

# **SERIALMFD.DLL**

**Guia de Programação**  
**Versão 1.0.0.0**

**Junho 2004**



INTRODUÇÃO.....	3
ABRIR PORTA.....	4
APURAÇÃO AUTOMÁTICA DA VELOCIDADE.....	5
NOTIFICAÇÃO DE MENSAGEM.....	6
ASSINALAR TEMPO PARA NOTIFICAR MENSAGEM DE TIME OUT .....	7
ALTERAR PRIORIDADE DA CPU .....	7
ALTERAR TEMPO DE CONEXÃO .....	8
HABILITAR TESTE DE SINAIS.....	8
ALTERA TEMPO DO ACK .....	9
ENVIAR DE COMANDOS.....	10
RECEBER DADOS DO ECF.....	11
FECHAR A PORTA.....	11

Neste documento, serão descritas as funcionalidades da **SerialMFD.DLL**. Esta biblioteca estabelece a comunicação entre o software aplicativo de Automação Comercial com os Emissores de Cupom Fiscal SWEDA da linha ST (IF ST100 e IF ST1000).

A conexão, através da Porta SERIAL (RS232) utiliza o protocolo de comunicação STX. Dependendo da função, estão previstos quatro tipos de dados que podem ser informados:

<b>void</b>	= não há retorno de valor da função.
<b>Int</b>	= inteiro de 32 bits com sinal.
<b>unsigned int</b>	= inteiro de 32 bits sem sinal.
<b>BYTE *</b>	= referência à uma área alocada pelo compilador ou adquirida do sistema operacional. Muitos compiladores exigem o preenchimento da área para que ela se torne válida.

O formato de chamada da função deve ser do tipo `__stdcall`, o mesmo usado para chamar as funções WIN32 API.

Funções disponíveis:

<b>ECFOpenSerial ( )</b>	= Abre a porta serial.
<b>ECFGetBaudRateSerial( )</b>	= Apura a velocidade após abrir a porta.
<b>ECFGetMessageONSerial ( )</b>	= Habilita a notificação de mensagens.
<b>ECFSetTempoOFFSerial ( )</b>	= Define tempo de notificação (OFF LINE).
<b>ECFSetPrioritySerial ( )</b>	= Altera prioridade da CPU.
<b>ECFSetTimeConexaoSerial ( )</b>	= Tempo de espera CTS / DSR.
<b>ECFSetSinaisSerial ( )</b>	= Habilita/Desabilita teste de CTS / DSR.
<b>ECFSetTempoAckSerial( )</b>	= Tempo de espera do ACK.
<b>ECFWriteSerial ( )</b>	= Envia comandos.
<b>ECFReadSerial ( )</b>	= Consulta e busca registros recebidos.
<b>ECFCloseSerial ( )</b>	= Fecha o protocolo.

Chamar a função *ECFOpenSerial ( )* para efetuar a abertura da porta.

```
unsigned int ECFOpenSerial ( unsigned int  Porta,
                             unsigned int  Velocidade,
                             unsigned int  Timeout,
                             unsigned int  Log,
                             unsigned int  Mostra,
                             BYTE *        "Serial" )
```

**Porta** = Informar o número da porta.  
           1 = COM1.  
           2 = COM2.  
           3 = COM3.  
           até 8.

**Velocidade** = Valores possíveis:  
                   9600  
                   19200  
                   38400  
                   57600  
                   115200.

**Timeout** = Expressa em segundos o tempo de espera do CTS / DSR referente ao comando enviado. Deve ser maior que UM e não exceder UM minuto.

**Log** = Habilita ou desabilita a geração do arquivo tipo LOG. Ele é gerado no diretório C:\TEMP\ECF-MFD.  
           Zero = O arquivo não é gerado.  
           UM = Semanal.  
           > 1 = Define o tamanho limite do arquivo em Kbytes. Exemplo: valor dois = 2 K.  
                   Atingindo o limite, o arquivo é apagado.

**Mostra** = Habilita a exibição de uma caixa de diálogo quando ocorrer erro como, por exemplo, falta de CTS / DSR.  
           TRUE = mostrar.  
           FALSE = não mostrará mensagem de erro.

**"SERIAL"** = Texto em letras maiúsculas e / ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", assumirá que o formato da chamada da função não é do tipo "stdcall" e interromperá o comando.

**Retorno em um INTEIRO SEM SINAL DE 32 BITS:**

```
FALSE (ZERO) = OK
TRUE       = ERRO
```

Caso não consiga abrir a porta, exibirá a ocorrência em uma caixa de diálogo se o valor do argumento **Mostra** for TRUE. Velocidade superior a 19200 alocará prioridade da CPU acima do normal para o protocolo, podendo esta prioridade ser alterada posteriormente através da função **ECFSetPrioritySerial ( )**.

A velocidade real poderá ser automaticamente apurada após a abertura da porta, usando a função **ECFGetBaudRateSerial ( )**. Neste caso, informar um valor válido (Exemplo: 9600) no argumento **Velocidade** desta função. Se a porta foi aberta com sucesso, a aplicação poderá enviar e receber comandos indefinidamente.

É assumido como padrão o teste dos sinais CTS / DSR. O sinal CTS indicaria que o cabo está presente enquanto o DSR se o ECF está ligado. O teste dos sinais deve ser desabilitado se a aplicação estiver usando a porta do Fisco. A função **ECFSetSinaisSerial ( )** permite habilitar e desabilitar os sinais. A função **ECFSetTimeConexaoSerial ( )** poderá ser usada para alterar o número de tentativas de transmissão de um comando (o padrão é 3) e para alterar o tempo de espera do CTS/DSR sem precisar fechar e reabrir a porta.

O tamanho do buffer serial no ECF é de 1024 bytes.

---

## APURAÇÃO AUTOMÁTICA DA VELOCIDADE

---

Chamar a função **ECFGetBaudRateSerial( )** para apurar a velocidade.

**Int**    **ECFGetBaudRateSerial ( )**

**Retorno em um INTEIRO SEM SINAL DE 32 BITS:**

**< 0    =    erro.**  
**= 0    =    ECF desligado.**  
**> 0    =    velocidade detectada. O valor da velocidade é retornado (9600,19200, ...).**

O aplicativo abre a porta informando uma velocidade qualquer (Ver lista de valores possíveis). Antes de iniciar a transmissão ou após a abertura da porta, a função abaixo poderá ser chamada para determinar a velocidade real programada no ECF. A programação da porta serial no ECF é feita, ligando o ECF e pressionando os dois botões traseiros simultaneamente.

## NOTIFICAÇÃO DE MENSAGEM

---

Chamar a função **ECFGetMessageONSerial ( )** para habilitar a notificação.

**unsigned int** ECFGetMessageONSerial ( **unsigned int** *HJanela*,  
**BYTE \*** *"SERIAL"*)

*HJanela* = Valor que identifica o HANDLE da janela para onde o WINDOWS enviará a mensagem.

*"SERIAL"* = Texto em letras maiúsculas e/ou minúsculas, é usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", assumirá que o formato da chamada da função não é do tipo "stdcall" e interromperá o comando.

**Retorno em um INTEIRO SEM SINAL DE 32 BITS, indicando o valor da mensagem a ser testado na interrupção do evento.**

**Exemplo:**

```
Status = ECFOpenSerial( PortaSerial,
                        Velocidade,
                        TimeOut,
                        TipoLog,
                        1,
                        "serial"); // inic porta serial
                                // informa handle da janela
Mensagem = ECFGetMessageONSerial(Janela,"SERIAL");
RET
```

----- interrupção -----

Na interrupção do evento é informado a mensagem e um segundo valor (**WPARAM**) que indicará:

**FALSE** = causa TIME OUT.  
**TRUE** = chegou registro.

```
Janelador (HWND hDlg, UINT message, UINT WPARAM, LONG lParam)
{
    if ((message == Mensagem) && Mensagem)
    {
        // se veio mensagem do ECF - processa
        ProcessarECF(WPARAM);
    }
}
```

Não é exigido que a porta serial esteja aberta.

O aplicativo informa o HANDLE da sua janela para que o protocolo possa avisar a aplicação que um evento ocorreu. Os tipos de eventos que gerarão a notificação são:

- TIME OUT.
- CHEGOU REGISTRO.

Para que a aplicação possa identificar a causa da interrupção, será retornado aqui o valor da mensagem (MESSAGE) que será usado para indicar que a causa do evento é comunicação.

---

## ASSINALAR TEMPO PARA NOTIFICAR MENSAGEM DE TIME OUT

---

Chamar a função *ECFSetTempoOFFSerial ( )*.

**void** ECFSetTempoOFFSerial ( **unsigned int** *Tempo*,  
**BYTE \*** *"SERIAL"* )

*Tempo* = Intervalo em segundos de espera de uma resposta do ECF. Se não chegou qualquer registro do ECF, o protocolo notificará a aplicação.

*"SERIAL"* = Texto em letras maiúsculas e/ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", assumirá que o formato da chamada da função não é do tipo "stdcall" e interromperá o comando.

Assinalando um tempo em segundos, o protocolo avisará o aplicativo que o tempo de espera de um registro do ECF esgotou. A notificação somente ocorrerá se o aplicativo passar o HANDLE da janela usando a função *ECFGetMessageONSerial ( )*.

---

## ALTERAR PRIORIDADE DA CPU

---

Chamar a função *ECFSetPrioritySerial ( )*.

**void** ECFSetPrioritySerial ( **int** *prior*,  
**BYTE \*** *"SERIAL"* )

*Prior* = Valores possíveis:  
+2 = THREAD\_PRIORITY\_HIGHEST  
+1 = THREAD\_PRIORITY\_ABOVE\_NORMAL  
-1 = THREAD\_PRIORITY\_BELOW\_NORMAL  
-2 = THREAD\_PRIORITY\_LOWEST  
0 = THREAD\_PRIORITY\_NORMAL

*"SERIAL"* = Texto em letras maiúsculas e / ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", assumirá que o formato da chamada da função não é do tipo "stdcall" e interromperá o comando.

A prioridade já é estabelecida na abertura da porta.  
Esta função permite alterar o padrão estabelecido.

## ALTERAR TEMPO DE CONEXÃO

---

Chamar a função *ECFSetTimeConexaoSerial* ( ).

```
void ECFSetTimeConexaoSerial ( unsigned int Tempo,  
                               unsigned int Tentativas,  
                               BYTE * "SERIAL" )
```

*Tempo* = Intervalo de segundos. Mínimo é UM SEGUNDO.

*Tentativas* = Número de tentativas. Mínimo é UMA TENTATIVA.

*"SERIAL"* = Texto em letras maiúsculas e/ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", assumirá que o formato da chamada da função não é do tipo "stdcall" e interromperá o comando.

A aplicação poderá definir o tempo máximo de espera dos sinais CTS e DSR do ECF, se o teste estiver habilitado.

## HABILITAR TESTE DE SINAIS

---

Chamar a função *ECFSetSinaisSerial* ( ).

```
void ECFSetSinaisSerial ( unsigned int CTS,  
                          unsigned int DSR,  
                          BYTE * "SERIAL" )
```

*CTS* = Valores:  
False (zero) = desabilita o teste de CTS.  
True = habilita o teste de CTS.

*DSR* = Valores:  
False (zero) = desabilita o teste de DSR.  
True = habilita o teste de DSR.

*"SERIAL"* = Texto em letras maiúsculas e/ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", assumirá que o formato da chamada da função não é do tipo "stdcall" e interromperá o comando.

## ALTERA TEMPO DO ACK

---

Chamar a função *ECFSetTempoAckSerial ( )*.

**Void** ECFSetTempoAckSerial (**unsigned int** *Tempo*,  
**BYTE \*** *"SERIAL"*)

*Tempo* = Intervalo de segundos. Mínimo é DOIS SEGUNDOS.

*"SERIAL"* = Texto em letras maiúsculas e/ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", assumirá que o formato da chamada da função não é do tipo "stdcall" e interromperá o comando.

A aplicação poderá alterar o tempo de espera do ACK. O caracter ACK é enviado pelo ECF após ter recebido um registro consistente do aplicativo. O tempo padrão é 5 segundos.

## ENVIAR DE COMANDOS

---

Chamar a função *ECFWriteSerial* ( ) para enviar qualquer comando.

Por exemplo, se a aplicação quiser obter a data e a hora do relógio do ECF, deverá informar *\*34/16*. O protocolo envolverá o texto com os caracteres STX (0x02), ETX (0x03) e o CHECK-SUM calculado, enviando o comando para o ECF e aguardando ACK. O recebimento do caracter ACK do ECF indicará que o comando foi recebido. Esta função retorna para a aplicação após ter recebido o ACK ou ter esgotado o tempo de espera.

```
int ECFWriteSerial ( BYTE *      Comando,  
                   unsigned int Tamanho,  
                   BYTE *      "SERIAL" )
```

*Comando* = Informar o comando a ser enviado.  
O caracter enviado pela aplicação pode variar de 0x00 a 0xFF.

*Tamanho* = Comprimento do texto a ser enviado. Limite é 1 k.

*"Serial"* = Texto em letras maiúsculas e/ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", a DLL assume que o formato da chamada da função não é "stdcall".

**Retorno:**    **FALSE** (zero)    - OK - ACK RECEBIDO DO ECF  
              -3                - Protocolo ocupado. Tentar mais tarde.  
              Outro valor      - ERRO.

Se a transmissão está sendo feita em uma interrupção de evento e o retorno da função é -3, o seguinte procedimento é recomendado:

- Assinalar um tempo de notificação usando a função *ECFSetTempoOFFSerial* ( ).
- Sair da interrupção do evento.
- O protocolo voltará a chamar novamente porque chegou registro ou esgotou o tempo acima. A aplicação poderá então tentar enviar o comando.

---

## RECEBER DADOS DO ECF

---

Chamar a função **ECFReadSerial ( )** para obter o registro recebido do ECF. Os caracteres de controle usados pelo protocolo são removidos e é retornado apenas o texto.

```
int ECFReadSerial ( BYTE *      Status,  
                   unsigned int extensão,  
                   BYTE *      "SERIAL" )
```

***Status*** = Área que receberá a resposta do ECF. Alocar 1 Kbyte. Algumas linguagens exigem a iniciação da área pelo aplicativo (preenchendo de espaços ou qualquer outro caracter) para que o espaço seja efetivamente alocado.

***extensão*** = Extensão física da área ***Status***.

***"SERIAL"*** = Texto em letras maiúsculas e/ou minúsculas, usado apenas para a consistência dos argumentos da função. Se não for igual a "serial", a DLL assume que o formato da chamada da função não é "stdcall".

**Retorno:** 0 = sem registro do ECF.  
> 0 = Tem registro. Retornando o comprimento.  
-1 = ERRO.  
-2 = extensão não é suficiente para receber a mensagem.  
-3 = Protocolo ocupado. Consulta não pode ser feita agora.

Se a chamada está sendo feita em uma interrupção de evento e o retorno da função é **-3**, o seguinte procedimento é recomendado:

- Assinalar um tempo de notificação usando a função **ECFSetTempoOFFSerial ( )**.
- Sair da interrupção do evento.
- O protocolo voltará a chamar novamente porque chegou registro ou esgotou o tempo acima. A aplicação poderá então tentar consultar se tem registro agora.

---

## FECHAR A PORTA

---

Usado antes de encerrar a aplicação e serve para liberar recursos alocados do sistema.

**void** ECFCloseSerial ( )