TEC-IT

WWW.TEC-IT.COM

TBarCode/X

Barcode Solution for Linux®, UNIX® and Mac OS X®

Version 9

User Documentation

September 3, 2010

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
1.1 Table of Figures 6
1.2 List of Tables 7

2 Disclaimer 9

3 About TBarCode/X 10
3.1 Features 10
3.1.1 TBarCode/X 10
3.1.2 2D Symbologies 10
3.1.3 Barcode Quality 10
3.2 Usage 10
3.3 System Requirements 11
3.3.1 Supported Platforms 11
3.3.2 Supported Output Devices 11
3.4 Functional Restriction of the Demo Version 11
3.5 Version History 11
3.5.1 TBarCode/X Version History 11
3.5.1.1 What's new in V9 11
3.5.1.2 What's new in V8 12
3.5.1.3 What's new in V7 12

4 Overview 13
4.1 The TBarCode/X Technology 13
4.1.1 TBarCode/X Command Line Tool 14
4.1.1.1 Create Barcodes on Command Line 14
4.1.1.2 Using TBarCode/X to Process Data Streams 14
4.1.2 TBarCode/X Library 14
4.1.2.1 TBarCode/X Library Java Interface 14
4.1.3 TBarCode/X Daemon 15
4.2 About this Manual 15

5 Installation 16
5.1 Install TBarCode/X from a BIN Package 16
5.1.1 Common Problems 16
5.2 Install TBarCode/X from an RPM Package 16
5.2.1 Debian, Ubuntu 17
5.2.2 Common Problems 17
5.2.3 Remove TBarCode/X 17
5.3 Install TBarCode/X from a TAR-GZ Package 17
5.3.1 Prerequisites 17
5.3.1.1 Prerequisites for AIX 18
5.3.2 Installation procedure: 18
5.3.2.1 Installation from tar/gz files on AIX 18
5.3.3 Missing /usr/local directory 19
5.3.4 Common Problems 19
5.3.5 Uninstall TBarCode/X 19
5.4 Install TBarCode/X on SCO® Operating Systems 19
5.4.1 Remove TBarCode/X 20
5.5 Install TBarCode/X on Mac OS® 20
5.5.1 Remove TBarCode/X 20
5.6 File Permissions 20
5.6.1 TBarCode/X with Daemon 20
5.6.2 TBarCode/X without Daemon 21
5.7 SAP® R/3® and mySAP® Integration 21

6 Testing TBarCode/X 22
6.1 Run TBarCode/X from Command Line 22
6.1.1 Run the TBarCode Command 22
6.1.2 Run TBarCode as Filter 22
6.2 Demo License Restriction 22
6.3 TBarCode/X isn't Working? 22

7 Using TBarCode/X 23
7.1 Create a Barcode 23
7.1.1 Create a Barcode in EPS (PostScript®) Format 23
7.1.2 Create a Barcode in PCL®-5 (HP-GL/2®) Format 23
7.1.3 Create a Barcode in PDF (Portable Document) Format 23
7.1.4 Create a Barcode in Bitmap Format 23
7.2 Filter a Print Job or Document File 24
7.2.1.1 Control Sequence Structure 24

7.2.2 Insert a Barcode into a PostScript® Document 24
7.2.3 Insert a Barcode into a PCL® Document 25
7.3 TBarCode/X Command Line Tool 25
7.3.1 Usage 25
7.4 Options 26
7.4.1 General Options 26
7.4.2 Filter Options 27
7.4.3 Compatibility Options (V1 Format) 28
7.4.4 Error Messages and Debug Options 28
7.4.5 Informative Output 29
7.5 Barcode Settings 29
7.5.1 Barcode Type, Barcode Data 29
7.5.2 Output Format 31
7.5.3 Barcode Size and Drawing Position 32
7.5.4 Appearance (Quiet Zone, Print Ratio…) 34
7.5.5 Quality Enhancement 35
7.5.6 Text Settings 36
7.5.7 Filter Settings 37
7.5.8 PDF417 Settings 37
7.5.9 Micro PDF417 Settings 38
7.5.10 Macro PDF417 Settings 38
7.5.11 Data Matrix Settings 38
7.5.12 MaxiCode Settings 39
7.5.13 QR-Code Settings 39
7.5.14 Micro QR-Code Settings 39
7.5.15 Codablock-F Settings 40
7.5.16 Aztec Code Settings 40
7.5.17 RSS Expanded Stacked Settings 41
7.5.18 Composite Barcode Settings 41
7.5.19 Multiple Barcodes 41
7.5.20 Deprecated Barcode Settings 43
7.6 TBarCode/X Configuration Files 43
7.6.1 Path of Configuration Files 43
7.6.2 Syntax of a Configuration File 44
7.6.2.1 Options and Barcode Settings 44
7.6.2.2 Comments 44
7.6.3 tbarcode.conf 44
7.6.4 tbarcoded.conf 44
7.6.5 Priority of Options and Barcode Settings 44

8 TBarCode/X as Spool Filter 45
8.1 LPRng Printing System 45
8.1.1 Testing the Printer Filter 46
8.2 CUPS Printing System 47
8.2.1 Setting up TBarCode/X Spool Filter for PostScript Output 47
8.2.2 Setting up TBarCode/X Spool Filter for PCL Output 47
8.3 AIX's Printing System 49
8.4 HP-UX's Printing System 49
8.4.1 Spool System 49
8.4.2 Using a Local Printer 49
8.4.3 Using a Remote Printer 50
8.4.4 Printing Script HP-UX 11.00 or HP-UX 11.23 50
8.4.5 Printing Script HP-UX 11.11 50
8.4.6 Other Printing Scripts 51
8.4.7 Make a Test Print 51
8.5 Solaris Printing System 51
8.5.1 Spool System Integration 51
8.5.1.1 Register Filter (Setup) 51
8.5.1.2 Create Virtual Printer 52
8.5.1.3 Print To Filtered Printer 52
8.5.2 Print Barcode Filter Test File 52
8.5.2.1 Background Information Solaris Printing 52
8.6 TBarCode/X with UNISPOOL® (Holland House B.V.) 52
8.7 SAP® R/3® and mySAP® Integration 53

9 Generating Bitmap Images 54
9.1 Direct Method: Create Bitmap Images with TBarCode/X 54
9.1.1 Samples 54
9.2 Indirect Method: Convert PostScript Output to Bitmap 55
9.3 Web Applications (PHP) 56
9.3.1 Display a Barcode in a Browser 56
9.3.1.1 Example #1 56
9.3.1.2 Example #2 56
9.3.2 Hints for using `shell_execute()` 56

10 Licensing 58
10.1 License Key and License Types 58
10.2 License File 58

11 Contact and Support Information 59

Appendix A Library Dependencies 60
A.1 Dependencies 60
A.1.1 List Dynamic Dependencies 60
A.1.2 GCC Runtime Libraries 60
A.1.2.1 GCC for Linux 60
A.1.2.2 GCC for AIX 61
A.1.2.3 GCC for HP UX 61
A.1.3 ICONV Libraries 61
A.1.3.1 Iconv for AIX 62
A.1.3.2 Iconv for HP-UX 62
A.2 Shared Library Path 62
A.2.1 Background 63
A.2.2 Linux 63
A.2.2.1 LD_LIBRARY_PATH 63
A.2.2.2 Not Finding "libtbarcode" on Debian 4 64
A.2.3 HP UX 64
A.2.3.1 Enable/Disable Search Path 64
A.2.3.2 SHLIB Path Being Ignored 64
A.2.4 AIX 64
A.2.4.1 GCC Lib Conflicts 65

Appendix B : Troubleshooting (FAQ) 66
B.1 General Questions 66
B.1.1 Can I use the old parameter format as it was used in *TBarCode for Linux Version 1.x?* 66
B.1.2 I have troubles with "`convert`" (gray bars inside the barcode). 66
B.1.3 How can I encode an XML string with the TBarCode Command? 66
B.1.4 How to license the product? 66
B.1.5 How can I retrieve the hostname for buying a single license? 66
B.1.6 TBarCode/X reports that a shared library is missing! 67
B.1.7 Where can I read `syslog` messages? 67
B.1.8 Why is a horizontal bar drawn across the barcodes? 67
B.2 Questions about Filtering/Printing 67
B.2.1 CUPS: How to tell which filters are in place (and maybe failing?) or missing? 67
B.2.2 How can I filter ASCII files? 67
B.2.3 Why is there no barcode when I'm testing the TBarCode/X with LPRng? 68
B.2.4 How to replace printer specific control sequences with TBarCode control sequences? 68
B.2.5 How can I filter Easybar control sequences? 68
B.2.6 How can I print barcodes within a text file? 68
B.2.7 How can I send a file without modification to a printer? 69
B.2.8 LPRng Spool System: How can I find out what data the printer gets from the queue/spooler? 69
B.3 Where I can get more help? 69

Appendix C : Barcode Parameters 70
C.1 Barcode Symbologies 70
C.2 Check Digit Methods 73
C.3 PDF417 Parameters 74
C.3.1 Encoding Mode 74
C.4 Micro PDF417 Parameters 74
C.4.1 Version (Symbol Sizes) 74
C.4.2 Mode 75
C.5 Data Matrix Parameters 75
C.5.1 Symbol Sizes 75
C.5.2 Format 75
C.6 MaxiCode Parameters 76
C.6.1 Mode 76
C.7 QR-Code Parameters 76
C.7.1 Version (Symbol Sizes) 76
C.7.2 Format 77
C.7.3 Error Correction Level 77
C.8 Micro QR-Code Parameters 77
C.8.1 Version (Symbol Sizes) 77
C.8.2 Error Correction Level 77
C.9 Codablock-F Parameters 78
C.9.1 Format 78
C.10 Aztec Code Parameters 78
C.10.1 Symbol Sizes 78
C.10.2 Format 78
C.11 Encoding Bytes and Control Characters in Input Data 79
C.11.1 Implemented Escape Sequences 79

C.11.2 Encoding Bytes 79
C.11.3 Symbology Specific Control Characters 79
C.12 Formatting Barcode Data 80
C.13 PCL Font Numbers 81

Appendix D : Using Version 1.x Format 82
D.1 Overview V1 Format 82

Appendix E : TBarCode Daemon 84
E.1 Usage 84
E.2 Options 84
E.2.1 General Options 84
E.2.2 Daemon and IPC Options 84
E.3 Error Message and Debug Options 85
E.3.1 Informative Output 85

Appendix F : ASCII Table 86

Appendix G : Knowledge Base 87
G.1 Unix Printing (HP-UX and Solaris) 87
G.1.1 SVR4 Spooling System 87
G.1.2 Interface Programs (BSD and SVR4) 87
G.1.3 Printer Interface Scripts (HP-UX) 87
G.1.4 Links 88

1.1 Table of Figures

Figure 1: TBarCode/X with Daemon as Background Server Process 13
Figure 2: TBarCode/X without Daemon 13
Figure 3: Multiple Data Matrix Barcodes (1) 41
Figure 4: Multiple Data Matrix Barcodes (2) 42
Figure 5: Printing with TBarCode/X 45
Figure 6: HP-UX Printer Models/Interfaces 50

1.2 List of Tables

Table 1: General Options 27
Table 2: Filter Options 28
Table 3: Compatibility Options 28
Table 4: Error Message and Debug Options 29
Table 5: Informative Output 29
Table 6: General Barcode Settings 31
Table 7: Output Format Settings 32
Table 8: Barcode Size and Position 34
Table 9: Barcode Appearance Options 35
Table 10: Barcode Quality Options 36
Table 11: Barcode Text Options 36
Table 12: Filtering Options 37
Table 13: PDF417 Options 38
Table 14: Micro PDF417 Options 38
Table 15: Macro PDF417 Options 38
Table 16: Data Matrix Options 38
Table 17: MaxiCode Options 39
Table 18: QR-Code Options 39
Table 19: Micro QR-Code Options 40
Table 20: Codablock-F Options 40
Table 21: Aztec Code Options 40
Table 22: RSS Expanded Stacked Options 41
Table 23: Composite Barcode Options 41
Table 24: Multiple Barcodes Options 43
Table 25: Deprecated Options 43
Table 26: TBarCode/X Configuration Files 43
Table 27: Barcode Symbologies 73
Table 28: Check Digit Methods and Enumerators 74
Table 29: PDF417 Encoding Mode 74
Table 30: Micro PDF417 Symbol Sizes 74
Table 31: Micro PDF417 Modes 75
Table 32: Data Matrix Symbol Sizes 75
Table 33: Data Matrix Formats 75
Table 34: MaxiCode Modes 76
Table 35: QR-Code Symbol Sizes 76
Table 36: QR-Code Format Options 77
Table 37: QR-Code Error Correction Levels 77
Table 38: Micro QR-Code Symbol Sizes 77
Table 39: QR-Code Error Correction Levels 77
Table 40: Codablock-F Parameters 78
Table 41: Aztec Code Symbol Sizes 78
Table 42: Aztec Code Format Options 78

Table 43: Implemented Escape Sequences 79
Table 44: Extended Escape Sequences 80
Table 45: Format Placeholders 80
Table 46: Format Examples 81
Table 47: PCL Font Numbers 81
Table 48: Overview Parameter Syntax of Version 1.x 83
Table 49: TBarCode Daemon – General Options 84
Table 50: TBarCode Daemon – Daemon and IPC Options 85
Table 51: TBarCode Daemon – Error Message and Debug Options 85
Table 52: TBarCode Daemon – Informative Output 85
Table 53: ASCII Table 86

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 partner 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.

© 1998-2008
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 About TBarCode/X

3.1 Features

3.1.1 TBarCode/X

- reduces the costs for barcode printing.
- makes it possible to print barcodes on any **PCL®** or **PostScript®** compatible printer
- does not require costly barcode extension cartridges or special barcode fonts. Thus you can print barcodes in a complete device independent way.
- works in a completely transparent way.
- is available as precompiled barcode-engine for **Linux®, AIX®, HPUX®, and Mac OS®.** Other operating systems on request.

3.1.2 2D Symbologies

Besides linear barcodes (e. g. 2of5, 2of5 ITL, Code39, Code128, EAN128, EAN, UPC...) **TBarCode/X** also supports 2D symbologies like:

- PDF417
- Data Matrix
- MaxiCode
- QR-Code, Micro QR Code
- Aztec Code

These 2D-symbologies feature very high data capacities with enhanced data security and are required by several enterprises for their documents (and labels) – a selection:

- MaxiCode by UPS®
- PDF417 by General Motors®
- PDF417 and MaxiCode by the AIAG (B-10, Automotive Industry Action Group).

3.1.3 Barcode Quality

TBarCode/X offers the possibility to specify all barcode parameters – these are for example:

- The module width in absolute units (completely device independent).
- Selection of the subsets of Code128 (subsets A, B and C – and automatic mode).
- Advanced 2D bar code properties (PDF417 error correction level etc).
- The barcodes are created as vector graphics (EPS and PCL), therefore utilizing the maximum of the available printing resolution.
- And many others…

3.2 Usage

There are two main usages of **TBarCode/X**

- Create Bar Codes on the Command Line
  All necessary parameters are passed to a command line program and barcodes are saved as vector or bitmap graphics files.
- Filter Print Jobs
  TBarCode/X can process PostScript or PCL print jobs. During the filter process **TBarCode/X** searches for barcode control sequences and replaces them with the barcode graphics. Barcode parameters are specified in the document as part as part of the control sequence.

3.3 System Requirements

3.3.1 Supported Platforms

TBarCode/X binaries are available for

- Linux® (SUSE®, Red Hat®, and other distributions; Intel® x86)
- FreeBSD® 5.4 + 6 (Intel x86)
- AIX® 4.3 + 5.2/5.3 + 6.1 (PowerPC®)
- HP-UX® 11.00 + 11.11 (PA-RISC®), HP-UX® 11.23 (Itanium® 2)
- OS/400® (AS/400®)
- SCO OpenServer® 5.0.7 + 6, SCO UnixWare® 7.1.4 (Intel x86)
- Solaris® 8+9 (SPARC®), Solaris 10 (Intel x86)
- SUSE SLES9 (AMD Opteron® 64 Bit)
- Mac OS X (>= 10.4)
- Please visit our website http://www.tec-it.com to check out the latest versions and supported platforms. Binaries for special platforms are available on request.

3.3.2 Supported Output Devices

- PostScript® Level 2
- PCL® Level 5
- PDF 1.3

3.4 Functional Restriction of the Demo Version

The unlicensed version contains a black bar drawn horizontally across the whole barcode. This horizontal bar disappears as soon as you have installed a valid license file.

► If you want to test the product without this horizontal bar you can request a temporary license key for free. Simply contact sales@tec-it.com.

Section 10 “Licensing” describes how you can acquire a valid license from TEC-IT.

3.5 Version History

3.5.1 TBarCode/X Version History

The detailed version history (including the list of bug fixes) can be found on the web:

http://www.tec-it.com/software/barcode-software/barcode-linux-unix-mac-os-x/history/Default.aspx

Below you find a brief introduction into the main versions:

3.5.1.1 What‘s new in V9

- New Barcode types: **Micro QR Code,** Health Industry Bar Codes (**HIBC**).
- Full support of Aztec Code (optimized encoding).
- Improved quiet zone adjustment. Works for all barcodes now.
- Calculate optimal bitmap sizes.
- PDF output.
- Improved bar width reduction handling (units, decimal numbers).
- Adding ZLib to the X64 version setup.
- New installation path: /usr/local/share/tbarcode9.

3.5.1.2 What's new in V8

- New Barcode type **Aztec Code** inclusive the according settings.
- Several 1D bar codes have been implemented: **DAFT Code, Italian Postal 2 of 5, DPD**.
- **Callback** functions for **MaxiCode**.
- New attributes for parameter bearer type.
- New installation path: /usr/local/share/tbarcode8.

3.5.1.3 What's new in V7

- Image output is now integrated in **TBarCode/X**, ImageMagick is not required anymore.
- The code-base is now identical with **TBarCode DLL** (Library for Windows® and Windows Mobile).
- Detailed version info (including version of **TBarCode** library and revision number).
- New parameters: linebyline, onnodata, compress, bearerwidth, bearertype, reduction, defaultset, mustfit, decoder, sizemode, align, RSSseg
- New parameters for multiple barcodes: multiple, rows, columns, hdist, vdist, datalimit, dynamicsize, structapp
- Deprecated parameters: guardline, barsimmdefaults
- Filter scripts now run with /bin/sh instead of /bin/bash
- New installation path: /usr/local/share/tbarcode7
- **TBarCode** without Daemon: Memory-limitation removed. **TBarCode** automatically reallocates more memory if required.
- More samples added to user documentation.

4 Overview

This section gives you some insight how **TBarCode/X** works and in which ways you can use it. This section is not essential – if you are only looking for the installation instructions you can skip ahead to the according section.

4.1 The TBarCode/X Technology

TBarCode/X exists in two versions:

- **TBarCode/X** without Daemon
- **TBarCode/X** with Daemon

In the version "**TBarCode/X** with Daemon" the barcode generation is performed in a background server process whereas in the other version the barcode generation is done in a single program.

The two versions are actually equivalent:

- Same usage.
- Same functionality.
- Same price.
- Same license – if you have a license for **TBarCode/X** you can use either of the two versions.

The only differences are:

- **TBarCode/X** with Daemon is faster.
- **TBarCode/X** with Daemon is perhaps more difficult to configure.
- **TBarCode/X** with Daemon requires inter-process communication, which is not available on all platforms.

Here is a schematic overview of the **TBarCode/X** components:

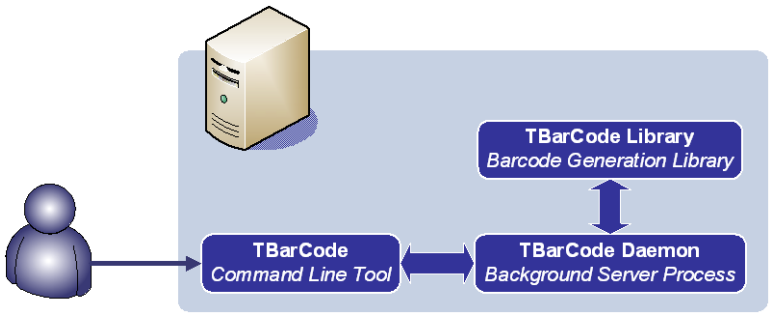

Figure 1: TBarCode/X with Daemon as Background Server Process

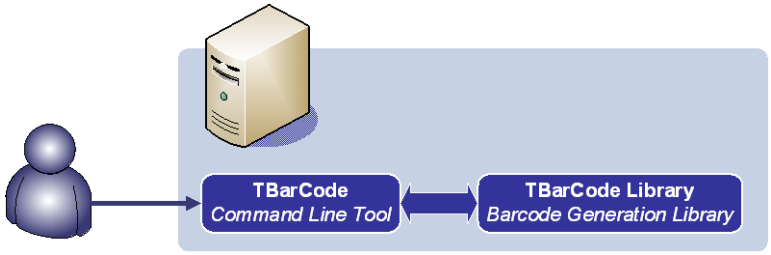

Figure 2: TBarCode/X without Daemon

4.1.1 TBarCode/X Command Line Tool

TBarCode/X is shipped with a command line tool, which can be called from any console (shell) to create barcodes. It can also be used or invoked by shell scripts and applications.

4.1.1.1 Create Barcodes on Command Line

TBarCode/X supports different output formats:

- Vector image formats such as PostScript® (PS, EPS) and PCL®
- Bitmap image formats: BMP, GIF, JPG, PNG, and TIF[1]

The following example command creates a barcode of type “Code 128” that contains the data “abc1234”.

```
tbarcode --output=barcode.eps --barcode=20 --data="abc1234"
```

The resulting barcode is stored as Encapsulated PostScript (*.eps) in the file “barcode.eps”.

4.1.1.2 Using TBarCode/X to Process Data Streams

With the **TBarCode/X** command line application you can also process data streams (like print-jobs). In this “filter mode” the **TBarCode/X** command line application reads data from standard input `(stdin)` and writes the results to standard output `(stdout)`. All barcode related control sequences are replaced by the corresponding barcodes automatically. For example:

```
tbarcode --filter <input.ps >output.ps
```

This command processes the PostScript document `input.ps` and searches for certain barcode control sequences in the file. The control sequences are replaced with barcodes. The resulting document that includes the barcodes is written to `output.ps`. **TBarCode/X** can be installed in the printing system to automatically filter print jobs.

4.1.2 TBarCode/X Library

TBarCode/X Library (also referred to as **LibTBarCode**) is available as static and shared library and as framework for Mac OS. It provides functions to generate barcodes. The **TBarCode/X** command line application uses the functions of the library to create the barcodes.

Programmers can use the library to add barcode generation capabilities to their own applications. By default all required library files and header files (for C/C++) are automatically installed. The complete documentation of the **TBarCode/X Library** API, is available online: http://www.tec-it.com *(Download ► TBarCode/X).*

► To develop your own applications with the **TBarCode/X Library** you need to acquire a developer license from TEC-IT. Just visit our website http://www.tec-it.com or contact us to find out more.

4.1.2.1 TBarCode/X Library Java Interface

For **LibTBarCode** we provide a Java Interface. This interface uses JNI and consists of Java Classes and a JNI interface library (more information see TBarCode/X Developer Manual). The Java Interface is available on request - please contact our support and provide information about your target platform.

[1] Please note: TIF is only supported on Linux systems

4.1.3 TBarCode/X Daemon

The **TBarCode/X Daemon** is a background server process which performs barcode calculations. If the **TBarCode/X Daemon** is installed then the **TBarCode/X** command line application is only a light-weight frontend for the daemon. The separation of the barcode generation process into a light-weight frontend and a server process improves the overall performance.

The daemon can be found at

```
/usr/local/share/tbarcode9/tbarcoded
```

or under Mac OS at

```
/tecit/TBarCode9/tbarcoded
```

The daemon is started automatically as soon as the **TBarCode/X** command line application is invoked. In general there is no need to start the daemon manually.

Please note that the **TBarCode/X Library** does not use or require the daemon.

4.2 About this Manual

Here is a quick overview of the most important sections in this manual.

- The installation of **TBarCode/X** is described in Section 5, “Installation”.
- After installation some basic tests can be performed to see whether **TBarCode/X** was installed correctly. These tests are described in Section 6 “Testing TBarCode/X”.
- The command line usage of **TBarCode/X** is described in Section 7 “Using TBarCode/X”.
- Section 8 “TBarCode/X as Spool Filter” explains how **TBarCode/X** can be configured as a printer filter that automatically filters print jobs.
- Before you use **TBarCode/X** commercially, you need to acquire a valid license from TEC-IT. Section 10 “Licensing” explains how to install a valid license.

5 Installation

TBarCode/X is available in binary form only. The installation package is available in three versions:

- as BIN package, which is a combination of RPM or DEB archive with an installation script
- as RPM or DEB package, which are common formats for Linux operating systems, or
- as TAR-GZ package with installation scripts for UNIX operating systems (AIX, HPUX...).
- as MPKG package for Mac Operating Systems.

Depending on the type of package you have received or downloaded, the installation is slightly different.

5.1 Install TBarCode/X from a BIN Package

If you have received the **TBarCode/X** software as BIN package, then follow the instructions in this section.

BIN packages can be executed like shell scripts and consist of:

- a shell script, which displays the license information and installs the product
- a binary package - the software installation base (RPM or DEB file)

The following steps need to be performed as administrator (user `root`).

1. Open a new console (terminal).
2. Type the command

   ```
   sh ./tbarcode-9.0.0-0.SuSE11.0.i586.rpm.bin
   ```

   Replace the above file name with the file you have downloaded for your operating system.
3. Walk through the displayed license terms with `Space` and confirm them (agree) by entering `Yes` followed by `[Return]`
4. The package will be installed.

> ► If you install an `rpm.bin` package, see also the instructions in 5.2 Install TBarCode/X from an RPM Package

5.1.1 Common Problems

- Failed dependencies, e.g. GCC runtime libraries missing - see A.1 Dependencies
- Shared library "libtbarcode..." not found - see A.2 Shared Library Path.

5.2 Install TBarCode/X from an RPM Package

If you have received the **TBarCode/X** software as RPM package, then follow the instructions in this section.

> ► RPMs (file extension `.rpm`) are archive files for automated software installation. They require an RPM package manager installed on your system.

The following steps need to be performed as administrator (user `root`).

1. Open a new console (terminal).
2. Type the command

```
rpm -i tbarcode-9.0.0-0.i586.rpm
```

   The name of your package might be different. Replace the above file name with the file you have downloaded (or received) for your operating system.

3. Register the **TBarCode/X** libraries (see section A.2 Shared Library Path)
4. Check the execute permissions of **TBarCode/X** (see Section 5.6 “File Permissions”).
5. Installation is complete.

Steps 3 and 4 are actually optional, but they are recommended to ensure that everything is installed properly.

► Instead of using the rpm command in step 2 you can use any package manager that comes with your Linux distribution – for example `gnorpm`, `kpackage`, etc.

5.2.1 Debian, Ubuntu

On Debian-based Linux distributions (such as Ubuntu) the `rpm` command might be missing. In this case consult the manual of your Linux distribution and look for an alternative command.

On Ubuntu, for example, you can install RPM packages using the following command:

```
alien -i tbarcode-9.0.0-0.i586.rpm
```

5.2.2 Common Problems

- Failed dependencies, e.g. GCC runtime libraries missing - see A.1 Dependencies
- Shared library "libtbarcode..." not found - see A.2 Shared Library Path

5.2.3 Remove TBarCode/X

If you have installed **TBarCode/X** from a RPM package, you can remove it with the following steps (as administrator):

1. Open a new console (terminal).
2. Type the command

```
rpm -e tbarcode
```

3. Uninstallation is complete.

Alternatively, you can use any package manager that comes with your Linux distribution.

5.3 Install TBarCode/X from a TAR-GZ Package

TAR-GZ packages are files with the extension `.tar.gz` or `.tgz`.

If you have received **TBarCode/X** as a TAR-GZ package, then follow the instructions in this section.

5.3.1 Prerequisites

Please make sure that the required GCC runtime (see A.1.2) and Iconv (see A.1.3) libraries are installed on the target system.

If the /usr/local directory is missing on your system, follow the steps in section 5.3.3.

5.3.1.1 Prerequisites for AIX

- GZip is a free utility you can download from the AIX Toolbox for Linux Applications. http://www-03.ibm.com/servers/aix/products/aixos/linux/
- For TBarCode/X V9+: Please update your *bos.iconv.ucs* file sets (see A.1.3).

5.3.2 Installation procedure:

The following steps need to be performed as administrator (user `root`).

1. Open a new console (terminal).
2. Type the commands

```
tar xzf SetupTBarCode.tar.gz
cd SetupTBarCode
./install.sh
```

   - The name of your package might be different. Replace the above file name with the file you have downloaded (or received) for your operating system.
   - For command syntax on AIX see section 5.3.2.1 Installation from tar/gz files on AIX

3. Register the **TBarCode/X** libraries (see section A.2 Shared Library Path).
4. Check the execute permissions of **TBarCode/X** (see section 5.6 “File Permissions”).
5. Installation is complete.

Step 3 + 4 is optional, but recommended to ensure that everything is installed properly. Here is an example that shows what the installations procedure could look like:

```
SuSE93:~/temp # tar xzf SetupTBarCode.tar.gz
SuSE93:~/temp # ls -l
total 1058
drwxr-xr-x   3 root root     120 2005-12-20 09:37 .
drwx------  20 root root     784 2005-12-20 09:36 ..
drwxr-xr-x   5 root root     216 2005-11-08 11:45 SetupTBarCode
-rw-r--r--   1 root root 1078102 2005-12-20 09:35 SetupTBarCode.tar.gz
SuSE93:~/temp # cd SetupTBarCode
SuSE93:~/temp/SetupTBarCode # ./install.sh
 TBarCode for Unix - Installation
 --------------------------------
Copying include files...
Copying libraries...
Copying tbarcode files...
Registering TBarCode Library...
Creating link for TBarCode executable...
Setting file permissions...
Installation finished.
SuSE93:~/temp/SetupTBarCode #
```

5.3.2.1 Installation from tar/gz files on AIX

1. First convert gz to tar by typing the following command:

```
gzip -d SetupTBarCode-V9.0.0-AIX5.2-PPC.tar.gz
```

2. Extract the tar file to the directory using this command:

```
tar -xf SetupTBarCode-V9.0.0-AIX5.2-PPC.tar
```

5.3.3 Missing /usr/local directory

TBarCode/X V9 setup scripts create all required directories for you. Earlier versions don't create all required directories, so follow these steps:

1. If the usr/local directory is missing on your system, you need to create the following directories manually:

```
mkdir /usr/local
mkdir /usr/local/bin
mkdir /usr/local/include
mkdir /usr/local/lib
mkdir /usr/local/share
```

2. Give the directories the same rights/permissions as /usr

5.3.4 Common Problems

- Failed dependencies, e.g. GCC runtime libraries missing - see A.1 Dependencies
- Shared library "libtbarcode..." not found - see A.2 Shared Library Path

5.3.5 Uninstall TBarCode/X

If you have installed **TBarCode/X** from a TAR-GZ package, you can remove it with the following steps (as administrator):

1. Open a new console (terminal).
2. Type the commands

```
tar xzf SetupTBarCode-V9.0.0-AIX5.2-PPC.tar.gz
cd SetupTBarCode
./uninstall.sh
```

3. Uninstallation is complete.

5.4 Install TBarCode/X on SCO® Operating Systems

When you are using a SCO operating system, such as SCO OpenServer or SCO UnixWare, you receive **TBarCode/X** as a native package image.

The package usually has the extension `.ds` and the file has a name like `tbarcode-9.0.0.ds`. The name of your package might be different. You will need to substitute the name `tbarcode-9.0.0.ds` with the exact name of your package in the following instructions.

The following steps need to be performed as administrator (user `root`).

1. Open a new console (terminal).
2. Type the command

```
pkgadd -d /home/userXYZ/tbarcode-9.0.0.ds tbarcode
```

3. Check the execute permissions of **TBarCode/X** (see Section 5.6 “File Permissions”).
4. Installation is complete.

You can verify whether the package was installed correctly by typing the following command:

```
pkginfo -l tbarcode
```

5.4.1 Remove TBarCode/X

If you have installed **TBarCode/X** on a SCO operating system, you can remove it with the following steps (as administrator):

1. Open a new console (terminal).
2. Type the command

```
pkgrm tbarcode
```

3. Uninstallation is complete.

5.5 Install TBarCode/X on Mac OS®

When you are using a Mac operating system (Mac OS X) you receive **TBarCode/X** as an installation package which was built with the tool PackageMaker. The package usually has the extension `.mpkg` and the file has a name like `TBarCodeX-9.0.0.mpkg`. (The name of your package might be slightly different.)

The following steps need to be performed (you need an administrator password).

1. Open the package by double-clicking it.
2. Follow the instructions of the installation wizard.

TBarCode has been installed at following paths:

```
/tecit/TBarCode9                          → TBarCodeX
/Library/Frameworks/TBarCode.framework    → TBarCode Framework
```

5.5.1 Remove TBarCode/X

If you have installed **TBarCode/X** on Mac OS, you can remove it by moving following directories to the Trash:

```
/tecit/TBarCode9
/Library/Frameworks/TBarCode.framework
```

5.6 File Permissions

The executables of **TBarCode/X** require certain file permissions. After installation these permissions should be set properly. You can ensure this by checking the directory entries of `/usr/local/share/tbarcode9` using the command

```
ll /usr/local/share/tbarcode9
```

Depending on the operation mode of **TBarCode/X** (with our without daemon process) the following files are displayed by the ls-command.

5.6.1 TBarCode/X with Daemon

The output should look like this:

```
userxy@SuSE93:~> ll /usr/local/share/tbarcode9
total 496
-rwxr-xr-x  1 root root   1581 2005-11-09 09:06 filtercups_pcl.sh
-rwxr-xr-x  1 root root   1649 2005-11-09 09:06 filtercups_ps.sh
-rwxr-xr-x  1 root root   1464 2005-11-09 09:06 filterlprng_fwd.sh
-rwxr-xr-x  1 root root   1064 2005-11-09 09:06 filterlprng.sh
-rw-r--r--  1 root root   1086 2005-11-09 09:06 license.ini
drwxr-xr-x  2 root root    192 2005-12-20 15:51 samples
-rwsr-xr-x  1 root root 261588 2005-11-09 09:06 tbarcode
```

```
-rw-r--r--  1 root root   2547 2005-11-09 09:06 tbarcode.conf
-rwxr--r--  1 root root 216976 2005-11-09 09:06 tbarcoded
-rw-r--r--  1 root root   1702 2005-11-09 09:06 tbarcoded.conf
```

Dates and file sizes may vary – the important information is marked **bold.** The file `tbarcode` needs execute rights and the user-id (SUID) bit needs to be set. The file `tbarcoded` needs to have execute rights for its owner. Missing attributes may be set (by `root` only) with:

```
chmod a+rsx tbarcode
chmod u+x tbarcoded
```

5.6.2 TBarCode/X without Daemon

If **TBarCode/X** was installed without daemon only check the file permissions for these files:

```
userxy@SuSE93:~> ll /usr/local/share/tbarcode9
total 496
-rwxr-xr-x  1 root root   1581 2005-11-09 09:06 filtercups pcl.sh
-rwxr-xr-x  1 root root   1649 2005-11-09 09:06 filtercups ps.sh
-rwxr-xr-x  1 root root   1464 2005-11-09 09:06 filterlprng_fwd.sh
-rwxr-xr-x  1 root root   1064 2005-11-09 09:06 filterlprng.sh
-rw-r--r--  1 root root   1086 2005-11-09 09:06 license.ini
drwxr-xr-x  2 root root    192 2005-12-20 15:51 samples
-rwsr-xr-x  1 root root 261588 2005-11-09 09:06 tbarcode
-rw-r--r--  1 root root   2547 2005-11-09 09:06 tbarcode.conf
```

5.7 SAP® R/3® and mySAP® Integration

TBarCode/X can be used with SAP systems to generate bar codes during printing. You can get more information about the required configuration steps under the following email addresses:

- sap@tec-it.com
- support@tec-it.com

6 Testing TBarCode/X

After installation of **TBarCode/X** it is advisable to test it**.** This can be done from any console (terminal).

6.1 Run TBarCode/X from Command Line

6.1.1 Run the TBarCode Command

Open a new console (terminal) and type the following command:

```
tbarcode --output=barcode.eps --barcode=20 --data="abc1234"
```

This should create new PostScript barcode. You can view the barcode using a PostScript viewer, for example `KGhostview` or similar:

```
kghostview barcode.eps
```

6.1.2 Run TBarCode as Filter

Type the following command:

```
tbarcode --filter  </usr/local/share/tbarcode9/samples/testfile.ps  >output.ps
```

`testfile.ps` is a simple sample document that includes some barcode control sequences. The command processes the document and replaces all barcode control sequences with real barcodes. The result is stored in `output.ps`. Again, you can view the result in any PostScript viewer or directly send `output.ps` to a PostScript printer. For example with:

```
lp -d name_of_printer output.ps
```

Verify that the resulting page contains barcodes.

6.2 Demo License Restriction

When testing **TBarCode/X,** you will probably see a black bar drawn horizontally across the whole barcode. This bar only appears in the unlicensed version of **TBarCode/X.** As soon as you have installed a valid license, all barcodes will be drawn correctly. Section 10 “Licensing” describes how you can acquire a valid license from TEC-IT.

6.3 TBarCode/X isn’t Working?

Please read through the previous sections. Make sure you have performed all required steps during installation. Consult the section “Appendix B: Troubleshooting (FAQ” in case of persisting problems.

7 Using TBarCode/X

7.1 Create a Barcode

The samples below give you a quick start for generating barcodes.

For more detailed instructions read ahead in section 7.3 .

7.1.1 Create a Barcode in EPS (PostScript®) Format

The command below creates a Data Matrix barcode with the data content “2D Code”

```
tbarcode -fPS -oBarcode.ps -b71 -m0.508 -d"2D Code"
```

Parameter	Description
-fPS	Use PostScript® output format (default).
-oBarcode.ps	Write barcode to the output file “Barcode.ps” (specify full path if required)
-b71	Generate Barcode Type Data Matrix (71) – see C.1for more types
-m0.508	Set the module width (X Dimension) to 0.508 mm
-d"2D Code"	Encode the data “2D Code”

7.1.2 Create a Barcode in PCL®-5 (HP-GL/2®) Format

The command below creates an EAN-13 barcode with the data content “123456789012”

```
tbarcode -fPCL -oEAN13.pcl -b13 -m0.508 -d"123456789012"
```

Parameter	Description
-fPCL	Use PCL® output format.
-oEAN13.pcl	Write barcode to the output file “EAN13.pcl” (specify full path if required)
-b13	Generate Barcode Type EAN-13 (13) – see C.1for more types
-m0.508	Set the module width (X Dimension) to 0.508 mm
-d"123456789012"	Encode the data “123456789012” (the check digit is calculated automatically)

7.1.3 Create a Barcode in PDF (Portable Document) Format

The command below creates an EAN-13 barcode with the data content “123456789012”

```
tbarcode -fPDF -oEAN13.pdf -b13 -m0.508 -d"123456789012"
```

Parameter	Description
-fPDF	Use PDF output format.
-oEAN13.pdf	Write barcode to the output file “EAN13.pdf” (specify full path if required)
-b13	Generate Barcode Type EAN-13 (13) – see C.1for more types
-m0.508	Set the module width (X Dimension) to 0.508 mm
-d"123456789012"	Encode the data “123456789012” (the check digit is calculated automatically)

Alternatively the Parameter `-fPDFFRAG` could be used instead of `-fPDF`. In this case a PDF-fragment would be generated instead of a valid document. The fragment could be embedded into other PDF files.

7.1.4 Create a Barcode in Bitmap Format

The command below creates Code 39 barcode with the data content “DATA1234” as GIF image.

```
tbarcode -fIMAGE -iGIF -obarcode.gif -b8 -O -d"DATA1234"
```

Parameter	Description
-fIMAGE	Generate bitmap image
-iGIF	Selected image format = GIF (other formats may be BMP, JPG, PNG, or TIF)
-obarcode.gif	Write barcode to the output file “barcode.gif” (specify full path if required)
-b8	Generate Barcode Type Code-39 (8) – see section C.1for more types
-O	Optimize resolution (required for bitmap graphics).
-d"DATA1234"	Encode the data “DATA1234”

See section 9.1 “Direct Method: Create Bitmap Images with TBarCode/X” for more bitmap samples.

7.2 Filter a Print Job or Document File

7.2.1.1 Control Sequence Structure

The print job (or document) must contain *filter control sequences* in the following format:

Prefix **Data** **Suffix**

$_tbcs [bar code settings] -d **Barcode Data** **$_tbce**

In filter mode TBarCode/X will decode these control sequences and insert a bar code image (either PCL or PostScript format) instead into the print job or document.

The samples below give you a quick start for using **TBarCode/X** in filter mode. For more detailed instructions read ahead in section 7.3 and 7.5 .

7.2.2 Insert a Barcode into a PostScript® Document

Place the following sequence into your document (e.g. infile.ps) to create a barcode.

```
[Text before Barcode]
$ tbcs -fPS -b71 -m0.508 -dMyBarcodeData $ tbce
[Text after Barcode]
```

Barcode sequence parameters:

Parameter	Description
$_tbcs	Begin of barcode control sequence
-fPS	Format of output is PostScript® (default)
-b71	Generate barcode type Data Matrix (71) – see C.1 for more types
-m0.508	Set the module width (X dimension) to 0.508 mm
-dMyBarcodeData	-d marks the begin of barcode data (all characters following will be encoded)
$_tbce	End of barcode sequence

Then call `tbarcode` with the following parameters:

```
tbarcode --filter --stream=PS <infile.ps >outfile.ps
```

Now the `outfile.ps` will contain the original file plus the drawing commands for the barcode.

► For automatic barcode generation by your spool system see chapter 8.

7.2.3 Insert a Barcode into a PCL® Document

Place the following sequence into your document (e.g. infile.pcl) to create a barcode.

```
[Text before Barcode]
$_tbcs -fPCL -b71 -m0.508 -dMyBarcodeData $_tbce
[Text after Barcode]
```

Barcode sequence parameters:

Parameter	Description
`$_tbcs`	Begin of barcode control sequence
`-fPCL`	Format of output is PCL®
`-b71`	Generate barcode type Data Matrix (71) – see C.1 for more types
`-m0.508`	Set the module width (X dimension) to 0.508 mm
`-dMyBarcodeData`	-d marks the begin of barcode data (all characters following will be encoded)
`$_tbce`	End of barcode sequence

Then call `tbarcode` with the following parameters:

```
tbarcode --filter --stream=PCL <infile.pcl >outfile.pcl
```

Now the `outfile.pcl` will contain the original file plus the PCL-5 (HPGL) drawing commands for the barcode.

► For automatic barcode generation by your spool system see chapter 8.

7.3 TBarCode/X Command Line Tool

All features of **TBarCode/X** are available through a single command:

```
tbarcode
```

The executable `tbarcode` is usually located in `/usr/local/bin` or `/usr/bin`. If the path to the `tbarcode` executable is not set in the environment variable `PATH`, you will need to specify the full path to start it. For example:

```
/usr/local/bin/tbarcode
```

7.3.1 Usage

```
tbarcode options barcodesettings
```

► The `options` are used to specify general functionality of the **TBarCode/X** command line application (see section 7.4 “Options”).

► The `barcodesettings` are used to adjust barcode parameters (see section 7.5 "Barcode Settings”).

The parameters may be specified in

- Short style (POSIX style), for example:

  ```
  tbarcode -obarcode.eps -b20 -d"abc1234"
  ```

- Long style (GNU style), for example:

  ```
  tbarcode --output=barcode.eps --barcode=20 --data="abc1234"
  ```

- Windows/DOS style, for example:

```
tbarcode /output=barcode.eps /b=20 /data="abc1234"
```

The available options and barcode settings are described below (using long style and short style). Please note: Only the most important parameters are available in short style.

7.4 Options

You can view the options of the **TBarCode/X** command line application with

```
tbarcode --help
```

7.4.1 General Options

For generating a barcode an output file name is required. All other parameters are optional.

Short	Long	Description

`-o`	`--output=`*FILE*	Specifies the name of the output file.
		Examples:
		`-o/tmp/barcode.eps`
		`--output=/tmp/b.ps`

	`--inifile=`*FILE*	Sets the path and name of the configuration file. The default is `/usr/local/share/tbarcode9/tbarcode.conf`. Example: `--inifile=/home/userXYZ/myTbarcode.conf`
	`--license=`*DIRECTORY*	Sets the path where the license file is located. The default is `/usr/local/share/tbarcode9`. Example: `--license=/etc` The name of the license file is always `license.ini`.
	`--globalxoffset=`*X*	Sets an offset for the x-coordinate. This offset is added to the x-coordinate of the barcode positions. Unit of measurement: millimeters. Example: `--globalxoffset=10.5`
	`--globalyoffset=`*Y*	Sets an offset for the y-coordinate. This offset is added to the y-coordinate of the barcode positions. Unit of measurement: millimeters. Example: `--globalyoffset=-5`
	`--memory=`*SIZE*	Changes the size of the memory reserved for barcode creation. Only relevant when using the **TBarCode/X Daemon.** The daemon uses a fixed memory block for the inter-process communication to exchange barcodes with the **TBarCode/X** command line application. When creating only small barcodes (linear barcodes with la small amount of data), the memory consumption can be reduced by setting this value. The memory block needs to be big enough to hold a complete barcode (= the size of the resulting barcode file). The **TBarCode/X** command line application and the daemon must use identical memory settings – see also the configuration files `tbarcode.conf` and `tbarcoded.conf`. If unsure what to set, then do not edit this parameter manually. Example:

		`--memory=65000`

Table 1: General Options

7.4.2 Filter Options

These parameters allow you to enable and configure the filter mode of **TBarCode/X.** These are optional.

Short	Long	Description
	`--filter`	Enables filter mode. In filter mode the **TBarCode/X** command line application reads data from standard input (`stdin`) and writes the results to standard output (`stdout`). The input stream is scanned for barcode control sequences. Each valid control sequence is replaced with a barcode. The input stream must be PostScript or PCL. All other input streams are not modified by **TBarCode/X.**
	`--stream=`*TYPE*	Sets the type of the input stream. Possible values: ▪ `PS` … PostScript data stream, ▪ `PCL` … PCL data. If not set, **TBarCode/X** automatically detects the type of the input stream. Example: `--stream=PS`
	`--escapebegin=`*STRING*	Sets a string that identifies the beginning of a barcode control sequence. The default value is: `$_tbcs` This string must be distinguishable from any PostScript or PCL/PJL command. In particular: ▪ It must not begin with @, because @ has special meaning in PJL. ▪ It must not begin with <, %, or any other special character that has a special meaning in PostScript. ▪ It must be different than the string set with `escapeend` Example: `--escapebegin=BARCODEBEGIN`
	`--escapeend=`*STRING*	Sets a string that identifies the end of a barcode control sequence. The default value is: `$_tbce` This string must be distinguishable from any PostScript or PCL/PJL command. In particular: ▪ It must not begin with @, because @ has special meaning in PJL. ▪ It must not begin with <, %, or any other special character that has a special meaning in PostScript. ▪ It must be different than the string set with `escapebegin` Example: `--escapeend=BARCODEEND`
	`--insert=`*MODE*	Only for experts: Sets the insert position for the barcode data within the PS or PCL file. Possible values: ▪ `beforeline` ▪ `afterline` ▪ `beforestring` (default) ▪ `afterstring` Example: `--insert=afterline`
	`--linebyline`	Only for experts: Filters the data stream line by line. Normally, a barcode escape sequence can span multiple lines of the input file: The end of the escape sequence (marked with "`$_tbce`" by default) can be

		several lines after the start of the escape start sequence. When line-by-line filtering is activated, the escape sequence is limited to the current line of the input file/stream. This flag can help to recover from filter errors in invalid or unsupported input files.
	`--pclreset`	Creates PCL reset commands at the beginning and the end of the PCL stream in filter mode.
`-S`	`--SAP`	This flag should be set when printing from an SAP environment. (When using Code 39 the characters `*` will be trimmed from begin to the end of the data. For example: `--data=*123*` will be interpreted as `--data=123`)
	`--easybar=`*STATE*	Enables or disables the handling of EasyBar control sequences. Possible values: ▪ `on` ▪ `off` (default) EasyBar control sequences are another type of control sequences for embedding barcodes in PCL data streams. Example: `--easybar=ON`

Table 2: Filter Options

Additionally, there are a number of filter options that can be set *individually for each barcode* – see Section 7.5.7 “Filter Settings”.

7.4.3 Compatibility Options (V1 Format)

The format of the barcode parameters has changed from **TBarCode/X** version 1.*x* to version 2.0 (and higher). This also implies that the syntax of the barcode control sequences has changed.

The **TBarCode/X** command line application can be run in compatibility mode to support the old barcode parameter format. In this way you can easily migrate from version 1.*x* to version 2.0 (or higher).

Short	Long	Description

	`--v1format`	Enables compatibility mode with **TBarCode/X** version 1.*x*.
		All barcode control sequences will be interpreted as with **TBarCode/X** 1.x**.**
		
		If you set this parameter in the `tbarcode.conf` configuration file the `tbarcode` command works like the `tbarcodeclient` in **TBarCode/X** 1.x.
		
		Here is a sample control sequence for **TBarCode/X** 1.x (`--v1format`):
		`$_tbcs tPS b20 dHello World$_tbce`
		
		Here is the same control sequence for **TBarCode/X** 2.0 (and newer):
		`$_tbcs -fPS -b20 -d"Hello World" $_tbce`

Table 3: Compatibility Options

7.4.4 Error Messages and Debug Options

With these additional parameters the creation of debug information and/or log files can be enabled.

Short	Long	Description
	`--errorfile=`*FILE*	Saves all messages in the given file. This should only be used for debugging and not in a production system! Example: `--errorfile=/tmp/tbarcode_errors.log`
	`--syslog`	Logs all messages using the syslog service.

	`--nostderr`	Prevents messages from being written to standard error channel (`stderr`).
	`--trace=`*LEVEL*	Sets the trace level to a certain value. The trace level defines the amount of log messages that are written to an error file, syslog or stderr. Possible values (sorted from minimal to maximal information output): ▪ `error` (default) ▪ `warning` ▪ `info` ▪ `verbose` Example: `--trace=INFO`
	`--onerror=`*ACTION*	Defines the action if wrong barcode settings are applied. Possible values: ▪ `ignore` ▪ `message` (default) When using the default setting (`--onerror=message`) **TBarCode/X** reports wrong barcode parameters. Additionally the exit value is set to the corresponding error code. When `--onerror=ignore` is set **TBarCode/X** ignores errors.
	`--onnodata=`*ACTION*	Defines the action if the data parameter is missing (`-d` or `--data`). Possible values: ▪ `ignore` ▪ `message` (default) When using the default setting (`--onnodata=message`) **TBarCode/X** reports missing barcode data. Additionally the exit value is set to the corresponding error code. When `--onnodata=ignore` is set **TBarCode/X** ignores missing barcode data.

Table 4: Error Message and Debug Options

7.4.5 Informative Output

Use one of these parameters for displaying help information.

Short	Long	Description
`-s`	`--barcodesettings`	Shows a help text for all barcode settings.
`-?`	`--help`	Shows a help text for general option.
	`--shorthelp`	Shows a short help text.
	`--version`	Shows the version information.

Table 5: Informative Output

7.5 Barcode Settings

You can view the available parameters for barcode settings with

```
tbarcode --barcodesettings
tbarcode -s
```

► Please be aware that the following bar code settings are used by TBarCode in command line mode as well as in spool filter mode (filter control sequence prefix).

7.5.1 Barcode Type, Barcode Data

These parameters are used for specifying the general barcode settings.

Short	Long	Description
`-b`	`--barcode=`*NUMBER*	Sets the type of barcode. The *NUMBER* of the barcode type can be looked up in Section C.1 “Barcode Symbologies”. Default is `--barcode=20`, which is “Code 128”.

		Examples:
		`-b20`
		`--barcode=71`

`-d`	`--data=`*DATA*	Sets the data of a barcode.
		Alternatively, you can specify a file that contains the data with `--datafile`.
		Examples:
		`-d12345`
		`--data=12345`
		`-d"ABCD 12345"`
		`--data="ABCD 12345"`
		Double quotes (") need to be escaped with two double quotes (""). So if you want to encode the data
		`Text "123"`
		into a barcode you need to write
		`--data="Text ""123"""`
		The encoding of the characters passed with the `--data` parameter must be "CP1252" (ANSI ASCII)[2]. Most Linux/UNIX systems have set UTF-8 as default command line encoding. If you encode only characters in the code range of "0x00 - 0x7F" (ASCII area) no conversion is required, but for other characters (passed via command line) you have to convert UTF-8 to CP1252, e.g. with the "iconv" utility.
		Please contact support@tec-it.com if you have questions.

	`--datafile=`*FILE*	Sets the file that contains the barcode data. *FILE* can be any ASCII or binary file. Alternatively, you can directly specify the data as command line parameter with `--data`. Example: `--datafile=/home/userXY/bcdata.dat`

`-c`	`--checkdigit=`*NUMBER*	Sets the check-digit calculation method. The *NUMBER* of the check-digit method can be looked up in Section C.2 "Check Digit Methods".
		For specific bar code types the standard check digit is calculated by default.
		Examples:
		`-c3`
		`--checkdigit=3`

	`--autocorrect=`*STATE*	Enables or disables auto-correction. Relevant for Code 2of5 Interleaved only. This feature adds a leading zero to the barcode data to produce an even number of digits. Possible values: ▪ `on` ▪ `off` Example: `--autocorrect=ON`
`-e`	`--translation=`*STATE*	Enables or disables the translation of escape sequences (see also *Escape Sequences* in the *Barcode Reference*). Possible values: ▪ `on` ▪ `off`

[2] Upcoming releases of TBarCode/X may support UTF-8 encoding through a separate option.

		Example:
		`-eon`
		`--translation=ON`

	`--formatstring=`*FORMAT*	Sets the format string. The format string syntax can be looked up in Section C.12 “Formatting Barcode Data”. Example: `--formatstring="A##B&"`
	`--compress=`*ALGORITHM*	Compresses the data by using a compression algorithm. Possible algorithms: ▪ `NONE` (default) ▪ `DEFLATE` ▪ `GZIP` ▪ `ZLIB` Compression may be applied when a large amount of data has to be encoded as barcodes. Compression should only be used in closed applications only with barcode symbologies that support binary data (e.g. Data Matrix, PDF417, MicroPDF, QR Code, etc.). After reading the barcode the data has to be decompressed using the appropriate algorithm. Example: `--compress=DEFLATE` Important: To use compression the **ZLib** compression library (available at: http://www.zlib.net/) has to be installed on your Linux or UNIX server.

	`--bcfile=`*FILE*	Instead of specifying the barcode settings as command line parameters, you can specify a file that contains the barcode settings.
		Example:
		`tbarcode -obc.eps --bcfile=settings.dat --data=0123`
		Example content of “`settings.dat`”:
		`barcode=20`
		`modulewith=0.352`
		`width=35`
		`height=15`
		The syntax of a barcode settings file is identical to the syntax of a configuration file. See Section 7.6.2 “Syntax of a Configuration File”.

	`--defaultset=`*NUMBER*	Use a certain set of default bar code settings. `--defaultset=1` should be used when you are migrating from a hardware-based barcode printing solution to **TBarCode/X**.

Table 6: General Barcode Settings

7.5.2 Output Format

These parameters are used for specifying the output format (like bitmap, EPS, PCL, color modes, background etc.).

Short	Long	Description
`-f`	`--format=`*TYPE*	Defines the output format. Possible values: ▪ `PS` … PostScript ▪ `PCL` … PCL ▪ `PDF` … PDF ▪ `PDFFRAG` … PDF fragment ▪ `IMAGE` … a bitmap image format The default setting is `--format=PS`. In filter mode the output format is the same as the format of the input stream. When `--format=IMAGE` is set, then the parameter `--imageformat`

		determines the bitmap format. Example, creating a barcode as GIF: `--format=IMAGE -imageformat=GIF --output=Barcode.gif`
`-i`	`--imageformat=`*FORMAT*	Defines the bitmap format which is used for output. This parameter is only relevant when `--format=IMAGE` is set. *FORMAT* is the extension of the bitmap format. Currently supported formats are: `BMP` (default), `GIF`, `JPG`, `PNG` and `TIF` Example: `--format=IMAGE --imageformat=GIF --output=Barcode.gif`
	`--dpi=`*DPI*	Sets the resolution of the image. Unit of measurement: Dots per inch (dpi).
	`--nooverhead`	Suppresses the PCL or PostScript overhead. PCL: Reset commands are omitted on begin and at the end of the file. PostScript: The overhead for encapsulated PostScript (EPS) is omitted.
	`--bkmode=`*MODE*	The background mode of the generated output. Possible values: ▪ `transparent` … background is transparent (no background) ▪ `opaque` … a white filled rectangle is drawn (default) Example: `--bkmode=transparent`
	`--colormode=`*MODE*	The color mode of the output. Only relevant for PostScript. Possible values: ▪ `CMYK` … CMYK color space ▪ `GRAY` … Grayscale color space ▪ `RGB` … RGB color space (default) Example: `--colormode=CMYK`
	`--pclmode=`*MODE*	The PCL output mode. By default **TBarCode/X** creates PCL Level 5 compatible output. PCL Level 5 compatible output includes HP-GL/2 drawing operations. Some barcode types, such as MAXICODE, can only be drawn with HP-GL/2. Unfortunately, some printers are not fully PCL Level 5 compatible and do not understand HP-GL/2 drawing operations. Therefore, HP-GL/2 output can be disabled with this option. Possible values: ▪ `PCL5` … PCL5 compatible output ▪ `PCL5noHPGL` … PCL5 compatible output without HP-GL/2 operations Example: `--pclmode=PCL5noHPGL`

Table 7: Output Format Settings

7.5.3 Barcode Size and Drawing Position

Use these parameters to adjust the bar code size and to optimize the bar code quality. All of these parameters are optional.

Short	Long	Description

`-w`	`--width=`*WIDTH*	Sets the width of the barcode (see also `--sizemode=fit`).
		Unit of measurement: millimeters.
		Examples:
		`-w25.4`
		`--width=55`

`-h`	`--height=`*HEIGHT*	Sets the height of the barcode.
		Unit of measurement: millimeters.
		Examples:
		`-h15`
		`--height=25.4`

`-m`	`--modulewidth=`*WIDTH*	Sets the module width. Unit of measurement: millimeters. Example: `--modulewidth=0.254` See also `--sizemode --printratio --optimalwidth`
	`--mustfit=`*STATE*	When activated **TBarCode/X** returns an error if the barcode does not fit into the given bounding rectangle (width x height). Possible values: ▪ `on` ▪ `off` (default) Example: `--mustfit=OFF`

`-r`	`--rot=`*ROTATION*	Sets the rotation of the barcode:
		Unit of measurement: degrees (counterclockwise, only 90° angles are supported).
		Possible values:
		▪ `0` (default)
		▪ `90`
		▪ `180`
		▪ `270`
		Examples:
		`-r90`
		`--rot=180`

	`--sizemode=`*MODE*	Sets the mode that determines the barcode size. ▪ `fit` … The parameters `--width` and `--height` determine the size. ▪ `module` … The parameter `--modulewidth` determines the size (width). ▪ `minimal` … The parameters `--decoder` and `--dpi` determine the size. The default size mode is `fit`. If no width and/or height is specified, the bar code size depends on internal default values. When `--sizemode=MINIMAL` is used **TBarCode/X** automatically considers the decoding solution and the resolution of the document. It will then create a barcode with minimal size that should be optimally readable under the given conditions. Example scenario: You are receiving documents per FAX (200 dpi) and you want to decode the barcodes on a server (software solution). You can optimize the printed barcodes by specifying the following options: `--decoder=software --dpi=200 --sizemode=MINIMAL`
	`--decoder=`*TYPE*	Specifies the type of barcode decoder which will be used for scanning the barcode. Used in combination with `--sizemode=MINIMAL` Possible values: ▪ `any` … Default. The type of decoder is unknown. ▪ `hardware` … A hardware barcode scanner (such as a handheld-device). ▪ `software` … A software barcode decoder. ▪ `tbarcode` … The **TBarCode Scanner**. The **TBarCode Scanner** is a software decoding solution. It is available on request – just contact office@tec-it.com By setting the type of decoder, **TBarCode/X** can optimize the size of the barcode to ensure optimal readability.

		Example scenario: You are receiving documents per FAX (200 dpi) and you want to decode the barcodes on a server (software decoding solution). You can optimize the printed barcodes by specifying the following options: `--decoder=software --dpi=200 --sizemode=MINIMAL`
	`--dpi=`*DPI*	Sets the resolution of the image. Unit of measurement: Dots per inch (dpi).

`-x`	`--xpos=`*POSITION*	Sets the (absolute or relative) x-position of the barcode.
		Unit of measurement: millimeters.
		The positioning mode (absolute or relative positioning) can be set with `--pos`.
		Examples:
		`--pos=abs --xpos=100`
		`--pos=rel --xpos=-10.5`

`-y`	`--ypos=`*POSITION*	Sets the (absolute or relative) y-position of the barcode.
		Unit of measurement: millimeters.
		The positioning mode (absolute or relative positioning) can be set with `--pos`.
		Examples:
		`--pos=abs --ypos=100`
		`--pos=rel --ypos=-10.5`

	`--origin=`*ORIGIN*	Sets the origin of the barcode. The origin is the coordinate that can be set with `--xpos` and `--ypos`. Possible values: ▪ `top` (The origin is the top left corner of the barcode.) ▪ `bottom` (The origin is the bottom left corner of the barcode.) Example: `--origin=TOP`

Table 8: Barcode Size and Position

7.5.4 Appearance (Quiet Zone, Print Ratio…)

These parameters are used for specifying appearance options like Bearer Bar, Quiet Zone and Narrow To Wide Bar Ratio (Print Ratio). All of these parameters are optional.

Short	Long	Description
	`--bearertype=`*TYPE*	Sets the type of the bearer bar. Possible values: ▪ `none` … default ▪ `topandbottom` … Horizontal bar above and below the barcode. ▪ `rectangle` … Rectangle around the barcode. ▪ `top` … Horizontal bar above the barcode. ▪ `bottom` … Horizontal bar below the barcode. ▪ The former value `horizontal` is deprecated and has been replaced by `topandbottom`. Example: `--bearertype=TOPANDBOTTOM`
	`--bearerwidth=`*WIDTH*	Sets the width of a bearer bar. Unit of measurement: millimeters. Example: `--bearerwidth=1.5`
	`--notchheight=`*HEIGHT*	Set the notch height. Unit of measurement: millimeters. Example:

		`--notchheight=2.0`
	`--printratio=`*RATIO*	Sets the print ratio (ratio of narrow to wide bars and spaces). See also *Print Ratio* in the *Barcode Reference*. Example: `--printratio="1:2:1:3"`
	`--quietzoneunit=`*UNIT*	The unit of quiet zones (see also `-quietzoneh` and `-quietzonev`). Possible values: ▪ `none` … No Quiet Zone (default) ▪ `mod` … In Number of Modules. ▪ `mm` … Millimeters. ▪ `mils` … Mils (1 Mil = 1/1000 Inch). ▪ `inch` … Inches. ▪ `px` … Pixels. Example: `--quietzoneunit=mod`
	`--quietzoneh=`*UNITS*	Sets the width of the horizontal quiet zone. A horizontal quiet zone is the empty space in the left and the right of a barcode. Unit of measurement: see `-quietzoneunit`. Example: `--quietzoneh=10`
	`--quietzonev=`*UNITS*	Sets the height of the vertical quiet zone. A vertical quiet zone is the empty space in the top and the bottom of a barcode. Unit of measurement: see `-quietzoneunit`. Example: `--quietzonev=10`

Table 9: Barcode Appearance Options

7.5.5 Quality Enhancement

These parameters are used for enhancing bar code quality depending on output format. All of these parameters are optional.

Short	Long	Description
	`--dpi=`*DPI*	Sets the resolution of the image. Unit of measurement: Dots per inch (dpi).
`-O`	`--optimalwidth` `--72dpiraster`	Optimizes the module width for the given output resolution. This option reduces aliasing effects in bitmaps and minimizes printing tolerance. For low output resolution the module width optimization is a must to get a readable bar code! Use this parameter in combination with the `--dpi` parameter. When this setting is turned on, the X dimension (module width) will become exactly a multiple of a single printer dot (or Pixel). Module widths with fractional parts are avoided. This option is useful if you want create bitmap barcodes with maximal quality. All drawing operations will fit exactly into the pixel raster of a bitmap. See also *Optimize Barcode for the Output Device* in the *Barcode Reference*.
	`--reduction=`*REDUCTION*	Bar width reduction (also known as "Pixel Shaving" or BRW). Reduction of the nominal bar width dimension to compensate for systematic errors (e.g. dot gain) in some printing processes – usually applied on film masters or printing plates but also useful to compensate ink bleeding or high toner saturation. The given value reduces the width of the modules by a certain amount. The BWR unit has to be specified with the argument `--reductionunit`. Example for reducing the module width 10%:

		`--reduction=10 --reductionunit=perc`
	`--reductionunit=`*UNIT*	The unit of the bar width reduction value (see also `--reduction`). Possible values: ▪ `perc` … Percent (default) ▪ `mm` … Millimeters. ▪ `mils` … Mils (1 Mil = 1/1000 Inch). ▪ `inch` … Inches. Example: `--reductionunit=perc`

Table 10: Barcode Quality Options

7.5.6 Text Settings

These parameters can be used to fine-tune the output of the human readable text. They are optional.

Short	Long	Description

`-t`	`--text=`*POS*	Sets the position of the barcode of the readable barcode text or hides the barcode text.
		Possible values:
		▪ `below` Draws the text below the bars (default for most barcodes)
		▪ `above` Draws the barcode text above the bars.
		▪ `h | hide` Hides the barcodes text (draws only the bars).
		Examples:
		`-th`
		`--text=HIDE`

	`--align=`*ALIGNMENT*	Sets the horizontal text alignment. Possible values: ▪ `default` ▪ `left` ▪ `center` ▪ `right` Example: `--align=left`
	`--fontsize=`*SIZE*	Sets the size of the readable barcode text. Unit of measurement: Points

	`--font=`*NAME*	Sets the font that is used for drawing the readable barcode text.
		The font is only relevant when creating PostScript or PCL output. (Bitmaps do not yet support text output.)
		Example (Postscript):
		`--font=Helvetica`
		PCL fonts are specified using a PCL font number (see Section C.13 “PCL Font Numbers”).
		Example (PCL):
		`--font=4099`

	`--textdist=`*DISTANCE*	Sets the distance between the bars and the readable barcode text. Unit of measurement: millimeters.
	`--trimwhitespaces`	Removes all whitespaces (spaces, tabs, etc.) from begin and the end of the barcode data.

Table 11: Barcode Text Options

7.5.7 Filter Settings

To enable filtering **TBarCode/X** has to be called with the program option `--filter`. See Section 7.4.2 “Filter Options”.

The following filter parameters can be used to fine-tune single barcodes individually. These parameters are optional.

Short	Long	Description
	`--initgraphics`	Calls *initgraphics* in PostScript.
	`--movecursor`	Moves cursor in the PCL code to end of the barcode.
	`--remove`	Removes barcode control sequence from the data stream after filtering. (The default behavior is to overwrite the barcode control sequences with blanks.)
	`--embed=`*STATE*	Defines the type of PostScript/PCL code that is created. Possible values: ▪ `on` (default for filtering) ▪ `off` `--embed=on` creates a barcode that can be inserted into a PostScript/PCL stream or file. `--embed=off` creates a stand-alone PostScript/PCL file.
	`--pos=POS`	Sets the positioning mode to relative or absolute coordinates. Possible values: ▪ `abs` (default for PostScript) ▪ `rel` (default for PCL)

Table 12: Filtering Options

7.5.8 PDF417 Settings

All of these parameters are optional and can be used to fine-tune the generation of PDF417.

Short	Long	Description
	`--PDFrows=`*ROWS*	Sets the number of rows. Possible values: ▪ `3 … 90` Example: `--PDFrows=10`
	`--PDFcols=`*COLUMNS*	Sets the number of columns. Possible values: ▪ `1 … 30` Example: `--PDFcols=9`
	`--PDFratio=`*RATIO*	Sets the row-columns-ratio. Example: `--PDFratio=”3:1”`
	`--PDFauto`	Automatically chooses the row-column-ratio.
	`--PDFrowheight=`*HEIGHT*	Sets the height of a row. Units of measurement: millimeters. Examples: `--PDFrowheight=5.0`
	`--PDFecl=`*LEVEL*	Sets the error correction level. Possible values: ▪ `0 … 8`

		Example: `--PDFecl=0`
	`--PDFmode=`*MODE*	Sets the PDF417 encoding mode. The PDF417 modes can be looked up in Section C.3.1 “Encoding Mode”.

Table 13: PDF417 Options

7.5.9 Micro PDF417 Settings

All of these parameters are optional and can be used to fine-tune the generation of Micro PDF417.

Short	Long	Description
	`-- MPDFversion=`*VERSION*	Sets the Micro PDF417 version (symbol size). The possible values can be looked up in Section C.4.1 “Version (Symbol Sizes)”.
	`-- MPDFmode=`*MODE*	Sets the Micro PDF417 mode. The Micro PDF417 modes can be looked up in Section C.4.2 “Mode”.

Table 14: Micro PDF417 Options

7.5.10 Macro PDF417 Settings

All of these parameters are optional and can be used to fine-tune the generation of Macro PDF417.

Short	Long	Description
	`--PDFindex=`*INDEX*	Sets the segment index.
	`--PDFid=`*ID*	Sets the file ID.
	`--PDFlast`	Last segment
	`--PDFfile=`*NAME*	Sets the file name.
	`--PDFcount=`*COUNT*	Sets the segment count.
	`--PDFtime=`*TIMESTAMP*	Sets timestamp.
	`--PDFsender=`*SENDER*	Sets the sender.
	`--PDFaddr=`*ADDRESSEE*	Sets the addressee.
	`--PDFsize=`*SIZE*	Sets the file size.
	`--PDFchecksum=`*SUM*	Sets the checksum.

Table 15: Macro PDF417 Options

7.5.11 Data Matrix Settings

All of these parameters are optional and can be used to fine-tune the generation of Data Matrix.

Short	Long	Description
	`--DMsize=`*SIZE*	Sets the Data Matrix size. The Data Matrix sizes can be looked up in Section C.5.1 “Symbol Sizes”.
	`--DMformat=`*FORMAT*	Sets the Data Matrix format. The Data Matrix formats can be looked up in Section C.5.2 “Format”.
	`--DMbinary`	Barcode content is encoded in the binary mode.
	`--DMrect`	Draws Data Matrix as a rectangle. (Square is default.)
	`--DMsum=`*SUM*	Sets sum of structured append. Possible values: ▪ `2 … 16`
	`--DMindex=`*INDEX*	Sets index of structured append. Possible values: ▪ `1 … 16`
	`--DMfile=`*ID*	Sets the file id of structured append.

Table 16: Data Matrix Options

7.5.12 MaxiCode Settings

All of these parameters are optional and can be used to fine-tune the generation of MaxiCode.

Short	Long	Description
	`--MCmode=`*MODE*	Sets the mode of the MaxiCode. Possible values: ▪ `2` … `5`
	`--MCundercut=`*UNDERCUT*	Sets the undercut of the hexagons. Unit of measurement: Percents Possible values: ▪ `0` … `100`
	`--MCpre=`*PREAMBLE*	Sets the preamble.
	`--MCsum=`*SUM*	Sets the total number of symbols of the symbol chain.
	`--MCindex=`*INDEX*	Sets the index of the symbol in the structured appends symbol chain. Possible values: ▪ `1` … `8`
	`--MCservice=`*SERVICE*	Sets the service class of the structured carrier message.
	`--MCcountry=`*COUNTRY*	Sets the country code of the structured carrier message.
	`--MCpostal=`*POSTAL*	Sets the postal code of the structured carrier message.

Table 17: MaxiCode Options

7.5.13 QR-Code Settings

All of these parameters are optional and can be used to fine-tune the generation of QR-Code.

Short	Long	Description
	`--QRversion=`*VERSION*	Sets the QR-Code version (symbol size). The possible values can be looked up in Section C.7.1 “Version (Symbol Sizes)”.
	`--QRformat=`*FORMAT*	Sets the QR-Code format. The possible values can be looked up in Section C.7.2 “Format”.
	`--QRind=`*INDICATOR*	Sets the format application indicator.
	`--QRecl=`*LEVEL*	Sets the number of the error correction level. The error correction levels can be looked up in Section C.7.3 “Error Correction Level”. Possible values: ▪ `0` ▪ `1` (default) ▪ `2` ▪ `3`
	`--QRmask=`*PATTERN*	Sets the mask pattern (0..7).
	`--QRsum=`*SUM*	Sets the total number of symbol in a symbol chain. Possible values: ▪ `2` … `16`
	`--QRindex=`*INDEX*	Sets the index of the current symbol in the symbol chain. Possible values: ▪ 1 … 16
	`--QRparity=`*PARITY*	Sets the parity byte (structured append).

Table 18: QR-Code Options

7.5.14 Micro QR-Code Settings

All of these parameters are optional and can be used to fine-tune the generation of Micro QR-Code.

Short	Long	Description
	`--MQRversion=`*VERSION*	Sets the Micro QR-Code version (symbol size). The possible values can be looked up in Section C.8.1 “Version (Symbol Sizes)”.

	`--MQRecl=`*LEVEL*	Sets the number of the error correction level. The error correction levels can be looked up in Section C.8.2 “Error Correction Level”. Possible values: ▪ `0` ▪ `1` (default) ▪ `2` ▪ `3`
	`--MQRmask=`*PATTERN*	Sets the mask pattern (0..4).

Table 19: Micro QR-Code Options

7.5.15 Codablock-F Settings

All of these parameters are optional and can be used to fine-tune the generation of Codablock-F.

Short	Long	Description
	`--CBrows=`*ROWS*	Sets the number of rows. Possible values: ▪ `2 … 44`
	`--CBcols=`*COLUMNS*	Sets the number of columns. Possible values: ▪ `4 … 62`
	`--CBrowheight=`*HEIGHT*	Sets the height of a row. Unit of measurement: millimeters.
	`--CBsepheight=`*HEIGHT*	Sets the height of the row-separator. Unit of measurement: millimeters.
	`--CBformat=`*FORMAT*	Sets the format.

Table 20: Codablock-F Options

7.5.16 Aztec Code Settings

All of these parameters are optional and can be used to fine-tune the generation of Aztec Code.

Short	Long	Description
	`--ACsize=`*SIZE*	Sets the Aztec Code symbol size. The possible values can be looked up in Section C.10.1 “Symbol Sizes”.
	`--ACbinary`	Barcode content is encoded in the binary mode.
	`--ACecl=LEVEL`	Sets the error correction level in percent. Possible values: ▪ `0 … 89`
	`--ACrunes`	Switch into “Runes” mode. *Aztec Runes* are small distinct machine-readable marks which are able to encode values from 0 … 255 (8 bits).
	`--ACformat=`*FORMAT*	Sets the Aztec Code format. The possible values can be looked up in Section C.10.2 “Format”.
	`--ACspec=`*SPECIFIER*	Sets the format specifier. Is considered only when ACformat is set to **2** (*Industry format*).
	`--ACappend`	Enables the structured append mode.
	`--ACsum=`*SUM*	Sets the total number of the structured append symbols. Possible values: ▪ `'A' … 'Z'`
	`--ACindex=`*INDEX*	Sets the index of the structured append symbol. Possible values: ▪ `'A' … 'Z'`
	`--ACmessage=`*ID*	Sets the structured append message id.

Table 21: Aztec Code Options

7.5.17 RSS Expanded Stacked Settings

This parameter is optional and can be used to fine-tune the generation of RSS Expanded Stacked.

Short	Long	Description
	`--RSSseg=`*SEGMENTS*	Sets the number of segments per row. Possible values: ▪ `2 … 22`

Table 22: RSS Expanded Stacked Options

7.5.18 Composite Barcode Settings

This parameter is optional and can be used to fine-tune the generation of Composite Barcodes.

Short	Long	Description
	`--CCtype=`*TYPE*	Sets the type of composite component. Possible values: ▪ `none` ▪ `auto` ▪ `A` ▪ `B` ▪ `C`

Table 23: Composite Barcode Options

7.5.19 Multiple Barcodes

Creating “multiple barcodes“ is a new feature in **TBarCode/X 9.0.** This feature allows the user to encode data into multiple barcodes instead of a single barcode. In this way large amounts of data can be encoded, even by barcodes with limited data capacity.

Multiple barcodes should only be used with the following barcode symbologies:

- Data Matrix
- PDF417
- MicroPDF
- QR code
- Aztec Code

Example:
The following command creates multiple barcodes.

```
tbarcode --output=barcodes.eps -b71 --width=40 --height=10 --dynamicsize=vertical
         --multiple=on --columns=4 --sizemode=minimal --decoder=hardware
         --structapp=standard --data=…
```

The initial size of the barcode is 40 mm × 10 mm (`--width=40 --height=10`). Depending on the amount of data, **TBarCode/X** automatically creates multiple barcodes as shown in the following figure.

Figure 3: Multiple Data Matrix Barcodes (1)

When more data is encoded, **TBarCode/X** adds more barcodes automatically. Up to four barcodes are drawn per row (`--columns=4`).

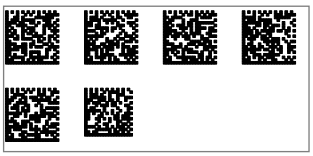

Figure 4: Multiple Data Matrix Barcodes (2)

For even larger amounts of data **TBarCode/X** starts a new row and the image size grows vertically (`--dynamicsize=vertical`).

A structured append information is added to the barcodes (`--structapp=standard`). The barcodes can be scanned in any order. The barcode scanner will use the structured append information to identify the correct order of the barcodes and decode the data correctly[3].

For generating multiple barcodes you have to set `–multiple=on`. All other parameters are optional.

Short	Long	Description
	`--multiple=`*STATE*	Enables or disable multiple barcodes. Possible values: ▪ `on` ▪ `off`
	`--rows=`*ROWS*	Sets the number of barcode rows. For example, to draw 3 rows with barcodes: `--rows=3`
	`--columns=`*COLUMNS*	Sets the number of barcode columns. For example, to draw 4 columns with barcodes: `--columns=4`
	`--hdist=`*DISTANCE*	Sets a minimal horizontal distance between barcodes. Unit of measurement: millimeters. Example: `--hdist=5`
	`--vdist=`*DISTANCE*	Sets a minimal vertical distance between barcodes. Unit of measurement: millimeters. `--vdist=5`
	`--datalimit=`*LIMIT*	Sets the maximal amount of data (data bytes) that is encoded in a single barcode. Example: `--datalimit=1000`
	`--dynamicsize=`*MODE*	Allows the barcode to grow in horizontal or vertical direction. Possible values: ▪ `none` …default ▪ `horizontal` … The barcode can grow in horizontal direction. ▪ `vertical` … The barcode can grow in vertical direction. Example:

[3] Not all barcode scanners support “Structured Append”.

		`--dynamicsize=VERTICAL`
	`--structapp=`*MODE*	Sets the structured append mode that is used for multiple barcodes. Possible values: ▪ `none` … No structured append mode is used. ▪ `standard` … The barcodes own structured append mode is used. ▪ `tbarcode` … A proprietary structured append mode is used. Important: `--structapp=STANDARD` can only be used with barcode symbologies that provide a structured append mode (for example: Data Matrix, PDF417, etc.). `--structapp=TBARCODE` creates a proprietary structured append mode. This structured append mode can only be decoded by using the **TBarCode Scanner** solution. It is not supported by standard scanner devices.

Table 24: Multiple Barcodes Options

7.5.20 Deprecated Barcode Settings

The following options are deprecated. They are still supported in the current release, though it is not guaranteed that they will be supported in future releases.

Deprecated Option	New Option (Replacement)
`--guardline=`*WIDTH*	`--bearerwidth=`*WIDTH* `and --bearertype=`*TYPE*
`--barsimmdefaults`	`--defaultset=1`

Table 25: Deprecated Options

7.6 TBarCode/X Configuration Files

Each **TBarCode/X** executable has a configuration file to define global settings.

Executable	Name of Configuration File.
`tbarcode`	`tbarcode.conf`
`tbarcoded`	`tbarcoded.conf`

Table 26: TBarCode/X Configuration Files

Each time `tbarcode` or `tbarcoded` is started the application reads the configuration file.

- `tbarcoded` is the **TBarCode/X Daemon**, which is the background server process of **TBarCode/X**. This file is not available in all releases of **TBarCode/X**.
- Only system administrators can edit the **TBarCode/X** configuration files.

7.6.1 Path of Configuration Files

TBarCode/X searches in the following directories for a suitable configuration files "`tbarcode.conf`" or "`tbarcoded.conf`":

1. In the current directory.
2. In the directory of the executable.
3. In `/usr/local/share/tbarcode9`.
4. For Mac OS: in /tecit/TBarCode9

The path and the name of the configuration file can be overwritten with the command line option: `--inifile`.

An administrator can edit these files to set global settings for **TBarCode/X**. These settings are applied each time when a new instance of **TBarCode/X** is started. The settings in the configuration files have the same functionality as the settings on the command line of **TBarCode/X**.

7.6.2 Syntax of a Configuration File

The syntax of the **TBarCode/X** configuration files is similar (but not identical) to the syntax of most UNIX configuration files.

A line of configuration file contains either:

- an option or a barcode setting, or
- a comment.

7.6.2.1 Options and Barcode Settings

These have the following syntax

```
option
```

or

```
option=value
```

7.6.2.2 Comments

If the first character in a line is `#`, then this line is treated as a comment – its content is ignored.

```
# This is a comment.
```

7.6.3 tbarcode.conf

The configuration file `tbarcode.conf` can contain options and barcode settings as described in Section 7.4 “Options” and 7.5 “Barcode Settings”.

Example:

```
#Sample tbarcode.conf

memory=524288
SAP
v1format
defaultset=1
errorfile=/tmp/tbarcode.log
nosyslog
nostderr
trace=verbose
globalxoffset=10
globalyoffset=10
escapebegin=BARCODE_START
escapeend=BARCODE_END
```

7.6.4 tbarcoded.conf

The configuration file `tbarcoded.conf` can contain the following options:

`memory, license, errorfile, nosyslog, nostderr, trace`

7.6.5 Priority of Options and Barcode Settings

As shown so far, options can be set at three levels:

- As command line parameter.
- In a configuration file.
- In a custom barcode settings file (using `--bcfile`).

The same options could be set multiple times. In this case the options on the command line and in the barcode settings file override the options in the configuration files.

8 TBarCode/X as Spool Filter

TBarCode/X can be installed in the spool system to automatically filter print jobs. **TBarCode/X** works with all PostScript- or PCL-based printing queues. **TBarCode/X** scans all print jobs for certain barcode control sequences. Here is an example of a barcode control sequence:

```
$_tbcs -b3 -d"1234567890"$_tbce
```

When **TBarCode/X** detects such a sequence it automatically replaces the sequence with a barcode.

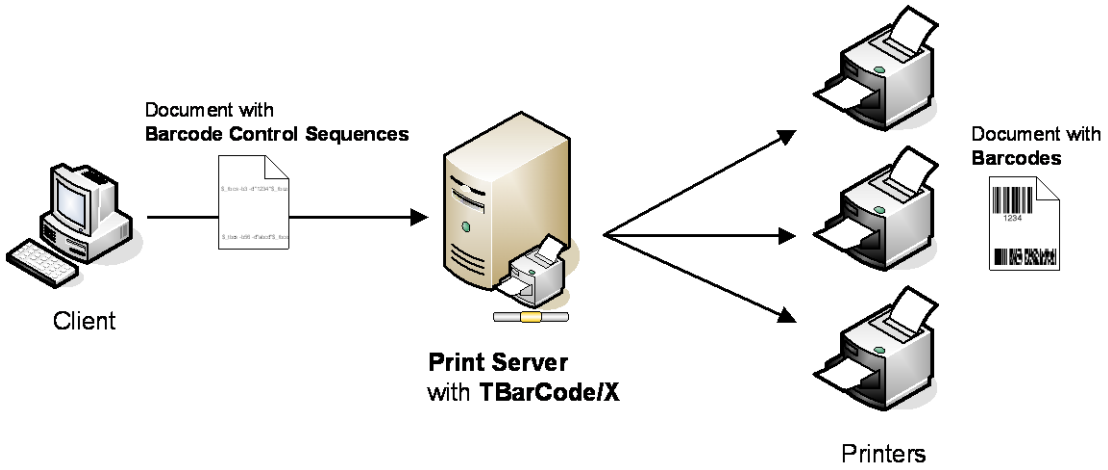

Figure 5: Printing with TBarCode/X

In the following sections you will find information on how to setup **TBarCode/X** for the most important print spooling systems.

8.1 LPRng Printing System

To install **TBarCode/X** as a filter in a print queue, we provide two scripts:

Script	Description
`filterlprng.sh`	This script should be used if the print queue is configured without local filtering. It reads printing data from `stdin`, adds barcodes and sends the result to `stdout`.
`filterlprng_fwd.sh`	This script should be used if the print queue is configured for local filtering. It reads printing data from `stdin`, adds barcodes and forwards the result to the original filter. This script needs to be modified depending on your local configuration.

One of these filter scripts need to be added to the printcap file of the print server.

The following steps are necessary:

1. Open the printcap file (`/etc/printcap`) of the print server.
2. Locate the printer queue for which you want to add **TBarCode/X**. Here is an example what a printcap entry could look like:

```
printer:\
        :sh:\
        :ml=0:\
        :mx=0:\
        :sd=/var/spool/lpd/printer:\
        :af=/var/spool/lpd/printer/printer.acct:\
        :lp=/dev/lp0:\
```

```
        :lpd_bounce=true:\
        :if=/usr/share/printconf/util/mf_wrapper:
```

3. If your entry does not contain the parameter `if=...` then perform step 4, otherwise go to step 5.
4. The parameters `lpd_bounce`, `lpr_bound` and `if` need to be added to the printcap entry. Add the following lines:

```
       …printcap entry… \
       :lpd_bounce=true:\
       :lpr_bounce=true:\
       :if=/usr/local/share/tbarcode9/filterlprng.sh:
```

   Then continue with step 7.
5. Remember the original filter (in our example: /usr/share/printconf/util/mf_wrapper). Change the `if` parameter to

```
       :if=/usr/local/share/tbarcode9/filterlprng_fwd.sh:
```

   The printcap entry of our example would then look like

```
    printer:\
        :sh:\
        :ml=0:\
        :mx=0:\
        :sd=/var/spool/lpd/printer:\
        :af=/var/spool/lpd/printer/printer.acct:\
        :lp=/dev/lp0:\
        :lpd_bounce=true:\
        :if=/usr/local/share/tbarcode9/filterlprng_fwd.sh:
```

6. Now open the script /usr/local/share/tbarcode9/filterlprng_fwd.sh and substitute `path_of_original_filter` (in line 25) with the path and name of the original filter (in our example `/usr/share/printconf/util/mf_wrapper`).
7. Restart the print service (LPD):

```
/etc/init.d/lpd restart
```

8.1.1 Testing the Printer Filter

You can now test the **TBarCode/X** printer filter. Enter the following line (substitute `name_of_printer` with the name of your printer):

```
lpr -P name_of_printer /usr/local/share/tbarcode9/samples/testfile.ps
```

This command should print a simple test file. Check the printout – it should contain several bar-codes.

► Advice: Certain configuration tools might overwrite you changes. So backup your configuration files, as soon as you have done all required changes.

8.2 CUPS Printing System

In CUPS you can install filters for specific mime type conversions. TBarCode Filter can be installed to handle mime types *application/postscript*, *application/pcl*[4] and *application/octet-stream*.

> ► You cannot apply **TBarCode/X** filter on a "raw queue" because such queues will ignore all filters. You have to use queues with "local filtering" using a printer driver (`*.ppd`).

8.2.1 Setting up TBarCode/X Spool Filter for PostScript Output

The following changes have to be made to the MIME type conversion file of CUPS.

1. Open `/etc/cups/mime.convs` or `/usr/share/cups/mime/mime.convs`
2. Search for the line with the *pstops* conversion rule:

```
application/postscript application/vnd.cups-postscript 66 pstops
```

CUPS 1.0/1.1

Replace `pstops` with `/usr/local/share/tbarcode9/filtercups_ps.sh`
The line should look like this:

```
application/postscript application/vnd.cups-postscript 66                    ↵
    /usr/local/share/tbarcode9/filtercups_ps.sh
```

CUPS 1.2.x (and later)

In mime.convs change the line as follows:

```
application/postscript application/vnd.cups-postscript 66 filtercups_ps.sh
```

Retrieve your CUPS installation directory:

```
cups-config --serverbin
```

It should be `/usr/lib/cups`
In this case filters must be in the filter subdir `/usr/lib/cups/filter`

Create a sym link[5] to place the TBarCode/X filter script into this sub dir:

```
ln -s /usr/local/share/tbarcode9/filtercups ps.sh
      /usr/lib/cups/filter/filtercups_ps.sh
```

3. Restart the printing service:

```
/etc/init.d/cups restart
```

8.2.2 Setting up TBarCode/X Spool Filter for PCL Output

Setting up **TBarCode/X** to filter PCL data streams requires a bit more work because CUPS by default does not filter PCL data. We have to add a custom MIME type for PCL documents.

1. Open `/etc/cups/mime.types` or `/usr/share/cups/mime/mime.types`
2. Add the new MIME type `application/pcl` to the list of "Application-generated files…"

[4] application/pcl is a custom mime type based upon application/vnd.cups-raw
[5] If the sym links in CUPS filter directory already exist (as they are created by the setup), use the existing ones.

```
...
application/vnd.hp-HPGLhpgl string(0,<1B>&)\
                        string(0,<1B>E<1B>%0B) \
                        string(0,<1B>%-1B) string(0,<201B>)\
                        string(0,BP;) string(0,IN;) string(0,DF;) \
                        string(0,BPINPS;) \
                        (contains(0,128,<1B>%-12345X) + \
                         (contains(0,1024,"LANGUAGE=HPGL") \
                          contains(0,1024,"LANGUAGE = HPGL")))
application/pcl         (string(0,<1B>E) + !string(2,<1B>%0B)) \
                         string(0,<1B>@) \
                         (contains(0,128,<1B>%-12345X) + \
                          (contains(0,1024,"LANGUAGE=PCL") \
                           contains(0,1024,"LANGUAGE = PCL")))
...
```

The lines that should be added are marked **bold**.

Make sure to insert the new mime type `application/pcl` before `application/vnd.cups-raw` since it needs higher priority and must be handled first. The type `application/vnd.cups-raw` also must remain in the file.

3. **CUPS 1.0.x/1.1.x**
   Open the MIME type conversion file `/etc/cups/mime.convs`
   Add the conversion rule for `application/pcl`

```
application/pdf         application/postscript          33   pdftops
application/postscript  application/vnd.cups-postscript 66   pstops
application/pcl         application/vnd.cups-raw        66                    ↵
    /usr/local/share/tbarcode9/filtercups_pcl.sh
...
```

CUPS 1.2.x (and later)
Open the MIME type conversion file `/etc/cups/mime.convs`
Add the conversion rule for `application/pcl`

```
application/pdf         application/postscript          33   pdftops
application/postscript  application/vnd.cups-postscript 66   pstops
application/pcl         application/vnd.cups-raw        66   filtercups_pcl.sh
...
```

Retrieve your CUPS installation directory:

```
cups-config --serverbin
```

It should be `/usr/lib/cups`
Filters must be then in the filter subdir `/usr/lib/cups/filter`

Create a sym link[6] to place the TBarCode filter script into this sub dir:

```
ln -s /usr/local/share/tbarcode9/filtercups_pcl.sh
      /usr/lib/cups/filter/filtercups_pcl.sh
```

4. Restart the printing service:

```
/etc/init.d/cups restart
```

[6] If the sym links in CUPS filter directory already exist (as they are created by the setup), use the existing ones.

8.3 AIX's Printing System

To install **TBarCode/X** in an AIX printer queue follow these steps:

1. Choose in which printer queue you want to use **TBarCode/X.**
2. Assign **TBarCode/X** as a user-defined filter in the virtual printer definition of this queue:
   - Use the tool "`lsvirprt`" to edit the attributes of the virtual printer definition. The attributes `f1, f2, f3, f4, f5` may specify user-defined filters.
   - Set the value of `f1` to "`/usr/local/share/tbarcode9/tbarcode --filter Parameters`". For example:

```
f1=/usr/local/share/tbarcode9/tbarcode --filter
```

If you want to print and filter barcodes, call `qprt` with the parameter –f1. For example:

```
qprt -PPrinterName -f1 /usr/local/share/tbarcode9/testfile.ps
```

To select **TBarCode/X** permanently, edit the virtual printer definition with `lsvirprt`. Set the value of the attribute `_f` to "1". With this setting all print jobs for this queue will be filtered automatically with **TBarCode/X.**

8.4 HP-UX's Printing System

This section describes how to setup **TBarCode/X** as a filter in the **standard lp spooler** coming with HP-UX 11.xx. Please read Section G.1 "Unix Printing (HP-UX and Solaris)" for background information.

First you should perform a basic test to see if the filter works on your system. These tests are described in Section 6.1.2, "Run TBarCode as Filter".

8.4.1 Spool System

HP-UX 11.xx can use the **lp** spooler or the **HPDPS** spooler (both can be configured with SAM). Below we focus on the SVR4 based **lp** spooler, which is the default printing mechanism in HP-UX 11.

If you have installed LPRng, which is also available for HP-UX, the installation procedure would be the same as for Linux.

HPDPS is also supported by **TBarCode/X**, but the installation is more complex → Please contact our support if you need help.

8.4.2 Using a Local Printer

TBarCode/X can be integrated into the "model files" located in `/usr/lib/lp`. These files are scripts that handle and describe the characteristics supported by a printer. You can either add an own model file to this directory or modify an existing one.

It is very easy to call the filter inside such a script: each time a printout is made the filter will be called (because the script is run for each spool/job file).

Which model file/script is used (and which model file/script has to be modified) depends on the settings within SAM: it is adjusted in the input field "printer model / interface".

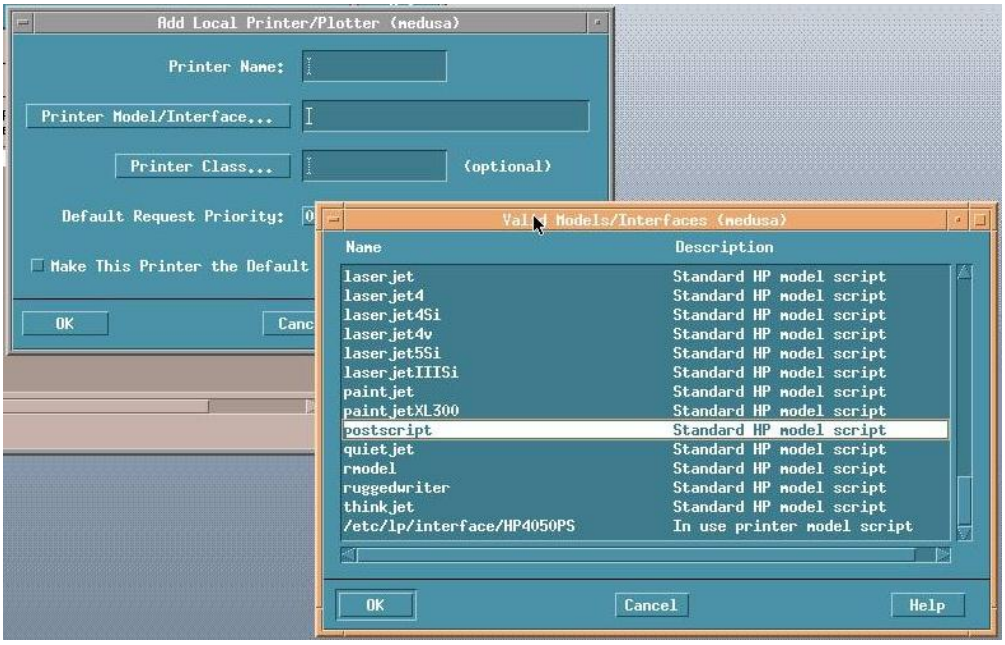

Figure 6: HP-UX Printer Models/Interfaces

You need to add some commands to the model script in order to call **TBarCode/X** – see next section, they are the same as with remote printers.

8.4.3 Using a Remote Printer

For remote printers (LPR) there are other scripts responsible for calling the filter – they are placed in `/etc/lp/interface`.

For instance, if you have a remote printer HP4050PS - the script, which handles the print-out, is located in `/etc/lp/interface/HP4050PS` - please check if you have a script in this place.

8.4.4 Printing Script HP-UX 11.00 or HP-UX 11.23

Edit the script and insert the bold lines before the final rlp command:

```
#############################################################
# start TBarCode filter
#
# create temp file name
INPUT="$(mktemp /tmp/tbarcode.$$.XXXXXX)"
# call tbarcode to insert barcode, output to temp file
/usr/local/share/tbarcode9/tbarcode --filter <$1  >$INPUT
# overwrite original spool file
mv $INPUT $1
# end filter
#############################################################
/usr/sbin/rlp -I$requestid $BSDC $BSDJ $BSDT $BSDi $BSD1 $BSD2 $BSD3 $BSD4 $BSDw ...
```

Result: the filter is called before the spool job is sent to the printer with the `rlp` command

8.4.5 Printing Script HP-UX 11.11

Edit the script and insert the bold lines before the `$Realmodel` command:

```
while :
do
  ###########################################################
  # START TECIT
  # generate a temp filename
  PROCFILE="$(mktemp -d /tmp -p tbarcode)"
  # call barcode engine (-S for SAP output)
```

```
/usr/local/share/tbarcode9/tbarcode --filter <$1 >$PROCFILE
# replace original file with process file
mv $PROCFILE $1
# END   TECIT
##########################################################
#
# Save the stderr messages in a temporary log file
# and discard stdout which is the peripheral output.
$REALMODEL $job $user "$title" $copy "$options" $files | $HPNPF $HPNPFOPT 2>>$LOG >
  /dev/null
```

8.4.6 Other Printing Scripts

Please contact support@tec-it.com if you need help with your printing script.

8.4.7 Make a Test Print

```
lp -d Printer /usr/local/share/tbarcode9/samples/testfile.ps
```

On a PostScript printer the printout should contain several barcodes.

8.5 Solaris Printing System

This section describes how to setup **TBarCode/X** as a filter in the standard `lp` spooler coming with Solaris. Please read section G.1 “Unix Printing (HP-UX and Solaris)” for background information.

First you should perform a basic test to see if the filter works on your system. These tests are described in section 6.1.2 Run TBarCode as Filter.

8.5.1 Spool System Integration

Solaris spool filter integration can be done through modifying the *netstandard interface file*[7] or by registering an *lpfilter* as shown below.

TBarCode/X for Solaris contains two filter definition files:[8]

```
filterlp_ps.fd
filterlp_pcl.fd
```

With these files it is possible to include TBarCode filter into the `lp` spool system. In order to support filtering with remote printer queues we use "Virtual printers" as workaround.

Proceed as described below:

8.5.1.1 Register Filter (Setup)

During the installation phase, the setup program[9] registers automatically the TBarcode/X filters using the following commands:

```
/usr/sbin/lpfilter -f tbarcode ps -F /usr/local/share/tbarcode9/filterlp ps.fd
/usr/sbin/lpfilter -f tbarcode_pcl -F /usr/local/share/tbarcode9/filterlp_pcl.fd
```

- `tbarcode_ps` is the filter used for print jobs in PostSript® printer language.
- `tbarcode_pcl` is the filter used for print jobs in PCL printer language.

[7] For experts only: Please contact TEC-IT support if you want to change the netstandard interface script
[8] Please contact TEC-IT support if the Solaris filter scripts are not part of your installation.
[9] Filter registration by the setup is implemented in TBarCode/X V9.0.4 and later versions.

TIPS:

```
# remove the filter
/usr/sbin/lpfilter -f tbarcode ps -x

# send content of the registered filter to console
/usr/sbin/lpfilter -f tbarcode_ps -l
```

8.5.1.2 Create Virtual Printer

In order to use the filter, create a printer based on the tbarcode input.

```
# Example local printers (PostScript)
lpadmin -p tbcprintlocal1 -v /dev/bpp0 -D "TBarCode Printer on parallel port"
lpadmin -p tbcprintlocal1 -m <lp printer model> -n </path/ppdfile>
lpadmin -p tbcprintlocal1 -I tbarcode_ps -T unknown
/usr/bin/enable tbcprintlocal1
/usr/sbin/accept tbcprintlocal1

lpadmin -p tbcprintlocal2 -v /dev/cua/b -D "TBarCode Printer on serial port"
lpadmin -p tbcprintlocal2 -m <lp printer model> -n </path/ppdfile>
lpadmin -p tbcprintlocal2 -I tbarcode_ps -T unknown
/usr/bin/enable tbcprintlocal2
/usr/sbin/accept tbcprintlocal2

# Example network printer (PCL)
lpadmin -p tbcprintremote -v /dev/null -D "TBarCode network Printer"
lpadmin -p tbcprintremote -o dest=<printer-ip>:<printer-port> -o protocol=tcp -o timeout=5
lpadmin -p tbcprintremote -m <lp printer model> -n </path/ppdfile>
lpadmin -p tbcprintremote -I tbarcode_pcl -T unknown
/usr/bin/enable tbcprintremote
/usr/sbin/accept tbcprintremote

# Real example (PostScript)
lpadmin -p tecitdevel -v /dev/null -D "TBarCode network Printer"
lpadmin -p tecitdevel -o dest=172.16.100.104:9100 -o protocol=tcp -o timeout=5
lpadmin -p tecitdevel -I tbarcode_ps -T unknown
/usr/bin/enable tecitdevel
/usr/sbin/accept tecitdevel
```

8.5.1.3 Print To Filtered Printer

```
# Print the file report.ps to the local Postscript printer
lp -d tbcprintlocal1 -o nobanner report.ps

# Print the file report_v1.ps (contains the TBarCodeX commands in the old V1 format)
# to the local PostScript printer
lp -d tbcprintlocal1 -y v1format -o nobanner report_v1.ps

# Print PCL output to the network PCL printer
lp -d tbcprintremote -o nobanner output.pcl
lp -d tbcprintremote -y v1format -o nobanner output_v1.pcl
```

8.5.2 Print Barcode Filter Test File

```
lp -d PSPrinterName /usr/local/share/tbarcode9/samples/testfile.ps
```

8.5.2.1 Background Information Solaris Printing

http://developers.sun.com/solaris/articles/basicprinting.html

8.6 TBarCode/X with UNISPOOL® (Holland House B.V.)

Create a script `/home/unispool/tbc_filter_script`. The script should have the following content:

```
cat $6 | tbarcode --filter | /home/unispool/cexpand
```

In UNISPOOL® use the filter by calling `/home/unispool/tbc_filter_script`.

8.7 SAP® R/3® and mySAP® Integration

TBarCode/X can be used with SAP systems to generate bar codes during printing. You can get more information about the required configuration steps under the following email addresses:

- sap@tec-it.com
- support@tec-it.com

9 Generating Bitmap Images

There are currently two ways to create barcodes as raster images:

- Direct Bitmap Generation
  TBarCode/X supports built-in bitmap output, but the human readable text will not be drawn. In addition certain barcode symbologies, such as the MAXICODE, are not support for direct bitmap output.

- Indirect Bitmap Generation via PostScript
  Create PostScript output and convert the PostScript output to the desired raster image format. This method requires a bit more work, but there are more possibilities. All barcode symbologies and text output are supported.

9.1 Direct Method: Create Bitmap Images with TBarCode/X

Here is an example how a bitmap barcode can be created:

```
tbarcode --format=IMAGE --imageformat=PNG --output=barcode.png --barcode=20 --data="1234"
--optimalwidth
```

Or using the short-form for the parameters:

```
tbarcode -fIMAGE -iPNG -obarcode.png -b20 -d"1234" -O
```

This will create a barcode where one module (thinnest bar) is exactly one pixel. The parameter `--optimalwidth` (`-O`) ensures that the modules are exactly an integer multiple of pixels (no fractional part). Without this parameter the barcode might not be readable.

TBarCode/X automatically chooses an appropriate width and height for the barcode. If the resulting size of the barcode does not fit your needs, you can specify the width and height yourself: The parameters `--width` and `--height` specify the size of the barcode in millimeters.

The actual size of the bitmap in pixels depends on the image resolution which is set. If nothing is specified a resolution of 72 dpi (dots per inch) is assumed. A custom resolution can be set with the parameter `--dpi`.

For example, the parameters `--width=50 --height=20 --dpi=600` will create a barcode which is 1181 x 472 pixels large. (If this barcode is printed at a resolution of 600 dpi the resulting barcode will be 50 x 20 mm.)

If you specify the parameters `--width` and `--optimalwidth` at the same time, then **TBarCode/X** will to the following: **TBarCode/X** will choose the optimal size which is closest to the specified width. An optimal size is where all bar widths are exactly integer multiples of pixels.

> ► Always use the option `--optimalwidth` (or just `-O`) when creating bitmap barcodes. This will guarantee the readability of the resulting barcodes.

9.1.1 Samples

Here are some more examples:

```
tbarcode -fIMAGE -iPNG -obarcode.png -b71 -O -d"0123456789"
```

creates a small Data Matrix barcode. The resulting bitmap is 12 x 12 pixels large, where a module is exactly one pixel.

```
tbarcode -fIMAGE -iTIF -obarcode.tif -b20 -w50 -h20 --dpi=200 -d"0123456789"
```

creates a Code 128 barcode, which is 50 mm × 20 mm when printed at 200 dpi.

> ► Warning: This barcode might not be readable, because the module width is not aligned with the pixel raster. Use the parameter `--optimalwidth` (or just `-O`). This will ensure that the barcode is perfectly readable. See the next example.

Examples:

```
tbarcode -fIMAGE -iTIF -obarcode.tif -b20 -w50 -h20 --dpi=200 -d"0123456789" -O
```

creates a Code 128 barcode, which fits into an area of 50 mm × 20 mm. The actual size of the resulting bitmap is 45.7 mm × 19.9 mm. This barcodes is guaranteed to be readable.

```
tbarcode -fIMAGE -iTIF -obarcode.tif -b71 --sizemode=MINIMAL --dpi=200 ↵
         --decoder=SOFTWARE -d"DATA 0123456789"
```

creates a barcode that is optimized for 200 dpi and software barcode decoder.

9.2 Indirect Method: Convert PostScript Output to Bitmap

The following steps demonstrate an alternative method how to create barcode bitmap images with **TBarCode/X.** You will have to use this method, if you want to see the barcode text in the bitmap.

1. Generate a new barcode

   ```
   tbarcode -obarcode.eps -b20 -w80 -h50 --fontsize=24 -O -d"Demo123"
   ```

   This creates a barcode of with a size of 80 mm x 50 mm. The parameter `-O` (`--optimalwidth`) ensures the that all bars fit into a 72 dpi raster, which is the native resolution in PostScript.

   Instead of setting the width of the barcode directly, you can also specify the desired module width. For example:

   ```
   tbarcode -obarcode.eps -b20 --modulewidth=0.35278 --fontsize=24 -O -d"Demo123"
   ```

   If you set the module width directly, make sure that the module width is an integer multiple of 0.35278 mm. Because 0.35278 mm matches exactly one dot (pixel) in PostScript.

2. Convert the EPS-file to bitmap format.

   Several programs can be used to convert PostScript (*.eps, *.ps) images to bitmap format. Here are two examples:

   - `convert` is a command line tool that comes with the free ImageMagick® software suite (http://www.imagemagick.org).

     ```
     convert barcode.eps barcode.png
     ```

   - You can use the option `+antialias` disable antialiasing, for example:

     ```
     convert +antialias barcode.eps barcode.png
     ```

   - `gs` is a command line tool that comes with Ghostscript, which is contained in most Linux distributions). The following command creates a black and white PNG image:

     ```
     gs -dNOPAUSE -dBATCH -sDEVICE=pngmono -r72 -g225x143 -sOutputFile=barcode.png ↵
       barcode.eps
     ```

- The parameter `-g225x143` sets the size of the image. The size (“bounding box”) can be determined with:

```
gs -dBATCH -sDEVICE=bbox barcode.eps
```

9.3 Web Applications (PHP)

You can use **TBarCode/X** in dynamic web pages on your Linux server. To create a barcode on demand in your PHP script, you can execute “`tbarcode`” in a shell. The syntax for creating bitmap barcodes (*.JPG, *.GIF, *.PNG, etc.) is described in the previous section.

9.3.1 Display a Barcode in a Browser

Here are two examples how to generate barcodes and display them in a web-application:

9.3.1.1 Example #1

Create the barcode image file with PHP and reference it in your html output:

```
// mypage.php – principle data flow
$tmp_bc_file = get_random_file_name() . ".gif";
$r = shell_execute ("tbarcode … barcode parameters… $tmp_bc_file");
<html><body>
… <img src="/imgpath/$tmp bc file">
</body>

// after a while you need to delete the temporary created image files
// otherwise your hard drive will be flooded with barcode image files
```

9.3.1.2 Example #2

Reference a php script, which creates a barcode image data stream on demand

```
// mypage.php
<img src="SendMeABarcode.php?data=1234">

// SendMeABarcode.php  – principle data flow
header("Content-type: image/JPEG");
$unique_filename = dirname($PATH_TRANSLATED) . "\\" . "~" . uniqid(rand()) . ".jpg";
$r = shell execute ("tbarcode … $data… $unique filename");

// read the whole file and send it to the browser
$fp=fopen($unique_filename,"rb");
fpassthru($fp); // pass through as binary data stream (JPG image format)
flush();
unlink ($unique filename.”.jpg”); // delete file immediately
// make sure that you don’t add unwanted white space outside of the <?  ?> tags
```

Instead of the `shell_execute()` function you could also use `exec()` or `system()`.

9.3.2 Hints for using `shell_execute()` [10]

If you're not getting any output from `echo shellexec( "prog" )` [for instance], at least try `./prog` before bothering with the full path.

Add `2>&1` to the end of your shell command to have STDERR returned as well as STDOUT

```
$shell_return = shell_exec($shell_command." 2>&1");
```

[10] taken from php.net

Note: You can't used `shell_exec()` when safemode = on (it's disabled), instead use `exec()` and copy the needed program into the `/nonexec` directory (by default, set in php.ini).

When running sub processes via `shell_exec` (and maybe others) from Apache/mod_php4, Apache's environment variables don't seem to be passed on to the sub process environment unless you specifically force them by using `putenv` something like this:

```
$remaddr = getenv("REMOTE_ADDR");
putenv("REMOTE ADDR=$remaddr");
shell_exec("/path/to/subprocess");
```

10 Licensing

10.1 License Key and License Types

As long as you have not licensed **TBarCode/X** an additional horizontal bar will be printed across the generated barcodes. Usually this horizontal bar does not affect the readability of the code for evaluation purposes.

Purchasing a license removes this restriction. Please contact TEC-IT for available license modes. Just send an email to office@tec-it.com.

10.2 License File

The license file is named "`license.ini`" and contains the license information and your license key.

Please copy this file into the directory of **TBarCode/X:**

```
/usr/local/share/tbarcode9
```

Or for Mac OS:

```
/tecit/TBarCode9
```

You have to copy this file to each system (client) where you want to use **TBarCode/X**. Overwrite the original (demo) `license.ini` file that was installed during setup.

On systems where **TBarCode/X Daemon** is installed, the background server process needs to be restarted after installing the `license.ini` file. To restart the background server process call

```
/usr/local/share/tbarcode9/tbarcoded --restart
```

You need root privileges to do this. (If you cannot find `tbarcoded`, then you are probably using a **TBarCode/X** version without **TBarCode/X Daemon.** In this case there is no need to restart any process.)

Additional information can be found on our web site http://www.tec-it.com.

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: mailto:barcode@tec-it.com

Web: http://www.tec-it.com

AIX®, AS/400®, OS/400® and PowerPC® are registered trademarks of IBM Corporation.
AMD® and Opteron® are trademarks of Advanced Micro Devices, Inc.
BarSIMM® is a registered trademark of JetMobile, France
Debian® is a registered trademark of Software In The Public Interest, Inc.
The mark FreeBSD is a registered trademark of The FreeBSD Foundation.
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.
HP-UX® and PA-RISC® are registered trademarks of Hewlett-Packard Company.
ImageMagick® is a registered trademark of ImageMagick Studio LLC, P.O. Box 40, Landenberg, PA 19350, United States.
Intel® and Itanium® are registered trademarks of Intel Corporation.
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.
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® , mySAP® are trademarks or registered trademarks of SAP AG in Germany (and in several other countries).
SCO® and SCO OpenServer® are registered trademarks of The SCO Group, Inc. in the United States and other countries.
Solaris® is a registered trademark of Sun Microsystems, Inc.
SPARC® is registered trademark of SPARC International, Inc.
SUSE® is registered trademark of SUSE AG, a Novell business.
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 (mailto:office@tec-it.com).

Appendix A Library Dependencies

A.1 Dependencies

The **TBarCode/X** software uses shared libraries to provide a smaller total distribution size. You have to make sure that all required libraries (dependencies) are installed on your system.

> ► In most cases you need to install the GCC runtime libraries (or a specific version of them). If the correct `libgcc` package has been already installed you may have to adapt the library path - see A.2 Shared Library Path

A.1.1 List Dynamic Dependencies

If some dependencies are not installed or not found, **TBarCode/X** can function incorrectly or fail. In this situation, you must check that all the shared libraries are installed and can be found.

The following commands list the dynamic dependencies of executable files and libraries. Use them for trouble shooting failed dependencies:

- `ldd` on Solaris and Linux systems
- `chatr` on HP-UX systems
- `dump -H` on AIX systems

Here are some examples:

OS	Commands	Samples

Linux
Solaris
	`ldd`	The output of these commands lists the dynamic dependencies and indicates which libraries cannot be found.
		`ldd /usr/local/share/tbarcode9/tbarcode`
		`ldd /usr/local/lib/libtbarcode*`

HP-UX	`chatr`	The output of these commands shows the shared library path and whether the environment variable is enabled or disabled.
		`chatr /usr/local/share/tbarcode9/tbarcode`
		`chatr /usr/local/lib/libtbarcode*`

AIX	`dump -H`	The output of this command lists the dynamic dependencies and indicates which libraries cannot be found. `dump -H /usr/local/share/tbarcode9/tbarcode`

A.1.2 GCC Runtime Libraries

Depending on the downloaded **TBarCode/X** binary (and target platform) there are specific versions of the GCC runtime libraries required. Please download and install the missing gcc runtime libraries by using the information below.

A.1.2.1 GCC for Linux

To find out which gcc version or libgcc version is required for your **TBarCode/X** binary follow the notes in the README file coming with the installation (see installation directory). If there is no readme file available, then please contact support@tec-it.com

A.1.2.2 GCC for AIX

TBarCode/X	Required Runtime Libraries / Download Links
V7.0.4 AIX5.2 PPC V9.0.2 AIX5.2 PPC	▪ libgcc 3.3.2 ▪ libstdcplusplus-3.3.2 TEC-IT Package (untar into root dir /): ▪ http://www.tec-it.com/Download/Unix/AIX/gcc-libs-3.3.2-AIX5.2.tar.gz IBM Package (download libgcc and libstdc++ for AIX 5.2) ▪ ftp://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/RPMS/ppc/gcc/ ()
Others	Please see the README file in the products installation directory (if available) or contact TEC-IT Support (support@tec-it.com).

A.1.2.3 GCC for HP UX

TBarCode/X	Required Runtime Libraries / Download Links
V2.0.2 HP UX 11.11 PA 2.0-64	▪ gcc libs or gcc 4.0.2 hppa64 ▪ http://hpacxx.external.hp.com/gcc/
V7.0.1 HP UX 11.00 PA 2.0	▪ gcc libs 3.3.2 (or gcc 3.3.2) ▪ http://www.tec-it.com/Download/Unix/HPUX/gcc-3.3.2-sd-11.00.depot.gz
V8.0.4 HP-UX 11.11 PA 2.0	▪ gcc libs 3.2.2 ▪ http://www.tec-it.com/download/Unix/HPUX/gcc/gcc-libs-3.2.2-hpux11.11.tar.gz ▪ http://www.tec-it.com/download/Unix/HPUX/gcc/gcc-libs-3.2.2.tar.bz2
V9.0.0 HP UX 11.23 IA64 V9.0.4 HP UX 11.23 IA64	▪ libgcc 4.3.1 (or gcc 4.3.1) ▪ http://hpacxx.external.hp.com/gcc/ (gcc-ia64-4.3.1.depot.gz) ▪ http://www.tec-it.com/download/Unix/HPUX/gcc/libgcc-4.3.1-HPUX11.23-ia64.tar.gz
V9.0.0 HP UX 11.23 IA64 V9.0.4 HP UX 11.23 IA64	▪ libgcc 4.3.1 (or gcc 4.3.1) ▪ http://hpacxx.external.hp.com/gcc/ (gcc-ia64-4.3.1.depot.gz) ▪ http://www.tec-it.com/download/Unix/HPUX/gcc/libgcc-4.3.1-HPUX11.23-ia64.tar.gz
V9.0.4 HP UX 11.31 IA64	▪ libgcc 4.4.0 (or gcc 4.4.0) ▪ http://hpacxx.external.hp.com/gcc/ (gcc-ia64-4.4.0.depot.gz)
Others	Please see the README file in the products installation directory (if available).

Please contact TEC-IT support@tec-it.com if you have problems with downloading or getting the required libraries.

A.1.3 ICONV Libraries

On UNIX systems "Iconv" is used to convert between different character sets. Iconv is available on the command line as well as through the shared library *libiconv*.

Usually the installed iconv libraries contain all character sets and conversion tables required for the basic linear bar code types (Latin-1, ASCII etc).

But specific 2D codes may require a more advanced code page conversion as currently installed on your system.

- QR-Code (Japan) requires CP932 / SHIFT-JIS
- PDF417 requires CP437

In this case you have to extend your iconv installation as follows (3 options):

- Install the missing conversion tables from your system installation base.
- Update your iconv installation (e.g. by downloading a suitable binary)
- Use the Iconv library provided by TEC-IT (if available for your platform)

A.1.3.1 Iconv for AIX

Update your iconv packages (install missing Codepages) as shown in the following example:

```
root@lpar19ml16ed_pub[/mnt/AIX53Base_lpp/installp/ppc] > lslpp -l bos.iconv.ucs.pc
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.iconv.ucs.pc           5.3.0.0  COMMITTED  Unicode Converters for
                                                 Additional PC Code Sets

root@lpar19ml16ed_pub[/mnt/AIX53Base_lpp/installp/ppc] > lslpp -l bos.iconv.ucs.com
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.iconv.ucs.com          5.3.9.0  COMMITTED  Unicode Base Converters for
                                                 AIX Code Sets/Fonts
```

In a special case the math library may also be required: package *bos.adt.libm*

A.1.3.2 Iconv for HP-UX

On HP-UX there may be problems with PDF417 bar code generation due to missing code page(s) in the pre-installed HP-UX iconv library:

- UCS-4BE to cp437 conversion not supported (HPUX 11.23)
- cp1252 to cp437 conversion not supported on (HPUX 11.23)

Solution: TEC-IT provides the following pre-built *libiconv* binaries with full code page support.

TBarCode/X	TEC-IT Iconv Library Download Links
V9.0.2 HP-UX 11.11 PA 2.0	▪ GNU Iconv V1.13 ▪ http://www.tec-it.com/download/Unix/HPUX/dep/tec-iconv-hpux-11.11-pa.tar.gz
V9.0.4 HP UX 11.23 IA64	▪ GNU Iconv V1.11.1 ▪ http://www.tec-it.com/download/Unix/HPUX/dep/tec-iconv-hpux-11.23-ia64.tar.gz
Installation	Untar into root directory to install into `\usr\local\share\teciticonv`

Force the use of the TEC-IT Iconv library with this command (before calling tbarcode):

```
      export LD_PRELOAD=/usr/local/share/teciticonv/lib/libiconv.so
```

A.2 Shared Library Path

TBarCode/X is shipped with the shared library **libtbarcode**[11] (installed in `/usr/local/lib`). **TBarCode/X** also depends on the GCC runtime libraries.

If the dynamic linker can't find one of these libraries, it will abort loading the program. You see messages like:

```
error while loading shared libraries: libtbarcode9.so.0: cannot open shared object file:
No such file or directory
```

► The solution for this problem is to add the library installation path to the shared library search path of your system. See next chapters for how to do this for your operating system.

[11] Some TBarCode/X binaries are statically linked and contain no libtbarcode

A.2.1 Background

Each version of UNIX has its own specific method of searching for libraries; specifically, this process is done by a program called ld.so or (on HP-UX) dld.so or similar.

You can add the library path as system-wide default or you can prefer to use a wrapper script that sets the library path appropriately before launching the executable.

- If you have a compelling reason to add a path to many programs, you can change the system's default search path.
- Setting a specific library path in a wrapper script can provide a workaround in the rare case where two applications require incompatible versions of a library.

In addition to the system search process each program has an *embedded library search path*. The following commands list the embedded library search path of executables and libraries. Use them for trouble shooting:

- Linux

```
readelf -d <binary> | grep RPATH
```

- Solaris, HPUX

```
ldd -s <binary>
```

- AIX

```
rdump -H <binary>
```

► For trouble shooting shared library problems see also A.1.1 List Dynamic Dependencies

A.2.2 Linux

By default the libraries of **TBarCode/X** are installed in `/usr/local/lib`.

You can add this library installation path system-wide to the runtime linker search path with

```
ldconfig /usr/local/lib
```

Alternatively, you can add the path `/usr/local/lib` manually to the ld config file `/etc/ld.so.conf`

In most cases the `/usr/local/lib` installation path is already listed in the ld config file. All you have to do is to rebuild the runtime linker cache with

```
ldconfig
```

Afterwards the **TBarCode/X** libraries should be found.

A.2.2.1 LD_LIBRARY_PATH

You can use the environment variable LD_LIBRARY_PATH if you want to extend the library search path inside of a wrapper script.

Example:

```
export LD_LIBRARY_PATH=/opt/gcc-lib/3.3.2:$LD_LIBRARY_PATH
```

This adds /opt/gcc-lib/3.3.2 to the library path - but only for the current session.

A.2.2.2 Not Finding "libtbarcode" on Debian 4

Enter on the command line

```
echo "/usr/local/lib" > /etc/ld.so.conf.d/tbarcode.conf
ldconfig
```

This adds the installation path to the ld config directory and rebuilds the ld cache.

A.2.3 HP UX

In HP-UX, several places influence the search for dynamic libraries:

- LD_LIBRARY_PATH (environment variable, standard PATH format)
- SHLIB_PATH (environment variable, standard PATH format - from /etc/SHLIB_PATH)
- LD_PRELOAD (environment variable, specifies library names, full path name)
- RPATH (the executable's embedded path, embedded at link time by ld)
- /etc/dld.sl.conf (one library directory per line)

Example:

The following command adds your gcc lib path to the system variable SHLIB_PATH:

```
export SHLIB_PATH=/your-gcc-lib-path:$SHLIB_PATH
```

Next time you call `tbarcode`, the gcc libraries should be found.

The following command adds the `libtbarcode` installation path to the system variable SHLIB_PATH:

```
export SHLIB_PATH=/usr/local/lib:$SHLIB_PATH
```

A.2.3.1 Enable/Disable Search Path

Each executable contains also an embedded path (permanently set during linking).

With the `chatr` command you can control (for a specific executable) how the runtime linker evaluates the library path.

Example:

```
# look at the internal attributes of tbarcode
chatr tbarcode

# specify that the runtime-linker first searches in SHLIB PATH for dependent
# libraries, and then in the embedded path (/usr/local/lib).
chatr +s enable +b enable tbarcode
```

A.2.3.2 SHLIB Path Being Ignored

SHLIB Path being ignored for 'Non-root' users

For looking up shared libraries for setuid applications, the dynamic loader uses only the paths listed in /etc/dld.sl.conf. In other words - the security system won't let SUID root applications look at SHLIB_PATH/LD_LIBRARY_PATH. You must include the paths also in:

```
/etc/dld.sl.conf
```

A.2.4 AIX

On AIX the system variable LIBPATH is used to set the runtime linker search path.

On demand add the installation path of the TBarCode/X libraries (libtbarcode...) as follows:

```
export LIBPATH=/usr/local/lib:$LIBPATH
```

If the gcc runtime libraries are not found, export the LIBPATH environment variable to the correct GCC library version; like so….

```
export LIBPATH=/opt/freeware/lib/gcc-lib/powerpc-ibm-aix5.3.0.0/3.3.2:$LIBPATH
```

A.2.4.1 GCC Lib Conflicts

How to avoid conflicts between different versions of gcc libraries installed at the same time?

You could put the gcc libraries required by TBarCode into a sub folder of tbarcode and adjust the library path before calling tbarcode like so...

```
export LIBPATH=/usr/local/share/tbarcode9/gcc-lib/:/usr/local/lib:$LIBPATH
```

Appendix B: Troubleshooting (FAQ)

B.1 General Questions

B.1.1 Can I use the old parameter format as it was used in *TBarCode for Linux Version 1.x?*

Yes - add the line

```
v1format
```

to the **TBarCode/X** configuration file `tbarcode.conf`. Now all barcode control sequences are interpreted as in **TBarCode** for Linux Version 1.x.

When you have the **TBarCode Daemon** installed, you will have to restart the daemon before change comes into effect.

B.1.2 I have troubles with "`convert`" (gray bars inside the barcode).

The convert utility was originally made for image conversion (photographs) and has a built-in antialias filter. During conversion from 72 dpi EPS files to bitmap files this filter can produce blurred bars with grayscales. There is an option called "`+antialias`" to switch off the filter but due to a bug in some version this option may work or not.

Use the following workaround to get a clear image with convert:

The antialiasing filter doesn't produce gray scaled bars if the resolution of the input file is big enough.

1. Create the barcode 4 times bigger than you need it:
   If you have a module width parameter of `-m0.353` use → `-m1.411`
   If you have a width parameter `-w100` use `-w400` (multiply your value with 4)
   If you have a height parameter of `-h20` use `-h80` (height * 4)
2. During conversion reduce the size to 25%:

   ```
   convert -scale 25\% barcode.eps barcode.png
   ```

3. Now you have the size you want and an image with clear content.

B.1.3 How can I encode an XML string with the TBarCode Command?

The best solution is to store the XML string in a data file and call the **TBarCode/X** command line application with the parameter `--datafile=`*File*. For example:

```
tbarcode -obarcode.ps -b71 --datafile=data.xml
```

B.1.4 How to license the product?

After you have ordered **TBarCode/X** you will receive your license key stored in a file `license.ini`. This file must be copied into the installation directory of **TBarCode/X** – usually `/usr/local/share/tbarcode9`. See Section 10 "Licensing" for more information.

B.1.5 How can I retrieve the hostname for buying a single license?

For a single license we need the hostname of the computer (the client) where you want to use **TBarCode/X.**

To get this hostname enter the following command at the command line (of the target system):

```
hostname
```

B.1.6 TBarCode/X reports that a shared library is missing!

When starting **TBarCode/X** you receive the following error message or similar:

```
error while loading shared libraries: libtbarcode9.so.0: cannot open shared object file:
No such file or directory
```

Solution:

- Make sure that `libtbarcode9.so` is in `/usr/local/lib` or `/usr/lib`. If it is missing, reinstall **TBarCode/X**.
- If the problem still remains run the following command (Linux only):

```
ldconfig /usr/local/lib
```

B.1.7 Where can I read `syslog` messages?

`Syslog` messages will be written to the appropriate file specified in `/etc/syslog.conf`. Normally this is set to `/var/log/messages`.

B.1.8 Why is a horizontal bar drawn across the barcodes?

You are currently working with the restricted demo version. There is either no license file or an invalid license file installed. Please refer to Section 10 “Licensing” or contact us for a valid license file.

B.2 Questions about Filtering/Printing

B.2.1 CUPS: How to tell which filters are in place (and maybe failing?) or missing?

You can switch on the debug mode in CUPS: Open `/etc/cups/cupsd.conf` and add the line

```
LogLevel debug
```

Afterwards restart the CUPS daemon with

```
/etc/init.d/cups restart
```

When you now print a job, a lot of information is written to the CUPS error log file (usually `/var/log/cups/error_log`). You can read which filters and backends are called in which order.

For more information about printing problems see www.linuxprinting.org.

B.2.2 How can I filter ASCII files?

To filter an ASCII text directly the file must be converted into PostScript or PCL format first.

There are several ASCII-to-PostScript filters available (from the Linux/Unix vendors or third party). One of the common tools is “`a2ps`”.

If your printer has no PostScript capability, in most cases it can decode PCL Level 5 (very common, e.g. LaserJet 4/5). In this case the input to our **TBarCode/X** filter must be PCL. Either your application creates PCL input or you find an ASCII-to-PCL filter to do this.

One of the filter products is Magicfilter, which converts ASCII to PCL on demand. This program is standard for several Linux distributions and often installed in the spool system by default.

The converted document can then be passed on to **TBarCode/X. TBarCode/X** adds barcodes in the proper format (either PostScript or PCL).

B.2.3 Why is there no barcode when I'm testing the TBarCode/X with LPRng?

- The print data has to include a barcode control sequence – for example:

```
$_tbcs -fPCL -b20 -m0.254 -h10.2 -d0123456789$_tbce
```

- The filter must be registered in the printcap file (see Section 8 "TBarCode/X as Spool Filter").
- Sometimes lp uses "raw" mode (no filtering) use lpr instead.

B.2.4 How to replace printer specific control sequences with TBarCode control sequences?

All device specific control sequences (for example as used by BarSIMM®) need to be replaced with **TBarCode** control sequences. Here is an example for the symbology "2of5 Interleaved":

	TBarCode/X Control Sequence
Prefix:	`$_tbcs -fPCL -b3 -m0.254 -h13 -tHIDE --origin=BOTTOM -d`
Suffix:	`$_tbce`

Please note: If you want to edit PCL print data directly (e.g. within a spool file during tests), please consider that a standard text editor could corrupt the print data during saving (umlauts, character set differences and CR/LF conversion). Use a hex editor for PCL editing – for example: KHexEdit.

If you omit the parameters `-m` or `-h`, **TBarCode/X** will use default values. With the parameter `--defaultset=1` **TBarCode/X** uses default values which are common for most barcode applications. You can specify this parameter in the **TBarCode/X** configuration file `tbarcode.conf`.

If you don't know, which barcode parameters to use, please contact support@tec-it.com.

B.2.5 How can I filter Easybar control sequences?

Easybar control sequences can be filtered directly with **TBarCode Command**. Use the following syntax:

```
tbarcode -filter --easybar=on <input.pcl >output.pcl
```

When **TBarCode/X** is installed in your print spool system, you can enable *Easybar* support permanently by adding the line

```
easybar=on
```

to the **TBarCode/X** configuration file `tbarcode.conf`.

You should also consider using the option `--remove`. This option removes the *Easybar* control sequence from the filter stream, which is the default behavior of most *Easybar* devices.

B.2.6 How can I print barcodes within a text file?

PCL printers can accept normal text files (ASCII files). Reports and lists are often printed as normal ASCII files.

You can filter a text file with **TBarCode/X** and let it create PCL barcodes. The resulting document will contain the ASCII print data together with PCL commands for drawing barcodes. This document can be sent directly to the printer. But you need to ensure that this document is sent directly to the printer without going through the Unix standard spool filters. The spool filter could convert the barcode drawing commands back to normal text.

Read the next FAQ item for additional information.

B.2.7 How can I send a file without modification to a printer?

How can I avoid that my file is processed by spool filters (e.g. the “magic filter”)?

This can be achieved with a remote queue. In order to pass the file directly to this queue, you have to use `lpr` with parameter `-b`.

For example:

```
lpr -P PCLQueue@remotehost -b file.pcl
```

Here is an example, how to filter a text file with the **TBarCode/X** command line application and then send it directly to a printer.

```
tbarcode --filter <file.txt >file_with_barcodes.pcl
lpr -PHP4050PCL@karthago -b file_with_barcodes.pcl
```

B.2.8 LPRng Spool System: How can I find out what data the printer gets from the queue/spooler?

You can stop an individual printer with “`lpc stop`“:

```
lpc stop PrinterXYZ
```

When a document is printed on the print queue `PrinterXYZ`, the print job is created but not sent to the printer. The print job can be found in the spool directory of the printer – for example: `/var/spool/lpd/PrinterXYZ`.

Copy this data to analyze the print job. When you restart the print queue all pending print jobs are processed.

```
lpc start PrinterXYZ
```

B.3 Where I can get more help?

Your question is not listed here? Please, contact us (see Section, 11 “Contact and Support Information”). We do our best to support our customers.

Appendix C: Barcode Parameters

C.1 Barcode Symbologies

These are the barcode symbologies that are currently supported by **TBarCode/X.** The barcode symbology can be set with the parameter `--barcode=`*ID*. For example: `--barcode=1` sets the barcode type “Code 11”.

Remark: The enumeration values, such as `eBC_Code11`, are only relevant for programmers who want to use the **TBarCode/X Library** directly. They are irrelevant for users of the **TBarCode/X** command line application.

> ► For more detailed information on supported barcode types, please refer to the “Barcode Reference” which is available as separate document. The Barcode Reference can be downloaded from www.tec-it.com.

Column descriptions:

ID:	Internal barcode ID. If not supported in the current version marked with *
Barcode Name:	Name of the barcode symbology
Print Ratio:	Standard Print Ratio of the barcode. Predefined corresponding to the barcode symbology.
Ratio Format String:	Format of the Print Ratio. Helpful to understand the definition of the Print Ratio. xB (1B, 2B, ...) width of the single Bars xS (1S, 2S, ...) width of the single Spaces (also called gaps)
Check-Digit:	Enumeration of the pre-selected check digit method for each barcode symbology.

ID	Barcode Name	Print Ratio	Ratio Format String (Ratio Hint)	Default Check Digit
0	Not a valid type	-------	--------	
1	Code 11	1:2.24:3.48:1:2.24	1B:2B:3B:1S:2S	eCDNone
2	Code 2 of 5 (Standard)	1:3:4.5:1:3	1B:2B:3B:1S:2S	eCDNone
3	Interleaved 2 of 5 Standard	1:3:1:3	1B:2B:1S:2S	eCDNone
4	Code 2 of 5 IATA	1:3:1	1B:2B:1S	eCDNone
5	Code 2 of 5 Matrix	1:3:4.5:1:3	1B:2B:3B:1S:2S	eCDNone
6	Code 2 of 5 Data Logic	1:3:1:3	1B:2B:1S:2S	eCDNone
7	Code 2 of 5 Industrial	1:3:1	1B:2B:1S	eCDNone
8	Code 3 of 9 (Code 39)	1:3:1:3	1B:2B:1S:2S	eCDNone
9	Code 3 of 9 (Code 39) ASCII	1:3:1:3	1B:2B:1S:2S	eCDNone
10	EAN8	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN8
11	EAN8 - 2 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN8
12	EAN8 - 5 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN8
13	EAN13	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
14	EAN13 - 2 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
15	EAN13 - 5 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
16	EAN128	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
17	UPC 12 Digits	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA

18	Codabar (2 widths)	1:3:1:3	1B:2B:1S:2S	eCDNone
19*	Codabar (18 widths)			----
20	Code128 automatic subset switching / auto compress (Code128 A, B, C see below!)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
21	Deutsche Post Leitcode	1:3:1:3	1B:2B:1S:2S	eCDDPLeit
22	Deutsche Post Identcode	1:3:1:3	1B:2B:1S:2S	eCDDPIdent
23*	Code 16K			----
24*	Code 49			----
25	Code 93	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCD2Mod47
26	Identical to eBC_UPCA	-------		----
27*	UPCD1			----
28	Flattermarken	1:1	1B:1S	eCDNone
29	RSS-14	1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9	1B:2B:3B:4B:5B:6B:7B:8B:9B:1S:2S:3S:4S:5S:6S:7S:8S:9S	eCDNone
30	RSS Limited	1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9	1B:2B:3B:4B:5B:6B:7B:8B:9B:1S:2S:3S:4S:5S:6S:7S:8S:9S	eCDNone
31	RSS Expanded	1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9	1B:2B:3B:4B:5B:6B:7B:8B:9B:1S:2S:3S:4S:5S:6S:7S:8S:9S	----
32	Telepen Alpha	1:3:1:3	1B:2B:1S:2S	eCDNone
33	UCC128 (= EAN128)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
34	UPC A	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA
35	UPC A – 2 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA
36	UPC A – 5 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA
37	UPC E	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCE
38	UPC E – 2 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCE
39	UPC E – 5 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCE
40	PostNet-5 (ZIP 5 digits)	1:1	1B:1S	eCDPostNet
41	PostNet-6 (ZIP 5 digits + check digit)	1:1	1B:1S	eCDPostNet
42	PostNet -9 (ZIP + 4)	1:1	1B:1S	eCDNone
43	PostNet-10 (ZIP + 4 + check digit)	1:1	1B:1S	eCDPostNet
44	PostNet-11 (ZIP + 4 + 2)	1:1	1B:1S	eCDPostNet
45	PostNet -12 (ZIP + 4 + 2+ check digit)	1:1	1B:1S	eCDPostNet
46	Plessey Code	1:2:1:2	1B:2B:1S:2S	eCDPlessey
47	MSI Code	1:2:1:2	1B:2B:1S:2S	eCDMSI1
48	SSCC18	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod10
49*	FIM			
50	LOGMARS	1:3:1:3	1B:2B:1S:2S	eCDNone
51	Pharmacode One-Track	1:3:2:4:2:3	1B:2B:1C:2C:1S:2S	eCDNone
52	Pharmazentralnumber	1:2.5:1:2.5	1B:2B:1S:2S	eCDPZN
53	Pharmacode Two-Track	1:1	1B:1S	eCDNone
54*	General Parcel			
55	PDF417	1:2:3:4:5:6:7:8:1:2:3:4:5:6	1B:2B:3B:4B:5B:6B:7B:8B:1S:2S:3S:4S:5S:6S	eCDNone
56	PDF417 Truncated	1:2:3:4:5:6:7:8:1:2:3:4:5:6	1B:2B:3B:4B:5B:6B:7B:8B:1S:2S:3S:4S:5S:6S	eCDNone
57	MaxiCode	1:1		----

58	QR-Code	1:1	(1B:1S)	----
59	Code128 (Subset A)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
60	Code128 (Subset B)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
61	Code128 (Subset C)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
62	Code 93 Ascii	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCD2Mod47
63	Australian Post Standard Customer	1:1	1B:1S	eCDNone
64	Australian Post Customer 2	1:1	1B:1S	eCDNone
65	Australian Post Customer 3	1:1	1B:1S	eCDNone
66	Australian Post Reply Paid	1:1	1B:1S	eCDNone
67	Australian Post Routing	1:1	1B:1S	eCDNone
68	Australian Post Redirection	1:1	1B:1S	eCDNone
69	ISBN Code (=EAN13P5)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
70	Royal Mail 4 State (RM4SCC)	1:1	1B:1S	eCDNone
71	Data Matrix	1:1		----
72	EAN-14	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN14
73*	Codablock-E			eCDCodablockE
74	Codablock-F	1:2:3:4:1:2:3:4	1B:2B:1S:2S	eCDCodablockF
75	NVE-18	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod10
76	Japanese Postal Code	1:1	1B:1S	eCDNone
77	Korean Postal Authority Code	1:3:4	1B:1S:2S	eCDMod10Kor
78	RSS-14 Truncated	1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9	1B:2B:3B:4B:5B:6B:7B:8B:9B:1S:2S:3S:4S:5S:6S:7S:8S:9S	eCDNone
79	RSS-14 Stacked	1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9	1B:2B:3B:4B:5B:6B:7B:8B:9B:1S:2S:3S:4S:5S:6S:7S:8S:9S	eCDNone
80	RSS-14 Stacked Omnidirectional	1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9	1B:2B:3B:4B:5B:6B:7B:8B:9B:1S:2S:3S:4S:5S:6S:7S:8S:9S	eCDNone
81	RSS Expanded Stacked	1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9	1B:2B:3B:4B:5B:6B:7B:8B:9B:1S:2S:3S:4S:5S:6S:7S:8S:9S	eCDNone
82	Planet 12 digits	1:1	1B:1S	eCDMod10Pla
83	Planet 14 digits	1:1	1B:1S	eCDMod10Pla
84	Micro PDF417	1:2:3:4:5:6:1:2:3:4:5:6	1B:2B:3B:4B:5B:6B:1S:2S:3S:4S:5S:6S	eCDNone
85	USPS OneCode 4-State Customer Barcode	1:1	1B:1S	eCDNone
86	Plessey Code with bidirectional reading support	1:2:3:1:2	1B:2B:3T:1S:2S	eCDPlessey
87	Telepen	1:3:1:3	1B:2B:1S:2S	eCDNone
88	GS1-128 (EAN/UCC-128)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDNone
89	ITF-14	1:3:1:3	1B:2B:1S:2S	eCDNone
90	KIX	1:1	1B:1S	eCDNone
91	BC412			----
92	Aztec Code	1:1	1B:1S	eCDNone
93	DAFT Code	1:1	1B:1S	eCDNone
94	Italian Postal 2 of 5	1:3:1:3	1B:2B:1S:2S	eCDNone
95	Italian Postal 3 of 9	1:3:1:3	1B:2B:1S:2S	eCDNone
96	DPD	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDNone
97	Micro QR-Code	1:1	1B:1S	eCDNone
98	HIBC LIC 128	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod43
99	HIBC LIC 39	1:3:1:3	1B:2B:1S:2S	eCDMod43

100	HIBC PAS 128	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod43
101	HIBC PAS 39	1:3:1:3	1B:2B:1S:2S	eCDMod43
102	HIBC LIC Data Matrix	1:1	1B:1S	eCDMod43
103	HIBC PAS Data Matrix	1:1	1B:1S	eCDMod43
104	HIBC LIC QR-Code	1:1	1B:1S	eCDMod43
105	HIBC PAS QR-Code	1:1	1B:1S	eCDMod43
106	HIBC LIC PDF417	1:2:3:4:5:6:7:8:1:2:3:4:5:6	1B:2B:3B:4B:5B:6B:7B:8B:1S:2S:3S:4S:5S:6S	eCDMod43
107	HIBC PAS PDF417	1:2:3:4:5:6:7:8:1:2:3:4:5:6	1B:2B:3B:4B:5B:6B:7B:8B:1S:2S:3S:4S:5S:6S	eCDMod43
108	HIBC LIC MicroPDF417	1:2:3:4:5:6:1:2:3:4:5:6	1B:2B:3B:4B:5B:6B:1S:2S:3S:4S:5S:6S	eCDMod43
109	HIBC PAS MicroPDF417	1:2:3:4:5:6:1:2:3:4:5:6	1B:2B:3B:4B:5B:6B:1S:2S:3S:4S:5S:6S	eCDMod43
110	HIBC LIC Codablock-F	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod43
111	HIBC PAS Codablock-F	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod43

Table 27: Barcode Symbologies

C.2 Check Digit Methods

The check digit calculation method can be set with the parameter `--checkdigit=`*Number*. For example: `--checkdigit=2` sets the Modulo 10 check digit method.

Index	Check Digit Calculation Methods	Enumeration (C/C++)
0	No check digit will be computed	eCDNone
1	Default check digit of barcode type will be used (see table in C.1)	eCDStandard
2	Modulo 10 check digit (usually used with Interleaved 2of5)	eCDMod10
3	Modulo 43 check digit (suggested for Code39, 1 check digit)	eCDMod43
4	Modulo 47 (2 check digits)	eCD2Mod47
5	DP Leitcode check digit	eCDDPLeit
6	DP Identcode check digit	eCDDPIdent
7	Code11 (1 check digit)	eCD1Code11
8	Code11 (2 check digits)	eCD2Code11
9	USPS Postnet check digit	eCDPostnet
10	MSI (1 digit) check digit	eCDMSI1
11	MSI (2 digits) check digit	eCDMSI2
12	Plessey check digit	eCDPlessey
13	EAN 8 check digit	eCDEAN8
14	EAN 13 check digit	eCDEAN13
15	UPC A check digit	eCDUPCA
16	UPC E check digit	eCDUPCE
17	EAN 128 internal method (Modulo 103) check digit	eCDEAN128
18	Code 128 internal method (Modulo 103) check digit	eCDCode128
19	Royal Mail 4 State check digit	eCDRM4SCC
20	PZN Mod-11 check digit	eCDPZN
21	Mod-11 (Weighting = 7) check digit	eCDMod11W7
22	EAN 14 (GTIN) check digit	eCDEAN14
23	Korean Postal Authority – Modulo 10 check digit	eCDMod10Kor
24	Planet - Modulo 10 check digit	eCDMod10Pla
25	Italian Postal 2/5 (Modulo 10 based) check digit	eCDMod10ItlPst25
26	DPD Barcode Modulo 36 (ISO/IES 7064) check digit	eCDMod36

27	Codabar Modulo 16 check digit	eCDMod16

Table 28: Check Digit Methods and Enumerators

C.3 PDF417 Parameters

C.3.1 Encoding Mode

The PDF417 encoding mode cane be set with the parameter `--PDFmode =`*Index.*

Index	Mode
0	Normal/default encoding
1	Binary Compaction

Table 29: PDF417 Encoding Mode

C.4 Micro PDF417 Parameters

C.4.1 Version (Symbol Sizes)

This table shows the possible user defined symbol sizes for Micro PDF-Code. The symbol size can be defined by the parameter `--MPDFversion=`*Index*.

Index	Symbol size (rows x cols)	Index	Symbol size (rows x cols)
0	automatic calculation	18	3 x 15
1	1 x 11	19	3 x 20
2	1 x 14	20	3 x 26
3	1 x 17	21	3 x 32
4	1 x 20	22	3 x 38
5	1 x 24	23	3 x 44
6	1 x 28	24	4 x 4
7	2 x 8	25	4 x 6
8	2 x 11	26	4 x 8
9	2 x 14	27	4 x 10
10	2 x 17	28	4 x 12
11	2 x 20	29	4 x 15
12	2 x 23	30	4 x 20
13	2 x 26	31	4 x 26
14	3 x 6	32	4 x 32
15	3 x 8	33	4 x 38
16	3 x 10	34	4 x 44
17	3 x 12		

Table 30: Micro PDF417 Symbol Sizes

C.4.2 Mode

The Micro PDF417 mode can be set with the parameter `--MPDFmode=`*Index*.

Index	Format
0	Default format
1	UCC/EAN/GS1-128 Emulation
2	Code 128 Emulation
3	Code 128/FNC2 Emulation
4	Linked UCC/EAN/GS1-128
5	05 Macro
6	06 Macro
7	CC-A Data Mode
8	CC-B Prefix

Table 31: Micro PDF417 Modes

C.5 Data Matrix Parameters

C.5.1 Symbol Sizes

The user-defined symbol sizes for Data Matrix can be set with the parameter `--DMsize=`*Index*.

Index	Symbol size (rows x cols)	Index	Symbol size (rows x cols)
0	automatic calculation	16	64 x 64
1	10 x 10	17	72 x 72
2	12 x 12	18	80 x 80
3	14 x 14	19	88 x 88
4	16 x 16	20	96 x 96
5	18 x 18	21	104 x 104
6	20 x 20	22	120 x 120
7	22 x 22	23	132 x 132
8	24 x 24	24	144 x 144
9	26 x 26	25	8 x 18
10	32 x 32	26	8 x 32
11	36 x 36	27	12 x 26
12	40 x 40	28	12 x 36
13	44 x 44	29	16 x 36
14	48 x 48	30	16 x 48
15	52 x 52		

Table 32: Data Matrix Symbol Sizes

C.5.2 Format

The Data Matrix format can be set with the parameter `--DMformat=`*Index*.

Index	Format
0	Default format
1	UCC/EAN
2	Industry
3	Macro 05
4	Macro 06

Table 33: Data Matrix Formats

C.6 MaxiCode Parameters

C.6.1 Mode

This table shows the possible modes for MaxiCode. The mode can be defined by the parameter `--MCmode=`*Index*.

Index	Mode
2	SCM Numeric
3	SCM Alphanumeric
4	Default Mode
5	Full EEC

Table 34: MaxiCode Modes

C.7 QR-Code Parameters

C.7.1 Version (Symbol Sizes)

This table shows the possible user defined symbol sizes for QR-Code. The symbol size can be defined by the parameter `--QRversion=`*Index*.

Index	Symbol size (rows x cols)	Index	Symbol size (rows x cols)
0	automatic calculation	21	101 x 101
1	21 x 21	22	105 x 105
2	25 x 25	23	109 x 109
3	29 x 29	24	113 x 113
4	33 x 33	25	117 x 117
5	37 x 37	26	121 x 121
6	41 x 41	27	125 x 125
7	45 x 45	28	129 x 129
8	49 x 49	29	133 x 133
9	53 x 53	30	137 x 137
10	57 x 57	31	141 x 141
11	61 x 61	32	145 x 145
12	65 x 65	33	149 x 149
13	69 x 69	34	153 x 153
14	73 x 73	35	157 x 157
15	77 x 77	36	161 x 161
16	81 x 81	37	165 x 165
17	85 x 85	38	169 x 169
18	89 x 89	39	173 x 173
19	93 x 93	40	177 x 177
20	97 x 97		

Table 35: QR-Code Symbol Sizes

C.7.2 Format

This table shows the possible formats for QR-Code barcodes. The format can be defined by the control sequence `--QRformat=`*Index*.

Index	Format
0	default format
1	UCC/EAN
2	Industry

Table 36: QR-Code Format Options

C.7.3 Error Correction Level

This table shows the possible Error Correction Levels for QR-Code barcodes. The Error Correction Level can be defined by the parameter `--QRecl=`*Index*.

Index	Error Correction Level
0	Low
1	Medium
2	Quartile (Default)
3	High

Table 37: QR-Code Error Correction Levels

C.8 Micro QR-Code Parameters

C.8.1 Version (Symbol Sizes)

This table shows the possible user defined symbol sizes for Micro QR-Code. The symbol size can be defined by the parameter `--MQRversion=`*Index*.

Index	Symbol size (rows x cols)	Index	Symbol size (rows x cols)
0	automatic calculation	3	15 x 15
1	11 x 11	4	17 x 17
2	13 x 13		

Table 38: Micro QR-Code Symbol Sizes

C.8.2 Error Correction Level

This table shows the possible Error Correction Levels for Micro QR-Code barcodes. The Error Correction Level can be defined by the parameter `--MQRecl=`*Index*.

Index	Error Correction Level
0	Low
1	Medium
2	Quartile
3	High (not used)

Table 39: QR-Code Error Correction Levels

C.9 Codablock-F Parameters

C.9.1 Format

This table shows the possible formats for Codablock-F barcodes. The format can be defined by the parameter `--CBformat=`*Index*.

Index	Format
0	default format
1	UCC/EAN

Table 40: Codablock-F Parameters

C.10 Aztec Code Parameters

C.10.1 Symbol Sizes

This table shows the possible user defined symbol sizes for Aztec Code. The symbol size can be defined by the parameter `--ACsize=`*Index*.

Index	Symbol size (rows x cols)	Index	Symbol size (rows x cols)
0	automatic calculation	17	83 x 83
1	15 x 15	18	87 x 87
2	19 x 19	19	91 x 91
3	23 x 23	20	95 x 95
4	27 x 27	21	101 x 101
5	31 x 31	22	105 x 105
6	37 x 37	23	109 x 109
7	41 x 41	24	113 x 113
8	45 x 45	25	117 x 117
9	49 x 49	26	121 x 121
10	53 x 53	27	125 x 125
11	57 x 57	28	131 x 131
12	61 x 61	29	135 x 135
13	67 x 67	30	139 x 139
14	71 x 71	31	143 x 143
15	75 x 75	32	147 x 147
16	79 x 79	33	151 x 151
And 3 special sizes, usually used only for reader programming			
34	19 x 19 (reader progr.)	36	27 x 27 (reader progr.)
35	23 x 23 (reader progr.)		

Table 41: Aztec Code Symbol Sizes

C.10.2 Format

This table shows the possible formats for Aztec Code barcodes. The format can be defined by the control sequence `--ACformat=`*Index*.

Index	Format
0	default format
1	UCC/EAN
2	Industry

Table 42: Aztec Code Format Options

C.11 Encoding Bytes and Control Characters in Input Data

If you want to use non-printable or special characters in your barcode data, you have to use "Escape Sequences". These sequences start with a backslash ('\') followed by the sequence (see table below). You can use them also for encoding binary data (bytes) in your barcode, but only if the symbology offers this feature (e. g. PDF417 or Data Matrix).

► If you want to use escape sequences in your input data, put the data string into single quotation marks (like '123\F') and enable translation of escape sequences with --translation=on.

C.11.1 Implemented Escape Sequences

Escape-Sequence	Description
\a	Bell (alert)
\b	Backspace
\f	Form feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\\	The backslash \ itself
\0*ooo*	ASCII-character in octal notation: ooo … octal digits (0..7)
ddd	ASCII-character in decimal notation: ddd … decimal digits (0..9)
\x*hh*	For encoding bytes or ASCII-characters in hexadecimal notation *hh* … hexadecimal digits (0..F)
\F	FNC1 or Gs (\x1d), used in UCC/EAN codes as field separator
\E	ECI (Extended Character Interpretation), used in 2D codes like MaxiCode, Data Matrix and QR-Code. Is used for switching between various code pages (multiple character sets) – contact us to get further information.
\EB, \EE	Special ECI identifiers for nesting ECIs. \EB (ECI Begin) opens a nesting level; \EE (ECI End) closes it. Used in QR-Code
\G	GLI (Global Language Identifier), similar to ECI, but only used in PDF417.

Table 43: Implemented Escape Sequences

► Please keep in mind that when translation of escape sequences is enabled, you cannot code a backslash ("\") directly. Use "\\" instead.

Please refer to the "Barcode Reference" (http://www.tec-it.com ► *Support* ► *Knowledge Base*) for more information.

C.11.2 Encoding Bytes

With \x*hh* you can encode Bytes in hexadecimal notation, e.g. '\x01\xFF' encodes the Byte 1 and 255.

Note when using from the command line: Put the input data into single quotes, otherwise you need to encode a double-backslash ('\\') to get a single one.

C.11.3 Symbology Specific Control Characters

If you have enabled translation of Escape sequences (parameter --translation=on) you can encode the following control characters (barcode type dependent).

The input data must contain the escape sequence that corresponds to the control character.

Note when using from the command line: Put the input data into single quotes (e.g. `'123\210456'`), otherwise you would need a double-backslash (like `123\\210456`).

Control character	Escape Sequence	Barcode type(s)
FNC1	\210	Code128, EAN128, UCC128
FNC2	\211	Code128, EAN128, UCC128
FNC3	\212	Code128, EAN128, UCC128
FNC4	\213	Code128, EAN128, UCC128
DC1	\x11	Code93, Code93Ext
DC2	\x12	Code93, Code93Ext
DC3	\x13	Code93, Code93Ext
DC4	\x14	Code93, Code93Ext
Rs	\x1e	MaxiCode (Mode 3,4 SCM)
Gs	\x1d	MaxiCode (Mode 3,4 SCM)
Eot	\x04	MaxiCode (Mode 3,4 SCM)

Table 44: Extended Escape Sequences

C.12 Formatting Barcode Data

The Format string specifies how the input data should be processed prior to encoding it (please do not mix up the *Format* with the *Ratio Format*). Placeholders in the specified format string can be mixed with constant data characters to build a final barcode data string. Also control characters are supported. With this feature it is possible to:

- Select subsets in Code 128, EAN 128 and UCC 128 (even within the code!).
- Select the required start/stop character for CODABAR.
- Change the position of the check digit.
- For MaxiCode: Set the values of Date, Preamble, and Service Class, Postal- and Country code directly in the barcode data (in conjunction with special escape sequences).

The placeholders are as follows:

Placeholder character	Description
#	Stands for the next data character of the input data (property `Text`)
&	Stands for all remaining data characters in the input data (property `Text`)
^	Stands for the next check digit (use only if check digits will be computed!) ▪ TBarCode 6 (or earlier) computes the check digit for all characters in the input data. ▪ TBarCode 8 (or later) only uses input data left of the check digit placeholder for check digit computation (see examples below!).
A	Switch to Subset A (used in: Code 128, EAN 128, UCC 128) Start- or stop character A (only in: CODABAR)
B	Switch to Subset B (used in: Code 128, EAN 128, UCC 128) Start- or stop character B (only in: CODABAR)
C	Switch to Subset C (used in: Code 128, EAN 128, UCC 128) Start- or stop character C (only in: CODABAR)
D	Start- or stop character D (only in: CODABAR)
S	Only for MaxiCode: enables setting the values of Date, Preamble, Service Class, Postal- and Country-Code directly in the barcode data (only in conjunction with escape sequences).
J	For Japanese Postal codes: the Address B data fields can be automatically compressed, i.e. Japanese symbols are converted into ASCII symbols.

Table 45: Format Placeholders

Examples:

Input data	Barcode type	Format string	Data used for encoding	Notes
123	Irrelevant		123	
123	Irrelevant	5&	5123	
123	Irrelevant	&6	1236	
123	Irrelevant	q#w#e#	q1w2e3	
123	Irrelevant	#q&	1q23	
123	Irrelevant	&^	123c	
123	Irrelevant	^&	c123	This format string may be used for TBarCode 6 (or earlier). – TBarCode 8 always returns 0 in this case.
12345	Irrelevant	####^#	1234c5	When using Modulo 10 for check digit calculation, c will be ▪ Mod-10 (12345) = 5 for TBarCode 6 (or earlier). ▪ Mod-10 (1234) = 0 for TBarCode 7 (or later).
Hello	Code 128	A&	Hello	
Hello	Code 128	A##B&	Hello	
Hello4711	Code 128	A##B&	Hello4711	
Hello4711	Code 128	A##B###C&	Hello4711	

Table 46: Format Examples

red characters represented in subset A
gray characters represented in subset B
green characters represented in subset C
c represents the place of the check digit

C.13 PCL Font Numbers

Use these font numbers in combination with the parameter `--font=`*Number*.

Typeface Family	PCL Number
Albertus	4362
Antique Olive	4168
Claredon	4140
Coronet	4116
Courier	4099
Garamond Antiqua	4197
Letter Gothic	4102
Marigold	4297
CG Omega	4113
CG Times	4101
Univers	4148

Table 47: PCL Font Numbers

Appendix D: Using Version 1.x Format

TBarCode for Linux/Unix Version 1.x was the predecessor of **TBarCode/X Version 2.0 (and newer)**.

The format of the required command line parameters and barcode control sequences has changed from version 1.x to current version of **TBarCode/X**. But **TBarCode/X** can be run in a compatibility mode where it supports the old barcode control sequences.

Here is an example of an old barcode control sequence:

```
$_tbcs b55 n w40 h20 r90 d123abc$_tbce
```

In **TBarCode/X** the same barcode can be created with the following barcode control sequence:

```
$_tbcs -b55 -thide -w40 -h20 -r90 -d123abc$_tbce
```

The parameter `--v1format` enables the compatibility mode: When this parameter is set, all barcode control sequences are interpreted as in **TBarCode for Linux/Unix** 1.x. It is best to specify the parameter `--v1format` in the **TBarCode/X** configuration file `tbarcode.conf`. But be aware that you cannot mix old and new barcode control sequences.

- Use `v1format` if you upgrade from version 1.x to the actual version and if you don't want to change the *Filter Control Sequences* in your application.
- In SAP® R/3® due to the limitation of length for Print controls the v1format is used because it needs less space.
- If possible, we recommend using the new parameters, because they are more flexible and more intuitive.

D.1 Overview V1 Format

Below you find a short overview about the most important parameters in Version 1.x format. For the detailed list see the manual of **TBarCode for Linux/UNIX Version 1.x**

Parameters of Version 1.x	Description
`$_tbcs`	Marks the beginning of the sequence (used with filter)
`$_tbce`	Marks the end of the sequence (used with filter)
`dContent`	Content = data of barcode; must be the last parameter before $_tbce
`xPosition`	Absolute x position in mm (* see above)
`yPosition`	Absolute y position in mm (* see above)
`wWidth`	Width of barcode in mm (e.g. w50 or w53.12)
`hHeight`	Height of barcode in mm
`ot`	Orientation: Top (x/y-Position sets the upper left corner of the barcode. Default in PostScript.)
`ob`	Orientation: Bottom (x/y-Position sets the lower left corner of the barcode. Default in PCL.)
`bBarcodeNo`	Number of barcode (see Barcode Types in the Appendix)
`cMethodNo`	Number of check digit calculation method
`rRotation`	Rotation in degrees (0, 90, 180 or 270)
`T(on|off)`	Show human readable text.
`n`	Do not print human readable text (same as Toff).
`a`	Print the human readable text above the barcode (default is below)
`s(on|off)`	Translate escape sequences in input data

A(on|off)	Turn auto correct on or off
gGuardWidth	Width of guarding line in mm
fFontname	Font name in PostScript or Typeface Family Value in PCL PostScript: Times-Roman, Courier, Helvetica, … PCL: 4101, 4099, 16602, … If the number from the f parameter is 1000 or bigger than 1000, it will be identified as PCL-Font number.
fFontsize	Size of font in points.
tFormat	Output format: PS (=PostScript, default) or PCL
iDistance	Text distance in mm
NHeight	Notch height in mm
mModWidth	Module width (narrow bar width) in µm (= 1/1000 mm), if used the W parameter for the symbol width is irrelevant.
RRatio	Print ratio
FFormat	Format string used for formatting barcode data prior to printing it
O	Calculate optimal width of barcode
QhHorzQZ	Horizontal quiet zone in mm (e.g. Qh1.34 or Qh5). The specified quiet zone is a blank space, which is added to the left and right side of the symbol. Usually the Quiet zone should be 10 times the module width or higher.
QvVertQZ	Vertical quiet zone in mm (e.g. Qv1.34 or Qv5). The specified quiet zone is a blank space, which is added to the top and bottom of the symbol. Usually the Quiet zone should be 10 times the module width or higher.
I	Use *initgraphics* command in PostScript. This may improve the positioning of the barcode if relative positioning is used in PostScript documents.
e	Move cursor to end of barcode in PCL.
W	Remove leading and trailing spaces from content.

Table 48: Overview Parameter Syntax of Version 1.x

Appendix E: TBarCode Daemon

The **TBarCode Daemon** is a background server process that performs the barcode generation. The **TBarCode Daemon** is an optional component. It is not available (and not required) for certain distributions of **TBarCode/X.** The daemon is usually located in

```
/usr/local/share/tbarcode9/tbarcoded
```

In general there is no need to manually start or stop the **TBarCode Daemon.** It is started automatically. It is only necessary to restart the daemon when the configuration files or license files have changed.

E.1 Usage

You need to have root privileges to run the **TBarCode Daemon.**

```
/usr/local/share/tbarcode9/tbarcoded options
```

Examples:

```
/usr/local/share/tbarcode9/tbarcoded
/usr/local/share/tbarcode9/tbarcoded --help
/usr/local/share/tbarcode9/tbarcoded --stop
```

E.2 Options

E.2.1 General Options

Short	Long	Description
	`--inifile=`*FILE*	Sets the path and name of the configuration file. (Default is `/usr/local/share/tbarcode9/tbarcoded.conf`.) Example: `--inifile=/home/userXYZ/myTbarcoded.conf`
	`--license=`*DIRECTORY*	Sets the path where the license file is located. (Default is `/usr/local/share/tbarcode9`.) Example: `--license=/etc` The name of the license file is always `license.ini`.

Table 49: TBarCode Daemon – General Options

E.2.2 Daemon and IPC Options

Short	Long	Description
`-r`	`--restart`	Restarts daemon.
`-s`	`--stop`	Stops daemon.
	`--kill`	Kills daemon.
	`--check`	Checks state of daemon.
	`--cleanup`	Cleans up resources.
	`--id=`*ID*	Sets identification number to ID.
	`--memory=`*SIZE*	Changes the size of the memory reserved for barcode creation. **TBarCode Daemon** uses a fixed memory block for the inter-process communication to exchange barcodes with the **TBarCode Command**. When creating only small barcodes (linear barcodes with little data), the memory consumption can be reduced by setting this value.

		The memory block needs to be big enough to hold a complete barcode (= the size of the resulting barcode file). The **TBarCode Command** and the **TBarCode Daemon** have to be called with the same memory settings. So it is best to set an equal memory size in the configuration files (`tbarcode.conf` and `tbarcoded.conf`). If unsure what to set, then do not edit this parameter manually. Example: `--memory=65000`

Table 50: TBarCode Daemon – Daemon and IPC Options

E.3 Error Message and Debug Options

Short	Long	Description
	`--errorfile=`*FILE*	Saves all messages in the given file. (This should only be used for debugging and not in the productive system.) Example: `--errorfile=/tmp/tbarcoded_errors.log`
	`--nosyslog`	Do not log messages using syslog.
	`--nostderr`	Do not log messages to stderr.
	`--trace=`*LEVEL*	Sets the trace level to a certain value. The trace level defines the amount of log messages that are written to an error file, syslog or stderr. Possible values (sorted from minimal to maximal information output): ▪ `error` (default) ▪ `warning` ▪ `info` ▪ `verbose` Example: `--trace=INFO`

Table 51: TBarCode Daemon – Error Message and Debug Options

E.3.1 Informative Output

Short	Long	Description
`-?`	`--help`	Shows a help text for general option.
	`--version`	Shows the version information.

Table 52: TBarCode Daemon – Informative Output

Appendix F: ASCII Table

This table helps you to enter the Print Controls in Hex-Format. For each character an equivalent hexadecimal code exists.

For example: “C“ = 43 hexadecimal or “2“ = 32 hexadecimal.

Hex Code	Symbol	Hex Code	Symbol	Hex Code	Symbol	Hex Code	Symbol
0	NUL	20	[space]	40	@	60	`
1	SOH	21	!	41	A	61	a
2	STX	22	"	42	B	62	b
3	ETX	23	#	43	C	63	c
4	EOT	24	$	44	D	64	d
5	ENQ	25	%	45	E	65	e
6	ACK	26	&	46	F	66	f
7	BEL	27	'	47	G	67	g
8	BS	28	(	48	H	68	h
9	HAT	29	)	49	I	69	i
A	LF	2A	*	4A	J	6A	j
B	VT	2B	+	4B	K	6B	k
C	FF	2C	,	4C	L	6C	l
D	CR	2D	-	4D	M	6D	m
E	SO	2E	.	4E	N	6E	n
F	ST	2F	/	4F	O	6F	o
10	SLE	30	0	50	P	70	p
11	CS1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	STB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	US	3F	?	5F	_	7F	□

Table 53: ASCII Table

Appendix G: Knowledge Base

G.1 Unix Printing (HP-UX and Solaris)

G.1.1 SVR4 Spooling System

Solaris and HP-UX uses the SVR4 print services. Under the SVR4 spooling system, the `lp` command accepts the data to be printed and makes a copy of it in the spool directory associated with the destination. The destination consists of a printer name and an optional specification of a class to which the printer belongs. When the specified printer is busy the job is sent to another printer in the same class. The spool directory is normally `/var/spool/lp/request/printer-name` and the print file is given a unique name to identify both the job and the user.

Access to the printer is controlled by `lpsched` daemon. It picks up the jobs from the spool directory and sends them to appropriate destination when it becomes available. `lpsched` also keeps a log, usually in `/usr/spool/lp/log`. The log file would indicate any error in processing the print jobs, as well as the user-name.

G.1.2 Interface Programs (BSD and SVR4)

Both BSD and SVR4 spooling systems support the concept of an interface program. The interface program, referred to as filters under the BSD system, is usually a shell script that translates the print file to a format suitable for the output device. The tasks performed by the interface program include: adding a banner and trailer pages, adding or removing a line feed character, generating accounting information and setting the correct modes on the output device. A standard interface program may be found in `/usr/lib/lpf` for the BSD systems and in `/usr/spool/lp/model` for the SVR4 system.

G.1.3 Printer Interface Scripts (HP-UX)

There are printer interface script "models" you can choose from that have been created for you in the `/usr/spool/lp/model` directory. Many of them have names that match the model numbers of Hewlett-Packard printers and plotters.

When you configure your printer into the lp spooler (e.g. with SAM), you must specify which printer model interface script you want to use. The model will be automatically copied from the `/usr/spool/lp/model` directory into the `/usr/spool/lp/interface` directory and given the name that you specified as printer name.

If you list the `/usr/spool/lp/model` directory you will find printer interface scripts like:

```
HPGL1, draftpro, hp2560, HPGL2, dumb, laserjet, PCL1, dumbplot,
laserjetIIIS, PCL2, fonts, hp2565a,hp33447a, paintjet, PCL3, hp2225a,
hp2566b, hp3630a, quietjet, PCL4, hp2225d, hp2567b, hp7440a, rmodel,
PRINT3K.model, hp2227a,hp2631g,hp7475a,rmttroff, bf_remote, hp2228a,
hp2684a, hp7550a, ruggedwriter, colorpro, thinkjet, deskjet - and many
others.
```

If you have an HP printer, you will probably find a model script that matches its model number or name. Those interface model scripts that match your printers typically do not need to be changed - except we want to include TEC-IT barcode software.

► In order to use TBarCode/X we need some shell programming to customize the printer interface model scripts to meet our printing needs.

If you do not have an HP printer, try using the `dumb` interface model. You might have to modify it to be able to use all of the features of your non-HP printer, but `dumb` should work for basic ASCII text printing. If the dumb printer interface model script does not work, contact your printer supplier for a UNIX line printer spooler interface script or try the script that most closely matches your non-HP printer type.

G.1.4 Links

- Printing under Unix (BSD, SVR4…)
  http://www.ussg.iu.edu/usail/peripherals/printers/

- Adding print queues (BSD, Solaris, IRIX, HP-UX…)
  http://www.fisica.uniud.it/~cabras/assistenza/print/tek/man/P740man/74086.htm

- AIX/HP-UX Printing Guide / Interoperability
  http://www.rz.uni-karlsruhe.de/Uni/RZ/Betriebssysteme/HP-UX/doc/interworks/Tech/aix_hpux_interop/chap08_print.html