**Protocolo de Comunicação Modbus RTU/ASCII**

***VERSÃO 1.0 – 28/08/2000***

# Parâmetros de programação dos indicadores ALFA Instrumentos
# Protocolo de comunicação ModBus RTU / ASCII

## 1. Introdução

Este documento destina-se a programadores e/ou profissionais que de uma forma ou de outra, serão responsáveis pela programação de um PLC **Modbus**, nativo ou não, sendo desejável que tais profissionais tenham conhecimentos de bases numéricas (decimal, hexadecimal, código ASCII), do funcionamento básico de um PLC pois as informações contidas neste manual serão empregadas na sua programação, bem como dos conceitos de comunicação de dados, necessários para o bom funcionamento do programa.

É descrito em detalhes o protocolo de comunicação serial assíncrona padrão **ModBus**, implementado nos indicadores da ALFA Instrumentos, modelos **3104B Modbus** e **3107 Modbus**. Este protocolo deve ser utilizado para realizar a comunicação entre estes indicadores e qualquer controlador ou computador que esteja capacitado a transferir dados através do **protocolo ModBus**, nos modos **ASCII** ou **RTU**.

A interface serial dos indicadores ALFA pode operar tanto nos padrões elétricos **RS232** como **RS485**, configuráveis por hardware. Nestas condições o protocolo **ModBus** embarcado nos indicadores funciona normalmente pois independe do meio elétrico.

### 1.1. Terminologia

Descrição de alguns termos empregados ao longo deste documento:

PLC/CLP = Programable Logic Controller (Controlador Lógico Programável), dispositivo que controla e processa todas as informações de um sistema industrial

protocolo de comunicação = realização da troca de informações (mensagens) entre 2 ou mais dispositivos seguindo uma normalização específica, dependendo do tipo do protocolo

dispositivo = qualquer tipo de equipamento conectado a uma rede com capacidade de enviar e receber mensagens

mensagem = conjunto de dados que juntos, compõem uma série de informações passadas de um dispositivo a outro

mestre = dispositivo que inicia a transmissão de uma mensagem

escravo = dispositivo que responde a uma mensagem enviada por um dispositivo mestre

caracter = dado propriamente dito contido numa palavra de dados

palavra de dados (ou dados) = informação contendo o caracter, start bit, bits de paridade e stop bits

framing = conjunto de palavras de dados que compõem uma mensagem

barramento = meio físico por onde trafegam as mensagens

campo = uma mensagem é composta por vários campos, cada qual com uma informação específica

registrador = região de memória interna de um dispositivo

H = abreviação da base numérica hexadecimal

(+) = parte mais significativa da informação

(-) = parte menos significativa da informação

time-out = intervalo de tempo antes de se abortar um processo de comunicação

retry (retries) = retransmissão de informações / mensagens

aplicação = programa que estiver sendo executado no terminal mestre ou programa principal do indicador responsável pela interpretação dos comandos ModBus

## 2. Conceitos de Mestre / Escravo e Enlace/Aplicação

Dois conceitos importantes numa comunicação de dados dizem respeito ao equipamento mestre e escravo. Em qualquer interligação na qual esteja sendo utilizado um protocolo necessariamente deverá existir um e somente um equipamento mestre e pelo menos um escravo.

Define-se como mestre o equipamento responsável por toda a iniciativa do processo de comunicação, ou seja, uma troca de dados entre os equipamentos, quer para transmissão ou recepção, é iniciado e finalizado pelo mestre. Defini-se como escravo o equipamento que, no processo de comunicação, só realiza alguma função sob solicitação e controle do mestre.

Um equipamento pode ser mestre ou escravo dependendo somente da sua aplicação no sistema. Isto quer dizer que em um determinado sistema, um equipamento pode ser configurado como mestre. Sendo que em outro sistema, o mesmo equipamento pode estar configurado com escravo.

Na comunicação entre um equipamento mestre e um escravo existem duas situações possíveis:

1. mestre deseja enviar/receber dados para/do escravo
2. escravo deve enviar/receber dados para/do escravo

No primeiro caso, como o mestre tem o poder de iniciar o processo de comunicação, ele envia a qualquer momento, de acordo com suas necessidades e independente do estado em que se encontrar o escravo, uma mensagem requisitando ao escravo a realização de uma determinada função. Ao receber a mensagem enviada pelo mestre, o escravo executa a função solicitada e envia uma resposta contendo o resultado desta função. Este processo da comunicação é chamado de **select**.

No segundo caso, como o escravo não pode tomar a iniciativa de começar o processo de comunicação, ele aguarda que o mestre envie uma mensagem. Ao receber a mensagem, o escravo realiza a função solicitada e envia uma resposta contendo o resultado. Este processo da comunicação é chamado de **polling**.

O nível de enlace se preocupa exclusivamente com os procedimentos a serem seguidos na transmissão e recepção das mensagens, recuperação de erros, sincronismo entre os equipamentos em relação à comunicação, conexão e desconexão, etc., deixando para o nível de aplicação o tratamento das informações relativas ao processo que se deseja supervisionar e/ou controlar.

O controle da comunicação é disciplinado por códigos HEXA ou ASCII (dependendo do protocolo utilizado) transmitidos em uma determinada sequências sendo que os procedimentos referentes ao sincronismo entre as mensagens e equipamentos são implementadas pelos processos de select e polling.

## 3. Protocolo ModBus

A comunicação com os indicadores ALFA Instrumentos através do protocolo ModBus, tanto no modo ASCII como RTU, seguem fielmente as normas estabelecidas pela **Modicon Inc**, desenvolvedora deste protocolo.

### 3.1. Introdução

Este protocolo foi desenvolvido pela Modicon Inc. para ser utilizado como meio de comunicação entre computadores existentes numa rede. Este protocolo basicamente define uma **estrutura de mensagens** composta por **bytes**, que os mais diversos tipos de dispositivos são capazes de reconhecer, independentemente do tipo de rede utilizada.

Durante a comunicação, o protocolo determina como cada dispositivo:

- identificar seu endereço na rede
- reconhecer uma mensagem endereçada a ele
- determinar o tipo de ação a ser executada
- obter toda a informação necessária para executar a ação

Quando há a necessidade de devolver uma resposta ao comando recebido, o dispositivo monta uma mensagem e a envia, mesmo que esta indique um erro na comunicação.

### 3.2. Comunicação

A comunicação é feita através da técnica mestre-escravo, onde apenas um dispositivo (mestre) pode iniciar a comunicação (query). Os outros dispositivos (escravos) respondem enviando os dados solicitados pelo mestre. O mestre pode endereçar individualmente um escravo ou acessar a todos em uma rede através de mensagens em cadeia (broadcast), Apenas o escravo endereçado retorna uma resposta (response) a uma query e nunca são geradas responses quando uma query for do tipo broadcast.

O protocolo ModBus estabelece o formato da query definindo:

- o endereço do escravo (ou código para acesso broadcast)
- o código da função, que indica qual ação deve ser tomada pelo escravo
- parâmetros ou dados pertinentes à função definida
- o campo de checksum para checar a integridade da mensagem enviada

A resposta do escravo é gerada nos mesmos moldes porém, obedecendo o formato correspondente à função recebida pelo mestre que basicamente define:

- a confirmação a função realizada
- parâmetros ou dados pertinentes à função solicitada
- o campo de checksum para checar a integridade da mensagem enviada

Quando ocorrer um erro na comunicação ou se o escravo não estiver apto para atender à função requisitada, ele monta uma mensagem específica (exception) justificando o seu não atendimento.

### 3.3. Modos de transmissão

Os dispositivos podem ser configurados para transmitir em um dos dois modos disponíveis: **ASCII** ou **RTU**. O modo de transmissão define basicamente como os dados serão "empacotados" na mensagem. Estando definido o modo de transmissão, deve-se definir seus parâmetros de comunicação: baud rate, paridade, stop bits. Tanto o modo como os parâmetros de comunicação devem ser os mesmos para **todos** os dispositivos da rede.

#### 3.3.1. Modo ASCII

Quando o dispositivo for configurado para este modo, para cada palavra de dados da mensagem são enviados dois caracteres no padrão **ASCII**. A principal vantagem do modo ASCII é a possibilidade de haver intervalos grandes entre o envio dos dados de uma mesma mensagem. Em relação à **formação da palavra de dados** que comporá o conjunto de dados (**framing**) da mensagem, são adotados alguns critérios.

Como todos os dados são enviados no padrão ASCII, o framing apresentará apenas valores **de 30H à 39H e 41H à 46H,** que correspondem respectivamente aos números **0 à 9 e A à F** no padrão hexadecimal e **0 à 9 e 10 à 15** no padrão decimal.

A quantidade de bits por cada palavra de dados **sempre** será **igual a 10**, independente dos parâmetros de comunicação, que são os seguintes:

- 1 START bit, 7 DATA bits, SEM paridade (não há bit de paridade), 2 STOP bits

- 1 START bit, 7 DATA bits, paridade PAR (1 bit de paridade), 1 STOP bit

- 1 START bit, 7 DATA bits, paridade IMPAR (1 bit de paridade), 1 STOP bit

O campo de **checksum,** usado para checar a integridade da mensagem enviada, é gerada pelo método **LRC**, que será abordado mais adiante neste documento.

#### 3.3.2. Modo RTU

Quando o dispositivo for configurado para este modo, para cada palavra de dados da mensagem é enviado **apenas um** caracter no padrão **HEXADECIMAL**. A principal vantagem do modo RTU em relação ao ASCII é a maior densidade de caracteres que é enviada numa mesma mensagem, aumentando o desempenho da comunicação. Em relação à **formação da palavra de dados** que comporá o conjunto de dados (**framing**) da mensagem, são adotados alguns critérios.

A quantidade de bits por cada palavra de dados **sempre** será **igual a 11**, independente dos parâmetros de comunicação, que são os seguintes:

- 1 START bit, 8 DATA bits, SEM paridade (não há bit de paridade), 2 STOP bits

- 1 START bit, 8 DATA bits, paridade PAR (1 bit de paridade), 1 STOP bit

- 1 START bit, 8 DATA bits, paridade IMPAR (1 bit de paridade), 1 STOP bit

O campo de **checksum,** usado para checar a integridade da mensagem enviada, é gerada pelo método **CRC**, que será abordado mais adiante neste documento.

### 3.4. Framing da mensagem

Na transmissão de uma mensagem ModBus, há **identificadores de início e fim** de framing, específicos para cada um dos modos de transmissão. Este recurso permite aos dispositivos da rede detectarem o início de uma mensagem, ler o seu campo de endereço e determinar qual dispositivo está sendo endereçado (ou todos no caso do acesso ser tipo broadcast) e finalmente, ler todo o conteúdo da mensagem até o seu final. Será visto mais adiante que mensagens podem ser lidas apenas **parcialmente** com posterior geração de códigos de erro (**exceptions**) de acordo com a situação.

#### 3.4.1. Framing no modo ASCII

Neste modo o **início** das mensagens são identificadas através do caracter **dois pontos** ( **:** ), correspondente ao valor ASCII 3AH, e o seu término é identificado pelo conjunto de caracteres **Retorno de Carro** (Carriage Return – **CR**) e **Avanço de Linha** (Line Feed – **LF**), respectivamente com correspondentes em ASCII aos valores 0DH e 0AH.

Os caracteres permitidos para transmissão para todo o resto da mensagem são **0** à **9** e **A** à **F**, respectivamente correspondentes aos caracteres ASCII 30H à 39H e 41H à 46H.

Os dispositivos da rede monitoram continuamente o barramento e quando é detectado o caracter **3AH** tem início a decodificação do próximo campo, que indica para quem é a mensagem que está sendo transmitida.

Intervalos de até **1 segundo** podem ocorrer entre o envio de cada caracter dentro de uma mesma mensagem sendo que para intervalos maiores, o escravo assume ocorrência de erro. A seguir é mostrado um framing típico no modo ASCII.

| **Início de framing** | **Endereço do Escravo** | | **Função ModBus** | **Dados para o Escravo** | **Checksum** | | **Fim de framing** | |
|---|---|---|---|---|---|---|---|---|
| 3AH | char + | char - | 2 chars | N chars | LRC + | LRC - | 0DH | 0AH |

O exemplo a seguir utiliza valores em DECIMAL e respectivos em HEXA e ASCII, usados para ilustrar a formação do framing de dados:

- endereço do indicador: 69 = 45H = 34H 35H

- função ModBus: leitura de registradores: 03 = 03H = 30H 33H

- registrador inicial a ser lido: 11 = 0BH – pela norma ModBus: 000AH = 30H 30H 30H 41H

- número total de registradores a serem lidos: 1 = 0001H = 30H 30H 30H 31H

- Checksum (LRC) gerado: 173 = ADH = 41H 44H

Para os valores acima será gerado o seguinte framing de dados:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 34H | 35H | **30H** | **33H** | 30H | 30H | 30H | 41H | **30H** | **30H** | **30H** | **31H** | 41H | 44H | **0DH** | **0AH** |

### 3.4.2. Framing no modo RTU

Diferentemente do modo ASCII, o modo RTU **não possui** bytes que indiquem início e fim de framing. Para identificar estes campos, não deve haver nenhuma transmissão de dados por um período mínimo, **equivalente a 3.5 vezes o tamanho da palavra de dados** (**silent**).

Por exemplo, suponha que a taxa de transmissão seja de **19200 bps**. Para esta taxa, o tempo total para envio de 1 palavra de dados (11 bits) é de **572.9 us** (11 x (1 / 19200)) portanto, para identificar um início e/ou término de framing, não deve haver transmissão por um período mínimo de **2.005 ms** (3.5 x 572.9 us).

Para o restante da mensagem são aceitos **todos** os caracteres hexadecimais.

Os dispositivos ficam monitorando o barramento e checando intervalos silent que, após detectados, dá início à recepção da mensagem, de maneira similar ao modo ASCII. Após a recepção de toda a mensagem, deve ser gerado pelo mestre um intervalo silent similar ao do início da mensagem, caracterizando o fim da mesma.

Neste modo, toda a mensagem deve ser enviada de maneira **contínua**. Se um intervalo **maior que 1.5 vezes o tamanho da palavra de dados** for detectado antes que toda a mensagem tenha sido recebida, o escravo descarta os dados já recebidos da mensagem atual e assume que o próximo caracter será o campo de endereço de uma nova mensagem. De modo similar, se uma nova mensagem for recebida em um intervalo **menor** que o intervalo silent, o escravo assume que esta mensagem é uma continuação da última mensagem recebida. Esta condição irá gerar um erro pois o campo **CRC** não corresponderá aos dados enviados na mensagem. A seguir é mostrado um framing típico no modoRTU.

| **Início de framing** | **Endereço do Escravo** | **Função ModBus** | **Dados para o Escravo** | **Checksum** | | **Fim de framing** |
|---|---|---|---|---|---|---|
| TInício | 1 char | 1 char | N chars | CRC - | CRC + | TFim |

Os campos **Endereço do Indicador** e **Função ModBus** possuem um **único byte** ao invés de 2 como no modo ASCII) e outra particularidade está na **sequência** de envio dos bytes de checksum da mensagem. O primeiro byte enviado é o menos significativo e depois o mais significativo.

O exemplo a seguir utiliza os mesmos valores do exemplo empregado no modo ASCII:

- endereço do indicador: 69 = 45H

- função ModBus: leitura de registradores: 03 = 03H

- registrador inicial a ser lido: 11 = 0BH – pela norma ModBus: 000AH

- número total de registradores a serem lidos: 1 = 0001H

- Checksum (CRC) gerado: 19627 = 4CABH – pela norma ModBus RTU = ABH 4CH

Para os valores acima será gerado o seguinte framing de dados:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **45H** | 03H | **00H** | **0AH** | 00H | 01H | **ABH** | **4CH** |

### 3.5. Campo Endereço do Escravo

Este campo possui **dois caracteres**, modo ASCII, ou **oito bits**, modo RTU. A faixa de endereços válidos para os escravos é de **0 à 247**. **Individualmente** os escravos são endereçados de 1 à 247 e o **endereço 0** é reservado para acesso tipo **broadcast**, que é o único endereço, além do próprio, que **todos** os escravos reconhecem.

Um mestre endereça o(s) escravo(s) colocando o endereço do escravo no campo de endereço da mensagem. Quando o escravo envia a response, ele coloca seu próprio endereço no campo de endereço da mensagem para sinalizar ao mestre qual escravo está respondendo. Para acessos tipo broadcast não há response por parte de **nenhum** escravo.

### 3.6. Campo Função ModBus

Este campo possui **dois caracteres**, modo ASCII, ou **oito bits**, modo RTU. A faixa de valores válidos é de **1 à 255**. Destes valores, a maioria é compatível para todos os CLPs enquanto alguns estão disponíveis apenas para os modelos mais novos e outros estão reservados para uso futuro. As funções disponíveis para os indicadores ALFA são abordadas no item 4.

Quando uma mensagem é recebida pelo escravo, este campo indica qual ação deve ser tomada, por exemplo, ler e/ou programar um ou mais sensores, ler e/ou programar um ou mais registradores, ler o status do escravo.

Na response do escravo este campo é usado para informar ao mestre se a ação foi executada com sucesso ou se ocorreu algum tipo de erro (**exception**). Para funções executados com sucesso o escravo simplesmente devolve o próprio código da função e quando ocorreu algum erro, o escravo devolve o código da função com o seu bit mais significativo em nível 1.

Por exemplo, se ocorrer um erro na escrita de um registrador, ao invés do escravo retornar o código 06H – 0000 0110 (código de escrita em um único registrador), ele retorna o código 86H – 1000 0110. Juntamente com esta sinalização, o escravo passa no **campo de dados** um outro código que identifica a causa do erro. Estes códigos serão vistos mais adiante.

É responsabilidade do **mestre** gerenciar e analisar mensagens de exception, reenviando a mensagem que originou o erro ou enviando uma função de diagnóstico para identificar melhor a causa do erro.

### 3.7. Campo Dados para o Escravo

Este campo é formado por conjuntos de 2 dígitos hexadecimais, variando de 00H à FFH. No modo ASCII é composto por dois caracteres ASCII e no modo RTU por um único byte.

#### 3.7.1. Comandos sem erro

As informações contidas neste campo estão relacionadas ao tipo de função definido no campo **função ModBus**, como por exemplo, quantos sensores ou dados devem ser escritos ou lidos, os dados a serem programados em determinados registradores, que também são definidos neste campo, e a quantidade de bytes que estão sendo enviados na mensagem.

Se não ocorrer erro na realização do comando, este campo da mensagem conterá as informações relativas ao comando enviado, caso contrário, conterá um **código de exception**, identificando o motivo que causou o erro e orientando o mestre na execução do próximo comando.

### 3.7.2. Comandos com erro

Quando o mestre envia uma mensagem para um escravo, é esperada uma resposta normal porém, pode ocorrer uma das seguintes possibilidades:

- se o escravo receber uma mensagem sem nenhum tipo de erro de comunicação e a mesma poder ser tratada corretamente, o escrevo reponde ao mestre os dados pertinentes ao comando recebido;
- se o escravo não receber a mensagem devido a um erro de comunicação, **nenhuma** mensagem é retornada ao mestre e provavelmente o mestre executará um procedimento de time-out;
- se o escravo receber uma mensagem mas detectar que houve um erro de comunicação (framing, checksum), nenhuma mensagem é retornada ao mestre e provavelmente o mestre executará um procedimento de time-out;
- se o escravo receber uma mensagem sem erro de comunicação mas não puder atendê-la, ele retornará uma mensagem de exception informando ao mestre a natureza do erro;

As mensagens de exception possuem dois campos que a diferenciam de uma mensagem normal.

#### 3.7.2.1. Campo Código da Função

Em uma resposta normal, o escravo repete o código da função no respectivo campo da sua mensagem, sendo que nestes casos, **todos** os códigos possuem o bit mais significativo **igual a 0**. Quando ocorre um erro, porém, este mesmo bit passa a ter valor igual a 1 caracterizando uma resposta com exception. Uma vez detectada, o mestre passa a analisar o próximo campo da mensagem, **Campo de Dados,** que deve conter o código do motivo que originou a exception.

#### 3.7.2.2. Campo de Dados

Em mensagens sem erro o escravo utiliza este campo para retornar informações pertinentes ao comando solicitado. Quando ocorre um erro, este campo contém o código da causa da exception. A seguir estão relacionados os códigos de exception e os respectivos significados.

| código | identificação | Significado |
|---|---|---|
| 1 | Função inválida | A função ModBus solicitada pelo mestre não está implementada pelo escravo. |
| 2 | Sensor ou registrador inválido | O escravo não possui o(s) sensor(es) ou registrador(es) especificado(s) no comando enviado pelo mestre. |
| 3 | Valor de dado inválido | O valor de algum dado(s) contido no **Campo de Dados** é inválido. |
| 4 | Falha no dispositivo | Ocorrência de erro por parte do escravo durante a execução do comando solicitado pelo mestre. |
| 5 | Estado de espera | O escravo reconheceu o comando enviado pelo mestre mas o notifica de que o mesmo será processado num período de tempo maior que o normal. Este tipo de código é enviado para evitar a ocorrência de time-out por parte do mestre. Nestas condições, o mestre pode ficar monitorando as atividades do escravo até a realização do comando. |
| 6 | Dispositivo ocupado | O escravo está ocupado atendendo a outro comando. O mestre pode retransmitir a mensagem mais tarde quando o escravo já tiver completado o comando em execução. |
| 7 | Não reconhecimento | O escravo não conseguiu executar o comando. É gerado quando o mestre envia um comando através das funções 13 ou 14. |
| 8 | Erro de paridade na memória | O escravo detectou erro de paridade na leitura da sua memória estendida. |

O exemplo a seguir adota o modo de comunicação RTU. O mestre envia um comando para o escravo número 69H da rede, solicitando que seja programado o seu registrador 0058H com o valor 05AFH. A mensagem enviada para este comando é a seguinte:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **69H** | 06H | **00H** | **58H** | 05H | AFH | **43H** | **DDH** |

onde:

- 69H = endereço do escravo
- 06H = código da função ModBus de escrita em um único registrador
- 00H 58H = número do registrador interno do escravo
- 05H AFH = valor a ser programado no registrador 0058H do escravo
- 43H DDH = checksum da mensagem enviada pelo mestre

Antes de executar o comando de escrita, o escravo identifica que **não existe** o registrador 0058H em seu dispositivo e retorna a seguinte mensagem de exception:

| | | | | |
|---|---|---|---|---|
| 69H | **86H** | **02H** | 42H | 7DH |

onde:

- 69H = endereço do escravo
- **86H** = código da função ModBus de escrita em um único registrador com sinalização de erro
- **02H** = código de exception indicando que não existe o registrador solicitado
- 42H 7DH = checksum da mensagem a ser enviada pelo escravo

### 3.8. Campo Checksum

Há dois tipos de cálculo de checagem da integridade (**checksum**) de dados usados pelo protocolo ModBus: checagem da **paridade**, que pode ou não ser aplicada para cada caracter da mensagem, e checagem do **framing**, que pode ser do tipo **LRC** – aplicado a todos os **campos** de uma mensagem no modo **ASCII**, ou do tipo **CRC** – aplicado ao conteúdo de **toda** a mensagem no modo **RTU**.

Tanto a checagem de paridade como a de framing é gerada ora no mestre ora no escravo, antes da transmissão. Ora o escravo ora o mestre, checa o conteúdo de cada caracter durante a recepção.

Todo mestre é configurado para esperar por um intervalo de tempo pré determinado antes de abortar a comunicação, chamado de **time-out**. Este time-out deve ser programado para ser longo o bastante para dar tempo ao escravo de responder aos queries de maneira normal. Se o escravo detectar um erro de transmissão, a mensagem não mais será validada e **não** será enviada nenhuma response ao mestre. Assim, o time-out será atingido, o que permitirá ao mestre gerenciar a ocorrência do erro. Observe que uma mensagem endereçada a um dispositivo escravo **não existente** na rede também acarretará num time-out.

#### 3.8.1. Checagem da paridade

Esta checagem ocorre através da configuração de um **único bit** na palavra de dados de um caracter (**itens 3.3.1** e **3.3.2**) e pode ser configurada para paridade **PAR**, **ÍMPAR**, ou simplesmente para não ser aplicada, neste caso sendo configurada para **SEM** paridade.

Se for configurada paridade PAR ou ÍMPAR, será contada a quantidade de **bits em nível 1** do caracter a ser enviado: **7 bits** no modo **ASCII** e **8 bits** no modo **RTU**. Dependendo da quantidade total de bits em nível 1, o bit de paridade d palavra de dados será programado para 0 ou 1.

Vamos supor que o caracter **0111 1011** tenha que ser transmitido em modo RTU. A quantidade total de bits em nível 1 é **PAR**. Se fosse configurada paridade PAR, o bit de paridade da palavra de dados seria programado com **valor 0**, fazendo a quantidade total de bits em nível 1 ser **igual a um número par**, neste caso, seis. Caso fosse configurada paridade **ÍMPAR**, o bit de paridade seria programado com **valor 1**, fazendo a quantidade total de bits em nível 1 ser **igual a um número ímpar**, no caso, sete.

Quando a mensagem é transmitida, o bit de paridade é calculado e aplicado a toda palavra de dado. O escravo, ao receber a mensagem, conta a quantidade de bits em nível 1 de cada caracter e sinaliza um erro caso não seja igual ao bit de paridade especificado na respectiva palavra de dados do caracter analisado. Observe que **todos** os dispositivos conectados a uma mesma rede **devem** usar a **mesma** configuração de paridade.

O bit de paridade porém, não garante que não houve erro na transmissão de um caracter. Suponha que o mesmo caracter enviado: **0111 1011** seja recebido com o valor **0001 1011**. A quantidade de bits em nível 1 **permanece a mesma**, ou seja, PAR, mas ocorreu uma alteração do conteúdo durante a transmissão, portanto, a ocorrência de um erro, mas **não detectado** pelo método de checagem da paridade. Para contornar este problema, foram desenvolvidos métodos que analisam não o caracter isoladamente **nas toda a mensagem**, garantindo assim uma maior integridade na checagem dos dados transmitidos.

Quando é configurada a opção **SEM** paridade, o bit de paridade **não é transmitido** e sim, um **STOP BIT adicional,** para manter a quantidade de bits da palavra de dados: **10 bits** no modo **ASCII** e **11 bits** no modo **RTU**.

### 3.8.2. Checagem do framing - LRC

Quando é utilizado o modo de transmissão ASCII, o método de cálculo de checksum adotado é o LRC, Longitudinal Redundancy Check, que calcula o conteúdo **dos campos** da mensagem **exceto** os caracteres de identificação de início: 3AH - e fim de mensagem: 0DH / 0AH.

O valor gerado por este cálculo é de 8 bits portanto, possui dois caracteres ASCII para representá-lo, sendo que na composição final do campo, o caracter **mais significativo** é enviado **primeiro**. Este é o penúltimo campo da mensagem antes do identificador de término da mesma. O escravo realiza o cálculo do LRC **durante a recepção** da mensagem e ao final, compara seu valor com o do LRC enviado no campo de checksum do mestre. Se não forem iguais caracteriza-se um erro de comunicação, **não sendo gerada** mensagem de exception e o sistema aguarda a ocorrência do time-out.

O LRC é calculado adicionando-se sucessivamente os 8 bits **dos campos** da mensagem, **descartando-se** eventuais bits de estouro (**carry/overflow bits**), e submetendo o **resultado final** à lógica de **complemento de dois**. Como o LRC é um valor de **8 bits**, é quase certo que a soma sucessiva dos bytes exceda o valor máximo de 255: 1111 1111. Como não há o nono bit para o cálculo do LRC, este é simplesmente descartado.

Apesar de eficiente, não é um método seguro pois analisa apenas os valores dos campos e não o que realmente é enviado na mensagem, **fisicamente falando**. No exemplo do **item 3.4.1**, não é feita a soma dos bytes **34H** e **35H**, relativos ao Campo de Endereço, ao invés, considera-se o seu valor numérico, ou seja, **69 decimal**. O mesmo ocorre para o campo Função ModBus. Já nos campos Endereço do Registrador Inicial: 30H 30H 30H 41H e Quantidade de Registradores: 30H 30H 30H 31H, campos de 16 bits, soma-se **individualmente** seus valores numéricos dos 8 bits mais significativos e em seguida, os menos significativos, ou seja, (0 + 10) e (0 + 1).

Como regra geral, o procedimento para o cálculo do LRC é o seguinte:

1. adiciona-se todos os bytes dos campos da mensagem, **exceto** os campos de Início e Fim de mensagem, descartando-se todos os bits de estouro;
2. subtrai-se o valor obtido na soma do item 1 de 255 decimal (FFH), que nada mais é do que o método **complemento de 1**, que resulta na inversão simples dos valores dos bits;
3. adiciona-se 1 ao valor obtido no item 2, caracterizando o **complemento de 2.**

Para ilustrar melhor o cálculo do LRC, analisemos a seguinte mensagem e os campos a serem somados:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 34H | 35H | **30H** | **33H** | 30H | 30H | 30H | 41H | **30H** | **30H** | **30H** | **31H** | 41H | 44H | **0DH** | **0AH** |

1. soma sucessiva:

| Campo analisado | Valores em HEXA | | Valor a ser somado |
|---|---|---|---|
| Campo de Endereço | 34H | 35H | **69 / 45H** |
| Campo Função ModBus | 30H | 33H | **3 / 03H** |
| Endereço Registrador Inicial (+) | 30H | 30H | **0 / 00H** |
| Endereço Registrador Inicial (-) | 30H | 41H | **10 / 0AH** |
| Quantidade de Registradores (+) | 30H | 30H | **0 / 00H** |
| Quantidade de Registradores (-) | 30H | 31H | **1 / 01H** |
| **Valor resultante da soma** | | | **83 / 53H** |

2. complemento de 1:

**FFH – 53H = ACH**

3. complemento de 2:

**ACH + 01H = ADH**

Como o modo de transmissão é o ASCII, o valor do LRC calculado ADH equivale a 41H, que representa a **letra A** e 44H, que representa a **letra D**.

### 3.8.3. Checagem do framing - CRC

No modo RTU, o cálculo de checksum adotado é o CRC, Cyclical Redundandcy Chceck, que calcula o conteúdo de **toda a mensagem**. É gerado um valor de 16 bits sendo que na composição final desde campo, os 8 bits **menos significativos** são enviados **primeiro**. Este é o último campo da mensagem sendo que os 8 bits **mais significativos** representam o **último** byte da mensagem.

O dispositivo transmissor calcula o valor do CRC e o integra à mensagem, transmitindo-a em seguida ao dispositivo receptor, que por sua vez, recalcula o CRC de toda a mensagem após a sua total recepção e o compara ao campo CRC da mensagem enviada, sinalizando erro caso não sejam iguais.

Este método, apesar de levar mais tempo para ser executado em relação ao método LRC, é muito mais **confiável** pois, como será visto a seguir, analisa o real conteúdo dos dados, **bit a bit**, que estão sendo transferidos na linha de comunicação, **fisicamente** falando.

O cálculo do CRC é iniciado primeiramente carregando-se um registrador / variável de memória (referenciado de agora em diante simplesmente como **registrador CRC**) de 16 bits com valor FFFFH. Apenas os **8 bits menos significativos** deste registrador CRC serão utilizados para o cálculo efetivo do CRC. Os bits de configuração: start, paridade e stop bits, não são utilizados no cálculo do CRC, apenas os bits do caracter propriamente dito.

Durante a geração do CRC, cada caracter é submetido a uma **lógica XOR** (OU exclusivo) com os 8 bits menos significativos do registrador CRC, cujo resultado é retornado a ele mesmo e **deslocado** (não é rotacionado) **uma posição** (1 bit) **à direita**, em direção ao bit menos significativo, sendo que a posição do bit mais significativo é preenchida com valor **0 (zero)**. Após esta operação, o **bit menos significativo** é examinado, ocorrendo o seguinte processamento:

1. se o valor deste bit for igual a 0, nada ocorre e a rotina de cálculo do CRC continua normalmente;

2. se o valor do bit for igual a 1, o conteúdo de todo o registrador CRC (16 bits) é submetido a uma **lógica XOR** com um **valor constante A001H** e o resultado é retornado ao registrador CRC.

Este processo se repete até que ocorram **8 deslocamentos** para **cada caracter da mensagem** que é submetido à lógica XOR com o registrador CRC portanto, o processo só terminará após **todos** os caracteres da mensagem terem sido submetidos à lógica XOR com o registrador CRC, gerando o valor do CRC que será colocado no **Campo Checksum** da mensagem.

Como regra geral, o procedimento para o cálculo do LRC é o seguinte:

1. carrega-se o registrador CRC com o valor FFFFH;
2. submete-se o caracter da mensagem a uma lógica XOR com os 8 bits menos significativos do registrador CRC, retornando o resultado no registrador CRC;
3. desloca-se o conteúdo do registrador CRC 1 bit par a direita programando seu bit mais significativo com 0 (zero);
4. examina-se o bit menos significativo do registrador CRC e:

   - se bit igual a 0, repete-se o processo a partir do item 3;

   - se bit igual a 1, submete-se o registrador CRC a uma lógica XOR com a constante A001H retornando o resultado no registrador CRC, em seguida, repete-se o processo a partir do item 3;
5. repetem-se os itens 3 e 4 até que tenham ocorrido 8 deslocamentos;
6. repetem-se os itens 2 até 5 para o próximo caracter da mensagem e assim sucessivamente até que todos os caracteres tenham sido analisados;
7. o valor final do registrador CRC é o valor do Campo Checksum;
8. primeiramente coloca-se o byte **menos** significativo do registrador CRC na mensagem e depois o **mais** significativo.

O processo descrito acima é o chamado cálculo **discreto** do CRC que, infelizmente, consome **muito tempo** para se realizar e começa a ficar crítico à medida que as mensagens passam a ter vários bytes a serem transmitidos. Para minimizar este problema, foram criadas **duas tabelas** de **256 bytes cada** uma, contendo **todas** as possíveis combinações tanto para o byte mais significativo como para o menos significativo do registrador CRC. O inconveniente deste recurso é que ele requer que o dispositivo possa dispor de pelo menos 512 bytes da memória de programa para armazenar as duas tabelas porém, o cálculo será realizado bem mais rápido pois é feito através de **indexação** dos seus valores. As tabelas e respectivos valores são mostradas ao final deste item.

Para esta solução o procedimento para o cálculo de CRC é o seguinte:

1. carrega-se um registrador CRC + com FFH e outro registrador CRC – com FFH;
2. as tabelas referenciada como tab CRC + e tab CRC – devem estar previamente programadas com os respectivos valores das combinações;
3. submete-se o byte da mensagem a uma lógica XOR com o conteúdo do registrador CRC +, retornando o resultado em um variável de 8 bits referenciada como **index**;
4. submete-se o valor da tab CRC +, indexada pela variável **index,** a uma lógica XOR com o registrador CRC – , retornando o resultado no registrador CRC +;
5. carrega-se o registrador CRC – com o valor da tab CRC – , indexada pela variável **index;**
6. repete-se os itens 3 à 5 até que todo o conteúdo da mensagem tenha sido analisado;

7. após este processo, os registradores CRC + e CRC – já possuem os respectivos valores a serem programados no Campo Checksum da mensagem.

### Tabela CRC +

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,

### Tabela CRC –

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,

## 4. Funções ModBus

Existem diversas funções ModBus disponíveis, de acordo com o modelo de CLP utilizado. Os indicadores da ALFA, entretanto, estão programados para reconhecer **apenas** as que realizam os comandos pertinentes à pesagem de maneira mais rápida e eficaz. Serão descritas em detalhes apenas estas funções, nos modos ASCII e RTU mas antes, é necessário saber como os dados são referenciados no protocolo ModBus.

### 4.1. Codificação do endereço

Todos os dados relativos a endereçamento têm o **zero** como referência, ou seja, se o registrador do dispositivo estiver no endereço 0001H, seu endereço na **mensagem** ModBus será o 0000H, e assim sucessivamente. Tomemos o exemplo do **item 3.7.2.2.**

Na realidade, o registrador que será programado no dispositivo é o de número **0059H** porém, como ele, o registrador, está sendo acessado através do protocolo ModBus, o mesmo deve ser referenciado como número 0058H na **mensagem**. O mesmo ocorre para os sensores e demais registradores de acesso do dispositivo.

Nos indicadores da ALFA, todos os registradores são do tipo **holding**, ou seja, podem tanto ser acessados para leitura como para escrita e, segundo os padrões da Modicon, estes registradores são numerados a partir do endereço **40001**, cujo endereço correspondente na mensagem é 0000H. Recorrendo uma vez mais ao exemplo do **item 3.7.2.2**, o registrador 0058H, na realidade, possui valor 40089, **em decimal**.

### 4.2. Conteúdo dos campos

A seguir é mostrado um exemplo de um comando e resposta, tanto em modo ASCII como em RTU. O mestre envia um comando de **leitura de registradores** tipo holding para o dispositivo escravo número 7BH, desejando ler o conteúdo dos registradores 40108 à 40110, portanto, um total de 3 registradores. Note que na mensagem o endereço do registrador inicial é 0107 decimal, equivalente a 006BH.

Na resposta, o escravo repete o código da função ModBus, indicando que o comando foi executado com sucesso. O campo **Byte Count** especifica quantos bytes estão sendo retornados ao mestre. No modo ASCII seu valor é igual à metade dos caracteres enviados no **Campo de Dados**, ou seja, para cada quatro bits é necessário o envio de um caracter ASCII portanto, para cada byte de dado são enviados dois caracteres ASCII.

No exemplo, um dos dados lidos possui valor 7BH. No modo RTU é enviado este mesmo valor: 01111011, pois é uma grandeza de 8 bits. Já no modo ASCII, são enviados dois caracteres para representar este valor: 37H - 00110111, que equivale ao número 7 em ASCII e 42H - 01000010, que equivale à **letra B** em ASCII. Neste caso, apesar de serem enviados dois caracteres para cada dado, totalizando portanto 12 bytes, o campo Byte Count **deve** conter a metade da quantidade real enviada.

**Mensagem query**:

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 7BH | 7 B | 0111 1011 |
| Função ModBus | 03H | 0 3 | 0000 0011 |
| Endereço inicial (+) | 00H | 0 0 | 0000 0000 |
| Endereço inicial (-) | 6BH | 6 B | 0110 1011 |
| No. de registradores (+) | 00H | 0 0 | 0000 0000 |
| No. de registradores (-) | 03H | 0 3 | 0000 0011 |
| Checksum caracter 1<br>Checksum caracter 2 | | 1<br>4 | 0111 1111<br>1000 1101 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 17 | 8 |

Conteúdo em **hexa** da mensagem query no **modo ASCII**:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 37H | 42H | **30H** | **33H** | 30H | 30H | 36H | 4BH | **30H** | **30H** | **30H** | **33H** | 31H | 34H | **0DH** | **0AH** |

Conteúdo em **hexa** da mensagem query no **modo RTU**:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **7BH** | 03H | **00H** | **6BH** | 00H | 03H | **7FH** | **8DH** |

**Mensagem response**:

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 7BH | 7 B | 0111 1011 |
| Função ModBus | 03H | 0 3 | 0000 0011 |
| Campo Byte Count | 06H | 0 6 | 0000 0110 |
| Dado do 1º registrador (+) | 00H | 0 0 | 0000 0000 |
| Dado do 1º registrador (-) | 5FH | 5 F | 0101 1111 |
| Dado do 2º registrador (+) | 01H | 0 1 | 0000 0001 |
| Dado do 2º registrador (-) | A8H | A 8 | 1010 1000 |
| Dado do 3º registrador (+) | 3CH | 3 C | 0011 1100 |
| Dado do 3º registrador (-) | 69H | 6 9 | 0110 1001 |
| Checksum caracter 1<br>Checksum caracter 2 | | C<br>F | 1111 1111<br>0010 1000 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 23 | 11 |

Conteúdo em **hexa** da mensagem response no **modo ASCII**:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 37H | 42H | **30H** | **33H** | 30H | 36H | **30H** | **30H** | **35H** | **46H** | 30H | 31H | 41H | 38H |
| **33H** | **43H** | **36H** | **39H** | 43H | 46H | **0DH** | **0AH** | | | | | | | |

Conteúdo em **hexa** da mensagem response no **modo RTU**:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **7BH** | 03H | **06H** | **00H** | **5FH** | 01H | A8H | **3CH** | **69H** | FFH | 28H |

### 4.3. Leitura de Bloco de Registradores – Função 03

#### 4.3.1. Descrição

Lê o conteúdo de um bloco de registradores tipo **holding** (referenciados como 4XXXX)**.** Para este comando não são válidos acessos tipo broadcast.

#### 4.3.2. Comando enviado

A mensagem query especifica o registrador inicial e a quantidade de registradores a serem lidos. Lembrar que os registradores são endereçados a partir do endereço 0, ou seja, os registradores 1 à 99, são endereçados como 0 à 98. No exemplo a seguir é solicitada uma leitura dos registradores 40108 à 40110 do dispositivo 17.

Observe que as informações relativas aos registradores são formatadas em **2 bytes** sendo que o primeiro byte contém a parte mais significativa da informação.

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 11H | 1 1 | 0001 0001 |
| Função ModBus | 03H | 0 3 | 0000 0011 |
| Endereço inicial (+) | 00H | 0 0 | 0000 0000 |
| Endereço inicial (-) | 6BH | 6 B | 0110 1011 |
| No. de registradores (+) | 00H | 0 0 | 0000 0000 |
| No. de registradores (-) | 03H | 0 3 | 0000 0011 |
| Checksum caracter 1<br>Checksum caracter 2 | | 7<br>E | 0111 0110<br>1000 0111 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 17 | 8 |

Conteúdo em **hexa** da mensagem query no **modo ASCII**:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 31H | 31H | **30H** | **33H** | 30H | 30H | 36H | 4BH | **30H** | **30H** | **30H** | **33H** | 37H | 45H | **0DH** | **0AH** |

Conteúdo em **hexa** da mensagem query no **modo RTU**:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **11H** | 03H | **00H** | **6BH** | 00H | 03H | **76H** | **87H** |

### 4.3.3. Resposta ao comando

Na resposta, os dados dos registradores também são formatados em **2 bytes** sendo que o primeiro byte contém a parte mais significativa do dado.

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 11H | 1 1 | 0001 0001 |
| Função ModBus | 03H | 0 3 | 0000 0011 |
| Campo Byte Count | 06H | 0 6 | 0000 0110 |
| Dado do 1º registrador (+) | 00H | 0 0 | 0000 0000 |
| Dado do 1º registrador (-) | 5FH | 5 F | 0101 1111 |
| Dado do 2º registrador (+) | 01H | 0 1 | 0000 0001 |
| Dado do 2º registrador (-) | A8H | A 8 | 1010 1000 |
| Dado do 3º registrador (+) | 3CH | 3 C | 0011 1100 |
| Dado do 3º registrador (-) | 69H | 6 9 | 0110 1001 |
| Checksum caracter 1<br>Checksum caracter 2 | | 3<br>9 | 0010 1001<br>1000 1010 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 23 | 11 |

Conteúdo em **hexa** da mensagem response no **modo ASCII**:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 31H | 31H | **30H** | **33H** | 30H | 36H | **30H** | **30H** | **35H** | **46H** | 30H | 31H | 41H | 38H |
| **33H** | **43H** | **36H** | **39H** | 33H | 39H | **0DH** | **0AH** | | | | | | | |

Conteúdo em **hexa** da mensagem response no **modo RTU**:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **11H** | 03H | **06H** | **00H** | **5FH** | 01H | A8H | **3CH** | **69H** | 29H | 8AH |

De acordo com o comando, o registrador 40108 contém 005FH (95 decimal), o registrador 40109 contém 01A8H (424 decimal) e o registrador 40110 contém 3C69H (15465 decimal).

### 4.4. Escrita em único Registrador – Função 06

#### 4.4.1. Descrição

Programa um único registradores tipo **holding** (referenciado como 4XXXX)**.** Para acessos tipo broadcast, este mesmo registrador de **todos** os escravos da rede serão programados com o mesmo valor.

#### 4.4.2. Comando enviado

A mensagem query especifica o registrador a ser programado. Lembrar que os registradores são endereçados a partir do endereço 0, ou seja, os registradores 1 à 99, são endereçados como 0 à 98. No exemplo é programado o valor 07D5H no registrador 40351 do dispositivo 17.

Observe que as informações relativas aos registradores são formatadas em **2 bytes** sendo que o primeiro byte contém a parte mais significativa da informação.

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 11H | 1 1 | 0001 0001 |
| Função ModBus | 06H | 0 6 | 0000 0110 |
| Endereço inicial (+) | 01H | 0 1 | 0000 0001 |
| Endereço inicial (-) | 5EH | 5 E | 0101 1110 |
| Valor programado (+) | 07H | 0 7 | 0000 0111 |
| Valor programado (-) | D5H | D 5 | 1101 0101 |
| Checksum caracter 1<br>Checksum caracter 2 | | A<br>E | 0010 1000<br>1101 1011 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 17 | 8 |

Conteúdo em **hexa** da mensagem query no **modo ASCII**:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 31H | 31H | **30H** | **36H** | 30H | 31H | 35H | 45H | **30H** | **37H** | **44H** | **35H** | 41H | 45H | **0DH** | **0AH** |

Conteúdo em **hexa** da mensagem query no **modo RTU**:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **11H** | 06H | **01H** | **5EH** | 07H | D5H | **28H** | **DBH** |

### 4.4.3. Resposta ao comando

Se o comando for realizado com sucesso, a mensagem de resposta é uma **cópia exata** da mensagem query.

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 11H | 1 1 | 0001 0001 |
| Função ModBus | 06H | 0 6 | 0000 0110 |
| Endereço inicial (+) | 01H | 0 1 | 0000 0001 |
| Endereço inicial (-) | 5EH | 5 E | 0101 1110 |
| Valor programado (+) | 07H | 0 7 | 0000 0111 |
| Valor programado (-) | D5H | D 5 | 1101 0101 |
| Checksum caracter 1<br>Checksum caracter 2 | | A<br>E | 0010 1000<br>1101 1011 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 17 | 8 |

Conteúdo em **hexa** da mensagem response no **modo ASCII**:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 31H | 31H | **30H** | **36H** | 30H | 31H | 35H | 45H | **30H** | **37H** | **44H** | **35H** | 41H | 45H | **0DH** | **0AH** |

Conteúdo em **hexa** da mensagem response no **modo RTU**:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **11H** | 06H | **01H** | **5EH** | 07H | D5H | **28H** | **DBH** |

### 4.5. Escrita em Bloco de Registradores – Função 16

#### 4.5.1. Descrição

Programa um bloco de registradores tipo **holding** (referenciado como 4XXXX) sequencialmente**.** Para acessos tipo broadcast, este mesmo bloco de registradores de **todos** os escravos da rede serão programados igualmente.

#### 4.5.2. Comando enviado

A mensagem query especifica o bloco de registradores a ser programado. Lembrar que os registradores são endereçados a partir do endereço 0, ou seja, registradores 1 à 99, são endereçados como 0 à 98. No exemplo, é programado no dispositivo 17 um bloco de registradores, a partir do registrador 40070 ao 40072, com os seguintes dados, respectivamente: 350BH (13579 decimal), 6068H (24680 decimal) e FF98H (65432 decimal).

O campo de dados deste comando define o endereço do registrador inicial do bloco, o número de registradores do bloco a serem programados, a quantidade de bytes que será programada e os bytes propriamente ditos. Observe que as informações relativas aos registradores são formatadas em **2 bytes** sendo que o primeiro byte contém a parte mais significativa da informação.

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 11H | 1 1 | 0001 0001 |
| Função ModBus | 10H | 1 0 | 0001 0000 |
| Endereço inicial (+) | 00H | 0 0 | 0000 0000 |
| Endereço inicial (-) | 45H | 4 5 | 0100 0101 |
| No. de registradores (+) | 00H | 0 0 | 0000 0000 |
| No. de registradores (-) | 03H | 0 3 | 0000 0011 |
| Campo Byte Count | 06H | 0 6 | 0000 0110 |
| Dado do 1º registrador (+) | 35H | 3 5 | 0011 0101 |
| Dado do 1º registrador (-) | 0BH | 0 B | 0000 1011 |
| Dado do 2º registrador (+) | 60H | 6 0 | 0110 0000 |
| Dado do 2º registrador (-) | 68H | 6 8 | 0110 1000 |
| Dado do 3º registrador (+) | FFH | F F | 1111 1111 |
| Dado do 3º registrador (-) | 98H | 9 8 | 1001 1000 |
| Checksum caracter 1<br>Checksum caracter 2 | | 0<br>3 | 1011 0101<br>0011 0110 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 31 | 15 |

Conteúdo em **hexa** da mensagem query no **modo ASCII**:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 31H | 31H | **31H** | **30H** | 30H | 30H | 34H | 35H | **30H** | **30H** | **30H** | **33H** | 30H | 36H | |
| **33H** | **35H** | **30H** | **42H** | 36H | 30H | 36H | 38H | **46H** | **46H** | **39H** | **38H** | 30H | 33H | **0DH** | **0AH** |

Conteúdo em **hexa** da mensagem query no **modo RTU**:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **11H** | 10H | **00H** | **45H** | 00H | 03H | **06H** | 35H | 0BH | **60H** | **68H** | FFH | 98H | **B5H** | **36H** |

### 4.5.3. Resposta ao comando

Se o comando for realizado com sucesso, a mensagem de resposta retorna apenas o endereço do escravo, o código da função ModBus, o endereço inicial do bloco de registradores e a quantidade de registradores programados.

| Nome do campo | Valores em HEXA | Caracteres ASCII | Bytes em RTU |
|---|---|---|---|
| Identificador de início | | **:** | NÃO HÁ |
| Endereço do escravo | 11H | 1 1 | 0001 0001 |
| Função ModBus | 10H | 1 0 | 0001 0000 |
| Endereço inicial (+) | 00H | 0 0 | 0000 0000 |
| Endereço inicial (-) | 45H | 4 5 | 0100 0101 |
| No. de registradores (+) | 00H | 0 0 | 0000 0000 |
| No. de registradores (-) | 03H | 0 3 | 0000 0011 |
| Checksum caracter 1<br>Checksum caracter 2 | | 9<br>7 | 1001 0011<br>0100 1101 |
| Indicador de fim | | CR LF | NÃO HÁ |
| Total de bytes enviados | | 17 | 8 |

Conteúdo em **hexa** da mensagem response no **modo ASCII**:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3AH** | 31H | 31H | **31H** | **30H** | 30H | 30H | 34H | 35H | **30H** | **30H** | **30H** | **33H** | 30H | 33H | **0DH** | **0AH** |

Conteúdo em **hexa** da mensagem response no **modo RTU**:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **11H** | 10H | **00H** | **45H** | 00H | 03H | **93H** | **4DH** |

## 5. Procedimentos de transmissão e recepção

Este item descreve os procedimentos a serem seguidos na transmissão e recepção das mensagens bem como o comportamento dos terminais mestre e escravo. Neste item, toda a referência feita a um terminal mestre identifica ou em CLP ou um sistema supervisório visto que os indicadores ALFA **sempre** são configurados como escravos em uma rede de comunicação.

### 5.1. Mestre

Em uma rede de comunicação o mestre pode ter dois comportamentos distintos:

- transmissor de mensagens para o escravo
- receptor de mensagens enviadas pelo escravo

#### 5.1.1. Transmissor

Como transmissor, o mestre envia uma mensagem ao(s) escravo(s) e dispara uma contagem máxima (time-out) de espera para confirmar se o escravo recebeu essa mensagem. Podem ocorrer as seguintes situações em relação à resposta esperada:

- execução com sucesso
- time-out
- função inválida
- registrador inválido
- valor de dado inválido
- estado de espera
- dispositivo ocupado

#### 5.1.2. Receptor

Como receptor, o mestre analisa a mensagem respondida pelo escravo e podem ocorrer as seguintes situações:

- execução com sucesso: o terminal escravo reconhece a mensagem enviada pelo mestre, executa o comando definido na mensagem e responde ao mestre os valores obtidos. O mestre finaliza o processo de recepção e um novo pedido de transmissão fica a cargo da aplicação que estiver sendo executada.
- time-out: o terminal escravo não responde absolutamente nada ao mestre após a ocorrência de time-out. O time-out ocorre ou quando não existe o escravo endereçado na mensagem transmitida pelo mestre, ou quando o checksum da mensagem não coincide com o valor do campo **Checksum CRC** (ModBus RTU) / **LRC** (ModBus ASCII) ou ainda quando há qualquer outro erro de comunicação. Nestes casos, geralmente o mestre executa uma sequência de retries (novas tentativas), retransmitindo a mensagem enviada anteriormente. Este processo de retry se repete até que o escravo responda uma mensagem coerente ou até que seja atingido o número máximo de retries programados pelo mestre. Neste último caso, o mestre assume a ocorrência de um erro de comunicação, finaliza o processo de recepção e a análise do time-out fica a cargo da aplicação que estiver sendo executada.
- função inválida: o escravo endereçado pelo mestre reconhece a mensagem mas identifica que a função ModBus solicitada pelo mestre não está implementada no terminal escravo. Neste caso, o mestre recebe

uma mensagem de Exception do escravo. O mestre finaliza o processo de recepção e um novo pedido de transmissão fica a cargo da aplicação que estiver sendo executada.

- registrador inválido: o escravo endereçado pelo mestre reconhece a mensagem mas identifica que o(s) registrador(es) especificado(s) na mensagem transmitida pelo mestre não existe(m). Neste caso, o mestre recebe uma mensagem de Exception do escravo. O mestre finaliza o processo de recepção e um novo pedido de transmissão fica a cargo da aplicação que estiver sendo executada.
- valor de dado inválido: o escravo endereçado pelo mestre reconhece a mensagem mas identifica que o valor de algum dado(s) contido(s) no **Campo de Dados** da mensagem enviada é inválido. Neste caso, o mestre recebe uma mensagem de Exception do escravo. O mestre finaliza o processo de recepção e um novo pedido de transmissão fica a cargo da aplicação que estiver sendo executada.
- estado de espera: o escravo endereçado pelo mestre reconhece a mensagem mas o notifica de que a mesma será processada num período de tempo maior que o normal. Neste caso, o mestre recebe uma mensagem de Exception do escravo. Este tipo de mensagem é enviada ao mestre para evitar a ocorrência de time-out. O mestre finaliza o processo de recepção e um novo pedido de transmissão fica a cargo da aplicação que estiver sendo executada, que pode definir ao mestre ficar monitorando as atividades do escravo até que este realize o comando.
- dispositivo ocupado: o escravo endereçado pelo mestre reconhece a mensagem e o notifica de que a mesma não pode ser processada pois o escravo está ocupado atendendo a outro comando. Neste caso, o mestre recebe uma mensagem de Exception do escravo. Este tipo de mensagem é enviada ao mestre para evitar a ocorrência de time-out. O mestre finaliza o processo de recepção e um novo pedido de transmissão fica a cargo da aplicação que estiver sendo executada, que pode definir ao mestre retransmitir a mensagem mais tarde, quando o escravo já tiver completado o comando em execução.

A formação de todas as mensagens de Excpetion está descrita no item **3.7. Campo Dados para o Escravo** deste documento.

### 5.2. Escravo

Em uma rede de comunicação o escravo pode ter dois comportamentos distintos:

- receptor das mensagens enviadas pelo mestre
- transmissor de mensagens para o mestre quando for solicitado

#### 5.2.1. Receptor

Como receptor, o escravo analisa a mensagem enviada pelo mestre e podem ocorrera as seguintes situações:

- recepção com sucesso: o escravo valida o endereço definido no campo **Endereço do Escravo** e as demais informações contidas nos outros campos da mensagem, executando o comando solicitado.
- dados inválidos: uma mensagem é considerada inválida quando ou o endereço definido no seu campo **Endereço do Escravo** não corresponder ao endereço do terminal escravo que estiver recebendo a mensagem, ou quando o checksum da mensagem não coincidir com o valor do campo **Checksum CRC** (ModBus RTU) / **LRC** (ModBus ASCII) ou ainda quando houver qualquer outro erro de comunicação.
- função inválida: o escravo endereçado pelo mestre reconhece a mensagem mas identifica que a função ModBus solicitada pelo mestre não está implementada.

- registrador inválido: o escravo endereçado pelo mestre reconhece a mensagem mas identifica que o(s) registrador(es) especificado(s) na mensagem transmitida não existe(m).
- valor de dado inválido: o escravo endereçado pelo mestre reconhece a mensagem mas identifica que o valor de algum dado(s) contido(s) no **Campo de Dados** da mensagem enviada é inválido.
- estado de espera: o escravo endereçado pelo mestre reconhece a mensagem mas o notifica de que a mesma será processada num período de tempo maior que o normal.
- dispositivo ocupado: o escravo endereçado pelo mestre reconhece a mensagem e o notifica de que a mesma não pode ser processada pois o escravo está ocupado atendendo a outro comando.

### 5.2.2. Transmissor

Como transmissor, o escravo envia uma mensagem ao mestre contendo informações relativas à mensagem recebida e podem ocorrer as seguintes situações:

- recepção com sucesso: após validar e executar o comando definido na mensagem, responde ao mestre com as informações relativas ao comando executado. A formação deste tipo de mensagem está descrita no item **6.3. Comandos disponíveis nos indicadores ALFA** deste documento.
- dados inválidos: neste caso o escravo não responde nada ao mestre.
- função inválida: o escravo responde com uma mensagem de Exception.
- registrador inválido: o escravo responde com uma mensagem de Exception.
- valor de dado inválido: o escravo responde com uma mensagem de Exception.
- estado de espera: o escravo responde com uma mensagem de Exception. Este tipo de mensagem é enviada ao mestre para evitar a ocorrência de time-out.
- dispositivo ocupado: o escravo responde com uma mensagem de Exception. Este tipo de mensagem é enviada ao mestre para evitar a ocorrência de time-out.

A formação de todas as mensagens de Excpetion está descrita no item **3.7. Campo Dados para o Escravo** deste documento.

## 5.3. Retry

Os indicadores ALFA estão preparados para atender a qualquer que seja o número de retries programado pelo mestre, que devem ocorrer num intervalo **mínimo** de 100 ms.

## 6. Protocolo na aplicação

No estágio da aplicação, o protocolo se preocupa com o tratamento das informações embutidas em uma mensagem enviada ou recebida pela rede de comunicação. Estas mensagens compõem o **Quadro de Comandos** e **Quadro de Respostas** que são compostos pelos **Comandos disponíveis nos indicadores ALFA** .

### 6.1. Quadro de Comandos

Toda mensagem enviada por um equipamento conectado a uma rede cujo conteúdo especifique a execução de uma tarefa define um Quadro de Comandos. A tarefa requisitada pode ser o simples envio de dados do sistema, dados relativos a alarmes, alteração de parâmetros internos, etc..

Todo Quadro de Comandos é composto pelo identificador de início de mensagem, quando o protocolo assim o requerer, campo de endereço do escravo, função ModBus a ser executada, dados relativos à função ModBus, quando necessário, campo de checksum da mensagem e identificador de fim de mensagem, quando o protocolo assim o requerer.

Todos os campos acima estão abordados detalhadamente nos itens **3. Protocolo ModBus** e **4. Funções ModBus**. O campo de dados relativo à função ModBus está descrito em detalhes no item **6.3. Comandos disponíveis nos indicadores ALFA** .

### 6.2. Quadro de Respostas

Para todo Quadro de Comandos recebido e validado pelo escravo selecionado, **obrigatoriamente** deve existir um Quadro de Respostas, que é a mensagem retornada por ele ao dispositivo que solicitou a tarefa. Esta mensagem ou contém os dados relativos à tarefa solicitada pelo mestre ou o status atual do escravo indicando ao mestre o porquê de ainda não estar apto a retornar os dados relativos à tarefa solicitada, etc..

Todo Quadro de Respostas é composto pelo identificador de início de mensagem, quando o protocolo assim o requerer, campo de endereço do escravo, função ModBus que foi executada, dados relativos à função ModBus executada, quando necessários, campo de checksum da mensagem e identificador de fim de mensagem, quando o protocolo assim o requerer.

Todos os campos acima estão abordados detalhadamente nos itens **3. Protocolo ModBus** e **4. Funções ModBus**. O campo de dados relativos à função ModBus está descrito em detalhes no item **6.3. Comandos disponíveis nos indicadores ALFA** .

### 6.3. Comandos disponíveis nos indicadores ALFA

Na descrição destes comandos serão analisados apenas os dados do campo **Dados para o Indicador** pois os valores dos campos **Início** e **Fim de framing** são definidos de acordo com o modo de transmissão: ASCII ou RTU (**item 3**). O campo **Endereço do Indicador** pode assumir qualquer valor de 0 à 255 e também será definido de acordo com o modo de transmissão (**item 3**), o mesmo acontecendo com o campo **Checksum**, que pode ser no padrão LRC ou CRC (**item 3**). O campo **Função ModBus** (**item 4**) conterá o comando que identifica o tipo de acesso que estará sendo feito ao indicador.

| Início de framing | Endereço do Indicador | Função ModBus | **Dados para o Indicador** | Checksum | Fim de framing |
|---|---|---|---|---|---|

O campo **Dados para o Indicador** deverá ser gerado de acordo com o modo de transmissão, ASCII ou RTU (**item 4**) além de obedecer aos padrões do ModBus, ou seja, 4 bytes ou 2 bytes (modo ASCII ou RTU respectivamente) para valores que identifiquem o número(s) de registrador(es) e valores lidos ou programados nos registradores.

Na descrição dos comandos do protocolo ModBus a seguir, o conteúdo do campo **Dados para o Indicador** é analisado do ponto do vista **do indicador**.

### 6.3.1. Verificação do status do indicador

Função ModBus: 3 – Leitura de bloco de registradores

Retorna ao mestre o estado (status) de operação do indicador, de acordo com as seguintes faixas de valores:

| **Código** | **Estado do indicador** |
|---|---|
| 00H | = indicação de pesos |
| 01H à 0DH | = ajuste de parâmetros |
| 0EH à 11H | = ajuste de valores de setpoints |
| 13H à 18H | = autocalibração |

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 11

**No. total de registradores:** 1

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. de bytes lidos:** 2

**Valor lido do registrador 11:** Código da tabela acima

### 6.3.2. Inicialização do indicador

Função ModBus: 06 – Programação de 1 único registrador

Programa o indicador para o estado de indicação de pesos.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador:** 21

**Valor programado:** 0

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. do registrador:** 21

**Valor programado:** 0

### 6.3.3. Programação do nível dos SetPoints

Função ModBus: 16 – Programação de bloco de registradores

Programa os níveis de corte (SetPoints) do indicador juntamente com o valor da configuração **VAZIA**. Todos os valores devem ser passados, independentemente de quais níveis sejam alterados.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

| | |
|---|---|
| **No. do registrador inicial:** | 31 |
| **No. total de registradores:** | 9 |
| **Total de bytes programados:** | 18 |
| **Valor programado no registrador 31:** | bits 15 à 8 = 0<br>bit 7 = 1<br>bit 6 = 0 – não gravar dados em memória não volátil<br>= 1 – gravar dados em memória não volátil<br>bits 5 à 0 = 4 em binário (número de SetPoints) |
| **Valor programado no registrador 32:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint1. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg32*65536) + conteúdo Reg33 |
| **Valor programado no registrador 33:** | 16 bits menos significativos do valor do SetPoint1, limitado a 65535 unidades. |
| **Valor programado no registrador 34:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint2. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg34*65536) + conteúdo Reg35 |
| **Valor programado no registrador 35:** | 16 bits menos significativos do valor do SetPoint2, limitado a 65535 unidades. |
| **Valor programado no registrador 36:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint3. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg36*65536) + conteúdo Reg37 |
| **Valor programado no registrador 37:** | 16 bits menos significativos do valor do SetPoint3, limitado a 65535 unidades. |
| **Valor programado no registrador 38:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint VAZIA. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg38*65536) + conteúdo Reg39 |
| **Valor programado no registrador 39:** | 16 bits menos significativos do valor do SetPoint VAZIA, limitado a 65535 unidades. |

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

| | |
|---|---|
| **No. do registrador inicial:** | 21 |
| **No. total de registradores:** | 0 |

### 6.3.4. Verificação da programação do nível dos SetPoints

Função ModBus: 3 – Leitura de bloco de registradores

Lê os níveis de corte (SetPoints) do indicador juntamente com o valor da configuração **VAZIA**.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 41

**No. total de registradores:** 9

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

| | |
|---|---|
| **No. de bytes lidos:** | 18 |
| **Valor lido do registrador 41:** | bits 15 à 8 = 0<br>bit 7 = 1<br>bit 6 = 0<br>bits 5 à 0 = 4 em binário (número de SetPoints) |
| **Valor lido do registrador 42:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint1. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg42*65536) + conteúdo Reg43 |
| **Valor lido do registrador 43:** | 16 bits menos significativos do valor do SetPoint1, limitado a 65535 unidades. |
| **Valor lido do registrador 44:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint2. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg44*65536) + conteúdo Reg45 |
| **Valor lido do registrador 45:** | 16 bits menos significativos do valor do SetPoint2, limitado a 65535 unidades. |
| **Valor lido do registrador 46:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint3. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg46*65536) + conteúdo Reg47 |
| **Valor lido do registrador 47:** | 16 bits menos significativos do valor do SetPoint3, limitado a 65535 unidades. |
| **Valor lido do registrador 48:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do SetPoint VAZIA. O indicador trata o valor total deste SetPoint da seguinte forma: (conteúdo Reg48*65536) + conteúdo Reg49 |
| **Valor lido do registrador 49:** | 16 bits menos significativos do valor do SetPoint VAZIA, limitado a 65535 unidades. |

### 6.3.5. Programação da configuração dos SetPoints

Função ModBus: 16 – Programação de bloco de registradores

Programa todos os parâmetros de configuração dos SetPoints.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 51

**No. total de registradores:** 6

**Total de bytes programados:** 12

**Valor programado no registrador 51:**
bits 15 à 7 = 0
bits 6 à 0 = valor da histerese em porcentagem, variando de 0 e 99

**Valor programado no registrador 52:**
bits 15 à 1 = 0
bit 0 = 0 – lógica do relê: normalmente aberto
= 1 – lógica do relê: normalmente fechado

**Valor programado no registrador 53:**
bits 15 à 8 = 0
bit 7 = 1
bit 6 = 0 – não gravar dados em memória não volátil
= 1 – gravar dados em memória não volátil
bits 5 à 0 = 3 em binário (número de SetPoints exceto o VAZIA)

**Valor programado no registrador 54:**
bits 15 à 1 = 0
bit 0 = 0 – SetPoint1 sem trava
= 1 – SetPoint1 com trava

**Valor programado no registrador 55:**
bits 15 à 1 = 0
bit 0 = 0 – SetPoint2 sem trava
= 1 – SetPoint2 com trava

**Valor programado no registrador 56:**
bits 15 à 1 = 0
bit 0 = 0 – SetPoint3 sem trava
= 1 – SetPoint3 com trava

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. do registrador inicial:** 51

**No. total de registradores:** 6

### 6.3.6. Verificação da configuração dos SetPoints

Função ModBus: 3 – Leitura de bloco de registradores

Verifica a configuração de todos os SetPoints.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 61

**No. total de registradores:** 6

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. de bytes lidos:** 12

**Valor lido do registrador 61:** bits 15 à 7 = 0
bits 6 à 0 = valor da histerese em porcentagem, variando de 0 e 99

**Valor lido do registrador 62:** bits 15 à 1 = 0
bit 0 = 0 – lógica do relê: normalmente aberto
= 1 – lógica do relê: normalmente fechado

**Valor lido do registrador 63:** bits 15 à 8 = 0
bit 7 = 1
bit 6 = 0
bits 5 à 0 = 3 em binário (número de SetPoints exceto o VAZIA)

**Valor lido do registrador 64:** bits 15 à 1 = 0
bit 0 = 0 – SetPoint1 sem trava
= 1 – SetPoint1 com trava

**Valor lido do registrador 65:** bits 15 à 1 = 0
bit 0 = 0 – SetPoint2 sem trava
= 1 – SetPoint2 com trava

**Valor lido do registrador 66:** bits 15 à 1 = 0
bit 0 = 0 – SetPoint3 sem trava
= 1 – SetPoint3 com trava

### 6.3.7. Programação do endereço do indicador

Função ModBus: 06 – Programação de 1 único registrador

Define novo endereço do indicador na rede.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador:** 71

**Valor programado:** novo endereço do indicador, entre 0 e 99

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. do registrador:** 71

**Valor programado:** novo endereço do indicador, entre 0 e 99

### 6.3.8. Leitura do Peso e status do indicador

Função ModBus: 3 – Leitura de bloco de registradores

Solicita o envio do peso, tara e status atuais do indicador.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 81

**No. total de registradores:** 6

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

| | |
|---|---|
| **No. de bytes lidos:** | 18 |
| **Valor lido do registrador 81:** | bits 15 à 8 = 0<br>bit 7 = 1<br>bit 6 = 1 – ocorreu sobrecarga<br>bit 5 = 1 – ocorreu saturação<br>bit 4 = 1 – balança está em movimento<br>bit 3 = 1 – o peso aplicado à balança é negativo<br>bits 2 à 0 = posição do ponto decimal, em binário |
| **Valor lido do registrador 82:** | bits 15 à 8 = 0<br>bit 7 = 1<br>bit 6 = 1 – ocorreu alteração local de parâmetros pelo display ou teclado<br>bits 5 e 4 = 0<br>bit 3 = 1 – ocorreu passagem pelo SetPoint VAZIA<br>bit 2 = 1 – ocorreu passagem pelo SetPoint3<br>bit 1 = 1 – ocorreu passagem pelo SetPoint2<br>bit 0 = 1 – ocorreu passagem pelo SetPoint1 |
| **Valor lido do registrador 83:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do PESO. O indicador trata o valor total do PESO da seguinte forma: (conteúdo Reg83*65536) + conteúdo Reg84 |
| **Valor lido do registrador 84:** | 16 bits menos significativos do valor do PESO, limitado a 65535 unidades. |
| **Valor lido do registrador 85:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos da TARA. O indicador trata o valor total da TARA da seguinte forma: (conteúdo Reg85*65536) + conteúdo Reg86 |
| **Valor lido do registrador 86:** | 16 bits menos significativos do valor da TARA, limitado a 65535 unidades. |

### 6.3.9. Acionamento remoto das teclas de função do indicador

Função ModBus: 06 – Programação de 1 único registrador

Acionamento remoto das funções do indicador.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador:** 91

**Valor programado:** bits 15 à 5 = 0
bit 4 = 1 – aciona DESTRAVA
bit 3 = 1 – aciona DESTARA
bit 2 = 0
bit 1 = 1 – aciona TARA
bit 0 = 0 – aciona ZERO

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. do registrador:** 91

**Valor programado:** idêntico ao valor recebido

### 6.3.10. Programação da faixa de pesos da saída analógica

Função ModBus: 16 – Programação de bloco de registradores

Programa a faixa de atuação da saída 0/20mA ou 4/20mA e referência de peso: bruto ou líquido.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

| | |
|---|---|
| **No. do registrador inicial:** | 101 |
| **No. total de registradores:** | 5 |
| **Total de bytes programados:** | 10 |
| **Valor programado no registrador 101:** | bits 15 à 8 = 0<br>bits 7 à 0 = 43H – referência: peso bruto<br>= 4CH – referência: peso líquido<br>bits 5 à 0 = 4 em binário (número de SetPoints) |
| **Valor programado no registrador 102:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do peso para 0mA ou 4mA. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg102*65536) + conteúdo Reg103 |
| **Valor programado no registrador 103:** | 16 bits menos significativos do valor do peso para 0mA ou 4mA, limitado a 65535 unidades. |
| **Valor programado no registrador 104:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do peso para 20mA. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg104*65536) + conteúdo Reg105 |
| **Valor programado no registrador 105:** | 16 bits menos significativos do valor do peso para 20mA, limitado a 65535 unidades. |

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. do registrador inicial:** 101

**No. total de registradores:** 5

### 6.3.11. Verificação da faixa de pesos da saída analógica

Função ModBus: 3 – Leitura de bloco de registradores

Leitura da faixa de atuação da saída 0/20mA ou 4/20mA e respectiva referência de peso.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 111

**No. total de registradores:** 5

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

| | |
|---|---|
| **No. de bytes lidos:** | 10 |
| **Valor lido do registrador 111:** | bits 15 à 8 = 0<br>bits 7 à 0 = 43H – referência: peso bruto<br>= 4CH – referência: peso líquido<br>bits 5 à 0 = 4 em binário (número de SetPoints) |
| **Valor lido do registrador 112:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do peso para 0mA ou 4mA. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg112*65536) + conteúdo Reg113 |
| **Valor lido do registrador 113:** | 16 bits menos significativos do valor do peso para 0mA ou 4mA, limitado a 65535 unidades. |
| **Valor lido do registrador 114:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do peso para 20mA. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg114*65536) + conteúdo Reg115 |
| **Valor lido do registrador 115:** | 16 bits menos significativos do valor do peso para 20mA, limitado a 65535 unidades. |

### 6.3.12. Leitura do peso acumulado

Função ModBus: 3 – Leitura de bloco de registradores

Disponível apenas nos indicadores que possuem função de acumulação de peso.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 121

**No. total de registradores:** 3

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

| | |
|---|---|
| **No. de bytes lidos:** | 6 |
| **Valor lido do registrador 121:** | bits 15 à 7 = 0<br>bit 6 = 1 – não efetuou acumulação, estouro da capacidade do buffer<br>bit 5 = 1 – não efetuou acumulação, peso na balança em movimento<br>bit 4 = 1 – não efetuou acumulação, operação já foi realizada através de comando manual ou remoto, sem ter sido liberada<br>bit 3 = 0<br>bits 2 à 0 = posição da casa decimal no valor acumulado |
| **Valor lido do registrador 122:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do peso acumulado. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg122*65536) + conteúdo Reg123 |
| **Valor lido do registrador 123:** | 16 bits menos significativos do valor do peso acumulado, limitado a 65535 unidades. |

### 6.3.13. Leitura do peso acumulado com posterior reset

Função ModBus: 3 – Leitura de bloco de registradores

Disponível apenas nos indicadores que possuem função de acumulação de peso.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 131

**No. total de registradores:** 3

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

| | |
|---|---|
| **No. de bytes lidos:** | 6 |
| **Valor lido do registrador 131:** | bits 15 à 7 = 0<br>bit 6 = 1 – não efetuou acumulação, estouro da capacidade do buffer<br>bit 5 = 1 – não efetuou acumulação, peso na balança em movimento<br>bit 4 = 1 – não efetuou acumulação, operação já foi realizada através de comando manual ou remoto, sem ter sido liberada<br>bit 3 = 0<br>bits 2 à 0 = posição da casa decimal no valor acumulado |
| **Valor lido do registrador 132:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do peso acumulado. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg132*65536) + conteúdo Reg133 |
| **Valor lido do registrador 133:** | 16 bits menos significativos do valor do peso acumulado, limitado a 65535 unidades. |

### 6.3.14. Programação dos parâmetros de calibração

Função ModBus: 16 – Programação de bloco de registradores

Programa o indicador para realizar todos os comandos do menu interno de calibração. Estes parâmetros podem ser alterados a qualquer instante, mesmo que já tenho sido feito um processo de calibração, desde que seja executado o comando **Geração da constante de calibração** no final da operação.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

| | |
|---|---|
| **No. do registrador inicial:** | 801 |
| **No. total de registradores:** | 9 |
| **Total de bytes programados:** | 18 |
| **Valor programado no registrador 801:** | bits 15 à 3 = 0<br>bits 2 à 0 = número de CASAS DECIMAIS, em binário |
| **Valor programado no registrador 802:** | bits 15 à 3 = 0<br>bits 2 à 0 = valor do DEGRAU, em binário |
| **Valor programado no registrador 803:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do PESO DE CALIBRAÇÃO. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg803*65536) + conteúdo Reg804 |
| **Valor programado no registrador 804:** | 16 bits menos significativos do valor do PESO DE CALIBRAÇÃO, limitado a 65535 unidades. |
| **Valor programado no registrador 805:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor da CAPACIDADE MÁXIMA. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg805*65536) + conteúdo Reg806 |
| **Valor programado no registrador 806:** | 16 bits menos significativos do valor da CAPACIDADE MÁXIMA, limitado a 65535 unidades. |
| **Valor programado no registrador 807:** | 0 = ZERO manual e automático desabilitados<br>1 = ZERO automático habilitado<br>2 = ZERO manual habilitado<br>3 = ZERO manual e automático habilitados |
| **Valor programado no registrador 808:** | fator de programação do FILTRO DIGITAL, variando de 0 (mais rápido) a 8 (mais lento) |
| **Valor programado no registrador 809:** | 0 = TARA não sucessiva<br>1 = TARA não sucessiva memorizada<br>2 = TARA sucessiva<br>3 = TARA sucessiva e memorizada |

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. do registrador inicial:** 801

**No. total de registradores:** 9

### 6.3.15. Verificação dos parâmetros de calibração

Função ModBus: 3 – Leitura de bloco de registradores

Lê todos os parâmetros de calibração do indicador.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 811

**No. total de registradores:** 9

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

| | |
|---|---|
| **No. de bytes lidos:** | 18 |
| **Valor lido do registrador 811:** | bits 15 à 3 = 0<br>bits 2 à 0 = número de CASAS DECIMAIS, em binário |
| **Valor lido do registrador 812:** | bits 15 à 3 = 0<br>bits 2 à 0 = valor do DEGRAU, em binário |
| **Valor lido do registrador 813:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor do PESO DE CALIBRAÇÃO. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg803*65536) + conteúdo Reg804 |
| **Valor lido do registrador 814:** | 16 bits menos significativos do valor do PESO DE CALIBRAÇÃO, limitado a 65535 unidades. |
| **Valor lido do registrador 815:** | valor múltiplo de 65536, correspondente aos 16 bits mais significativos do valor da CAPACIDADE MÁXIMA. O indicador trata o valor total deste PESO da seguinte forma: (conteúdo Reg805*65536) + conteúdo Reg806 |
| **Valor lido do registrador 816:** | 16 bits menos significativos do valor da CAPACIDADE MÁXIMA, limitado a 65535 unidades. |
| **Valor lido do registrador 817:** | 0 = ZERO manual e automático desabilitados<br>1 = ZERO automático habilitado<br>2 = ZERO manual habilitado<br>3 = ZERO manual e automático habilitados |
| **Valor lido do registrador 818:** | fator de programação do FILTRO DIGITAL, variando de 0 (mais rápido) a 8 (mais lento) |
| **Valor lido do registrador 819:** | 0 = TARA não sucessiva<br>1 = TARA não sucessiva memorizada<br>2 = TARA sucessiva |

3 = TARA sucessiva e memorizada

### 6.3.16. Calibração do indicador SEM PESO

Função ModBus: 3 – Leitura de bloco de registradores

Inicia o processo de calibração com o sistema **sem peso**. Nesta etapa **não deve existir nenhum peso** no sistema de pesagem e os acessórios que fazem parte do **peso morto** devem estar em seus lugares de trabalho.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 821

**No. total de registradores:** 1

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. de bytes lidos:** 2

**Valor lido do registrador 821:** 0 = sem erro

3 = peso instável

6 = conversão fora dos limites

### 6.3.17. Calibração do indicador COM PESO

Função ModBus: 3 – Leitura de bloco de registradores

Executa o processo de calibração com o sistema **com peso**. Nesta etapa **o peso** deve ser colocado no sistema de pesagem.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 831

**No. total de registradores:** 1

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. de bytes lidos:** 2

**Valor lido do registrador 831:** 0 = sem erro

3 = peso instável

conversão fora dos limites

### 6.3.18. Geração da constante de calibração

Função ModBus: 3 – Leitura de bloco de registradores

Calcula a constante e armazena os parâmetros de calibração. Este comando **sempre** deve ser executado ao final do processo de calibração.

Conteúdo do campo **Dados para o Indicador** no quadro de comandos:

**No. do registrador inicial:** 841

**No. total de registradores:** 1

Conteúdo do campo **Dados para o Indicador** no quadro de respostas:

**No. de bytes lidos:** 2

**Valor lido do registrador 841:** 0 = sem erro

1 = peso da balança vazia >= peso de calibração

2 = span insuficiente

9 = peso de calibração > capacidade máxima