

Componente de Comunicação Telemed

TelemedDLL Single v2.0.0

1- Objetivo

O objetivo deste documento é descrever tecnicamente a biblioteca de comunicação TelemedDLL_Single_v2_0_0.dll.

Este componente foi desenvolvido com o objetivo de disponibilizar uma biblioteca de comunicação para facilitar a integração dos equipamentos ELET1, MT1 e MTV1 com os sistemas de gerenciamento de postos de gasolina ou qualquer sistema que necessite desta comunicação.

Embora o protocolo implementado nos equipamentos seja Multi-ponto, ou seja, pode-se conectar vários equipamentos de tipos diferentes em uma mesma conexão física, esta versão da biblioteca foi desenvolvida para suportar um único equipamento em cada conexão, independentemente do seu tipo.

Os modelos de equipamentos suportados por esta biblioteca são: ELET1, MT1 ou MTV1.

2- Considerações Gerais

A Telemed não se responsabiliza por possíveis erros que possam ocorrer na utilização da biblioteca nem mesmo por inconvenientes causados por seu uso. É de inteira responsabilidade dos integradores a correções de possíveis erros e adaptações necessárias à utilização da mesma.

Para suporte contatar a Telemed pelo telefone: **(11) 3674-7799**.

3- Plataforma

Todos os programas foram desenvolvidos na linguagem C# para a plataforma Windows no ambiente de desenvolvimento Visual Studio 2019.

Os binários da biblioteca devem ser utilizados nas plataformas que suportem o formato de biblioteca DLL do Windows.

Para utilizar esta biblioteca em outra plataforma, será necessário a conversão/adaptações, pelos integradores, para a plataforma desejada.

4- Instalação

Copiar o conteúdo do diretório de distribuição recebido para o diretório desejado.

A estrutura de diretório será:

- TelemedDLL Single v2.0.0
 - Documento
 - Contém os documentos referentes a DLL.
 - Executável
 - TelemedDLL_Single_v2_0_0.dll
 - TstConsoleTelemedDLL_Single.exe

- TstConsoleTelemedDLL_Single.exe.config
- TstTelemedDLL_Single.exe
- TstTelemedDLL_Single.exe.config
- Fontes
 - Components
 - Console
 - TstTelemedDLL

No subdiretório Executável temos a biblioteca TelemedDLL_Single_v2_0_0.dll e um programa de teste de comunicação que pode ser utilizado também como exemplo de utilização da biblioteca.

A estrutura de Fontes abaixo contém as Soluções/Projetos da DLL e do programa de Teste/Exemplo de Comunicação.

Essa estrutura representa a estrutura de solução/projeto do Visual Studio 2019 do Windows.

5- Programa de Exemplo

O programa TstTelemedDLL_Single.exe pode ser executado, como uma aplicativo normal do Windows, para os testes de comunicação. Este programa possibilita a execução de todos os comandos disponíveis nos equipamentos suportados. Ao executar será mostrado na tela o comando enviado e a resposta formata das mensagens recebida.

Se preferir o mesmo programa poderá ser executado através do programa TstConsoleTelemedDLL_Single.exe, neste caso além de executar o programa, será aberto um console no qual serão mostradas as mensagens processadas pelos componentes Comm e ComIO da DLL.

Caba ressaltar que as configurações de fabrica dos equipamentos saem com o tipo de comunicação para IMPRESSORA, portanto antes de utilizar este programa é necessário configura/alterar a configuração para MICROCOMPUTADOR com o endereço 01.

6- Manual do Componente

6.1- COMIO - PORTA SERIAL

6.1.1- Propriedades

Todas as propriedades foram definidas com privadas, por se tratarem de parâmetros que na maioria das vezes são os valores utilizados pelos equipamentos.

Todos os equipamentos saem de fábrica com estes parâmetros já configurados com estes valores.

6.1.2- ComIO - construtor

Sintaxe: ComIO(string pPort)

O construtor recebe como parâmetro o nome da porta serial COM ao qual o equipamento está conectado e cria o objeto ComIO. O ComIO é derivado do componente SerialPort, portanto caso necessário, pode-se acessar todas as propriedades e os métodos do objeto SerialPort.

6.1.3- Open

Sintaxe: Open()

Abre a porta serial COM cujo nome foi fornecido na criação do componente e com as características definidas nas propriedades privadas da classe. Caso seja necessário essas propriedades podem ter seus valores alterados, mas será necessária uma recompilação do componente.

6.1.4- Close

Sintaxe: Close()

Fecha a porta serial COM.

6.1.5- Send

Sintaxe: Send(byte[] pMessage, int pTimeout)

Recebe como parâmetro um array de bytes com a mensagem a ser enviada ao equipamento (<stx><informação><etx><lrc> - descrita no manual de comunicação) e o tempo máximo de espera (milissegundos) para a mensagem ser enviada.

6.1.6- Receive

Sintaxe: byte[] Receive(int pTimeout)

Recebe como parâmetro o tempo máximo de espera (milissegundos) para a resposta da solicitação e retorna a mensagem recebida do equipamento (<stx><informação><etx><lrc> - descrita no manual de comunicação).

Como cada solicitação tem processamentos diferentes e envolve dispositivos com tempo de resposta diferentes, portanto cada solicitação deverá ter um tempo de espera de acordo com cada solicitação. Veja programa exemplo para calibrar os tempos de espera para cada solicitação. Note que, em caso de solicitações que envolva várias respostas, o tempo de espera é reiniciado para cada mensagem retornada.

6.2- MESSAGE - MENSAGENS

6.2.1- Propriedades

Buf - retorna um array de bytes com a mensagem

Length - retorna um inteiro com o tamanho total da mensagem do **<stx>** ao **<lrc>** inclusive

Information - retorna uma string com a parte de informação (Code + Body) da mensagem do **<stx>** ao **<etx>** exclusive

Code - retorna uma string de tamanho 2 com o código do comando da mensagem

Body - retorna uma string com os parâmetros correspondentes ao comando da mensagem

LRC - retorna um byte com o LRC calculado da mensagem (XOR dos bytes de **<stx>** (exclusive) até **<etx>** (inclusive))

Next - retorna true se a mensagem tem continuação (terminador **<soh>**) e false caso contrário (terminador **<cr>**)

Error - retorna true se é uma mensagem de erro (ETT) e false caso contrario

6.2.2- Message - construtor

Sintaxe: Message(string pInformation)

Message(string pCode, string pBody)

Message(byte[] pBuf)

Recebe como parâmetro uma string com o campo de **<informação>** da mensagem ou uma string com o Código e outra com os Parâmetros e retorna um objeto Message. Também pode receber um array de byte com uma mensagem completa (**<stx><informação><etx><lrc>**) e retorna um objeto Message.

6.2.3- ToString

Sintaxe: ToString()

Retorna uma string formatada para impressão composta dos campos da mensagem.

6.3- COMM - EQUIPAMENTO

6.3.1- Propriedades

PortName - retorna uma string com o nome da porta serial

Type - retorna o tipo de equipamento

Address - retorna um inteiro com endereço do equipamento

NumeroTanques - retorna um inteiro com número de tanques configurado no equipamento

OFE - retorna uma string com o número da OFE

6.3.2- ComIO - construtor

Sintaxe: Comm(eType pType, int pAddress, string pPortName)

Recebe como parâmetros o Tipo do equipamento, o endereço do equipamento e o nome da porta serial (COM) e cria o componente Comm e ComIO

6.3.3- Open

Sintaxe: Open()

Abre a porta serial e seleciona o equipamento (envia a mensagem de seleção do equipamento) e em seguida lê a configuração (envia a mensagem de leitura da configuração): Número de tanque, Número da OFE

6.3.4- Close

Sintaxe: Close()

Fecha a porta serial.

6.3.5- Send

Sintaxe: Send(string pCommand, string pBody, int pTimeout)

Send(string pInformation, int pTimeout)

Recebe com parâmetro uma string com o comando outra com os parâmetros do comando e o tempo de espera máximo de espera em milisegundos para envia a mensagem.

Monta a mensagem de transmissão através do Message e envia a mensagem ao equipamento através da ComIO.

Um valor de 1000 milisegundo para o tempo de espera de envio normalmente é mais que suficiente.

6.3.6- Receive

Sintaxe: string Receive(int pTimeout)

Recebe como parâmetro o tempo máximo de espera de recepção de mensagem e espera a mensagem enviada pelo equipamento.

Caso a mensagem tenha continuação espera as próximas mensagens até receber a última e retorna uma string concatenando os campos de <**informações**> de todas as mensagens.

Como cada solicitação tem processamentos diferentes e envolve dispositivos com tempo de resposta diferentes, cada solicitação deverá ter um tempo de espera de acordo com

cada solicitação. Veja programa exemplo para calibrar os tempos de espera para cada solicitação. Note que, em caso de solicitações que envolva várias respostas como por exemplo: a solicitação Medição Automática - MA, o tempo de espera é reiniciado para cada mensagem retornada.