Modbus RTU

SCA06

Manual do Usuário

Manual do Usuário Modbus RTU

Série: SCA06

Idioma: Português

N ° do Documento: 10001625768 / 01

Data da Publicação: 12/2012

SUMÁRIO

SUMÁRIO 3

SOBRE O MANUAL 5

ABREVIAÇÕES E DEFINIÇÕES 5
REPRESENTAÇÃO NUMÉRICA 5
DOCUMENTOS 5

1 INTRODUÇÃO À COMUNICAÇÃO SERIAL 6

2 DESCRIÇÃO DAS INTERFACES 7

2.1 MÓDULO DE EXPANÇÃO E COMUNICAÇÃO RS232 E RS485 ECO1 7
2.2 RS232 7
2.2.1 Indicações 7
2.2.2 Conexão com a Rede RS232 7
2.2.3 Cabos para Ligação em RS232 7
2.2.4 Pinagem do Conector 7
2.3 RS485 8
2.3.1 Indicações 8
2.3.2 Características da interface RS485 8
2.3.3 Pinagem do Conector 8
2.3.4 Resistor de terminação 9
2.3.5 Conexão com a Rede RS485 9

3 PARAMETRIZAÇÃO 10

3.1 SÍMBOLOS PARA DESCRIÇÃO DAS PROPRIEDADES 10
P00650 – ENDEREÇO DO SERVOCONVERSOR NA COMUNICAÇÃO SERIAL 1 – RS232 10
P00656 – ENDEREÇO DO SERVOCONVERSOR NA COMUNICAÇÃO SERIAL 2 – RS485 10
P00652 – BIT RATE SERIAL 1 – RS232 10
P00658 – BIT RATE SERIAL 2 – RS485 10
P00653 – CONFIGURAÇÃO SERIAL 1 – RS232 11
P00659 – CONFIGURAÇÃO SERIAL 2 – RS485 11
P00654 – SELECIONA PROTOCOLO SERIAL 1 – RS232 11
P00660 – SELECIONA PROTOCOLO SERIAL 2 – RS485 11
P00662 – AÇÃO PARA ERRO DE COMUNICAÇÃO 11
P00663 – WATCHDOG SERIAL 12
P00664 – SALVA PARÂMETROS EM MEMÓRIA NÃO VOLÁTIL 12
P00667 – SALVA EM MARCADORES 12

4 PROTOCOLO MODBUS RTU 14

4.1 MODOS DE TRANSMISSÃO 14
4.2 ESTRUTURA DAS MENSAGENS NO MODO RTU 14
4.2.1 Endereço 14
4.2.2 Código da Função 14
4.2.3 Campo de Dados 14
4.2.4 CRC 14
4.2.5 Tempo entre Mensagens 15

5 OPERAÇÃO NA REDE MODBUS RTU – MODO ESCRAVO 16

5.1 FUNÇÕES DISPONÍVEIS E TEMPOS DE RESPOSTA 16
5.2 MAPA DE MEMÓRIA 16

6 DESCRIÇÃO DETALHADA DAS FUNÇÕES 19

6.1 FUNÇÃO 03 – READ HOLDING REGISTER 19
6.2 FUNÇÃO 06 – WRITE SINGLE REGISTER 19
6.3 FUNÇÃO 16 – WRITE MULTIPLE REGISTERS 20
6.4 FUNÇÃO 43 – READ DEVICE IDENTIFICATION 21
6.5 ERROS DE COMUNICAÇÃO 21

7 FALHAS E ALARMES RELACIONADOS COM A COMUNICAÇÃO MODBUS RTU 23

A00128/F00028 – TIMEOUT NA RECEPÇÃO DE TELEGRAMAS 23

I. APÊNDICES 24

APÊNDICE A. TABELA ASCII 24
APÊNDICE B. CÁLCULO DO CRC UTILIZANDO TABELAS 25

SOBRE O MANUAL

Este manual fornece a descrição necessária para a operação do servoconversor SCA06 utilizando o protocolo Modbus. Este manual deve ser utilizado em conjunto com manual do usuário do SCA06.

ABREVIAÇÕES E DEFINIÇÕES

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

REPRESENTAÇÃO NUMÉRICA

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

DOCUMENTOS

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

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

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

1 INTRODUÇÃO À COMUNICAÇÃO SERIAL

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

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

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

2 DESCRIÇÃO DAS INTERFACES

Para possibilitar a comunicação serial no servoconversor SCA06, é necessário utilizar o módulo de expansão e comunicação ECO1 descrito a seguir. Informações sobre a instalação deste módulo podem ser obtidas no guia que acompanha o acessório.

2.1 MÓDULO DE EXPANÇÃO E COMUNICAÇÃO RS232 E RS485 ECO1

- Item WEG: 11330271.
- Composto pelo módulo de comunicação ECO1 (figura ao lado) e um guia de montagem.
- Interface isolada galvanicamente e com sinal diferencial, conferindo maior robustez contra interferência eletromagnética.
- Os sinais RS232 e RS485 são canais independentes, podendo ser utilizados simultaneamente.

2.2 RS232

2.2.1 Indicações

O LED LA121 indica (acende) quando há transmissão de dados através da comunicação RS232.

2.2.2 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.

2.2.3 Cabos para Ligação em RS232

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

Table 2.1: *Cabos RS232*

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.

2.2.4 Pinagem do Conector

A conexão para a interface RS232 está disponível através dos conectores XA121 e XA122 utilizando a seguinte pinagem:

Tabela 2.1: *Pinagem do conector para RS232 XA121*

Pino	Função
1	Terra
2	RX_232
3	TX_232
4	Reservado[1]
5	GND
6	Reservado
7	232 RTS
8	A
9	B

Tabela 2.2: *Pinagem do conector para RS232 XA122*

Pino	Função
1	NC
2	RX_232
3	TX_232
4	Reservado
5	GND
6	NC
7	232 RTS
8	NC
9	NC
Carcaça	Terra

2.3 RS485

2.3.1 Indicações

O LED LA122 indica (acende) quando há transmissão de dados através da comunicação RS485.

2.3.2 Características da interface RS485

- Interface segue o padrão EIA/TIA-485.
- Pode operar como escravo da rede Modbus RTU.
- Possibilita comunicação utilizando taxas de 9600 até 57600 Kbit/s.
- Interface isolada galvanicamente e com sinal diferencial, conferindo maior robustez contra interferência eletromagnética.
- Permite a conexão de até 32 dispositivos no mesmo segmento. Uma quantidade maior de dispositivos pode ser conectada com o uso de repetidores.[2]
- Comprimento máximo do barramento de 1000 metros.

2.3.3 Pinagem do Conector

A conexão para a interface RS485 está disponível através do conector XC1 utilizando a seguinte pinagem:

Tabela 2.3: *Pinagem do conector para RS485 XA121*

Pino	Função
1	Terra
2	RX_232
3	TX_232
4	Reservado
5	GND
6	Reservado
7	232 RTS
8	A
9	B

[1] Não conectar os pinos reservados.
[2] O número limite de equipamentos que podem ser conectados na rede também depende do protocolo utilizado.

Tabela 2.4: *Pinagem do conector para RS485 XA123*

Pino	Função
1	NC
2	NC
3	NC
4	NC
5	GND
6	Reservado
7	NC
8	A (data -)
9	B (data +)
Carcaça	Terra

2.3.4 Resistor de terminação

Para cada segmento da rede RS485, é necessário habilitar um resistor de terminação nos pontos extremos do barramento principal. O acessório ECO1 possui duas chaves DIP Switch que podem ser ativadas (colocando ambas as chaves SA121 na posição ON) para habilitar o resistor de terminação.

Figura 2.1: *Resistor de terminação*

2.3.5 Conexão com a Rede RS485

Para a ligação do servoconversor SCA06 utilizando a interface RS485, os seguintes pontos devem ser observados:

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

3 PARAMETRIZAÇÃO

A seguir serão apresentados apenas os parâmetros do servoconversor SCA06 que possuem relação direta com a comunicação Modbus RTU.

3.1 SÍMBOLOS PARA DESCRIÇÃO DAS PROPRIEDADES

RW	Parâmetro de leitura e escrita
AC	Parâmetro visível na HMI somente quando o acessório correspondente esta conectado

P00650 – ENDEREÇO DO SERVOCONVERSOR NA COMUNICAÇÃO SERIAL 1 – RS232

P00656 – ENDEREÇO DO SERVOCONVERSOR NA COMUNICAÇÃO SERIAL 2 – RS485

Faixa de Valores:	1 a 247	**Padrão:** 1
Propriedades:	RW, AC	

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 servoconversor.

- P00654/P00660 = 1 (WEGTP) → endereços válidos: 1 to 30.
- P00654/P00660 = 2 (Modbus RTU) → endereços válidos: 1 to 247.

P00652 – BIT RATE SERIAL 1 – RS232

P00658 – BIT RATE SERIAL 2 – RS485

Faixa de Valores:	0 = 4800 bits/s 1 = 9600 bits/s 2 = 14400 bits/s 3 = 19200 bits/s 4 = 24000 bits/s 5 = 28800 bits/s 6 = 33600 bits/s 7 = 38400 bits/s 8 = 43200 bits/s 9 = 48000 bits/s 10 = 52800 bits/s 11 = 57600 bits/s	**Padrão:** 1
Propriedades:	RW, AC	

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.

P00653 – CONFIGURAÇÃO SERIAL 1 – RS232

P00659 – CONFIGURAÇÃO SERIAL 2 – RS485

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 6 = 7 bits de dados, sem paridade, 1 stop bit 7 = 7 bits de dados, paridade par, 1 stop bit 8 = 7 bits de dados, paridade impar, 1 stop bit 9 = 7 bits de dados, sem paridade, 2 stop bits 10 = 7 bits de dados, paridade par, 2 stop bits 11 = 7 bits de dados, paridade ímpar, 2 stop bits	**Padrão:** 3
Propriedades:	RW, AC	

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.

P00654 – SELECIONA PROTOCOLO SERIAL 1 – RS232

P00660 – SELECIONA PROTOCOLO SERIAL 2 – RS485

Faixa de Valores:	1 = WEGTP 2 = Modbus RTU	**Padrão:** 1
Propriedades:	RW	

Descrição:
Permite selecionar o protocolo desejado para a interface serial. A descrição detalhada do protocolo é feita no item 4 deste manual.

P00662 – AÇÃO PARA ERRO DE COMUNICAÇÃO

Faixa de Valores:	0 = Indica Alarme 1 = Causa Falha 2 = Causa alarme e Stop 3 = Causa alarme e desabilita	**Padrão:** 0
Propriedades:	RW	

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 3.1: *Opções para o parâmetro P00662*

Opção	Descrição
0 = Apenas indica alarme	Apenas mostra o código de alarme na HMI do servoconversor.
1 = Causa Falha	Causa falha e o servoconversor só volta a operar caso seja feito reset de falhas.
2 = Indica alarme e executa STOP	Será feita a indicação de alarme juntamente com a execução do comando STOP. Para que o servo saia desta condição, será necessário realizar o reset de falhas ou desabilitar o drive.
3 = Indica alarme e desabilita drive	Será feita a indicação de alarme juntamente com a execução do comando desabilita.

São considerados erros de comunicação os seguintes eventos:

Comunicação Serial (RS232/RS485):
- Alarme A00128/Falha F00028: *timeout* da interface serial.

P00663 – WATCHDOG SERIAL

Faixa de Valores:	0.0 a 999.0s	**Padrão:** 0.0
Propriedades:	RW	

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 A00128 na HMI (ou falha F00028, dependendo da programação feita no P00662) e a ação programada no P00662 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.

P00664 – SALVA PARÂMETROS EM MEMÓRIA NÃO VOLÁTIL

Faixa de Valores:	0 = Não salva parâmetro na memória não volátil 1 = Salva parâmetro na memória volátil	**Padrão:** 1
Propriedades:	RW	

Descrição:
Permite selecionar se a escrita de parâmetros via serial deve ou não salvar o conteúdo dos parâmetros em memória não volátil (EEPROM). Quando utilizado o protocolo Modbus e apenas esse parâmetro que determina se os parâmetros escritos via serial serão ou não salvos na memória não volátil. Porém, quando utilizado o protocolo WEGTP, deve-se observar que no byte de código do telegrama consta a informação sobre salvar ou não o parâmetro na EEPROM. Para que via WEGTP os mesmos sejam salvos em memória não volátil, é necessário que as duas informações, o byte de código do telegrama e o parâmetro P00664, sejam verdadeiras.

NOTA!
Este tipo de memória possui um numero limite de escritas (100.000 vezes). Dependendo da aplicação, este limite pode ser ultrapassado, caso alguns parâmetros sejam escritos ciclicamente via serial (referencia de velocidade, torque, etc.). Nestes casos, pode ser desejado que, durante a operação do servoconversor, a escrita via serial não salve o conteúdo dos parâmetros em memória não volátil, para não ultrapassar o limite de escritas no servoconversor.

NOTA!
Esse parâmetro não se aplica quando a escrita e feita utilizando a interface USB.

P00667 – SALVA EM MARCADORES

Faixa de Valores:	0 = Lê e escreve normalmente o conteúdo no parâmetro correspondente 1 = Lê e escreve conteúdo em marcadores de WORD volátil a partir do MW13000	**Padrão:** 0
Propriedades:	RW	

Descrição:
Propriedade verificada quando parâmetro e escrito e lido via serial. Seleciona se quer que conteúdo a ser escrito/lido seja salvo em parâmetro ou em marcador de Word volátil.

NOTA!
Sendo este parâmetro P00667 = 1, ao escrever via serial no parâmetro P00105 = 30, o conteúdo do parâmetro será armazenado no marcador de Word 13105 (MWinicial + Numero_par => 13000 + 105). Portanto, MW13105 = 30.

Observação: Uma vez que P00667 = 1, o mesmo não poderá ser alterado via serial. Pois na tentativa de escrever no parâmetro P00667 estará escrevendo no marcador de Word P13667.

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 servoconversor SCA06 utiliza somente o modo RTU para a transmissão de telegramas. Os bytes são de acordo com a configuração feita através do P00659 ou P00663.

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 servoconversor irá descartar os bytes já recebidos e montará um novo telegrama com os bytes que estiverem sendo transmitidos.

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

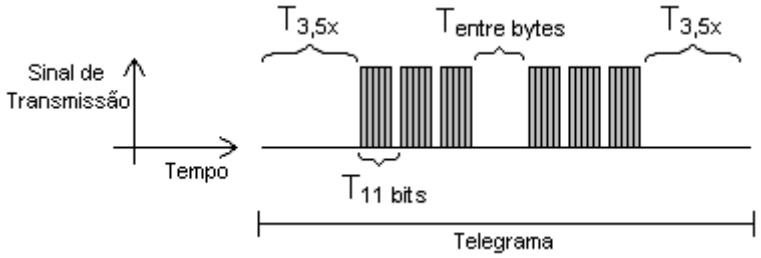

Tabela 4.1: *Taxas de comunicação e tempos envolvidos na transmissão de telegramas*

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

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

5 OPERAÇÃO NA REDE MODBUS RTU – MODO ESCRAVO

Como escravo da rede Modbus RTU, o servoconversor SCA06 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 servoconversor 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 SCA06, os parâmetros foram definidos como sendo registradores do tipo *holding*. Para acessar estes dados, foram disponibilizados os seguintes serviços (ou funções):

- Read Coils[3]
  Descrição: leitura de bloco de bits do tipo *coil*.
  Código da função: 01.

- Read Discrete Inputs[3]
  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[3]
  Descrição: leitura de bloco de registradores do tipo *input*.
  Código da função: 04.

- Write Single Coil[3]
  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[3]
  Descrição: escrita em bloco de bits do tipo *coil*.
  Código da função: 15.

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

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

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

5.2 MAPA DE MEMÓRIA

A comunicação Modbus para o servoconversor SCA06 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

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

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 a interface Modbus RTU*

Número do Parâmetro	Endereço do dado Modbus	
	Decimal	Hexadecimal
P0000	0	0000h
P0001	1	0001h
⋮	⋮	⋮
P0099	99	0063h
⋮	⋮	⋮

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):
- P0002 (holding register 2): Velocidade do motor
- P0052 (holding register 52): Posição angular: fração de volta

Comando (escrita):
- P0121 (holding register 121): Referência de velocidade
- P01100 (holding register 1100): Parâmetro do usuário 1100

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

Além dos parâmetros, outros tipos de dados como marcadores de *bit*, *word*, *float e double* também podem ser acessados utilizando o protocolo Modbus RTU. Estes marcadores são utilizados pela função Ladder disponível no SCA06. O endereçamento dos marcadores também é feito com offset igual a zero, o que significa que o número do marcador equivale ao número do coil ou registrador.

Tabela 5.2: *Mapa de memória para a interface Modbus RTU (Marcadores)*

Número do Marcador de BIT Retentivo	Endereço do dado Modbus	
	Decimal	Hexadecimal
MX6000	6000	1770h
⋮	⋮	⋮
MX7999	7999	1F3Fh

Número do Marcador de BIT Volátil	Endereço do dado Modbus	
	Decimal	Hexadecimal
MX8000	8000	1F40h
⋮	⋮	⋮
MX9999	9999	270Fh

Número do Marcador de WORD Retentivo	Endereço do dado Modbus	
	Decimal	Hexadecimal
MW12000	12000	2EE0h
⋮	⋮	⋮
MW12999	12999	32C7h

Número do Marcador de WORD Volátil	Endereço do dado Modbus	
	Decimal	Hexadecimal
MW13000	13000	32C8h
⋮	⋮	⋮
MW14999	14999	3A97h

Número do Marcador de FLOAT Retentivo	Endereço do dado Modbus	
	Decimal	Hexadecimal
MF16000	16000	3E80h
⋮	⋮	⋮
MF16499	16499	4073h

Número do Marcador de FLOAT Volátil	Endereço do dado Modbus	
	Decimal	Hexadecimal
MF17000	17000	4268h
⋮	⋮	⋮
MF17999	17999	464Fh

Número do Marcador de DOUBLE Retentivo	Endereço do dado Modbus	
	Decimal	Hexadecimal
MD18000	18000	4650h
⋮	⋮	⋮
MD18249	18249	4749h

Número do Marcador de DOUBLE Volátil	Endereço do dado Modbus	
	Decimal	Hexadecimal
MD19000	19000	4A38h
⋮	⋮	⋮
MD19549	MF19549	4C5Dh

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.
- Os marcadores são criados somente se forem utilizados no aplicativo ladder, sendo que são criados todos do primeiro ao último marcador utilizado. Exemplo: se no aplicativo o último marcador de word utilizado for o MW13002, serão criados os marcadores MW13000, MW13001 e MW13002.

6 DESCRIÇÃO DETALHADA DAS FUNÇÕES

Neste item é feita uma descrição detalhada das funções disponíveis no servoconversor SCA06 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 SCA06 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 = 1000 rpm e P0003 = 3,5 A).

- Endereço: 1 = 01h (1 byte)
- Endereço do registrador inicial: 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	P0002 (high)	03h
Quantidade de registradores (high)	00h	P0002 (low)	E8h
Quantidade de registradores (low)	02h	P0003 (high)	00h
CRC-	65h	P0003 (low)	23h
CRC+	CBh	CRC-	3Bh
		CRC+	9Ah

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 (P0121) em 2000 rpm para o escravo no endereço 3.
- Endereço: 3 = 03h (1 byte)
- Endereço do registrador inicial: 121 = 0079h (2 bytes)
- Valor para o parâmetro: 07D0h (2 bytes)

Pergunta (Mestre)		**Resposta (Escravo)**	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	03h	Endereço do escravo	03h
Função	06h	Função	06h
Registrador (high)	00h	Registrador (high)	00h
Registrador (low)	79h	Registrador (low)	79h
Valor (high)	07h	Valor (high)	07h
Valor (low)	D0h	Valor (low)	D0h
CRC-	5Ah	CRC-	5Ah
CRC+	5Dh	CRC+	5Dh

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 da função das três entradas digitais P0300, P0301 e P0302 igual a 4, 4 e 10, respectivamente, de um escravo no endereço 15.
- Endereço: 15 = 0Fh (1 byte)
- Endereço do registrador inicial: 300 = 012Ch (2 bytes)
- Valor para o primeiro parâmetro: 4 = 0004h (2 bytes)
- Valor para o segundo parâmetro: 4 = 0004h (2 bytes)
- Valor para o terceiro parâmetro: 10 = 000Ah (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)	01h	Registrador inicial (high)	01h
Registrador inicial (low)	2Ch	Registrador inicial (low)	2Ch
Quantidade de registradores (high)	00h	Quantidade de registradores (high)	00h
Quantidade de registradores (low)	03h	Quantidade de registradores (low)	03h
Byte Count	06h	CRC-	41h
P0300 (high)	00h	CRC+	13h
P0300 (low)	04h		
P0301 (high)	00h		
P0301 (low)	04h		
P0302 (high)	00h		
P0302 (low)	0Ah		
CRC-	05h		
CRC+	A1h		

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 servoconversor SCA06, 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 (SCA06).
- 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 SCA06 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

A00128/F00028 – 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 P00663.

Atuação:
O parâmetro P00663 permite programar um tempo dentro do qual o servoconversor 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 servoconversor.

Depois de identificado o timeout na comunicação serial, será sinalizada através da HMI a mensagem de alarme A00128 – ou falha F00028, dependendo da programação feita no P00662. 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 P00663.
- Desabilitar esta função no P00663.

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,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40 };


/* Table of CRC values for low–order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04,
0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8,
0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10,
0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C,
0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0,
0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C,
0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54,
0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40 };


/* The function returns the CRC as a unsigned short type */
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg;                 /* message to calculate CRC upon    */
unsigned short usDataLen;               /* quantity of bytes in message     */
{
    unsigned char uchCRCHi = 0xFF;      /* high byte of CRC initialized     */
    unsigned char uchCRCLo = 0xFF;      /* low byte of CRC initialized      */
    unsigned uIndex;                    /* will index into CRC lookup table */
    while (usDataLen--)                 /* pass through message buffer      */
    {
        uIndex = uchCRCLo ^ *puchMsgg++; /* calculate the CRC               */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex};
        uchCRCHi = auchCRCLo[uIndex];
    }

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

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