

Balanças de Pesagem

Abordagem prática para balanças TOLEDO

por *Victory Fernandes e Murilo Plínio*

A automação atinge hoje indústrias, laboratórios, restaurantes e frigoríficos, que utilizam balanças de pesagem destinadas às mais diversas aplicações, desde balanças de alta precisão até balanças de pesagem de carga que suportam dezenas de toneladas.

Alguns dos principais problemas dos desenvolvedores são:

- ? Como integrar as balanças de pesagem ao sistema gerencial automatizando o processo produtivo?
- ? Como fazer isso de forma transparente sem precisar conhecer a fundo a comunicação de baixo nível com o *hardware*?
- ? Onde obter as informações para comunicação, suporte e meios para efetuar testes de comunicação sem possuir o *hardware* necessário no laboratório de desenvolvimento?

Tendo em vista estas e outras dificuldades, apresentamos este artigo descrevendo todo o processo de configuração, implementação e teste da comunicação com balanças de pesagem TOLEDO, senão o maior, um dos maiores fabricantes do setor no Brasil.

Abordaremos desde a configuração dos registradores internos da balança, passando pela comunicação inicial e testes com o *HyperTerminal*, aplicativo demo de comunicação em *Delphi*, simulador da balança para testes em laboratório sem a necessidade do equipamento, até a análise de baixo nível do protocolo de comunicação utilizado pelo fabricante.

Configuração dos Registradores da Balança

O primeiro passo para leitura das informações da balança a partir do seu sistema é fazer a correta configuração das opções dos registradores. Para tanto, coloque a balança no modo de programação digitando a seguinte sequência de caracteres no indicador digital (A senha padrão de fábrica é 1234):

```
[F1] + 1 + SENHA + ENTER
```

Uma vez no modo de programação, utilize as teclas [ENTER] para avançar os registradores e [>T] para voltar. Localize então os registradores de configuração da comunicação descritos a seguir e altere seus valores.

Registrador C00 – Modo de Operação

As balanças podem operar em modo pesador “d” ou modo contador de unidades “L”. Configure inicialmente este registrador para o valor “d”, para efetuar os testes de pesagem.

Registrador C03 – Sensor de Movimento

Permite que a indicação de peso só seja atualizada no *display* quando não houver movimento na plataforma de pesagem. Enquanto existir movimento o *display* ficará retido na ultima indicação de peso. Configure inicialmente este registrador para o valor “d”, inibindo o sensor de movimento, para efetuar os testes de pesagem.

Registrador C12 – *Checksum*

Este registrador permite enviar o *byte* de *checksum* no pacote de dados pela saída serial. O valor “d” inibe o envio e o valor “L” ativa o envio do *checksum*.

O cálculo do *checksum* é obtido através do complemento de 2(dois) da soma de todos os *bytes* recebidos do STX, inclusive a CR, como veremos no tópico *Protocolo de Comunicação – P03*.

Registrador C13 – *BaudRate*

Determina a velocidade de transmissão dos dados, e deve estar configurado para comunicação em 4800 *Bauds*.

Registrador C14 – Formato dos Dados

Determina o tipo de pacote de dados enviados, e deve ser configurado para o valor P03, saída contínua de dados.

Registrador C15 – Transmissão Contínua

Permite que os dados sejam transmitidos continuamente, modo contínuo “L”, ou que a transmissão só ocorra quando a tecla de transmissão for pressionada no indicador digital. Configure inicialmente este registrador para o valor “d”, modo demanda, para efetuar os testes de pesagem através do pressionamento do botão de transmissão.

Comunicação via HyperTerminal

Uma vez configurados os parâmetros da balança, é preciso verificar se a mesma está comunicando corretamente com o software padrão de comunicação do *Windows*, o *HyperTerminal*. Para tanto, basta criar uma nova conexão do *HyperTerminal* e configurá-la como segue:

Bits por segundo: 4800
Bits de Dados: 8
Paridade: Nenhum
Bits de Parada: 2
Controle de Fluxo: Nenhum

Uma vez concluída a conexão, após o pressionamento do botão de transmissão das informações no indicador digital da balança, o *HyperTerminal* efetua a leitura da *string* na porta serial e o resultado final será semelhante ao mostrado na *Figura 01*:

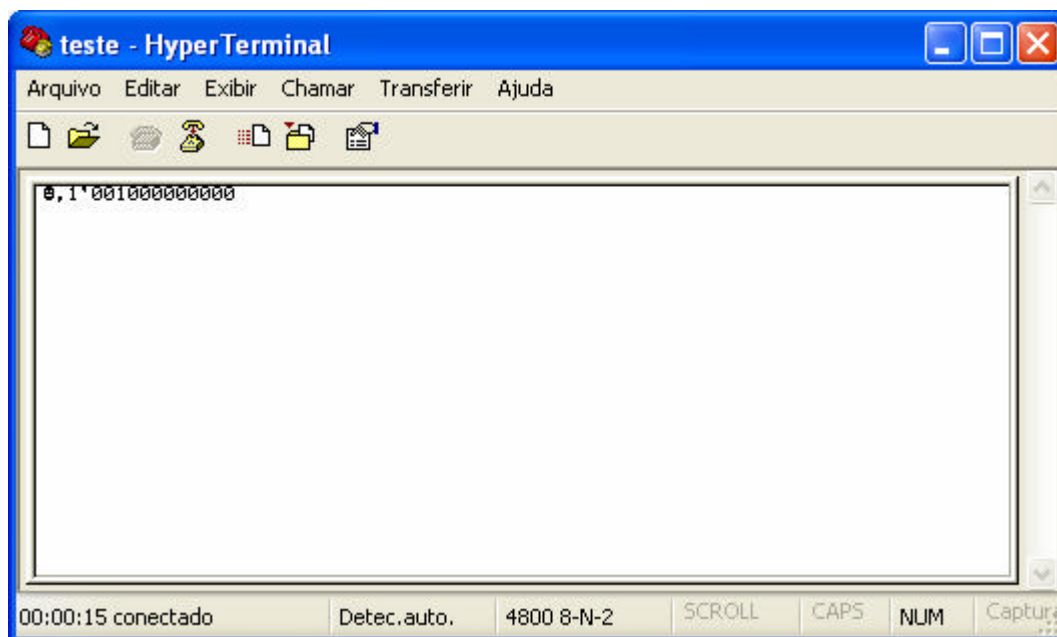


Figura 01: Comunicação com a balança via *HyperTerminal* e *string* lida da porta serial.

A *string* lida no *HyperTerminal* aparece segundo a interpretação em ASCII dos bytes enviados pela balança de acordo com o protocolo de comunicação. Portanto não se espante se os dados aparecerem truncados ou cheios de símbolos ilegíveis; esse é realmente o resultado esperado, e a tradução e interpretação desta *string* ilegível para um formato amigável é justamente o trabalho da *P03_Unit*, que será apresentada nos tópicos seguintes.

A leitura das informações da balança por parte do *HyperTerminal* garante que nosso *hardware*, configuração de *drivers* e *software* estão corretos, e que podemos prosseguir com os testes do aplicativo demo. Caso contrário, se faz necessário a verificação dos itens que estejam apresentando problemas antes de prosseguir. Alguns dos problemas mais comuns encontrados até essa etapa são:

- ? A configuração dos parâmetros na balança está incorreta.
- ? A configuração dos parâmetros no *HyperTerminal* está incorreta.
- ? A placa de comunicação serial da balança está com problemas.
- ? A porta serial do computador está com problemas físicos ou de configuração de *driver* no sistema operacional.
- ? A configuração sugerida dos parâmetros do *HyperTerminal* é a *default* e não apresentou problemas nos modelos testados. Entre em contato com o fabricante para maiores informações caso todos os itens anteriores aparentem estar operando corretamente e mesmo assim a comunicação não funcione corretamente.

Aplicativo Demo

O aplicativo demo é um programa que faz a leitura da *string* enviada pela balança e os entrega sob a forma de variáveis do *Delphi* ao programador. Esta leitura e tradução são implementadas utilizando as funções da *P03_Unit* que, implementa internamente o tratamento completo das informações do protocolo P03, protocolo padrão de comunicação das balanças TOLEDO, tornando o processo como um todo muito mais simples.

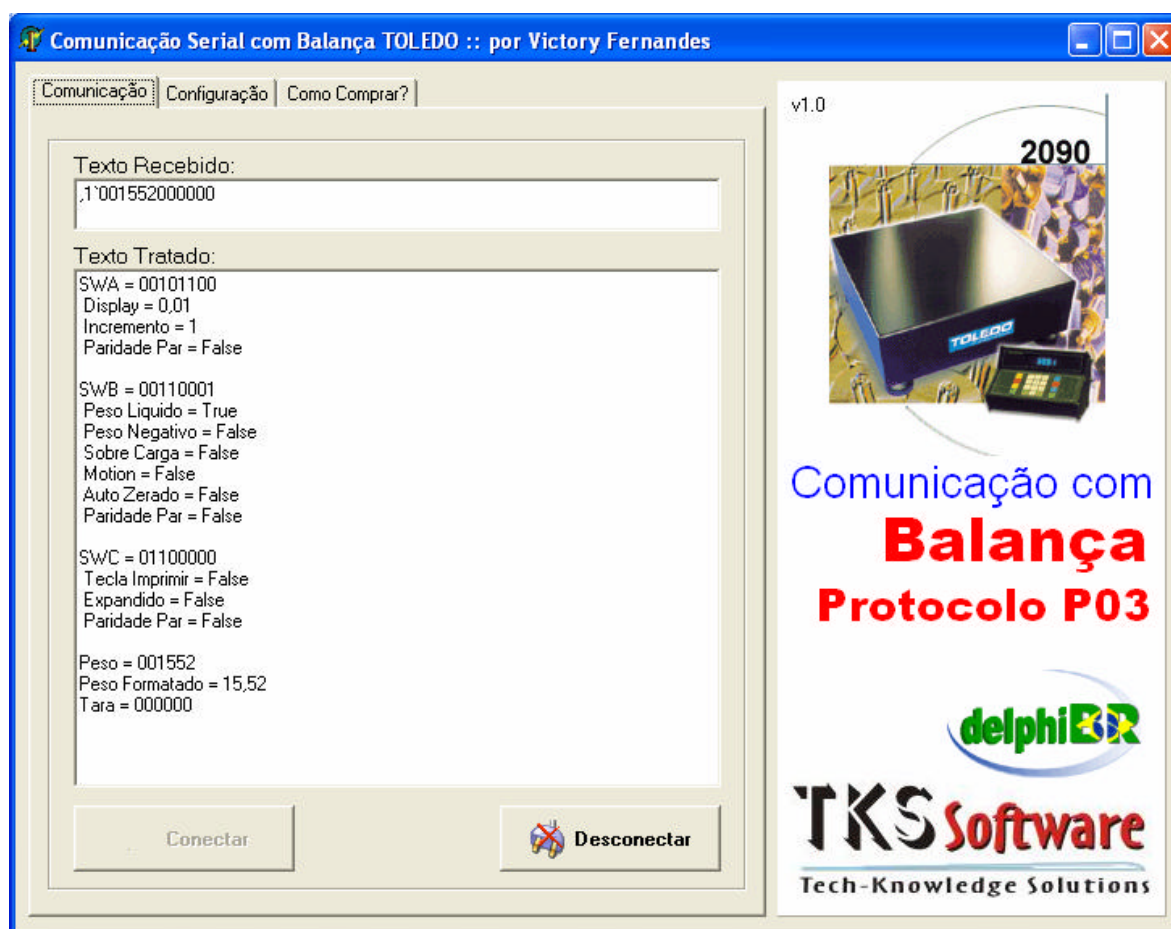


Figura 02: Demo de Comunicação com balança de pesagem TOLEDO utilizando a P03_Unit

O demo já vem previamente configurado para os parâmetros descritos na *Comunicação via HyperTerminal*. Sendo assim, basta clicar no botão *Conectar* para que ele passe a receber, assim como o *HyperTerminal*, a *string* enviada via porta serial pela balança no campo *Texto Recebido*.

A principal diferença no entanto, é que o demo apresenta separadamente no campo *Texto Tratado*, as informações enviadas pela balança de uma forma legível e de fácil interpretação.

Caso deseje recompilar o programa demo para alterar seus valores ou implementar outras funcionalidades, é necessário copiar o arquivo *P03_Unit.dcu* para o diretório *Lib* de instalação do *Delphi*, no caso do *Delphi 6* por exemplo, *C:\Arquivos de programas\Borland\Delphi6\Lib*

Agora vamos prosseguir, e saber como a comunicação, leitura e tradução da *string* lida é implementada pelo demo.

Comunicação do Demo com a porta serial

A comunicação com a porta serial para a leitura da *string* enviada pela balança pode ser feita utilizando qualquer componente disponível para este fim. Existem diversos componentes gratuitos para porta serial, disponíveis para *download* na Internet, e você pode utilizar aquele que achar melhor, ou até mesmo desenvolver seu próprio componente de comunicação.

No demo utilizamos o componente gratuito *TcommPortDriver* para efetuar a comunicação por já estarmos familiarizados com o mesmo. O *TCommPortDriver* é um componente gratuito para

Delphi que reúne todo o código de baixo nível necessário para o controle de portas seriais. Suas propriedades e métodos de conexão e transmissão de dados possibilitam utilizar as portas seriais facilmente.

Para instalar o componente *TCommPortDriver* descompacte o arquivo *CommPortDriver.zip*, que acompanha o demo em questão, na pasta desejada. No *Delphi* abra o arquivo *ComDrv32.dpk* na pasta de instalação do componente e clique no botão *Compile* e depois em *Install* para concluir a instalação.

Depois de configurados os parâmetros de comunicação descritos na *Comunicação via HyperTerminal*, o demo implementa o evento *OnReceiveData* do componente *TCommPortDriver* para fazer as chamadas das funções de tradução da *P03_Unit*, toda vez que o componente receber dados pela porta serial, como mostrado a seguir:

```
procedure TForm1.cpDrvReceiveData(Sender: TObject; DataPtr: Pointer;
  DataSize: Cardinal);
var
  temp: string;
  //declaração de variaveis para uso em Trata_Variaveis
  SWA, SWB, SWC, Peso, Tara: string;
  //declaração de variaveis para uso em Trata_SWA
  display, incremento: string; paridade_par_SWA: boolean;
  //declaração de variaveis para uso em Trata_SWB
  peso_liquido, peso_negativo, sobrecarga, motion, auto_zerado, paridade_par_SWB: boolean;
  //declaração de variaveis para uso em Trata_SWC
  tecla_imprimir, expandido, paridade_par_SWC: boolean;
begin
  //Captura dados lidos em formato string
  temp := StringOfChar(' ', DataSize);
  move(DataPtr^, pchar(temp)^, DataSize);

  //Remove caracteres invalidos da string lida
  temp := Remove_Invalidos(temp);

  //Caso o 15 bytes (sem checksum) não tenham sido todos lidos incrementa buffer
  if length(buffer) < 15 then
  begin
    buffer := buffer + temp;
  end;

  if length(buffer) >= 15 then
  begin
    //Limpa RichEdits
    RichEdit.Lines.Clear;
    RichEdited.Lines.Clear;

    //Acrescenta o string lida no primeiro RichEdit
    RichEdit.Lines.Append(buffer);

    //Trata string lida conforme protocolo P03 da Toledo
    Trata_Buffer(Buffer, SWA, SWB, SWC, Peso, Tara);
    Trata_SWA(SWA, display, incremento, paridade_par_SWA);
    Trata_SWB(SWB, peso_liquido, peso_negativo, sobrecarga, motion, auto_zerado, paridade_par_SWB);
    Trata_SWC(SWC, tecla_imprimir, expandido, paridade_par_SWC);

    //Adiciona variaveis tratadas ao segundo RichEdit
    with RichEdited.Lines do
    begin
      Append('SWA = ' + SWA);
      Append(' Display = ' + display);
      Append(' Incremento = ' + incremento);
      Append(' Paridade Par = ' + BoolToStr(paridade_par_SWA, True));
      Append('');
      Append('SWB = ' + SWB);
      Append(' Peso Liquido = ' + BoolToStr(peso_liquido, True));
      Append(' Peso Negativo = ' + BoolToStr(peso_negativo, True));
      Append(' Sobre Carga = ' + BoolToStr(sobrecarga, True));
      Append(' Motion = ' + BoolToStr(motion, True));
      Append(' Auto Zerado = ' + BoolToStr(auto_zerado, True));
      Append(' Paridade Par = ' + BoolToStr(paridade_par_SWB, True));
      Append('');
    end;
  end;
end;
```

```

Append('SWC = ' + SWC);
Append(' Tecla Imprimir = ' + BoolToStr(tecla_imprimir, True));
Append(' Expandido = ' + BoolToStr(expandido, True));
Append(' Paridade Par = ' + BoolToStr(paridade_par_SWC, True));
Append('');
Append('Peso = ' + Peso);
Append('Peso Formatado = ' + floattostr(strtofloat(Peso) * strtofloat(display)));
Append('Tara = ' + Tara);
end;

Buffer := '';
end;
end;

```

P03_Unit – Tratamento do protocolo de comunicação da balança

No evento de *OnReceiveData* descrito no código anterior, após fazer a leitura, a *string* lida da porta serial é armazenada na variável *Buffer*, são feitas então 4(quatro) chamadas às funções de tratamento do protocolo presentes na *P03_Unit*, são elas:

```

procedure Trata_Buffer(Buffer: String; var SWA, SWB, SWC, Peso, Tara: String);
procedure Trata_SWA(SWA: String; var display, incremento: string; var paridade_par: boolean);
procedure Trata_SWB(SWB: String; var peso_liquido, peso_negativo, sobrecarga,
motion, auto_zerado, paridade_par: boolean);
procedure Trata_SWC(SWB: String; var tecla_imprimir, expandido, paridade_par: boolean);

```

Como veremos no tópico *Protocolo de Comunicação – P03*, a *string* lida se divide em 5 partes principais que contém as informações enviadas pela balança: SWA, SWB, SWC, Peso e Tara.

Então, a primeira chamada feita à função *Trata_Buffer* passa a *string* lida como parâmetro, e serve para “quebrar” a *string* nessas 5(cinco) partes principais, retornando as mesmas em 5 variáveis respectivamente.

Como também veremos a seguir no tópico *Protocolo de Comunicação – P03*, as partes denominadas SWA, SWB e SWC devem então ser subdivididas para que se possa retirar delas as outras informações que elas contêm. Para isso então são feitas as chamadas às funções *Trata_SWA*, *Trata_SWB* e *Trata_SWC* passando como parâmetros as partes respectivas, e recebendo como retorno as variáveis relativas aos dados contidos dentro de cada uma dessas partes.

Pronto! Está feita de forma simples e muito rápida a comunicação e interpretação dos valores transmitidos pela balança TOLEDO, através da chamada de apenas 4(quatro) funções de tratamento, sem que fosse necessário conhecer o tratamento de baixo nível do protocolo em questão. Logo podemos ressaltar como vantagens na utilização a *P03_Unit*.

- ? Tratamento completo do protocolo P03 com abstração total da camada de baixo nível.
- ? Velocidade na implementação da comunicação com a balança.
- ? Facilidade na obtenção das informações transmitidas.
- ? Demo de comunicação com a balança.
- ? Fontes em *Delphi* totalmente comentados.
- ? FAQ de implementação e uso.

A TOLEDO possui no mercado uma infinidade de modelos de balanças, e muitas pessoas se questionam sobre a compatibilidade da *P03_Unit* com esse ou aquele modelo. É importante lembrar que a compatibilidade da *P03_Unit* não é função necessariamente do modelo, e sim do protocolo de comunicação utilizado entre a balança e o computador.

A *unit* está implementada para interpretar o protocolo de comunicação P03, utilizado no modelo mostrado na *Figura 02* e em diversos outros. No entanto, sugiro que o desenvolvedor entre em contato com o suporte do fabricante e confirme se o seu modelo utiliza o protocolo P03, para só então realizar os testes de comunicação com o demo disponibilizado.

Simulador de Balanças TOLEDO

Agora que já vimos como fazer a comunicação utilizando a balança, apresentamos uma solução para uma das maiores dificuldades encontradas por implementadores nesta área: a necessidade de ter uma balança disponível no laboratório de desenvolvimento para fazer testes.

O simulador, mostrado na *Figura 03*, possibilita realizar os testes de implementação do seu software para comunicação com balanças TOLEDO, sem que para isso seja necessário ter a balança, bastando que este esteja sendo executado em um outro computador conectado pela porta serial.



Figura 03: Simulador de Balanças TOLEDO – Envio de informações

Na aba *Configuração*, podem-se configurar todas as opções de valores transmitidos, e após clicar em *Conectar*, o simulador, além de transmitir as informações pela porta serial, exibe no campo *String Tratada* todos os parâmetros enviados e seus valores atribuídos, tais como Peso, Tara e etc. Isso facilita as atividades de batimento entre informações enviadas e recebidas pelo implementador, que tem todas as informações exibidas em tela, ao contrário da própria balança que somente informa o peso no *display*.

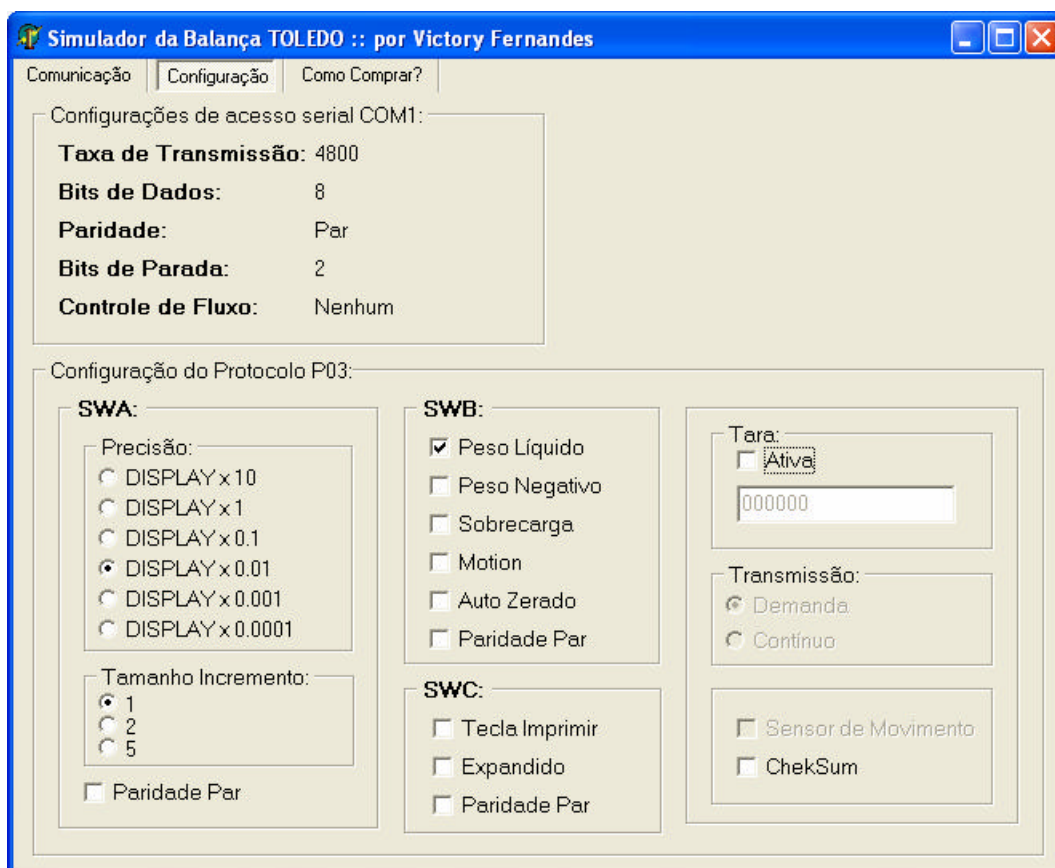


Figura 04: Simulador de balanças TOLEDO – Configurações do simulador

É importante ressaltar que algumas funcionalidades como modo de transmissão contínua e sensor de movimento, não estão implementadas nesta versão, ficando assim fixado o *modo transmissão* sob demanda e *sensor de movimento* desabilitado.

Protocolo de Comunicação – P03

Agora que já vimos como fazer a implementação da comunicação com a balança, vamos aprofundar um pouco mais o nosso estudo, e entender a forma como as informações são enviadas ao computador.

A balança transmite as informações para o computador obedecendo a um protocolo de comunicação do fabricante chamado P03. Este protocolo define as posições das variáveis na sequência binária transmitida bem como os valores assumidos pelas mesmas.

O formato básico da sequência binária transmitida que será recebida pelo computador é mostrado abaixo:

```
STX,SWA,SWB,SWC,I,I,I,I,I,I,T,T,T,T,T,T,CR,(CS)
```

Na sequência binária mostrada, os *bits*, ou grupo de *bits* indicam configurações ou estados da balança. Vamos agora analisar cada parte da sequência e o seu significado na comunicação.

STX – Start of Text

Assume o valor 02 e é apenas um *byte* de sincronia da comunicação.

SWA – Status Word “A”

Os *bits* 2, 1, 0 indicam a configuração de precisão da balança e podem assumir os valores:

```
001 = x 10
010 = x 1
011 = x 0.1
100 = x 0.01
101 = x 0.001
110 = x 0.0001
```

Os *bits* 4 e 3 indicam a configuração de tamanho do incremento da balança e podem assumir os valores:

```
01 = Tamanho do Incremento 1
10 = Tamanho do Incremento 2
11 = Tamanho do Incremento 5
```

Os *bits* 5 e 6 assumem sempre o valor 01.

O *bit* 7 indica a configuração de paridade da comunicação serial da balança e assume os valores:

```
0 = Paridade Ímpar
1 = Paridade Par
```

SWB – Status Word “B”

```
Bit 0 = Peso Líquido = 1
Bit 1 = Peso Negativo = 1
Bit 2 = Sobrecarga = 1
Bit 3 = Motion = 1
Bit 4 = Sempre = 1
Bit 5 = Sempre = 1
Bit 6 = Se AUTO Zerado = 1
Bit 7 = Paridade Par
```

SWC – Status Word “C”

```
Bit 0 = Sempre = 0
Bit 1 = Sempre = 0
Bit 2 = Sempre = 0
Bit 3 = Tecla Imprimir = 1
Bit 4 = Expandido = 1
Bit 5 = Sempre = 1
Bit 6 = Sempre = 1
Bit 7 = Paridade Par
```

I – Peso indicado no *display* (Líquido ou Bruto). Se existir sobrecarga da balança, o campo de peso *IIIIII* apresentará 000000.

T – Tara configurada na balança.

CR – *Carriage Return* – Assume o valor 0DH (10 em Hexadecimal) e precede a informação de CheckSum.

CS – *CheckSum* – Caso o registrador C12 da balança esteja configurado como = L, o *checksum* dos dados é enviado pela balança.

Percebe-se então que a *string* lida se divide em:

- ? 1 *Byte* de sincronia STX que pode ser usado para subdividir a *string* recebida, caso o envio ou recebimento seja concatenado.
- ? 5 partes principais que contém as informações enviadas pela balança, SWA, SWB, SWC, Peso e Tara que devem ser tratadas separadamente para a obtenção das informações transmitidas.

- ? 1 Byte de *Checksum*, que pode ser usado para identificar erros na *string* lida através da comparação dos valores.

Considerações Finais

Esperamos com este artigo exemplificar o processo, esclarecer dúvidas e abrir novas possibilidades aos leitores, facilitando a implementação de sistemas para operação em conjunto com os diversos modelos de balança TOLEDO, bem como facilitando a implementação para modelos de outros fabricantes, uma vez que o paradigma apresentado é, em muito, semelhante aos demais utilizados no mercado.

Victory Fernandes é Engenheiro Mestrando em Redes de Computadores, e desenvolvedor sócio da TKS Software - Soluções de Automação e Softwares Dedicados.

Pode ser contactado em victory@igara.com.br, ou através dos sites www.igara.com.br - www.victory.hpg.com.br

Murilo Plínio é estudante de Eng. Mecatrônica, Técnico em Informática e desenvolvedor da TKS Software.

Pode ser contactado em muriloplinio@yahoo.com