



TEC-IT

WWW.TEC-IT.COM

TBarCode Library

Barcode Generation Library

Version 8.0

Developer Manual

3 December 2007

TEC-IT Datenverarbeitung GmbH
Wagnerstrasse 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 Library?	4
3.3	Scope of this Document	4
3.4	Restrictions of the Demo Version	5
4	Installation	6
4.1	TBarCode for Windows	6
4.1.1	Requirements	6
4.1.2	Download and Setup	6
4.2	TBarCode/X for UNIX and Linux	6
5	General	7
5.1	TBarCode Library for Windows	7
5.1.1	Additional Dependencies	7
5.1.2	C/C++ Header Files	7
5.1.3	Linking	7
5.2	TBarCode Library for UNIX and Linux	7
5.2.1	C/C++ Header Files	8
5.2.2	Linking	8
6	Using TBarCode	9
6.1	Important Functions	9
6.2	ANSI and UNICODE	10
7	C/C++ Sample Code	11
7.1	C/C++ Sample Code for Windows	11
7.2	C/C++ Sample Code for Unix/Linux	12
8	Custom Drawing Functions for Special Devices	14
8.1	Why Custom Drawing Functions?	14
8.2	The General Concept	14
8.3	Linear Barcodes & PDF417	14
8.4	Data Matrix & QR-Code	15
8.4.1	About Drawing	15
9	How to License TBarCode	16
9.1	Demo Limitations	16
10	Redistributing TBarCode	17
10.1	Redistributing TBarCode Library for Windows	17
10.1.1	Dependencies	17
10.1.2	Redistribution	17
10.2	Redistributing TBarCode Library for UNIX	18
10.2.1	TBarCode as Static Library	18
10.2.2	TBarCode as Shared Library	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.

© 1998-2007
TEC-IT Datenverarbeitung GmbH
Wagnerstr. 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 barcodes. More than 70 different symbologies (linear as well as 2D or stacked barcode variants) can be printed or exported as graphics file. All industry formats, in the highest possible resolution and quality, are generated.

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

The following versions are included in the Windows setup:

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.

Additionally the following versions are available as separate downloads:

TBarCode/X	<p>TBarCode software for Linux® and UNIX® platforms. TBarCode/X includes:</p> <ul style="list-style-type: none"> ▪ command-line tools, ▪ filter scripts, and ▪ TBarCode Library for UNIX®.
-------------------	--

3.2 What is TBarCode Library?

TBarCode Library is a software library for barcode generation. It is the core of all **TBarCode** products. The library is available for software developers to integrate barcode generation in custom applications. TEC-IT provides the **TBarCode Library** for a wide range of platforms:

- 32-bit Microsoft® Windows® platforms
- 64-bit Microsoft® Windows® platforms
- Microsoft® Windows® CE / Windows® Mobile platforms (available on request – please contact us with your platform details)
- Most Linux® and UNIX® platforms

The **TBarCode** setup for Windows® includes **TBarCode Library** as dynamically linked library (DLL). A static library is available on request.

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

TBarCode Library for Windows is often just called “**TBarCode DLL**” and the UNIX version is often called “**LibTBarCode**”.

3.3 Scope of this Document

This document explains how you can use the **TBarCode Library** 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 PostScript® 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/>).
- ▶ For more information on licensing **TBarCode**, please refer to chapter 9, “How to License TBarCode”.

4 Installation

4.1 TBarCode for Windows

4.1.1 Requirements

The following operating systems are supported:

- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows 98 Second Edition
- Microsoft Windows ME
- Microsoft Windows 2000
- Microsoft Windows XP
- Microsoft Windows Server 2003
- Microsoft Windows Vista (RC2)

4.1.2 Download and Setup

Download **TBarCode** from <http://www.tec-it.com/Download> ► *Software* ► *Barcode Tools for Microsoft Windows* and execute the setup application. Depending on your selection one or more of the following components are installed – along with documentations and sample applications:

- **TBarCode OCX**
A barcode ActiveX control for Microsoft Office users and for software developers. User manual and API reference are included in the setup.
- **TBarCode Library**
For software developers. Developer manual and API reference are included in the setup.
- **TBarCode .NET**
For .NET developers. Developer manual and API reference are included in the setup.

► Administrative rights are required to install **TBarCode**.

4.2 TBarCode/X for UNIX and Linux

Please refer to the „**TBarCode/X User Documentation**“ for an in-depth description. The document is available on the TEC-IT web-site: <http://www.tec-it.com> ► *Download* ► *Barcode Software for UNIX and Linux* ► *Manuals*.

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 for Windows

The dynamically linked library is named *TBarCode8.dll* and can be found in the directory *Bin* of the installation directory (usually *C:\Program Files\TEC-IT\TBarCode8*).

- ▶ Make sure that the **TBarCode Library** (*TBarCode8.dll*) is located in a path included in the '*PATH*' environment variable (or in the same directory as your application).

5.1.1 Additional Dependencies

TBarCode Library further uses the DLL *zlib1.dll*. This DLL file is optional: *zlib1.dll* is only required if compression of barcode data is applied.

If *zlib1.dll* is missing at runtime the **TBarCode Library** is still working, but data compression is disabled.

- ▶ Make sure that *zlib1.dll* is located in the same directory as *TBarCode8.dll*.
- ▶ *zlib1.dll* is only provided for 32-bit Windows platforms.

5.1.2 C/C++ Header Files

The DLL is delivered with C/C++ header files needed for embedding the DLL (the library) into your own application. These files are located in the folder *C:\Program Files\TEC-IT\TBarCode8\Include* (or in the folder you have chosen during setup).

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

```
#include "tbarcode.h"
```

Header files for miscellaneous programming environments (for example Delphi) are included in the appropriate source code samples. Several samples are installed with **TBarCode**; more samples are available on TEC-IT's web site <http://www.tec-it.com>.

5.1.3 Linking

TEC-IT provides the file *TBarCode8.lib*, which contains the entry points to the DLL interface. Link your application using this file. This file is located in the folder *C:\Program Files\TEC-IT\TBarCode8\Lib* (or in the folder you have chosen during setup).

5.2 TBarCode Library for UNIX and Linux

TBarCode Library for Linux and UNIX 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.

5.2.1 C/C++ Header Files

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

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

```
#include <libTBarCode8/tbarcode.h>
```

The file is usually installed at the following location:

```
/usr/local/include/libTBarCode8/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.

5.2.2 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/ -lTBarCode8
```

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.

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 ANSI and UNICODE

Since version 8 of the **TBarCode Library** each function that has parameters or return values of type string (=character pointer) is implemented in 2 versions, one for ANSI strings and one for UNICODE strings. ANSI functions end with the suffix A whereas UNICODE functions end with W.

Preprocessor defines are provided to be compatible to ANSI and UNICODE builds. These defines do not have a suffix.

Example:

```
// ANSI function - pass an ANSI text
BCSetTextA(t_BarCode* pBarCode, LPCSTR szText, LONG nLen)

// UNICODE function - pass a UNICODE/wide character text
BCSetTextW(t_BarCode* pBarCode, LPCWSTR szText, LONG nLen)

// Compatibility Define - pass a UNICODE/wide character text
// text or an ANSI depending on whether _UNICODE is set or not
BCSetText (...)
```

To use the compatibility defines for the function names (like *BCSetText()*) while creating a UNICODE build define the preprocessor variable `_UNICODE` before including `tbarcode.h`:

```
#define _UNICODE
#include "tbarcode.h"
```

If `_UNICODE` is undefined, the compatibility defines refer to the ANSI functions.

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.

7.1 C/C++ Sample Code for Windows

Include the header-file:

```
#include "tbarcode.h"
```

Sample code for barcode generation (excerpt):

```
// Initialize library
// nothing to do for Windows (except when using TBarcode as static library)

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

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

// Adjust symbology
BCSetBCType(pBC, eBC_Code128);

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

// Set font height for the human readable text
LOGFONTA* pLF = BCGetLogFont(pBC);
pLF->lfHeight = 14;

// Find out wrong characters (check if data can be encoded) (optional)
eCode = BCCheck(pBC)
if (eCode != ErrOk)
{
    // your error handling
}

// Calculate check-digits (optional)
BCCalcCD(pBC);

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

// Set barcode size
// Below we use Hi-Metric [0.01 mm] for hDC
rect.left = 0;           // 0 mm
rect.bottom = 0;         // 0 mm
rect.right = 5000;       // 50 mm
rect.top = 3000;         // 30 mm

// Draw to device context
// Hi-Metric [0.01 mm] for hDC
SetMapMode(hPrinterDC, MM_HIMETRIC);
OffsetRect(&rect, Xpos, Ypos); // position
BCDraw(pBC, hPrinterDC, &rect);

// Save to Postscript file
// Unit is [0.001 mm] for Vector-EPS format
BCSaveImage(pBC, "barcode.eps", eIMPsVector, 50000, 30000, 0, 0);
```

```
// Save barcode image to buffer
// Unit is [Pixel] for JPG format
nWidthPx = 2 * BCGetCountModules (pBC);
BCSaveImageToBuffer(pBC, &lpszBuffer, eIMJpg, nWidthPx, 100, 96, 96);

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

// Clean up
// nothing to do for Windows (except when using TBarCode as static library)
```

7.2 C/C++ Sample Code for Unix/Linux

Include the header file:

```
#include <libTBarCode8/tbarcode.h>
```

Sample code for barcode generation (excerpt):

```
// Initialize library
BCInitLibrary("/usr/local/share/TBarCode8");

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

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

// Adjust symbology
BCSetBCType(pBC, eBC_Code128);

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

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

// Find out wrong characters (check if data can be encoded)
eCode = BCCheck(pBC)
if (eCode != ErrOk)
{
    // your error handling
}

// Calculate check-digits
BCCalcCD(pBC);

// 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(pBC, (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, OS-400 specific printers, ...

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 9, “How to License TBarCode”, 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*, *BCGetCountSpaces*.

8.4 Data Matrix & QR-Code

Data Matrix and QR-Code consist of several rows. For each row, the corresponding row pattern will be transmitted to a user-defined call-back function (for drawing a row of the symbol).

The row-pattern is built by uppercase and lowercase letters. Uppercase letters are place-holders for black bars (or squares) – lowercase letters are placeholders for spaces (white squares):

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

Pattern

Zelle

```

AaAaAaAaAaAa
AAaaaAaaAaAa
AAaaAaaaaaa
AAaaaAAaAA
AAaaAaaaAaa
AaAaaaaaaAa
AAaAaAAaaaa
AAaAaAaAaaA
AAAAAaaAaaAa
AAaAAaAaAAa
AAAAAAaAaaAa
AAAAAAAAAAAA

```

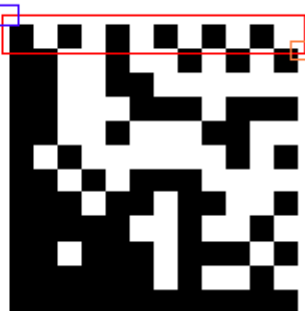


Figure 1: Custom Barcode Drawing

The example above shows Data Matrix, but QR-Code works the same.

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.

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 call-back functions (for custom barcode drawing functions) are disabled.

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

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

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

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



Figure 2: Barcodes rendered **without** valid license.



Figure 3: Barcode 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).

10.1 Redistributing TBarCode Library for Windows

10.1.1 Dependencies

An application that uses **TBarCode** requires the following files:

- **TBarCode8.dll**
This is a native Win32 library that performs barcode generation. This file is mandatory.
- **zlib1.dll**
This is a 3rd-party library that performs data compression. This file is optional. It is only required, when data compression in barcodes is used.

These files are located in the folder *C:\Program Files\TEC-IT\TBarCode8\Bin* (or in the folder you have chosen during setup).

TBarCode Library further requires the **Microsoft VC80 Common Runtime DLLs**. See next section how to distribute them with your application.

10.1.2 Redistribution

When redistributing a custom application the files described above need to be redistributed together with the application. The DLLs should be located in the same folder as the executable.

Other files than those listed above must not be redistributed.

You may have to redistribute the **Visual C++ 2005 SP1 runtime components** (MS CRT8.0 DLLs) with your application and ensure they are installed on the target computer.

There are two options to install them:

- You can install these components with the *Microsoft Visual C++ 2005 SP1 Redistributable Package (x86)* available at <http://www.microsoft.com/downloads/details.aspx?familyid=200b2fd9-ae1a-4a14-984d-389c36f85647>
The package installs all required runtime DLLs.
- If you use a setup tool and your installer supports merge modules (*.msm files) you can add the required msm packages to your setup:
 - *Microsoft_VC80_CRT_x86.msm*
 - *policy_8_0_Microsoft_VC80_CRT_x86.msm*

Please contact TEC-IT Support if you need help.

10.2 Redistributing TBarCode Library for UNIX

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

10.2.1 TBarCode as Static Library

The static library can be found at the following location:

```
/usr/local/lib/libTBarCode8.a
```

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

10.2.2 TBarCode as Shared Library

The shared library consists of the following files:

```
/usr/local/lib/libTBarCode8.so  
/usr/local/lib/libTBarCode8.so.0  
/usr/local/lib/libTBarCode8.so.0.0.0
```

libTBarCode8.so and *libTBarCode8.so.0* are symbolic links to *libTBarCode8.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/libTBarCode8.so*
```

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

11 Contact and Support Information

TEC-IT Datenverarbeitung GmbH

Address: Wagnerstr. 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, 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).