

TBarCode/X

Barcode Generator Software for Linux®, UNIX® and Mac OS® X

Version 11.0

Developer Manual

Oktober 23, 2013

TEC-IT Datenverarbeitung GmbH
Hans-Wagner-Strasse 6
A-4400 Steyr, Austria

t ++43 (0)7252 72720
f ++43 (0)7252 72720 77
office@tec-it.com
www.tec-it.com

1 Content

1		**Content**	**2**
2		**Disclaimer**	**3**
3		**Introduction**	**4**
3.1		What is TBarCode?	4
3.2		What is TBarCode/X?	4
3.3		Scope of this Document	4
3.4		Restrictions of the Demo Version	5
4		**Installation**	**6**
5		**General**	**7**
5.1		TBarCode Library	7
	5.1.1	C/C++ Header Files	7
	5.1.2	Linking	7
5.2		TBarCode Framework (for Mac OS)	8
	5.2.1	C/C++ Header Files	8
	5.2.2	Compiling and Linking	8
5.3		LibTBarCode Java Interface	8
6		**Using TBarCode**	**9**
6.1		Important Functions	9
6.2		Calling Order	10
6.3		ANSI and UNICODE	10
7		**C/C++ Sample Code**	**11**
8		**Custom Drawing Functions for Special Devices**	**12**
8.1		Why Custom Drawing Functions?	**Error! Bookmark not defined.**
8.2		The General Concept	**Error! Bookmark not defined.**
8.3		Linear Barcodes & PDF417	**Error! Bookmark not defined.**
8.4		Data Matrix, QR-Code, Aztec Code, and Micro QR-Code	**Error! Bookmark not defined.**
	8.4.1	About Drawing	**Error! Bookmark not defined.**
9		**How to License TBarCode**	**12**
9.1		Demo Limitations	17
10		**Redistributing TBarCode**	**18**
10.1		TBarCode as a Static Library	18
10.2		TBarCode as a Shared Library	18
10.3		TBarCode as a Framework (Mac OS)	18
11		**Contact and Support Information**	**19**

2 Disclaimer

The actual version of this product (document) is available as is. TEC-IT declines all warranties which go beyond applicable rights. The licensee (or reader) bears all risks that might take place during the use of the system (the documentation). TEC-IT and its contractual partners cannot be penalized for direct and indirect damages or losses (this includes non-restrictive, damages through loss of revenues, constriction in the exercise of business, loss of business information or any kind of commercial loss), which is caused by use or inability to use the product (documentation), although the possibility of such damage was pointed out by TEC-IT.



We reserve all rights to this document and the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.



Für dieses Dokument und den darin dargestellten Gegenstand behalten wir uns alle Rechte vor. Vervielfältigung, Bekanntgabe an Dritte oder Verwendung außerhalb des vereinbarten Zweckes sind nicht gestattet.

© 1998-2013
TEC-IT Datenverarbeitung GmbH
Hans-Wagner-Str. 6

A-4400 Austria
t.: +43 (0)7252 72720
f.: +43 (0)7252 72720 77
http://www.tec-it.com

3 Introduction

3.1 What is TBarCode?

TBarCode is a set of professional tools for the generation of bar codes. More than 100 different symbologies (linear barcodes, 2D barcodes and stacked barcode variants) can be printed or exported as graphics files. All industry formats are supported. The barcodes can be generated in the highest possible resolution and quality.

TBarCode is available in several versions for different operating systems, applications and programming environments.

The following versions are included in the Linux®/UNIX® setup:

TBarCode/X	Barcode software for Linux and UNIX platforms. TBarCode/X includes: ▪ command-line tools, ▪ filter scripts, ▪ shared library ▪ Java interface (on request only)

Additionally the following products for the Microsoft® Windows platform are available:

TBarCode OCX	A Microsoft® ActiveX® compliant barcode control. It can be used with Microsoft® Office applications as well as by software developers.
TBarCode .NET	A .NET barcode library for software developers. It includes barcode controls for Windows Forms and ASP.NET 2.0.
TBarCode Library	A dynamically linked library (DLL) for software developers.

3.2 What is TBarCode/X?

TBarCode/X is a software tool for barcode generation on Linux®/UNIX® and Mac OS®. It consists of a command line tool, filter scripts, and the **TBarCode Library** for UNIX® or the **TBarCode Framework** for Mac OS®. On request a Java interface is also available.

Whereas this document mainly covers the usage of the library, please refer to the **„TBarCode/X User Documentation“** for an in-depth description of the other parts (see chapter *4 Installation*).

The **TBarCode/X** setup for Linux/UNIX includes the **TBarCode Library** as static library and as shared library.

The **TBarCode/X** setup for Mac OS® includes the **TBarCode Framework**.

TBarCode Library for UNIX is also often called “**LibTBarCode**”.

3.3 Scope of this Document

This document explains how you can use the **TBarCode Library** for UNIX® and **TBarCode Framework** for Mac OS® in your own applications. The complete application programming interface (API) is described in the *TBarCode Library Developer Reference*.

3.4 Restrictions of the Demo Version

In the demo version the barcodes will be drawn with a demo-hint. That means that the word “Demo” or the phrase “www.tec-it.com” is drawn partially over the barcode (see Figure). The demo-hint does not influence the readability of the barcode in a negative way.

When barcodes are generated in an image, PostScript®, PDF, or PCL® format, an additional horizontal bar is drawn across the barcode. Like the other demo-hint this bar usually does not influence the readability of the barcode. Its sole purpose is to indicate that the barcodes were generated with a demo version of **TBarCode**.

- In special cases (e.g. very small or high-resolution barcodes) you may want to test the product without restrictions. To obtain a temporary license key contact sales@tec-it.com.
- For enabling the full-featured version (without the demo hints) you can obtain a license key from TEC-IT (http://www.tec-it.com/order/).

3.5 For more information on licensing TBarCode, please refer to chapter 8.1, “Why Custom Drawing Functions?

TBarCode Library offers the possibility to implement custom drawing functions. This is useful whenever you control a device which is not supported by any standard-driver functionality. Good examples are laser marking devices, printer firmware (in combination with source code license), OS-400 specific printers and others.

Custom drawing functions can be registered as so called call-back functions. When drawing a barcode the **TBarCode Library** will call the custom drawing functions instead of using the internal drawing routines.

- **IMPORTANT**: Custom drawing functions will only work if a valid **TBarCode** license is provided! Temporary license keys are available on request – please contact support@tec-it.com. Section 8.1, “**Error! Not a valid bookmark self-reference.**”, describes how to apply a license.

3.6 The General Concept

TBarCode computes a barcode using a so-called meta-description. This meta-description defines in a complete device independent way where bars and where spaces are to be drawn.

Such a meta-description consists of upper- and lowercase letters:

- Uppercase letters are placeholders for bars (or dots)
- Lowercase letters are placeholders for spaces
- The letter itself (A or B or C or …) defines the width of the bar (space) to be drawn.

3.7 Linear Barcodes & PDF417

For barcodes, which are using multiple widths for the bars (or spaces), multiple uppercase (or lowercase) letters are passed to the call-back function:

Uppercase letters = bars:

- A ... bar (actual width = 1 * module width X)
- B ... bar (actual width = 2 * module width X)
- C ... bar (actual width = 3 * module width X)

- D ... bar (actual width = 4 * module width X)
- E ... and so on

The factors for the module width depend on the current print-ratio. In this example the print-ratio for the bars is 1:2:3:4

Lowercase letters = spaces:

- a ... space (actual width = 1 * module width X)
- b ... space (actual width = 2 * module width X)
- c ... space (actual width = 3 * module width X)
- d ... space (actual width = 4 * module width X)
- e ... and so on

The factor for the module widths depend on the current print-ratio. In this example the print-ratio for the bars is 1:2:3:4

X represents the module width. All actual widths of bars or spaces are usually multiples of the module width.

Each barcode symbology uses a pre-defined print-ratio (and this ratio can be adjusted by the user). For example Code39 uses the following print-ratio: 1:3:1:3

- A ... 1 X
- B ... 3 X
- a ... 1 X
- b ... 3 X

It is possible to query the used print-ratio for a specific barcode symbology – please check out the relevant functions *BCGetRatioString, BCGetRatioHint, BCGetCountBars, and BCGetCountSpaces.*

3.8 2D Matrix Codes (Data Matrix, QR-Code, Aztec Code, etc.)

2D matrix codes like Data Matrix, QR-Code, and Aztec Code consist of several rows. The corresponding row patterns are transmitted to a user-defined callback function separately row by row. The callback function is responsible for drawing a single barcode row.

The row pattern consists of uppercase and lowercase letters. Uppercase letters serve as placeholders for black bars (or squares) – lowercase letters are placeholders for spaces (white squares):

Uppercase “A” - black dot/bar, Lowercase “a” - white dot/space

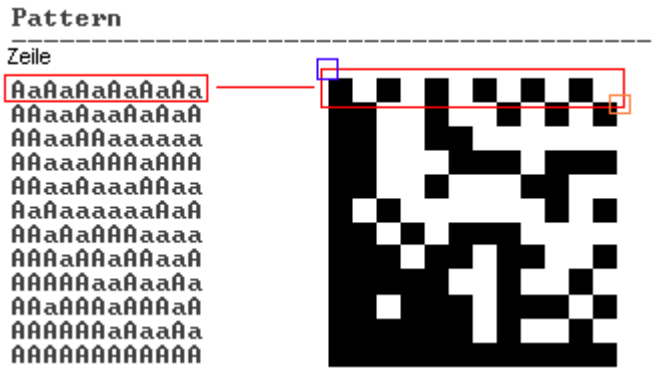


Figure 1: Custom Barcode Drawing

The example above shows Data Matrix, but QR-Code and Aztec Code work in the same manner.

3.8.1 About Drawing

The pattern itself contains no absolute sizes. The matrix dots (A and a) have the same width and height X. This is called the module width. By adjusting the module width to the size of the device dots (pixels) you can minimize the printing tolerances.

3.9 Postal Codes with Bars of Different Height

In contradiction to other linear barcodes many postal codes don't use multiple widths for the bars but multiple heights. Instead of letters, digits are passed to the call-back function.

A single digit is a placeholder for a sequence of one bar and one space. All bars and spaces have the same width of 1X (=module width). The heights of the bars differ.

Depending on the barcode type the pattern string may contain two, three, or four different digits (=heights).

3.9.1 Barcode with 2 different heights

Barcodes with 2 different heights consist of long bars and short bars, both sharing the same base line, growing bottom up.

- 1 ... long bar
- 2 ... short bar



Figure 2: US Postal Code with 2 different heights

3.9.2 Barcodes with 3 different heights

Barcodes with 3 different heights consist of long bars and short bars either growing bottom up or to down, and short bars.

- 0 ... long bar
- 1 ... short bar, growing top down
- 2 ... short bar, growing bottom up



Figure 3: Pharmacy Code Two-Track with 3 different heights

3.9.3 Barcodes with 4 different heights

Barcodes with 4 different heights consist of long bars, medium sized bars either growing bottom up or to down, and short bars, drawn vertically centered.

- 0 ... long bar (**Full**)
- 1 ... medium sized bar, growing top down (**Ascender**)
- 2 ... medium sized bar, growing bottom up (**Descender**)
- 3 ... short bar (**Tracker**)



Figure 4: Australian Postal Code with 4 different heights

3.9.4 About Drawing

The pattern itself contains no absolute sizes. Following table gives you detailed hints how to convert a given pattern to a valid barcode.

Size and position of a bar is defined by the bar's height (in percent of a full height bar) and the bar's distance from the upper edge (also in percent of a full height bar).

Barcode Type(s)	Pattern Digit			
	0	1	2	3
US Postal, CEPNet, Planet	**Height:** 100% **Distance:** 0%	**Height:** 38.5% **Distance:** 61.5%	--	--
Pharmacode 2-Track	**Height:** 100% **Distance:** 0%	**Height:** 50% **Distance:** 50%	**Height:** 50% **Distance:** 0%	--
Australian Postal	**Height:** 100% **Distance:** 0%	**Height:** 62% **Distance:** 0%	**Height:** 62% **Distance:** 38%	**Height:** 24% **Distance:** 38%
Royal Mail 4 State, KIX	**Height:** 100% **Distance:** 0%	**Height:** 62,5% **Distance:** 0%	**Height:** 62,5% **Distance:** 37,5%	**Height:** 25% **Distance:** 37,5%
Intelligent Mail® Barcode, DAFT, Japanese Postal	**Height:** 100% **Distance:** 0%	**Height:** 66,7% **Distance:** 0%	**Height:** 66,7% **Distance:** 33,3%	**Height:** 33,3% **Distance:** 33,3%

Table 1: Drawing Barcodes with Multiple Heights

3.10 Control Patterns

Apart from letters and digits the pattern string may contain control characters. In the following you find a short overview.

3.10.1 Protruding Bars for EAN and UPC Codes

Following patterns specify changes of the bar length

- ASCII (254) ... Begin of section with long (=protruding) bars
- ASCII (255) ... End of section with long (=protruding) bars
- ASCII (253) ... Begin of an add-on section which last until the end of the code

The barcode types **EAN 8/13** and **UPC A/E** contain protruding bars on the begin, in the middle, and on the end of the barcode. With **ASCII (254)** the section with protruding bars starts, after **ASCII (255)** it ends. Bars of "normal" length follow. Protruding bars are extended on the bottom side by about the half height of the human readable text.

Add-on sections start with **ASCII (253)**. They continue until the end of the code. Add-on bars leave space for the text above the barcode and align at the bottom with the protruding bars.



Figure 5: EAN8 with 5 add-on digits

3.10.2 Increment and Decrement the Bar Width for EAN and UPC Codes

- ASCII (252) ... Begin bar width increment
- ASCII (251) ... End bar width increment
- ASCII (250) ... Begin bar width decrement
- ASCII (249) ... End bar width decrement

Based on the definition of EAN and UPC codes the width of some bars has to be increased whereas the width of other bars has to be decreased. This is done by the control patterns shown above.

The amount of bar width increment/decrement is 1/13 of the actual module width. If a bar becomes wider, the following space becomes narrower and vice versa.

Please examine the *EAN specification* for a detailed description.

► How to License TBarCode".

4 Installation

Please refer to the **„TBarCode/X User Documentation“** for an in-depth description.

The document is either included with the setup or (always the latest version) available on the TEC-IT web-site: http://www.tec-it.com ► Download ► TBarCode/X.

5 General

Please keep in mind that **TBarCode Library** is a software component. It is not an executable by its own. Read this document and check out the accompanying sample applications to learn how to embed **TBarCode** into your own application.

5.1 TBarCode Library

TBarCode Library (LibTBarCode) is included in the **TBarCode/X** package. Depending on the operating system **TBarCode/X** is delivered as tar-ball, RPM or another appropriate installation package.

Binaries are available for:

- Linux (x86 + IA64)
- FreeBSD (x86)
- AIX (PowerPC)
- HP-UX (PA-RISC 1.1/2.0 + IA64)
- Sun Solaris (x86 + Sparc)
- SCO OpenServer/UnixWare
- And others.

► If there are no binaries available for your operating system please contact TEC-IT (support@tec-it.com). Most likely TEC-IT is able to compile a suitable binary.

5.1.1 C/C++ Header Files

TBarCode/X is delivered with the required header and library files of LibTBarCode.

Include the file *tbarcode.h* in your project in order to get full access to the shared LIB functions within C/C++:

```
#include <libtbarcode11/tbarcode.h>
```

The file is usually installed at the following location:

```
/usr/local/include/libtbarcode11/tbarcode.h
```

You will have to add the option

```
-I/usr/local/include
```

when calling the preprocessor/compiler, to ensure that the preprocessor/compiler finds the header files.

5.1.2 Linking

TBarCode/X is available as static library or as shared library. A shared library is comparable to a DLL under Windows. Per default it is installed in */usr/local/lib*. You can link against **TBarCode/X** using the linker options

```
-L/usr/local/lib/ -ltbarcode11
```

The foregoing linker options prefer the usage of the shared library (in */usr/local/lib*). If the shared library was not found the static library will be used.

5.2 TBarCode Framework (for Mac OS)

The **TBarCode Framework** is a special version of the **TBarCode Library** for Mac OS. It is included in the **TBarCode/X** installation package for Mac OS.

5.2.1 C/C++ Header Files

The required header files are a substantial ingredient of the **TBarCode Framework**. Include the file *tbarcode.h* in your project in order to get full access to the library functions within C/C++ (the first *TBarCode* stands for the name of the framework):

```
#include <TBarCode/tbarcode.h>
```

The file (and the other needed include files) is installed in the framework bundle which is usually located at following path:

```
/Library/Frameworks/TBarCode.framework
```

5.2.2 Compiling and Linking

If you want to compile and link your application to the **TBarCode Framework**, just add it to your project. A framework is comparable with a DLL under Windows or a shared library under Linux/UNIX, but it not just only a file, but a full-featured bundle that also contains the public header files and the documentation.

5.3 LibTBarCode Java Interface

The Java interface is a software layer between the **TBarCode Library** (LibTBarCode) and the Java Virtual Machine through Java Native Interface (JNI) technology.

The architecture of this layer is composed by:

- The **TBarCode Library** in order to execute the bar code generation.
- The JNI library **TBarCode11_Java** (DLL or shared library), to translate the Java calls to the LibTBarCode API requests and responses.
- The Java library **TBarCode11_Java.jar**, to simplify the access to the bar code generator JNI interface.

It is important that the paths of the **TBarCode Library** and the JNI library are included in the java.library.path of the Java Virtual Machine. To make these paths available they can be

- defined in LD_LIBRARY_PATH (Linux and Macintosh), SHLIB (HPUX) or LIBPATH (AIX),
- passed in the parameter -Djava.library.path of the java command or
- defined as system libraries.

The Java library **TBarCode11_Java.jar** must be included in the classpath of the javac/java command in order to compile and execute your projects.

► The **TBarCode** Java Interface is available on request and is built for your specific platform on demand. Please contact our support for a suitable version for your platform.

6 Using TBarCode

6.1 Important Functions

The basic function calls to produce a barcode are as follows (in the appropriate order).

- *BCLicenseMe()*
  This function licenses **TBarCode** and removes the demo restrictions. Licensing must be performed before you draw a barcode (e.g. after **TBarCode** has been loaded to memory).

- *BCAlloc()*
  This function sets up and initializes the internal barcode structure. You receive a handle that is used for all other function calls (*pBarCode*). This function must be called before any other function expecting a **pBarCode** parameter.

- *BCSetBCType()*
  Sets the type of the barcode (symbology); e. g. *Code39, Code128, UPC, EAN, 2OF5, ...*

- *BCSetText()*
  Sets the data to be encoded as barcode.

- *BCSetModWidth()* (optional)
  This function is used if an application requires a specific module width. Without this function the module width is computed automatically by **TBarCode**. It adapts to the barcode dimensions (specified via a bounding rectangle) and the current input data.

- More optional barcode settings
  Set the barcode properties according to your application; e.g. *BCSet_PDF417_RowHeight(), BCSetCDMethod(), BCSetBearerBarWidth(), BCSetRatio(), BCSetTextDist(), BCSetLogFont(), ...*

- *BCCheck()* (optional)
  This function checks if the data characters are valid for the selected barcode type. If invalid data was encountered it returns an error-code. If escape-sequences are used, they are not translated in this function. It must be called before *BCCalcCD()*.
  Note: This function call is optional; *BCCreate()* calls this function in any case automatically.

- *BCCalcCD()* (optional)
  This function computes the check-digit(s) for the given input data and the selected check-digit method. The check digits are added to the barcode data automatically. On demand you can retrieve the check digits with *BCGetCheckDigits()*. Please consider that symbology internal check digits (like *Modulo 103* of Code-128) are not calculated with this function – they are always part of the created barcode.
  Note: This function call is optional; *BCCreate()* calls this function in any case automatically.

- *BCCreate()*
  This function prepares the barcode structure (pattern) to be drawn with *BCDraw()*. It returns *ErrOk* if everything is ok. If not, it returns an error code (of type *ERRCODE*) that specifies the error in more detail. After *BCCreate()* all parameters of the resulting barcode are available (e.g. number of modules, dimensions, check-digits, meta-description).

- Get Dimensions (optional)
  After *BCCreate()* you can call the methods *BCGetBarcodeHeight(), BCGetBarcodeWidth(), ...*

- *BCDraw()*
  This function draws the barcode into the given device context. The barcode dimensions are set through passing the coordinates of a bounding rectangle. No special mapping is performed.
  Note: Only available in **TBarCode Library for Windows**!

- *BCPostscriptToFile(), BCPCLToFile()*
  These function save the barcode in PostScript or PCL output format.

- *BCFree()*
  This function de-initializes the barcode info-structure and frees allocated memory. It must be called as last function.

► If any of the *BCxxxx* functions in the above described order returns an error code not equal to zero then DO NOT call subsequent *BCxxxx* functions (except of *BCFree()*). An error code <> 0 indicates an error condition - subsequent calls (except of *BCFree()*) may fail and produce unexpected results.

6.2 Calling Order

► Please note: Starting with TBarCode V8 and higher the following calling order must be maintained to guarantee the correct conversion of the input data to the target character set:

1. First set all barcode properties (like barcode type, translation of escape sequences, etc.)
2. Then call *BCSetBCText()*
3. Finally call *BCCreate()*

6.3 ANSI and UNICODE

Since version 8.x the **TBarCode Library** for UNIX provides UNICODE functionality. All functions with parameters or return values of data type string are implemented in 2 ways as **A**NSI and **W**ide String function. The names of ANSI-functions end with 'A' whereas the Wide String-functions end with 'W'. If you want to work with UNICODE you have to use the W-functions.

As UNICODE characters consist of 2 bytes and most of the barcode types are only able to encode one byte per character, it is not always clear how the input data should be interpreted. So we provide 2 ways to control.

- *Encoding Mode*
  The input data can either be converted to a selected code page (see below) or interpreted byte per byte (lower byte only , lower before upper byte, or the other way round).

- *Code Page*
  The user can choose among several pre-defined code pages (e.g. ANSI, ISO 8559-1 Latin I, UTF-8, Shift-JIS…) or add a custom code page.

7 C/C++ Sample Code

Below are the steps to create a barcode image in C/C++ (only for demonstrative purposes, not all variables declared).

> ► Also check out the fully functional samples provided with the setup – or available as separate download.

Include the header file:

```
#include <libtbarcode11/tbarcode.h>
```

Sample code for barcode generation (excerpt):

```
// Initialize library
BCInitLibrary("/usr/local/share/tbarcode11");

// License the product
BCLicenseMe("LicenseeName", eLicKindDeveloper, 1, "MyKey", eLicProd2D);

// Allocate memory and retrieve barcode handle (pointer)
t_BarCode* pBC;
BCAlloc(&pBC);

// (Optional:) Set font type and height for the human readable text
BCSetFontName(pBC, "Helvetica");
BCSetFontHeight(pBC, 10);           // 10 points

// Adjust symbology
BCSetBCType(pBC, eBC Code128);

// Set barcode data
char* demo = "12345678";
BCSetText(pBC, demo, strlen(demo));

// Create barcode pattern (bars, spaces)
BCCreate(pBC);

// Set barcode size (PostScript bounding rectangle)
// Units are [0.001 mm]
rect.left = 0;         // 0 mm
rect.bottom = 0;       // 0 mm
rect.right = 50000;    // 50 mm
rect.top = 30000;      // 30 mm

// Draw to device context
// not supported in Linux/UNIX, because only the Windows GDI uses a "device context"

// Save to Postscript file
BCPostscriptToFile(pBCode, (void *) "barcode.eps", &rect);

// Save barcode image to buffer
// Unit is [0.001mm] for Postscript and PCL
void* pPSBuffer = malloc(0xffff);
BCPostscriptToMemory(pBC, pPSBuffer, 0xffff, &rect);

if (pPSBuffer)
    free(pPSBuffer); // Release allocated memory after use

// Release memory for barcode structure
BCFree(pBC);

// Clean up
BCDeInitLibrary();
```

8 Custom Drawing Functions for Special Devices

8.1 Why Custom Drawing Functions?

TBarCode Library offers the possibility to implement custom drawing functions. This is useful whenever you control a device which is not supported by any standard-driver functionality. Good examples are laser marking devices, printer firmware (in combination with source code license), OS-400 specific printers and others.

Custom drawing functions can be registered as so called call-back functions. When drawing a barcode the **TBarCode Library** will call the custom drawing functions instead of using the internal drawing routines.

► **IMPORTANT**: Custom drawing functions will only work if a valid **TBarCode** license is provided! Temporary license keys are available on request – please contact support@tec-it.com. Section 8.1, "**Error! Not a valid bookmark self-reference.**", describes how to apply a license.

8.2 The General Concept

TBarCode computes a barcode using a so-called meta-description. This meta-description defines in a complete device independent way where bars and where spaces are to be drawn.

Such a meta-description consists of upper- and lowercase letters:

- Uppercase letters are placeholders for bars (or dots)
- Lowercase letters are placeholders for spaces
- The letter itself (A or B or C or …) defines the width of the bar (space) to be drawn.

8.3 Linear Barcodes & PDF417

For barcodes, which are using multiple widths for the bars (or spaces), multiple uppercase (or lowercase) letters are passed to the call-back function:

Uppercase letters = bars:

- A ... bar (actual width = 1 * module width X)
- B ... bar (actual width = 2 * module width X)
- C ... bar (actual width = 3 * module width X)
- D ... bar (actual width = 4 * module width X)
- E ... and so on

The factors for the module width depend on the current print-ratio. In this example the print-ratio for the bars is 1:2:3:4

Lowercase letters = spaces:

- a ... space (actual width = 1 * module width X)
- b ... space (actual width = 2 * module width X)
- c ... space (actual width = 3 * module width X)
- d ... space (actual width = 4 * module width X)
- e ... and so on

The factor for the module widths depend on the current print-ratio. In this example the print-ratio for the bars is 1:2:3:4

X represents the module width. All actual widths of bars or spaces are usually multiples of the module width.

Each barcode symbology uses a pre-defined print-ratio (and this ratio can be adjusted by the user). For example Code39 uses the following print-ratio: 1:3:1:3

- A ... 1 X
- B ... 3 X
- a ... 1 X
- b ... 3 X

It is possible to query the used print-ratio for a specific barcode symbology – please check out the relevant functions *BCGetRatioString, BCGetRatioHint, BCGetCountBars, and BCGetCountSpaces.*

8.4 2D Matrix Codes (Data Matrix, QR-Code, Aztec Code, etc.)

2D matrix codes like Data Matrix, QR-Code, and Aztec Code consist of several rows. The corresponding row patterns are transmitted to a user-defined callback function separately row by row. The callback function is responsible for drawing a single barcode row.

The row pattern consists of uppercase and lowercase letters. Uppercase letters serve as placeholders for black bars (or squares) – lowercase letters are placeholders for spaces (white squares):

Uppercase “A” - black dot/bar, Lowercase “a” - white dot/space

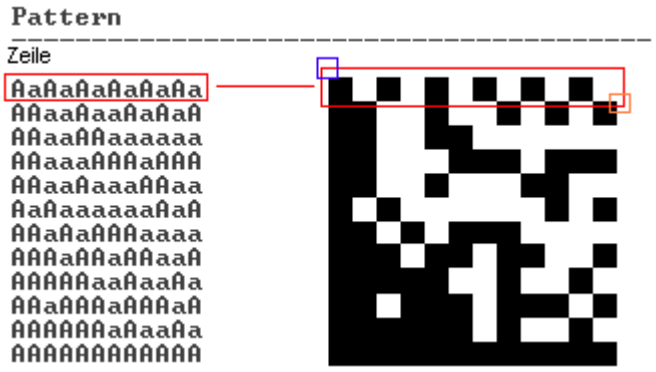


Figure 1: Custom Barcode Drawing

The example above shows Data Matrix, but QR-Code and Aztec Code work in the same manner.

8.4.1 About Drawing

The pattern itself contains no absolute sizes. The matrix dots (A and a) have the same width and height X. This is called the module width. By adjusting the module width to the size of the device dots (pixels) you can minimize the printing tolerances.

8.5 Postal Codes with Bars of Different Height

In contradiction to other linear barcodes many postal codes don’t use multiple widths for the bars but multiple heights. Instead of letters, digits are passed to the call-back function.

A single digit is a placeholder for a sequence of one bar and one space. All bars and spaces have the same width of 1X (=module width). The heights of the bars differ.

Depending on the barcode type the pattern string may contain two, three, or four different digits (=heights).

8.5.1 Barcode with 2 different heights

Barcodes with 2 different heights consist of long bars and short bars, both sharing the same base line, growing bottom up.

- 1 ... long bar
- 2 ... short bar



Figure 2: US Postal Code with 2 different heights

8.5.2 Barcodes with 3 different heights

Barcodes with 3 different heights consist of long bars and short bars either growing bottom up or to down, and short bars.

- 0 ... long bar
- 1 ... short bar, growing top down
- 2 ... short bar, growing bottom up



Figure 3: Pharmacy Code Two-Track with 3 different heights

8.5.3 Barcodes with 4 different heights

Barcodes with 4 different heights consist of long bars, medium sized bars either growing bottom up or to down, and short bars, drawn vertically centered.

- 0 ... long bar (**Full**)
- 1 ... medium sized bar, growing top down (**Ascender**)
- 2 ... medium sized bar, growing bottom up (**Descender**)
- 3 ... short bar (**Tracker**)



Figure 4: Australian Postal Code with 4 different heights

8.5.4 About Drawing

The pattern itself contains no absolute sizes. Following table gives you detailed hints how to convert a given pattern to a valid barcode.

Size and position of a bar is defined by the bar's height (in percent of a full height bar) and the bar's distance from the upper edge (also in percent of a full height bar).

Barcode Type(s)	Pattern Digit			
	0	1	2	3
US Postal, CEPNet, Planet	**Height:** 100% **Distance:** 0%	**Height:** 38.5% **Distance:** 61.5%	--	--
Pharmacode 2-Track	**Height:** 100% **Distance:** 0%	**Height:** 50% **Distance:** 50%	**Height:** 50% **Distance:** 0%	--
Australian Postal	**Height:** 100% **Distance:** 0%	**Height:** 62% **Distance:** 0%	**Height:** 62% **Distance:** 38%	**Height:** 24% **Distance:** 38%
Royal Mail 4 State, KIX	**Height:** 100% **Distance:** 0%	**Height:** 62,5% **Distance:** 0%	**Height:** 62,5% **Distance:** 37,5%	**Height:** 25% **Distance:** 37,5%
Intelligent Mail® Barcode, DAFT, Japanese Postal	**Height:** 100% **Distance:** 0%	**Height:** 66,7% **Distance:** 0%	**Height:** 66,7% **Distance:** 33,3%	**Height:** 33,3% **Distance:** 33,3%

Table 1: Drawing Barcodes with Multiple Heights

8.6 Control Patterns

Apart from letters and digits the pattern string may contain control characters. In the following you find a short overview.

8.6.1 Protruding Bars for EAN and UPC Codes

Following patterns specify changes of the bar length

- ASCII (254) ... Begin of section with long (=protruding) bars
- ASCII (255) ... End of section with long (=protruding) bars
- ASCII (253) ... Begin of an add-on section which last until the end of the code

The barcode types **EAN 8/13** and **UPC A/E** contain protruding bars on the begin, in the middle, and on the end of the barcode. With **ASCII (254)** the section with protruding bars starts, after **ASCII (255)** it ends. Bars of "normal" length follow. Protruding bars are extended on the bottom side by about the half height of the human readable text.

Add-on sections start with **ASCII (253)**. They continue until the end of the code. Add-on bars leave space for the text above the barcode and align at the bottom with the protruding bars.



Figure 5: EAN8 with 5 add-on digits

8.6.2 Increment and Decrement the Bar Width for EAN and UPC Codes

- ASCII (252) ... Begin bar width increment
- ASCII (251) ... End bar width increment
- ASCII (250) ... Begin bar width decrement
- ASCII (249) ... End bar width decrement

Based on the definition of EAN and UPC codes the width of some bars has to be increased whereas the width of other bars has to be decreased. This is done by the control patterns shown above.

The amount of bar width increment/decrement is 1/13 of the actual module width. If a bar becomes wider, the following space becomes narrower and vice versa.

Please examine the *EAN specification* for a detailed description.

9 How to License TBarCode

In order to enable the full-featured version, you need a valid license key from TEC-IT. A description of the available license-types as well as all necessary information for ordering can be found at http://www.tec-it.com/prices.

If you don't know the license type according to your application, please ask our sales team (sales@tec-it.com).

For placing an online order check out http://www.tec-it.com/order/.

► For testing the call-back API or other evaluation purposes you can request a time-limited license key from support@tec-it.com.

9.1 Demo Limitations

Whenever **TBarCode** is not licensed with a valid license key, an additional text “Demo” or an additional horizontal bar is drawn across the barcode. In addition all custom drawing call-back functions are disabled.

To remove the demo limitations call *BCLicenseMe()* with valid a license key.

Here is an example for programmatic licensing:

```
ERRCODE eCode = BCLicenseMe("John Smith", eLicKindSite, 1,
                            "01234567890ABCDEFGHIJKLMNOPQRSTU", eLicProd2D);
```

In Windows: *BCLicenseMe()* should be called as the first function of **TBarCode Library**.

In UNIX: *BCLicenseMe()* should be called directly after *BCInitLibrary()*.



*Figure 6: Barcodes rendered **without** valid license*



Figure 7: Barcodes rendered with valid license

10 Redistributing TBarCode

This chapter explains what is important when redistributing a custom application that uses the **TBarCode Library**.

> ► Please note that in most cases you need a developer license for re-distribution of **TBarCode Library** (except for in-house applications which are bound to one or more sites).

As a developer you can choose whether you link **TBarCode** as static library or as shared object.

10.1 TBarCode as a Static Library

The static library can be found at the following location:

```
/usr/local/lib/libtbarcode11.a
```

When you link against the static library, then you do not have to redistribute anything else, except your own application.

10.2 TBarCode as a Shared Library

The shared library consists of the following files:

```
/usr/local/lib/libtbarcode11.so
/usr/local/lib/libtbarcode11.so.0
/usr/local/lib/libtbarcode11.so.0.0.0
```

libtbarcode11.so and *libtbarcode11.so.0* are symbolic links to *libtbarcode11.so.0.0.0*. The version numbers might be different on your system – depending on the type of operating system and the actual version of **TBarCode**. You can find the right files by running

```
ls -l /usr/local/lib/libtbarcode11.so*
```

When you link you application against the shared library, then you will have to redistribute these files (including the symbolic links) with your application.

10.3 TBarCode as a Framework (Mac OS)

The framework can be found at the following location:

```
/Library/Frameworks/TBarCode.framework
```

When you link against the framework, then you will have to redistribute the framework directory. Just copy it to the location shown above.

11 Contact and Support Information

TEC-IT Datenverarbeitung GmbH

Address: Hans-Wagner-Str. 6
AT-4400 Steyr
Austria/Europe
Phone: +43 / (0)7252 / 72 72 0
Fax: +43 / (0)7252 / 72 72 0 – 77
Email: office@tec-it.com
Web: http://www.tec-it.com

AIX® is a registered trademark of IBM Corporation.
HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C, World Wide Web Consortium, Laboratory for Computer Science NE43-358, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139.
JAVA® is a registered trademark of Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA.
JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
Linux® is a registered trademark of Linus Torvalds in several countries.
Microsoft®, Windows®, Microsoft Word®, Microsoft Excel® are registered trademarks of Microsoft Corporation.
Navision is a registered trademark of Microsoft Business Solutions ApS in the United States and/or other countries.
Oracle® is a registered trademark of Oracle Corporation.
PCL® is a registered trademark of the Hewlett-Packard Company.
PostScript® is a registered trademark of Adobe Systems Inc.
SAP, SAP Logo, R/2, R/3, ABAP, and SAPscript are trademarks or registered trademarks of SAP AG in Germany (and in several other countries).
UNIX® is a registered trademark of The Open Group

All other products mentioned are trademarks or registered trademarks of their respective companies. If any trademark on our web site or in this document is not marked as trademark (or registered trademark), we ask you to send us a short message (office@tec-it.com).