Easybar/DLL

Distribution Files

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

EASYBAR.DLL
EASYBAR.LIB
EASYBAR.H

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

EASYBAR.DLL

Example

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);
/* ...      */
```

Functions

Basic Functions

BarCreate
BarDestroy
BarSetData
BarSetAddOnData
BarDraw

Information Functions

EnumBarcodeTypes
BarGetAddOnData
BarGetData
BarGetAddOnRect
BarGetMainRect
BarGetRect

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

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 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.

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.

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

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

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, you should not attempt to set them any more as part of the data. Also, by default, you will not see '*'s in the display text; to show them, you have to resort to the lpMsg parameter in the BarSetData function.

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

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. 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

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

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

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

See Also

EAN-13 + 2, EAN-13 + 5

EAN-8

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

See Also

UPC-A, UPC-A + 2

UPC-E + 2

See Also

UPC-E, UPC-E + 5

UPC-E + 5

See Also

UPC-E, UPC-E + 2

EAN-13 + 2

See Also

EAN-13, EAN-13 + 5

EAN-13 + 5

See Also

EAN-13, EAN-13 + 2

EAN-8 + 2

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 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

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.

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.

Overview

Easybar/DLL is an API for creating 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. 20 barcode types are supported, of which 12 are basic types and 10 with add-ons.

For further information, please contact:

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

Fax: 905 276-7692
CSID: 75333,1235