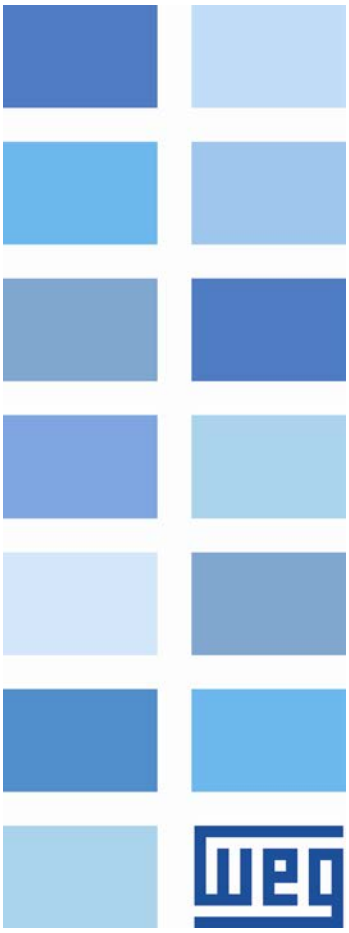
Serial

SIW700

Manual de Comunicação Serial RS232 / RS485

Idioma: Português



Manual da Comunicação Serial RS232 / RS485

Série: SIW700

Idioma: Português

Número do Documento: 10003748025 / 00

Data de Publicação: 08/2015

Sumário

Sobre o Manual

Este manual fornece a descrição necessária para a operação do inversor solar SIW700 utilizando as interfaces seriais RS232 ou RS485. Este manual deve ser utilizado em conjunto com manual do usuário do SIW700.

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 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, que é o protocolo Modbus-RTU.

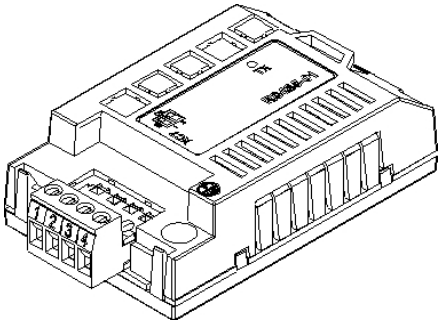
A seguir serão apresentadas características das interfaces seriais RS232 e RS485 disponíveis para o inversor SIW700, bem como o protocolo Modbus-RTU. O módulo RS485 é instalado no SIW700 como padrão de fábrica. Existe a possibilidade de substituir este módulo por um módulo RS232.

2 Módulo de Comunicação

Cada inversor solar SIW700 tem um modulo de comunicação RS485.

2.1 RS485

2.1.1 Kit RS485-01



- ☑ 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.

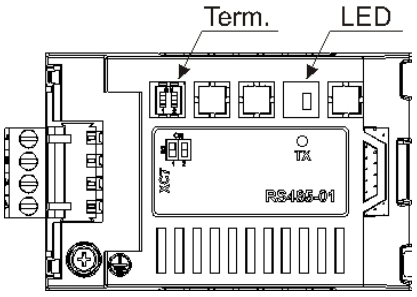
2.1.2 Pinagem do Conector

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

Tabela 2.1 - Pinagem do conector de 4 vias para RS485

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

2.1.3 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.

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

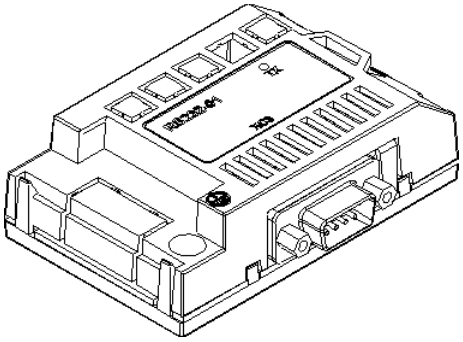
2.1.4 Conexão com a Rede RS485

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.

2.2 RS232

2.2.1 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 SIW700 e o mestre da rede (ponto-a-ponto).
- ☑ Distância máxima para ligação dos dispositivos de 10 metros.

2.2.2 Pinagem do Conector

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

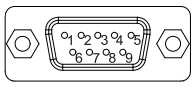


Tabela 2.2 - Pinagem do conector DB9 para RS232

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	-

2.2.3 Indicações e Chaves

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

2.2.4 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.

2.2.5 Cabos para Ligação em RS232

Caso seja desejado, a WEG pode fornecer os seguintes cabos para ligação em RS232 entre o inversor SIW700 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.

3 Parametrização do Inversor

A seguir serão apresentados apenas os parâmetros do inversor solar SIW700 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 inversor em Modo Ready
Net	Parâmetro visível através da HMI se o inversor possuir interface de rede instalada – RS232, RS485 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

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 = 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 = Reservado; 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 = Alarme; 1 = Modo Shutdown; 2 = 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 = Alarme	O inversor deverá gerar o alarme A128 se a comunicação falhar pelo tempo programado em P0314 e deverá limpar o alarme se a comunicação voltar.
1 = Modo Shutdown	O inversor deverá se desconectar da rede se a comunicação falhar pelo tempo programado em P0314 e deverá se conectar novamente se a comunicação voltar.
2 = Falha	O inversor deverá gerar a falha F128 se a comunicação falhar pelo tempo programado em P0314, conectando-se novamente depois do tempo de auto-reset (P0340). Se a comunicação não tiver voltado após P0340 uma nova falha será gerada quando o inversor for habilitado durante o processo de reconexão.

Para a interface serial, é considerado erro de comunicação apenas o evento de *timeout* da interface serial – alarme A128/F128. Este *timeout* é programado através do parâmetro P0314.

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
  - 113 Serial RS232 / 485

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. Somente então a ação programada no P0313 será executada mostrado o alarme A128 (P0313=0) ou falha F128 (P0313=2). Se P0313=1 o inversor entrará em modo Shutdown.

Após energizado, o inversor começará a contar este tempo a partir do primeiro telegrama válido recebido. Configurando o valor 0,0 essa função é desabilitada.

P0316 – Estado da Interface Serial

Faixa de Valores: 0 = Inativo
1 = Ativo
2 = Erro *Watchdog*

Padrão: -

Propriedades: RO

Grupos de acesso via HMI:

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

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 serial 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 *Watchdog*	Interface serial ativa, mas detectado erro de comunicação serial – A128/ F128.

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. A comunicação Modbus-RTU do inversor SIW700 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.

4.1 Modos de Transmissão

Na especificação do protocolo estão definidos dois modos de transmissão: ASCII e RTU. Os modos definem a forma como são transmitidos os bytes da mensagem. Não é possível utilizar os dois modos de transmissão na mesma rede.

O inversor solar SIW700 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.

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.

No SIW700, os dados relativos aos parâmetros estão disponibilizados como registradores do tipo *holding* (referenciados a partir do endereço 40000 ou '4*x*').

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 4.4).

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 nos apêndices B e C.

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[2]). 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:

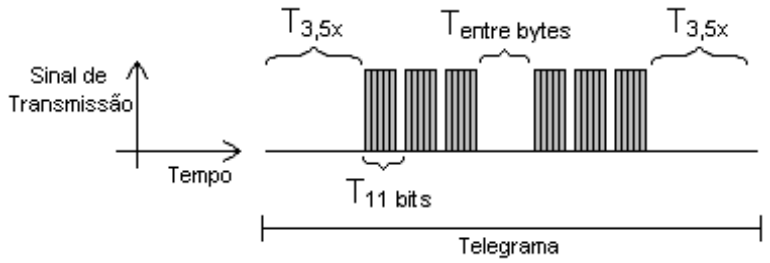


Tabela 4.1 - Taxas de comunicação e tempos envolvidos na transmissão de telegramas

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}).

[2] 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.

4.3 Operação do SIW700 na Rede Modbus-RTU

O SIW700 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 0).
- ☑ Permite a parametrização e controle do inversor através do acesso a parâmetros.

4.3.1 Funções Disponíveis e Tempos de Resposta

Na especificação do protocolo Modbus-RTU são definidas funções utilizadas para acessar diferentes tipos de registradores. No SIW700, os parâmetros foram definidos como sendo registradores do tipo *holding*. Para acessar estes registradores, foram disponibilizados os seguintes serviços (ou funções):

- ☑ *Read Coils*[3]
  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 SIW700, 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.

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

4.3.2 Endereçamento dos Dados e Offset

O endereçamento dos dados no SIW700 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 4.2 - 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..

4.4 Descrição Detalhada das Funções

Neste item é feita uma descrição detalhada das funções disponíveis no SIW700 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 SIW700 para obter a resolução utilizada para cada parâmetro.

4.4.1 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 tensão da linha (P0002) e corrente de linha (P0003) do SIW700 no endereço 1 (supondo P0002 = 220 V e P0003 = 20,9 A).

☑ Endereço: 1 = 01h (1 byte)

- ☑ Número do primeiro parâmetro: 2 = 0002h (2 bytes)
- ☑ Valor do primeiro parâmetro: 220 = 00DCh (2 bytes)
- ☑ Valor do segundo parâmetro: 209 = 00D1h (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)	00h
No. de registradores (high)	00h	P002 (low)	DCh
No. de registradores (low)	02h	P003 (high)	00h
CRC-	65h	P003 (low)	D1h
CRC+	CBh	CRC-	FBh
		CRC+	95h

4.4.2 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 na limitação de potência ativa (P0133) em 50,0% para o SIW700 no endereço 3.
- ☑ Endereço: 3 = 03h (1 byte)
- ☑ Número do parâmetro: 133 = 0085h (2 bytes)
- ☑ Valor para o parâmetro: 0500 = 01F4h (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)	85h	Registrador (low)	85h
Valor (high)	01h	Valor (high)	01h
Valor (low)	F4h	Valor (low)	F4h
CRC-	99h	CRC-	99h
CRC+	D6h	CRC+	D6h

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

4.4.3 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: seleção do Modo de Operação=FP Fixo (P0965=1) e configuração do FP Fixo=0,950 (P0966=950) de um SIW700 no endereço 15.

☑ Valores convertidos para hexadecimal:
- Endereço: 15 = 0Fh (1 byte)
- Número do primeiro parâmetro: 965 = 03C5h (2 bytes)
- Valor para o primeiro parâmetro: 1 = 0001h (2 bytes)
- Valor para o segundo parâmetro: 950 = 03B6h (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)	03h	Registrador (high)	03h
Registrador inicial (low)	C5h	Registrador (low)	C5h
No. de registradores (high)	00h	Valor (high)	00h
No. de registradores (low)	02h	Valor (low)	02h
Byte Count	04h	CRC-	50h
P0965 (high)	00h	CRC+	9Fh
P0965 (low)	01h		
P0966 (high)	03h		
P0966 (low)	B6h		
CRC-	CFh		
CRC+	5Eh		

4.4.4 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 sequência de caracteres ASCII. Para o SIW700, apenas informações básicas estão disponíveis, formadas por três objetos:

- ☑ Objeto 00h – VendorName: Sempre 'WEG'.
- ☑ Objeto 01h – ProductCode: Formado pelo código do produto (SIW700) mais a tensão e corrente nominal do inversor (ex. 'SIW700 330 V 2915 A').
- ☑ 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 sequência ou individualmente. No caso, o SIW700 suporta os códigos 01 (informações básicas em sequência), e 04 (acesso individual aos objetos). Os demais campos são especificados pelo protocolo e para o SIW700 possuem valores fixos.

Exemplo 4: leitura das informações básicas em sequência, a partir do objeto 01h, de um SIW700 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-	B1h	More Follows	00h
CRC+	B7h	Próximo Objeto	00h
		Número de objetos	02h
		Código do Objeto	01h
		Tamanho do Objeto	1Ch
		Valor do Objeto	'SIW700 330 V 2915 A'
		Código do Objeto	02h
		Tamanho do Objeto	05h
		Valor do Objeto	'V1.50'
		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 'V1.50' foi transmitido como sendo cinco caracteres ASCII, que em hexadecimal possuem os valores 56h ('V'), 31h ('1'), 2Eh ('.'), 35h ('5') e 30h ('0').

4.4.5 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 SIW700 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 33 (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)	21h	CRC-	C3h
Valor (high)	00h	CRC+	A1h
Valor (low)	00h		
CRC-	D9h		
CRC+	C0h		

5 Falhas e Alarmes Relacionados com a Comunicação Serial

A128/F128 – 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 F128, 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 0.

```
/* 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 0.

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+).