

## As Impressoras Não Fiscais

### **MP-4000 TH / MP-4200 TH**



Varejos que possuem grande volume de impressão de tickets, ordens de serviço e outros comprovantes não fiscais, têm à sua disposição a impressora mais eficiente do mercado: a MP-4000 TH. Uma impressora robusta, veloz e com a qualidade que só a Bematech pode oferecer.

A MP-4200 TH é a evolução das impressoras térmicas, atende todos os segmentos do varejo que possuem grande volume de impressão. Seu desempenho e qualidade otimizam todo o processo de atendimento ao cliente.

### **MP-2500 TH**



A MP-2500 TH traz durabilidade e qualidade de impressão a baixo custo. Atende a todos os segmentos em operações não fiscais.

### **MP-20 MI**

A MP-20 é a impressora mais tradicional do mercado, indicada para impressão de tickets, ordens de serviços e outros comprovantes sem valor fiscal.



## Apresentando a DLL

Para utilizar as impressoras não-fiscais da Bematech em ambiente Windows, você poderá usar uma DLL de comunicação própria. Esta DLL tem como objetivo, comunicar o seu Aplicativo com a impressora não-fiscal.

A comunicação é realizada através de funções específicas da MP2032.DLL.

A MP2032.DLL deve ser utilizada em linguagens de programação da plataforma Windows®.

A DLL pode estar junto com a aplicação (em um mesmo diretório) ou no diretório de sistema do Windows®, exemplo: C:\Windows\System32.

As linguagens de programação para Windows® oferecem, no geral, maneiras de fazer o acesso as funções de uma DLL e, para isso, o programador deve saber:

- a. O nome da DLL;
- b. A declaração da DLL e sua funções;
- c. O nome da função;
- d. O tipo de retorno da função (embora nem sempre seja necessário) e;
- e. Os parâmetros esperados pela função.

Seu funcionamento é definido através de funções, acessadas pela aplicação.

## Declarando a DLL

A MP2032.dll deve ser declarada com suas funções respectivas no aplicativo para que o mesmo possa interagir com a impressora.

Cada linguagem de programação possui uma maneira diferente de declarar uma DLL externa. Consulte sempre o manual de programação de sua linguagem para verificação deste procedimento.

↳ [Declarando em Visual Basic](#)



↳ [Declarando em Delphi](#)

## Interface Serial

Para a comunicação da MP2032.dll com a interface serial da impressora, basta copiar a dll para a pasta de sistema do Windows (ex: \Windows\System32) ou para a pasta da aplicação.

Detalhe: Deve ser informada a string "COM1" ou o número da serial onde a impressora foi conectada para a função [IniciaPorta](#).

## Interface LPT

**Observação:** Para efetuar a instalação, o computador deve ser logado em modo Administrador.

ü Para **Windows 98®**:

Para esta versão do sistema operacional, a aplicação passará a usar diretamente a **MP2032.DLL** e suas respectivas funções.

ü Para **Windows XP®, Windows VISTA®** ou **Windows 7®**:

Para esta versão do sistema operacional, é necessário executar o aplicativo **installer.exe** que se encontra no caminho "Pacote DLL\LPT". Após realizado este procedimento, a aplicação passará a usar diretamente a dll e suas respectivas funções.

O computador precisa ser reiniciado.

Detalhes:

- Deve ser informada a string "LTP1" para a função [IniciaPorta](#).

## Interface USB

**Observação:** Para efetuar a instalação, o computador deve ser logado em modo Administrador.

ü Para **Windows XP®**, **Windows VISTA®** ou **Windows 7®**:

- Modelos MP-4000 TH e MP-4200 TH

Para esta versão do sistema operacional, é necessário executar o aplicativo **installer.exe** que se encontra no caminho "Pacote DLL\USB" e seguir as orientações. Após realizado este procedimento, a aplicação passará a usar diretamente a dll e suas respectivas funções.

O computador precisa ser reiniciado.

- Modelo MP-2500 TH

Executar o aplicativo BemaUni\_setup\_v2.0.8.exe que se encontra no caminho "VirtualComUSB\32 bits" e seguir as orientações.

Após instalado, irá surgir uma porta COM virtual na lista de portas disponíveis no Gerenciador de Dispositivos. A COM que estiver lá é a que será usada na abertura da porta (função [IniciaPorta](#)).

O computador precisa ser reiniciado.

Detalhe:

- Deve ser informada a string "USB" para a função [IniciaPorta](#).
- Caso seja necessário reinstalar a interface de comunicação, execute o arquivo BemaUninstaller.exe que se encontra na pasta "Pacote DLL\LPT".

## Interface Ethernet

Para a comunicação da MP2032.dll com a interface ethernet da impressora, basta copiar a dll para a pasta de sistema do Windows (ex: \Windows\System32) ou para a pasta da aplicação.

Observações:

- Deve ser informada a string com o IP da impressora para a função [IniciaPorta](#).
- Para configura o IP e demais informações da Wi-Fi, utilize o "Software do Usuário" que se encontra na área de downloads do site Bematech.

## Interface Wi-Fi

Esta interface está disponível apenas para o modelo MP-4200 TH.

Para a comunicação da MP2032.dll com a interface Wi-Fi da impressora, basta copiar a dll para a pasta de sistema do Windows (ex: \Windows\System32) ou para a pasta da aplicação.

Observações:

- Deve ser informada a string com o IP da impressora para a função [IniciaPorta](#).
- Para configura o IP e demais informações da Wi-Fi, utilize o "Software do Usuário" que se encontra na área de downloads do site Bematech.

## Função AcionaGuilhotina()

Esta função aciona a guilhotina, contando o papel em modo parcial ou total.

### **Parâmetros do tipo INTEIRO:**

0 (zero): Modo Parcial (Partial Cut).

1 (um): Modo Total (Full Cut).

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = AcionaGuilhotina(1)
```

#### **// Exemplo em Delphi**

```
iRetorno := AcionaGuilhotina(1);
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-2 (menos dois): Parâmetro Inválido.

**Observação:** O modo de acionamento parcial não deve ser usado nos Blocos Impressores que possuem **PRESENTER**.

## Função AjustaLarguraPapel()

Seleciona a largura da bitola do papel da impressora.

### Parâmetro:

iWidth: Variável do tipo INTEIRA, indicando a bitola do papel em milímetros. Podendo ser: 48, 58, 76, 80 ou 112.

### Exemplo:

' **Exemplo em Visual Basic &ndash; Seleciona bitola para 72 mm.**

```
iRetorno = AjustaLarguraPapel(76)
```

// **Exemplo em Delphi**

```
iRetorno := AjustaLarguraPapel(76);
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-4 (menos quatro): Parâmetro inválido.

**Observação:** A largura *default* utilizada pela DLL é de 48 mm, caso não se configure uma nova largura, a impressão do bitmap será limitada a esse valor.

## Função AutenticaDoc()

Esta função permite autenticar documentos com uma maior facilidade, sem a necessidade do aplicativo enviar os comandos de autenticação para a impressora. Basta passar o texto que deseja autenticar e o tempo de espera, definido em milissegundos.

### Parâmetros:

Texto: STRING que será usada na autenticação.

Tempo: INTEIRO com o tempo de espera.

### Modo de usar:

```
AutenticaDoc( "Texto a ser impresso", 5000 )
```

onde:

5000 = 5 segundos de espera para a inserção do documento

### Exemplo:

' Exemplo em Visual Basic

```
iRetorno = AutenticaDoc("Texto a ser Autenticado",  
5000)
```

// Exemplo em Delphi

```
cTexto := 'Texto a ser Autenticado';  
iRetorno := AutenticaDoc( pchar( cTexto ), 5000);
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um) : Sucesso. A função foi executada sem problemas.

0 (zero) : Time-Out. O documento não foi inserido no tempo determinado.

## Função BematechTX()

Esta função é utilizada na impressão de textos, enviando um conjunto com várias linhas.

### Parâmetro:

Texto: STRING com o texto a ser impresso.

### Exemplo:

#### ' Exemplo em Visual Basic

```
sTexto = "Total bruto: 12.500,00" + chr( 10 )  
sTexto = sTexto + "Total líquido: 9.600,00" + chr( 10 )  
iRetorno = BematechTX( sTexto )
```

#### // Exemplo em Delphi

```
sTexto := 'Total bruto: 12.500,00' + #10;  
sTexto := sTexto + 'Total líquido: 9.600,00' + #10;  
iRetorno := BematechTX( pchar( sTexto ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): Sucesso. A função foi executada sem problemas.

0 (zero): Erro na comunicação.

## Função CaracterGrafico()

Esta função tem por objetivo imprimir o caracter gráfico criado.

### Exemplo:

```
' Exemplo em Visual Basic
```

```
' DESENHO
```

```
'      1 2 3 4 5 6 7 8 9
```

```
' bit 7 = 128 *           *
```

```
' bit 6 = 064 * *         *
```

```
' bit 5 = 032 * * *       *
```

```
' bit 4 = 016 * * * *     *
```

```
' bit 3 = 008 * * * * *   *
```

```
' bit 2 = 004 * * * * * * *
```

```
' bit 1 = 002 * * * * * * * *
```

```
' bit 0 = 001 * * * * * * * * *
```

```
' Comando que habilita o modo grafico com 9 pinos (9 colunas)
```

```
sBuffer = Chr(27) + Chr(94) + Chr(18) + Chr(0)
```

```
iRetorno = ComandoTX(sBuffer, 4)
```

```
' Sequencia de bytes para a montagem do desenho acima
```

```
sBuffer = Chr(255) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(127) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(63) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(31) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(15) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(7) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(3) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(1) + Chr(0) + Chr(0) + Chr(0) + _
```

```
Chr(255) + Chr(0) + Chr(0) + Chr(0)
```

```
' Descarrega o buffer na impressora.
```

```
sBuffer = sBuffer + Chr(13) + Chr(10)
```

' Função CaracterGrafico.

iRetorno = CaracterGrafico(sBuffer, Len(sBuffer))

// Exemplo em Delphi

{

DESENHO

```
      1 2 3 4 5 6 7 8 9
bit 7 = 128 *           *
bit 6 = 064 **         *
bit 5 = 032 ***       *
bit 4 = 016 ****      *
bit 3 = 008 *****  *
bit 2 = 004 ********  *
bit 1 = 002 ********** *
bit 0 = 001 ********* *
```

**begin**

// Comando que habilita o modo grafico com 9 pinos (9 colunas)

cmd := chr( 27 ) + chr( 94 ) + chr( 18 ) + chr( 0 );

comando := ComandoTX( cmd, Length( cmd ) );

// Sequencia de bytes para a montagem do desenho acima

```
cmd := chr(255) + chr(000) + chr(000) + chr(000) +
chr(127) + chr(000) + chr(000) + chr(000) +
chr(063) + chr(000) + chr(000) + chr(000) +
chr(031) + chr(000) + chr(000) + chr(000) +
chr(015) + chr(000) + chr(000) + chr(000) +
chr(007) + chr(000) + chr(000) + chr(000) +
chr(003) + chr(000) + chr(000) + chr(000) +
chr(001) + chr(000) + chr(000) + chr(000) +
chr(255) + chr(000) + chr(000) + chr(000);
```

cmd := cmd + #13 + #10;

// Função CaracterGrafico

```
comando := CaracterGrafico( cmd, length( cmd ) );
```

**end;**

O **retorno** desta função é dado através de um , onde se o retorno for:

1 (um): Sucesso. A função foi executada sem problemas.

0 (zero): Erro ao fechar a porta de comunicação.

## Função ComandoTX()

Esta função é utilizada no envio de comandos para a impressora, como por exemplo: comandos de Autenticação, comando para Acionamento de Gaveta, comandos para Habilitar Tabelas de Caracteres, etc.

### Parâmetros:

Comando: STRING com o comando que deseja executar.

Tamanho do Comando: INTEIRO com o tamanho do comando que será enviado.

Exemplo:

```
' Exemplo em Visual Basic  
' Comando para Acionar a Gaveta de Dinheiro  
sComando = chr( 27 ) + chr( 118 ) + chr( 140 )  
iRetorno = ComandoTX( sComando, Len( sComando )
```

```
// Exemplo em Delphi  
// Comando para Acionar a Gaveta de Dinheiro  
sComando := #27 + #118 + #140;  
iRetorno := ComandoTX( sComando, Length( sComando );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): Sucesso. A função foi executada sem problemas.

0 (zero): Erro na comunicação.

## Função ConfiguraModeloImpressora()

Esta função é utilizada para configurar o modelo da impressora não fiscal em uso.

### **IMPORTANTE**

**Esta função deve ser usada antes da função IniciaPorta.**

### **Parâmetros do tipo INTEIRO:**

- 0: MP-20 TH, MP-2000 CI ou MP-2000 TH
- 1: MP-20 MI, MP-20 CI ou MP-20 S
- 2: Blocos térmicos (com comunicacao serial DTR/DSR)
- 3: Bloco 112 mm
- 4: ThermalKiosk
- 5: MP-4000 TH
- 7: MP-4200 TH

O Default é 0 (zero).

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ConfiguraModeloImpressora(1)
```

#### **// Exemplo em Delphi**

```
iRetorno := ConfiguraModeloImpressora( 1 );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): OK.

-2 (menos dois): Parâmetro Inválido.

## Função ConfiguraTamanhoExtrato()

**Observação:** Somente utilizada para os Blocos Impressores.

Esta função configura a quantidade de linhas impressas no extrato, antes de começar a expulsá-lo (eject). A quantidade de linhas pode variar de 1 a 150 linhas. O Default é 90 linhas.

### **Parâmetro:**

Tamanho: INTEIRO com o tamanho do extrato.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ConfiguraTamanhoExtrato(50)
```

#### **// Exemplo em Delphi**

```
iRetorno := ConfiguraTamanhoExtrato(50);
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-2 (menos dois): Parâmetro Inválido.

### **Importante**

**Esta função não terá efeito, caso a função HabilitaExtratoLongo não for executada.**

## Função ConfiguraTaxaSerial()

Esta função configura o taxa de transmissão para a porta serial. Ela deve ser chamada ANTES da função IniciaPorta.

### Parâmetro:

Taxa: INTEIRO com a taxa em bps (bits por segundo). Esta taxa pode ser 9600 ou 115200.

### Exemplo:

```
' Exemplo em Visual Basic
Private Sub Form_Load()
    ConfiguraTaxaSerial(115200)
End Sub
```

```
// Exemplo em Delphi
procedure TForm1.FormCreate(Sender: TObject);
begin
    ConfiguraTaxaSerial( 115200 );
end;
```

**Observação:** Esta função não possui retorno.

## Função DocumentInserted()

Esta função tem por objetivo verificar a presença de documento antes da autenticação.

### **Exemplo:**

#### ' Exemplo em Visual Basic

```
iRetorno = DocumentInserted()
```

#### // Exemplo em Delphi

```
iRetorno := DocumentInserted();
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): Existe documento inserido para autenticação.

0 (zero): Não existe documento inserido para autenticação.

## Função EsperaImpressao()

Esta função segura a execução do Aplicativo, até que todo o texto enviado seja impresso.

### **Parâmetro:**

Modo: INTEIRO com o modo de espera.

1 (um): Habilita

0 (zero): Desabilita

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = HabilitaEsperaImpressao(1)
```

#### **// Exemplo em Delphi**

```
iRetorno := HabilitaEsperaImpressao(1);
```

Não há retorno nesta função.

### **Importante**

**Esta função só terá efeito, caso a função HabilitaEsperaImpressao tenha sido executada anteriormente.**

## Função FechaPorta()

Esta função tem por objetivo fechar a porta de comunicação, liberando a porta para outras atividades.

### **Exemplo:**

#### ' Exemplo em Visual Basic

```
iRetorno = FechaPorta()
```

#### // Exemplo em Delphi

```
iRetorno := FechaPorta();
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): Sucesso. A função foi executada sem problemas.

0 (zero): Erro ao fechar a porta de comunicação.

**Observação:** A função **FechaPorta()** deve ser usada, somente na saída da Aplicação, ou seja, quando for encerrar o trabalho com a Impressora.

## Função FormataTX()

Esta função tem por objetivo enviar textos para a impressora, com formatações, informadas pelos parâmetros.

### **TipoLetra:**

1 = comprimido

2 = normal

3 = elite

### **Itálico:**

1 = ativa o modo itálico.

0 = desativa o modo itálico

### **Sublinhado:**

1 = ativa o modo sublinhado

0 = desativa o modo sublinhado

### **Expandido:**

1 = ativa o modo expandido

0 = desativa o modo expandido

### **Enfatizado (negrito):**

1 = ativa o modo enfatizado

0 = desativa o modo enfatizado

Modo de usar:

FormataTX( "Texto a ser impresso", TipoLetra, Itálico, Sublinhado, Expandido, Enfatizado)

### **Exemplo:**

' **Exemplo em Visual Basic**

`Private Sub cmdImprimeTexto_Click()`

```
iRetorno = FormataTX("Texto a ser Impresso", 2, 0, 0, 1, 1)  
End Sub
```

### // Exemplo em Delphi

```
procedure TForm1.cmdImprimeTextoClick(Sender: TObject);  
var iRetorno: Integer;  
    cTexto: string;  
begin  
    cTexto := 'Texto a ser Impresso';  
    iRetorno := FormataTX( pchar( cTexto ), 2, 0, 0, 1, 1 );  
end;
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): Sucesso. A função foi executada sem problemas.

0 (zero): Erro de comunicação física.

## Função HabilitaEsperaImpressão()

Esta função habilita ou desabilita o envio do caracter **ETX** (03h), que mantém a impressora ocupada até o término da impressão de todo o texto (string).

### **Parâmetro do tipo INTEIRO:**

0 (zero): Desabilitado.

1 (um): Habilitado.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = HabilitaEsperaImpressao(1)
```

#### **// Exemplo em Delphi**

```
iRetorno := HabilitaEsperaImpressao(1);
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): OK.

-2 (menos dois): Parâmetro Inválido.

## Função HabilitaExtratoLongo()

**Observação:** Somente utilizada para os Blocos Impressores.

Esta função habilita ou desabilita a quantidade de linhas, configurada na função [ConfiguraTamanhoExtrato](#). Caso esta função não for executada, a quantidade de linhas não será a Default (90 linhas), será a quantidade que for enviada.

### **Parâmetro do tipo INTEIRO:**

0 (zero): Desabilitado.

1 (um): Habilitado.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = HabilitaExtratoLongo(1)
```

#### **// Exemplo em Delphi**

```
iRetorno := HabilitaExtratoLongo(1);
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-2 (menos dois): Parâmetro Inválido.

## Função HabilitaPresenterRetratil()

Habilita ou Desabilita a função retrátil do Presenter.

### **Parâmetro:**

iFlag: Variável do tipo INTEIRA, no tamanho de 1 (um) dígito. Onde:

1 = Habilitado

0 = Desabilitado

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = HabilitaPresenterRetratil(1)
```

#### **// Exemplo em Delphi**

```
iRetorno := HabilitaPresenterRetratil(1);
```

### **O retornovalor inteiro**

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função IoControl()

Set / Reset opção do driver.

### Parâmetro:

option: INTEGER com o código da opção, pode ser:

32: utiliza status estendido para retornar informações adicionais (as mesmas retornadas pela interface USB) pela porta serial através da função [Le\\_Status](#).

mode: Boolean, True (habilita opção) e False (desabilita opção)

### Exemplo:

' Exemplo em Visual Basic - Imprime imagem sem redimensionamento ou rotação.

```
IoControl 32, true  
status = Le_Status  
IoControl 32, false
```

```
// Exemplo em Delphi  
// Reduzir dimensão da imagem pela metade e rotacionando-a em 90°
```

```
IoControl 32, true  
status = Le_Status  
IoControl 32, false
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

1 (um): OK.

-1 (menos um): Erro de Execução.

## Função IniciaPorta()

Esta função tem por objetivo abrir a porta de comunicação, onde a impressora está conectada.

### **IMPORTANTE**

**Esta função deve ser usada no início do Aplicativo, logo após a função ConfiguraTaxaSerial**

#### **Parâmetro:**

Porta: STRING com o nome da porta de comunicação.

#### **Exemplo:**

##### **' Exemplo em Visual Basic**

```
Private Sub Form_Load()  
    iRetorno = IniciaPorta("LPT1")  
End Sub
```

##### **// Exemplo em Delphi**

```
procedure TForm1.FormCreate(Sender: TObject);  
var iRetorno: Integer;  
begin  
    iRetorno := IniciaPorta( pchar( 'LPT1' ) );  
end;
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

menor ou igual a 0 (zero): problemas ao abrir a porta de comunicação.

1 (um): porta de comunicação iniciada sem problemas.

## Função Le\_Status()

Esta função tem por objetivo retornar o estado da impressora. Estes estados são definidos como: ON-LINE, OFF-LINE, Desligada ou Sem Papel.

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = Le_Status()
```

#### // Exemplo em Delphi

```
iRetorno := Le_Status();
```

O **retorno** desta função é dado através de um **valor inteiro**, onde cada estado da impressora será definido por um valor. É importante executar esta função em cada estado da impressora e armazenar os valores para que os mesmos sejam comparados futuramente. A função retornará um valor para ON-LINE, um valor para OFF-LINE e, assim, sucessivamente.

### Valores retornados:

- **MP 20 MI**
  - Conexão Serial

Retorno	Descrição
0	Erro de comunicação/ "OFFLINE"
24	Impressora "ONLINE"
32	Impressora sem papel

- Conexão Paralela

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação/"OFFLINE"/sem papel
144	Impressora "ONLINE"

- **MP 2100 TH**

- Conexão Serial

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação/"OFFLINE"
24	Impressora "ONLINE" e/ou Pouco papel
32	Impressora sem papel

- Conexão Paralela

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação
79	Impressora "OFFLINE"
144	Impressora "ONLINE" e/ou pouco papel

- **MP 4000 TH**

- Conexão USB

<b>Retorno</b>	<b>Descrição</b>
24	Impressora "ONLINE" e/ou pouco papel
32	Impressora sem papel
68	Erro de comunicação/"OFFLINE"

- Conexão Paralela

<b>Retorno</b>	<b>Descrição</b>
0	Impressora com pouco papel
24	Impressora ONLINE
40	Erro de comunicação/"OFFLINE"

- Conexão Serial

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação/"OFFLINE"
5	Impressora com pouco papel
24	Impressora "ONLINE"
32	Impressora sem papel

- Conexão Ethernet

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação/"OFFLINE"
24	Impressora "ONLINE" e/ou pouco papel
32	Impressora sem papel

- **MP-4200 TH**

- Conexão Serial

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação
	Impressora com pouco

5	papel
9	Tampa aberta
24	Impressora "ONLINE"
32	Impressora sem papel

- Conexão USB

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação
5	Impressora com pouco papel
9	Tampa aberta
24	Impressora "ONLINE"
32	Impressora sem papel

- Conexão Ethernet

<b>Retorno</b>	<b>Descrição</b>
0	Erro de comunicação
5	Impressora com pouco papel
9	Tampa aberta
24	Impressora "ONLINE"
32	Impressora sem papel

## Função Le\_Status\_Gaveta()

Esta função retorna o estado da gaveta de dinheiro.

### Exemplo:

```
' Exemplo em Visual Basic  
iRetorno = Le Status Gaveta()
```

```
// Exemplo em Delphi  
iRetorno := Le Status Gaveta();
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): Gaveta Aberta.

2 (dois): Gaveta Fechada.

### Observação:

- Nas versões anteriores a MP-2000, os retornos de status da gaveta não estão disponíveis para a comunicação serial.

## Função LeituraStatusEstendido()

Lê 5 bytes de informação sobre o dispositivo.

### Parâmetro:

buffer: array de 5 bytes para receber a informação de status da impressora, conforme descrito abaixo:

#### 1) Printer Status

1	Buffer Status	Buffer Status	0	On / Off line	Error overrun	0	0
---	---------------	---------------	---	---------------	---------------	---	---

Bit 2: 0: data received will be printed  
1: data received will be lost

Bit 3: 0: on-line  
1: off-line

Bit 5&6: 00: buffer empty  
01: buffer less 1/3 full  
10: buffer more 1/3 full  
11: buffer more 3/4 full

#### 2) Off-line Status

1	Error	sensor de fim de papel	presenter sensor	head up	sensor de pouco papel	0	1
---	-------	------------------------	------------------	---------	-----------------------	---	---

Bit 2: 0: Paper OK  
1: Printer near paper end

Bit 3: 0: Head OK  
1: Printer head-up

Bit 4: 0: paper is NOT positioned in the presenter  
1: paper is positioned in the presenter

Bit 5: 0: Paper OK  
1: End of paper

Bit 6: 0: printer with on error  
1: printer on error

### 3) Error Status

1	Recoverable error	Non recoverable error	1	Paper cutter error	cutter	0	0
---	-------------------	-----------------------	---	--------------------	--------	---	---

Bit 2: 0: cutter presente  
1: cutter ausente

Bit 3: 0: cutter OK  
1: cutter on error

Bit 5: 0: No non-recoverable error reported  
1: non-recoverable error reported

Bit 6: 0: No recoverable error reported  
1: recoverable error reported

### 4) Continuous Paper Sensor Status

1	0	Internal paper jam	1	Paper eject error	head temperature	0	1
---	---	--------------------	---	-------------------	------------------	---	---

Bit 2: 0: normal temperature  
1: temperature above normal

Bit 3: 0: printer with no eject error &ndash; paper jam (after presenter

sensor)

1: printer with eject error &ndash; paper jam

Bit 5: 0: No internal paper jam error (before presenter sensor)

1: Internal paper jam error

## 5) Firmware Version

0	Major firmware version	Major firmware version	Major firmware version	Minor firmware version	Minor firmware version	Minor firmware version	Minor firmware version
---	------------------------------	------------------------------	------------------------------	------------------------------	------------------------------	------------------------------	------------------------------

### Exemplo:

```
// Exemplo em Delphi
...
var
  buffer: array[ 0..5 ] of byte ;
begin
  ...
  status := LeituraStatusEstendido( buffer );
  if status = 1 then
    begin
      if ( integer( buffer[ 2 ] ) and 8 ) <> 0
then
      ShowMessage( 'Cutter Error' )
    end;
  ...
end;
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-4 (menos quatro): Parâmetro inválido.

## Função PrinterReset()

Reset a impressora

### **Parâmetro:**

Nenhum

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = PrinterReset
```

#### **// Exemplo em Delphi**

```
iRetorno := PrinterReset;
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

## Função ProgramaPresenterRetratil()

Programa o tempo de espera para retração do papel, caso o mesmo não seja retirado do bocal do Presenter.

### **Parâmetro:**

iTempo: Variável do tipo INTEIRA, entre 0 (zero) e 60 segundos, definindo o tempo de espera. Default = 5 (cinco) segundos.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ProgramaPresenterRetratil(30)
```

#### **// Exemplo em Delphi**

```
iRetorno := ProgramaPresenterRetratil(30);
```

**retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função SeleccionaQualidadeImpressao()

Selecciona a qualidade de impressão, somente disponível para as impressora não fiscais térmicas e para os blocos térmicos.

### Parâmetro:

iTipoQualidade: Variável do tipo INTEIRA, onde:

0: Baixa

1: Média

2: Normal

3: Alta

4: Altíssima

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = SeleccionaQualidadeImpressao(2)
```

#### // Exemplo em Delphi

```
iRetorno := SeleccionaQualidadeImpressao( 2 );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-4 (menos quatro): Parâmetro inválido.

-5 (menos cinco): Modelo de Impressora Inválido.

### Observação:

- largura do papel estiver configurada para 112 mm e o modelo da impressora for o bloco KC4112, somente será possível ajustar esta qualidade para Normal ou Alta. Se alguma outra qualidade for usada a

função retornará -5.

## Função AtualizaFirmware()

Atualiza o firmware do dispositivo.

### **Parâmetro:**

Nome do arquivo contendo o binário do novo firmware.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = AtualizaFirmware("c:\update.bin")
```

#### **// Exemplo em Delphi**

```
cArquivo := 'c:\update.bin';
```

```
iRetorno := AtualizaFirmware( pchar( cArquivo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

-1 (menos um): Erro de execução da função.

0 (zero): Problemas de comunicação.

1 (um): sucesso.

-2 (dois): Arquivo não encontrado.

-3 (três): Arquivo inválido ou erro ao processar arquivo.

## Função VerificaPapelPresenter()

Verifica se há papel posicionado no Presenter.

### **Parâmetro:**

Não há

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = VerificaPapelPresenter()
```

#### **// Exemplo em Delphi**

```
iRetorno := VerificaPapelPresenter();
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

- 1 (menos um): Erro de execução da função.
- 0 (zero): Problemas da verificação do papel no presenter.
- 1 (um): Papel posicionado no presenter.
- 2 (dois): Papel não posicionado no presenter.
- 3 (três): Erro desconhecido.

## Função ConfiguraCodigoBarras()

Esta função configura os códigos de barras, definindo Altura, Largura e Posição dos caracteres.

### **Parâmetros:**

**Altura** - Inteiro entre 1 à 255. (default 162)

**Largura** - Inteiro entre 0 à 2.

Largura = 0 (barras finas)

Largura = 1 (barras médias) - default

Largura = 2 (barras grossas)

**Posição dos Caracteres** - Inteiro entre 0 à 3.

Posição = 0 (não imprime os caracteres do código)

Posição = 1 (imprime os caracteres acima do código)

Posição = 2 (imprime os caracteres abaixo do código) - default

Posição = 3 (imprime os caracteres acima e abaixo do código)

**Fonte** - Inteiro entre 0 à 1.

Fonte = 0 (normal)

Fonte = 1 (condensado)

**Margem** - Inteiro entre 0 à 575 (dots pitch)

Margem = 0 (sem margem) default

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ConfiguraCodigoBarras(100, 1, 3, 0, 5)
```

#### **// Exemplo em Delphi**

```
iRetorno := ConfiguraCodigoBarras( 100, 1, 3, 0, 5 );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasCODABAR()

Esta função faz a impressão do código de barras CODABAR.

### Parâmetro:

Código: STRING do código que será gerado.

O tamanho da string é dada pela tabela abaixo.

Largura das Barras	Quantidade de Caracteres
0	20
1	12
2	8

A Largura das Barras é 1 (default).

### Observações:

- Será acrescentado, automaticamente, o dígito verificador.
- Aceita dígitos entre 0 à 9.
- Aceita as letras A, B, C e D (maiúsculas e minúsculas).
- Aceita os caracteres: "\$", "+", "-", ".", "/" e ":".

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasCODABAR("123-ABC/001")
```

#### // Exemplo em Delphi

```
cCodigo := '123-ABC/001';
```

```
iRetorno := ImprimeCodigoBarrasCODABAR( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um , onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasCODE128

Esta função faz a impressão do código de barras CODE128.

### Parâmetro:

Código: STRING com o código que será gerado.

O tamanho da string é dada pela tabela abaixo:

Largura das Barras	Quantidade de Caracteres
0	42
1	28
2	16

A Largura das Barras é 1 (default).

### Observações:

- Aceita os caracteres da tabela ASCII, na faixa de valores de 001 à 127.

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasCODE128("Bematech")
```

#### // Exemplo em Delphi

```
cCodigo := 'Bematech'
```

```
iRetorno := ImprimeCodigoBarrasCODE128( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 : Parâmetro Inválido.

## Função ImprimeCodigoBarrasCODE39()

Esta função faz a impressão do código de barras CODE39.

### Parâmetro:

Código: STRING com o código que será gerado.

O tamanho da string é dada pela tabela abaixo:

Largura das Barras	Quantidade de Caracteres
0	15
1	9
2	6

A Largura das Barras é 1 (default).

### Observações:

- Será acrescentado, automaticamente, o dígito verificador.
- Aceita dígitos entre 0 à 9.
- Aceita letras de A à Z (maiúsculas e minúsculas).
- Aceita os caracteres: "espaço em branco", "\$", "%", "+", "-", ".", e "/".
- As letras não podem se maiúsculas e minúsculas simultaneamente.

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasCODE39("abc-123")
```

#### // Exemplo em Delphi

```
cCodigo := 'abc-123';  
iRetorno := ImprimeCodigoBarrasCODE39( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um , onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasCODE93()

Esta função faz a impressão do código de barras CODE93.

### Parâmetro:

Código: STRING com o código que será gerado.

O tamanho da string é dada pela tabela abaixo:

Largura das Barras	Quantidade de Caracteres
0	15
1	9
2	6

A Largura das Barras é 1 (default).

### Observações:

- Será acrescentado, automaticamente, o dígito verificador.
- Aceita dígitos entre 0 à 9.
- Aceita as letras A à Z (maiúsculas e minúsculas).
- Aceita os caracteres: "espaço em branco", "\$", "%", "+", "-", "." e "/".

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasCODE93("123-ABC")
```

#### // Exemplo em Delphi

```
cCodigo := '123-ABC';
```

```
iRetorno := ImprimeCodigoBarrasCODE93( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasEAN13()

Esta função faz a impressão do código de barras EAN13.

### **Parâmetro:**

Codigo: STRING com o tamanho de 12 dígitos de 0 à 9.

**Observação:** Será acrescentado, automaticamente, o dígito verificador.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ImprimeCodigoBarrasEAN13("123456789012")
```

#### **// Exemplo em Delphi**

```
cCodigo := '123456789012';  
iRetorno := ImprimeCodigoBarrasEAN13( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasEAN8()

Esta função faz a impressão do código de barras EAN8.

### **Parâmetro:**

Codigo: STRING com o tamanho de 7 dígitos de 0 à 9.

**Observação:** Será acrescentado, automaticamente, o dígito verificador.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ImprimeCodigoBarrasEAN8("1234567")
```

#### **// Exemplo em Delphi**

```
cCodigo := '1234567';
```

```
iRetorno := ImprimeCodigoBarrasEAN8( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasISBN()

Esta função faz a impressão do código de barras ISBN.

### Parâmetro:

Código: STRING com 9 dígitos, composto por dígitos de 0 à 9, "-" (hifen) e "X". O "-" (hifen) e o "X" não são somados. Após o 9º dígito, podem aparecer ainda hifens seguidos por "X" ou algum número com o tamanho de 5 caracteres, não somando o espaço após o "-X" ou após o número. Exemplo: "1-56592-292-X 9000" ou "1-56592-291-1 900000".

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasISBN("1-56592-292-X 9000")
```

#### // Exemplo em Delphi

```
cCodigo := '1-56592-292-X 9000';  
iRetorno := ImprimeCodigoBarrasISBN( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1: OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasITF()

Esta função faz a impressão do código de barras ITF (Intercalado 2/5).

### Parâmetro:

Código: STRING com o código que será gerado.

O tamanho da string é dada pela tabela abaixo:

Largura das Barras	Quantidade de Caracteres
0	30
1	20
2	14

A Largura da Barras é 1 (default).

**Observações:** Aceita dígitos entre 0 à 9.

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasITF("0123456789012345")
```

#### // Exemplo em Delphi

```
cCodigo := '0123456789012345';  
iRetorno := ImprimeCodigoBarrasITF( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasMSI()

Esta função faz a impressão do código de barras MSI.

### Parâmetro:

Código: STRING com o código que será gerado.

O tamanho da string é dada pela tabela abaixo:

Largura das Barras	Quantidade de Caracteres
0	16
1	10
2	7

A Largura da Barras é 1 (default).

### Observações:

- Será acrescentado, automaticamente, o dígito verificador.
- Aceita dígitos entre 0 à 9.

### Exemplo:

' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasMSI("123")
```

// Exemplo em Delphi

```
cCodigo := '123';
```

```
iRetorno := ImprimeCodigoBarrasMSI( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero) : Erro de Comunicação.

1 (um) : OK.

-1 (menos um) : Erro de Execução.

-2 (menos dois) : Parâmetro Inválido.

## Função ImprimeCodigoBarrasPDF417()

Esta função faz a impressão do código de barras PDF417.

### Parâmetros:

Nível de Correção de Erros - Inteiro entre 0 à 8.

- Quanto mais alto o nível, melhor a leitura do código, maior a impressão e menor o número de informações que poderão ser impressas.

Altura - Inteiro entre 1 à 8.

Altura do caracter do código (pitch). 1 pitch = altura de 0,125 mm.

Largura - Inteiro entre 1 à 4.

Largura do caracter do código (pitch). 1 pitch = altura de 0,125 mm.

Número de Colunas Impressa na Linha - Inteiro entre 0 à 30.

- "0" (zero) utiliza o máximo de colunas que o mecanismo permite para a largura informada (pitch). Caso não caiba na linha a impressora ajusta, automaticamente, para o máximo de colunas permitido na linha.

Código - String do código que deseja gerar.

### Exemplo:

```
' Exemplo em Visual Basic  
iRetorno = ImprimeCodigoBarrasPDF417(4, 3, 2, 0,  
"Bematech. Sempre presente nas melhores soluções  
!!!")
```

```
// Exemplo em Delphi
cTexto := 'Bematech. Sempre presente nas melhores
soluções !!!';
iRetorno := ImprimeCodigoBarrasPDF417( 4, 3, 2, 0,
pchar( cTexto ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

- 0 (zero): Erro de Comunicação.
- 1 (um): OK.
- 1 (menos um): Erro de Execução.
- 2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasPLESSEY()

Esta função faz a impressão do código de barras PLESSEY.

### Parâmetro:

Código: STRING com o código que será gerado.

O tamanho da string é dada pela tabela abaixo:

Largura das Barras	Quantidade de Caracteres
0	13
1	7
2	4

A Largura das Barras é 1 (default).

### Observações:

- Será acrescentado, automaticamente, o dígito verificador.
- Aceita dígitos entre 0 à 9.
- Aceita as letras A à Z (maiúsculas e minúsculas).
- As letras não podem ser maiúsculas e minúsculas simultaneamente.

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeCodigoBarrasPLESSEY("123-ABC")
```

#### // Exemplo em Delphi

```
cCodigo := '123-ABC';
```

```
iRetorno := ImprimeCodigoBarrasPLESSEY( pchar( cCodigo ) );
```

O desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasUPCA()

Esta função faz a impressão do código de barras UPCA.

### **Parâmetro:**

Codigo: STRING com o tamanho de 11 dígitos de 0 à 9.

**Observação:** Será acrescentado, automaticamente, o dígito verificador.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ImprimeCodigoBarrasUPCA("12345678901")
```

#### **// Exemplo em Delphi**

```
cCodigo := '12345678901'
```

```
iRetorno := ImprimeCodigoBarrasUPCA( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoBarrasUPCE()

Esta função faz a impressão do código de barras UPCE.

### **Parâmetro:**

Código: STRING com o tamanho de 6 dígitos de 0 à 9.

**Observação:** Será acrescentado, automaticamente, o dígito verificador.

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = ImprimeCodigoBarrasUPCE("123456")
```

#### **// Exemplo em Delphi**

```
iRetorno := ImprimeCodigoBarrasUPCE( pchar( cCodigo ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0: Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função ImprimeCodigoQRCODE()

Esta função faz a impressão do código de barras QRCODE.

### **Parâmetros:**

Nível de Restauração: variável INT para definir o nível de restauração do código, onde:

- 0 - Aproximadamente 7% de restauração.
- 1 - Aproximadamente 15% de restauração.
- 2 - Aproximadamente 25% de restauração.
- 3 - Aproximadamente 30% de restauração.

Tamanho do módulo: variável INT para definir o tamanho do módulo do código, compreendido entre 1 e 127.

Tipo: variável INT com o tipo do código, compreendido entre 0 e 1, onde 0 é para a impressão em tamanho normal e 1 para tamanho reduzido.

Versão do código: variável INT para definir a versão do código QRCODE, compreendido entre 1 e 40.

Formato do Código: variável INT para definir quais dados serão impressões no código QRCODE, onde:

- Para o tamanho normal.

- 0 &endash; Somente números até 7.089 caracteres.
- 1 &endash; Alfanumérico com até 4.296 caracteres.
- 2 &endash; Binário (8 bits) com até 2.953 bytes.
- 3 &endash; Kanji com até 1.817 caracteres.

- Para o tamanho reduzido.

- 0 &endash; Somente números até 35 caracteres.

- 1 &endash; Alfanumérico com até 21 caracteres.
- 2 &endash; Binário (8 bits) com até 15 bytes.
- 3 &endash; Kanji com até 9 caracteres.

Código: variável STRING com os dados que serão impressos no código QRCODE.

### **Exemplo:**

#### **Exemplo em Visual Basic**

```
errorCorrectionLevel = 1
moduleSize = 10
codeType = 0
QRCodeVersion = 10
encodingModes = 1
codeQr = "123ABC"
iRetorno = ImprimeCodigoQRCODE(errorCorrectionLevel, moduleSize,
codeType, QRCodeVersion, encodingModes, codeQr)
```

#### **Exemplo em Delphi**

```
errorCorrectionLevel := 1;
moduleSize           := 10;
codeType             := 0;
QRCodeVersion        := 10;
encodingModes        := 1;
codeQr               := '123ABC';
iRetorno := ImprimeCodigoQRCODE(
errorCorrectionLevel, moduleSize, codeType,
QRCodeVersion, encodingModes, pchar( codeQr ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

- 0: Erro de Comunicação.
- 1 (um): OK.
- 1 (menos um): Erro de Execução.

-2 (menos dois): Parâmetro Inválido.

## Função SelectDithering()

Seleciona qual algoritmo de *dithering* (*halftoning*), a ser utilizado na impressão do bitmap.

### Parâmetro:

iType: Variável do tipo INTEIRA, indicando o algoritmo. Podendo ser: 0 (Bayer) ou 1 (Floyd-Steinberg).

### Exemplo:

#### ' Exemplo em Visual Basic &ndash; Seleciona Floyd-Steinberg

```
iRetorno = SelectDithering(1)
```

#### // Exemplo em Delphi

```
iRetorno := SelectDithering(1);
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-4 (menos quatro): Parâmetro inválido.

### Observação:

- O algoritmo *default* utilizado pela DLL é o Bayer que se baseia em padrões de dither ordenado, é ligeiramente mais rápido. Já o algoritmo Floyd-Steinberg, se baseia em um filtro de propagação de erros à vizinhança. Produz uma imagem, ligeiramente, mais nítida (quando na redução), mas pode perder informações de tons acinzentados.

## Função ImprimeBitmap()

Imprime uma imagem bitmap na impressora não fiscal.

### Parâmetro:

sFileName: STRING com o caminho completo do arquivo contendo o bitmap.

iMode: Variável do tipo INTEIRA, podendo ser:

0 (zero): indicando orientação RETRATO.

1(um) indicando orientação PAISAGEM.

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = ImprimeBitmap("C:\IMAGENS\CARRO.BMP", 0)
```

#### // Exemplo em Delphi

```
cArquivo := 'C:\IMAGENS\CARRO.BMP';
```

```
iRetorno := ImprimeBitmap( pchar( cArquivo ), 0 );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Arquivo inexistente.

-3 (menos três): Erro na leitura do arquivo, arquivo inválido.

-4 (menos quatro): Parâmetro inválido.

## Função ImprimeBmpEspecial()

Imprime uma imagem bitmap na impressora não fiscal com atributos especiais de impressão.

### Parâmetro:

sFileName: STRING com o caminho completo do arquivo contendo o bitmap.

xScale: Variável do tipo INTEIRA, indicando o escalonamento da imagem na horizontal em porcentagem. Ex: 100 (%) indica largura atual; 200 (%) indica largura 2 vezes maior; -1 (menos um) indica ajuste da largura na página.

yScale: Variável do tipo INTEIRA, indicando o escalonamento da imagem na vertical em porcentagem. Ex: 100 (%) indica altura atual; 50 (%) indica metade da altura. Ignorado se parâmetro xScale seja &endash;1 (menos um).

iAngle: Variável do tipo INTEIRA, usada para girar o bitmap na impressão. Ex: 0 (°) indica sem rotacionamento da imagem; 45 (°) indica rotacionar a imagem em 45 graus.

### Exemplo:

**' Exemplo em Visual Basic - Imprime imagem sem redimensionamento ou rotação.**

```
iRetorno = ImprimeBitmapEspecial("C:\IMAGENS\CARRO.BMP", 100, 100, 0)
```

**// Exemplo em Delphi**

**// Reduzir dimensão da imagem pela metade e rotacionando-a em 90°**

```
cArquivo := 'C:\IMAGENS\CARRO.BMP';
```

```
iRetorno := ImprimeBitmapEspecial( pchar( cArquivo ), 50, 50, 90 );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um) : OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Arquivo inexistente.

-3 (menos três): Erro na leitura do arquivo, arquivo inválido.

-4 (menos quatro): Parâmetro inválido.

## Função DefineNVBitmap()

Armazena na memória não volátil da impressora até 7 imagens diferentes.

### Parâmetros:

iQtde: INT com o número de imagens a serem gravadas.

cImagem: Array de STRING com caminho+nome do arquivo, de acordo com a quantidade definida

### Exemplo:

#### ' Exemplo em Visual Basic

```
cNomeBitMap(0) = "C:\logo.bmp"  
iRetorno = DefineNVBitmap(1, cNomeBitMap(0))
```

#### // Exemplo em Delphi

```
cNomeBitMap[ 0 ] := 'C:\logo.bmp';  
iRetorno := DefineNVBitmap( 1, cNomeBitMap[ 0 ] );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Arquivo inexistente.

-3 (menos três): Erro na leitura do arquivo, arquivo inválido.

-4 (menos quatro): Parâmetro inválido.

## Função PrintNVBitmap()

Imprime a imagem da memória não volátil da impressora.

### **Parâmetros:**

iIndice: INT com o índice da imagem que se deseja imprimir.

iModo: INT com o modo de impressão da imagem, onde:

0 = normal

1 = altura dupla

2 = largura dupla

3 = ambos

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = PrintNVBitmap(1, 0)
```

#### **// Exemplo em Delphi**

```
iRetorno := PrintNVBitmap( 1, 0 );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

## Função Define1NVBitmap()

Armazena na memória não volátil da impressora apenas 1 (uma) imagem.

### Parâmetro:

cImagem: STRING com caminho+nome do arquivo bitmap

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = Define1NVBitmap("C:\logo.bmp")
```

#### // Exemplo em Delphi

```
cNomeBitMap := 'C:\logo.bmp';  
iRetorno := Define1NVBitmap( pchar( cNomeBitMap ) );
```

### Observação:

- Para efetuar a impressão do bitmap armazenado através desta função, basta usar a função [PrintNVBitmap](#), passando o índice 1 (um) diretamente.

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Arquivo inexistente.

-3 (menos três): Erro na leitura do arquivo, arquivo inválido.

-4 (menos quatro): Parâmetro inválido.

## Função DefineDLBitmap()

Armazena na memória volátil da impressora apenas 1 (uma) imagem.

### Parâmetro:

cImagem: STRING com caminho+nome do arquivo bitmap

### Exemplo:

#### ' Exemplo em Visual Basic

```
iRetorno = DefineDLBitmap("C:\logo.bmp")
```

#### // Exemplo em Delphi

```
cNomeBitMap := 'C:\logo.bmp';  
iRetorno := DefineDLBitmap( pchar( cNomeBitMap ) );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

-1 (menos um): Erro de Execução.

-2 (menos dois): Arquivo inexistente.

-3 (menos três): Erro na leitura do arquivo, arquivo inválido.

-4 (menos quatro): Parâmetro inválido.

## Função PrintDLBitmap()

Imprime a imagem da memória volátil da impressora.

### **Parâmetro:**

iModo: INT com o modo de impressão da imagem, onde:

0 = normal

1 = altura dupla

2 = largura dupla

3 = ambos

### **Exemplo:**

#### **' Exemplo em Visual Basic**

```
iRetorno = PrintDLBitmap(0)
```

#### **// Exemplo em Delphi**

```
iRetorno := PrintDLBitmap( 0 );
```

O **retorno** desta função é dado através de um **valor inteiro**, onde se o retorno for:

0 (zero): Erro de Comunicação.

1 (um): OK.

## Leitura de Status da Impressora através de API do Windows

### GetPrinterData.

Um novo *Port Monitor* foi desenvolvido para nossas impressoras não fiscais e bl permitindo um maior controle do dispositivo às aplicações que não utilizam a dll subsistema de Spooler do Windows.

### Prótipo da API:

```
DWORD GetPrinterData(  
    HANDLE hPrinter,    // handle para a impressora  
    LPTSTR pValueName,  // string que identifica o dado a ser recebido  
    LPDWORD pType,      // não utilizado  
    LPBYTE pData,       // buffer para receber a informação do status  
    DWORD nSize,        // tamanho do buffer  
    LPDWORD pcbNeeded   // tamanho necessário em bytes  
);
```

### Parâmetros desta API:

**hPrinter:** Handle para a impressora. Use a função `OpenPrinter` para obter esse h

**pValueName:** Deve ser 'PRINTERSTATUS'.

**pType:** Deve ser NULL.

**pData:** Referência para inteiro que recebe a informação.

**nSize:** Tamanho do buffer acima.

**pcbNeeded:** Tamanho necessário.

**Observação:** Se a função executar com sucesso, o status será retornado na variável interpretado com um OU lógico dos seguintes valores:

OffLine/Desconectada/Desligada	0x00000000 (USB/Paralela/Serial)
On-line	0x00000001 (USB/Paralela/Serial)

Fim de papel	0x00000002 (USB/Paralela/Serial)
Pouco papel	0x00000004 (USB/Paralela/Serial)
Cabeça levantada	0x00000008 (USB/Paralela/Serial)
Buffer da impressora cheio	0x00000010 (USB)
Erro na guilhotina	0x00000020 (USB)
Temperatura alta da cabeça	0x00000040 (USB)
Papel Enroscado	0x00000080 (USB)
Papel no presenter*/Sensor da Gaveta**	0x00000100 (USB/Paralela/Serial)

\*Para blocos

\*\*Para mini impressoras

// Exemplo em Delphi

```
procedure TBematechPrinter.GetStatus(Sender: TObject);
```

```
var
```

```
    Printer          : THandle;
    need             : Cardinal;
    ret              : Bool;
    data             : integer;
    pData            : Pointer;
    ret1             : integer;
    PrinterDefaults : TPrinterDefaults;
    Statusspooler   : Tmemo;
```

```
begin
```

```
    with PrinterDefaults do
```

```
    begin
```

```
        DesiredAccess := PRINTER_ACCESS_ADMINISTER;
```

```
        pDatatype := nil;
```

```
        pDevMode := nil;
```

```
    end;
```

```
    pData := @data;
```

```

    ret := OpenPrinter(pCHAR('Thermal Printer
80mm'),hPrinter,@PrinterDefaults);
    if ret = true then
    begin
        ret1 :=
GetPrinterData(hPrinter, 'PRINTERSTATUS',nil,pData,size
    if ret1 = ERROR_SUCCESS then
        if ( data = 0 ) then
            statusspooler.Lines.Add('Off/Off-line/Disc
        else
            begin
                if (data and 1) <> 0 then
                    statusspooler.Lines.Add('On-Line');
                if (data and 2) <> 0 then
                    statusspooler.Lines.Add('End of pap
                if ( data and 4) <> 0 then
                    statusspooler.Lines.Add('Low paper'
                if ( data and 8 ) <> 0 then
                    statusspooler.Lines.Add('Head up');
                if ( data and 16 ) <> 0 then
                    statusspooler.Lines.Add('Buffer is
                if ( data and 32 ) <> 0 then
                    statusspooler.Lines.Add('Cutter err
                if ( data and 64 ) <> 0 then
                    statusspooler.Lines.Add('Head high
                if ( data and 128 ) <> 0 then
                    statusspooler.LInes.Add('Paper Jam'
                if ( data and 256 ) <> 0 then
                    statusspooler.Lines.Add('Paper in t
            end;
        end;
    ClosePrinter(hPrinter);
end;
end;

```

## Sobre

Este arquivo de ajuda foi desenvolvido pela área de parcerias da Bematech (Bematech MAIS Software Partners), com o objetivo de auxiliá-lo no desenvolvimento de seu aplicativo para as impressoras Bematech.

Neste arquivo você encontrará todas as funções da DLL, incluindo declarações e exemplos em Visual Basic e Delphi.

Qualquer dúvida, que por ventura você venha a ter, poderá esclarece-la através de nosso Suporte Técnico, nos canais:



**0800 644 SBSP (7277)**



[suporte.desenvolvedor@bematech.com.br](mailto:suporte.desenvolvedor@bematech.com.br)



<http://partners.bematech.com.br/forum>

- Visite nossa home-page: <http://www.bematech.com.br>
- Acesse nosso [Portal de Desenvolvedores](#).
- <http://twitter.com/partnerbematech> este é o TWITTER do parceiro Bematech. Siga-nos!

## Declarando a MP2032.DLL e suas funções em Visual Basic

```
Public Declare Function IniciaPorta Lib "MP2032.DLL"  
(ByVal Porta As String) As Integer
```

```
Public Declare Function BematechTX Lib "MP2032.DLL"  
(ByVal comando As String) As Integer
```

```
Public Declare Function ComandoTX Lib "MP2032.DLL"  
(ByVal BufTrans As String, ByVal Flag As Integer) As  
Integer
```

```
Public Declare Function CaracterGrafico Lib  
"MP2032.DLL" (ByVal BufTrans As String, ByVal  
TamBufTrans As Integer) As Integer
```

```
Public Declare Function Le_Status Lib "MP2032.DLL" ()  
As Integer
```

```
Public Declare Function AutenticaDoc Lib "MP2032.DLL"  
(ByVal texto As String, ByVal tempo As Integer) As  
Integer
```

```
Public Declare Function DocumentInserted Lib  
"MP2032.DLL" () As Integer
```

```
Public Declare Function FechaPorta Lib "MP2032.DLL"  
( ) As Integer
```

```
Public Declare Function Le_Status_Gaveta Lib  
"MP2032.DLL" ( ) As Integer
```

```
Public Declare Function ConfiguraTamanhoExtrato Lib  
"MP2032.DLL" (ByVal NumeroLinhas As Integer) As  
Integer
```

```
Public Declare Function HabilitaExtratoLongo Lib
```

"MP2032.DLL" (ByVal Flag As Integer) As Integer

Public Declare Function HabilitaEsperaImpressao Lib  
"MP2032.DLL" (ByVal Flag As Integer) As Integer

Public Declare Function EsperaImpressao Lib  
"MP2032.DLL" () As Integer

Public Declare Function ConfiguraModeloImpressora Lib  
"MP2032.DLL" (ByVal ModeloImpressora As Integer) As  
Integer

Public Declare Function AcionaGuilhotina Lib  
"MP2032.DLL" (ByVal Modo As Integer) As Integer

Public Declare Function FormataTX Lib "MP2032.DLL"  
(ByVal BufTrans As String, ByVal TpoLtra As Integer,  
ByVal Italic As Integer, ByVal Sublin As Integer,  
ByVal Expand As Integer, ByVal Enfat As Integer) As  
Integer

Public Declare Function HabilitaPresenterRetratil Lib  
"MP2032.DLL" (ByVal iFlag As Integer) As Integer

Public Declare Function ProgramaPresenterRetratil Lib  
"MP2032.DLL" (ByVal iTempo As Integer) As Integer

Public Declare Function VerificaPapelPresenter Lib  
"MP2032.DLL" () As Integer

### ' **Função para Configuração dos Códigos de Barras**

Public Declare Function ConfiguraCodigoBarras Lib  
"MP2032.DLL" (ByVal Altura As Integer, ByVal Largura  
As Integer, ByVal PosicaoCaracteres As Integer, ByVal  
Fonte As Integer, ByVal Margem As Integer) As Integer

## ' Funções para impressão dos códigos de barras

```
Public Declare Function ImprimeCodigoBarrasUPCA Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasUPCE Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasEAN13 Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasEAN8 Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasCODE39 Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasCODE93 Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasCODE128  
Lib "MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasITF Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasCODABAR  
Lib "MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasISBN Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasMSI Lib  
"MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasPLESSEY  
Lib "MP2032.DLL" (ByVal Codigo As String) As Integer
```

```
Public Declare Function ImprimeCodigoBarrasPDF417 Lib  
"MP2032.DLL" (ByVal NivelCorrecaoErros As Integer,  
ByVal Altura As Integer, ByVal Largura As Integer,  
ByVal Colunas As Integer, ByVal Codigo As String) As  
Integer
```

```
Public Declare Function ImprimeCodigoQRCode Lib  
"MP2032.DLL" ( ByVal errorCorrectionLevel As Integer,  
ByVal moduleSize as Integer, ByVal codeType As  
Integer, ByVal QRCodeVersion As Integer, ByVal  
encodingModes As Integer, ByVal codeQr As String) As  
Integer
```

### ' Funções para impressão de BitMap

```
Public Declare Function ImprimeBitmap Lib  
"MP2032.DLL" (ByVal Name As String, ByVal mode As  
Integer) As Integer
```

```
Public Declare Function ImprimeBmpEspecial Lib  
"MP2032.DLL" (ByVal Name As String, ByVal xScale As  
Integer, ByVal yScale As Integer, ByVal angle As  
Integer) As Integer
```

```
Public Declare Function AjustaLarguraPapel Lib  
"MP2032.DLL" (ByVal width As Integer) As Integer
```

```
Public Declare Function SelectDithering Lib  
"MP2032.DLL" (ByVal type As Integer) As Integer
```

```
Public Declare Function PrinterReset Lib "MP2032.DLL"  
( ) As integer
```

```
Public Declare Function LeituraStatusEstendido Lib  
"MP2032.DLL" (ByVal A() As Byte) As Integer
```

```
Public Declare Function IoControl Lib "MP2032.DLL"  
(ByVal flag as Integer, ByVal mode as Boolean) as  
Integer
```

```
Public Declare Function DefineNVBitmap Lib  
"MP2032.DLL" (ByVal count As Integer, ByVal  
filenames() As String) As Integer
```

```
Public Declare Function PrintNVBitmap Lib  
"MP2032.DLL" (ByVal image As Integer, ByVal mode As  
Integer)As Integer
```

```
Public Declare Function Define1NVBitmap Lib  
"MP2032.DLL" (ByVal filename As String) As Integer
```

```
Public Declare Function DefineDLBitmap Lib  
"MP2032.DLL" (ByVal filename As String) As Integer
```

```
Public Declare Function PrintDLBitmap Lib  
"MP2032.DLL" (ByVal mode As Integer) As Integer
```

### ' **Função de Firmware**

```
Public Declare Function AtualizaFirmware Lib  
"MP2032.DLL" ( ByVal fileName as String) as Integer
```

## Declarando a MP2032.DLL e suas funções em Delphi

```
function IniciaPorta( Porta: string ): integer;
stdcall; far; external 'MP2032.DLL';

function FechaPorta: integer; stdcall; far; external
'MP2032.DLL';

function BematechTX( BufTrans: string ): integer;
stdcall; far; external 'MP2032.DLL';

function ComandoTX( BufTrans: string; TamBufTrans:
integer ): integer; stdcall; far; external
'MP2032.DLL';

function CaracterGrafico( BufTrans: string;
TamBufTrans: integer ): integer; stdcall; far;
external 'MP2032.DLL';

function DocumentInserted: integer; stdcall; far;
external 'MP2032.DLL';

function Le_Status: integer; stdcall; far; external
'MP2032.DLL';

function AutenticaDoc( texto: string; tempo: integer
): integer; stdcall; far; external 'MP2032.DLL';

function Le_Status_Gaveta: integer; stdcall; far;
external 'MP2032.DLL';

function ConfiguraTamanhoExtrato( NumeroLinhas:
Integer ): integer; stdcall; far; external
'MP2032.DLL';

function HabilitaExtratoLongo( Flag: Integer ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function HabilitaEsperaImpressao( Flag: Integer ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function EsperaImpressao: integer; stdcall; far;
external 'MP2032.DLL';
```

```
function ConfiguraModeloImpressora( ModeloImpressora:
integer ): integer; stdcall; far; external
'MP2032.DLL';
```

```
function AcionaGuilhotina( Modo: integer ): integer;
stdcall; far; external 'MP2032.DLL';
```

```
function FormataTX (BufTras: string; TpoLtra:
integer; Italic: integer; Sublin: integer; expand:
integer; enfat: integer ): integer; stdcall; far;
external 'MP2032.DLL';
```

```
function HabilitaPresenterRetratil( iFlag: integer ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ProgramaPresenterRetratil( iTempo: integer
): integer; stdcall; far; external 'MP2032.DLL';
```

```
function VerificaPapelPresenter: integer; stdcall;
far; external 'MP2032.DLL';
```

### **// Função para Configuração dos Códigos de Barras**

```
function ConfiguraCodigoBarras( Altura: integer;
Largura: integer; PosicaoCaracteres: integer; Fonte:
integer; Margem: integer ): integer; stdcall; far;
external 'MP2032.DLL';
```

### **// Funções para impressão dos códigos de barras**

```
function ImprimeCodigoBarrasUPCA(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasUPCE(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasEAN13(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasEAN8(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasCODE39(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasCODE93(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasCODE128(Codigo: string
): integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasITF(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasCODABAR(Codigo: string
): integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasISBN(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasMSI(Codigo: string ):
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasPLESSEY(Codigo: string
): integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoBarrasPDF417(
```

```
NivelCorrecaoErros: integer; Altura: integer;  
Largura: integer; Colunas: integer; Codigo: string );  
integer; stdcall; far; external 'MP2032.DLL';
```

```
function ImprimeCodigoQRCODE(errorCorrectionLevel:  
integer; moduleSize: integer; codeType: integer;  
QRCodeVersion: integer; encodingModes: integer;  
codeQr: string): integer; stdcall; far; external  
'MP2032.DLL';
```

### **// Funções para impressão de BitMap**

```
function ImprimeBitmap ( name: string; mode:  
integer): integer; stdcall; far; external  
'MP2032.DLL';
```

```
function ImprimeBmpEspecial ( name: string; xScale:  
integer; yScale: integer; angle: integer): integer;  
stdcall; far; external 'MP2032.DLL';
```

```
function AjustaLarguraPapel ( width: integer):  
integer; stdcall; far; external 'MP2032.DLL';
```

```
function SelectDithering ( mode: integer): integer;  
stdcall; far; external 'MP2032.DLL';
```

```
function PrinterReset : integer; stdcall; far;  
external 'MP2032.DLL';
```

```
function LeituraStatusEstendido ( A: array of byte ):  
integer; stdcall; far; external 'MP2032.DLL';
```

```
function IoControl ( flag: Integer; mode : Boolean ):  
integer; stdcall; far; external 'MP2032.DLL';
```

```
function DefineNVBitmap ( count: integer; filenames:  
array of string ): integer; stdcall; far; external
```

```
'MP2032.DLL';
```

```
function PrintNVBitmap ( image: integer; mode:  
integer ): integer; stdcall; far; external  
'MP2032.DLL';
```

```
function Define1NVBitmap ( filename : string ):  
integer; stdcall; far; external 'MP2032.DLL';
```

```
function DefineDLBitmap ( filename: string ):  
integer; stdcall; far; external 'MP2032.DLL';
```

```
function PrintDLBitmap ( mode: integer ): integer;  
stdcall; far; external 'MP2032.DLL';
```

```
// Função de Firmware
```

```
function AtualizaFirmware ( fileName: string):  
integer; stdcall; far; external 'MP2032.DLL';
```