades: CFG, Serial

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 113 Serial RS232 / 485

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.

P0312 – Protocolo Serial

Faixa de Valores: 1 = TP
2 = Modbus-RTU

Padrão: 2

Propriedades: CFG, Serial

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 113 Serial RS232 / 485

Descrição:
Permite selecionar o protocolo desejado para a interface serial. A descrição detalhada dos protocolos é feita nos itens seguintes deste manual.

P0313 – Ação para Erro de Comunicação

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

Padrão: 0

Propriedades: CFG, Net

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 111 Estados / Comandos

Descrição:
Este parâmetro permite selecionar qual a ação deve ver executada pelo inversor, caso um erro de comunicação seja detectado.

Tabela 3 .1 - Valores para o parâmetro P0313

Opções	Descrição
0 = Inativo	Nenhuma ação é tomada, inversor permanece no estado atual.
1 = Pára por Rampa	O comando de parada por rampa é executado, e o motor pára de acordo com a rampa de desaceleração programada.
2 = Desabilita Geral	O inversor é desabilitado geral, e o motor pára por inércia.
3 = Vai para Local	O inversor é comandado para o modo local.
4 = Vai para Local e mantém comandos e referência	O inversor é comandado para o modo local, mas os comandos de habilitação e a referência de velocidade recebidos via rede são mantidos em modo local, caso o inversor seja programado para utilizar em modo local comandos via HMI ou 3 wire start stop, 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 inversor, sendo necessário fazer o reset de falhas do inversor para o retorno da sua operação normal.

Para a interface serial, é considerado erro de comunicação apenas o evento de *timeout* da interface serial – alarme A128/falha F228. Este *timeout* é programado através do parâmetro P0314.

As ações descritas neste parâmetro são executadas através da escrita automática dos respectivos bits no parâmetro de controle via serial / USB – P0682. Para que a ação executada tenha efeito, é necessário que o inversor esteja programado para ser controlado via serial. Esta programação é feita através dos parâmetros P0220 até P0228.

P0314 – Watchdog Serial

Faixa de Valores: 0,0 a 999,0s

Padrão: 0,0s

Propriedades: CFG, Serial

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 111 Estados / Comandos

Descrição:
Permite programar um tempo para a detecção de erro de comunicação via interface serial. Caso o inversor 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.

Após energizado, o inversor 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

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 111 Estados / Comandos

Descrição:
Permite identificar se o cartão de interface serial RS232 ou RS485 está devidamente instalado, e se a comunicação serial apresenta erros.

Tabela 3.2 - Valores do parâmetro P0316

Opções	Descrição
0 = Inativo	Interface seria inativa. Ocorre quando o inversor não possui cartão de interface RS232/ RS485 instalado.
1 = Ativo	Cartão de interface RS232/ 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 - FFFh

Padrão: -

Propriedades: RO

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 111 Estados / Comandos

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4 a 0
Função	Em Falha	Manual/ Automático	Subtensão	LOC/REM	JOG	Sentido de Giro	Habilitado Geral	Rampa Habilitada	Em Alarme	Em modo de configuração	Segunda Rampa	Reservado

Tabela 3.3 - Funções dos bits para o parâmetro P0680

Bits	Valores
Bits 0 a 4	Reservado.
Bit 5 Segunda Rampa	0: Inversor está configurado para utilizar como rampa de aceleração e desaceleração para o motor a primeira rampa, programada nos parâmetros P0100 e P0101. 1: Inversor 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: Inversor operando normalmente. 1: Inversor em modo de configuração. Indica uma condição especial na qual o inversor 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 inversor. Obs.: É possível obter a descrição exata do modo especial de operação no parâmetro P0692.
Bit 7 Em Alarme	0: Inversor não está no estado de alarme. 1: Inversor está no estado de alarme. Obs.: o número do alarme pode ser lido através do parâmetro P0048 – Alarme Atual.
Bit 8 Rampa Habilitada (RUN)	0: Motor está parado. 1: Inversor está girando o motor à velocidade de referência, ou executando rampa de aceleração ou desaceleração.
Bit 9 Habilitado Geral	0: Inversor está desabilitado geral. 1: Inversor está habilitado geral e pronto para girar motor.
Bit 10 Sentido de Giro	0: Motor girando no sentido anti-horário. 1: Motor girando no sentido horário.
Bit 11 JOG	0: Função JOG inativa. 1: Função JOG ativa.
Bit 12 LOC/REM	0: Inversor em modo local. 1: Inversor em modo remoto.
Bit 13 Subtensão	0: Sem subtensão. 1: Com subtensão.
Bit 14 Manual/ Automático	0: Em modo manual (função PID). 1: Em modo automático (função PID).
Bit 15 Em Falha	0: Inversor não está no estado de falha. 1: Alguma falha registrada pelo inversor. 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 - 32768 **Padrão:** -

Propriedades: RO

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
- 49 Comunicação
  - 111 Estados / Comandos

Descrição:
Permite monitorar a velocidade do motor. Esta palavra utiliza resolução de 13 bits com sinal para representar a rotação síncrona do motor:

☑ P0681 = 0000h (0 decimal) → velocidade do motor = 0 rpm
☑ P0681 = 2000h (8192 decimal) → velocidade do motor = rotação síncrona

Valores de velocidade em rpm intermediários ou superiores podem ser obtidos utilizando esta escala. Por exemplo, para um motor de 4 pólos e 1800 rpm de rotação síncrona, caso o valor lido seja 2048 (0800h), para obter o valor em rpm deve-se calcular:

8192 – 1800 rpm
2048 – velocidade em rpm

$$\text{velocidade em rpm} = \frac{1800 \times 2048}{8192}$$

Velocidade em rpm = 450 rpm

Valores negativos para este parâmetro indicam motor girando no sentido anti-horário.

P0682 – Palavra de Controle via Serial / USB

Faixa de Valores: 0000h - FFFFh **Padrão:** 0000h

Propriedades: Serial e USB

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 111 Estados / Comandos

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

Para que os comandos escritos neste parâmetro sejam executados, é necessário que o inversor 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 inversor.

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

Tabela 3.4 - Funções dos bits para o parâmetro P0682

Bits	Valores
Bit 0 Gira/Pára	0: Pára 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 inversor, interrompendo a alimentação para o motor. 1: Habilita geral o inversor, permitindo a operação do motor.
Bit 2 Sentido de Giro	0: Girar motor no sentido oposto ao da referência. 1: Girar motor no sentido indicado na referência.
Bit 3 JOG	0: Desabilita a função JOG. 1: Habilita a função JOG.
Bit 4 LOC/REM	0: Inversor vai para o modo local. 1: Inversor vai para o modo remoto.
Bit 5 Utiliza Segunda Rampa	0: Inversor utiliza como rampa de aceleração e desaceleração do motor os tempos da primeira rampa, programada nos parâmetros P0100 e P0101 1: Inversor utiliza como rampa de aceleração e desaceleração do motor os tempos da segunda rampa, programada nos parâmetros P0102 e P0103
Bit 6	Reservado
Bit 7 Reset de Falhas	0: Sem função. 1: Se em estado de falha, executa o reset do inversor.
Bits 8 a 15	Reservado.

P0683 – Referência de Velocidade via Serial/ USB

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

Propriedades: Serial e USB

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 111 Estados / Comandos

Descrição:
Permite programar a referência de velocidade para o inversor via interface serial. Este parâmetro somente pode ser alterado via interface serial ou USB. Para as demais fontes (HMI, CAN, etc.) ele se comporta como um parâmetro somente de leitura.

Para que a referência escrita neste parâmetro seja utilizada, é necessário que o inversor 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 rotação síncrona do motor:

☑ P0683 = 0000h (0 decimal) → referência de velocidade = 0 rpm
☑ P0683 = 2000h (8192 decimal) → referência de velocidade = rotação síncrona

Valores de referência intermediários ou superiores podem ser programados utilizando esta escala. Por exemplo, para um motor de 4 pólos e 1800 rpm de rotação síncrona, caso deseje-se uma referência de 900 rpm, deve-se calcular:

1800 rpm – 8192
900 rpm – referência em 13 bit

$$\text{referência em 13 bit} = \frac{900 \times 8192}{1800}$$

Referência em 13 bit = 4096 (valor correspondente a 900 rpm na escala em 13 bits)

Este parâmetro também aceita valore 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 horário
☑ Bit 2 = 1 e P0683 < 0: referência para o sentido anti-horário
☑ Bit 2 = 0 e P0683 > 0: referência para o sentido anti-horário
☑ Bit 2 = 0 e P0683 < 0: referência para o sentido horário

P0695 – Valor para as Saídas Digitais

Faixa de Valores: 0000h - FFFFh **Padrão:** 0000h

Propriedades: Net

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
∟ 49 Comunicação
∟ 111 Estados / Comandos

Descrição:
Possibilita o controle das saídas digitais através das interfaces de rede (Serial, USB, 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 P0280.

Bits	15 a 5	4	3	2	1	0
Função	Reservado	Valor para DO5	Valor para DO4	Valor para DO3 (RL3)	Valor para DO2 (RL2)	Valor para DO1 (RL1)

Tabela 3.5 - Funções dos bits para o parâmetro P0695

Bits	Valores
Bit 0 Valor para DO1 (RL1)	0: saída DO1 aberta. 1: saída DO1 fechada.
Bit 1 Valor para DO2 (RL2)	0: saída DO2 aberta. 1: saída DO2 fechada.
Bit 2 Valor para DO3 (RL3)	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.

P0696 – Valor 1 para Saídas Analógicas

P0697 – Valor 2 para Saídas Analógicas

P0698 – Valor 3 para Saídas Analógicas

P0699 – Valor 4 para Saídas Analógicas

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

Propriedades: Net

Grupos de acesso via HMI:

01 GRUPOS PARÂMETROS
- 49 Comunicação
  - 111 Estados / Comandos

Descrição:
Possibilita o controle das saídas analógicas através das interfaces de rede (Serial, USB, CAN, etc.). Este parâmetro não pode ser alterado 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/ P0699", nos parâmetros P0251, P0254, P0257 ou P0260.

O valor deve ser escrito em uma escala de 15 bits (7FFFh = 32767)[2] 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 os parâmetros P0697/P0698/P0699. 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 a P0699 para ser o valor utilizado pela saída analógica 1. Neste exemplo, vamos escolher o P0696.

[2] Para a resolução real da saída, consulte o manual do CFW-11.

- ☑ Programar, na função da saída analógica 1 (P0254), a opção "Conteúdo P0696".
- ☑ Através da interface serial, escrever no P0696 o valor desejado para a saída analógica 1, entre 0 e 100 %, de acordo com a escala do parâmetro.

NOTA!
Caso a saída analógica seja programada para operar de -10V até 10V, valores negativos para estes parâmetros devem ser utilizados para comandar a saída com valores negativos de tensão, ou seja, -32768 até 32767 representa uma variação de -10V até 10V na saída analógica.

4 Protocolo TP

O protocolo TP foi desenvolvido com o objetivo de possibilitar a comunicação com CLPs da linha TP. Mas devido a sua flexibilidade e facilidade de uso, tem sido utilizado em outras aplicações, sendo muitas vezes implementado em CLPs e outros sistemas para controle e monitoração dos equipamentos WEG.

4.1 Campos do Protocolo

- ☑ **STX**: Byte de "Start of Transmition": Valor: 02h; 2 decimal.
- ☑ **ETX**: Byte de "End of Transmition": Valor: 03h; 3 decimal.
- ☑ **ADR**: Byte do endereço do inversor na rede, programável através do P0308.
  Faixa de Valores: 41h; 65 decimal; 'A' (ASCII) até 5Eh; 94 decimal; '^' (ASCII), representando os endereços de 1 ... 30 na rede.
  Especial 1: 40h; 64 decimal; '@' (ASCII) → Permite escrita ou leitura de todos os equipamentos conectados a rede.
  Especial 2: 5Fh; 95 decimal; '_' (ASCII) → Permite SOMENTE escrita em todos os equipamentos conectados a rede sem resposta de aceitação ou rejeição.

- ☑ **COD**: Byte do Código do Telegrama
  Leitura: 3Ch (hexadecimal); 60 (decimal); '<' (ASCII)
  Escrita: 3Dh (hexadecimal); 61 (decimal); '=' (ASCII) sem salvar o parâmetro na EEPROM
  Escrita: 3Eh (hexadecimal); 62 (decimal); '>' (ASCII) salvando o parâmetro na EEPROM

- ☑ **BCC**: Byte de Checksum longitudinal do telegrama, OU EXCLUSIVO (XOR) entre todos os bytes do telegrama. Tamanho de 1 byte (00h ... FFh hexadecimal)

- ☑ **DMW**: "Data Master Write". São 4 bytes de escrita que o mestre envia ao escravo, sendo que os 2 primeiros representam o parâmetro e os 2 últimos o valor a ser escrito neste parâmetro.
  PHi: Byte representando a parte alta do parâmetro
  PLo: Byte representando a parte baixa do parâmetro
  VHi: Byte representando a parte alta do valor a ser escrito
  VLo: Byte representando a parte baixa do valor a ser escrito
  Exemplo: Escrever 1FFFh na referência de velocidade (P0683) → PHi = 02h, PLo = ABh, VHi = 1Fh, VLo = FFh.

- ☑ **DMR**: "Data Master Read". São 2 bytes de leitura que o mestre envia ao escravo que representam o parâmetro a ser lido.
  PHi: Byte representando a parte alta do parâmetro
  PLo: Byte representando a parte baixa do parâmetro
  Exemplo: Ler o valor contido no parâmetro de tensão de saída (P0007) → PHi = 00h,
  PLo = 07h.

- ☑ **NUM**: Byte que representa o número de DMW ou DMR a serem transmitidos, conforme o COD do telegrama. Faixa de valores: 1 ... 6 (decimal).

- ☑ **DSV**: "Data Slave Value". São 2 bytes que o escravo envia ao mestre após uma solicitação de um telegrama de leitura do mestre, representando o valor contido no parâmetro solicitado.
  VHi: Byte representando a parte alta do valor a ser escrito
  VLo: Byte representando a parte baixa do valor a ser escrito
  Exemplo: Resposta a solicitação de leitura do parâmetro de estado do inversor (P0680) → VHi = 13h, VLo = 00h.

- ☑ **ACK**: Byte de aceitação do escravo após uma escrita do mestre. Valor: 06h; 6 decimal;

- ☑ **NAK**: Byte de rejeição do escravo após uma leitura ou escrita do mestre. Pode ocorrer quando o mestre solicita uma escrita ou leitura de um parâmetro inexistente ou o valor a ser escrito no parâmetro está fora da faixa de valores permitida.
  Valor: 15h; 21 decimal.

4.2 Formato dos Telegramas

A seguir serão apresentados os formatos dos telegramas de leitura e escrita em parâmetros. É importante observar que cada telegrama no protocolo TP permite realizar a leitura ou escrita de até 6 parâmetros por vez. Telegramas que possuem erro no formato ou BCC incorreto serão ignorados pelo inversor, que não enviará resposta para o mestre.

4.2.1 Telegrama de Leitura

Mestre:

STX	ADR	COD	NUM	DMR	...	DMR	ETX	BCC

- ☑ COD: código para leitura → 3Ch (hexadecimal); 60 (decimal); '<' (ASCII)
- ☑ NUM: número de parâmetros lidos. Faixa de 1 ... 6.
- ☑ DMR: número do parâmetro solicitado. O número de DMRs deve ser igual ao valor configurado no byte NUM.

Escravo (CFW-11):

ADR	DSV	...	DSV	BCC

ou

ADR	NAK

- ☑ DSV: valor do parâmetro solicitado. O número de DSVs é igual ao valor configurado no byte NUM

Lembrando que:

DMR	
PHi	PLo

DSV	
VHi	VLo

4.2.2 Telegrama de Escrita

Mestre:

STX	ADR	COD	NUM	DMW	...	DMW	ETX	BCC

- ☑ COD: código para escrita
  - 3Eh (hexadecimal); 62 (decimal); '>' (ASCII) → salvando na EEPROM
  - 3Dh (hexadecimal); 61 (decimal); '=' (ASCII) → sem salvar na EEPROM
- ☑ NUM: número de parâmetros escritos. Faixa de 1 ... 6.
- ☑ DMW: número e conteúdo para o parâmetro. O número de DMWs deve ser igual ao valor configurado no byte NUM.

Escravo (CFW-11):

ADR	ACK

ou

ADR	NAK

Lembrando que:

DMW			
PHi	PLo	VHi	VLo

4.3 Exemplo de Telegramas Utilizando o Protocolo TP

Todos exemplos a seguir consideram que o inversor está programado com o endereço 1 (P0308=1), logo o campo ADR é enviado com valor 41h (consulte a Tabela 1.7).

Exemplo 1: leitura de dois parâmetros do inversor:

☑ Velocidade do motor – P0002 (supondo P0002 em 1200rpm = 04B0h).
☑ Corrente do motor – P0003 (supondo P0003 em 5,0A = 0032h).

Mestre:

02h	41h	3Ch	02h	00h	02h	00h	03h	03h	7Fh
STX	ADR	COD	NUM	DMR:P0002		DMR:P0003		ETX	BCC
				Parâmetro		Parâmetro			

Escravo (CFW-11):

41h	04h	B0h	00h	32h	C7h
ADR	DSV:1200		DSV:50		BCC
	Valor		Valor		

Exemplo 2: Programar 6 parâmetros para a operação do inversor:

☑ Telegrama de escrita salvando na EEPROM.
☑ P0100 = 50 (100 em decimal = 0064h, 50 em decimal = 0032h)
☑ P0101 = 150 (101 em decimal = 0065h, 150 em decimal = 0096h)
☑ P0220 = 6 (220 em decimal = 00DChh, 6 em decimal = 0006h)
☑ P0222 = 9 (222 em decimal = 00DEh, 9 em decimal = 0009h)
☑ P0226 = 5 (226 em decimal = 00E2h, 5 em decimal = 0005h)
☑ P0227 = 2 (227 em decimal = 00E3h, 2 em decimal = 0002h)

Mestre (requisição):

02h	41h	3Eh	06h	00h	64h	00h	32h	00h	65h	00h	96h
STX	ADR	COD	NUM	DMW:P0100 = 50				DMW:P0101 = 150			
				Parâmetro		Valor		Parâmetro		Valor	

00h	DCh	00h	06h	00h	DEh	00h	09h	00h	E2h	00h	05h
DMW:P0220 = 6				DMW:P0222 = 9				DMW:P0226 = 5			
Parâmetro		Valor		Parâmetro		Valor		Parâmetro		Valor	

00h	E3h	00h	02h	03h	D6h
DMW:P0227 = 2				ETX	BCC
Parâmetro		Valor			

Escravo (resposta):

41h	06h
ADR	ACK

Exemplo 4: escrita do comando de habilitar e da referência de velocidade via serial:

☑ Telegrama de escrita sem salvar na EEPROM.
☑ P0682 = 0013h – controle via serial com comandos de alteração para modo remoto, habilita geral e habilita rampa (682 em decimal = 02AAh).
☑ P0683 = 1000h – referência de velocidade via serial programada para metade da velocidade síncrona do motor (683 em decimal = 02ABh).

Mestre (requisição):

02h	41h	3Dh	02h	02h	AAh	00h	13h	02h	ABh	10h	00h	03h	7Dh
STX	ADR	COD	NUM	DMW:0682 = 0013h				DMW:P0683 = 1000h				ETX	BCC
				Parâmetro		Valor		Parâmetro		Valor			

Escravo (resposta):

41h	06h
ADR	ACK

5 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. A comunicação Modbus-RTU do inversor CFW-11 foi desenvolvida com base nos seguintes documentos:

- ☑ MODBUS Protocol Reference Guide Rev. J, MODICON, June 1996.
- ☑ MODBUS Application Protocol Specification, MODBUS.ORG, May 8th 2002.
- ☑ MODBUS over Serial Line, MODBUS.ORG, December 2nd 2002.

Nestes documentos estão definidos os formatos das mensagens utilizados pelos elementos que fazem parte da rede Modbus, os serviços (ou funções) que podem ser disponibilizados via rede, e também como estes elementos trocam dados na rede.

5.1 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 freqüência CFW-11 utiliza somente o modo RTU para a transmissão de telegramas. Os bytes são transmitidos no formato hexadecimal, e sua configuração depende da programação feita através do P0311.

5.2 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)

5.2.1 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.

5.2.2 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.

No CFW-11, os dados relativos aos parâmetros estão disponibilizados como registradores do tipo *holding* (referenciados a partir do endereço 40000 ou '4*x*').

5.2.3 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).

5.2.4 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 nos apêndices B e C.

5.2.5 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[3]). 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 irá descartar os bytes já recebidos e montará um novo telegrama com os bytes que estiverem sendo transmitidos.

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

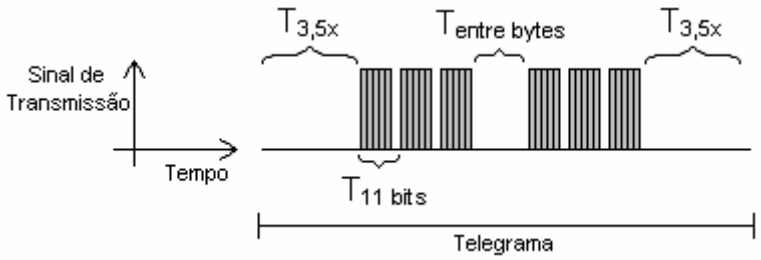


Tabela 5.1 - Taxas de comunicação e tempos envolvidos na transmissão de telegramas

Taxa de Comunicação	$T_{11\ bits}$	$T_{3,5x}$
9600 bits/s	1,146 ms	4,010 ms
19200 bits/s	573 µs	2,005 ms
38400 bits/s	573 µs	2,005 ms
57600 bits/s	573 µs	2,005 ms

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

[3] Sempre é considerado o tempo de 11 bits como o tempo para transmissão de um byte, mesmo que no parâmetro P0311 seja programado um formato de telegrama onde cada byte possua apenas 10 bits.

5.3 Operação do CFW-11 na Rede Modbus-RTU

O CFW-11 possui as seguintes características quando operado em rede Modbus-RTU:

☑ Conexão da rede via interface serial RS-232 ou RS-485 (ver item 2).
☑ Endereçamento, taxa de comunicação e formato dos bytes definidos através de parâmetros (ver item 3).
☑ Permite a parametrização e controle do inversor através do acesso a parâmetros.

5.3.1 Funções Disponíveis e Tempos de Resposta

Na especificação do protocolo Modbus-RTU são definidas funções utilizadas para acessar diferentes tipos de registradores. No CFW-11, os parâmetros foram definidos como sendo registradores do tipo *holding*. Para acessar estes registradores, foram disponibilizados os seguintes serviços (ou funções):

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

☑ *Read Discrete Inputs*[4]
Descrição: leitura de bloco 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 bit 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 drive.
Código da função: 43.

O tempo de resposta do inversor CFW-11, do final na transmissão do mestre até o início da resposta do escravo, varia de 2 a 10 ms, para qualquer uma das funções acima.

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

5.3.2 Endereçamento dos Dados e Offset

O endereçamento dos dados no CFW-11 é feito com offset igual a zero, o que significa que o número do endereço equivale ao número dado. Os parâmetros são disponibilizados a partir do endereço 0 (zero). A tabela a seguir ilustra o endereçamento dos parâmetros, que podem ser acessados como registradores do tipo *holding*:

Tabela 5.2 - Endereço dos dados para a interface Modbus-RTU

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

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*.

☑ Além dos parâmetros, outros tipos de dados como marcadores de bit, *word* ou *float* também podem ser acessados utilizando a interface Modbus-RTU. Estes marcadores são utilizados principalmente pela função SoftPLC disponível para o CFW-11. Para a descrição destes marcadores, bem como o endereço para acesso via Modbus, deve-se consultar o Manual da SoftPLC.

5.4 Descrição Detalhada das Funções

Neste item é feita uma descrição detalhada das funções disponíveis no CFW-11 para comunicação Modbus-RTU. Para a elaboração dos telegramas, é importante observar o seguinte:

☑ Os valores são sempre transmitidos 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 parâmetro do CFW-11 para obter a resolução utilizada para cada parâmetro.

5.4.1 Função 03 – Read Holding Register

Lê o conteúdo de um grupo de registradores, que necessariamente devem estar em seqüência numérica. Esta função possui a seguinte estrutura para os telegramas de leitura e resposta (os valores são sempre hexadecimal, e 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)
Número de registradores (byte high)	Dado 1 (low)
Número de registradores (byte low)	Dado 2 (high)
CRC-	Dado 2 (low)
CRC+	etc...
	CRC-
	CRC+

Exemplo 1: leitura da velocidade do motor (P0002) e corrente do motor (P0003) do CFW-11 no endereço 1 (supondo P0002 = 1000 rpm e P0003 = 3,5 A).

☑ Endereço: 1 = 01h (1 byte)
☑ Número do primeiro parâmetro: 2 = 0002h (2 bytes)
☑ Valor do primeiro parâmetro: 1000 = 03E8h (2 bytes)
☑ Valor do segundo parâmetro: 35 = 0023h (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	P002 (high)	03h
No. de registradores (high)	00h	P002 (low)	E8h
No. de registradores (low)	02h	P003 (high)	00h
CRC-	65h	P003 (low)	23h
CRC+	CBh	CRC-	3Bh
		CRC+	9Ah

5.4.2 Função 06 – Write Single Register

Esta função é utilizada para escrever um valor para um único registrador. Possui a seguinte estrutura (os valores são sempre hexadecimal, e 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 2: escrita da referência de velocidade (P0683) em 900 rpm (supondo velocidade síncrona de 1800 rpm), para o CFW-11 no endereço 3.

☑ Endereço: 3 = 03h (1 byte)
☑ Número do parâmetro: 683 = 02AB (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.

5.4.3 Função 16 – Write Multiple Registers

Esta função permite escrever valores para um grupo de registradores, que devem estar em seqüência numérica. Também pode ser usada para escrever um único registrador (os valores são sempre hexadecimal, e 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)
Número de registradores (byte high)	Número de registradores (byte high)
Número de registradores (byte low)	Número 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 3: escrita do tempo de aceleração (P0100) igual a 1,0s e tempo de desaceleração (P0101) igual a 2,0s, de um CFW-11 no endereço 15.

☑ Valores convertidos para hexadecimal:
- Endereço: 15 = 0Fh (1 byte)
- Número do primeiro parâmetro: 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 (high)	00h
Registrador inicial (low)	64h	Registrador (low)	64h
No. de registradores (high)	00h	Valor (high)	00h
No. de registradores (low)	02h	Valor (low)	02h
Byte Count	04h	CRC-	01h
P100 (high)	00h	CRC+	39h
P100 (low)	0Ah		
P101 (high)	00h		
P101 (low)	14h		
CRC-	E0h		
CRC+	91h		

5.4.4 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 seqüência de caracteres ASCII. Para o CFW-11, apenas informações básicas estão disponíveis, formadas por três objetos:

- ☑ Objeto 00h – VendorName: Sempre 'WEG'.
- ☑ Objeto 01h – ProductCode: Formado pelo código do produto (CFW-11) mais a tensão e corrente nominal do inversor (ex. 'CFW-11 220 - 230 V 10A / 8A').
- ☑ Objeto 02h – MajorMinorRevision: indica a versão de firmware do drive, no formato 'VX.XX'.

O código de leitura indica quais as categorias de informações são lidas, e se os objetos são acessados em seqüência ou individualmente. No caso, o CFW-11 suporta os códigos 01 (informações básicas em seqüência), e 04 (acesso individual aos objetos). Os demais campos são especificados pelo protocolo e para o CFW-11 possuem valores fixos.

Exemplo 4: leitura das informações básicas em seqüência, a partir do objeto 01h, de um CFW-11 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	01h	Conformity Level	81h
CRC-	70h	More Follows	00h
CRC+	77h	Próximo Objeto	00h
		Número de objetos	02h
		Código do Objeto	01h
		Tamanho do Objeto	1Bh
		Valor do Objeto	'CFW-11 220 - 230 V 10A / 8A'
		Código do Objeto	02h
		Tamanho do Objeto	05h
		Valor do Objeto	'V4.50'
		CRC-	B2h
		CRC+	8Fh

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 'V4.50' foi transmitido como sendo cinco caracteres ASCII, que em hexadecimal possuem os valores 56h ('V'), 34h ('4'), 2Eh ('.'), 35h ('5') e 30h ('0').

5.4.5 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 CFW-11 poderá ou não enviar resposta para o mestre. Quando o mestre envia uma mensagem para um drive configurado em um determinado endereço da rede, o inversor não irá responder ao mestre caso ocorra:

- ☑ 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 inversor 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 (parâmetro) 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 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 5: mestre solicita para o escravo no endereço 1 a escrita no parâmetro 99 (parâmetro inexistente):

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	06h	Função	86h
Registrador (high)	00h	Código de erro	02h
Registrador (low)	63h	CRC-	C3h
Valor (high)	00h	CRC+	A1h
Valor (low)	00h		
CRC-	79h		
CRC+	D4h		

6 Falhas e Alarmes Relacionados com a Comunicação Serial

A128/F228 – Timeout na Recepção de Telegramas

Descrição:
Único alarme/falha relacionado com a comunicação serial. Indica que o inversor 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 deverá receber ao menos um telegrama válido via interface serial RS232 / 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.

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 e novos telegramas válidos sejam recebidos, a indicação do alarme será retirada da HMI.

Possíveis Causas/Correção:
- ☑ Verificar fatores que possam provocar falhas na comunicação (cabos, instalação, aterramento).
- ☑ Garantir que o mestre envie telegramas para o inversor 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. O algoritmo foi obtido e é explicado nos documentos referenciados no item 5.

```
/* 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);
}
```

Apêndice C. Cálculo do CRC Utilizando Deslocamento de Registradores

Neste item é descrito o algoritmo para o cálculo do CRC utilizado na comunicação Modbus-RTU, através do deslocamento de registradores. O algoritmo foi obtido e é explicado nos documentos referenciados no item 5.

O cálculo do CRC é iniciado primeiramente carregando-se uma variável de 16 bits (referenciado a partir de agora como variável CRC) com o valor FFFFh. Depois se executa os passos de acordo com a seguinte rotina:

1. Submete-se o primeiro byte da mensagem (somente os bits de dados - start bit , paridade e stop bit não são utilizados) a uma lógica XOR (OU exclusivo) com os 8 bits menos significativos da variável CRC, retornando o resultado na própria variável CRC.
2. Então, a variável CRC é deslocada uma posição à direita, em direção ao bit menos significativo, e a posição do bit mais significativo é preenchida com 0 (zero).
3. Após este deslocamento, o bit de *flag* (bit que foi deslocado para fora da variável CRC) é analisado, ocorrendo o seguinte:
   - ☑ Se o valor do bit for 0 (zero), nada é feito.
   - ☑ Se o valor do bit for 1, o conteúdo da variável CRC é submetido a uma lógica XOR com um valor constante de A001h e o resultado é retornado à variável CRC.
4. Repetem-se os passos 2 e 3 até que oito deslocamentos tenham sido feitos.
5. Repetem-se os passos de 1 a 4, utilizando o próximo byte da mensagem, até que toda a mensagem tenha sido processada.

O conteúdo final da variável CRC é o valor do campo CRC que é transmitido no final do telegrama. A parte menos significativa é transmitida primeiro (CRC-) e em seguida a parte mais significativa (CRC+).