Soft-Starter SSW-07/SSW-08 Manual da Comunicação Serial

04/2011

Série: SSW-07/SSW-08
Idioma: Português
Documento: 0899.5802 / 03

Sumário

Cuidado

- Ler o manual da Soft-Starter SSW-07/SSW-08 na íntegra, antes de instalar ou operar o mesmo.

- Seguir atentamente os cuidados e avisos de segurança contidos nele.

- Quando houver possibilidade de danos a pessoas ou equipamentos relacionados a motores acionados por Soft-Starters SSW-07/SSW-08, prever dispositivos de segurança eletromecânicos.

Aviso

- Seguir atentamente os cuidados definidos neste manual, no que diz respeito aos cabos de interconexão das duas interfaces para comunicação serial.

- Equipamento com componentes sensíveis à eletricidade estática. Os cartões eletrônicos devem ser manuseados com os seguintes cuidados:
  - Não tocar com as mãos diretamente sobre componentes ou ligações (conectores). Quando necessário tocar antes em um objeto metálico aterrado.

Definições

Termos utilizados

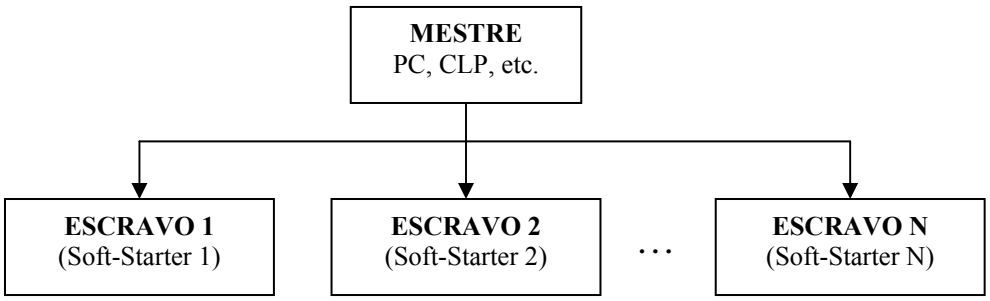
- Parâmetros: são aqueles existentes nas Soft-Starters SSW-07/SSW-08 cuja visualização ou alteração é possível através da interface homem-máquina (IHM).
- Variáveis básicas: valores internos da Soft-Starter SSW-07/SSW-08 que somente podem ser acessados através da serial, utilizados para monitoração dos estados, comandos e identificação do equipamento.
- Registradores: são endereços de memória interna da Soft-Starter. Podem ser usados para representar tanto variáveis básicas quanto parâmetros.
- EEPROM: é a memória não volátil que permite com que a Soft-Starter SSW-07/SSW-08 mantenha os valores dos parâmetros mesmo após desligar o equipamento.

Representação numérica

- Números decimais são representados através de dígitos sem sufixo.
- Números hexadecimais são representados com a letra ’h’ depois do número.

1. Introdução

O objetivo básico da comunicação serial é a ligação física entre dois ou mais equipamentos em uma rede configurada da seguinte forma:



Utilizando esta interface, o mestre da rede pode solicitar diversos serviços para cada escravo conectado na rede, tais como:

- IDENTIFICAÇÃO:
  - Tipo de equipamento (inversor de freqüência, servoconversor, soft-starter)
  - Monitoração dos estados
  - Leitura de erros

- PARAMETRIZAÇÃO
  - Leitura dos parâmetros (corrente, tensão, etc.)
  - Escrita de parâmetros para configuração do equipamento

- COMANDOS
  - Habilita / Desabilita
  - Desabilita geral
  - *Reset* de erros

A Soft-Starter SSW-07/SSW-08 utiliza o protocolo Modbus-RTU para comunicação através da sua interface serial. Este protocolo permite a integração da Soft-Starter SSW-07/SSW-08 em diferentes sistemas, uma vez que possibilita sua conexão a vários equipamentos, tais como:

- PC (mestre) para parametrização de uma ou várias Soft-Starters SSW-07/SSW-08 simultaneamente.
- SDCD monitorando variáveis e parâmetros da Soft-Starter SSW-07/SSW-08.
- CLP controlando a operação do equipamento em um processo industrial.

2. Parametrização da Soft-Starter SSW-07/SSW-08

A seguir serão descritos os parâmetros relacionados com a comunicação serial e operação via protocolo Modbus-RTU da Soft-Starter SSW-07/SSW-08.

2.1. P308 – Endereço da Soft-Starter SSW-07/SSW-08 na rede

Cada escravo da rede deve possuir um endereço diferente dos demais, para que o mestre possa enviar o telegrama desejado para um escravo específico da rede. Este parâmetro permite programar qual o endereço da Soft-Starter SSW-07/SSW-08 na rede.

Faixa de valores	Valor padrão	Acesso
1 ... 247	1	Leitura/ escrita

Sendo necessária a colocação de um repetidor no caso de se utilizarem mais que 30 equipamentos em uma mesma rede de comunicação.

2.2. P312 – Tipo de protocolo serial e taxa de comunicação

A Soft-Starter SSW-07/SSW-08 possui uma das seguintes opções para a comunicação através da interface serial do produto:

Faixa de valores	Valor padrão	Acesso
1 = Modbus-RTU, 9600 bit/s, sem paridade 2 = Modbus-RTU, 9600 bit/s, paridade ímpar 3 = Modbus-RTU, 9600 bit/s, paridade par 4 = Modbus-RTU, 19200 bit/s, sem paridade 5 = Modbus-RTU, 19200 bit/s, paridade ímpar 6 = Modbus-RTU, 19200 bit/s, paridade par 7 = Modbus-RTU, 38400 bit/s, sem paridade 8 = Modbus-RTU, 38400 bit/s, paridade ímpar 9 = Modbus-RTU, 38400 bit/s, paridade par	1	Leitura/ escrita

É necessário que todos os equipamentos que operam na mesma rede possuam a mesma configuração de comunicação.

2.3. P313 – Ação para erro de comunicação

Este parâmetro permite programar uma ação que o drive deve tomar em caso de ocorrência de erro de comunicação.

Faixa de valores	Padrão	Acesso
0 = Sem ação 1 = Desabilita 2 = Desabilita geral 3 = Vai para local	1	Leitura/ escrita

- **0 – Sem ação**: caso ocorra um dos erros citados, a SSW-07/SSW-08 permanece no estado atual e apenas indica o erro ocorrido.
- **1 – Desabilita**: a SSW-07/SSW-08 será desabilitada via rampa de tensão em caso de erro de comunicação.
- **2 – Desabilita geral**: nesta opção a chave de partida corta a alimentação para o motor, e este deverá parar por inércia.
- **3 – Vai para local**: caso a chave esteja operando no modo remoto e ocorra um erro de comunicação, ela irá automaticamente para o modo local.

Para a interface serial, apenas o erro de *timeout* na recepção de telegramas (E28 - Comunicação Serial Inativa) é considerado como erro na comunicação. O *timeout* na recepção de telegramas é programado através do parâmetro P314.

O erro E28 é indicado através de piscadas no LED Error do módulo de comunicação.

NOTA!
Os comandos de desabilitação e mudança para o modo local somente poderão ser executados se os mesmos forem controlados via serial. Esta programação é feita através dos parâmetros P220, P229 e P230.

2.4. P314 – Tempo para *timeout* na recepção de telegramas

Permite programar o tempo para detecção de *timeout* na recepção de telegramas. O valor 0 (zero) desabilita esta função.

Caso o drive seja controlado via serial e ocorra um problema na comunicação com o mestre (rompimento do cabo, queda de energia, etc.), não será possível enviar um comando via serial para a desabilitação do equipamento. Nas aplicações onde isto representa um problema, é possível programar no P314 um intervalo máximo dentro do qual a SSW-07/SSW-08 deve receber um telegrama serial válido, caso contrário ela irá considerar que houve falha na comunicação serial.

Faixa de valores	Valor padrão	Acesso
0 = Função desabilitada 0 ... 999 segundos	0	Leitura/ escrita

Uma vez programado este tempo, caso ele fique um tempo maior do que o programado sem receber telegramas seriais válidos, ele indicará E28 e tomará a ação programada no P313. Caso a comunicação seja restabelecida, a indicação de E28 será retirada.

NOTA!

- Quando esta função estiver habilitada, é necessário garantir que o mestre da rede envie telegramas periódicos para o escravo, respeitando o tempo programado, para que não ocorra erro de *timeout* na comunicação.
- A ocorrência de E28 também irá zerar o valor da variável básica 8 (ver item 4.2).

2.5. P220 – Seleção fonte local/remoto

Permite programar a fonte de comando que controla os modos local/remoto do equipamento.

Faixa de valores	Valor padrão	Acesso
0 = Sempre local 1 = Sempre remoto 2 = HMI (padrão local) 3 = HMI (padrão remoto) 4 = DI1...DI3 5 = Serial (padrão local) 6 = Serial (padrão remoto) 7 = Fieldbus (padrão local) 8 = Fieldbus (padrão remoto)	3	Leitura/ escrita

Caso se deseja controlar o modo de operação via serial, deve-se programar este parâmetro com o valor 5 ou 6. A indicação de "padrão local" ou "padrão remoto" informa qual o modo de operação que deve ser ativado após a inicialização do equipamento.

2.6. P229 – Seleção de comandos – situação local

Permite programar qual é a fonte dos comandos da Soft-Starter SSW-07/SSW-08 quando esta estiver no modo local.

Faixa de valores	Valor padrão	Acesso
0 = Teclado da HMI 1 = DI 2 = Serial 3 = Fieldbus	0	Leitura/ escrita

Caso deseje-se controlar os comandos via serial no modo local, deve-se programar este parâmetro em 2.

2.7. P230 – Seleção de comandos – situação remoto

Permite programar qual é a fonte dos comandos da Soft-Starter SSW-07/SSW-08 quando esta estiver no modo remoto.

Faixa de valores	Valor padrão	Acesso
0 = Teclado da HMI 1 = DI 2 = Serial 3 = Fieldbus	1	Leitura/ escrita

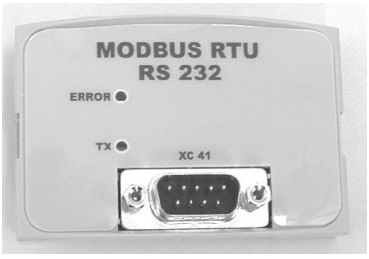
Caso deseje-se controlar os comandos via serial no modo remoto, deve-se programar este parâmetro em 2.

3. Descrição das Interfaces

3.1. RS-232

Para usar a RS-232 na Soft-Starter SSW-07/SSW-08 é nessário o uso do Kit Modbus RTU RS-232.

Maiores detalhes ver o Guia de instalação do Kit Modbus RTU RS-232.

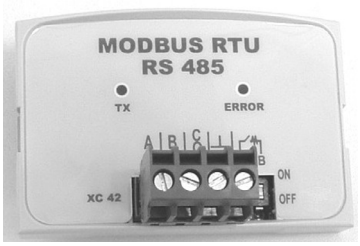


Esta interface possibilita a ligação de um mestre a uma Soft-Starter SSW-07/SSW-08 (ponto a ponto) em uma distância de até 10m. Para comunicação com o mestre, deve-se utilizar um fio para transmissão (TX), um para recepção (RX) e uma referência (terra).

3.2. RS-485

Para usar a RS-485 na Soft-Starter SSW-07/SSW-08 é necessário o uso do Kit Modbus RTU RS-485.

Maiores detalhes ver o Guia de instalação do Kit Modbus RTU RS-485.



Utilizando a interface RS-485, o mestre pode controlar diversos drives conectados em um mesmo barramento. O protocolo Modbus-RTU permite a conexão de até 247 escravos (1 por endereço), desde que utilizados também repetidores de sinal ao longo do

barramento. Esta interface possui uma boa imunidade a ruído, e o comprimento máximo permitido do cabo é de 1000 metros.

3.2.1. Utilização do Kit RS-485 para SSW-07/SSW-08

O kit RS-485 para Soft-Starter SSW-07/SSW-08 é composto basicamente por um módulo de interface e instruções para realizar a instalação no produto.

As seguintes recomendações devem ser observadas durante a instalação da rede utilizando esta interface:

- Geralmente utiliza-se um par de fios trançados com blindagem para a transmissão dos sinais A e B. Estes sinais devem ser conectados nos bornes A e B.
- O borne COM é utilizado para a conexão do sinal de referência para o circuito RS-485. Caso este sinal não seja utilizado, pode-se desconsiderar esta conexão.
- É muito importante aterrar corretamente todos os dispositivos conectados na rede RS-485, preferencialmente no mesmo ponto de terra. A blindagem do cabo também deve ser aterrada. Isto pode ser feito no conector XC42 na conexão terra. Caso a blindagem seja aterrada em outro ponto a blindagem deve ser conectada no conector COM.
- A passagem do cabo de rede deve ser feita separadamente, se possível, distante dos cabos para alimentação de potência.
- É necessário disponibilizar resistores de terminação no primeiro e no último dispositivo conectado no barramento principal. O módulo de interface para RS-485 já possui chaves para habilitação deste resistor. Basta colocar ambas as chaves para a posição 'on'.

4. Dados acessíveis via comunicação serial

Diversos dados podem ser acessados via interface serial na Soft-Starter SSW-07/SSW-08, para possibilitar sua parametrização, comando e monitoração. Basicamente, estes dados podem ser divididos em dois grupos: parâmetros e variáveis básicas.

4.1. Parâmetros da Soft-Starter SSW-07/SSW-08

Os parâmetros são aqueles disponíveis através da IHM da Soft-Starter SSW-07/SSW-08. Praticamente todos os parâmetros do drive podem ser acessados via serial, e a através destes parâmetros é possível configurar a forma como o equipamento irá operar, bem como monitorar informações relevantes para a aplicação, como corrente, erros, etc..

Deve-se consultar o manual de programação da Soft-Starter SSW-07/SSW-08 para a lista completa dos parâmetros.

4.2. Variáveis básicas disponíveis para a Soft-Starter SSW-07/SSW-08

As variáveis básicas são valores internos da Soft-Starter SSW-07/SSW-08 acessíveis somente através da interface serial do produto. Utilizando estas variáveis, é possível monitorar os estados da Soft-Starter bem como enviar comandos de habilitação, *reset*, etc..

Cada variável básica representa um registrador (16 bits). Para a Soft-Starter SSW-07/SSW-08 foram disponibilizadas as seguintes variáveis básicas:

4.2.1. Variável básica 1

- *Variável*: VB01 – estado da Soft-Starter SSW-07/SSW-08
- *Acesso*: somente leitura
- *Descrição*: indica o estado da Soft-Starter SSW-07/SSW-08. Cada bit desta palavra fornece uma indicação diferente:

Bit	Descrição
Bit 0	0 = motor parado. 1 = motor girando.
Bit 1	0 = quando desabilitada geral por qualquer um dos meios. 1 = quando está habilitada geral por todos os meios.
Bit 2	0 = sem Jog. (1) 1 = com Jog.
Bit 3	0 = não está acelerando. 1 = durante toda a aceleração.
Bit 4	0 = não está em limitação de corrente. 1 = limitação de corrente.
Bit 5	0 = sem tensão plena sobre o motor. 1 = com tensão plena sobre o motor.
Bit 6	Reservado
Bit 7	0 = não está desacelerando. 1 = durante toda a desaceleração.
Bit 8	0 = local. 1 = remoto.
Bit 9	0 = não está em frenagem CC. (1) 1 = durante a frenagem CC.
Bit 10	0 = não está invertendo sentido de giro. (1) 1 = durante o processo de troca do sentido de giro.
Bit 11	0 = horário. (1) 1 = anti-horário.
Bit 12	0 = com bypass aberto. 1 = com bypass fechado.
Bit 13	Reservado
Bit 14	0 = sem alimentação da potência. 1 = com alimentação da potência.
Bit 15	0 = sem erro. 1 = com erro.

(1) Função disponível a partir da Versão de Software 1.4x

4.2.2. Variável básica 3

- *Variável*: VB03 – comando lógico.
- *Acesso*: leitura e escrita
- *Descrição*: permite comandar a Soft-Starter SSW-07/SSW-08 via serial.

Esta palavra possui 16 bits, onde somente os oito primeiro bits possuem função. Cada bit possui o valor efetivo para cada comando que se deseja executar.

Bit	Descrição
Bit 0	0 = parar por rampa. 1 = girar por rampa.
Bit 1	0 = desabilita geral 1 = habilita geral.
Bit 2	0 = sem Jog. (1) 1 = com Jog.
Bit 3	0 = sentido horário. (1) 1 = sentido anti-horário.
Bit 4	0 = local. 1 = remoto.
Bit 5	Reservado
Bit 6	Reservado
Bit 7	0 = sem comando. 0 → 1 = executa *reset* (caso esteja em erro).

(1) Função disponível a partir da Versão de Software 1.4x

Sempre que um comando for enviado para a Soft-Starter SSW-07/SSW-08, esta somente irá executar o comando caso esteja programada para receber comandos via serial. Esta programação é feita através dos seguintes parâmetros:

- P220 - Seleção da fonte local / remoto.
- P229 - Seleção dos comandos no modo local.
- P230 - Seleção dos comandos no modo remoto.

Deve-se programar estes comandos para a opção "Serial" sempre que se desejar executar o referido comando via rede. O comando de *reset* pode ser executado via rede mesmo sem esta parametrização, mas somente se a Soft-Starter SSW-07/SSW-08 estiver em estado de erro.

NOTA!

- Erros do cartão de comunicação (E28, E29 ou E30) não podem ser "resetados" desta forma, pois dependem de ajustes fora dos valores enviados via rede para serem solucionados, e também porque nesta situação a SSW-07/SSW-08 não está conseguindo se comunicar com a rede.
- Caso tente-se realizar algum comando via rede, mas que não possa ser executado pela SSW-07/SSW-08 (por exemplo, um comando que não esteja programado para operar via serial), este comando não será executado.

4.2.3. Variável básica 8

- *Variável*: VB08 – comandos para as saídas digitais
- *Acesso*: leitura e escrita
- *Descrição*: permite comandar as saídas a relé disponíveis na Soft-Starter SSW-07/SSW-08. Esta palavra possui 16 bits, onde somente os dois primeiro bits possuem função:

Bit	Descrição
Bit 0	0 = desativa saída a relé RL1. 1 = ativa saída a relé RL1.
Bit 1	0 = desativa saída a relé RL2. 1 = ativa saída a relé RL2.
Bit 2 ... 15	Reservado

Para que as saídas digitais possam ser comandadas via serial, é necessário programar as suas funções para a opção "Serial", nos parâmetros P277 e P278. Caso a saída não esteja sendo controlada via serial, o valor recebido no bit correspondente é desconsiderado.

Caso ocorra um erro de comunicação com o mestre da rede (E28), os valores para as saídas digitais são zerados.

4.3. Alteração de parâmetros e variáveis básicas

Existem algumas particularidades quanto ao acesso de parâmetros e variáveis básicas da Soft-Starter SSW-07/SSW-08 via serial:

- Não existe senha para acesso via serial. É possível alterar parâmetros independente da senha estar ativa ou não.
- O valor para o P000 não é salvo na memória não volátil do equipamento (o mesmo acontece via IHM).
- Os parâmetros P200 e P215 não estão acessíveis via serial.
- Se for enviado o comando de gira na variável básica 3 durante a atuação do tempo de P630, o comando não será aceito, e o drive não responderá ao erro.

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 da Soft-Starter SSW-07/SSW-08 foi desenvolvida com base nos seguintes documentos:

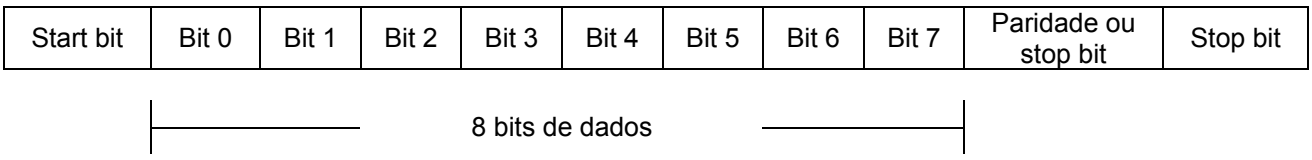
- MODBUS Protocol Reference Guide Rev. J, MODICON, June 1996.
- MODBUS Application Protocol Specification, MODBUS.ORG, May 8th 2002.
- MODBUS over Serial Line, MODBUS.ORG, December 2nd 2002.

Nestes documentos está definido o formato das mensagens utilizadas 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.

A Soft-Starter SSW-07/SSW-08 utiliza somente o modo RTU para a transmissão de telegramas. Os bytes são transmitidos no formato hexadecimal, onde cada byte transmitido possui 1 start bit, 8 bits de dados, 1 bit de paridade (opcional) e 1 stop bit (2 stop bits se não for utilizada paridade). A configuração do formato dos bytes é feita através do parâmetro P312.



5.2. Estrutura das mensagens no modo RTU

A rede Modbus-RTU utiliza o sistema mestre-escravo para a troca de mensagens. Ela pode possuir 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.

Para a Soft-Starter SSW-07/SSW-08, os dados relativos aos parâmetros e variáveis básicas 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 são fornecidas nos apêndices B e C.

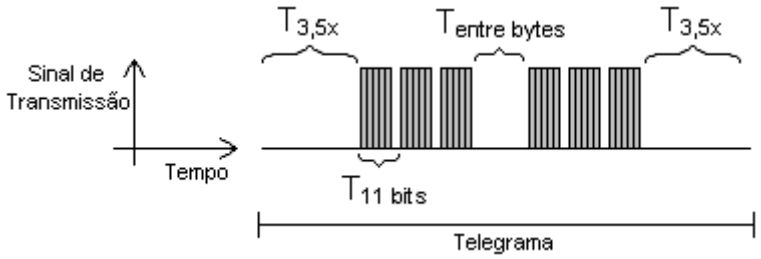
5.2.5. Tempo entre mensagens

No modo RTU não existe um caracter específico que indique o início ou o fim de um telegrama. Desta forma, o que indica quando uma nova mensagem começa ou quando ela termina é a 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 drive irá descartar os bytes já recebidos e montará um novo telegrama com os bytes que estiverem sendo transmitidos.

A tabela a seguir nos mostra os tempos para diferentes taxas de comunicação:



Taxa de Comunicação	T 11 bits	T 3,5x
9600 bits/seg	1,146 ms	4,010 ms
19200 bits/seg	573 µs	2,005 ms
38400 bits/seg	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}).

Para taxas de comunicação acima de 19200 bits/s, são considerados os mesmos tempos que o utilizado para 19200 bits/s.

5.3. Operação da Soft-Starter SSW-07/SSW-08 na rede Modbus-RTU

A Soft-Starter SSW-07/SSW-08 possui as seguintes características quando operado em rede Modbus-RTU:

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

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. Na Soft-Starter SSW-07/SSW-08, tanto parâmetros quanto variáveis básicas foram definidos como sendo registradores do tipo *holding*. Além destes registradores, também é possível acessar diretamente bits internos de comando e monitoração, denominados *coils*. Para acessar estes bits e registradores, foram disponibilizados os seguintes serviços (ou funções):

- *Read Coils*
  Descrição: Leitura de bloco de bits internos ou bobinas.
  Código da função: 01.
  Tempo de resposta: 5 a 20ms.

- *Read Holding Registers*
  Descrição: Leitura de bloco de registradores do tipo *holding*.
  Código da função: 03.
  Tempo de resposta: 5 a 20 ms.

- *Write Single Coil*
  Descrição: Escrita em um único bit interno ou bobina.
  Código da função: 05.
  Tempo de resposta: 5 a 20 ms.

- *Write Single Register*
  Descrição: Escrita em um único registrador do tipo *holding*.
  Código da função: 06.
  Tempo de resposta: 5 a 20 ms.

- *Write Multiple Coils*
  Descrição: Escrita em bloco de bits internos ou bobinas.
  Código da função: 15.
  Tempo de resposta: 5 a 20 ms.

- *Write Multiple Registers*
  Descrição: Escrita em bloco de registradores do tipo *holding*.
  Código da função: 16.
  Tempo de resposta: 20 ms para cada registrador escrito.

- *Read Device Identification*
  Descrição: Identificação do modelo do drive.
  Código da função: 43.
  Tempo de resposta: 5 a 20 ms.

5.3.2. Endereçamento dos dados e offset

O endereçamento dos dados na Soft-Starter SSW-07/SSW-08 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), enquanto que as variáveis

básicas são disponibilizadas a partir do endereço 5000. Da mesma forma, os bits de estado são disponibilizados a partir do endereço 0 (zero) e os bits de comando são disponibilizados a partir do endereço 100. A tabela a seguir ilustra o endereçamento de parâmetros e variáveis básicas:

PARÂMETROS		
Número do Parâmetro	Endereço Modbus	
	Decimal	Hexadecimal
P000	0	0x0000
P001	1	0x0001
⋮	⋮	⋮
P101	101	0x0065
⋮	⋮	⋮

VARIÁVEIS BÁSICAS		
Número da Variável Básica	Endereço Modbus	
	Decimal	Hexadecimal
V01	5001	0x1389
⋮	⋮	⋮
V08	5008	0x1390

BITS DE ESTADO		
Número do bit	Endereço Modbus	
	Decimal	Hexadecimal
Bit 0	00	00h
Bit 1	01	01h
⋮	⋮	⋮
Bit 15	15	0Fh

BITS DE COMANDO		
Número do bit	Endereço Modbus	
	Decimal	Hexadecimal
Bit 100	100	64h
Bit 101	101	65h
⋮	⋮	⋮
Bit 107	107	6Bh

Os bits de estado (0 até 15) possuem a mesma função de cada bit da variável básica 1 (ver item 4.2.1), enquanto que os bits de comando (100 até 107) possuem a mesma função dos bits menos significativos da variável básica 3, sem a necessidade de utilizar a máscara para comandar a SSW-07/SSW-08 (ver item 4.2.2).

NOTA!
Todos os registradores (parâmetros e variáveis básicas) são tratados como registradores do tipo *holding*. Dependendo do mestre utilizado, estes registradores são referenciados a partir do endereço base 40000 ou 4*x*. Neste caso, o endereço para um parâmetro ou variável básica que deve ser programado no mestre é o endereço mostrado na tabela acima adicionado do endereço base. De forma similar, os bits são referenciados a partir de 0000 ou 0*x*, denominados *coils*. Consulte a documentação do mestre para saber como acessar registradores do tipo *holding* e *coils*.

5.4. Descrição detalhada das funções

Neste item é feita uma descrição detalhada das funções disponíveis no SSW-07/SSW-08 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 (*high* e *low*).
- Os telegramas, tanto para pergunta quanto para resposta, não pode ultrapassar 256 bytes.
- Os valores transmitidos são sempre números inteiros, independente de possuírem representação com casa decimal. Desta forma, o valor 9,5 será transmitido como sendo 95 via serial. Consulte o manual do SSW-07/SSW-08 para obter a resolução utilizada para cada parâmetro.

5.4.1. Função 01 – Read Coils

Lê o conteúdo de um grupo de bits internos 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 bit inicial (byte high)	Campo Byte Count (no. de bytes de dados)
Endereço do bit inicial (byte low)	Byte 1
Número de bits (byte high)	Byte 2
Número de bits (byte low)	Byte 3
CRC-	etc...
CRC+	CRC-
	CRC+

Cada bit da resposta é colocado em uma posição dos bytes de dados enviados pelo escravo. O primeiro byte recebe os 8 primeiros bits a partir do endereço inicial indicado pelo mestre. Os demais bytes continuam a seqüência, caso o número de bits de

leitura seja maior que 8. Caso o número de bits lidos não seja múltiplo de 8, os bits restantes do último byte devem ser preenchidos com 0 (zero).

Exemplo 1: leitura dos bits de estado da habilitação (bit 0) e habilitação geral (bit 1) da SSW-07/SSW-08 no endereço 1 (supondo habilitação inativa e habilitação geral ativa).

- Endereço: 1 = 01h (1 byte)
- Número do bit inicial: 0 = 0000h (2 bytes)
- Número de bits lidos: 2 = 0002h (2 bytes)

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	01h	Função	01h
Bit inicial (high)	00h	Byte Count	01h
Bit inicial (low)	00h	Estado dos bits 1 e 2	02h
No. de bits (high)	00h	CRC-	D0h
No. de bits (low)	02h	CRC+	49h
CRC-	BDh		
CRC+	CBh		

No exemplo, como o número de bits lidos é menor que 8, o escravo precisou de apenas 1 byte para a resposta. O valor do byte foi 02h, que em binário tem a forma 0000 0010. Como o número de bits lidos é igual a 2, somente nos interessa os dois bits menos significativos, que possuem os valores 0 = habilitado e 1 = habilitado geral. Os demais bits, como não foram solicitados, são preenchidos com 0 (zero).

5.4.2. 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 representados em 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 2: leitura da corrente do motor em porcentagem (P002) e corrente do motor em ampéres (P003) da Soft-Starter SSW-07/SSW-08 no endereço 1 (supondo P002 = 50.0% e P003 = 40.0 A).

- Endereço: 1 = 01h (1 byte)
- Número do primeiro parâmetro: 2 = 0002h (2 bytes)
- Número de parâmetros lidos: 2 = 0002h (2 bytes)
- Valor lido do primeiro parâmetro:500 = 01F4h (2 bytes)
- Valor lido do segundo parâmetro: 400 = 0190h (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)	01h
No. de registradores (high)	00h	P002 (low)	F4h
No. de registradores (low)	02h	P003 (high)	01h
CRC-	65h	P003 (low)	90h
CRC+	CBh	CRC-	BBh
		CRC+	C1h

5.4.3. Função 05 – *Write Single Coil*

Esta função é utilizada para escrever um valor para um único bit (*coil*). O valor para o bit é representado utilizando dois bytes, onde o valor FF00h representa o bit igual a 1, e o valor 0000h representa o bit igual a 0 (zero). Possui a seguinte estrutura (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 bit (byte high)	Endereço do bit (byte high)
Endereço do bit (byte low)	Endereço do bit (byte low)
Valor para o bit (byte high)	Valor para o bit (byte high)
Valor para o bit (byte low)	Valor para o bit (byte low)
CRC-	CRC-
CRC+	CRC+

Exemplo 3: escrita do comando de *reset* (bit 107), em uma Soft-Starter no endereço 1.

- Endereço: 1 = 01h (1 byte)
- Número do bit: 107 = 006Bh (2 bytes)
- Valor para o bit: *reset* = 1, logo o valor que deve ser escrito é FF00h

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	01h	Endereço do escravo	01h
Função	05h	Função	05h
Número do bit (high)	00h	Número do bit (high)	00h
Número do bit (low)	6Bh	Número do bit (low)	6Bh
Valor para o bit (high)	FFh	Valor para o bit (high)	FFh
Valor para o bit (low)	00h	Valor para o bit (low)	00h
CRC-	FDh	CRC-	FDh
CRC+	E6h	CRC+	E6h

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

5.4.4. 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 4: escrita do comando lógico (variável básica 3), com os comando de habilita rampa e habilita geral, para a Soft-Starter SSW-07/SSW-08 no endereço 3.

- Endereço: 3 = 03h (1 byte)
- Número da variável: VB03, endereçada no registrador 5003 = 138Bh (2 bytes)
- Valor para a variável: habilita rampa → comando em 1 (bit 0)
  habilita geral → comando em 1 (bit 1)
  logo, valor para o comando = 0003h (2 bytes)

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	03h	Endereço do escravo	03h
Função	06h	Função	06h
Registrador (high)	13h	Registrador (high)	13h
Registrador (low)	8Bh	Registrador (low)	8Bh
Valor (high)	00h	Valor (high)	00h
Valor (low)	03h	Valor (low)	03h
CRC-	BCh	CRC-	BCh
CRC+	87h	CRC+	87h

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

5.4.5. Função 15 – *Write Multiple Coils*

Esta função permite escrever valores para um grupo de bits (*coils*), que devem estar em seqüência numérica. Também pode ser usada para escrever um único bit (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 bit inicial (byte high)	Endereço do bit inicial (byte high)
Endereço do bit inicial (byte low)	Endereço do bit inicial (byte low)
Número de bits (byte high)	Número de bits (byte high)
Número de bits (byte low)	Número de bits (byte low)
Campo Byte Count (no. de bytes de dados)	CRC-
Byte 1	CRC+
Byte 2	
Byte 3	
etc...	
CRC-	
CRC+	
Endereço do escravo	

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

Exemplo 5: escrita dos bits 100 e 101 para habilitar a rampa e habilitar geral uma Soft-Starter SSW-07/SSW-08 no endereço 20

- Endereço: 20 = 14h (1 byte)
- Número do primeiro bit: 100 = 0064h (2 bytes)
- Número de bits: 2 = 0002h (2 bytes)
- Valor para os bits: os dois bits devem ser colocados em 1, então valor = 03h (1 byte)

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	14h	Endereço do escravo	14h
Função	0Fh	Função	0Fh
Bit inicial (byte high)	00h	Bit inicial (byte high)	00h
Bit inicial (byte low)	64h	Bit inicial (byte low)	64h
No. de bits (byte high)	00h	No. de bits (byte high)	00h
No. de bits (byte low)	02h	No. de bits (byte low)	02h
Byte Count	01h	CRC-	97h
Valor para os bits	03h	CRC+	10h
CRC-	2Eh		
CRC+	6Dh		

5.4.6. 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 hexadecimais, 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 6: escrita do valor 2 em P313 e valor 5 em P314, para uma Soft-Starter SSW-07/SSW-08 no endereço 15.

- Endereço: 15 = 0Fh (1 byte)
- Número do primeiro parâmetro: P313, endereçado no registrador 313 = 139h (2 bytes)
- Valor para o primeiro parâmetro: 2 = 0002h (2 bytes)
- Valor para o segundo parâmetro: 5 = 0005h (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 (high)	01h
Registrador inicial (low)	39h	Registrador (low)	39h
No. de registradores (high)	00h	Valor (high)	00h
No. de registradores (low)	02h	Valor (low)	02h
Byte Count	04h	CRC-	91h
P313 (high)	00h	CRC+	17h
P313 (low)	02h		
P314 (high)	00h		
P314 (low)	05h		
CRC-	68h		
CRC+	6Ah		

5.4.7. Função 43 – *Read Device Identification*

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

Pergunta (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 a Soft-Starter apenas informações básicas estão disponíveis, formadas por três objetos:

- Objeto 0x00 - VendorName: sempre 'WEG'.
- Objeto 0x01 - ProductCode: formado pelo código do produto (SSW-07) mais a corrente nominal do drive (ex. 'SSW-07 85A').
- Objeto 0x02 - MajorMinorRevision: indica a versão de firmware do drive, no formato 'VX.XX'.

O código de leitura indica quais as categorias de informações estão sendo lidas, e se os objetos estão sendo acessados em seqüência ou individualmente. No caso, a SSW-07 suporta os códigos 01 (informações básicas em seqüência), e 04 (acesso individual aos objetos). Os demais campos para a SSW-07 possuem valores fixos.

Exemplo 7: leitura das informações básicas em seqüência, a partir do objeto 00h, de uma Soft-Starter SSW-07 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	00h	Conformity Level	51h
CRC-	70h	More Follows	00h
CRC+	77h	Próximo Objeto	00h
		Número de objetos	03h
		Código do Objeto	00h
		Tamanho do Objeto	03h
		Valor do Objeto	'WEG'
		Código do Objeto	01h
		Tamanho do Objeto	0Ch
		Valor do Objeto	'SSW-07 85A'
		Código do Objeto	02h
		Tamanho do Objeto	05h
		Valor do Objeto	'V1.20'
		CRC-	CBh
		CRC+	5Eh

Neste exemplo, o valor dos objetos não foi representado em hexadecimal, mas sim utilizando os caracteres ASCII correspondentes. Por exemplo, para o objeto 00h, o valor 'WEG' foi transmitido como sendo três caracteres ASCII, que em hexadecimal possuem os valores 57h ('W'), 45h ('E') e 47h ('G').

5.4.8. 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, a Soft-Starter SSW-07/SSW-08 poderá ou não enviar resposta para o mestre. Quando o mestre envia uma mensagem para um escravo configurado em um determinado endereço da rede, o escravo 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 drive 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).
  - Comando não está habilitado para ser executado via serial

NOTA!
É importante que seja possível identificar no mestre qual o tipo de erro ocorrido, para que seja possível 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 8: mestre solicita para o escravo no endereço 1 a escrita no parâmetro 89 (parâmetro inexistente):

Pergunta (Mestre)		Resposta (Escravo)	
Campo	*Valor*	*Campo*	*Valor*
Endereço do escravo	0x01	Endereço do escravo	0x01
Função	0x06	Função	0x86
Registrador (high)	0x00	Código de erro	0x02
Registrador (low)	0x59	CRC-	0xC3
Valor (high)	0x00	CRC+	0xA1
Valor (low)	0x00		
CRC-	0x59		
CRC+	0xD9		

APÊNDICES

Apêndice A - 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 4.

```
/* 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 B - 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 0xFFFF. 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 0xA001 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 primeira (CRC-) e em seguida a parte mais significativa (CRC+).