KBarcode
The Free Barcode and Labelprinting Solution

Dominik Seichter
domseichter@web.de

March 10, 2006

Contents

1 Introduction **5**
1.1 What is KBarcode? . . . 5
1.2 How much does it cost? . . . 5
1.3 Where can I find KBarcode? . . . 6
1.4 Where can I get support for KBarcode? . . . 6

2 Installation of KBarcode **7**
2.1 Dependencies . . . 7
2.1.1 Required . . . 7
2.1.2 Barcode Backends . . . 7
2.1.3 SQL Database . . . 8
2.2 Installation from Source . . . 9
2.3 Installation from RPM . . . 9
2.4 Starting KBarcode . . . 9
2.5 Configuring KBarcode using the Setup Wizard . . . 10

3 KBarcodes Main Window **12**
3.1 The four components of KBarcode . . . 12
3.2 The Preferences Dialog . . . 13
3.2.1 Print Settings . . . 13
3.2.2 SQL Settings . . . 13
3.2.3 Import . . . 13
3.2.4 On new . . . 14

4 Generating Barcodes **15**

5 The Label Editor **18**
5.1 Creating a new label . . . 19
5.2 Designing a label . . . 19
5.2.1 Moving and resizing items . . . 20
5.3 Properties . . . 20

5.3.1 Properties common to all items . . . 21
5.3.2 Barcode Properties . . . 23
5.3.3 Image Properties . . . 24
5.3.4 Text properties . . . 25
5.4 Label Preview . . . 25

6 Data fields 27
6.1 Simple Data fields . . . 27
6.2 Addresses . . . 28
6.3 User Defined Variables . . . 28
6.4 SQL Queries . . . 28
6.5 JavaScript Functions . . . 29

7 Batchprinting 30
7.1 A Quick start into Batchprinting using Presentations . . . 31
7.2 Printing labels without data . . . 31
7.3 Print articles from KBarcodes SQL database . . . 32
7.4 Import variables and print . . . 34
7.4.1 Enter the data manually . . . 35
7.4.2 Import from CSV file . . . 36
7.4.3 Import from SQL query . . . 37
7.5 Printing address book contacts . . . 37
7.6 The serial number . . . 38
7.7 Selecting an Output Device . . . 39
7.7.1 Printing to a system printer . . . 40
7.7.2 Generating images . . . 40
7.7.3 Printing to a special barcode printer . . . 41
7.8 Batch printing from the command line . . . 41

8 Editing KBarcodes SQL Tables 44

9 Support us! 45

10 Thanks To 46

11 Appendix 47
11.1 SQL Tables . . . 47
11.1.1 Label Definitions: label_def . . . 47
11.1.2 Articles: barcode_basic . . . 48
11.1.3 Customers: customer . . . 48
11.1.4 Customer Articles: customer_text . . . 49
11.2 Supported barcode types . . . 50

11.2.1 Barcode Writer in Pure Postscript 50
11.2.2 GNU Barcode 51
11.2.3 PDF417 Encode 51
11.2.4 TBarcode 51
11.3 GNU General Public License 52

1 Introduction

1.1 What is KBarcode?

KBarcode is a barcode and label printing application for KDE[1]. It can be used to print everything from simple business cards up to complex labels with several barcodes (e.g. article descriptions).

KBarcode comes with an easy to use WYSIWYG label designer, a setup wizard, batch import of data for batch printing labels (directly from the delivery note), thousands of predefined labels, database management tools and translations in many languages. Even printing more than 10.000 labels in one go is no problem for **KBarcode** . Data for printing can be imported from several different data sources, including SQL databases, CSV files[2] and the KDE address book.

Additionally it is a simple barcode generator (similar to the old xbarcode you might know). All major types of barcodes like EAN, UPC, CODE39 and ISBN are supported. Even complex 2D barcodes are supported using third party tools. The generated barcodes can be directly printed or you can export them into images to use them in another application.

1.2 How much does it cost?

KBarcode is free software released under the terms of the GNU GPL license[3], meaning that you are allowed to use it free of charge. You can download it freely from the Internet and install it on as many computers as you want. As **KBarcode** is a free software project done by volunteers, it would be great if you could give something back to the community (feedback, donations, ...).

[1] *http://www.kde.org*
[2] comma separated value
[3] *http://www.gnu.org/copyleft/gpl.html*

1.3 Where can I find KBarcode?

KBarcode is available for download on our webpage *http://www.kbarcode.net* in the downloads section. This webpage includes also lot's of other information regarding **KBarcode** . Additionally **KBarcode** is already included in many Linux distributions.

1.4 Where can I get support for KBarcode?

We offer different ways of support. Please choose one of them to get in touch with us:

- The first source of information is the handbook you are reading right now. It should give you a good start to use **KBarcode** .
- The main area of support we are offering is the mailing list. So, if you are using **KBarcode** in a professional environment, we strongly recommend joining the mailing list. To join the list, please visit: *http://lists.sourceforge.net/lists/listinfo/kbarcode-users* and add your email address. After this you will get a confirmation mail. Simply reply to this email and are member of the **KBarcode** mailing list (low traffic). All messages send to *kbarcode-users@lists.sourceforge.net* will be send to all list members. We will try to answer to all your questions as soon as possible (normally within the next 8 hours).
- We also provide a web forum as a place where you can ask questions and post your feedback. The forum can be found at *http://www.kbarcode.net/forum* . If you have a problem with **KBarcode** it is a good idea to search the forum first, maybe someone else had the same problem already and got a solution for it.
- If you need some quick help or just want to talk about **KBarcode** , you might be interested in our IRC support channel. If you are familiar with IRC please connect to the server *irc.freenode.net* and join the *#kbarcode* channel.

2 Installation of KBarcode

To install **KBarcode** on your system you need root privileges. If you do not have them or do not know, what we are talking about please contact your local system administrator.

Before trying to install **KBarcode** by yourself, please take a look in the package archive of your distribution or operating system. **KBarcode** should be at least available in FreeBSD's ports, in Gentoos portage, in OpenSuSE and in some more distributions. You will get a more recent version of **KBarcode** , of course, if you get it from our webpage and install it by yourself.

2.1 Dependencies

KBarcode requires some additional tools to work. Some of this tools are required for **KBarcode** to work correctly others add only extra features. Please install all of the required dependencies before trying to install **KBarcode** . Other dependencies can be installed later and should be detected by **KBarcode** during start up.

2.1.1 Required

Required dependencies of **KBarcode** are KDE, Qt and Ghostscript. All of these are pretty standard and should be part of any modern Linux distribution or Unix system. If you want to build **KBarcode** from source, please make sure that you have installed the KDE and Qt development packages, too.

2.1.2 Barcode Backends

KBarcode does not generate barcodes by itself but uses several different so called "barcode backends". These backends handle the task of barcode generation. Not all of them are required, if you do not need barcode support you do not have to install any of them.

Barcode Writer in Pure Postscript is the only barcode backend that is included into the **KBarcode** distribution by default. If you install a new version, it will be detected automatically. It can be found online at *http://www.terryburton.co.uk/barcodewriter/* .

GNU Barcode is the oldest supported barcode generator. It is strongly recommended that you install GNU Barcode if you want to use barcodes. You can download the latest version from *http://www.gnu.org/software/barcode/barcode.html* .

PDF417 Encode is a free barcode generator that is able to generate PDF 417 2D barcodes. It can be obtained from *http://sourceforge.net/projects/pdf417encode* .

TBarcode is the only proprietary barcode backend supported. It supports almost every barcode type on earth and a free demo version is available. Please take a look at *http://www.tec-it.com* for more information.

Please see the section "Supported barcode types" in the appendix for a complete list of the different barcode encoding types supported by each backend.

It should be noted that it is no problem to use all the barcode backends together at the same time. It is even a good idea to install all, as each of them supports barcodes other do not support.

2.1.3 SQL Database

KBarcode can fill labels with data from a SQL table before printing. Most databases should work, but only the tested ones are listed below. The SQL connection to the database is established using the Qt SQL interface, that is why you need a Qt driver for your database. Drivers are already available for most databases. Make sure you have the correct driver installed for your database.

A database is not required for **KBarcode** , but it is a good idea to use one if you want to print a large number of labels.

KBarcode was tested to work with the following free databases:

- MySQL
- PostgreSQL
- SQLite

2.2 Installation from Source

Installing from source sounds difficult, but in the end it's quite easy. This example is written for version 2.0.0. If you have another release, it is enough to just replace the version number in all commands. To install from source just download the latest .tar.gz file and extract it to a directory of your choice. In this directory you have to run *./configure && make*. After compilation has finished, you can install **KBarcode** as root using the command *make install*.

Example session:

```
dominik@laptop:~/001> ls
kbarcode-2.0.0.tar.gz
dominik@laptop:~/001> tar xfz kbarcode-2.0.0.tar.gz
dominik@laptop:~/001> cd kbarcode-2.0.0/
dominik@laptop:~/001/kbarcode-2.0.0> ./configure && make
dominik@laptop:~/001/kbarcode-2.0.0> su
Password:
laptop:/home/dominik/001/kbarcode-2.0.0 # make install
laptop:/home/dominik/001/kbarcode-2.0.0 # exit
```

2.3 Installation from RPM

Installation from RPM is easier than installing from source and does not require any development tools to be installed on your box. We try to provide RPMs for as many distributions as possible and we are always happy if users can provide RPMs for their distribution.

A RPM file can be installed as super user with the command *rpm -Uhv kbarcode-2.0.0.i586.rpm*.

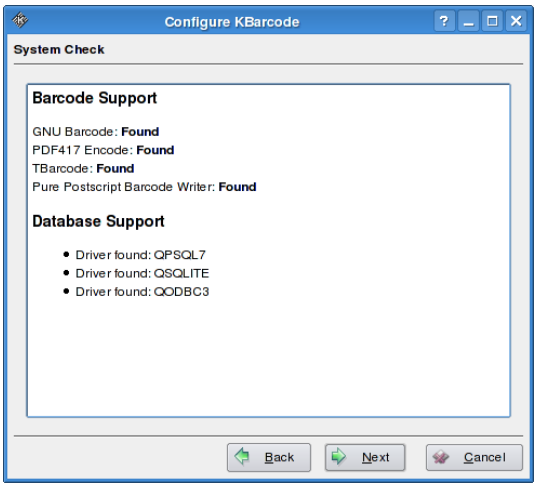
2.4 Starting KBarcode

After you have installed **KBarcode** successfully you can start it now. To start **KBarcode** open a terminal or *konsole* window and type *kbarcode*. If you are running KDE, the other way is to launch **KBarcode** via the KDE menu. It can be found in the Office sub-menu.

2.5 Configuring KBarcode using the Setup Wizard

When starting **KBarcode** for the first time the setup wizard is run automatically and helps you to configure **KBarcode** for your system. You can start the wizard at any time by going to the *Settings* menu in the main window of **KBarcode** and selecting *Start Configuration Wizard...* .

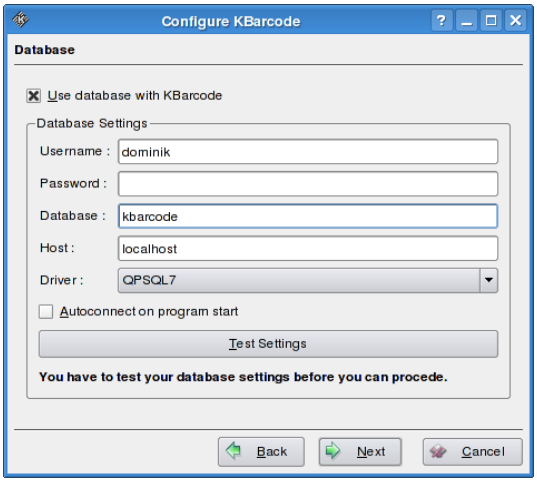
After a short introduction on the first page of the wizard, a system check is performed listing all found barcode backends and database drivers.



The system check has found four barcode backends and three database drivers.

If no database drivers or barcode backends are found during this step and you want database or barcode support, you should install the appropriate tools.

The database setup is the next step you have to perform. If you do not want to use a database with **KBarcode** you can uncheck *Use database with KBarcode* and finish the configuration process. Otherwise please fill in all the required fields to connect to your database (i.e. username, password, host and database driver). The field database is the name of the database used by **KBarcode** . If it does not yet exist it will be created later by **KBarcode** automatically. The database settings have to be tested successfully before you can continue with the wizard.



The database setup dialog, configured for connecting to a PostgreSQL database on the localhost.

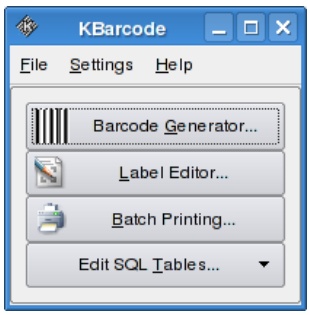
The last step in the configuration of **KBarcode** is (only if you decided to use a database) to create the required SQL tables in the selected database. Along with the creation of the tables, **KBarcode** will import all label definitions[1] in the database, so that the size of all labels can be read from the database which is faster than reading them from a file. The import may take some time. It is possible to import some example data now into the tables, so that some data is available to play with **KBarcode** .

[1] A label definition defines the size of label (i.e. an Avery or Zweckform label) on the page so that **KBarcode** can print it correctly.

3 KBarcodes Main Window

3.1 The four components of KBarcode

KBarcode was installed on your system and configured for it. You should see the main window now. From here on it is possible to configure **KBarcode** and to load the four modules that do the real work. The modules can be launched separately without the main window from the KDE menu or with command line options, too.



The main window right after starting **KBarcode** . The *Edit SQL Tables...* button is disabled if you did not configure a SQL database.

KBarcode has four different modules. All of them will be introduced later in a chapter of their own. A short overview is provided here.

Barcode Generator This is the right choice if you want to generate a single barcode and print it or export it as an image.

Label Editor The right choice for designing and printing business cards, address labels or designing labels for batch printing with data.

Batch Printing A very powerful mode of **KBarcode** that loads a label previously designed with the "Label Editor" and prints it with data provided by a data source like a SQL database, CSV file or the KDE address book.

Edit SQL Tables Helps maintaining the SQL tables used by **KBarcode** . Comes with an easy to use importer for comma separated value files.

3.2 The Preferences Dialog

Before describing the modules of **KBarcode** in detail, a short overview about some of the options in the preferences dialog is provided. The preferences dialog can be accessed in the main windows by selecting *Configure KBarcode ...* from the menu *Settings* .

3.2.1 Print Settings

On this page the resolution of the output created for the printer can be configured. **KBarcode** can generate output with 300, 600 and 1200dpi for any printer. Printing with 300dpi (default) is faster, but if you have problems scanning the barcodes printed at 300dpi you should select a higher resolution here.

The preview page format can be changed too in this dialog. It affects only the preview but not the actual printing.

3.2.2 SQL Settings

The SQL settings can be changed at any time and this dialog is the place to do it. All fields have the same meaning as in the configuration wizard. Configure the username, password, host and driver so that it is possible to connect to the database and set the database to the name you want to use for the database.

After changing the database settings in this dialog you might want to import to recreate the SQL tables and import the label definitions again. For this purpose select from the main windows *Settings* menu *Create Tables* and *Import Label Definitions* in this order.

3.2.3 Import

If you intend to use comma separated value (CSV) files with **KBarcode** , you have to configure the layout of these files in this dialog. CSV files can be used to import data during batch printing for user defined variables, to import the articles for printing during batch printing and to import data into a SQL table. For all these tasks the settings of this dialog are used.

Comment Lines starting with this character in CSV files are ignored as comments (Default Value: #)

Separator The column separator (Default Value: ;)

Quote Character Columns can be quoted using this character (Default: no value)

An example CSV file with this settings might look like this:

```
# The first line is a comment and is ignored
DataField1;DataField2;DataField3;
Line2_1;Line2_2;Line2_3;
```

An example CSV file using quoting looks like this:

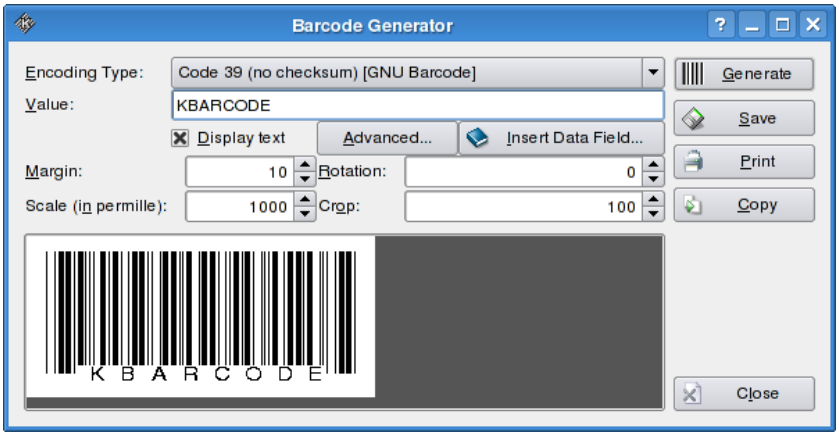
```
# The first line is a comment and is ignored
"DataField1";"DataField2";"DataField3";
"Line2_1";"Line2_2";"Line2_3";
```

It is possible to configure the meaning of the fields in the CSV file which is used to import articles during batch printing. Please note that this setting is only used for this special case! Any of the three fields in this CSV file can be configured to be interpreted as *Quantity*, *Article Number* or *Group*. The meaning of these fields will be introduced later during the explanation of the batch printing mode.

3.2.4 On new

The behavior of the batch printing module during printing articles from **KBarcode** SQL tables can be modified here. **KBarcode** can start a new page, do a line break, print a label with a big **X** or print the current article number whenever the current article or group changes. The printing of articles will be explained later.

4 Generating Barcodes



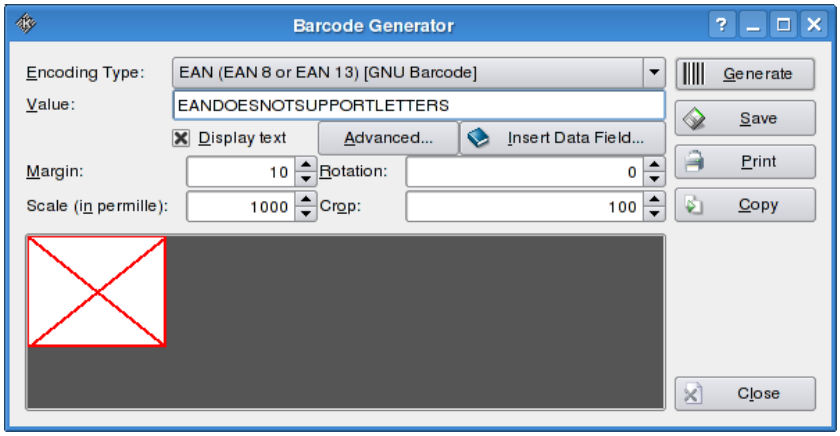
The barcode generator with a Code39 barcode created using GNU Barcode as backend

The barcode generation module creates barcodes using one of the installed barcode backends. Created barcodes can be printed directly, saved as an image in various formats or copied to the clipboard. The dialog used to set the barcode properties in the "Label Editor" has the same options. Therefore you should also be able to create barcodes in the "Label Editor" after reading this chapter.

To create a barcode, two fields are mandatory to be set. First you have to select the *Encoding Type* of the barcode. There are many different encoding types for barcodes, each has its own strength and weaknesses as well as special use cases. For example ISBN barcodes are used on books, EAN barcodes are used on products in Europe, UPC barcodes are used in the United States for products and Code39 was created for the US Army. It depends on your use case which encoding type to select.

The second mandatory field is of course the *Value* of the barcode, which is returned by the barcode scanner after reading the barcode. Be careful on what you enter. Certain barcode types accept only numbers (EAN or UPC), others accept only upper or lower case letters but not both. **KBarcode** will show a red box with a cross instead of a barcode if your input cannot be encoded into a barcode with the selected encoding type.

Press *Generate* to create the barcode. Once a barcode has been created successfully you can print, save or copy it to the clipboard.



A EAN barcode with an invalid value, as EAN does support only numbers and no letters.

The text below the barcode can be disabled using the *Display text* property. It is not necessary for a barcode to have a text line, because the scanner does only read the bars, but it is often better if humans can read the barcode, too.

Margin is a quiet zone around the barcode. The default value 10 is a good choice. Barcodes need an empty quiet zone around them so that a barcode scanner can find the end and beginning of the barcode more easily.

If a bigger or smaller barcode is needed the *Scaling* property can be used to shrink or enlarge the barcode. It takes a value in permille to control the amount of shrinking or enlarging. That is why 1000 is the default value (1000 permille equals 1).

Often barcodes are required to be rotated. The *Rotation* property supports barcodes rotated 90, 180 and 270 degrees. An arbitrary degree value can be entered, but the generated barcode will most likely not be readable if the degree is not a multiple of 90 degrees.

The height of the bars can be controlled by the *Crop* property. Sometimes the default bar height is to high and one wants to crop the bars at the top. This can be controlled by a percent value using this property. Certain barcodes are not allowed to be cropped, like 2D barcodes. The property is disabled for these barcodes.

Properties not common to all of the barcode backends are accessible through the *Advanced* button in the barcode generator dialog.

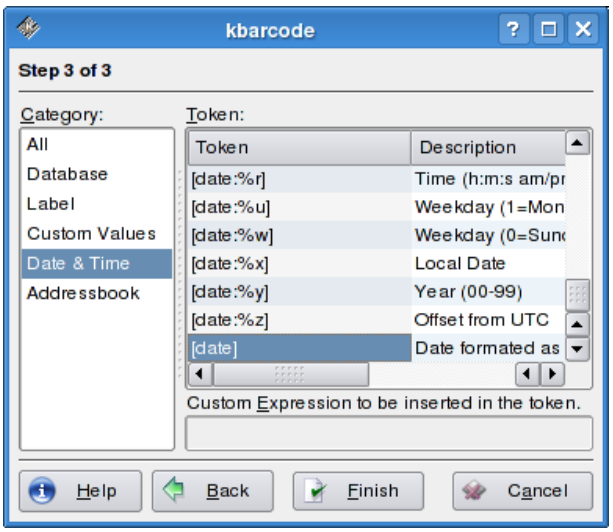
All barcode backends support "barcode sequences". Barcode sequences are more interesting in the label editor than here, but a short explanation will not do any harm. If barcode sequences are enabled, blocks of the character # can be used in the barcodes value. **KBarcode** will replace any # with a

number that is increased for every barcode label printed. The more # are used the more leading zeros are generated(i.e. ### will be replaced by 001).

Barcode Writer in Pure Postscript supports colored barcodes and enabling/disabling of checksums through the advanced dialog.

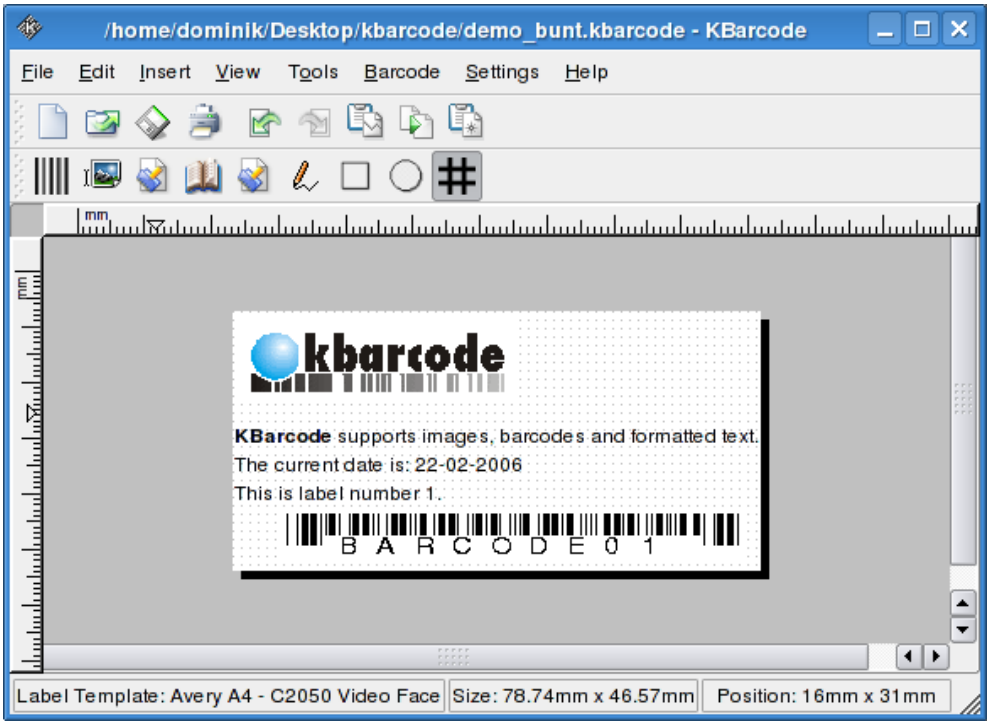
The **TBarcode** backend has options like controlling the module width, setting the height of the barcode and a few more.

The barcode value can be retrieved from a SQL database, a JavaScript statement or a variable like the current date here too. If you use any kind of function or variable, make sure that the return type of the function or the content of the variable can be encoded by the barcode type you have selected. To insert variables or functions click the button *Insert Data Field...* and follow the wizard. Data fields have the form `[SOMENAME]` where *SOMENAME* is the name of a function or variable. For example `[date]` will be replaced by the current date.



Inserting the current date as data field.

5 The Label Editor



A label with an image, a barcode and some formatted text in **KBarcode** 's label editor

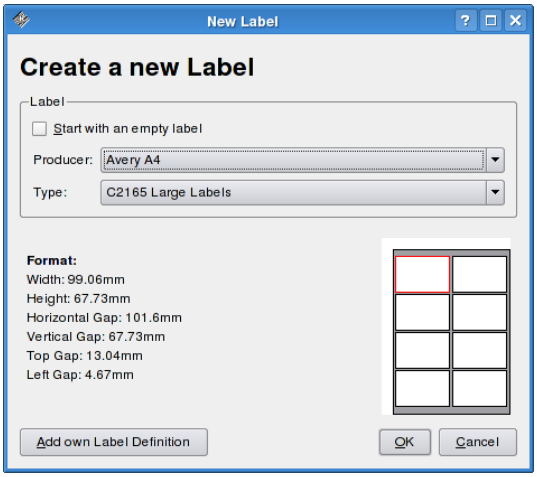
The label editor is the heart of **KBarcode** . All labels are designed in the powerful WYSIWYG[1] label editor. You can print labels like business cards or CD covers directly from the label editor, other labels like article labels are better printed from the batch printing module where they can be combined with data from a database or CSV file. Nonetheless they are still designed in the label editor.

The label editor supports file opening and saving as well as cut, copy and paste and undo and redo like any other KDE application. As a reason these features should be easy to use and are not explained explicitly.

[1] What you see is what you get

5.1 Creating a new label

When opening the label editor it will ask you to create a new label. If it does not ask, select *New* from the *File* menu. Select a predefined label template from the list sorted by producer and label type or create a new label definition, if none suits your needs. **KBarcode** comes with more than 1000 predefined labels, so most frequently used label types are already included. A preview of the currently selected label when printed on a page is shown on the bottom right side of the dialog, whereas the correct measurements of the selected label definition are shown on the bottom left side.



Selecting a label definition for a new label

To create a label definition of your own, select *Add own Label Definition*. A dialog opens where the measurements of the needed label can be entered. A real-time preview shows the results of the definition when being printed. *More Information* gives access to a graphic describing the meaning of the different measurement values. After the label definition was added, it can be used to create a new label.

Hint! Wether **KBarcode** uses inch or millimeters for its label definitions depends on the locale settings of your KDE installation.

5.2 Designing a label

A new label has been created and it is time to draw something on the label now. The toolbar at the top of the label editor window has an icon for each component (like barcodes, pictures and text) that can be placed on the label. After clicking on an icon the selected component is placed automatically on the label (on a randomly chosen position). You can move and resize it later

on the label. Alternatively components can be insert using the *Insert* menu.

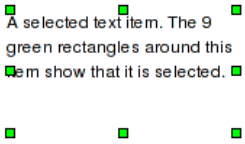
The toolbar in the label editor to add components like barcodes or text onto the label.

The components that can be inserted from the toolbar are in the order as on the picture:

- Barcode
- Image
- Text (multi-line and formatted)
- Data field
- Text Line (a single line of text without formatting)
- Line
- Rectangle
- Circle
- Toggle grid for easier positioning of components on the label

5.2.1 Moving and resizing items

To move an element, select it by clicking on to it with the left mouse button and moving the mouse with the left mouse button hold to the position you want the item to be. Resizing works similar. Select the item and drag one of the edges with the mouse to resize the item.



A selected item in the label editor. You can move and resize it now.

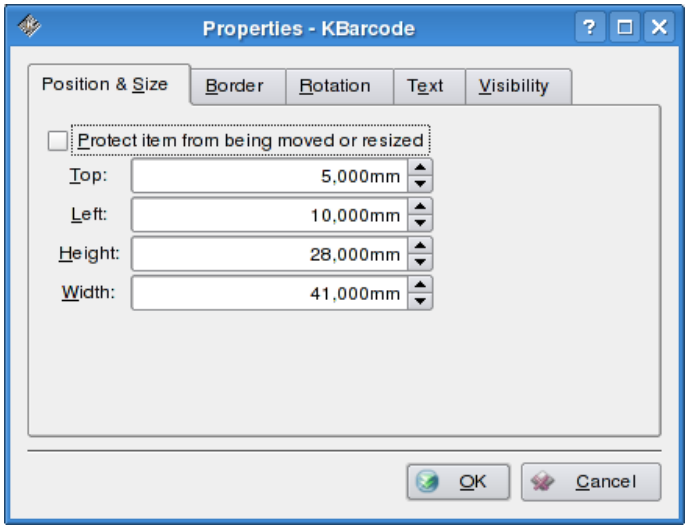
5.3 Properties

Every element that can be placed on the label has certain properties. Some of them are common to all elements, others are only available to special

elements. Properties can be modified either by double clicking on an element on the label or by right clicking on it and selecting *Properties* from the context menu.

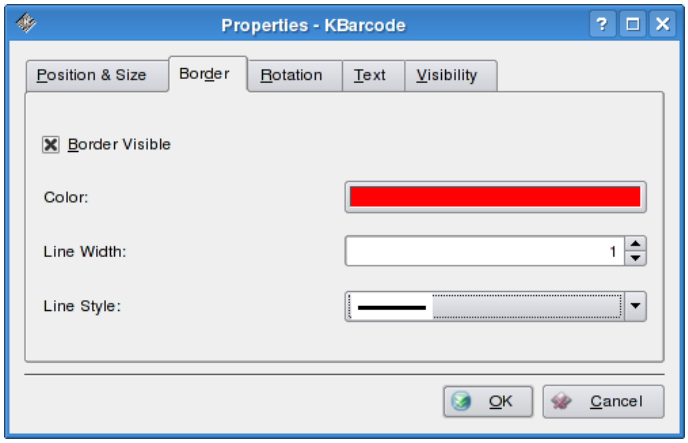
5.3.1 Properties common to all items

Every element on the label has a position and size. Even if the item can be moved and resized with the mouse it is more accurate to enter the position exactly. An exact position and size can be defined for every element in its properties dialog. Additionally it is possible to protect the position and size of an item, so that it is not moved with the mouse by accident.



The properties dialog page for modifying position and size.

Borders can be drawn around all elements. A border is a rectangle around the component. It has a color, a line width and a line drawing style. It should be mentioned that a border should be only drawn around a barcode, if it has a big enough quiet zone.



Setting a border for an element on the label.

Sometimes it is required to print a page of labels, but not all labels on the page should have the same contents. As a reason, every element has a so called "visibility property". That is a JavaScript function that is called for each label. If the function returns true, the item will be printed, otherwise not. These JavaScript functions are quite simple, but powerful if combined with **KBarcode** 's data fields which will be introduced later. The two examples below use the data field *[index]* which will be replaced during printing with the number of the label currently being printed (for the first label printed it is 1, for the second label it is 2 ...).

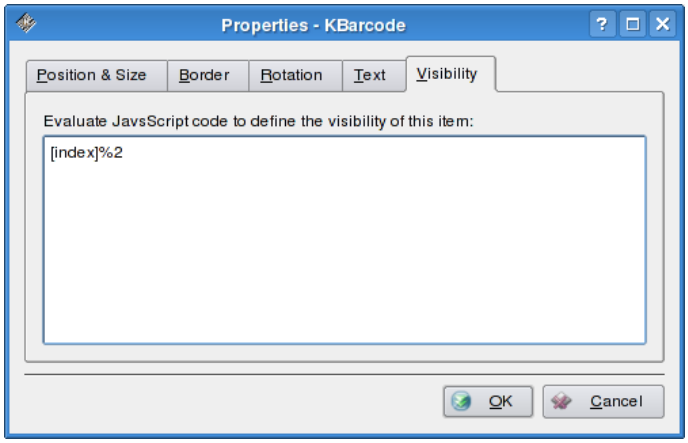
To print a element only on the label number 5 you could use a JavaScript function in the visibility field like this:

```
[index]==5;
```

To print only on every second label, you would use:

```
[index]%2;
```

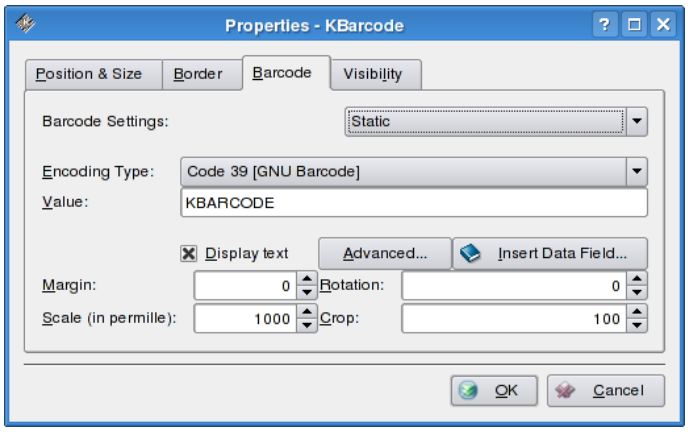
More powerful functions are of course possible, too, but won't be explained here in detail.



The visibility property.

5.3.2 Barcode Properties

The properties of a barcode in the label editor are the same as described in the chapter "Generating Barcodes" at page 15.



Barcode properties in the label editor.

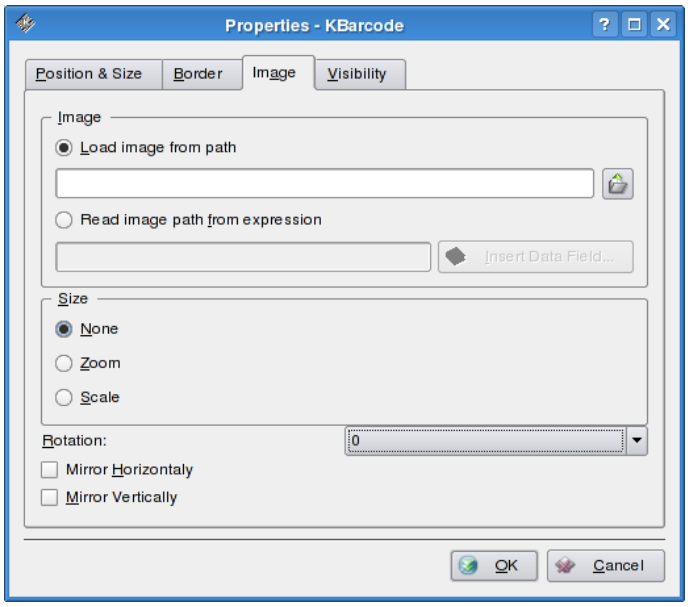
The only notable difference is the field *Barcode Settings* at the top of the page. The default value *Static* is a good choice. With this setting you can define barcodes with a fixed value for each label or barcodes which get their value from a user defined variable or one of **KBarcode** 's data fields which will be introduced later. All other values are only available if you use a database along with **KBarcode** . If you use the database tables for article printing, you can also choose barcode values from the database like *Main:EAN*. **KBarcode** will get the values for the barcode automatically from the database.

Barcode Sequences

Barcode sequences are a powerful feature of **KBarcode** . Under the *Advanced...* barcode options enable the option *Enable sequences*. It is now possible to use the character # in the *Value* field of the barcode settings. **KBarcode** is going to replace the # with a number that is increased for every printed label. More # mean leading zeros. I.e. `BARCODE##` will be replaced before printing with `BARCODE01` . Therefore it is very easy to print a page of, let's say 10 barcodes, numbered from 1 to 10. Enable barcode sequences and set the barcodes value to ##. Simply print 10 labels directly from the label editor and the barcodes will be numbered automatically from 1 to 10 on the print out.

5.3.3 Image Properties

KBarcode supports a wide range of image formats that can be printed on a label. The formats are not listed here as they depend on your KDE installation. Generally it can be said that **KBarcode** will support more different image formats the newer your KDE installation is.



The image properties dialog.

First of all, you have two possibilities to embed an image on a label. **KBarcode** can load a picture from a fixed path on your system and store it along with the label. This feature is used for company logos which are the same on each label. Additionally it is possible to define the path of the image to load during printing using an expression. The path to the image could be stored in a database. As a reason one can print different images for different customers using the same label. **KBarcode** expressions using data fields are explained later. If you chose to use the image path from an expression, **KBarcode** will draw a red box displaying the expression till it can be evaluated (usually only during printing).

Secondly there are options for scaling, rotating and mirroring the image.

Size: None The image is not resized anyway. If it is to big for the image frame, it is simply cropped. If it is to small, it won't be resized.

Size: Zoom The image is resized to fit the image frame (which can be resized using the mouse). The complete image frame is filled with the image.

Size: Scale The image is resized to fit the image frame, but the ratio is kept. This is usually a good choice.

Rotation Rotate the image by a multiply of 90 degrees.

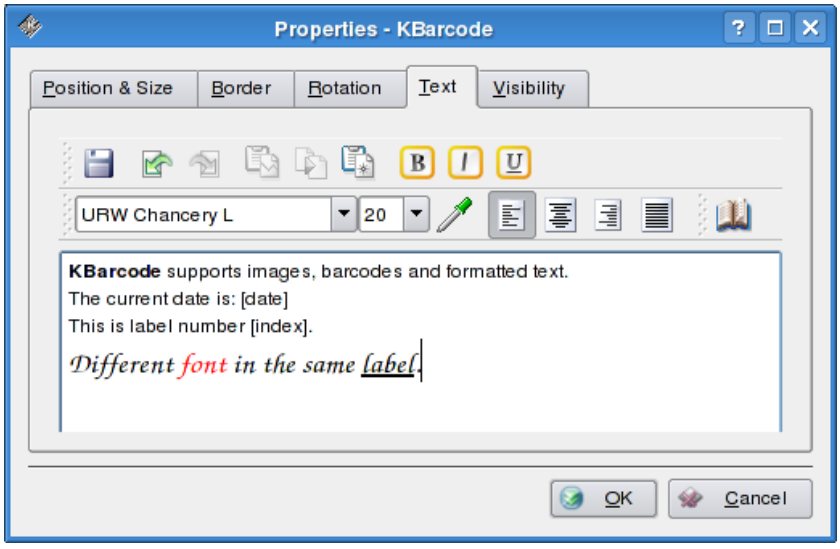
Mirror Horizontally The image is mirrored.

Mirror Vertically The image is mirrored vertically.

5.3.4 Text properties

Text is edited and written in the richt text editor included in **KBarcode** . Writing text works like in any modern word processor. The current font, font size, color or formattings like bold, italic and underlined can be applied easily to any selected part of the text.

Data fields can also be included into the rich text directly. You do not have to create an extra data field element on the label. When including a data field and you intend to format it somehow. Make sure that the whole data field has one formating and not two different formattings (e.g. "*[ind***ex]**" cannot be evaluated by **KBarcode**)

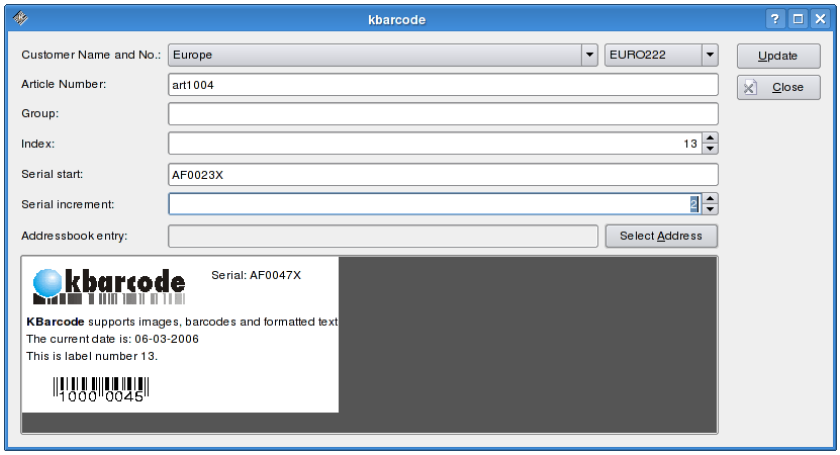


The rich text editor with formated text and two data fields.

5.4 Label Preview

Before printing the label it is a good idea to take a look on it with some real world data. Select *Preview...* from the *View* menu in the label editor. A dialog is opened. Several data fields like the index or serial number

can be set here. Also an article from the SQL database can be selected if a database is configured for usage with **KBarcode** . To see the label with the entered data, click on *Update* after all the required data has been entered.



The preview dialog.

6 Data fields

KBarcode has a powerful feature to include data into the printed labels. It is possible to use a so called data field into any text field, as value of a barcode or even to construct the path of an image to load. Data fields have the form `[SOMENAME]` where *SOMENAME* is the name of a function or variable.

There are different types of data fields. Some provide only static data, like the current date or time, others can execute SQL queries or JavaScript functions to create data for printing during runtime.

To use a data fields it is enough to type the name of the data field with the enclosing brackets [and] as text in a text field, as barcode value or as path to an image. Alternatively, **KBarcode** includes a comfortable wizard to insert all kinds of data fields.

6.1 Simple Data fields

Simple data fields are replaced by their value during printing. The data field `[date]` is replaced with the current date during printing. A few important data fields are listed in the below.

[**col**] The column of the printed label on the current page.

[**date**] The current date.

[**filename**] The filename including the complete path of the label being printed.

[**index**] The index of the label. The first label has the index 1, the second label has the index 2 ...

[**page**] The number of the current page.

[**row**] The row of the printed label on the current page.

[**serial**] A serial number. The start value of the serial number can be specified right before printing.

If you are printing articles from **KBarcode** 's SQL tables as explained later, you can use special data fields for the article number, the customer or the barcodes value.

6.2 Addresses

KBarcode can use the KDE address book as data source for label printing. Special data fields exist for almost every entry in the address book. A address book contact or a list of contacts can be selected before printing to fill the label with the correct information. To insert the complete address of the selected contact use the data field `[address]` , to insert the name only use the token `[address_name]` . For a complete list of the supported data fields, please use the data field wizard of **KBarcode** .

6.3 User Defined Variables

User defined variables are a powerful concept. The user can define variables of his own that work like ordinary data fields, but get their value right before printing from a CSV file, the result set of a SQL query or directly from the user.

To define a new variable, either use the wizard or type in any text field `[$NameOfTheVariable]` to create a variable named *NameOfTheVariable*. One variable can be used in a text field and a barcode (or in more than one text field and barcode) at the same time. Just make sure both variables have exactly the same name.

The usage of user defined variables will be explained more detailed in the section "Batchprinting".

6.4 SQL Queries

Results of SQL queries can be included into labels or used as data for barcode generation. The data field for an SQL query has the form `[sqlquery:SomeSQLQuery]` where *SomeSQLQuery* is a SQL query that may contain other data fields.

An example of an SQL query which could be used as a barcode value is:

```
[sqlquery:SELECT barcode_value FROM barcode_values_table
```

```
WHERE customer=[customer_no]]
```

As you can see one can use data fields inside of other data fields. `[customer_no]` is a data field of **KBarcode** that returns the name of the current customer during article printing. The table *barcode_values_table* on which the select is performed has to be a table in the database which was configured for usage with **KBarcode** before.

Another widely used type of query utilizes the `[index]` data field: `[sqlquery:SELECT value FROM table WHERE id=[index]]` .

6.5 JavaScript Functions

Small JavaScript functions can perform more complex tasks, like string manipulations or calculations. To insert a JavaScript function in a text field or barcode write a data field starting with *js:*, i.e. `[js:4+5]` . The JavaScript interpreter will add 4 and 5 in this example and the text field containing this script will have the value 9.

Note that JavaScript functions used in **KBarcode** have to return a value and that variables cannot be shared between different JavaScript data fields. Of course it is possible to use other data fields inside of JavaScript functions. Powerful tasks can be performed by a combinations of SQL queries, JavaScript and simple data fields.

```
[sqlquery:SELECT value FROM table WHERE id=[js:if( [index] % 2 )
1; else 2;]]
```

This function will insert value from table with id=1 for an even label index and the value with id=2 for an odd index.

A complete description of JavaScript with syntax and all supported functions would be to much for this handbook. If you want more information on JavaScript please take a look at *http://en.wikipedia.org/wiki/JavaScript* .

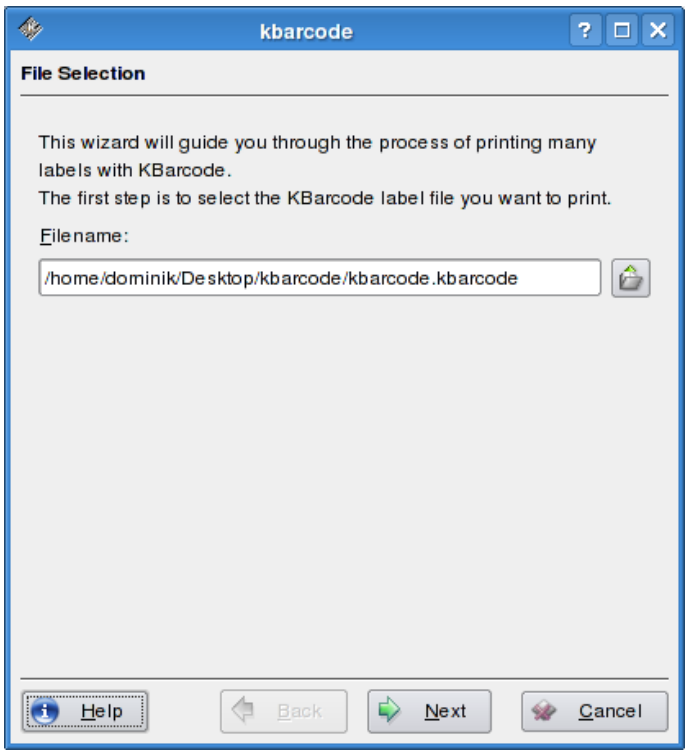
The language and syntax used in JavaScript data fields is the same as to define the visibility of components on the label.

7 Batchprinting

Batchprinting means to print a large number of labels in one go without user interaction between the labels. A label design and a data source is specified before printing. **KBarcode** will merge the data with the label design and print the labels.

This process can be automated using command line options and the DCOP[1] interface provided by **KBarcode** .

A wizard will guide you in batchprinting mode through the different steps. The first step when the wizard is launched is to select a label file.



Select the label to print as the first step in batchprinting mode.

[1] **Desktop Communication Protocol**: a way to control KDE applications from the command line and from shell scripts.

As second step you have decide what kind of labels you want to print with which data. All of the options below are described in detail in separate sections.

- Print labels without data
- Print articles from a SQL database
- Import user defined variables and print
- Print contacts from the address book

7.1 A Quick start into Batchprinting using Presentations

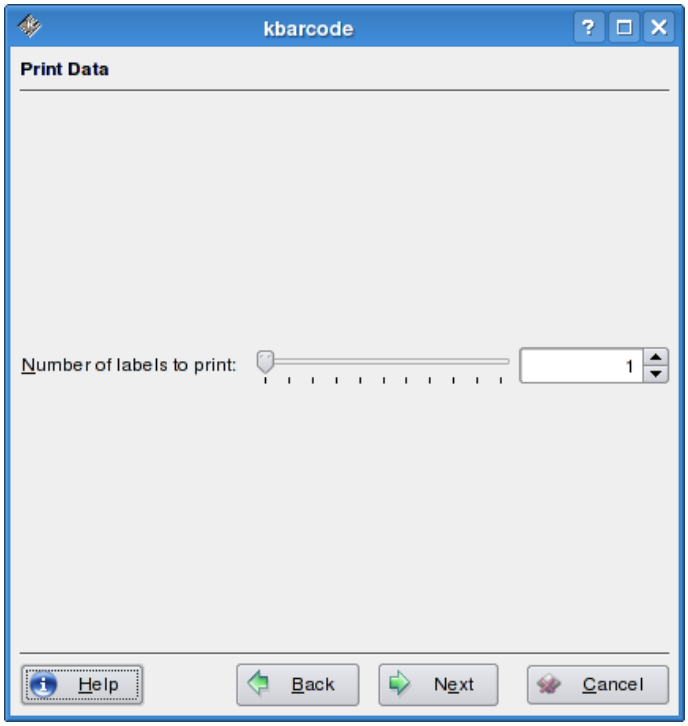
Two presentations that give a good overview over batchprinting using **KBarcode** are available online in the documentation section of our website. The first one describes batchprinting in general and the second one explains how to print address labels from the KDE address book.

http://www.kbarcode.net/Docs.17.0.html

Both presentations are a good help to guide you step by step through the printing process, so they are definitely worth a look if you are using **KBarcode** for the first time.

7.2 Printing labels without data

Sometimes one wants to print labels without any external data source. For example, this is the case for business cards. They are designed once and no external data is needed as all the content is already part of the design. For labels without data, it is only necessary to specify how many labels should be printed.



Specify the number of labels to print.

It is later possible to set the start value of a serial number (`[serial]`) to add some dynamic content to the label. Of course barcode sequences and simple data fields like `[date]` , `[index]` or even SQL queries will work, too.

7.3 Print articles from KBarcodes SQL database

This option is only enabled if you have configured a SQL database to work with **KBarcode** .

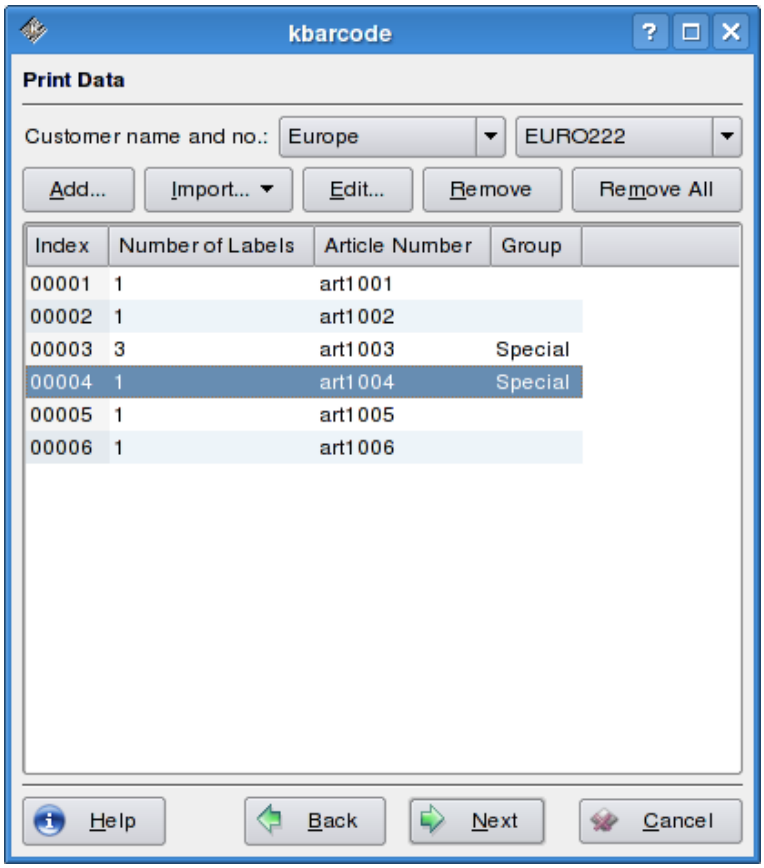
The information to print the labels is retrieved from the SQL tables that were created by the configuration wizard. Two tables are used to store the required information. Before starting to print articles, the tables should be filled with all the required information (customers and articles).

`customer` This table contains a customer number and the name of each customer. If you do not need special customer information, you can use one of the example customers that were created during setup or create a default customer by yourself.

`barcode_basic` All articles are stored in this table. Every article has an article number, which is used to identify this article. Every article in the database has to have an article number. An article description should be entered, too.

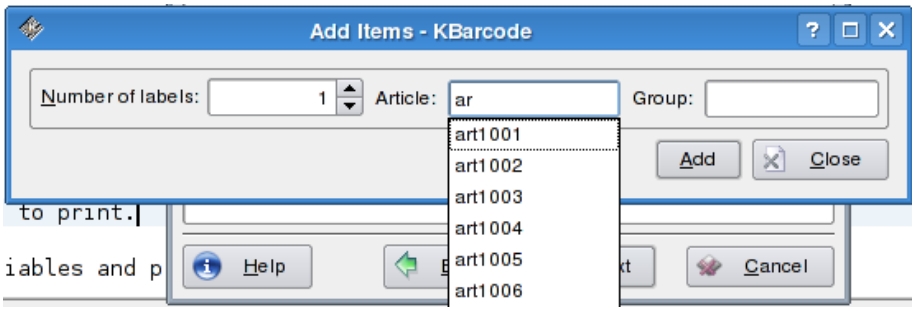
The fields `barcode_no` and `encoding_type` contain the required information to print a barcode for this article. `barcode_no` is the value of the barcode (e.g. "1231241") and `encoding_type` is the barcode encoding that should be used (e.g. "ean"). The name of the wanted barcode encoding can be found in the Appendix in the section **Supported barcode types**.

The additional table `customer_text` can be used to define the same article differently for different customers. The fields are a combination of those found in `barcode_basic` and `customer` .



Printing several articles from the database.

When the tables are setup with the correct information it is simple to print some article labels. The first step is to select the customer for the printed labels at the top of the dialog. The second and last step is to add the articles you want to print. Articles can be added manually with the dialog below. The same dialog is used to modify items once added to the list. Double click onto an item into the list to modify it (the article number, group or the number of labels to print).



Adding an article to the list of articles to print. Completion for the article number is available.

Adding all articles for printing one by one is uncomfortable for a large amount of articles. As a reason several ways to import articles are possible. The easiest one is to import all articles form `barcode_basic` . Most of the time you will want to import the list of articles from a specially created CSV file. The CSV has three columns. The meaning of the columns can be configured to your needs. See page 14 for the configuration options.

The default setting is a CSV file with the layout:
`NumberOfArticles;ArticleNumber;Group` .
A sample CSV to print two labels of article `art1001` and two labels of article `art1002` file might look like this:

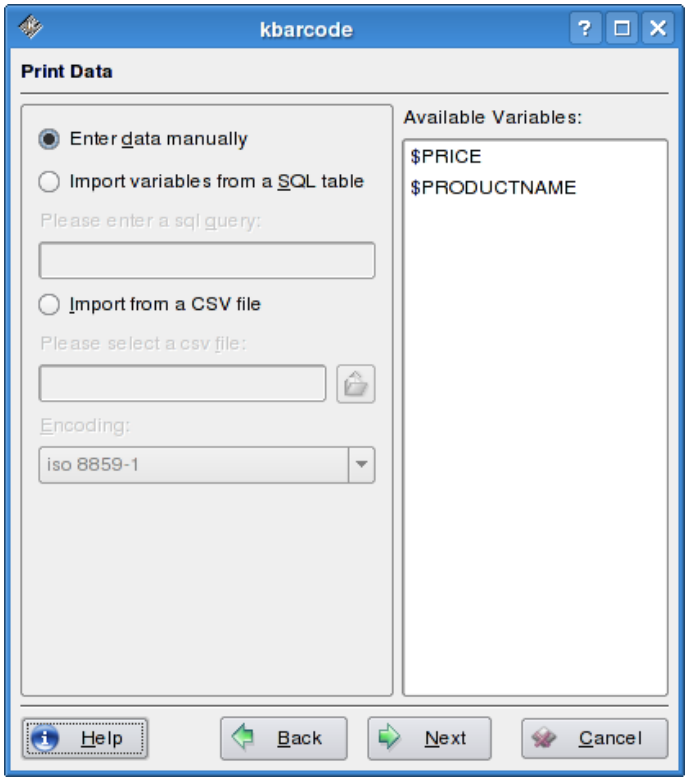
```
# The first line is a comment and is ignored
2;art1001;GROUP1;
2;art2002;GROUP2;
```

Data in the same format can be imported from the clipboard.

7.4 Import variables and print

If the label contains user defined variables (data fields of the form `[$MyVariable]`), it is necessary to import the data with which the variables are filled before printing in some way. Three different ways are available to fill the user defined variables contained in the label, which are shown on the right side of the dialog, with data.

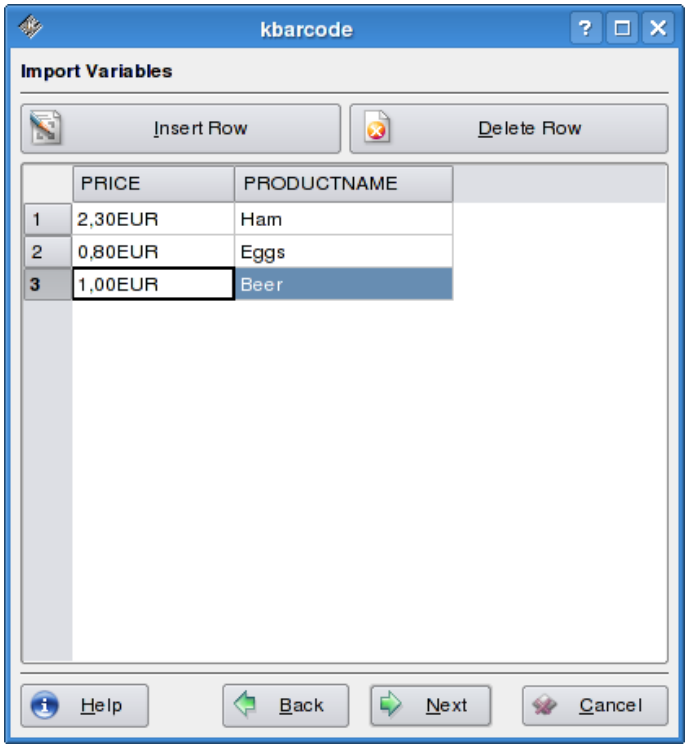
- Enter them manually
- Get the data from a SQL query
- Read the data from a CSV file



Select the data source for the user defined variables. Available variables are listed on the left.

7.4.1 Enter the data manually

The data for the user defined variables can be entered by hand. This is only useful for really small amounts of data. Data that comes from a CSV file or from a SQL query can be edited manually the same way as you enter your data here. To add data click on *Insert Row* , to delete a selected row of data click on *Delete Row* . To change the value of a field in the table double click on the cell you want to change.



Providing data for batchprinting.

7.4.2 Import from CSV file

Data can be exported into CSV files from most applications. So this is most likely the preferred way to get data into **KBarcode** . CSV files for this purpose have a special format. The first line of the CSV file is the header of the file. The header defines which column of data is used for which user defined variable. Therefore the first row contains the names of the user defined variables. A CSV file for two user defined variables `[$PRICE]` and `[$PRODUCTNAME]` should look like this:

```
PRICE;PRODUCTNAME;
# The following lines contain data for the two variables
10.000EUR;Car;
0.80EUR;Bread;
30.00EUR;T-Shirt;
```

Please note that it is not required to use the brackets and the dollar sign when defining the variable in the CSV file.

The encoding of the imported data can be specified. The data can be edited by hand after the import in the next step.

7.4.3 Import from SQL query

If the data is already available in a SQL table it is a good idea to import it directly from the SQL database. The SQL table containing the data must be in the database configured for usage with **KBarcode** .

We have a table `products` with some data:

```
kbarcode=> SELECT price, productname FROM products;
 price | productname
-------+-------------
 10000 | Car
  0.80 | Bread
    30 | T-Shirt
(3 rows)
```

The user defined variables used in **KBarcode** have exactly the same name as the columns in the SQL table. As a reason the data can be imported with an easy SQL statement:

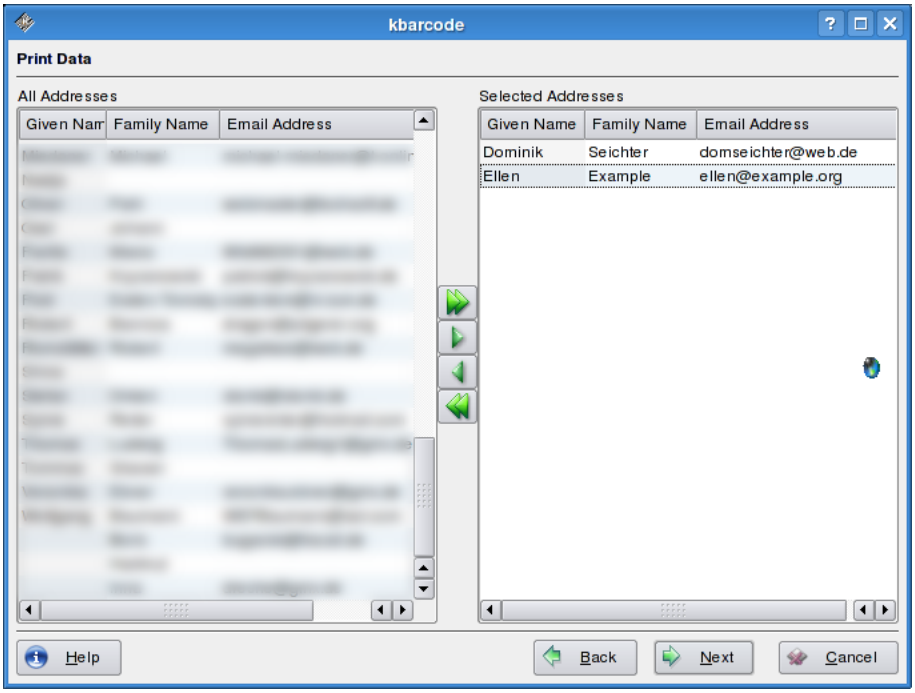
```
SELECT price, productname FROM products;
```

The imported data can again be edited by hand in the next step.

7.5 Printing address book contacts

KBarcode can print a label for a number of contacts from your address book. For each selected contact a label is printed and the data of the contact is used to fill the various address related data fields of **KBarcode** .

To select contacts for printing, mark them with a mouse click on the left side of the dialog and transfer them with the arrow in the middle to the right side. All contacts in the right list are going to be printed. It is also possible to move all contacts at once from the left list to the right list using the arrow button at the top.



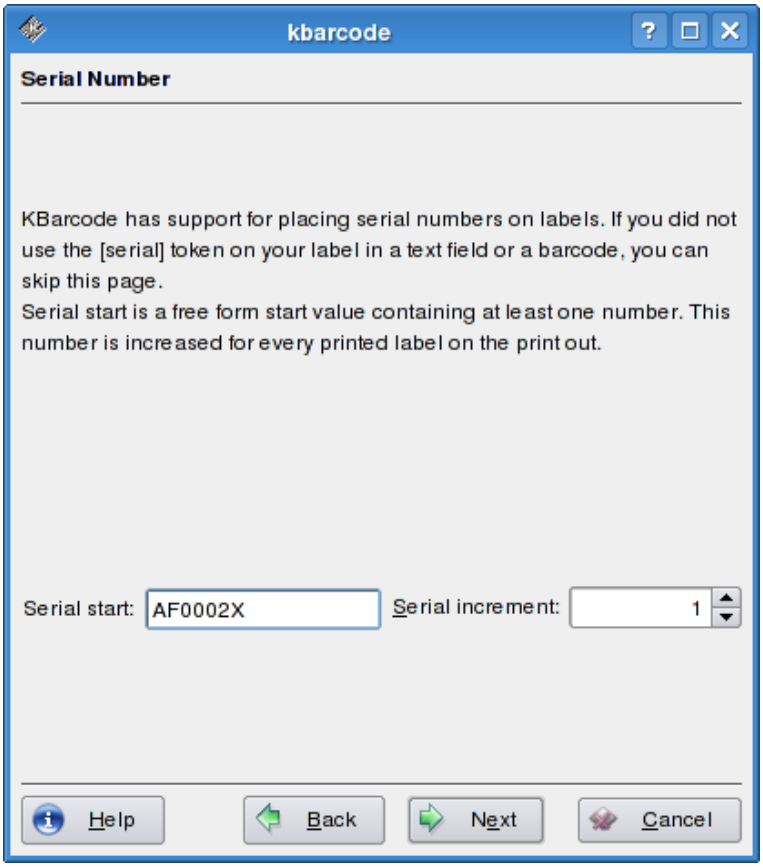
Selecting several contacts from all contacts of your address book[a].

[a] Addresses on the left are blurred for privacy reasons.

7.6 The serial number

Wether you print labels without data, with imported data, address labels or articles from a SQL database you can set the value of the `[serial]` data field before printing. The start value and the amount by which the serial number is increased for each label can be configured.

A serial number is a free form string which must contain at least one number. **KBarcode** will find the number in the string and increase it for every printed label. If the serial number is `AF0002X` , **KBarcode** will replace any `[serial]` data field on the first label with `AF0002X` and with `AF0003X` on the second label. As you can see, it is possible to add guiding zeros in front of the number. It is not necessary that any characters are contained in the serial number. The simplest serial number is a single digit like: `1` .



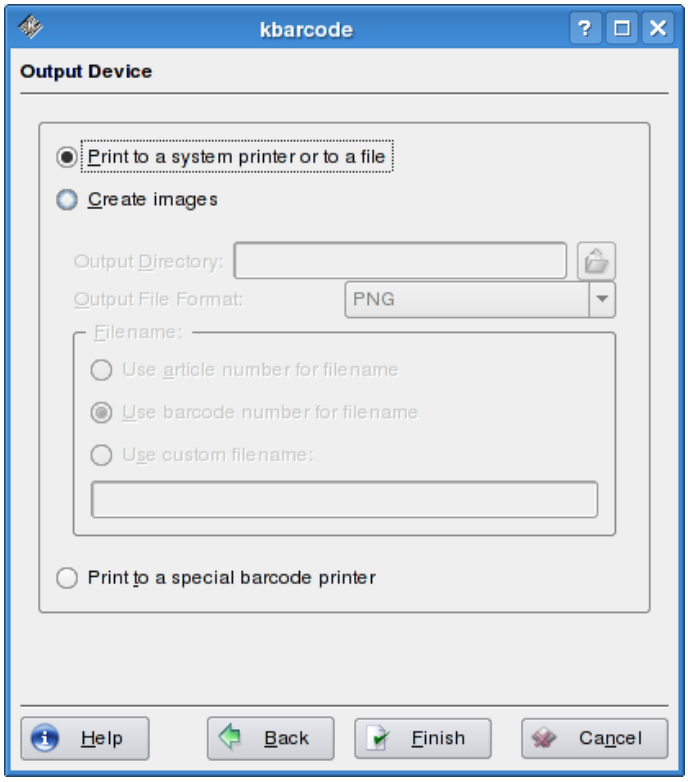
Setting the serial number.

7.7 Selecting an Output Device

KBarcode can print on any installed system printer and can generate PostScript and PDF files through the KDE printing system. Instead of printing each label to a real printer, it is possible to generate an image for each label which is quite useful for webpages. And finally output can be created for a few special barcode printers.

The last step during batch printing is to select one of these options as output device.

- Print to a system printer using CUPS and the KDE printing system
- Generate images
- Print directly on a special barcode printer



Setting the serial number.

7.7.1 Printing to a system printer

The standard KDE printing dialog is opened after a click on the *Finish* button. Please consult the KDE documentation for the usage of this dialog.

7.7.2 Generating images

When you intend to generate an image for every label that gets printed, an output directory has to be selected first as well as the output format of the images. All image formats supported by KDE can be used again to save the images. Last but not least you have to specify how the images should be named. If you are printing articles from the database it is a good idea to use the article number as filename, otherwise you may want to use a barcode number from the label if available or specify a custom filename.

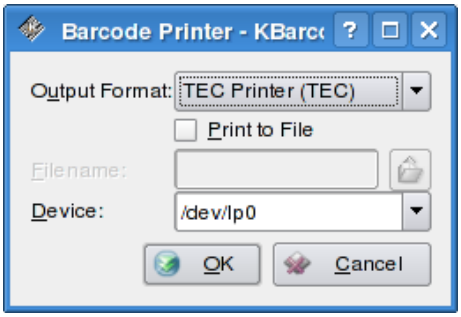
If a filename exists already, **KBarcode** will append a number to the filename instead of overwriting an existing file.

7.7.3 Printing to a special barcode printer

This feature is experimental and does not provide WYSIWYG support as most of **KBarcode** 's features are not supported by these barcode printers. The data generated for the printer can be send to a device directly or can be written to a file first. You have to send this file to the printer by yourself later in this case.

Supported printers include:

- TEC Barcode printers (TEC)
- Zebra printers (ZPL)
- Intermec printers (IPL)
- EPCL printers (EPCL)



Selecting a barcode printer

Please note that most barcode printers do not support multiline or formatted text fields.

7.8 Batch printing from the command line

KBarcode has a command line interface so that printing can be fully automated. Please note that a X server is still required even if you are printing directly from the command line. **KBarcode** will not work without a running X server. This might change with the switch to Qt4 but there are no concrete plans on this topic yet.

Not all command line options are documented in this chapter. All command line options with a short description can be seen using the `--help` option.

```
$ kbarcode --help
```

KBarcode has three command line options to start directly in the label designer, the batch printing module or into the barcode generator.

- --label
- --batch
- --barcode

To load a label file called *my_label.kbarcode* directly in the label editor from the command line use the following command line. If no file is specified along with the --label option, the label designer is started with a new label.

```
$ kbarcode --label my_label.kbarcode
```

There are several additional options to the --batch command line switch. Those options can be used to fully automate batchprinting with **KBarcode** from the command line. Usually you will also specify a filename along with the --batch option.

```
$ kbarcode --batch my_label.kbarcode
```

Additional options are:

`--numlabels` *`value`* Start the batchprinting without any data and prints *value* labels.

`--importsql` *`query`* Batchprinting with data imported from a SQL query. **KBarcode** has to be configured to connect to a database automatically on startup so that this option will work correctly.

`--importcsv` *`csvfile`* Data for user defined variables is imported from the given CSV file.

`--serialnumber` *`value`* Use *value* as serial number on the label.

`--serialinc` *`value`* Increment a serial number on the label by *value* for each printed label.

`--print` Print immediately with the specified options and exit afterwards.

An example that will print 10 labels with a serial number on the printer called *psc1500*.

```
$ kbarcode --batch serial_label.kbarcode
  --serialnumber AAA002
  --numlabels 10 --print --printer psc1500
```

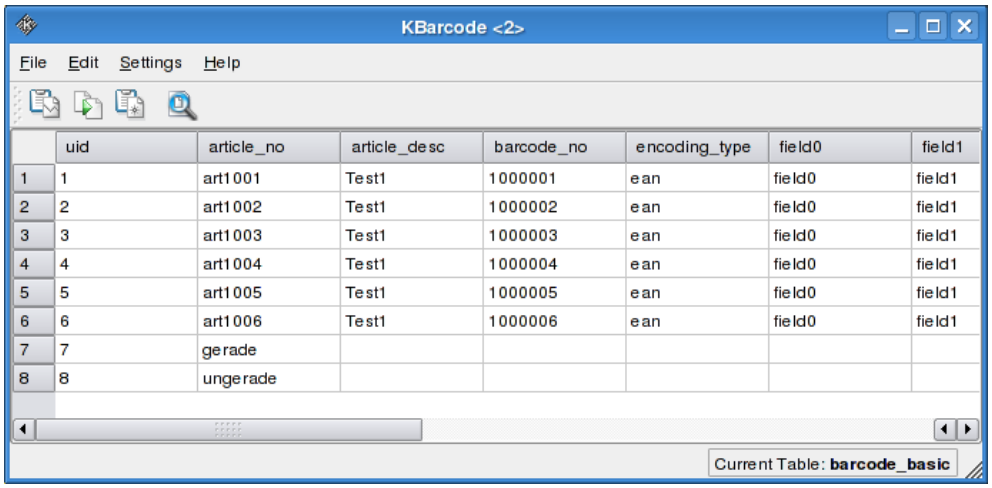
Another example which will import the data for batchprinting from the command line. **KBarcode** will only ask the user for the printer to print on.

```
$ kbarcode --batch label.kbarcode --importsql
  "SELECT customer, articleno, barcodevalue
  FROM articlecollection
  WHERE customer='customer1' OR customer='customer2'"
  --print
```

If you need more control over the batchprinting process, you should take a look at the DCOP options of **KBarcode** . You can easily browse the DCOP functions using `kdcop` . Examples for scripting **KBarcode** using DCOP are available in our forum and mailinglist.

8 Editing KBarcodes SQL Tables

KBarcode includes a comfortable editor for the database tables that come with **KBarcode** . Please not that it can only edit the default tables and it is not possible to create new tables. Other tools specially designed for this task should be used for database maintenance. The included database editor is only suitable for small amounts of data.



	uid	article_no	article_desc	barcode_no	encoding_type	field0	field1
1	1	art1001	Test1	1000001	ean	field0	field1
2	2	art1002	Test1	1000002	ean	field0	field1
3	3	art1003	Test1	1000003	ean	field0	field1
4	4	art1004	Test1	1000004	ean	field0	field1
5	5	art1005	Test1	1000005	ean	field0	field1
6	6	art1006	Test1	1000006	ean	field0	field1
7	7	gerade					
8	8	ungerade					

The database editor.

Only the four included tables can be edited:

- label_def
- barcode_basic
- customer
- customer_text

A comfortable wizard is included to import data from a CSV file directly into one of these tables.

9 Support us!

KBarcode is an open source project done by volunteers. We always need help! You can support us in several ways, even without being a Linux wizard or computer expert.

You may help us with...

- Reporting bugs (to the mailing list).
- Sending us feature requests to improve **KBarcode** .
- Giving us feedback about success or failure of the usage of **KBarcode** in your company.
- Translating **KBarcode** in your language.
- Creating RPM files for different Linux distributions.
- Writing an article about **KBarcode** for a magazine.
- Joining the mailing list and contributing directly to the project.
- Making advertisement for **KBarcode** and other Open Source Software.
- Donating some money to the PayPal account of Dominik, the main programmer of **KBarcode** . For details on how to donate please visit our webpage.

If you have found another way to support **KBarcode** , feel free to contact us. We are happy about any kind of feedback.

10 Thanks To

Many people contributed and helped to make **KBarcode** a successful project.

Not everyone who has contributed and helped **KBarcode** can be listed here. I want to name a few important persons though.

KBarcode is created by Stefan Onken *stonki@stonki.de* and Dominik Seichter *domseichter@web.de* .

All the programming is done by Dominik Seichter. The icons were designed by Anton Vaaranmaa *antonv@postikaista.net* and the logo was designed by Elrondo. I want to thank everyone you provided translations of **KBarcode** , created rpms, wrote a patch or is simply using the software.

Special thanks goes to all members of the **KBarcode** mailing list for frequent help and feedback.

Thanks goes to "Bike Alert Plc." for allowing Stefan Onken to work on this project.

Thanks to everyone who contributed.

11 Appendix

11.1 SQL Tables

The SQL tables used by **KBarcode** are listed here. They are created automatically by default, but it might me necessary to create it manually for certain databases. As a reason the complete table structure is listed here for all the tables. All SQL here was created using a MySQL database.

11.1.1 Label Definitions: label_def

This table contains the label definitions. How big a label is, which position it has on a page and how many labels are on a page. Normally it is not necessary to modify this table manually. **KBarcode** will work without these table, too. The label definitions are simply read from a file if this table is missing. Retrieving the label definition from a SQL database is faster though.

```
mysql> show fields from label_def;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| label_no     | int(11)      |      | PRI | NULL    | auto_increment |
| manufacture  | varchar(255) | YES  |     | NULL    |                |
| type         | varchar(255) | YES  |     | NULL    |                |
| paper        | char(1)      | YES  |     | NULL    |                |
| gap_top      | decimal(10,4)| YES  |     | NULL    |                |
| gap_left     | decimal(10,4)| YES  |     | NULL    |                |
| height       | decimal(10,4)| YES  |     | NULL    |                |
| width        | decimal(10,4)| YES  |     | NULL    |                |
| gap_v        | decimal(10,4)| YES  |     | NULL    |                |
| gap_h        | decimal(10,4)| YES  |     | NULL    |                |
| number_h     | int(11)      | YES  |     | NULL    |                |
| number_v     | int(11)      | YES  |     | NULL    |                |
```

```
| paper_type    | varchar(30)   | YES  |     | NULL    |                |
| compatibility | varchar(10)   | YES  |     | NULL    |                |
+---------------+---------------+------+-----+---------+----------------+
14 rows in set (0.00 sec)
```

11.1.2 Articles: barcode_basic

This table contains all articles. It might be necessary to import some data into it with an external tool other than **KBarcode** .

```
mysql> show fields from barcode_basic;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| uid           | int(11)      |      | PRI | NULL    | auto_increment |
| article_desc  | varchar(50)  | YES  |     | NULL    |                |
| barcode_no    | text         | YES  |     | NULL    |                |
| encoding_type | varchar(50)  | YES  |     | NULL    |                |
| field0        | varchar(50)  | YES  |     | NULL    |                |
| field1        | varchar(50)  | YES  |     | NULL    |                |
| field2        | varchar(50)  | YES  |     | NULL    |                |
| field3        | varchar(50)  | YES  |     | NULL    |                |
| field4        | varchar(50)  | YES  |     | NULL    |                |
| field5        | varchar(50)  | YES  |     | NULL    |                |
| field6        | varchar(50)  | YES  |     | NULL    |                |
| field7        | varchar(50)  | YES  |     | NULL    |                |
| field8        | varchar(50)  | YES  |     | NULL    |                |
| field9        | varchar(50)  | YES  |     | NULL    |                |
| article_no    | varchar(100) | YES  |     |         |                |
+---------------+--------------+------+-----+---------+----------------+
15 rows in set (0.01 sec)
```

11.1.3 Customers: customer

Customers are associated with a customer number in this table.

```
mysql> show fields from customer;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
```

```
+---------------+-------------+------+-----+---------+----------------+
| uid           | int(11)     |      | PRI | NULL    | auto_increment |
| customer_no   | varchar(20) | YES  |     | NULL    |                |
| customer_name | varchar(20) | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
3 rows in set (0.05 sec)
```

11.1.4 Customer Articles: customer_text

This table has the same fields as `barcode_basic`, only that the fields are associated with a customer number from the table `customer`.

```
mysql> show fields from customer_text;
+---------------------+-------------+------+-----+---------+----------------+
| Field               | Type        | Null | Key | Default | Extra          |
+---------------------+-------------+------+-----+---------+----------------+
| uid                 | int(11)     |      | PRI | NULL    | auto_increment |
| customer_no         | varchar(20) | YES  |     | NULL    |                |
| encoding_type       | varchar(50) | YES  |     | NULL    |                |
| article_no          | varchar(50) | YES  |     | NULL    |                |
| article_no_customer | varchar(50) | YES  |     | NULL    |                |
| barcode_no          | text        | YES  |     | NULL    |                |
| line0               | varchar(50) | YES  |     | NULL    |                |
| line1               | varchar(50) | YES  |     | NULL    |                |
| line2               | varchar(50) | YES  |     | NULL    |                |
| line3               | varchar(50) | YES  |     | NULL    |                |
| line4               | varchar(50) | YES  |     | NULL    |                |
| line5               | varchar(50) | YES  |     | NULL    |                |
| line6               | varchar(50) | YES  |     | NULL    |                |
| line7               | varchar(50) | YES  |     | NULL    |                |
| line8               | varchar(50) | YES  |     | NULL    |                |
| line9               | varchar(50) | YES  |     | NULL    |                |
+---------------------+-------------+------+-----+---------+----------------+
16 rows in set (0.01 sec)
```

11.2 Supported barcode types

11.2.1 Barcode Writer in Pure Postscript

Encoding	Database Name	Example
Australian Post	ps_auspost	
Code 11	ps_code11	
Code 128	ps_code128	
Code 2 of 5	ps_code2of5	
Code 39	ps_code39	
Code 93	ps_code93	
EAN 13	ps_ean13	
EAN 2	ps_ean2	
EAN 5	ps_ean5	
EAN 8	ps_ean8	
Interleaved 2 of 5	ps_interleaved2of5	
ISBN	ps_isbn	
Kix (Dutch Postal)	ps_kix	
MSI	ps_msi	
Plessey	ps_plessey	
Postnet	ps_postnet	
Rationalized Codabar	ps_rationalizedCodabar	
Royal Mail	ps_royalmail	
Symbol	ps_symbol	
UPCA	ps_upca	
UPCE	ps_upce	

11.2.2 GNU Barcode

Encoding	Database Name	Example
Raw code 128	`128raw`	
Codabar	`cbr`	
Codabar (no checksum)	`cbr -c`	
Code 128 (a,b,c: autoselection)	`code128`	
Code 128B, full printable ascii	`code128b`	
Code 128C (compact form digits)	`code128c`	
Code 39 (no checksum)	`code39 -c`	
Code 39	`code39`	
Code 93	`code93`	
EAN (EAN 8 or EAN 13)	`ean`	
interleaved 2 of 5 (only digits, no checksum)	`i25 -c`	
interleaved 2 of 5 (only digits)	`i25`	
ISBN (still EAN13)	`isbn`	
MSI	`msi`	
Plessey	`pls`	
UPC (12-digit EAN; UPCA and UPCB)	`upc`	

11.2.3 PDF417 Encode

Encoding	Database Name	Example
pdf 417 2D Barcode	`pdf417`	

11.2.4 TBarcode

TBarcode encodings are not yet listed here, but will be added in a future version of this handbook.

11.3 GNU General Public License

The **KBarcode** handbook, as well as **KBarcode** itself, are distributed under the terms of the GPL. The GPL can be read online at *http://www.gnu.org/licenses/gpl.html* .

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to

know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU General Public License

Terms and Conditions For Copying, Distribution and Modification

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License

will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide

if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

12. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the

IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) ¡year¿ ¡name of author¿
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

The End