Easybar32/DLL (v2.0) for Windows 95

Distribution Files

When developing applications using Easybar/DLL, you need the following files:

EZBAR32.DLL
EZBAR32.LIB
EASYBAR.H

To distribute an application using Easybar/DLL, simply include the following file with your application:

EZBAR32.DLL

Examples

The following example shows the simplest way to use Easybar/DLL. To see a more complex example, please refer to the source code of the Easybar/DLL Demonstration program.

```
HBARCODE      hBarcode;
/* ...        */
hBacode = BarCreate(BCT_3OF9, 0);
BarSetData(hBarcode, "12345678", 8, NULL, 0);
SetRect(&rect, 100, 100, 300, 200);
BarDraw(hBarcode, hdc, NULL, &rect, 0, 0);
BarDestroy(hBarcode);
/* ...        */
```

The following example shows you how to retrieve the position and width of each bar in a barcode; you might use the information to draw your own barcode for example:

```
HBARCODE      hBarcode;
int x, cx;
/* ...        */
hBacode = BarCreate(BCT_3OF9, 0);
BarSetData(hBarcode, "12345678", 8, NULL, 0);
BarEnumBarStart(hBarcode, 2); /* 2 = scale factor */
while (BarEnumBarNext(hBarcode, &x, &cx, &ulBarStyle))
{
      /* this bar starts at x and is of width cx      */
      /* and its style is indicated by ulBarStyle:    */
      /*    - if its a long bar (EAN, UPC, POSTNET)    */
      /*     - part of main barcode/add-on barcode     */
      /* you may write code to draw the bar here       */
      /* using PatBlt for example                      */
}
BarDestroy(hBarcode);
```

Functions

Basic Functions

BarCreate
BarDestroy
BarSetData
BarSetAddOnData
BarDraw

Information Functions

EnumBarcodeTypes
BarGetAddOnData
BarGetData
BarGetAddOnRect
BarGetMainRect
BarGetRect
GetBarcodeDefDrawFlags (1.21)
GetBarcodeDrawFlagsMask (1.21)
BarGetDrawFlagsMask (1.21)
BarGetDefDrawFlags (1.21)
BarEnumBarStart (1.22)
BarEnumBarNext (1.22)

Fine-Tuning Functions

BarGetBarExt
BarGetClearArea
BarGetInterCharExt
BarGetModuleCount
BarGetNumBarExts
BarGetNumSpaceExts
BarGetSpaceExt
BarSetBarExt
BarSetClearArea
BarSetInterCharExt
BarSetSpaceExt

EnumBarcodeTypes

int EnumBarcodeTypes(iPos, lpiType, lplpName)
int   iPos;
LPINT   lpiType;
LPSTR FAR   *lplpName;

This function allows to enumerate the supported barcode types.

Parameter	Description
iPos	Specifies the current enumeration position. When starting enumeration, use 0 for this parameter; in subsequent calls, use the return value of the previous call until 0 is returned (end of enumeration).
lpiType	Points to an integer that gets filled with the barcode type.
lplpName	Points to a string pointer that gets filled with the address of the barcode symbolic name string.

Returns
The return value indicates the current enumeration position. it is 0 when end of enumeration is reached.

See Also
Supported Barcode Types, BarCreate

BarCreate

HBARCODE BarCreate(iBarcodeType, uStyle)
int iBarcodeType;   /* barcode type */
UINT uStyle;   /* barcode style */

The **BarCreate** function creates a barcode object for the specified barcode type and style.

Parameter	Description
iBarcodeType	Specifies the barcode type to create. Can be one of the following values:

Value	Barcode Name
BCT_2OF5	2 of 5
BCT_INTERLEAVED2OF5	Interleaved 2 of 5
BCT_3OF9	Code 39
BCT_CODE93	Code 93
BCT_CODABAR	Codabar
BCT_EAN13	EAN-13
BCT_EAN13_2	EAN-13 + 2
BCT_EAN13_5	EAN-13 + 5
BCT_EAN8	EAN-8
BCT_EAN8_2	EAN-8 + 2
BCT_EAN8_5	EAN-8 + 5
BCT_UPCA	UPC-A
BCT_UPCA_2	UPC-A + 2
BCT_UPCA_5	UPC-A + 5
BCT_UPCE	UPC-E
BCT_UPCE_2	UPC-E + 2
BCT_UPCE_5	UPC-E + 5
BCT_EAN128A	EAN-128 A
BCT_EAN128B	EAN-128 B
BCT_EAN128C	EAN-128 C
BCT_POSTNET (1.20)	POSTNET

uStyle	Specifies the barcode style. Must be 0 currently.

Returns
The function returns the barcode handle when successful; 0 otherwise

Comments
The barcode created must be destroyed using the BarDestroy function.

See Also
Supported Barcode Types, BarDestroy

BarDestroy

void BarDestroy(hBarcode)
HBARCODE   hBarcode;   /* barcode handle */

The BarDestroy function destroys a barcode object that was created by the BarCreate function.

Parameter	Description
hBarcode	Specifies the handle of the barcode object to destroy.

See Also
BarCreate

BarSetData

BOOL BarSetData(hBarcode, lpData, cbData, lpMsg, cbMsg)
HBARCODE   hBarcode;   /* handle of barcode   */
LPSTR   lpData;   /* address of data   */
int   cbData;   /* number of bytes in data   */
LPSTR   lpMsg;   /* address of message   */
int   cbMsg;   /* number of bytes in message */

This function sets/changes the data and the display message of a barcode.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpData	Address of the data (excluding check character if any) to set to the barcode.
cbData	Count of bytes in lpData.
lpMsg	Address of the message to be displayed; may be NULL.
cbMsg	Count of bytes in lpMsg.

Returns
The return value is TRUE when the data is valid, FALSE otherwise.

Comments
The data of a barcode is used to generate the barcode graphic pattern. By default, it is also the text displayed with the barcode. If you wish to display a different text, use the lpMsg parameter.
When the barcode has an add-on part, lpData is intended only for the main part (left side); use BarSetAddOnData function for the add-on.

See Also
BarSetAddOnData

BarSetAddOnData

BOOL BarSetAddOnData(hBarcode, lpData, cbData, lpMsg, cbMsg)
HBARCODE   hBarcode;
LPSTR   lpData;
int   cbData;
LPSTR   lpMsg;
int   cbMsg;

This function sets/changes the data and the display message of the add-on part of a barcode.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpData	Address of the data to set to the add-on part of the barcode.
cbData	Count of bytes in lpData.
lpMsg	Address of the message to be displayed with the add-on; may be NULL.
cbMsg	Count of bytes in lpMsg.

Returns
The return value is TRUE when the data is valid, FALSE otherwise.

Comments
The data of a barcode is used to generate the barcode graphic pattern. By default, it is also the text displayed with the barcode. If you wish to display a different text, use the lpMsg parameter.

See Also
BarSetData

BarGetData

BOOL BarGetData(hBarcode, lpszData, cbMaxData, lpszMsg, cbMaxMsg)
HBARCODE   hBarcode;
LPSTR   lpszData;
int   cbMaxData;
LPSTR   lpszMsg;
int   cbMaxMsg;

This function retrieves the data and the display message of a barcode.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpData	Address of buffer to receive the barcode data.
cbMaxData	Number of bytes available in lpData.
lpMsg	Address of buffer to reveive the display message.
cbMaxMsg	Number of bytes available in lpMsg.

Returns
The return value is TRUE when successful, FALSE otherwise.

See Also
BarSetData

BarGetAddOnData

BOOL BarGetAddOnData(hBarcode, lpszData, cbMaxData, lpszMsg, cbMaxMsg)
HBARCODE   hBarcode;
LPSTR   lpszData;
int   cbMaxData;
LPSTR   lpszMsg;
int   cbMaxMsg;

This function retrieves the data and the display message of the add-on part of a barcode.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpData	Address of buffer to receive the add-on data.
cbMaxData	Number of bytes available in lpData.
lpMsg	Address of buffer to reveive the add-on display message.
cbMaxMsg	Number of bytes available in lpMsg.

Returns
The return value is TRUE when successful, FALSE otherwise.

See Also
BarSetAddOnData

BarDraw

BOOL BarDraw(hBarcode, hdcDraw, hicTarget, lpRect, iOrient, dwFlags)
HBARCODE   hBarcode;
HDC   hdcDraw;
HDC   hicTarget;
LPRECT   lpRect;
int   iOrient;
DWORD   dwFlags;

This function actually renders the barcode on a device context.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
hdcDraw	Specifies the device context handle to which the barcode is to be rendered.
hicTarget	Specifies the information context handle of the target device; may be NULL. This parameter is used to force WYSIWYG drawing on hdcDraw.
lpRect	Address of the rectangle to which the barcode is to be drawn. The rectangle is in the current logical coordinates of hdcDraw.
iOrient	Specifies the orientation of the barcode: 0, 90, 180 or 270 degrees (counter-clockwise rotation angle).
dwFlags	Drawing flags; may be a combination (ORd) of one or more of the following values:

Value	Meaning
BDF_LEFT	Drawing left-aligned (default).
BDF_RIGHT	Drawing right-aligned.
BDF_CENTER	Drawing centered.
BDF_TOP	Drawing top-aligned (default).
BDF_BOTTOM	Drawing bottom-aligned.
BDF_VCENTER	Drawing centered vertically.
BDF_UNIBARHEIGHT	Uses one unique bar height (instead of two which is the default for some EAN and UPC barcode types)
BDF_HIDEMAINTEXT	Hides the main text.
BDF_HIDEADDONTEXT	Hides the add-on text.
BDF_ADDONTEXTATTOP	Displays add-on text at the top of drawing.
BDF_ADDONTEXTATBOTTOM	Displays add-on text at the bottom of drawing.
BDF_MAINTEXTATTOP	Displays main text at the top of drawing.
BDF_MAINTEXTATBOTTOM	Displays main text at the bottom of drawing.
BDF_NOUPCSMALLFONT	Displays the first and last characters of the UPC-A barcode text using the normal font instead of a smaller one.
BDF_RETAINASPECTRATIO	Changes the height and width of the barcode proportionally should the barcode size be adjusted.
BDF_CALCSIZEONLY	No actual drawing should take place. The function is called only for calculating the actual size of the drawing (which can then be retrieved through the functions: BarGetRect, BarGetMainRect, BarGetAddOnRect).
BDF_NOPIXELALIGN	By default, the barcode size is adjusted so that the narrowest bar width is an integral number of pixels. When this flag is set, the barcode size will not be adjusted. When the hicTarget parameter is not NULL, this flag is

	automatically set internally.
BDF_NOSTETCHTEXT	By default, the barcode text is displayed stretched to occupy the whole width of the barcode. When this flag is set, the barcode text will not be stretched.
BDF_SHOWCODE39STARTSTOP (1.21)	By default, the Code 39 barcode's display text does not show the start and stop characters ('*'); but if you set this flag, they will be shown.
BDF_WYSIWYGEXCLCOLOR (1.21)	When you are drawing WYSIWYG, the color on the screen also mirrors that of the printer; if you set this flag, the color mirroring will be excluded.

Returns
TRUE when successful, FALSE otherwise.

Comments
When the hicTarget parameter is not NULL, you must make sure that hdcDraw and hicTarget have the same logical coordinates, i.e., any point (x,y) in one should correspond to (x,y) (same value) in the other. In this case, the drawing is WYSIWYG: the barcode will be drawn on hdcDraw the same as it would appear on hicTarget, in terms of size, as well as font and fore/background colors.

The lpRect specifies the rectangle onto which the barcode should be drawn. In most cases, the actual size of the barcode will be smaller than this rectangle. This is because, by default, the narrowest bar width of the barcode is chosen to be a multiple of pixels (pixel alignment), unless the BDF_NOPIXELALIGN flag is set or the hicTarget parameter is provided. In the latter case, the BDF_NOPIXELALIGN flag is automatically set and the size of the barcode reflects closely that on the target device context (most likely the printer) represented by hicTarget.

Usually you should not set the BDF_NOPIXELALIGN flag when outputing to the printer; otherwise the barcode may not be readable, unless your printer resolution is high enough and/or the barcode is big enough to make the bar width distortion negligible.

You can know the minimum width of the barcode by calling the function BarGetModuleCount. In fact, the minimum width in pixels of the barcode is equal to the the number of modules it has. And the other possible widths of the barcode are multiples of this minimum width, unless you set the BDF_NOPIXELALIGN flag or provide the parameter hicTarget.

The BarDraw function uses the following attributes currently selected into the information context hicTarget or the device context hdcDraw if hicTarget is NULL:
- Font, used to draw text
- Text Color, used to draw both the text and the barcode's bars
- Back Color, used to draw the barcode background when the Back Mode is OPAQUE
- Back Mode

See Also
Caveats, BarGetRect, BarGetMainRect, BarGetAddOnRect, BarGetModuleCount

BarGetRect

BOOL BarGetRect(hBarcode, lpRect)
HBARCODE   hBarcode;
LPRECT   lpRect;

This function retrieves the actual bounding rectangle of the entire barcode. Use this function only after the BarDraw function was called successfully.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpRect	Address of rectangle to receive the bounding rectangle.

Returns

The return value is TRUE when successful, FALSE otherwise.

See Also

BarGetMainRect, BarGetAddOnRect, BarDraw

BarGetMainRect

BOOL BarGetMainRect(hBarcode, lpRect)
HBARCODE   hBarcode;
LPRECT   lpRect;

This function retrieves the actual bounding rectangle of the main part of the barcode (cf. add-on) excluding the display text. Use this function only after the BarDraw function was called successfully.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpRect	Address of rectangle to receive the bounding rectangle.

Returns
The return value is TRUE when successful, FALSE otherwise.

See Also
BarGetRect, BarGetAddOnRect, BarDraw

BarGetAddOnRect

BOOL BarGetAddOnRect(hBarcode, lpRect)
HBARCODE   hBarcode;
LPRECT   lpRect;

This function retrieves the actual bounding rectangle of the add-on part of the barcode excluding the display text. Use this function only after the BarDraw function was called successfully.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpRect	Address of rectangle to receive the bounding rectangle.

Returns
The return value is TRUE when successful, FALSE otherwise.

See Also
BarGetRect, BarGetMainRect, BarDraw

BarGetModuleCount

int BarGetModuleCount(hBarcode)
HBARCODE   hBarcode;

This function retrieves the number of modules comprising the barcode. A module is the basic width unit in terms of which all the bar/space widths are expressed: they always correspond to an integral number of modules.

Parameter	Description
hBarcode	Specifies the handle of the barcode.

Returns

The function returns the number of modules comprising the barcode.

Comments

By default, the BarDraw function adjusts the barcode size so that a module is an integral number of pixels, except when the BDF_NOPIXELALIGN flag is set or the hicTarget parameter is provided. So, unless one or both of these two conditions are met, your minimum barcode width in pixels is equal to the number of modules of the barcode.

See Also

BarDraw

BarGetClearArea

int BarGetClearArea(hBarcode, iClearAreaID)
HBARCODE   hBarcode;
int   iClearAreaID;

This function retrieves the width (or height) of one of the barcode clear areas: left margin, right margin, top margin, bottom margin and the gap between the main part and add-on part. The value is in units of the narrowest bar width of the barcode.

Parameter	Description	
hBarcode	Specifies the handle of the barcode.	
iClearAreaID	Specifies the clear area ID. Can be one of the following:	

Value	Meaning
BGCA_LEFT	Left margin.
BGCA_RIGHT	Right margin.
BGCA_TOP	Top margin.
BGCA_BOTTOM	Bottom margin.
BGCA_MIDDLE	Gap between main part and add-on.

Returns
The return value is the width (or height) of the clear area, in units of the narrowest bar width of the barcode.

See Also
BarSetClearArea

BarSetClearArea

int BarSetClearArea(hBarcode, iClearAreaID, iNumNarrowBars)
HBARCODE   hBarcode;
int   iClearAreaID;
int   iNumNarrowBars;

This function modifies the width (or height) of one of the barcode clear areas: left margin, right margin, top margin, bottom margin and the gap between the main part and add-on part.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
iClearAreaID	Specifies the clear area ID. Can be one of the following:

Value	Meaning
BGCA_LEFT	Left margin.
BGCA_RIGHT	Right margin.
BGCA_TOP	Top margin.
BGCA_BOTTOM	Bottom margin.
BGCA_MIDDLE	Gap between main part and add-on.

iNumNarrowBars	Specifies the new width (height) of the clear area in units of the narrowest bar width.

Returns
The return value is the previous width (or height) of the clear area in units of the narrowest bar width.

See Also
BarGetClearArea

BarGetNumBarExts

int BarGetNumBarExts(hBarcode)
HBARCODE   hBarcode;

This function retrieves the number of bar width types of the barcode.

Parameter   Description

hBarcode   Specifies the handle of the barcode.

Returns

The return value is the number of bar width types of the barcode.

See Also

BarGetModuleCount, BarGetBarExt, BarSetBarExt

BarGetBarExt

int BarGetBarExt(HBARCODE hBarcode, int iBarIndex)
HBARCODE   hBarcode;
int   iBarIndex;

This function retrieves the relative bar width corresponding to a bar width type. The bar width is in units of modules, thus a value relative to the other bar or space widths.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
iBarIndex	Specifies the index of the bar width type. The narrowest bar type has the index of 0 and the widest has the biggest index which is the number of bar width types of the barcode minus 1.

Returns
The return value is the bar width in units of modules.

See Also
BarGetModuleCount, BarGetNumBarExts, BarSetBarExt

BarSetBarExt

int BarSetBarExt(hBarcode, iBarIndex, iNumModules)
HBARCODE   hBarcode;
int   iBarIndex;
int   iNumModules;

This function modifies the relative bar width corresponding to a bar width type. The bar width is in units of modules, thus a value relative to the other bar or space widths.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
iBarIndex	Specifies the index of the bar width type. The narrowest bar type has the index of 0 and the widest has the biggest index which is the number of bar width types of the barcode minus 1.
iNumModules	Specifies the bar width value in units of modules.

Returns
The return value is the previous bar width in units of modules.

See Also
BarGetModuleCount, BarGetNumBarExts, BarGetBarExt

BarGetNumSpaceExts

int BarGetNumSpaceExts(hBarcode)
HBARCODE   hBarcode;

This function retrieves the number of space width types of the barcode.

Parameter	Description
hBarcode	Specifies the handle of the barcode.

Returns
The return value is the number of space width types of the barcode.

See Also
BarGetModuleCount, BarGetSpaceExt, BarSetSpaceExt

BarGetSpaceExt

int BarGetSpaceExt(hBarcode, iSpaceIndex)
HBARCODE   hBarcode;
int   iSpaceIndex;

This function retrieves the relative space width corresponding to a space width type. The space width is in units of modules, thus a value relative to the other space or bar widths.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
iSpace	Specifies the index of the space width type. The narrowest space type has the index of 0 and the widest has the biggest index which is the number of space width types of the barcode minus 1.

Returns
The return value is the space width in units of modules.

See Also
BarGetModuleCount, BarGetNumSpaceExts, BarSetSpaceExt

BarSetSpaceExt

int BarSetSpaceExt(hBarcode, iSpaceIndex, iNumModules)
HBARCODE   hBarcode;
int   iSpaceIndex;
int   iNumModules;

This function modifies the relative space width corresponding to a space width type. The space width is in units of modules, thus a value relative to the other space or bar widths.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
iSpace	Specifies the index of the space width type. The narrowest space type has the index of 0 and the widest has the biggest index which is the number of space width types of the barcode minus 1.
iNumModules	Specifies the space width value in units of modules.

Returns
The return value is the previous space width in units of modules.

See Also
BarGetModuleCount, BarGetNumSpaceExts BarGetSpaceExt

BarGetInterCharExt

int BarGetInterCharExt(hBarcode)
HBARCODE   hBarcode;

This function retrieves the width of the inter-character gaps of a non-continuous (discrete) barcode. The width value is in units of modules, thus a value relative to the other space or bar widths.

Parameter	Description
hBarcode	Specifies the handle of the barcode.

Returns

The return value is the inter-character gap width in units of modules.

See Also

BarGetModuleCount, BarSetInterCharExt.

BarSetInterCharExt

int BarSetInterCharExt(HBARCODE hBarcode, int iNumModules)
HBARCODE   hBarcode;
int   iNumModules;

This function modifies the width of the inter-character gaps of a non-continuous (discrete) barcode. The width is in units of modules, thus a value relative to the other space or bar widths.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
iNumModules	Specifies the inter-character gap width in units of modules.

Returns

The return value is the previous inter-character gap width in units of modules.

See Also

BarGetModuleCount, BarGetInterCharExt.

GetBarcodeDefDrawFlags (1.21)

DWORD GetBarcodeDefDrawFlags(int iBarcodeType)
int   hBarcodeType;

This function is similar to BarGetDefDrawFlags. The only difference is that it doesn't take a barcode handle but the barcode type instead.

Parameter	Description
iBarcodeType	Specifies a barcode type. Refer to BarCreate for the list of values.

Returns
The return value is a combination of the default drawing flags (see BarDraw) corresponding to the barcode type.

See Also
BarCreate, BarGetDefDrawFlags, BarDraw.

GetBarcodeDrawFlagsMask (1.21)

DWORD GetBarcodeDrawFlagsMask(int iBarcodeType)
int   iBarcodeType;

This function allows you to know what drawing flags are applicable to the given barcode type. It is the same as BarGetDefDrawFlags except that it takes as parameter directly the barcode type instead of a barcode handle.

Parameter	Description
iBarcodeType	Specifies a barcode type. Refer to BarCreate for the list of values.

Returns
The return value is a combination of drawing flags (OR'd) that are applicable to the barcode.

Comments
The return value may also include the value 0x80000000 (i.e., highest bit on) to indicate that the barcode type has an add-on, though it is not a drawing flag.

See Also
BarGetDefDrawFlags, BarCreate.

BarGetDrawFlagsMask (1.21)

DWORD BarGetDrawFlagsMask(HBARCODE hBarcode)
HBARCODE   hBarcode;

This function allows you to know what drawing flags are applicable to the barcode type that the given barcode belong to. It can be useful in user interface design: when the user has selected a particular barcode type, you should disable (gray them on the UI) the drawing flag options that are not applicable to the barcode type.

Parameter	Description
hBarcode	Specifies the handle of the barcode.

Returns

The return value is a combination of drawing flags (OR'd) that are applicable to the barcode.

Comments

The return value may also include the value 0x80000000 (i.e., highest bit on) to indicate that the barcode type has an add-on, though it is not a drawing flag.

See Also

GetBarcodeDrawFlagsMask, BarDraw.

BarGetDefDrawFlags (1.21)

DWORD BarGetDefDrawFlags(HBARCODE hBarcode)
HBARCODE   hBarcode;

This function returns the default drawing flags for the barcode type that the barcode belongs to. Typically, you may pass on the return value to the BarDraw function.

Parameter	Description
hBarcode	Specifies the handle of the barcode.

Returns
The return value is a combination of the default drawing flags (see BarDraw) corresponding to the barcode type.

See Also
BarCreate, GetBarcodeDefDrawFlags, BarDraw.

BarEnumBarStart (1.22)

void BarEnumBarStart(HBARCODE hBarcode, int iModuleWidth)
HBARCODE   hBarcode;
int   iModuleWidth;

This function initializes the enumeration of bars, so that subsequent calls to BarEnumBarNext can be made.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
iModuleWidth	Specifies the scale factor.

See Also
BarEnumBarNext.

BarEnumBarNext (1.22)

BOOL BarEnumBarNext(HBARCODE hBarcode, LPINT lpx, LPINT lpWidth,
        DWORD FAR *lpdwBarStyle)
HBARCODE   hBarcode;
LPINT   lpx;
LPINT   lpWidth;
DWORD FAR   *lpdwBarStyle;

This function enumerates the bars in a barcode starting from the left-most bar. Each call to BarEnumBarNext will retrieve the specification of the next bar. The first call must be preceded by a call to BarEnumBarStart.

Parameter	Description
hBarcode	Specifies the handle of the barcode.
lpx	Address of an integer that will be filled with the starting position of the bar measureed from the left-most edge of the barcode.
lpWidth	Address of an integer that will be filled with the width of the bar.
lpdwBarStyle	Address of a long integer that will be filled with the style of the bar which may be a combination (ORd) of one or more of the following values:

Value	Meaning
BEBN_ADDONBARS	The current bar belongs to the add-on part of the barcode (significant when the barcode has an add-on).
BEBN_LONGBARS	The current bar is a long bar (significant only for EAN, UPC and POSTNET barcodes).

Returns
The return value is FALSE when the last bar of the barcode has already been reached; otherwise TRUE.

Comment
The values of the width and position of a bar are relative (in units of modules); in particular, they are not related to the rectangle with which you might have called the BarDraw() function. Its your responsibility to map them to physical units when you actually render the bar.

See Also
BarEnumBarStart.

Supported Barcode Types

Basic Types

2 of 5	Interleaved 2 of 5	Code 39
Code 93	Codabar	EAN-8
EAN-13	UPC-A	UPC-E
EAN-128 A	EAN-128 B	EAN-128 C
POSTNET (1.20)		

Types with Add-ons

EAN-8 + 2	EAN-8 + 5
EAN-13 + 2	EAN-13 + 5
UPC-A + 2	UPC-A + 5
UPC-E + 2	UPC-E + 5

2 of 5

2 of 5 is a non-continuous barcode without check character. Its character set is '0' to '9'.

The following default values are used:
- wide/narrow bar ratio = 3
- space width = narrow bar width
- intercharacter gap = narrow bar

Interleaved 2 of 5

371982465728

Interleaved 2 of 5 is a continuous barcode. Its character set is '0' to '9'. The number of characters should be even. The check character is optional and not implemented in Easybar/DLL. You can always integrate it into the data (last digit) should you need it.

The following default value is used:
- wide/narrow bar ratio = 3

Code 39

Code 39 is a non-continuous barcode. Its character set is composed of '0' to '9', 'A' to 'Z', SPACE, '*', '$', '/', '+', '-', '.', '%'. The check character is optional and not implemented in Easybar/DLL; you may always integrate it into the data (last digit) should you need it.

The following default values are used:
- wide/narrow bar ratio = 3
- intercharacter gap = 2 * narrow bar

The character '*' being built in as both the start and stop characters of the barcode (they may be shown or hidden according to the flag in the BarDraw() call), you should not attempt to set them any more as part of the data.

Code 93

CODE 93

Code 93 is a continuous barcode with character set composed of '0' to '9', 'A' to 'Z', '-', '.', SPACE, '*', '$', '/', '+', '%' (plus special characters). It has two check characters (not displayed as part of the display text in Easybar/DLL).

The following default value is used:
- bar width = space width

Codabar

A12345678B

Codabar is a non-continuous barcode without check character. Its character set is composed of '0' to '9', 'A', 'B', 'C', 'D', '$', '-', ':', '/', '.', '+'. 'A' to 'D' are eligible only for start/stop characters; in order for them to be part of the barcode, you have to include them in the barcode data. Though the characters 'T', 'n', '*', 'e' may be used instead of 'A', 'B', 'C', 'D', in Easybar/DLL, you cannot directly use them to set data. You may, of course, display them (as well as any other characters) through the lpMsg parameter in the BarSetData function.

The following default values are used:
- wide/narrow bar ratio = 3
- intercharacter gap = 2 * narrow bar

UPC-A

0 74470 79382 3

UPC-A has a fixed length of 11 characters plus the check character. Its character set is '0' to '9'.

See Also

UPC-A + 2, UPC-A + 5

UPC-E

0 3 7 8 4 9 2 9

UPC-E has a fixed length of 7 characters plus the check character. Its character set is '0' to '9'.

See Also

UPC-E + 2, UPC-E + 5

EAN-13

9 781556 153952

EAN-13 has a fixed length of 12 characters plus the check character. Its character set is '0' to '9'.

See Also

EAN-8

9 8 7 6 5 4 3 0

EAN-8 has a fixed length of 7 characters plus the check character. Its character set is '0' to '9'.

See Also

EAN-8 + 2, EAN-8 + 5

UPC-A + 2

See Also
UPC-A, UPC-A + 5

UPC-A + 5

`0 31267 64825 4` `94687`

See Also
UPC-A, UPC-A + 2

UPC-E + 2

0 3 7 8 4 9 2 9   1 4

See Also
UPC-E, UPC-E + 5

UPC-E + 5

0 3 6 4 8 2 5 4   7 9 4 6 2

See Also
UPC-E, UPC-E + 2

EAN-13 + 2

4 501645 096787   12

See Also
EAN-13, EAN-13 + 5

EAN-13 + 5

```
9  781556 153952      90000
```

See Also
EAN-13, EAN-13 + 2

EAN-8 + 2

9 8 7 6 5 4 3 0   2 1

See Also
EAN-8, EAN-8 + 5

EAN-8 + 5

See Also
EAN-8, EAN-8 + 2

EAN-128 A

EAN-128 A is a continuous barcode. Its character set is ASCII 0 to 95 (plus special characters). It has one check character (not shown as part of the dispaly text in Easybar/DLL).

EAN-128 B

ean-128 b

EAN-128 B is a continuous barcode. Its character set is ASCII 32 to 127 (plus special characters). It has one check character (not shown as part of the display text in Easybar/DLL).

EAN-128 C

128902

EAN-128 C is a continuous barcode. Its character set is '0' to '9' (plus special characters), internally coded every two characters. It has one check character (not shown as part of the display text in Easybar/DLL). The number of single digit characters (excluding the check character) must be even.

POSTNET (1.20)

|.|..|||....|.|.|..|||..|.|.|.|.|.|..|..|||.|...|||.|

POSTNET is the barcode that represents the U.S. zip code. It may have two fixed lengths: 5 or 9, plus the check character. Its character set is '0' to '9'. Though not part of the character set, the dash (-) can be used (optional), when working with a 9-digit ZIP code, to separate the first 5 and the last 4 digits in the BarSetData() function call.

Caveats

When you use a rotation angle other than 0 in the BarDraw function call,   the text of the barcode might not be positioned appropriately on the printer, depending on the printer that you are using. As a workaround, you may elect to hide the text by setting the BDF_HIDEMAINTEXT and BDF_HIDEADDONTEXT bits in the BarDraw function call. And then you can also write your own code to print text, the actual size and position of the barcode being retrievable through the functions BarGetRect, BarGetMainRect, BarGetAddOnRect.

Registration

What you have here is the RELEASE version of EASYBAR/DLL. However, EASYBAR/DLL IS NOT IN THE PUBLIC DOMAIN; IT IS PROTECTED BY COPYRIGHT LAWS AND INTERNATIONAL TREATIES. If you use this software and have not registered yet, please read the following instructions.

This software is free for previously registered users of any version of EASYBAR/DLL or VBX. For new users, the registration fee is US$35 (or CAN$45); with source code, $95 (or CAN$135), shipping by mail included. You may send cheques (U.S. & Canada; elsewhere, you may consider to get traveller's cheques), money order or cash (not for source code order; all major currencies accepted) directly to Bokai Corporation (see address below). All direct payments to Bokai Coporation will cause an invoice to be generated and sent to you. CompuServe members may also register in SWREG #4487; the registration will be confirmed by us sending you an e-mail giving you a registration number.

For more information, please contact:

Bokai Corporation
1221 Dundix Rd., #106
Mississauga, ON   L4Y 3Y9   (Canada)

Fax: (+1) 905 276-7692
Internet: 75333.1235@compuserve.com
CompuServe: 75333,1235

Disclaimer of Warranty

Bokai Corporation makes no warranties, express or implied, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. In no event shall Bokai Corporation be liable for any special, incidental or consequential damages even if Bokai Corporation has been advised of the possibility of the same.

Overview

Easybar/DLL is a DLL for creating and printing barcodes. It allows using various drawing attributes including rotation, text position, font, etc. It supports WYSIWYG drawing. The barcode parameters (bar/space widths, margins, etc.) are fine-tunable. Moreover, it allows you to render your own barcode by retrieving the position and width of each bar. 21 barcode types are supported, of which 13 are basic types and 8 with add-ons.

For more information on this and other products, please contact:

Bokai Corporation
1221 Dundix Rd., #106
Mississauga, ON   L4Y 3Y9   (Canada)

Fax: (+1) 905 276-7692
Internet: 75333.1235@compuserve.com