Capabilities and Precision

This demo shows, what you can do with VPE - the Virtual Print Engine.
VPE is a hammer! It solves all your printing problems, cooperates perfectly with your development tools and has a real big advantage in contrast to most available components:
It is easy, fast and robust!

What is the Virtual Print Engine?

You're still watching it. VPE is a completely new way to solve your printing and presentation problems.
It allows you the dynamic generation of absolutely free screen and printer output by calling functions during the runtime of your applications.
The concept behind it makes it as easy, as if you were using a flat DOS screen, whilst offering you the full power of Windows.

When and why should I use VPE?

In all cases of printing and data presentation, like:
- complex dynamic documents and reports
- drawing plans, graphs, diagrams, etc.
- filling-in forms exactly and printer-independent
- high performance printing
- using images and barcodes of all common types

Can I try it?

Of course! This demo contains the complete VPE-SDK, including the manuals as helpfiles, the DLL, ActiveX, VCL, .NET components and three megabytes of demo source codes for the most common programming languages / tools.

What are the key features of VPE?

- Exact and printer independent positioning, all coordinates in cm or inches
- Supports lines, polylines, polygons, ellipses, boxes, frames, text, true-color, hatching, BMP, WMF, EMF, JPEG, PNG, TIFF, GIF, PCX and 38 1D-barcode types plus four 2D-barcode types
- Fast vector graphics, easy to use
- Unlimited number of simultaneously open documents and pages
- Optional free scalable WYSIWYG preview
- The preview can be embedded into your own application's windows (like this one), or handled automatically by VPE (like the 'Welcome' demo)
- Intelligent printer setup
- Toolbar, statusbar, rulers, measuring: everything can be modified
- Powerful dynamic layout functions
- and many more...

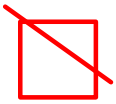
Overview and Tutorial

I. Drawing Primitives (Note: Colors and colored Text might not print with some drivers!)

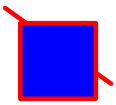
This line was drawn with the command 'Line(1.00, 3.00, 3.00, 3.30)'.
It means, that the starting coordinate (x, y) is 1cm from the right and 3cm from the top
and the ending coordinate (x2, y2) is 3cm from the right and 3.3cm from the top.
The structure of a positioning rectangle (x, y, x2, y2) is used for most of VPE's output functions.

This line looks the same as above, but it was drawn with: Line(1.00, 5.00, -2.00, -0.30)
The negative values mean a relative position to the starting coordinate.
In the example the ending coordinate is 2cm to the right and 0.3cm to the top of
the starting coordinate. This mechanism is usable for ALL drawing functions of VPE.

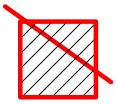
The line thickness and color can be modified as you want. This was done with:
'PenColor = COLOR_LTRED' and 'PenSize = 0.06' [= 0.6mm]
(for the DLL you would enter: 'VpeSetPenColor(hDoc, COLOR_LTRED)' and
'VpeSetPenSize(hDoc, 0.06)')
Note: you need to set the attributes, before inserting an object into the document.
All VPE-objects use their related current attribute settings, at the moment being inserted.



The box was drawn with the command 'Box(1.00, 10.00, 2.00, 11.00)'
As you realize, the current pencolor and pensize is automatically used.
The inner side of the rectangle is transparent, so that drawings beneath aren't overpainted.



You can turn off the transparent mode and specify a background color with:
'BkgMode = VBKG_SOLID' and 'BkgColor = COLOR_BLUE'



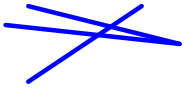
Different hatchstyles are also possible with: 'HatchStyle = HS_BDIAGONAL'

If you set the pensize = 0, the borders of the box aren't drawn.
Background color and hatching can be combined. 'HatchStyle = HS_FDIAGONAL'

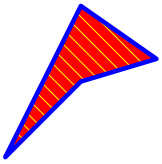
This circle was drawn with: 'Ellipse(1.00, 16.00, 2.00, 17.00)'
All styles for boxes also apply to ellipses.

This ellipse was drawn with: 'Ellipse(1.00, 17.50, 3.00, 18.50)'

If you want to draw a lot of lines, use polylines to save memory and increase performance.
long p = PolyLine(4) [reserve space for 4 points]
AddPolyPoint(p, 1.00, 19.50) AddPolyPoint(p, 3.00, 20.00)
AddPolyPoint(p, 0.70, 19.75) AddPolyPoint(p, 1.00, 20.50)



All penstyles also apply to polylines.
You can start a new line at a completely new position by inserting a -1, -1 pair, e.g.:
p = PolyLine(6) [reserve space for 6 points]
AddPolyPoint(p, 1.00, 22.00) AddPolyPoint(p, 3.00, 22.50)
AddPolyPoint(p, 0.70, 22.25) AddPolyPoint(p, -1, -1)
AddPolyPoint(p, 1.00, 23.00) AddPolyPoint(p, 2.50, 22.00)



All styles for boxes also apply to polygons.
The closing line from the last polygon point to the starting point is drawn automatically.
p = Polygon(4) [reserve space for 4 points]
AddPolygonPoint(p, 2.00, 25.00) AddPolygonPoint(p, 3.00, 25.70)
AddPolygonPoint(p, 2.00, 26.00) AddPolygonPoint(p, 1.00, 27.00)

Hi folks!

Text is drawn as easy as a line. This text was inserted with:
'Write(1.00, 2.00, 3.00, 2.60, "Hi folks!")'

Hi folks!

If you use 'WriteBox' instead of 'Write', a box is automatically drawn around the text.
'WriteBox(1.00, 3.00, 3.00, 3.60, "Hi folks!")'

Hi folks!

Of course, all current styles and settings for the box are used.
(pen size, pen color, background mode, background color, etc.)

Well, the above is very static. You always supply the rectangle (x, y, x2, y2), but if you use a different font size or different text lengths, how can you know about the size of the rectangle, so that the text will fit into it?
You don't. VPE does it for you!
There is a special constant called VFREE, you can use for y2. This means, you specify the starting coordinate (x, y) and the right margin (x2) and VPE computes the bottom coordinate y2 itself. Look at the following example:

Hi folks! I have been in cinema yesterday. The film was very good.

This output was created with the statement:
'WriteBox(1.00, 8.00, 5.00, VFREE, "Hi folks! I have been in cinema yesterday. The film was very good.")'

VPE computed the bottom of the rectangle (y2) itself, using the current font size and text length.
Note, that VPE does also the automatic word break. Now the same command, but with another font size:

Hi folks! I have been in cinema yesterday. The film was very good.

The font (Arial, 14pt) was selected with: 'SelectFont("Arial", 14)'
Note, that the rectangle is much higher than before.

You can specify the text alignment by setting the property 'Alignment' to one of the predefined constants ALIGN_LEFT, ALIGN_RIGHT, ALIGN_CENTER and ALIGN_JUSTIFIED (e.g. 'Alignment = ALIGN_RIGHT'):

Hi folks! I have been in cinema yesterday. The film was very good. (left aligned)

Hi folks! I have been in cinema yesterday. The film was very good. (right aligned)

Hi folks! I have been in cinema yesterday. The film was very good. (centered)

Hi folks! I have been in cinema yesterday. The film was very good. (justified)

The same without the frames, but some attributes:

Hi folks! I have been in cinema yesterday. The film was very good. (left aligned, bold)

Hi folks! I have been in cinema yesterday. The film was very good. (right aligned, italic)

Hi folks! I have been in cinema yesterday. The film was very good. (centered, underlined)

Hi folks! I have been in cinema yesterday. The film was very good. (justified, all attributes)

To set the different attributes, VPE offers 3 boolean properties: TextBold, TextItalic and TextUnderlined
Example: with 'TextBold = TRUE' you activate the bold-attribute, and with 'TextBold = FALSE' you deactivate it.

To modify the text color, just use the property 'TextColor':
This text color was set by the statement: 'TextColor = COLOR_BLUE'.

If text doesn't fit onto the current page, VPE is able to break it automatically over multiple pages. The behaviour of VPE is controlled by the property 'AutoBreakMode'. By default it is activated, and you can see it working in the following text:

Hi folks! I have been in cinema yesterday. The film was very good. Also I had been eating popcorn there, drank a coke and

felt very fine. I don't want to tell you, in what film I've been, because it might lead into trouble, making advertising here for films.

The command for the output of the box above - with its beginning at the bottom of the previous page - was:

```
WriteBox(5.00, 26.00, 15.00, VFREE,
        "Hi folks! I have been in cinema yesterday. The film was very good. "
        "Also I had been eating popcorn there, drank a coke and felt very fine. "
        "I don't want to tell you, in what film I've been, because it might "
        "lead into trouble, making advertising here for films.")
```

As the text reached the bottom of the page, VPE automatically did add a new page and inserted the rest of the text (including the surrounding box) on the new one.This is called AUTOMATIC TEXT BREAK. Note, that VPE is able to do this auto break over multiple pages with just one output command. See the 'Auto Page Break' demo (see in the menu of this application), which generates the whole listing by just ONE single call to 'WriteBox()'.
By the way: you can generate 'manually' page breaks (e.g. add new blank pages to the current end of the document) by calling the method 'PageBreak'.

II.1 Margins

The margins you define in VPE are virtual margins. That means, you can place text and other objects outside of them. But you can access (read / write) the margins with the properties nLeftMargin, nTopMargin, nRightMargin and nBottomMargin. So they can work as placeholders for you, containing the current document definitions. If you need to change the margins later during the development process, and your calls to VPE are considering them, you will have no problems with the new layout.

Nevertheless the margins play an important role for the behaviour of some methods:
- The bottom margin tells VPE, where to start with the Automatic Text Break, but you can change this by modifying the property 'AutoBreakMode' (see documentation).
- The top margin tells VPE, where to start with automatic broken text on successive pages.
- The right margin is important for the method 'Print', which will be explained below.

II.2 More About Text Output

Until now, you've only used the methods 'Write' and 'WriteBox'.
But there are two other methods: 'Print' and 'PrintBox'.
As you notice, they have no right margin (x2 is missing) and no bottom margin (y2 is missing).
Both methods are intended for 'quick usage', you don't need to care about the border coordinates, because VPE does.

'Print' outputs text until the whole text is processed, or the right page margin is reached. Then the remaining text is broken to the next line, starting again at the x-coordinate. This is done until the bottom margin of the page is reached. Depending on the setting for 'AutoBreakMode', text is then broken onto succeeding pages (or not). Note, that - regardless of the settings for the text alignment - the method 'Print' always outputs text left aligned (otherwise the function wouldn't make any sense).

Examples:

Hi folks! I have been in cinema yesterday. The film was very good.

command: 'PrintBox(5.00, 22.50, "Hi folks! I have been in cinema yesterday. The film was very good.")'

Hi folks! I have been in cinema yesterday. The film was very good.

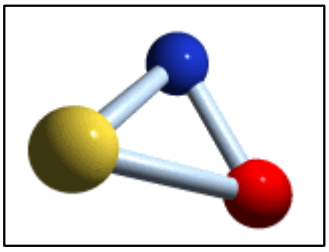
command: 'PrintBox(12.00, 24.00, "Hi folks! I have been in cinema yesterday. The film was very good.")'

The first example did the output in a single line. In the second example you can see, that VPE did break the text into two lines, because it reached the right margin.

VPE imports the following graphics file formats:

- Windows and OS/2 Bitmaps (BMP: 2 / 16 / 256 / True Color)
- Windows Placeable Metafiles (WMF) and Enhanced Metafiles (EMF - 32-bit version only)
- PNG (the new Portable Network Graphics format!)
- JPG (256 / True Color)
- TIFF 5.0 (2 / 16 / 256 / True Color, LZW / PackBits / Fax G3 & G4, Multipage)
- GIF (2 / 16 / 256 Colors, Multipage)
- PCX (2 / 16 / 256 / True Color)
- plus ICO, JNG, KOA, IFF/LBM, MNG, PBM, PCD, PGM, PPM, RAS, TGA, WAP/WBMP/WBM, PSD, CUT, XBM, XPM, DDS, HDR, G3, SGI

Images are easily inserted with the method 'Picture'.



Picture(1.00, 7.50, VFREE, VFREE, "..\Images\logo.bmp")

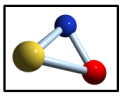
This is all you need to enter. The first 4 parameters are the position rectangle. Note, that x2 and y2 are VFREE, which means, that VPE computes the size of the image by retrieving the DPI resolution from the image and rendering it to its device independent metric coordinate system.

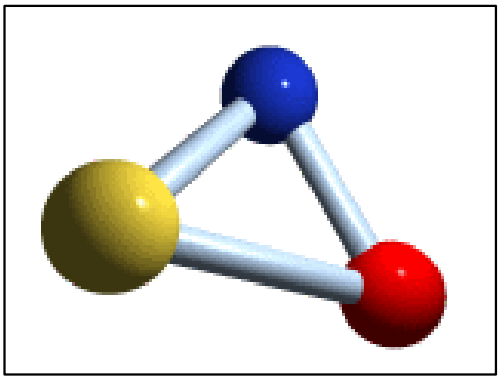
Don't care for the filetypes of the images. VPE automatically determines the right filetype by examining the suffix of the filename (e.g. .BMP / .TIF / .JPG / ...)

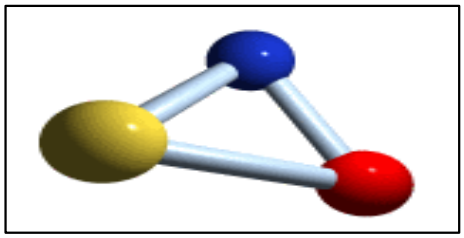
This image is a JPEG file and was created with:

Picture(1.00, 13.00, VFREE, VFREE, "sand.jpg")

Of course you can scale images (undistorted and distorted):







So far you have learned about the different graphical objects offered by VPE, and the basic techniques of using them. Now it is time, to introduce one of the most powerful concepts: the Dynamic Layout.

You have seen, that objects can be placed static (e.g. with absolute coordinates) and that text and images can be rendered. Also you will expirience during a development cycle, that layouts change, because even static placed objects need to be moved or resized and that dependend objects need to be moved also.

VPE offers some constants and properties to gain precise control over this task. First of all you have the four properties nLeftMargin, nRightMargin, nTopMargin and nBottomMargin. They contain the current margin settings, so you can place objects relative to them, which is very useful, if the margins change later during the development process.

For example this output was created with: Print(nLeftMargin, 7.00, "text")

This output was created with: Print(nLeftMargin + 1.00, 7.70, "text")

And this output was created with: WriteBox(10.00, 8.50, nRightMargin, VFREE, "text")

This is easy, and the next thing you are going to learn is as easy. VPE keeps track of the last inserted object, and stores its coordinates in the four properties nLeft, nRight, nTop and nBottom.

If you insert for example a text object with y2=VFREE, so that you don't know its bottom coordinate, but you want to place another object attached to this bottom coordinate, you do the following:

This is text A
This is text B

```
PrintBox(1.00, 13.00, "This is text A")
PrintBox(1.00, nBottom, "This is text B")
```

The example means nothing else, than that the starting y-coordinate of text B is the bottom coordinate of text A.

In the next example the same thing, but with some offsets:

This is text A

This is text B

```
PrintBox(1.00, 16.00, "This is text A")
PrintBox(nLeft + 0.50, nBottom + 0.50, "This is text B")
```

Imagine, what you can do with this powerful concept! You don't need to care about the size or position of objects, when placing others depended on them. Even if your source code is a big project with hundreds of nested functions, which might change dynamically the layout depending on the data they process, you nowhere need to care about anything, just use the n-properties!

VPE can also compute the dimensions of text, RTF and image objects in advance without inserting them into a document. See 'Rendering Objects' in the help file titled 'Programmer's Manual'.

Note: The examples are 100% correct, but the syntax was simplified.

For the OCX and VCL you access methods and properties with the syntax: <object>.method or <object>.property. So if you had named your OCX / VCL 'Doc', you would for example need to enter:

```
Doc.PrintBox(Doc.nLeft + 0.50, Doc.nBottom + 0.50, "This is text B")
```

Since the DLL cannot offer objects or properties, you create handles for each document by calling 'VpeOpenDoc'. After retrieving a handle to a document, each function of the DLL expects this handle as parameter. To read or write the n-properties, the DLL offers the functions 'VpeGet' and 'VpeSet'. So for the DLL the example would be:

```
VpePrintBox(hDoc, VpeGet(hDoc, VLEFT) + 0.50, VpeGet(hDoc, VBOTTOM) + 0.50, "This is text B")
```

Congratulations!

You have learned how to use 90% of VPE's powerful capabilities by just studying a five-page tutorial! If you want to learn more about VPE, we recommend to read the chapter 'Programming Techniques' in the manual / helpfile named 'Programmer's Manual'.

Please note, that all documents in this demo are layouted to fit as well onto DIN A4, as onto US-Letter format.

Now, after so much theory, let's see what we can do with it. The following pages will show some good examples about the precision and capabilities of VPE. There are also some other demos in the main menu, which show other features. A short description of each demo follows now.

Welcome

An introduction. Note, that the images are in true-color, they don't look very nice in 16 or 256 color video mode (this applies also to the "Capabilities + Precision" demo).

Capabilities + Precision

This demo shows text formatting features, drawing features, bitmap handling, form filling and of course printing. Important: the VPE-DLL "docks" its view inside of the window owned by vpedemo.exe! This is very easily done by a few lines of code! The menu-entry "Background" shows how to print without showing a preview and no setup-dialog (default printer is used). The preview window sends the VPE_HELP message to the calling application instead of showing the standard help dialog, so you see the message box "User requested help" generated by vpedemo.exe on the screen.

Speed + Tables

Here you can see how fast VPE builds a report with a size of about 110-130 pages:
A text file with random data is generated (journal.rpt). vpedemo.exe reads the text file line by line, interpreting it and instructing VPE how to build the report.
Since it is random data, the number of pages differs from 110-130 pages. Note, that this demo prints the number of generated pages finally on the FIRST page of the report in the upper left corner. This is done by the virtual processing of the document, where you can move to any page at any time to draw on it. In this case the demo generates all pages and then jumps to the first page to draw the message.

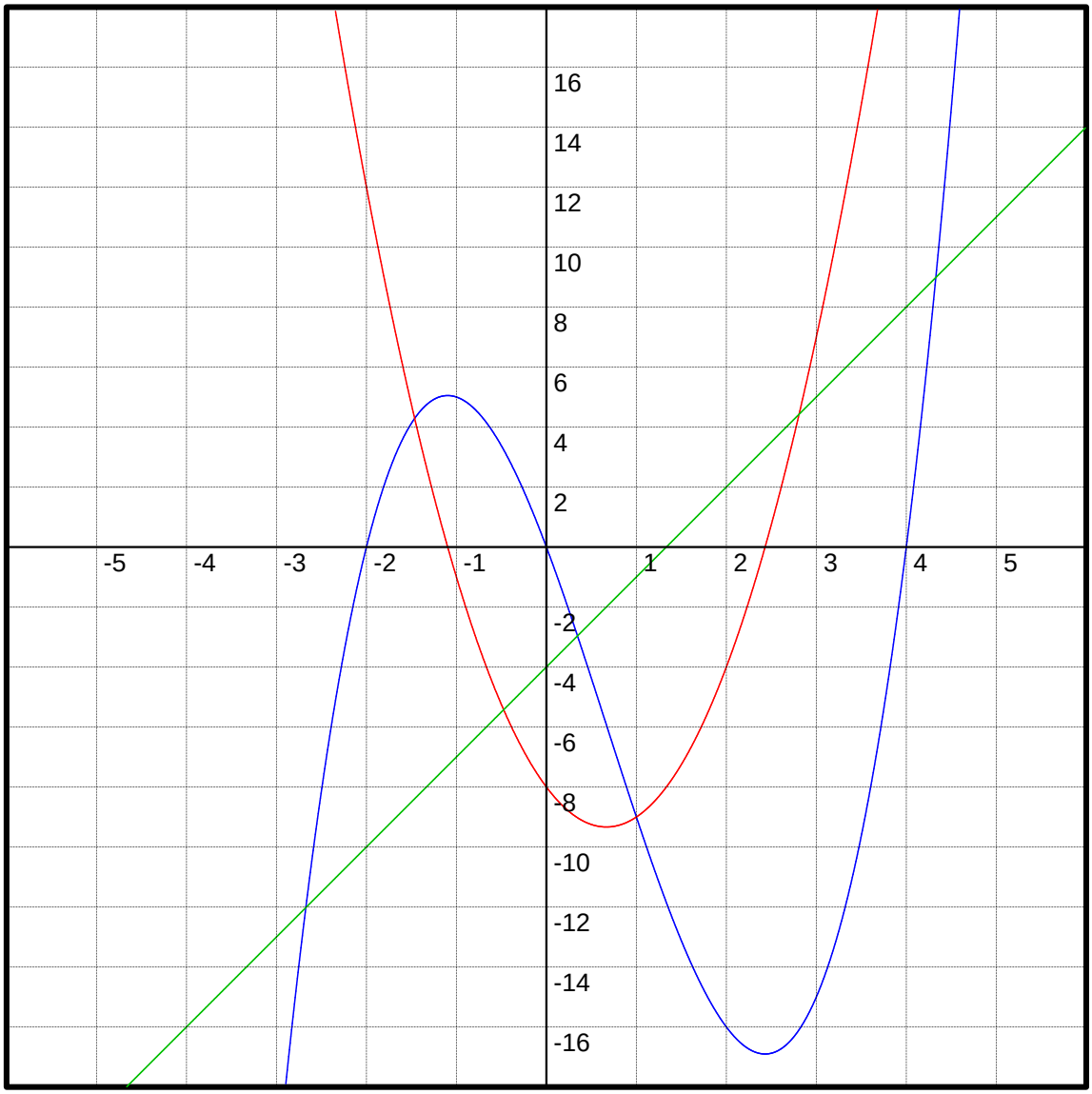
Colors

There you can see a fixed scaled window. Also, the toolbar has only the print and the e-mail button, and the status bar is hidden. The user cannot close the document; it can only be closed through vpedemo.exe by selecting the "Close Preview" menu entry. If you print the page to a color printer, you will get a true-color result.

Report

This is another report, showing various colors and a pie chart on the second page. The source code shows very fine, how easy creating reports is by encapsulating the different parts of the report into functions.

An example of drawing (better to turn the grid off here):



The three graphs together consist of 3021 (number determined during runtime) single lines!

VPE manages this data bulk for you FAST!

A Diagram on a Landscape Oriented Page in the Middle of the Document:

Note: if this page is not printed in landscape orientation, your printer driver is not capable of switching the orientation in the middle of a print job.

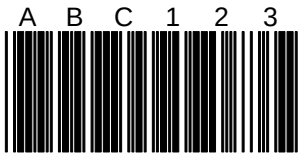
If you are viewing this in Acrobat Reader, zoom in to see the correct barcodes!

The supported barcode-types:

2 of 5:

Interleaved 2 of 5:

Code 39 (text on top):



Code 93 (rotated):

Codabar (rotated):

EAN-8 (rotated):

EAN-8 + 2:

EAN-8 + 5:

EAN-13:

EAN-13 + 2:

EAN-13 + 5:

GS1-128 / EAN-128 A:

GS1-128 / EAN-128 B:

GS1-128 / EAN-128 A/B/C:

Intelligent Mail:

Code-128 A:

Code-128 B:

Code-128 A/B/C:

The supported barcode-types (continued):

UPC-A:

UPC-A + 2:

UPC-A + 5:

UPC-E:

UPC-E + 2:

UPC-E + 5:

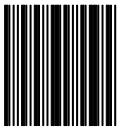
EAN 2:

EAN 5:

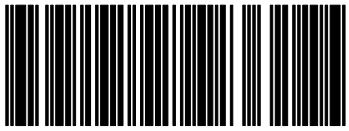
Code 39 Extended:

RM4SCC - Royal Mail:

MSI:



Code 93 Extended:

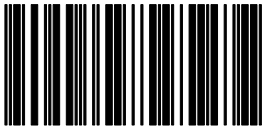


ISBN / ISBN+EAN5:

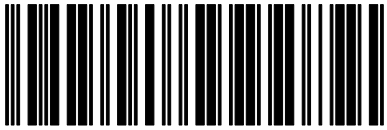
ISBN 3-518-10012-2

ISBN 3-518-10012-2

Identcode (Deutsche Post AG):



Leitcode (Deutsche Post AG):



PZN (Pharma):

Code 11:

Barcodes are as easily generated as images. An example:

Barcode(7.00, 16.60, 12.00, 18.60, BCT_EAN13_5, "9781556153952", "12345")

VPE supports 38 barcode types. Check digits can be auto-generated for all types if wanted. The barcodes can be rotated in 90 degree steps, the font of the descriptive text can be specified and the text can be hidden. Text and add-on text may be drawn on the bottom or top of the barcode and the ratio between the modules can be specified.

How about some gradients?

Virtual Print Engine

The printing and presentation

Solution!

Text and images

degree steps

can be freely

rotated in 90

Three-Dimensional Gradient

Rounded

Rounded

NOTE: Rounded rectangles with gradient fills do not work on Win9x/Me.

Freie Bank

Registernummer: 74894940

Gemeindekennzahl

Branche:

Kredit Antrag

nach §14 BaG oder §55c BaG sowie §1 BaG

Bitte mit Schreibmaschine oder in Blockschrift vollständig und gut lesbar ausfüllen sowie die zutreffenden Kästchen ankreuzen!

Angaben zum Betriebs-inhaber

Bei Personengesellschaften (z. B. OHG) ist für jeden geschäftsführenden Gesellschafter ein eigener Vordruck auszufüllen. Bei juristischen Personen (z. B. GmbH) ist bei Feld Nr. ③ bis ⑩ und Feld Nr. ㉚ und ㉛ der gesetzliche Vertreter anzugeben. Die Angaben für weitere gesetzliche Vertreter zu diesen Nummern sind auf der Rückseite des Vordrucks ☐ oder einem Beiblatt ☐ oder weiteren Vordrucken ☐ gemacht.

① Im Handels-, Genossenschafts- oder Vereinsregister eingetragener Name

Mustermann & Co.
Feinkost Im- und Export

② Ort und Nr. der Eintragung: Düsseldorf

③ Familienname: Schmidt

④ Vornamen: Peter - Thomas

⑤ Geburtsname (nur bei Abweichung vom Familiennamen)

⑥ Geburtsname der Mutter

⑦ Geburtsdatum: 24.7. 1947

⑧ Geburtsort (Ort, Kreis, Land): Krefeld

⑨ Staatsangehörigkeit: deutsch ☐ X andere:

⑩ Anschrift der Wohnung und Telefon-Nr.: Landstr. 93 0 27 84 / 16 45 98

Angaben zum Betrieb

⑪ Zahl der geschäftsführenden Gesellschafter (nur bei Personengesellschaften):
Zahl der gesetzlichen Vertreter (nur bei juristischen Personen): INHABER

⑫ Anschrift der Betriebsstätte und Telefon-Nr.

⑬ Anschrift der Hauptniederlassung und Telefon-Nr.: Parkstr. 17 0 27 84 / 23 54 90

⑭ Anschrift der früheren Betriebsstätte (nur bei Verlegung)

Nach der Änderung, Erweiterung oder Verlegung

⑮ wird neu ausgeübt (z. B. Möbeleinzelhandel)

⑯ wird weiterhin ausgeübt (z. B. Möbelgroßhandel)

⑰ Datum der Änderung, Erweiterung oder Verlegung: 01.06.1994

⑱ Art des umgemeldeten Betriebes: Industrie ☐ Handwerk ☐ Handel ☐ Sonstiges ☐

⑲ Anzahl der voraussichtlich im umgemeldeten Betrieb beschäftigten Arbeitnehmer:

Die Ummeldung wird erstattet für

⑳ einen selbständigen Betrieb ☐ eine Zweigniederlassung ☐ eine unselbständige Zweigstelle ☐

㉑ ein Automatenaufstellungsgewerbe ☐

㉒ ein Reisegewerbe ☐

Wegen

㉓ Änderung der Betriebstätigkeit (z. B. Umwandlung eines Großhandels in einen Einzelhandel) ☐

㉔ Erweiterung der Betriebstätigkeit (z. B. Erweiterung eines Großhandels um einen Einzelhandel) ☐

㉕ Verlegung des Betriebes ☐ X

Falls der Betriebsinhaber für die angemeldete Tätigkeit eine Erlaubnis benötigt, in die Handwerksrolle einzutragen oder Ausländer ist:

㉘ Liegt eine Erlaubnis vor? Nein ☐ Ja, erteilt am/von (Behörde):

㉙ Liegt eine Handwerkskarte vor? Nein ☐ Ja, ausgestellt am/von (Handwerkskammer):

㉚ Liegt eine Aufenthaltserlaubnis vor? Nein ☐ Ja, erteilt am/von (Behörde):

㉛ Die Aufenthaltserlaubnis enthält keine Auflage oder Beschränkung ☐ enthält folgende Auflage oder Beschränkung:

Bitte die **Hinweise** auf der **Rückseite** beachten.
Diese Ummeldung wird gemäß § 15 Abs. 1 GewO bescheinigt.
Verwaltungsgebühr _____ DM.

㉜ 01.06.1994 Datum

㉝ Unterschrift

Dienstsiegel

Ich bin darüber unterrichtet, daß die Übermittlung von Namen, Firma, betrieblicher Anschrift, von Gegenstand des Betriebs, vom Tag der Anzeige bzw. der Erlaubniserteilung an die einschlägigen Berufs- oder Interessenverbände zur Mitgliederwerbung, an Markt- oder Meinungsforschungsinstitute, an Unternehmen (z. B. Versicherungen, Banken) und andere Gewerbetreibende zur Geschäftsanbahnung zulässig ist, sofern ich nicht ausdrücklich widersprochen habe.

☐ Ich widerspreche der Übermittlung dieser Daten
☐ an einschlägigen Berufs- oder Interessenverbände zur Mitgliederwerbung
☐ an Markt- oder Meinungsforschungsinstitute
☐ an Unternehmen (z. B. Versicherungen, Banken) und andere Gewerbetreibende zur Geschäftsanbahnung

VPE is Unlimited Printing Power

The well engineered set of dynamic layout functions and the detailed manual / help will never leave you alone in the dark. It allows the rapid solution of all printing and presentation problems.

The Virtual Print Engine is the answer to the nightmare of printing.

Order Today

For more information please visit us at:

www.IdealSoftware.com

There you will find up-to-date information and the latest version for free download.

Online ordering is also possible, get your license key 24/7 within 5 minutes!

(German and English contents)