TBarCode/X

Barcode Solution for Linux® and UNIX®

Version 7.0.1

User Documentation

20 November 2006

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

1.1 Table of Figures

1.2 List of Tables

2 Disclaimer

The actual version of this product (document) is available as is. TEC-IT declines all warranties which goes 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.

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 Haftungsausschluss

Dieses Produkt (bzw. Dokument) steht Ihnen in der aktuellen Version „WIE BESEHEN - ohne Gewährleistung“ zur Verfügung. TEC-IT weist alle Garantien, die über das anwendbare Recht hinausgehen, zurück. Risiken, die aus der Benutzung des Produkts und der Dokumentation entstehen, trägt der Lizenznehmer bzw. Benutzer. TEC-IT und seine Vertragspartner dürfen nicht für direkte oder indirekte Schäden oder Verluste belangt werden (dies beinhaltet, uneingeschränkt, Schäden durch den Verlust von Einkünften, Einschränkungen in der Geschäftsausübung, Verlust von Geschäftsinformationen sowie andere wirtschaftliche Verluste), die aus der Benutzung oder Unfähigkeit zur Benutzung des Produkts (der Dokumentation) entstanden sind, selbst wenn TEC-IT auf die Möglichkeit solcher Schäden hingewiesen hat.

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

4 About TBarCode/X

4.1 Features

4.1.1 TBarCode/X

- reduces the costs for barcode printing, because the licensing scheme does not depend on the number of printers.
- makes it possible to print barcodes on any *PCL*® or *PostScript*® compatible printer, without the need of installing a barcode extension cartridge or a special barcode font. Thus you can print barcodes in a complete device independent way.
- works in a completely transparent way. Barcodes are computed by a daemon process in the background.
- is available as precompiled barcode-printing engine for *Linux*®, *AIX*® and *HPUX*®.
- Other operating systems on request.

4.1.2 2D Symbologies

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

- PDF417
- DataMatrix
- MaxiCode

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).

4.1.3 Barcode Quality

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

- Specification of the module width in absolute units (device independent)
- PDF417 format and properties like error correction level
- Selection of the subsets of Code128 (subsets A, B and C – and automatic compression mode)
- The barcodes are created in vector graphic format (EPS and PCL), which uses the maximum of the available printing resolution.
- And many others…

4.2 Usage

There are two main uses of *TBarCode/X*:

- **Directly creating barcodes and saving them as vector or bitmap graphics files.**
  All necessary parameters are passed to a command line program.
- **Filtering 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 that is printed.

4.3 System Requirements

4.3.1 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 (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)
- Please visit our website http://www.tec-it.com to see which other platforms are currently supported. Binaries for special platforms are available on request.

4.3.2 Supported Output Devices

- PostScript® Level 2
- PCL® Level 5

5 Overview

The purpose of this section is to give 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.

5.1 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 interprocess communication, which is not available on all platforms.

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

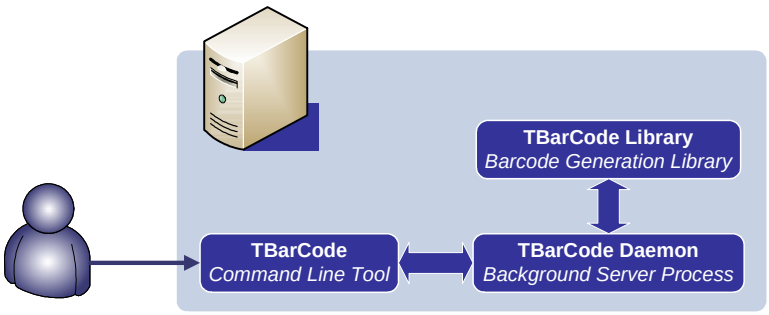


Figure 1: TBarCode/X *with Daemon as Background Server Process*

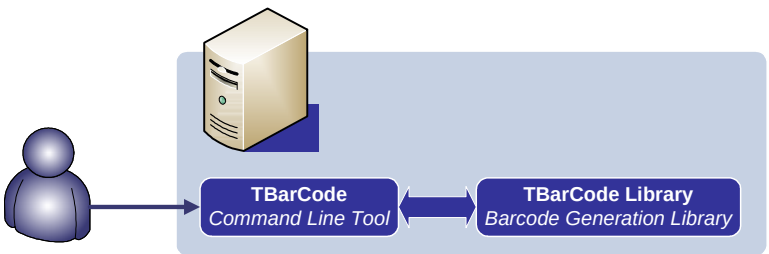


Figure 2: TBarCode/X *without Daemon*

5.1.1 TBarCode Command

TBarCode Command is a command line tool, which means it can be called from any console (shell) to create barcodes. It can also be used by shell scripts or applications.

5.1.1.1 Directly Creating Barcodes with TBarCode/X

TBarCode Command supports different output formats:

- Vector image formats such as PostScript® (PS, EPS) and PCL®
- Bitmap image formats such as BMP, GIF, JPG, PNG, TIF, etc.

Here is an example: The following 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`.

5.1.1.2 Using TBarCode/X to Process Data Streams

Another use of the *TBarCode Command* is to process data streams and automatically insert barcodes (“filter mode”). In filter mode the *TBarCode Command* reads data from standard input (`stdin`) and writes the results to standard output (`stdout`). 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.

5.1.2 TBarCode Library

TBarCode Library (also referred to as *LibTBarCode*) is available as static and dynamic library. It contains the functions for generating barcodes. The *TBarCode Command* uses the functions of the *TBarCode Library* to create the barcodes.

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

To develop your own applications with the *TBarCode 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.

5.1.3 TBarCode Daemon

The *TBarCode Daemon* is a background server process which performs the barcode calculations. If the *TBarCode Daemon* is installed then *TBarCode Command* is only a light-weight frontend for the *TBarCode 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/tbarcode7/tbarcoded
```

The daemon is started automatically as soon as *TBarCode Command* is invoked. In general there is no need to start the daemon manually.

5.2 The Following Sections

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

- The installation of *TBarCode/X* is described in Section 6, “Installation”.
- After installation some basic tests can be performed to see whether *TBarCode/X* is installed correctly. These tests are described in Section 7 “Testing TBarCode/X”.
- The command line usage of *TBarCode/X* is described in Section 8 “Using TBarCode/X”.
- Section 9 “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 11 “Licensing” explains how to install a valid license.

6 Installation

The *TBarCode/X* software comes as precompiled binaries. The installation package is available in two versions:

- as RPM package, which is the preferred format for Linux operating systems, or
- as TAR-GZ package with custom installation scripts, which is used on other Unix operating system.

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

6.1 Installing TBarCode/X from RPM Package

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

RPM packages are files with the extension `.rpm`. The *TBarCode/X* package usually has a name like `tbarcode-7.0.1-0D.i586.rpm`. The name of your package might be different. You will need to substitute the name `tbarcode-7.0.1-0D.i586.rpm` 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

```
rpm –i tbarcode-7.0.1-0D.i586.rpm
```

3. Register the *TBarCode/X* libraries (see Section 6.4 , “Registering TBarCode/X on the System”).
4. Check the execute permissions of *TBarCode/X* (see Section 6.5 “File Permissions”).
5. Installation is complete.

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.

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-7.0.1-0D.i586.rpm
```

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

6.1.1 Removing TBarCode/X

If you have installed *TBarCode/X* from 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.

6.2 Installing TBarCode/X from TAR-GZ Package

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

TAR-GZ packages are files with the extension `.tar.gz` or `.tgz`. The *TBarCode/X* package usually has a name such as `SetupTBarCode.tar.gz`. The name of your package might be different. You will need to substitute the name `SetupTBarCode.tar.gz` 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 commands

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

3. Check the execute permissions of *TBarCode/X* (see Section 6.5 “File Permissions”).
4. Installation is complete.

Steps 3 is actually optional, but this step is 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 #
```

6.2.1 Uninstalling TBarCode/X

If you have installed *TBarCode/X* from 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.tar.gz
cd SetupTBarCode
./uninstall.sh
```

3. Uninstallation is complete.

6.3 Installing TBarCode/X on SCO® Operating Systems

When you are using a SCO operating system, such as SCO OpenServer or SCO UnixWare, you will receive *TBarCode/X* as a native package image. The package usually has the extension `.ds` and file has a name like `tbarcode-7.0.1.ds`. (The name of your package might be different. You will need to substitute the name `tbarcode-7.0.1.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-7.0.1.ds tbarcode
```

3. Check the execute permissions of *TBarCode/X* (see Section 6.5 “File Permissions”).
4. Installation is complete.

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

```
pkginfo –l tbarcode
```

6.3.1 Removing 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.

6.4 Registering TBarCode/X on the System

Normally *TBarCode/X* should be ready for use after installing the packages as described above. But on some systems the following error message occurs when executing `tbarcode`:

```
error while loading shared libraries: libtbarcode7.so.0: cannot open shared object file:
No such file or directory
```

In this case the libraries of *TBarCode/X* are not properly registered by the system’s run-time linker. By default the libraries of *TBarCode/X* are installed in `/usr/local/lib`. This folder needs to be made public to the system’s runtime-linker.

In Linux this can be done by calling

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

on the command-line. (Alternatively, you can add the path `/usr/local/lib` to the file `/etc/ld.so.conf`.)

6.5 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/tbarcode7` using the command

```
ll /usr/local/share/tbarcode7
```

There are actually two versions of *TBarCode/X*: *TBarCode/X* where the barcode generator runs as daemon process and *TBarCode/X* where the barcode generator runs as normal program. The functionality of both versions is exactly the same.

6.5.1 TBarCode/X with Daemon

The output should look like this:

```
userxy@SuSE93:~> ll /usr/local/share/tbarcode7
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 as **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
```

6.5.2 TBarCode/X without Daemon

Note: Some *TBarCode/X* packages are shipped without `tbarcoded`. In this case you only need to check the file permissions of `tbarcode` as described in the section above.

7 Testing TBarCode/X

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

7.1 Running TBarCode/X from Command Line

7.1.1 Running 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
```

7.1.2 Running the TBarCode as Filter

Type the following command:

```
tbarcode --filter  </usr/local/share/tbarcode7/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.

7.2 License Bar

When testing *TBarCode/X*, you will probably see a black bar drawn horizontally across the whole barcode. Do not worry, 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 11 “Licensing” describes how you can acquire a valid license from TEC-IT.

7.3 TBarCode/X isn’t Working?

Please read through the previous sections. Make sure you have performed all required steps during installation. Consult the “Troubleshooting” section, if any problems remain.

8 Using TBarCode/X

8.1 Create a bar code

The samples below give you a quick start for generating bar codes. For more detailed instructions read ahead in chapter 8.3 .

8.1.1 Create a bar code in EPS (PostScript®) format

The command below creates a Data Matrix bar code 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 bar code to the output file “Barcode.ps” (specify full path if required)
-b71	Generate Barcode Type Data Matrix (71) – see 13.1 for more types
-m0.508	Set the module width (X Dimension) to 0.508 mm
-d”2D Code”	Encode the data “2D Code”

8.1.2 Create a bar code in PCL®-5 (HP-GL/2®) format

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

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

Parameter	Description
-fPCL	Use PCL® output format.
-oDataMatrix.pcl	Write bar code to the output file “DataMatrix.pcl” (specify full path if required)
-b13	Generate Barcode Type EAN-13 (71) – see 13.1 for 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)

8.1.3 Create a bar code in GIF format

The command below creates Code 39 bar code image (GIF) with the data content “DATA1234”

```
tbarcode –fIMAGE –iGIF -obarcode.gif -b8 -m0.254 -O -d"DATA1234"
```

Parameter	Description
-fIMAGE	Generate image – this feature needs ImageMagick libraries installed[1]
-iGIF	Selected image format = GIF (other formats may be JPG, PNG, TIF…)
-obarcode.gif	Write bar code to the output file “barcode.gif” (specify full path if required)
-b8	Generate Barcode Type Code-39 (8) – see 13.1 for more types
-m0.254	Set the module width (X Dimension) to 0.254 mm
-O	Optimize resolution (required for bitmap graphics)
-d”DATA1234”	Encode the data “DATA1234”

1 Please make sure that ImageMagick version 6.2.5 (or higher) is installed (previous version have an incompatible interface). If you can’t generate a bitmap format directly (due to incompatible Imagemagick versions) you can generate an EPS file and convert it to GIF (or TIF; JPG, PNG…) with the convert utility.

8.2 Filter a print job or document file

8.2.1 Insert a bar code into a PostScript® document

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

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

Bar code sequence parameters:

Parameter	Description
`$_tbcs`	Begin of barcode control sequence
`-fPS`	Format of output is PostScript® (default)
`-b71`	Generate Barcode Type Data Matrix (71) – see 13.1 for more types
`-m0.508`	Set the module width (X Dimension) to 0.508 mm
`-dMyBarcodeData`	-d marks the begin of bar code data (all characters following will be encoded)
`$_tbce`	End of bar code sequence

Then call `tbarcode` with the following parameters:

```
    $_tbcs –filter –stream=PS <infile.ps >outfile.ps
```

Now the `outfile.ps` will contain the original file plus the drawing commands for the bar code. For automatic bar code generation by your spool system see chapter 9

8.2.2 Insert a bar code into a PCL® document

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

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

Bar code sequence parameters:

Parameter	Description
`$_tbcs`	Begin of barcode control sequence
`-fPCL`	Format of output is PCL®
`-b71`	Generate Barcode Type Data Matrix (71) – see 13.1 for more types
`-m0.508`	Set the module width (X Dimension) to 0.508 mm
`-dMyBarcodeData`	-d marks the begin of bar code data (all characters following will be encoded)
`$_tbce`	End of bar code sequence

Then call `tbarcode` with the following parameters:

```
    $_tbcs –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 bar code. For automatic bar code generation by your spool system see chapter 9

8.3 TBarCode Command

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`. These paths need to be set in the environment variable `PATH`. If the path where `tbarcode` lies is not set, you will need to specify the full path to start it. For example:

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

8.3.1 Usage

```
tbarcode options barcodesettings
```

The *options* determine the way that *TBarCode Command* works, whereas the *barcode settings* describe a barcode. Parameters for *options* and *barcode settings* are described below.

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"
```

In the following we will only mention long style and short style, since these styles are more common. Not all parameters, only the most important ones, are available as short style.

8.4 Options

You can view the *TBarCode Command* options in a console with

```
tbarcode --help
```

8.4.1 General Options

Short	Long	Description

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

	`--inifile=`*FILE*	Optional: Sets the path and name of the configuration file. (Default is `/usr/local/share/tbarcode7/tbarcode.conf`.) Example: `--inifile=/home/userXYZ/myTbarcode.conf`
	`--license=`*DIRECTORY*	Optional: Sets the path where the license file is located. (Default is /usr/local/share/tbarcode7.) Example: `--license=/etc`

Short	Long	Description
		The name of the license file is always `license.ini`.
	`--globalxoffset=`*X*	Optional: 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*	Optional: 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*	Optional: Changes the size of the memory reserved for barcode creation. Only relevant when using the *TBarCode Daemon.* *TBarCode Daemon* uses a fixed memory block for the interprocess 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 1: General Options

8.4.2 Filter Options

Short	Long	Description
	`--filter`	Optional: Enables filter mode. In filter mode *TBarCode Command* 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.
	`--stream=`*TYPE*	Optional: Sets the type of the input stream. Possible values: `PS` … PostScript data stream, `PCL` … PCL data. If not set, *TBarCode Command* automatically detects the type of input stream. Example: `--stream=PS`
	`--escapebegin=`*STRING*	Optional: Sets a string that identifies the begin 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:

Short	Long	Description
		`--escapebegin=BARCODEBEGIN`
	`--escapeend=`*STRING*	Optional: 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*	Optional: Sets the insert mode for the barcode data.
		Possible values:
		`beforeline`
		`afterline`
		`beforestring` (default)
		`afterstring`
		Example:
		`--insert=afterline`

	`--linebyline`	Optional, 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 "`$_tbce`" by default) can be several lines after the start of the escape start sequence (marked with "`$_tbcs`" by default). 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`	Optional: Creates PCL reset commands at the beginning an the end of the PCL stream in filter mode.
-S	`--SAP`	Optional: This flag should be set when printing from an SAP environment. (When using Code 39 the characters `*` will trimmed from the begin and the end of the data. For example: `--data=*123*` will be interpreted as `--data=123`)
	`--easybar=`*STATE*	Optional: 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 8.5.4 “Filter Options”.

8.4.3 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.

But *TBarCode Command* 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`	Optional: Enables compatibility mode with *TBarCode/X* version 1.*x*.
		All barcode control sequences will be interpreted as with *TBarCode/X* 1.*x*.
		
		Hint: This parameter can be set in the `tbarcode.conf` configuration file. In this way the command `tbarcode` works exactly like the `tbarcodeclient` in *TBarCode/X* 1.*x*.
		Here is an example – a 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

8.4.4 Error Message and Debug Options

Short	Long	Description
	`--errorfile=`*FILE*	Optional: Saves all messages in the given file. (This should only be used for debugging and not in the productive system.) Example: `--errorfile=/tmp/tbarcode_errors.log`
	`--syslog`	Optional: Logs all messages using the syslog service.
	`--nostderr`	Optional: Prevents messages from being written to standard error channel (stderr).

	`--trace=`*LEVEL*	Optional: 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*	Optional: Defines the action that is done, when wrong barcode settings are applied. Possible values: `ignore` `message` (default) The default setting is `--onerror=message` where wrong barcode parameters are reported and *TBarCode Command* terminates with an exit value that indicates an error. With `--onerror=ignore` set *TBarCode Command* terminates normally without reporting an error.
	`--onnodata=`*ACTION*	Optional: Defines the action that is done, when the data parameter (`-d` or `--data`) is missing. Possible values: `ignore` `message` (default) The default setting is `--onnodata=message` where *TBarCode Command* terminates with an exit value that indicates an error. With `--onnodata=ignore` set *TBarCode Command* terminates normally without reporting an error.

Table 4: Error Message and Debug Options

8.4.5 Informative Output

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

8.5 Barcode Settings

You can view the available parameters for barcode settings with

```
tbarcode –-barcodesettings
```

or just

```
tbarcode –s
```

8.5.1 General Options

Short	Long	Description

`-d`	**`--data=`*`DATA`***	Sets the data of a barcode.
		Alternatively, you can specify a file the 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"""`

	`--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`

	`--bcfile=`*`FILE`*	Optional: Instead of specifying the barcode settings as command line parameters, you can specifies 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 8.6.2 "Syntax of a Configuration File".

	`--compress=`*`ALGORITHM`*	Optional: Compresses the data by using a compression algorithm.
		Possible algorithms:
		`NONE` (default)
		`DEFLATE`
		`GZIP`
		`ZLIB`

Short	Long	Description
		Compression is useful when a large amount of data has to be encoded as barcodes. Compression should only be used 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 decrompressed 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 UNIX server.
-f	**--format=*TYPE***	Optional: Defines the output format. Possible values: PS … PostScript PCL … PCL 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`

-b	**--barcode=*NUMBER***	Optional: Sets the type of barcode. The *NUMBER* of the barcode type can be looked up in Section 13.1 “Barcode Symbologies”. Default is `--barcode=20`, which is “Code 128”.
		Examples:
		`-b20`
		`--barcode=71`

-c	--checkdigit=*NUMBER*	Optional: Sets the type of the check-digit method. The *NUMBER* of the check-digit method can be looked up in Section 0 “Check Digit Methods”.
		Examples:
		`-c3`
		`--checkdigit=3`

	--autocorrect=*STATE*	Optional: Enables or disables auto-correction. Relevant for Code 2of5 Interleaved only. This feature adds a leading zero to the bar code data to produce an even number of digits (on demand). Possible values: on off Example: `--autocorrect=ON`
	--translation=*STATE*	Optional: Enables or disables the translation of escape sequences. Possible values: on off Example: `--translation=ON`
	--bearerwidth=*WIDTH*	Optional: Sets the width of a bearer. Unit of measurement: Millimeters. Example: `--bearerwidth=1.5`

Short	Long	Description
	`--bearertype=`*TYPE*	Optional: Sets the type of the bearer. Possible values: `none` … default `horizontal` … Horizontal bar above and below the barcode. `rectangle` … Rectangle around the barcode. Example: `--bearertype=HORIZONTAL`
	`--notchheight=`*HEIGHT*	Optional: Set the notch height. Unit of measurement: Millimeters. Example: `--notchheight=2.0`
-m	**`--modulewidth=`*WIDTH***	Optional: Sets the module width. Unit of measurement: Millimeters. Example: `--modulewidth=0.254`
	`--reduction=`*REDUCTION*	Optional: Reduces the width of the modules by a certain amount (given in %). Also known as “Pixel Shaving”. This option should be used for printers where the ink tends to bleed. Ink bleeding causes the barcode bars to be wider than they should be. Sometimes the barcode is no longer readable. In such cases the bar width can be reduced by using this parameter. Example: `--reduction=10`
	`--printratio=`*RATIO*	Optional: Sets the print ratio. Example: `--printratio=”1:2:1:3”`
	`--formatstring=`*FORMAT*	Optional: Sets the format string. The format string syntax can be looked up in Section 13.8 “Formatting Barcode Data”. Example: `--formatstring="A##B&"`
`-O`	`--optimalwidth` `--72dpiraster`	Optional: Optimizes barcode width. When this settings is turned on, the width of the modules are optimized. Each module width is exactly a multiple of a singe dot. 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.)
	`--quietzoneh=`*MODULES*	Optional. Sets the width of the horizontal quiet zone. A quiet zone is an empty space around a barcode. Unit of measurement: Modules. Example: `--quietzoneh=10`
	`--quietzonev=`*MODULES*	Optional: Sets the height of the vertical quiet zone. A quiet zone is an empty space around a barcode. Unit of measurement: Modules. Example: `--quietzoneh=10`
	`--nooverhead`	Optional: Suppresses the PCL or PostScript overhead.

Short	Long	Description
		PCL: Reset commands at the begin and end of the file are omitted. PostScript: The overhead for encapsulated PostScript (EPS) is omitted.
`-i`	`--imageformat=`*FORMAT*	Optional: 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, such as `BMP` (default), `GIF`, `JPG`, etc. Example: `--format=IMAGE –imageformat=GIF`
	`--dpi=`*DPI*	Optional: Sets the resolution of the image. Unit of measurement: Dots per inch (dpi).
	`--colormode=`*MODE*	Optional: 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*	Optional: The PCL output mode. By default *TBarCode* 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`
	`--defaultset=`*NUMBER*	Optional: Use a certain set of default values. `--defaultset=1` should be used when you are migrating from a hardware-based barcode printing solution to *TBarCode/X*.

Table 6: General Barcode Settings

8.5.2 Position and Size

Short	Long	Description

`-x`	`--xpos=`*POSITION*	Optional: 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*	Optional: 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`

`-w`	`--width=`*WIDTH*	Optional: Sets the width of the barcode. Unit of measurement: Millimeters Examples:

Short	Long	Description
		`-w25.4` `--width=55`

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

`-r`	`--rot=`*ROTATION*	Optional: 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`

	`--origin=`*ORIGIN*	Optional: 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`

	`--mustfit=`*STATE*	Optional: When activated TBarCode 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`

	`--decoder=`*TYPE*	Optional: Specifies the type of barcode decoder which will be used for scanning the barcode. 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 us: office@tec-it.com By setting the type of decoder, the TBarCode 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`
	`--sizemode=`*MODE*	Optional: Sets the mode that determines the barcode size. Possible values: `default`

Short	Long	Description
		`fit` ... The parameters `--width` and `--height` determine the size. `module` ... The parameter `--modulewidth` determines the size. `minimal` ... The parameters `--decoder` and `--dpi` determine the size. When `--sizemode=MINIMAL` then TBarCode 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`

Table 7: Barcode Position and Size

8.5.3 Text Options

Short	Long	Description
`-t`	`--text=`*POS*	Optional: 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.) `hide` (Hides the barcodes text. Draws only the bars.) Example: `--text=HIDE`

	`--align=ALIGNMENT`	Optional: Sets the horizontal text alignment.
		Possible values:
		`default`
		`left`
		`center`
		`right`
		Example:
		
		`--align=left`

	`--fontsize=`*SIZE*	Optional: Sets the size of the readable barcode text. Unit of measurement: Points
	`--font=`*NAME*	Optional: Sets the font that is used for drawing the readable barcode text.
	`--textdist=`*DISTANCE*	Optional: Sets the distance between the bars and the readable barcode text. Unit of measurement: Millimeters
	`--trimwhitespaces`	Optional: Removes all whitespaces (spaces, tabs, etc.) at the begin and the end of the barcode data.

Table 8: Barcode Text Options

8.5.4 Filter Options

To enable filtering *TBarCode* has to be called with the option `--filter`. See Section 8.4.2 “Filter Options” for a list of global filter options.

The following filter options can be set per barcode:

Short	Long	Description
	`--initgraphics`	Optional: Calls `initgraphics` in PostScript.
	`--movecursor`	Optional: Moves cursor in the PCL code to end of the barcode.
	`--remove`	Optional: Removes barcode control sequence from the data stream after filtering. (The default behaviour is to overwrite the barcode control sequences with

Short	Long	Description
		blanks.)
	`--embed=`*STATE*	Optional: 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*	Optional: Sets the positioning mode to relative or absolute coordinates. Possible values: `abs` (default for PostScript) `rel` (default for PCL)

Table 9: Filtering Options

8.5.5 PDF417 Options

Short	Long	Description

	`--PDFrows=`*ROWS*	Optional: Sets the number of rows.
		Possible values:
		`3 ... 90`
		Example:
		`--PDFrows=10`

	`--PDFcols=`*COLUMNS*	Optional: Sets the number of columns.
		Possible values:
		`1 ... 30`
		Example:
		`--PDFcols=9`

	`--PDFratio=`*RATIO*	Optional: Sets the row-columns-ratio. Example: `--PDFratio="3:1"`
	`--PDFauto`	Optional: Automatically chooses the row-column-ratio.
	`--PDFrowheight=`*HEIGHT*	Optional: Sets the height of a row. Units of measurement: Millimeters Examples: `--PDFrowheight=5.0`

	`--PDFecl=`*LEVEL*	Optional: Sets the error correction level.
		Possible values:
		`0 ... 8`
		Example:
		`--PDFecl=0`

Table 10: PDF417 Options

8.5.6 Macro PDF417 Options

Short	Long	Description
	`--PDFindex=`*INDEX*	Optional: Sets the segment index.
	`--PDFid=`*ID*	Optional: Sets the file ID.
	`--PDFlast`	Optional: Last segment

Short	Long	Description
	`--PDFfile=`*NAME*	Optional: Sets the file name.
	`--PDFcount=`*COUNT*	Optional: Sets the segment count.
	`--PDFtime=`*TIMESTAMP*	Optional: Sets timestamp.
	`--PDFsender=`*SENDER*	Optional: Sets the sender.
	`--PDFaddr=`*ADDRESSEE*	Optional: Sets the addressee.
	`--PDFsize=`*SIZE*	Optional: Sets the file size.
	`--PDFchecksum=`*SUM*	Optional: Sets the checksum.

Table 11: Macro PDF417 Options

8.5.7 DataMatrix Options

Short	Long	Description
	`--DMsize=`*SIZE*	Optional: Sets the DataMatrix size. The DataMatrix sizes can be looked up in Section 13.3.1 “Symbol Sizes”.
	`--DMformat=`*FORMAT*	Optional: Sets the DataMatrix format. The DataMatrix formats can be looked up in Section 13.3.2 “Format”.
	`--DMrect`	Optional: Draws DataMatrix as a rectangle. (Square is default.)
	`--DMsum=`*SUM*	Optional: Sets sum of structured append. Possible values: `2 … 16`
	`--DMindex=`*INDEX*	Optional: Sets index of structured append. Possible values: `1 … 16`
	`--DMfile=`*ID*	Optional: Sets the file id of structured append.

Table 12: Datamatrix Options

8.5.8 MaxiCode Options

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

Table 13: MaxiCode Options

8.5.9 QR-Code Options

Short	Long	Description
	`--QRversion=`*VERSION*	Optional: Sets the QR-Code version (symbol size). The possible values can be looked up in Section 13.5.1 “Version (Symbol Sizes)”.
	`--QRformat=`*FORMAT*	Optional: Sets the QR-Code format. The possible values can be looked up in Section 13.5.2 “Format”.
	`--QRind=`*INDICATOR*	Optional: Sets the format application indicator.

Short	Long	Description

	`--QRecl=`*LEVEL*	Optional: Sets the number of the error correction level. The error correction levels can be looked up in Section 13.5.3 “Error Correction Level”.
		Possible values:
		`0`
		`1` (default)
		`2`
		`3`

	`--QRmask=`*PATTERN*	Optional: Sets the mask pattern.
	`--QRsum=`*SUM*	Optional: Sets the sum of the structured append. Possible values: 2 … 16
	`--QRindex=`*INDEX*	Optional: Sets the index of the structured append. Possible values: 1 … 16
	`--QRparity=`*PARITY*	Optional: Sets the parity byte of the structured append.

Table 14: QR-Code Options

8.5.10 Codablock-F Options

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

Table 15: Codablock-F Options

8.5.11 RSS Expanded Stacked Options

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

Table 16: RSS Expanded Stacked Options

8.5.12 Composite Barcode Options

Short	Long	Description

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

Table 17: Composite Barcode Options

8.5.13 Multiple Barcodes

Creating „multiple barcodes“ is a new feature in *TBarCode/X 7.0*. This feature allows 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

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* automatically creates multiple barcodes as shown in the following figure.

Figure 3: Multiple Data Matrix Barcodes (1)

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

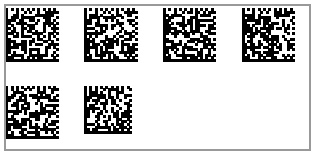


Figure 4: Multiple Data Matrix Barcodes (2)

If more data is added *TBarCode* 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[2].

Short	Long	Description

	`--multiple=`*STATE*	Enables or disable multiple barcodes.
		Possible values:
		`on`
		`off`

	`--rows=ROWS`	Optional: Sets the number barcode rows. For example, to draw 3 rows with barcodes: `--rows=3`

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

	`--columns=COLUMNS`	Optional: Sets the number of barcode columns. For example, to draw 4 columns with barcodes: `--columns=4`
	`--hdist=DISTANCE`	Optional: Sets a minimal horizontal distance between barcodes. Unit of measurement: Millimeters. Example: `--hdist=5`
	`--vdist=DISTANCE`	Optional: Sets a minimal vertical distance between barcodes. Unit of measurement: Millimeters: `--vdist=5`
	`--datalimit=LIMIT`	Optional: Sets the maximal amount of data (data bytes) that is encoded in a single barcode. Example: `--datalimit=1000`
	`--dynamicsize=MODE`	Optional: 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: `--dynamicsize=VERTICAL`
	`--structapp=MODE`	Optional: 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 18: Multiple Barcodes Options

8.5.14 Deprecated Options

The following options are deprecated. They have been replaced by a different parameter.

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

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

Table 19: Deprecated Options

8.6 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 20: TBarCode/X *Configuration Files*

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

`tbarcoded` is the *TBarCode 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.

8.6.1 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/tbarcode7`.

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 to all runs of *TBarCode/X*. The settings in the configuration files are the same settings as on the command line.

8.6.2 Syntax of a Configuration File

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

A line of configuration file contains either:

- an option, or
- an comment.

8.6.2.1 Options

Options have the syntax

```
option
```

or

```
option=value
```

8.6.2.2 Comments

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

```
# This would be a comment.
```

8.6.3 tbarcode.conf

The configuration file `tbarcode.conf` can contain *TBarCode Command* options and barcode settings as described in Section 8.4 "Options" and 8.5 "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
```

8.6.4 tbarcoded.conf

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

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

8.6.5 Priority of Options

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.

9 TBarCode/X as Spool Filter

TBarCode/X can be installed in the spool system to automatically filter print jobs. *TBarCode/X* can operates with all PostScript- or PCL-based printing queues. The application 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.

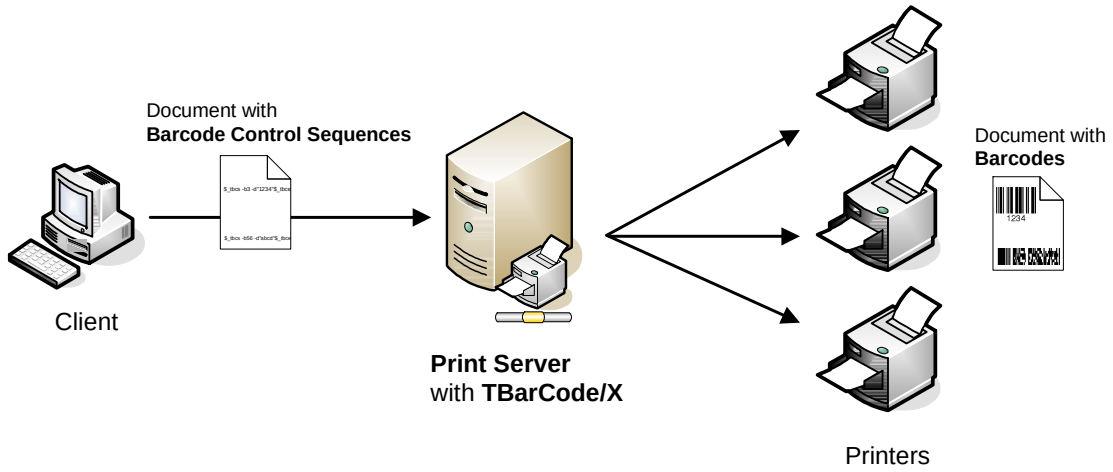


Figure 5: Printing with TBarCode/X

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

9.1 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 filterscripts 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. Search through the printcap file for the printer queue where 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/tbarcode7/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/tbarcode7/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/tbarcode7/filterlprng_fwd.sh:
```

6. Now open the script `/usr/local/share/tbarcode7/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
```

9.2 CUPS Printing System

One important note about CUPS: When you want your print jobs to be filtered by *TBarCode/X*, then do not use “raw queues”. Raw queues ignore all filters. You have to use queues with “local filtering” using a printer driver (`*.ppd`).

9.2.1 Setting up TBarCode/X for PostScript

The following changes have to be made to the MIME type conversion file (`/etc/cups/mime.convs`) of CUPS.

1. Open `/etc/cups/mime.convs`
2. Search for the following line

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

   Replace `pstops` with `/usr/local/share/tbarcode7/filtercups_ps.sh`. The line should look like this:

```
application/postscript application/vnd.cups-postscript 66                    ↵
    /usr/local/share/tbarcode7/filtercups_ps.sh
```

3. Restart the printing service:

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

9.2.2 Setting up *TBarCode/X* for PCL

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`
2. Add the new MIME type `application/pcl` to the list of "Application-generated files…"

```
…
application/postscript   ai eps ps string(0,%!) string(0,<04>%!) \
                         contains(0,128,<1B>%-12345X) + \
                         (contains(0,1024,"LANGUAGE=POSTSCRIPT") \
                          contains(0,1024,"LANGUAGE = Postscript") \
                          contains(0,1024,"LANGUAGE = PostScript") \
                          contains(0,1024,"LANGUAGE = POSTSCRIPT"))
application/vnd.hp-HPGL  hpgl 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**.)

3. Open the MIME type conversion file /etc/cups/mime.convs
4. Add the conversion rule for `application/pcl`

```
…
application/pdf        application/postscript          33   pdftops
application/postscript application/vnd.cups-postscript 66                    ↵
    /usr/local/share/tbarcode7/filtercups_ps.sh
application/pcl        application/vnd.cups-raw        66                    ↵
    /usr/local/share/tbarcode7/filtercups_pcl.sh
…
```

9.3 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/tbarcode7/tbarcode --filter Parameters`". For example:

```
f1=/usr/local/share/tbarcode7/tbarcode --filter
```

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

```
qprt –PPrinterName –f1 /usr/local/share/tbarcode7/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*.

9.4 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 17.1 "Unix Printing (HP-UX and Solaris)", in the appendix for background information.

First you should perform a basic test to see if the filter works on your system. These tests are describe in Section 7.1.2, "Running the TBarCode as Filter".

9.4.1 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.

9.4.2 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".

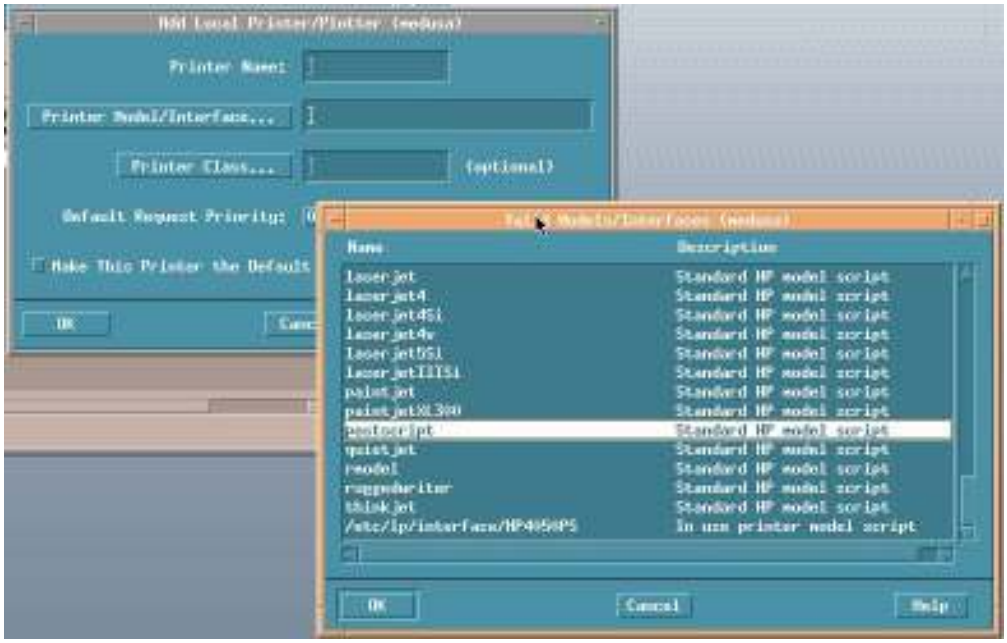


Figure 6: HP-UX Printer Models/Interfaces

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.

9.4.3 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.

9.4.4 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/tbarcode7/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

9.4.5 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/tbarcode7/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
```

Make a test call with:

```
lp -d Printer /usr/local/share/tbarcode7/samples/testfile.ps
```

On a PostScript printer the printout should contain several barcodes.

9.5 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`.

9.6 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/tbarcode7/samples/testfile.ps
```

This command should print a simple testfile. Check the printout – it should contain several barcodes.

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

10 Generating Raster Images

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

- Directly creating bitmap output with *TBarCode/X*

  TBarCode/X supports bitmap output for barcodes, but with certain limitations. To create bitmap output the ImageMagick® libraries (http://www.imagemagick.org/) must be installed on the system. *TBarCode/X* dynamically loads the ImageMagick libraries when needed. *TBarCode/X* currently does not support text output. No human readable text will be drawn, only bars. Certain barcode symbologies, such as the MAXICODE, are not support for direct bitmap output.

- Creating PostScript output and manually converting PostScript 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.

10.1 Generating Raster Images with TBarCode/X

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

```
tbarcode --output=barcode.bmp --format=IMAGE --imageformat=BMP --barcode=20 --data="1234"
```

With the parameters `--width` and `--height` one can customize the size of the barcode. The units of measurement are millimeters (mm).

With the option `--optimalwidth` you can enforce that a single bar is exactly an integer number of pixels (no fractional part). This parameter optimizes the readability of the barcode.

The size of single pixel depends on the image resolution. The default resolution is 72 dpi. A custom resolution can be set with the parameter `--dpi`.

It is possible to directly set the size of a single module[3]. This can be done with the parameter `--modulewidth`. Here is an example: The following line creates a Datamatrix barcode where one element is exactly on pixel in size.

```
tbarcode -obarcode.bmp -fIMAGE --imageformat=BMP –b71 --modulewidth=0.35278 -d"1234"
```

The module width depends on the resolution of the bitmap (default: 72 dpi). The module width can be calculated with the following formula:

$$Modulewidth = \frac{25.4}{R} \cdot p$$

where,

- R is the bitmap resolution in dpi, and
- p is the desired pixel size of a single module.

[3] A module is the smallest bar in a barcode. All bars are an integer multiple of a single module.

10.2 Converting PostScript Images to Raster Images

The following steps demonstrate how to create barcode bitmap images with *TBarCode/X*.

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
```

10.3 Dynamic 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.

10.3.1 Display a Barcode in a Browser

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

10.3.1.1 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();
$r = shell_execute ("tbarcode … barcode parameters… $tmp_bc_file.eps");
$r = shell_execute("ghostscript … conversion parameters… ./imgpath/$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
```

10.3.1.2 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());
$r = shell_execute ("tbarcode … $data… $unique_filename.eps");
$r = shell_execute("ghostscript … conversion parameters… . $unique_filename.jpg");
// read the whole file and send it to the browser
$fp=fopen($unique_filename."jpg","rb");
fpassthru($fp); // pass through as binary data stream (JPG image format)
flush();
unlink ($unique_filename.".eps"); // delete both files
unlink ($unique_filename.".jpg");
// 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()`.

10.3.2 Hints for using `shell_execute()`[4]

- 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");`
- 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).

[4] taken from php.net

- 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");
```

11 Licensing

11.1 License Key and License Types

TBarCode/X can be used immediately after installation. There is no license key required to evaluate *TBarCode/X*. But as long as you have not licensed *TBarCode/X,* an additional horizontal bar will be printed across the barcode. Usually this horizontal bar does not affect the readability of the code for evaluation purposes.

The purchase of a license (and applying the license key) removes this restriction. There are three possible license modes to choose from:

Kind of License	Scope
Single / Workstation	For the use of the Software on exactly one computer, limited to one concurrent user for printing on local printers. No usage over a network (LAN/WAN) for several users and no http-based applications. No use on a server or for network printers. We need your hostname for this license / refer to the section 12 Appendix – Troubleshooting/FAQ
Site	For the use of the Software on several computers within one Site, however limited to a maximum of 250 concurrent installations or 250 concurrent users.
Enterprise / World	For the use of the Software on several (or all) computers within several (or all) Sites of the Licensee's business without any restrictions regarding the number of installations or the number of users.
Developer	For the redistribution of *TBarCode Library* as part of the Licensee's software application(s). The number of redistributions per license is limited to 10000 installations or to 10000 users. Client applications only, no server use. Each developer building applications with the Software needs one Developer License.
Web	For the use of the Software on one server for internet-, intranet and other http-based applications. The number of users per license is limited to 10000. Each web-server instance needs one Web-License.
Server	For the use of the Software on one application server (on demand bundled with test and development server). Multi-Server licenses are available.

If you have questions about the license best suited for your requirements, contact sales@tec-it.com and also check out the license terms on www.tec-it.com

Note: Redistributing *TBarCode/X* is generally NOT allowed (except with a Developer license).

11.1.1 Integration into your software

If you are developing software as service and need our product to be integrated into your software we offer you a Developer license, which allows distribution and operation of *TBarCode Library* on clients (Workstations). If *TBarCode/X* is used on a server you need to license the product for each customer (either per site or per server). If this won't be applicable for you, please contact us to find a suitable license model (sales@tec-it.com).

11.2 License File

After you have sent us your order, you will receive a special file with your registration data. This 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/tbarcode7
```

If you are using a Site or a World License, you have to copy this file to each system (client) where you want to use the product. Overwriting the original (demo) `license.ini` file that was installed during setup.

On systems where *TBarCode 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/tbarcode7/tbarcoded --restart
```

You need root privileges to do this. (If you cannot find `tbarcoded`, then your are probably using a *TBarCode/X* version without *TBarCode 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/.

12 Appendix – Troubleshooting/FAQ

12.1 General Questions

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

Yes, that is possible. You can 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.

12.1.2 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.

12.1.3 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 *TBarCode Command* with the parameter `--datafile=`*File*. For example:

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

12.1.4 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/tbarcode7`. See Section 11 "Licensing" for more information.

12.1.5 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
```

12.1.6 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: libtbarcode7.so.0: cannot open shared object file:
No such file or directory
```

Solution:

- Make sure that `libtbarcode7.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
```

12.1.7 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`.

12.1.8 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 11 “Licensing” or contact us for a valid license file.

12.2 Questions about Filtering/Printing

12.2.1 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.

More information about printing problems --> www.linuxprinting.org

12.2.2 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, which converts ASCII to PCL on demand is Magicfilter. This program is standard for several Linux distributions and often installed in the spool system already.

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

12.2.3 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 9 "TBarCode/X as Spool Filter").
- Sometimes lp uses "raw" mode (no filtering) use lpr instead.

12.2.4 How can I exchange my printer specific (e.g. BarSIMM®) control sequences with TBarCode control sequences?

It is possible, but requires some work. All BarSIMM specific control sequences need to be replaced with *TBarCode* control sequences. Here is an example for the symbology "2of5 Interleaved":

	PCL BarSIMM:	*TBarCode/X* Control Sequence:
Prefix	`<Esc>(s1p37v24640T`	`$_tbcs -fPCL -b3 –m0.254 -h13 -tHIDE –-origin=BOTTOM -d`
Suffix	`<Esc>(0N_(s3T`	`$_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 in most barcode applications. In most cases the result will be a valid (and well decidable) barcode. You can specify this parameter in the *TBarCode/X* configuration file `tbarcode.conf`.

If you don't know, which barcode parameters are defined by your printer specific (e.g. BarSIMM®) Escape Sequences, please send them to support@tec-it.com. We will try to find the corresponding *TBarCode/X* control sequences for you.

12.2.5 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 options removes the Easybar control sequence from the filter stream, which is the default behavior of most Easybar devices.

12.2.6 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.

12.2.7 How can I send a file directly 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, 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 *TBarCode Command* and then send it directly to a printer.

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

12.2.8 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
```

12.3 Where I can get more help?

- Your question is not listed here? Please, contact us (see Section □, “Please note: For all Composite Symbologies the vertical bar “|” character is used to separate the data of the linear symbol and the 2D composite component.
- Example: 1234567890123|TEC-IT

29	**RSS-14 Composite Symbology** Valid characters RSS-14: “0”..”9”, 13 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied	 (01)12345678901231
	Encoded data: 1234567890123|TEC-IT	
	Notes: RSS-14 barcode with an attached 2D component (CC-A or CC-B). The leading Application Identifier “01” (GTIN) is prefixed automatically by the software and must not occur in the input data. The 2D component can encode additional information like lot number, quantity, expiration date …	
78	**RSS-14 Truncated Composite Symbology** Valid characters RSS-14: “0”..”9”, 13 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied	 (01)12345678901231

	Encoded data:	1234567890123|TEC-IT
	Notes:	RSS-14 Truncated barcode with an attached 2D component (CC-A or CC-B).
79	**RSS-14 Stacked Composite Symbology** Valid characters RSS-14: Valid characters CC-A/B: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits:	 "0"..."9", 13 digits + 1 check digit ISO 646 character set, up to 338 characters 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 EAN 14 (eCDEAN14) User supplied
	Encoded data:	1234567890123|TEC-IT
	Notes:	RSS-14 Stacked barcode with an attached 2D component (CC-A or CC-B).
80	**RSS-14 Stacked Omni directional Composite Symbology** Valid characters RSS-14: Valid characters CC-A/B: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits:	 "0"..."9", 13 digits + 1 check digit ISO 646 character set, up to 338 characters 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 EAN 14 (eCDEAN14) User supplied
	Encoded data:	1234567890123|TEC-IT
	Notes:	RSS-14 Stacked Omni directional barcode with an attached 2D component (CC-A or CC-B).
31	**RSS Expanded Composite Symbology** Valid characters RSS Exp.: Valid characters CC-A/B: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits:	 ASCII characters between 0..127 ISO 646 character set, up to 338 characters 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 None (eCDNone). Modulo 10 (eCDMod10). EAN-14 (eCDEAN14)
	Encoded data:	1234567890123|TEC-IT
	Notes:	RSS Expanded barcode with an attached 2D component (CC-A or CC-B).
81	**RSS Expanded Stacked Composite Symbology** Valid characters RSS ES: Valid characters CC-A/B: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits:	 ASCII characters between 0..127 ISO 646 character set, up to 338 characters 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 None (eCDNone). Modulo 10 (eCDMod10). EAN-14 (eCDEAN14)
	Encoded data:	ABCabc123+|TEC-IT
	Notes:	RSS Expanded Stacked barcode with an attached 2D component (CC-A or CC-B).
30	**RSS Limited Composite Symbology** Valid characters RSS Lim.: Valid characters CC-A/B: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits:	 "0"..."9", 13 digits + 1 check digit ISO 646 character set, up to 338 characters 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 EAN 14 (eCDEAN14) User supplied
	Encoded data:	1234567890123|TEC-IT
	Notes:	RSS Limited barcode with an attached 2D component (CC-A or CC-B).

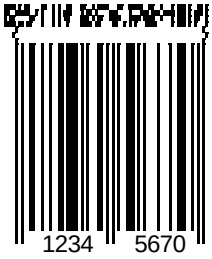
16	**UCC/EAN128 Composite Symbology** Valid characters EAN 128: ASCII-characters between 0..127 Valid characters CC-A/B/C: ISO 646 character set, up to 2361 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B1S:2S:3S:4S Default check-digit: None (eCDNone). Possible check-digits: Modulo 10 (eCDMod10). EAN-14 (eCDEAN14)	
	Encoded data: 1234567890|TEC-IT	
	Notes: EAN128 barcode with an attached 2D component (CC-A, CC-B or CC-C).	
10	**EAN-8 Composite Symbology** Valid characters EAN 8: "0".."9", 7 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-8 (eCDEAN8) Possible check-digits: User supplied	
	Encoded data: 1234567|TEC-IT	
	Notes: EAN-8 barcode with an attached 2D component (CC-A or CC-B).	
13	**EAN-13 Composite Symbology** Valid characters EAN 13: "0".."9", 12 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-13 (eCDEAN13) Possible check-digits: User supplied	
	Encoded data: 123456789012|TEC-IT	
	Notes: EAN-13 barcode with an attached 2D component (CC-A or CC-B).	
34	**UPC-A Composite Symbology** Valid characters UPC-A: "0".."9", 11 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: UPC-A (eCDUPCA) Possible check-digits: User supplied	
	Encoded data: 12345678901|TEC-IT	
	Notes: UPC-A barcode with an attached 2D component (CC-A or CC-B).	
37	**UPC-E Composite Symbology** Valid characters UPC-A: "0".."9", 7 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: UPC-E (eCDUPCE) Possible check-digits: User supplied	
	Encoded data: 1234567|TEC-IT	
	Notes: UPC-E barcode with an attached 2D component (CC-A or CC-B).	

Table 44: Supported Composite Symbologies and Enumerators

Contact and Support Information"). We do our best to support our customers.

13 Appendix – Barcode Parameters

13.1 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 e`BC_Code11`, are only relevant for programmers who want to use the *TBarCode Library* directly. They are irrelevant for users of the *TBarCode Command*.

ID	Internal barcode ID. If not supported in the current version marked with *
Enumeration	Enumeration of the barcode type. The numbering of the barcode type corresponds to the value defined by the enumerator.
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	Enumeration	Bar Code Name	Print Ratio	Ratio Format String (Ratio Hint)	Check Digit
0	eBC_None	Not a valid type	-------	--------	
1	eBC_Code11	Code 11	1:2.24:3.48:1:2.24	1B:2B:3B:1S:2S	eCDNone
2	eBC_2OF5	Code 2 of 5 (Standard)	1:3:4.5:1:3	1B:2B:3B:1S:2S	eCDNone
3	eBC_2OF5IL	Interleaved 2 of 5 Standard	1:3:1:3	1B:2B:1S:2S	eCDNone
4	eBC_2OF5IATA	Code 2 of 5 IATA	1:3:1	1B:2B:1S	eCDNone
5	eBC_2OF5M	Code 2 of 5 Matrix	1:3:4.5:1:3	1B:2B:3B:1S:2S	eCDNone
6	eBC_2OF5DL	Code 2 of 5 Data Logic	1:3:1:3	1B:2B:1S:2S	eCDNone
7	eBC_2OF5IND	Code 2 of 5 Industrial	1:3:1	1B:2B:1S	eCDNone
8	eBC_3OF9	Code 3 of 9 (Code 39)	1:3:1:3	1B:2B:1S:2S	eCDNone
9	eBC_3OF9A	Code 3 of 9 (Code 39) ASCII	1:3:1:3	1B:2B:1S:2S	eCDNone
10	eBC_EAN8	EAN8	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN8
11	eBC_EAN8P2	EAN8 - 2 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN8
12	eBC_EAN8P5	EAN8 - 5 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN8
13	eBC_EAN13	EAN13	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
14	eBC_EAN13P2	EAN13 - 2 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
15	eBC_EAN13P5	EAN13 - 5 digits add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
16	eBC_EAN128	EAN128	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128

ID	Enumeration	Bar Code Name	Print Ratio	Ratio Format String (Ratio Hint)	Check Digit
17	eBC_UPC12	UPC 12 Digits	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA
18	eBC_CodaBar2	CodaBar (2 widths)	1:3:1:3	1B:2B:1S:2S	eCDNone
19*	eBC_CodaBar18	CodaBar (18 widths)			----
20	eBC_Code128	Code128 automatic subset switching / autocompress (Code128 A, B, C see below!)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
21	eBC_DPLeit	Deutsche Post Leitcode	1:3:1:3	1B:2B:1S:2S	eCDDPLeit
22	eBC_DPIdent	Deutsche Post Identcode	1:3:1:3	1B:2B:1S:2S	eCDDPIdent
23*	eBC_Code16K	Code 16K			----
24*	eBC_49	Code 49			----
25	eBC_9OF3	Code 93	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCD2Mod47
26	eBC_UPC25	Identical to eBC_UPCA	-------		----
27*	eBC_UPCD1	UPCD1			----
28	eBC_Flattermarken	Flattermarken	1:1	1B:1S	eCDNone
29	eBC_RSS14	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	eBC_RSSLtd	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	eBC_RSSExp	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	eBC_TelepenAlpha	Telepen Alpha	1:3:1:3	1B:2B:1S:2S	eCDNone
33	eBC_UCC128	UCC128 (= EAN128)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
34	eBC_UPCA	UPC A	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA
35	eBC_UPCAP2	UPC A – 2 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA
36	eBC_UPCAP5	UPC A – 5 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCA
37	eBC_UPCE	UPC E	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCE
38	eBC_UPCEP2	UPC E – 2 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCE
39	eBC_UPCEP5	UPC E – 5 digit add on	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDUPCE
40	eBC_PostNet5	PostNet-5 (ZIP 5 digits)	1:1	1B:1S	eCDPostNet
41	eBC_PostNet6	PostNet-6 (ZIP 5 digits + check digit)	1:1	1B:1S	eCDPostNet
42	eBC_PostNet9	PostNet -9 (ZIP + 4)	1:1	1B:1S	eCDNone
43	eBC_PostNet10	PostNet-10 (ZIP + 4 + check digit)	1:1	1B:1S	eCDPostNet
44	eBC_PostNet11	PostNet-11 (ZIP + 4 + 2)	1:1	1B:1S	eCDPostNet
45	eBC_PostNet12	PostNet -12 (ZIP + 4 + 2+ check digit)	1:1	1B:1S	eCDPostNet

ID	Enumeration	Bar Code Name	Print Ratio	Ratio Format String (Ratio Hint)	Check Digit
46	eBC_Plessey	Plessey Code	1:2:1:2	1B:2B:1S:2S	eCDPlessey
47	eBC_MSI	MSI Code	1:2:1:2	1B:2B:1S:2S	eCDMSI1
48	eBC_SSCC18	SSCC18	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod10
49*	eBC_FIM	FIM			
50	eBC_LOGMARS	LOGMARS	1:3:1:3	1B:2B:1S:2S	eCDNone
51	eBC_Pharma1	Pharmacode One-Track	1:3:2:4:2:3	1B:2B:1C:2C:1S:2S	eCDNone
52	eBC_PZN	Pharmazentralnumber	1:2.5:1:2.5	1B:2B:1S:2S	eCDPZN
53	eBC_Pharma2	Pharmacode Two-Track	1:1	1B:1S	eCDNone
54*	eBC_GP	General Parcel			
55	eBC_PDF417	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	eBC_PDF417Trunc	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	eBC_MAXICODE	MaxiCode	1:1		----
58	eBC_QRCode	QR-Code	1:1	(1B:1S)	----
59	eBC_Code128A	Code128 (Subset A)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
60	eBC_Code128B	Code128 (Subset B)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
61	eBC_Code128C	Code128 (Subset C)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDCode128
62	eBC_9OF3A	Code 93 Ascii	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCD2Mod47
63	eBC_AusPostCustom	Australian Post Standard Customer	1:1	1B:1S	eCDNone
64	eBC_AusPostCustom2	Australian Post Customer 2	1:1	1B:1S	eCDNone
65	eBC_AusPostCustom3	Australian Post Customer 3	1:1	1B:1S	eCDNone
66	eBC_AusPostReplyPaid	Australian Post Reply Paid	1:1	1B:1S	eCDNone
67	eBC_AusPostRouting	Australian Post Routing	1:1	1B:1S	eCDNone
68	eBC_AusPostRedirect	Australian Post Redirection	1:1	1B:1S	eCDNone
69	eBC_ISBN	ISBN Code (=EAN13P5)	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN13
70	eBC_RM4SCC	Royal Mail 4 State (RM4SCC)	1:1	1B:1S	eCDNone
71	eBC_Data Matrix	Data Matrix	1:1		----
72	eBC_EAN14	EAN-14	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDEAN14
73*	eBC_CODABLOCK_E	Codablock-E			eCDCodablockE
74	eBC_CODABLOCK_F	Codablock-F	1:2:3:4:1:2:3:4	1B:2B:1S:2S	eCDCodablockF
75	eBC_NVE18	NVE-18	1:2:3:4:1:2:3:4	1B:2B:3B:4B:1S:2S:3S:4S	eCDMod10
76	eBC_JapanesePostal	Japanese Postal Code	1:1	1B:1S	eCDNone
77	eBC_KoreanPostalAuth	Korean Postal Authority Code	1:3:4	1B:1S:2S	eCDMod10Kor
78	eBC_RSS14Trunc	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

ID	Enumeration	Bar Code Name	Print Ratio	Ratio Format String (Ratio Hint)	Check Digit
79	eBC_RSS14Stacked	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	eBC_RSS14StackedOmni	RSS-14 Stacked Omnidirektional	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	eBC_RSSExpStacked	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	eBC_Planet12	Planet 12 digits	1:1	1B:1S	eCDMod10Pla
83	eBC_Planet14	Planet 14 digits	1:1	1B:1S	eCDMod10Pla
84	eBC_MicroPDF417	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	eBC_USPSOneCode4CB	USPS OneCode 4-State Customer Barcode	1:1	1B:1S	eCDNone
86	*eBC*_PlesseyBidir	Plessey Code with bidirectional reading support	1:2:3:1:2	1B:2B:3T:1S:2S	eCDPlessey

Table 21: Barcode Symbologies and Enumerators

13.2 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.

Check digit enumeration	Enumeration value	Check digit calculation methods
eCDNone	0	No check digit will be computed
eCDStandard	1	Standard check digit of barcode type will be used
eCDMod10	2	Modulo 10 (usually used with Interleaved 2of5)
eCDMod43	3	Modulo 43 (suggested for Code39 and Logmars, consist of 1 digit)
eCD2Mod47	4	Modulo 47 (2 digits)
eCDDPLeit	5	Method for DP Leitcode
eCDDPIdent	6	Method for DP Identcode
eCD1Code11	7	Method for Code11 (1 digit)
eCD2Code11	8	Method for Code11 (2 digits)
eCDPostnet	9	Method for USPS Postnet
eCDMSI1	10	Method for MSI (1 digit)
eCDMSI2	11	Method for MSI (2 digits)
eCDPlessey	12	Method for Plessey
eCDEAN8	13	Method for EAN 8
eCDEAN13	14	Method for EAN 13
eCDUPCA	15	Method for UPC A
eCDUPCE	16	Method for UPC E
eCDEAN128	17	EAN 128 internal method (Modulo 103)
eCDCode128	18	Code 128 internal method (Modulo 103)
eCDRM4SCC	19	Method for Royal Mail 4 State
eCDPZN	20	Mod-11 Method for PZN

eCDMod11W7	21	Mod-11 (Weighting = 7)
eCDEAN14	22	Method for EAN 14
eCDMod10Kor	23	Method for Korean Postal Authority - Modulo 10
eCDMod10Pla	24	Method for Planet - Modulo 10
eCDMod10ItlPst25	25	Method for Italian Postal 2/5 (Modulo 10 based)
eCDMod36	26	Modulo 36 (ISO/IES 7064) for DPD Barcode

Table 22: Check Digit Methods and Enumerators

13.3 DataMatrix Parameters

13.3.1 Symbol Sizes

The user-defined symbol sizes for DataMatrix 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 23: DataMatrix Symbol Sizes

13.3.2 Format

The DataMatrix 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 24: DataMatrix Formats

13.4 MaxiCode Parameters

13.4.1 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 25: MaxiCode Modes

13.5 QR-Code Parameters

13.5.1 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 26: QR-Code Symbol Sizes

13.5.2 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 27: QR-Code Format Options

13.5.3 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	Quartil (Default)
3	High

Table 28: QR-Code Error Correction Levels

13.6 Codablock-F Parameters

13.6.1 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 29: Codablock-F Parameters

13.7 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 DataMatrix).

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`.

13.7.1 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 (**E**xtended **C**haracter **I**nterpretation), used in 2D codes like MaxiCode, DataMatrix 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 (**E**CI **B**egin) opens a nesting level, \EE (**E**CI **E**nd) closes it. Used in QR-Code
\G	GLI (**G**lobal **L**anguage **I**dentifier), similar to ECI, but only used in PDF417.

Table 30: 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 also to the document "Barcode Reference" (available from http://www.tec-it.com).

13.7.2 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.

13.7.3 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 31: Extended Escape Sequences

13.8 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, 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 7 (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 32: 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 7 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 33: 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

13.9 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 34: PCL Font Numbers

14 Appendix – Using Version 1.x Format

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

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/X 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 V1 to the actual version and 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 V2 parameters, because they are more flexible and more intuitive.

14.1 Overview V1 Format

Below 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*

Parameter V1	Description
`$_tbcs`	Marks the beginning of the sequence (used with filter)
`$_tbce`	Marks the end of the sequence (used with filter)
d*Content*	Content = data of bar code; **must be the last parameter** before $_tbce
x*Position*	Absolute x position in mm (* see above)
y*Position*	Absolute y position in mm (* see above)
w*Width*	Width of bar code in mm (e.g. w50 or w53.12)
h*Height*	Height of bar code 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.)
b*BarcodeNo*	Number of bar code (see Barcode Types in the Appendix)
c*MethodNo*	Number of check digit calculation method
r*Rotation*	Rotation in degrees (0, 90, 180 or 270)
`T(on|off)`	Show plain text.
`a`	Print plain text above bar code (default is below)
`s(on|off)`	Translate escape sequences in input data
`A(on|off)`	Turn auto correct on or off

Parameter V1	Description
g*GuardWidth*	Width of guarding line in mm
f*Fontname*	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.
f*Fontsize*	Size of font in points.
t*Format*	Output format: PS (=PostScript, default) or PCL
i*Distance*	Text distance in mm
N*Height*	Notch height in mm
m*ModWidth*	Module width (narrow bar width) in µm (= 1/1000 mm), if used the W parameter for the symbol width is irrelevant.
R*Ratio*	Print ratio
F*Format*	Format string used for formatting bar code data prior to printing it
O	Calculate optimal width of bar code
Qh*HorzQZ*	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.
Qv*VertQZ*	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 bar code 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 35: Overview Parameter Syntax of Version 1

15 Appendix – 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/tbarcode7/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.

15.1 Usage

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

```
/usr/local/share/tbarcode7/tbarcoded options
```

Examples:

```
/usr/local/share/tbarcode7/tbarcoded
/usr/local/share/tbarcode7/tbarcoded --help
/usr/local/share/tbarcode7/tbarcoded --stop
```

15.2 Options

15.2.1 General Options

Short	Long	Description
	`--inifile=`*FILE*	Optional: Sets the path and name of the configuration file. (Default is `/usr/local/share/tbarcode7/tbarcoded.conf`.) Example: `--inifile=/home/userXYZ/myTbarcoded.conf`
	`--license=`*DIRECTORY*	Optional: Sets the path where the license file is located. (Default is /usr/local/share/tbarcode7.) Example: `--license=/etc` The name of the license file is always `license.ini`.

Table 36: TBarCode Daemon – General Options

15.2.2 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*	Optional: Changes the size of the memory reserved for barcode creation. *TBarCode Daemon* uses a fixed memory block for the interprocess 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 37: TBarCode Daemon – Daemon and IPC Options

15.3 Error Message and Debug Options

Short	Long	Description
	`--errorfile=`*FILE*	Optional: 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`	Optional: 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 38: TBarCode Daemon – Error Message and Debug Options

15.3.1 Informative Output

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

Table 39: TBarCode Daemon – Informative Output

16 Appendix – ASCII Table

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

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

Hexcode	Symbol	Hexcode	Symbol	Hexcode	Symbol	Hexcode	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 40: ASCII Table

17 Appendix – Knowledge Base

17.1 Unix Printing (HP-UX and Solaris)

17.1.1 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.

17.1.2 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.

17.1.3 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.

17.1.4 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

18 Appendix – Supported Barcodes

Below you find a list of supported barcodes. This list is divided into: 1D Symbologies, Reduced Space Symbologies (RSS), 2D Symbologies, and EAN.UCC Composite Symbologies. Each of the sections is ordered alphabetically.

If you need additional information for a bar code type, check out the document “Barcode Reference” available on www.tec-it.com or contact our support@tec-it.com.

18.1 1D (Linear) Symbologies

63	**Australian Post Customer** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Size:	 “0”..”9”, 8 digits left/right: 6 mm, top/bottom: 2 mm -- 1:1 1B:1S Automatic (symbology specific). see Notes	 12345678

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b63 d12345678$_tbce`
		`$_tbcs -b63 -d12345678$_tbce`	

	Notes:	Barcode height is between 4.2mm and 5.8mm. Length will depend on use of additional bars (code variants Customer 2 and Customer 3). Is used by the Australian Post for marking of shipments. Special code variants are available for redirections, replies, ….	
64	**Australian Post Customer 2** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, “A”..”Z”, “a”..”z”, Space, “#” left/right: 6 mm, top/bottom: 2 mm -- 1:1 1B:1S Automatic (symbology specific). see Australian Post Customer	 12345678ABab

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b64 d12345678ABab$_tbce`
		`$_tbcs -b64 -d12345678Abab$_tbce`	

	Notes:	Same as Australian Post Standard Customer, but with additional 5 characters for customer specific data. The first 8 characters must be digits.	
65	**Australian Post Customer 3** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, “A”..”Z”, “a”..”z”, Space, “#” left/right: 6 mm, top/bottom: 2 mm -- 1:1 1B:1S Automatic (symbology specific). see Australian Post Customer	 12345678ABCabc

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b65 d12345678ABCabc$_tbce`
		`$_tbcs -b65 -d12345678ABCabc$_tbce`	

	Notes:	Same as Australian Post Standard Customer, but with additional 10 characters for customer specific data. The first 8 characters must be digits.	
68	**Australian Post Redirection** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, 8 digits left/right: 6 mm, top/bottom: 2 mm -- 1:1 1B:1S Automatic (symbology specific). see Australian Post Customer	 12345678

	Sample control sequence TBarCode/X v1.x: `$_tbcs b68 d12345678$_tbce` TBarCode/X v2.0: `$_tbcs -b68 -d12345678$_tbce`	
	Notes: Used by the Australian Post	
66	**Australian Post Reply Paid** Valid characters: "0".."9", 8 digits Quiet zone: left/right: 6 mm, top/bottom: 2 mm Module width: -- Standard print ratio: 1:1 Ratio format string: 1B:1S Default check-digit: Automatic (symbology specific). Symbol size: see Australian Post Customer	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b66 d12345678$_tbce` TBarCode/X v2.0: `$_tbcs -b66 -d12345678$_tbce`	
	Notes: Used by the Australian Post	
67	**Australian Post Routing** Valid characters: "0.."9", 8 digits Quiet zone: left/right: 6 mm, top/bottom: 2 mm Module width: -- Standard print ratio: 1:1 Ratio format string: 1B:1S Default check-digit: Automatic (symbology specific). Symbol size: see Australian Post Customer	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b67 d12345678$_tbce` TBarCode/X v2.0: `$_tbcs -b67 -d12345678$_tbce`	
	Notes: Used by the Australian Post	
18	**CodaBar** Valid characters: "0.."9", "-", "$", ":", "/", ".", "+", "A", "B", "C", "D" Quiet zone: left/right: 10X Module width: X = 0.19 mm Standard print ratio: 1:3:1:3 Ratio format string: 1B:2B:1S:2S Default check-digit: None (eCDNone) Symbol size: +/- 0.066mm Module width Deviation	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b18 dA01234:/.+A$_tbce` TBarCode/X v2.0: `$_tbcs -b18 -dA01234:/.+A$_tbce`	
	Notes: "A", "B", "C", "D" are useable as start or stop characters only. Was invented 1972 by Monarch Marking Systems for retail purposes (price marking). 1977 the American Blood Commission defined Codabar 2 as standard symbology for blood banks (=ABC Codabar). Uses 2 element-widths and 4 different start/stop-characters (A, B, C, D). These can be utilized for additional information – e.g. "B1234B". Also known as 2of7 Code. The original symbology used 18 different element widths (standard Codabar). Double symbol is possible. The print ratio should be in the following range: 1:2 -1:3 (Pr >= 2,25:1)	
2	**Code 2 of 5 Standard / Code 2 of 5 Matrix** Valid characters: "0.."9" Quiet zone: left/right: 10X, min. ¼ inch Module width: X>= 0.19 mm Standard print ratio: 1:3:4.5:1:3 Ratio format string: 1B:2B:3B:1S:2S Default check-digit: None (eCDNone) Possible check-digits: Modulo 10 (eCDMod10) Symbol size: --	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b2 d0123456789$_tbce` TBarCode/X v2.0: `$_tbcs -b2 -d0123456789$_tbce`	
	Notes: Self-checking code. For industrial applications, article numbering, photo development, ticketing.	
6	**Code 2 of 5 Data Logic** Valid characters: "0.."9" Quiet zone: left/right: 10X, min. ¼ inch Module width: --	

	Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	1:3:1:3 1B:2B:1S:2S None (eCDNone) Modulo 10 (eCDMod10) --	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b6 d0123456789$_tbce`
		`$_tbcs -b6 -d0123456789$_tbce`	

	Notes:	Proprietary variant of Code 2 of 5 Standard	
4	**Code 2 of 5 IATA** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 "0".."9" left/right: 10X, min. ¼ inch X>= 0.19 mm 1:3:1 1B:2B:1S None (eCDNone) Modulo 10 (eCDMod10) --	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b4 d0123456$_tbce`
		`$_tbcs -b4 -d0123456$_tbce`	

	Notes:	Self-checking code; start/stop-characters identical to 2 of 5 Industry; supports distance reading (> 1m) and can be printed with very simple print techniques. Used for baggage handling in air-transport applications (International Air Transport Agency = IATA)	
7	**Code 2 of 5 Industrial** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 "0".."9" left/right: 10X, min. ¼ inch X>= 0.19 mm 1:3:1 1B:2B:1S None (eCDNone) Modulo 10 (eCDMod10) --	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b7 d0123456$_tbce`
		`$_tbcs -b7 -d0123456$_tbce`	

	Notes:		
3	**Code 2 of 5 Interleaved** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 "0".."9" left/right: 10X, min. ¼ inch X>= 0.19 mm 1:3:1:3 1B:2B:1S:2S None (eCDNone) Modulo 10 (eCDMod10) --	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b3 d0123456789$_tbce`
		`$_tbcs -b3 -d0123456789$_tbce`	

	Notes:	If the number of digits is odd a leading zero will be inserted automatically. This barcode type can encode only an even number of digits. 2 of 5 Interleaved is in wide-spread use (article-numbering, industrial applications). Self-Checking code with high data capacity due to encoding pairs of numbers (the first digit is encoded in the bars, the second in the spaces).	
1	**Code 11** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 "0".."9", "-" left/right: 10X X= 0.191 mm 1:2.24:3.48:1:2.24 1B:2B:3B:1S:2S None (eCDNone) 1 check-digit (eCD1Code11) - or 2 check-digits (eCD2Code11) --	
	Sample control sequence TBarCode/X v1.x:	 `$_tbcs b1 d0-123-456$_tbce`	

	TBarCode/X v2.0:	`$_tbcs -b1 –d0-123-456$_tbce`
	Notes:	Mainly used in telecommunications for marking equipment and components. Was invented in 1977 by INTERMEC. Similar to Matrix 2 of 5; Code is not self-checking, therefore 2 check-digits are recommended. High density code, but requires high-density output device.
8	**Code 39** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, “A”..”Z”, “-“, “.”, Space, “*”, “$”, “/”, “+”, “%” left/right: 10X, min. ¼ inch X >= 0.19 mm 1:3:1:3 1B:2B:1S:2S None (eCDNone) Modulo 43 (eCDMod43), Modulo 11 Weight 7 (eCDMod11W7) H>=15% of L (H>=6.3 mm!) H: Height of the barcode without human readable text L: width of the barcode

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b8 dAB12+$_tbce`
		`$_tbcs -b8 -dAB12+$_tbce`

	Notes:	Start- and stop characters “*” (asterisk) are created automatically and must not be included in the input data. They are not displayed in the human readable text. In heavy use in industry, organizations and commerce. Standardized by ANSI MH 10.8 M-1983 and MIL-STD-1189. Was developed 1974 by INTERMEC. Self-checking code. Code concatenation is possible (if first encoded character is a space subsequent barcodes are concatenated by the scanner); Distance-reading possible (> 1m).
9	**Code 39 Extended** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 ASCII-characters between 0..127 left/right: 10X, min. ¼ inch X >= 0.19 mm 1:3:1:3 1B:2B:1S:2S None (eCDNone) Modulo 43 (eCDMod43), Modulo 11 Weight 7 (eCDMod11W7) H>=15% of L (H>=6.3 mm!) H: Height of the barcode without human readable text L: width of the barcode

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b9 dABab$_tbce`
		`$_tbcs -b9 -dABab$_tbce`

	Notes:	Start- and stop characters “*” (asterisk) are created automatically and must not be included in the input data. They are not displayed in the human readable text. Code 39 Extended is rarely used because Code128 offers much better compression. Code39 Extended uses the same symbology as Code39 but encodes also lower-case letters and special characters („+A“ results in a lower case „a“ when scanned); Scanner must be configured correctly for decoding Code39 Extended.
25	**Code 93** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, “A”..”Z”, “-“, “.”, Space, “$”, “/”, “+”, “%” left/right: 10X, min. ¼ inch X >= 0.19 mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S Automatic (symbology specific). Modulo 47 (eCD2Mod47) --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b25 dABC123-/+$_tbce`
		`$_tbcs -b25 -dABC123-/+$_tbce`

	Notes:	Was invented 1982 by INTERMEC to achieve better information densities (compared to Code39). Code concatenation is possible (if first encoded character is a space subsequent barcodes are concatenated by the scanner).

62	**Code 93 Extended** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 ASCII-characters between 0..127 left/right: 10X, min. ¼ inch X >= 0.19 mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S Automatic (symbology specific). Modulo 47 (eCD2Mod47) --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b62 dABab12-/+$_tbce`
		`$_tbcs -b62 -dABab12-/+$_tbce`

	Notes:	Encodes the complete ASCII-character set. One of the four available control characters is used to encode all ASCII-characters.
20	**Code 128** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 ASCII-characters between 0..127 left/right: 10X, min. ¼ inch X >= 0.19 mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S Automatic (symbology specific). Modulo 103 (eCDCode128) Modulo 10, EAN-14 --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b20 dABab123+/-$_tbce`
		`$_tbcs -b20 -dABab123+/-$_tbce`

	Notes:	Input data is analyzed and the best suitable subset will be used. Subset switching is done automatically when necessary. No user interaction required. Heavily used in all areas. Modern high-density symbology. Invented 1981 by Computer Identics. In conjunction with the special control sequence "FNC1" Code128 is also known as UCC128 / EAN128 for retail purposes. 3 different internal characters sets are used (Code128A = Upper Case + ASCII Control Sequences, Code128B = Upper / Lower Case, Code128C = Numeric with doubled density); Character set switching inside the code with special control sequences (FNC1-4); Code128 uses a built-in check-digit (Mod103). This check-digit is part of the code and cannot be omitted. It is never printed in the human readable text. Scanners are checking it when reading a code but do not deliver the check digit to connected systems.
59	**Code 128 Subset A** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 ASCII-characters between 0..127 left/right: 10X, min. ¼ inch X >= 0.19 mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S Automatic (symbology specific). Modulo 103 (eCDCode128) Modulo 10, EAN-14 --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b59 dABab123+/-$_tbce`
		`$_tbcs -b59 -dABab123+/-$_tbce`

	Notes:	Symbology subset A is used. Suitable for encoding Upper Case + ASCII Control Sequences. Switches to other Code128s subset when required.
60	**Code 128 Subset B** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 ASCII-characters between 0..127 left/right: 10X, min. ¼ inch X >= 0.19 mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S Automatic (symbology specific). Modulo 103 (eCDCode128) Modulo 10, EAN-14 --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b60 dABab123+/-$_tbce`
		`$_tbcs -b60 -dABab123+/-$_tbce`

	Notes:	Symbology subset B is used. Suitable for encoding lower & upper case letters. Switches to other Code128 subsets when required.
61	**Code 128 Subset C** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 ASCII-characters between 0..127 left/right: 10X, min. ¼ inch X >= 0.19 mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S Automatic (symbology specific). Modulo 103 (eCDCode128) Modulo 10, EAN-14 --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b61 dABab123+/-$_tbce`
		`$_tbcs -b61 -dABab123+/-$_tbce`

	Notes:	Symbology subset C is used. Suitable for encoding digits. Switches to other Code128 subsets when required.
22	**Deutsche Post Identcode** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, 11 digits + 1 check digit left/right: 10X, min. ¼ inch -- 1:3:1:3 1B:2B:1S:2S Automatic (symbology specific). DP Identcode (eCDDPIdent) --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b22 d12345678$_tbce`
		`$_tbcs -b22 -d012345678$_tbce`

	Notes:	Used by Deutsche Post. The code is basically a 2 of 5 interleaved enhanced with a special check-digit calculation.
21	**Deutsche Post Leitcode** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, 13 digits + 1 check digit left/right: 10X, min. ¼ inch -- 1:3:1:3 1B:2B:1S:2S Automatic (symbology specific). DP Leitcode (eCDDPLeit) --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b21 d12345678$_tbce`
		`$_tbcs -b21 -d012345678$_tbce`

	Notes:	Used by Deutsche Post. The code is basically a 2 of 5 interleaved enhanced with a special check-digit calculation. Encoded ZIP-Code, Street and number of the shipment.
10	**EAN-8** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, 7 digits + 1 check digit left/right: 7X X=0.33mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S EAN-8 (eCDEAN8) User supplied Standardized symbol sizes. See EAN.

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b10 d9031101$_tbce`
		`$_tbcs -b10 -d9031101$_tbce`

	Notes:	Reserved for European Article Numbering (EAN) – EAN 8 is used for marking small articles with restricted space. It encodes a unique article number. This number is maintained by EAN and encodes manufacturer and product. The check digit is calculated automatically if not specified in the input data (that is when only 7 digits are used for creating the code).
11	**EAN-8 with 2 Digits Add-On** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string:	 “0”..”9”, 9 digits + 1 check digit left: 7-10X, right: 5X X=0.33mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S

	Default check-digit: EAN-8 (eCDEAN8) Possible check-digits: User supplied Symbol size: Standardized symbol sizes. See EAN.	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b11 d903110112$_tbce` TBarCode/X v2.0: `$_tbcs -b11 -d903110112$_tbce`	
	Notes: Same as EAN-8, but with 2 add-on digits (used for price or weight) enclosed. If not specified in the input data (e.g. 9031101712), the check digit will be calculated automatically. EAN8+2 is mainly used for paperbacks or newspapers. Encoded are a 2(3) digits country code and a 4(5) article code.	
12	**EAN-8 with 5 Digits Add-On** Valid characters: "0".."9", 12 digits + 1 check digit Quiet zone: left: 7-10X, right: 5X Module width: X=0.33mm Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-8 (eCDEAN8) Possible check-digits: User supplied Symbol size: Standardized symbol sizes. See EAN.	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b12 d903110112345$_tbce` TBarCode/X v2.0: `$_tbcs -b12 -d903110112345$_tbce`	
	Notes: Same as EAN-8, but with 5 add-on digits (for price or weight) enclosed. If not specified in the input data (e.g. 9031101712345), the check digit will be calculated automatically.	
13	**EAN-13** Valid characters: "0".."9", 12 digits + 1 check digit Quiet zone: left: 11X, right: 7X Module width: X=0.33mm Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-13 (eCDEAN13) Possible check-digits: User supplied Symbol size: Standardized symbol sizes. See EAN.	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b13 d978020137968$_tbce` TBarCode/X v2.0: `$_tbcs -b13 -d978020137968$_tbce`	
	Notes: Check digit is calculated automatically if not specified in the input data (that is when only 12 digits are used for creating the code). Reserved for European Article Numbering (EAN) – EAN 13 is used for identifying articles or products uniquely. Encoded is a 2-digit country code, 5-digits manufacturer code and a 5 digits products code. JAN and IAN are identical to EAN-13.	
14	**EAN-13 with 2 Digits Add-On** Valid characters: "0".."9", 14 digits + 1 check digit Quiet zone: left: 7-10X, right: 5X Module width: X=0.33mm Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-13 (eCDEAN13) Possible check-digits: User supplied Symbol size: Standardized symbol sizes. See EAN.	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b14 d97802013796812$_tbce` TBarCode/X v2.0: `$_tbcs -b14 -d97802013796812$_tbce`	
	Notes: Same as EAN-13, but with 2 add-on digits enclosed (like EAN8 + 2). If not specified in the input data (e.g. 978020137968612), the check digit will be calculated automatically.	
15	**EAN-13 with 5 Digits Add-On** Valid characters: "0".."9", 17 digits + 1 check digit Quiet zone: left: 7-10X, right: 5X Module width: X=0.33mm Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-13 (eCDEAN13) Possible check-digits: User supplied	

	Symbol size:	Standardized symbol sizes. See EAN.	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b15 d97802013796812345$_tbce`
		`$_tbcs -b15 -d97802013796812345$_tbce`	

	Notes:	Same as EAN-13, but with 5 add-on digits enclosed (like EAN8 + 5). If not specified in the input data (e.g. 978020137968612345), the check digit will be calculated automatically.	
72	**EAN-14** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 ASCII-characters between 0..127, 13 digits + 1 check digit see UCC/EAN-128, ITF-14 see UCC/EAN-128, ITF-14 see UCC/EAN-128, ITF-14 see UCC/EAN-128, ITF-14 EAN-14 (eCDEAN14) User supplied see UCC/EAN-128, ITF-14	 (01)ABCabc+/-12347

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b72 dABCabc+/-1234$_tbce`
		`$_tbcs -b72 -dABCabc+/-1234$_tbce`	

	Notes:	EAN-14 encodes the „GTIN“ (Global Trade Item Number). Within the EAN UCC System you can use 2 symbologies for encoding the GTIN: UCC/EAN-128 and ITF-14. Here we use EAN-128 with AI=01. The AI is prefixed automatically, it must not be part of the input data. The check digit is calculated automatically if not specified in the input data (that is when only 13 digits are used). Used for numbering trade items.	
16	**EAN-128** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 ASCII-characters between 0..127 left/right: 10X but min. ¼ in see Code128 see Code128 see Code128 Automatic (symbology specific). Modulo 103 (eCDEAN128) Modulo 10, EAN-14 see Code 128	 EAN128

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b16 dABab123+/-$_tbce`
		`$_tbcs -b16 -dABab123+/-$_tbce`	

	Notes:	EAN-128 is based upon Code-128, but with the FNC1 function character at 2nd position. This allows scanners and data processing software to differentiate EAN-128 from other symbologies. The FNC1 at 2nd position is inserted automatically by TEC-IT software. Symbology internal check digits (Mod 103) are also calculated automatically. Within the EAN UCC System you can use Application Identifiers (AI's) to prefix the encoded data. EAN-128 is in wide spread use (retail, logistics, food and beverage, …). Beside the article-number EAN-128 encodes also quantities, weights, prices, dates, … in a structured way. This is supported by the use of so-called Application Identifiers (or AI's).	
28	**Flattermarken** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 "0".."9" Application dependent 2-3 mm 1:1 1B:1S None (eCDNone) Symbol height between 5 and 10mm	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b28 r90 d1111$_tbce`
		`$_tbcs -b28 -r90 -d1111$_tbce`	

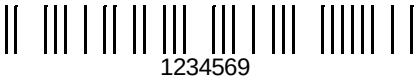
	Notes:	For the recognition of the correct sequence of pages. **Error! Objects cannot be created from editing field codes.**	
69	**ISBN Code** Valid characters: Quiet zone: Module width: Standard print ratio:	 "0".."9", 17 digits + 1 check digit See EAN13 + 5 See EAN13 + 5 See EAN13 + 5	 12345 9 780201 379686

	Ratio format string: Default check-digit: Possible check-digits: Symbol size:	See EAN13 + 5 EAN-13 (eCDEAN13) User supplied See EAN13 + 5	

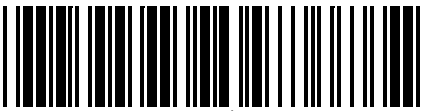
	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b69 d97802013796812345$_tbce`
		`$_tbcs -b69 -d97802013796812345$_tbce`	

	Notes:	Same as EAN-13 with 5 digits add on.	
76	**Japanese Postal** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, “A”..”Z”, “-“, 7 digits (ZIP code) + additional data left/right/top/bottom: 2 mm -- 1:1 1B:1S Automatic (symbology specific). --	 1234567

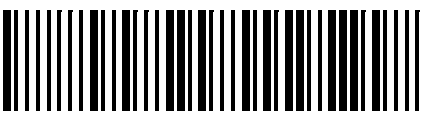
	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b76 d1234567$_tbce`
		`$_tbcs -b76 -d1234567$_tbce`	

	Notes:	This code is used by the Japanese Postal system. You can encode 7 digits followed by block and street number (uppercase alphanumeric). Special compaction mode of Japanese characters can be enabled on demand (Format Parameter “J”).	
77	**Korean Post Authority** Valid characters: Check digit method: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9”, 6 digits + 1 check digit Check digit included in the code 10X (not exactly specified) -- 1:3:4 1B:1S:2S Automatic (symbology specific). Modulo10 (eCDMod10Kor) --	 1234569

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b77 d123456$_tbce`
		`$_tbcs -b77 -d123456$_tbce`	

	Notes:	This code is used by the Korean Postal system. Encoded is 6-digit ZIP plus one check-digit	
50	**LOGMARS** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size: 	 “0”..”9”, “A”..”Z”, “+“, “-”, “*”, “/”, ”.”, ”$”, Space left/right: 10X, min. ¼ inch X>=0.19 mm 1:3:1:3 1B:2B:1S:2S None (eCDNone) Modulo 43 (eCDMod43), Modulo 11 Weight 7 (eCDMod11W7) H>=15% of L (H>=6.3 mm!) H: Height of barcode-symbol without human readable text L: Width of barcode	 AB12$+

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b50 dAB12$+$_tbce`
		`$_tbcs -b50 -dAB12$+$_tbce`	

	Notes:	Is used by the US Department of Defense (Logistics Applications of Automated Marking and Reading Symbols). The LOGMARS symbology is of variable length with Module 43 check digit.	
47	**MSI** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9” left/right: 12X -- 1:2:1:2 1B:2B:1S:2S MSI 1 digit (eCDMSI1) User supplied and MSI 2 digit (eCDMSI2) 14 digits incl. check-digits	 012345674

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b47 d01234567$_tbce`
		`$_tbcs -b47 -d01234567$_tbce`

	Notes:	The MSI-Code is a universal code. It is a variant of the Plessey-Code.
75	**NVE-18** Valid characters: Check digit method: Default check digit: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9” Modulo10 Modulo10 left/right: 10X, min. ¼ inch X >= 0.19 mm 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S Automatic (symbology specific). Modulo 10 (eCDMod10) and Modulo 103 (eCDEAN128) --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b75 d12345678901234567$_tbce`
		`$_tbcs -b75 -d12345678901234567$_tbce`

	Notes:	NVE stands for “Nummer der Versandeinheit”. This type uses EAN-128 symbology with AI prefix 00. Similar to SSCC-18. The AI “00” is inserted automatically and must not be included in the input data.
51	**Pharmacode One-Track** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 “0”..”9” or binary left/right: 6 mm 2-3 mm 1:3:2:4:2:3 1B:2B:1C:2C:1S:2S None (eCDNone) 5-10 mm height

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b51 d1234567890$_tbce`
		`$_tbcs -b51 -d1234567890$_tbce`

	Notes:	This code was specified/invented by Laetus® . It I used in pharmaceutical areas. This barcode supports colored bars. The data is encoded directly in the property *Text*: “0” (or “b”) is used for a narrow bar “1” (or “c”) is used for wide bars Color is specified by \Crrggbb (RGB hex). The barcode *Format* must be set to „D“, *Translate EscapeSequences* must be activated. After a color-change the following bar is wider – see also print ratio (above): narrow bar : wide bar : narrow bar after color change : wide bar after color change : narrow space : wide space
53	**Pharmacode Two-Track** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Symbol size:	 numeric [0..9] and generic; left/right: 6 mm -- 1:1 1B:1S None (eCDNone) see Notes

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b53 d1234567890$_tbce`
		`$_tbcs -b53 -d1234567890$_tbce`

	Notes:	Specification by Laetus® Two-Track bar width: 1 mm; Space bars: 1 mm; bar height above/below: 4-6 mm; height of the long bar: 8-12 mm; For medicine packing in pharmaceutically area; for small labels; Pharmacode will be used without readable text; Pharmacode assigns numeric values to the bars; readable very fast (200 readings per second); be able to print in several colors; printing tolerance is big (with normal aberrance it is not unreadable); universal code;
52	**Pharma Zentralnummer (PZN)** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit:	 “0”..”9”, 6 digits + 1 check digit see Code 39 see Code 39 see Code 39 see Code 39 PZN check digit (eCDPZN)

	Possible check-digits: Symbol size:	User supplied see Code 39	

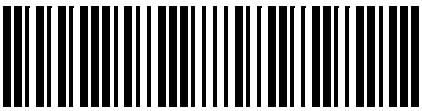
	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b52 d123456$_tbce`
		`$_tbcs -b52 -d123456$_tbce`	

	Notes:	PZN uses Code 39 as base symbology. It has a special check digit and the human readable text contains always the prefix “PZN-“ (not encoded in the barcode data).	
82	**Planet 12** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, 11 digits + 1 check digit left/right: 1/25 inch top/bottom: 1/8 inch -- 1:1 1B:1S Modulo 10 Planet (eCDMod10Pla) User supplied 11 digits + 1 check digit	 123456789014

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b82 d12345678901$_tbce`
		`$_tbcs -b82 -d12345678901$_tbce`	

	Notes:	This code was developed for the U.S. Postal Services. It is a 3-of-5 variant of the Postnet barcode	
83	**Planet 14** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, 13 digits + 1 check digit left/right: 1/25 inch top/bottom: 1/8 inch -- 1:1 1B:1S Modulo 10 Planet (eCDMod10Pla) User supplied 13 digits + 1 check digit	 12345678901239

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b83 d1234567890123$_tbce`
		`$_tbcs -b83 -d1234567890123$_tbce`	

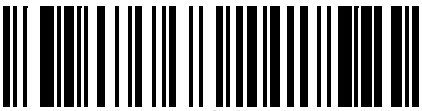
	Notes:	This code was developed for the U.S. Postal Services. It is a 3-of-5 variant of the Postnet barcode	
46	**Plessey Code** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 numeric [0..9] left/right: 12X -- 1:2:1:2 1B:2B:1S:2S Plessey (eCDPlessey) User supplied --	 ABC1233B

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b46 dABC123$_tbce`
		`$_tbcs -b46 -dABC123$_tbce`	

	Notes:	It has been used primarily in libraries and stores. It based on the pulse breadth modulation code, which was developed by the company "Plessey Company Limited". If you put in an additional modulo 10- or modulo 11-check digit, the code can have 14 characters, otherwise only 13 characters.	
70	**Royal Mail 4 State (RM4SCC)** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, “A”..”Z” left/right: 2 mm -- 1:1 1B:1S Automatic (symbology specific). User supplied max. 9 digits without check-digits	 12345678

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b70 d1234567ABC$_tbce`
		`$_tbcs -b70 -d1234567ABC$_tbce`	

	Notes:	This code is used in mass-mailing applications (Cleanmail, Mailsort) of the Royal Mail, UK and Singapore (also called SinPost barcode). Encoded are ZIPs.	

48	**SSCC-18** Valid characters: “0”..”9”, 17 digits + 1 check digit Quiet zone: see EAN 128, sometimes ¼ inch Module width: see EAN 128 Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: Automatic (symbology specific). Modulo 10 (eCDMod10) and Modulo 103 (eCDEAN128) Symbol size: see EAN 128	 (00)012345678901234567
	Sample control sequence TBarCode/X v1.x: `$_tbcs b48 d012345678901234567$_tbce` TBarCode/X v2.0: `$_tbcs -b48 -d012345678901234567$_tbce`	
	Notes: Used to encode the Serial Shipping Container Code based upon EAN-128 symbology with AI (00). The check digit is encoded automatically if 17 digits are used for the input data.	
32	**Telepen Alpha** Valid characters: ASCII characters between 0..127 Quiet zone: n/a Standard print ratio: 1:3:1:3 Ratio format string: 1B:2B:1S:2S Default check-digit: None (eCDNone) Symbol size: --	 12Az
	Sample control sequence TBarCode/X v1.x: `$_tbcs b32 d12Az$_tbce` TBarCode/X v2.0: `$_tbcs -b32 -d12Az$_tbce`	
	Notes: Telepen Alpha is the alphanumeric variant of Telepen.	
33	**UCC / EAN-128** Valid characters: ASCII characters between 0..127 Quiet zone: left/right: 10X, min ¼ inch Module width: see Code128 Standard print ratio: see Code128 Ratio format string: see Code128 Default check-digit: Automatic (symbology specific). Modulo 103 (eCDEAN128) Possible check-digits: Modulo 10, EAN-14 Symbol size: max width 165 mm	 ABab-/+
	Sample control sequence TBarCode/X v1.x: `$_tbcs b33 dABab-/+$_tbce` TBarCode/X v2.0: `$_tbcs -b33 -dABab-/+$_tbce`	
	Notes: EAN-128 is based upon Code-128 symbology. To identify an EAN-128 symbology the FNC1 symbology character is placed on the first position (encoded automatically by TEC-IT Software). Data is encoded with Application Identifiers (AI). UCC/EAN-128 is used for marking transport-units in supply chains.	
17	**UPC 12 Digits** Valid characters: “0”..”9”, 11 digits + 1 check digit Quiet zone: see UPC-A Module width: see UPC-A Standard print ratio: see UPC-A Ratio format string: see UPC-A Default check-digit: UPC-A (eCDUPCA) Possible check-digits: User supplied Symbol size: see UPC-A	 1 23456 78901 2
	Sample control sequence TBarCode/X v1.x: `$_tbcs b17 d12345678901$_tbce` TBarCode/X v2.0: `$_tbcs -b17 -d12345678901$_tbce`	
	Notes: UPC-A and UPC-12 are identical. Check digit is calculated automatically if not specified in the input data (that is when only 11 digits are used for creating the code).	
34	**UPC Version A** Valid characters: “0”..”9”, 11 digits + 1 check digit Quiet zone: 9X Module width: 0,33 mm Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: UPC-A (eCDUPCA) Possible check-digits: User supplied Symbol size: H=26.26mm; B=37.29mm; variable size	 0 12345 67890 5
	Sample control sequence	

	TBarCode/X v1.x: TBarCode/X v2.0:	`$_tbcs b34 d72527273070$_tbce` `$_tbcs -b34 -d72527273070$_tbce`
	Notes:	Check digit is calculated automatically if not specified in the input data (that is when only 11 digits are used for creating the code). Used for article bar coding. Used in the US (similar to EAN13) for marking of products in retail applications. The article number is maintained by (UCC) and identifies manufacturer and product; The code (11 digits + 1 check-digit) is built from one system-digit, 5 digits manufacturer code, 5 digits product code.
35	**UPC Version A, 2 Digits Add-On** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, 13 digits + 1 check digit left: 9-12X, right: 5X see UPC-A see UPC-A see UPC-A UPC-A (eCDUPCA) User supplied see UPC-A

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b35 d7252727207012$_tbce`
		`$_tbcs -b35 -d7252727207012$_tbce`

	Notes:	Same as UPC version A, but with 2 add-on digits enclosed. If not specified in the input data (e.g. 725272720707**1**2), the check digit will be calculated automatically. The check digit is not displayed in the human readable text.
36	**UPC Version A, 5 Digits Add-On** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, 16 digits + 1 check digit left: 9-12X, right: 5X see UPC-A see UPC-A see UPC-A UPC-A (eCDUPCA) User supplied see UPC-A

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b36 d72527272070123 45$_tbce`
		`$_tbcs -b36 -d72527272070123 45$_tbce`

	Notes:	Same as UPC version A, but with 5 add-on digits enclosed. If not specified in the input data (eg. 725272720707**1**2345), the check digit will be calculated automatically. The check digit is not displayed in the human readable text.
37	**UPC Version E** Valid characters: Quiet zone: Module width: Standard print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 “0”..”9”, 7 digits + 1 check digit left: 9X, right: 7X -- 1:2:3:4:1:2:3:4 1B:2B:3B:4B:1S:2S:3S:4S UPC-E (eCDUPCE) User supplied --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b37 d0123456$_tbce`
		`$_tbcs -b37 -d0123456$_tbce`

	Notes:	Used for product marking. Check digit is calculated automatically if not specified in the input data (that is when only 7 digits are used for creating the code). Used for article bar coding. Code must begin with “0” or “1”.
38	**UPC Version E, 2 Digits Add-On** Valid digits: Quiet zone: Module width: Default check-digit: Ratio format string: Check digit method: Possible check-digits: Symbol size:	 “0”..”9”, 9 digits + 1 check digit left: 9-12X, right: 5X see UPC-E see UPC-E see UPC-E UPC-E (eCDUPCE) User supplied --

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b38 d012345612$_tbce`
		`$_tbcs -b38 -d012345612$_tbce`

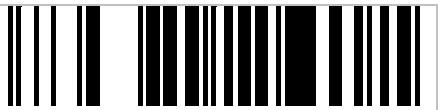
	Notes:	Same as UPC version E, but with 2 add-on digits enclosed. If not specified in the input data (eg. 0123456**5**12), the check digit will be calculated automatically. The check digit is not displayed in the human readable text.

39	**UPC Version E, 5 Digits Add-On** Valid digits: “0”..”9”, 12 digits + 1 check digit Quiet zone: left: 9-12X, right: 5X Module width: see UPC-E Standard print ratio: see UPC-E Ratio format string: see UPC-E Default check-digit: UPC-E (eCDUPCE) Possible check-digits: User supplied Symbol size: --	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b39 d012345612345$_tbce` TBarCode/X v2.0: `$_tbcs -b39 -d012345612345$_tbce`	
	Notes: Same as UPC version E, but with 5 add-on digits enclosed. If not specified in the input data (e.g. 012345**5**12345), the check digit will be calculated automatically. The check digit is not displayed in the human readable text.	
85	**USPS OneCode 4-State Customer Barcode** Valid characters: “0”..”9”, 20 digits + 0, 5, 9, or 11-digit ZIP Code. Quiet zone: vertical: 1/25 inch horizontal: 1/8 inch Module width: -- Standard print ratio: 1:1 Ratio format string: 1B:1S Default check-digit: Automatic (symbology specific). Symbol size: Up to 31 digits	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b85 d12345678901234567890123 45$_tbce` TBarCode/X v2.0: `$_tbcs -b85 -d12345678901234567890123 45$_tbce`	
	Notes: This Symbology is also known as OneCode 4CB, USPS 4CB, 4-CB, 4-State Customer Barcode, USPS OneCode Solution Barcode. Encoded Data: Barcode ID (1st digit: 0-9; 2nd digit: 0-4) Special Services (Range: 000-999) Customer ID (Range: 000000-999999) Sequence Number (Range: 000000000-999999999) Delivery Point ZIP Code (0, 5, 9, or 11-digit ZIP Code)	
40	**USPS Postnet 5** Valid characters: “0”..”9”, 5 digits + 1 check digit Quiet zone: vertical: 1/25 inch horizontal: 1/8 inch Module width: -- Standard print ratio: 1:1 Ratio format string: 1B:1S Default check-digit: POSTNET (eCDPostNet) Symbol size: 5 digits, 1 check digit	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b40 d12345$_tbce` TBarCode/X v2.0: `$_tbcs -b40 -d12345$_tbce`	
	Notes: Check digit is calculated automatically. It cannot be specified in the input data. This code is used by the United States Postal Services for mass-mailing applications. Encoded is ZIP-code using max. 5 digits (A Field).	
41	**USPS Postnet 6** Valid characters: “0”..”9”, 5 digits + 1 check digit Quiet zone: vertical: 1/25 inch horizontal: 1/8 inch Module width: -- Standard print ratio: 1:1 Ratio format string: 1B:1S Default check-digit: POSTNET (eCDPostNet) Possible check-digits: User supplied Symbol size: 5 digits, 1 check digit	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b41 d12345$_tbce` TBarCode/X v2.0: `$_tbcs -b41 -d12345$_tbce`	
	Notes: Check digit is calculated automatically if not specified in the input data (that is when only 5 digits are used for creating the code). This code is used by the United States Postal Services for mass-mailing applications. Encoded is a ZIP-code using max. 5 digits (A Field).	
42	**USPS Postnet 9** Valid characters: “0”..”9”, 9 + 1 check digit Quiet zone: vertical: 1/25 inch	

		horizontal: 1/8 inch	
	Module width:	--	
	Standard print ratio:	1:1	
	Ratio format string:	1B:1S	
	Default check-digit:	POSTNET (eCDPostNet)	
	Symbol size:	9 digits, 1 check digit	
	Sample control sequence		
	TBarCode/X v1.x:	`$_tbcs b42 d12345678$_tbce`	
	TBarCode/X v2.0:	`$_tbcs -b42 -d12345678$_tbce`	
	Notes:	Check digit is calculated automatically. It cannot be specified in the input data. This code is used by the United States Postal Services for mass-mailing applications. Encoded are a ZIP-code using 5 digits, additional 4 digits, 1 check-digit.	
43	**USPS Postnet 10**		 1234567895
	Valid characters:	"0".."9", 9 digits + 1 check digit	
	Quiet zone:	vertical: 1/25 inch horizontal: 1/8 inch	
	Module width:	--	
	Standard print ratio:	1:1	
	Ratio format string:	1B:1S	
	Default check-digit:	POSTNET (eCDPostNet)	
	Possible check-digits:	User supplied	
	Symbol size:	9 digits, 1 check digit	
	Sample control sequence		
	TBarCode/X v1.x:	`$_tbcs b43 d123456789$_tbce`	
	TBarCode/X v2.0:	`$_tbcs -b43 -d123456789$_tbce`	
	Notes:	Check digit is calculated automatically if not specified in the input data (that is when only 9 digits are used for creating the code). Encoded are a ZIP-code using 5 digits, additional 4 digits, 1 check-digit.	
44	**USPS Postnet 11**		 123456789014
	Valid characters:	"0".."9", 11 digits + 1 check digit	
	Quiet zone:	vertical: 1/25 inch horizontal: 1/8 inch	
	Module width:	--	
	Standard print ratio:	1:1	
	Ratio format string:	1B:1S	
	Default check-digit:	POSTNET (eCDPostNet)	
	Symbol size:	11 digits, 1 check digit	
	Sample control sequence		
	TBarCode/X v1.x:	`$_tbcs b44 d12345678901$_tbce`	
	TBarCode/X v2.0:	`$_tbcs -b44 -d12345678901$_tbce`	
	Notes:	Check digit is calculated automatically. It cannot be specified in the input data. Encoded are a ZIP-code, additional 4 to 9 digits, 1 check-digit.	
45	**USPS Postnet 12**		 123456789014
	Valid characters:	"0".."9", 11 digits + 1 check digit	
	Quiet zone:	vertical: 1/25 inch horizontal: 1/8 inch	
	Module width:	--	
	Standard print ratio:	1:1	
	Ratio format string:	1B:1S	
	Default check-digit:	POSTNET (eCDPostNet)	
	Possible check-digits:	User supplied	
	Symbol size:	1 digits, 1 check digit	
	Sample control sequence		
	TBarCode/X v1.x:	`$_tbcs b45 d12345678901$_tbce`	
	TBarCode/X v2.0:	`$_tbcs -b45 -d12345678901$_tbce`	
	Notes:	Check digit is calculated automatically if not specified in the input data (that is when only 9 digits are used for creating the code). Encoded are a ZIP-code, additional 4 to 9 digits, 1 check-digit.	

Table 41: Supported Linear Barcode Symbologies and Enumerators

18.2 Reduced Space Symbologies (RSS)

29	**RSS-14**		 (01)01234567890128
	Valid characters:	"0".."9"	
	Quiet zone:	none required (1X recommended)	
	Module width:	--	

	Print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	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 EAN 14 (eCDEAN14) User supplied 13 digits, 1 check digit, AI 01 is encoded automatically	

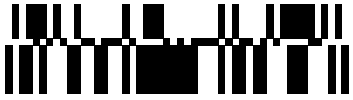
	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b29 d01234567890128$_tbce`
		`$_tbcs -b29 -d01234567890128$_tbce`	

	Notes:	Used to encode the GTIN (Global Trade Item Number) with AI "01". The GTIN consists of a packaging indicator (0..9) followed by a 12 digit number (taken from the EAN-13 article number system) followed by a check digit. The check digit on the 14th position is calculated automatically (if not provided in the input data). The height of the symbol should be at least 33X to support Omni directional scanning (X…module width). The software prefixes the bar code data with the AI "01" automatically, so do not provide the "01 with your input data.	
78	**RSS-14 Truncated** Valid characters: Quiet zone: Module width: Print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 "0".."9" none required (1X recommended) -- 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 EAN 14 (eCDEAN14) User supplied 13 digits, 1 check digit, AI 01 is encoded automatically	 (01)01234567890128

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b78 d01234567890128$_tbce`
		`$_tbcs -b78 -d01234567890128$_tbce`	

	Notes:	Similar to RSS-14 but height should be at least 13X (omni-directional scanning may not be possible).	
30	**RSS Limited** Valid characters: Quiet zone: Module width: Print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol size:	 "0".."9" none required (1X recommended) -- 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 EAN 14 (eCDEAN14) User supplied 13 digits, 1 check digit	 (01)01234567890128

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b30 d0123456789012$_tbce`
		`$_tbcs -b30 -d0123456789012$_tbce`	

	Notes:	Similar to RSS-14 but smaller in size and limited to packaging indicator 0 or 1 (first digit).	
79	**RSS-14 Stacked**[5] Valid characters: Quiet zone: Module width: Print ratio: Ratio format string: Default check-digit: Possible check-digits: Symbol Size:	 "0".."9", 13 digits + 1 check digit none required (1X recommended) -- 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 EAN 14 (eCDEAN14) User supplied --	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b79 d1234567890123$_tbce`
		`$_tbcs -b79 -d1234567890123$_tbce`	

	Notes:	Similar to RSS-14 but split into 2 rows to make the symbol smaller. Used for pharmaceutical packaging. No omni-directional scanning.	

[5] needs a 2D license of TBarCode (Multi-Row symbology)

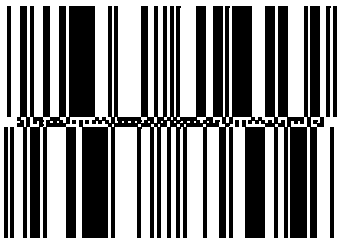
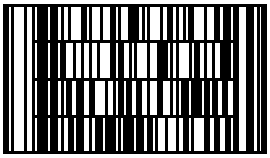
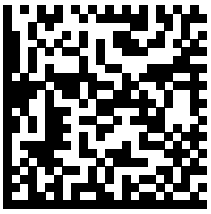
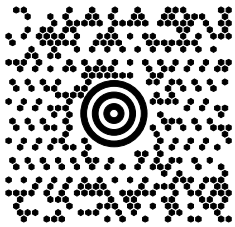
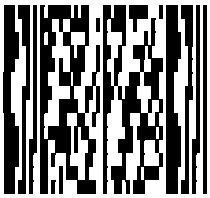
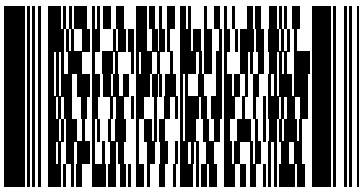
80	**RSS-14 Stacked Omni directional**[6] Valid characters: “0”..”9”, 13 digits + 1 check digit Quiet zone: none required (1X recommended) Module width: -- Print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied Symbol Size: --	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b80 d1234567890123$_tbce` TBarCode/X v2.0: `$_tbcs -b80 -d1234567890123$_tbce`	
	Notes: This version of RSS-14 Stacked supports omni-directional scanning.	
31	**RSS Expanded** Valid characters: “A”..”Z”, “a”..”z”, “0”..”9” + ISO 646 character set Quiet zone: none required (1X recommended) Module width: -- Print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: None (eCDNone). Possible check-digits: Modulo 10 (eCDMod10). EAN-14 (eCDEAN14) Symbol Size: Numeric: 74 digits Alphanumeric: 41 characters	 ABab+
	Sample control sequence TBarCode/X v1.x: `$_tbcs b31 dABab+$_tbce` TBarCode/X v2.0: `$_tbcs -b31 -dABab+$_tbce`	
	Notes: Variable length symbology; Encodes up to 74 numeric or 41 alphabetic characters. Data should be encoded with Application Identifiers. Omni-directional scanning is possible.	
81	**RSS Expanded Stacked**[7] Valid characters: “A”..”Z”, “a”..”z”, “0”..”9” + ISO 646 char set Quiet zone: none required (1X recommended) Module width: -- Print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: None (eCDNone). Possible check-digits: Modulo 10 (eCDMod10). EAN-14 (eCDEAN14) Symbol Size: --	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b81 d1234567890Az+$_tbce` TBarCode/X v2.0: `$_tbcs -b81 -d1234567890Az+$_tbce`	
	Notes: Stacked version of RSS Expanded. The number of data segments per row can vary between 4 and 22. The default number of data segments is 4.	

Table 42: Supported RSS Barcode Symbologies and Enumerators

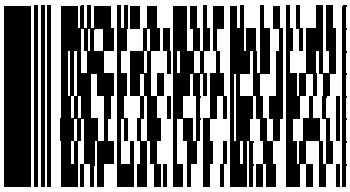
18.3 2D Symbologies

[6] needs a 2D license of TBarCode (Multi-Row symbology)
[7] needs a 2D license of TBarCode (Multi-Row symbology)

74	**Codablock F** Valid characters: ASCII 0-127 + ISO 8859-1 Quiet-zone: left/right/ top/bottom: 10X Module-width: X>=0.19mm Print-ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: Automatic (symbology specific). Size: 2 - 44 rows; 4 - 62 characters per row	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b74 dCodablock F$_tbce` TBarCode/X v2.0: `$_tbcs -b74 –d"Codablock F"$_tbce`	
	Notes: “Stacked Code128” symbology, based upon Code 128. Each row is a single Code 128 symbol extended with row indicator information and additional check-digits. The UCC/EAN format indicator is supported.	
71	**Data Matrix** Valid characters: Alphanumeric (ASCII 0.. 255) and/or bytes Quiet-zone: left/right/ top/bottom: 1X Module-width: -- Print-ratio: 1:1 Ratio format string: 1B:1S Default check-digit: Automatic (symbology specific). Size: .001 till 14.0 square inch	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b71 dHello - this is a Data Matrix by TEC-IT$_tbce` TBarCode/X v2.0: `$_tbcs -b71 –d"Hello - this is a DataMatrix by TEC-IT"$_tbce`	
	Notes: 2D symbology to encode large quantities of data. Size adjusts automatically depending on input data. For encoding large amounts of data, also ideal for marking small objects. Developed for the Space Shuttle Program, enhanced by NASA and Symbology Research Center. Standard in the following areas: automotive, aviation (SPEC2000), pharmaceutical areas. Data Matrix implementation by TEC-IT complies to ECC200, ANSI/AIM BC11, ISO/IEC 16022, Department of Defense UID, MIL-STD-130L and other specifications that require ECC200. Supported encoding modes BASE256, C40, TEXT and ASCII.	
57	**MaxiCode** Valid characters: Alphanumeric (ASCII 0.. 255) and/or bytes Default Mode: Mode-4 (standard symbol) Quiet-zone: left/right/ top/bottom: 1X Module-width: -- Print-ratio: n/a Ratio format string: n/a Default check-digit: Automatic (symbology specific). Size: Fix: 1.11 x 1.054 inch	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b57 dHello - this is a MaxiCode by TEC-IT$_tbce` TBarCode/X v2.0: `$_tbcs -b57 –d"Hello - this is a MaxiCode by TEC-IT"$_tbce`	
	Notes: Used (and invented) by UPS®. Modes for including postal information (SCM) can be adjusted. Printing size is set to a norm value. Made up of offset rows of hexagonal modules arranged around a unique finder pattern; Supports 4 different modes of operation; UPS Modes are Mode 2 (US Carrier) and Mode 3 (Intern. Carrier). A separate documentation about MaxiCode is available on request.	
84	**MicroPDF417** Valid characters: Alphanumeric and/or bytes Quiet-zone: left/right: 1X Module-width: -- Print-ratio: 1:2:3:4:5:6:1:2:3:4:5:6 Ratio format string: 1B:2B:3B:4B:5B:6B:1S:2S:3S:4S:5S:6S Default check-digit: Automatic (symbology specific). Size: --	
	Sample control sequence TBarCode/X v1.x: `$_tbcs b84 dThis is a MicroPDF417$_tbce` TBarCode/X v2.0: `$_tbcs -b84 –d"This is a MicroPDF417"$_tbce`	
	Notes: 2D symbology (multi-row) to encode large quantities of data. Data representation is	

		divided into rows and columns that adjust automatically (depending on input data) or can be set by printer commands.	
55	**PDF417** Valid characters: Quiet-zone: Module-width: Print-ratio: Ratio format string: Default check-digit: Size:	 Alphanumeric (ASCII 0.. 255) and/or bytes left/right: 2X -- 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 Automatic (symbology specific). --	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b55 dHello - this is a PDF417 by TEC-IT$_tbce`
		`$_tbcs -b55 –d"Hello - this is a PDF417 by TEC-IT"$_tbce`	

	Notes:	2D symbology (multi-row) to encode large quantities of data. Data representation is divided into rows and columns that adjust automatically (depending on input data). Up to 900 characters per square inch possible. Standard-2D symbology in the automotive industry.	
56	**PDF417 Truncated** Valid characters: Quiet-zone: Module-width: Print-ratio: Ratio format string: Default check-digit: Size:	 Alphanumeric (ASCII 0.. 255) and/or bytes left/right: 2X -- 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 Automatic (symbology specific). --	

	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b56 dHello - this is a PDF417 by TEC-IT$_tbce`
		`$_tbcs -b56 –d"Hello - this is a PDF417 by TEC-IT"$_tbce`	

	Notes:	2D symbology (multi-row) to encode large quantities of data. Data representation is divided into rows and columns that adjust automatically (depending on input data).	
58	**QR-Code** Valid characters: Quiet-zone: Module-width: Print-ratio: Ratio format string: Default check-digit: Size:	 Alphanumeric and/or bytes, Kanji character set left/right/ top/bottom: 4X -- 1:1 1B:1S Automatic (symbology specific). --	

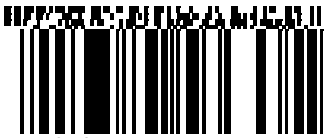
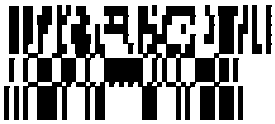
	Sample control sequence
	TBarCode/X v1.x:
	TBarCode/X v2.0:
		
		`$_tbcs b58 dHello - this is a QR Code by TEC-IT$_tbce`
		`$_tbcs -b58 -d"Hello - this is a QR-Code by TEC-IT"$_tbce`	

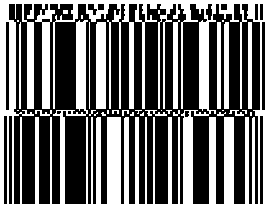
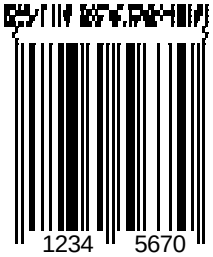
	Notes:	2D symbology to encode large quantities of data and developed for speedy scanning (QR = Quick Response Code); Size adjusts automatically depending on input data or can be set by printer commands. Special industry formats are supported.	

Table 43: Supported 2D Barcode Symbologies and Enumerators

18.4 EAN.UCC Composite Symbologies

- Please note: For all Composite Symbologies the vertical bar "|" character is used to separate the data of the linear symbol and the 2D composite component.
- Example: 1234567890123|TEC-IT

29	**RSS-14 Composite Symbology** Valid characters RSS-14: "0"..."9", 13 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied	 (01)12345678901231
	Encoded data: 1234567890123|TEC-IT	
	Notes: RSS-14 barcode with an attached 2D component (CC-A or CC-B). The leading Application Identifier "01" (GTIN) is prefixed automatically by the software and must not occur in the input data. The 2D component can encode additional information like lot number, quantity, expiration date …	
78	**RSS-14 Truncated Composite Symbology** Valid characters RSS-14: "0"..."9", 13 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied	 (01)12345678901231
	Encoded data: 1234567890123|TEC-IT	
	Notes: RSS-14 Truncated barcode with an attached 2D component (CC-A or CC-B).	
79	**RSS-14 Stacked Composite Symbology** Valid characters RSS-14: "0"..."9", 13 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied	
	Encoded data: 1234567890123|TEC-IT	
	Notes: RSS-14 Stacked barcode with an attached 2D component (CC-A or CC-B).	
80	**RSS-14 Stacked Omni directional Composite Symbology** Valid characters RSS-14: "0"..."9", 13 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied	
	Encoded data: 1234567890123|TEC-IT	
	Notes: RSS-14 Stacked Omni directional barcode with an attached 2D component (CC-A or CC-B).	
31	**RSS Expanded Composite Symbology** Valid characters RSS Exp.: ASCII characters between 0..127 Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: None (eCDNone). Possible check-digits: Modulo 10 (eCDMod10). EAN-14 (eCDEAN14)	 1234567890123
	Encoded data: 1234567890123|TEC-IT	
	Notes: RSS Expanded barcode with an attached 2D component (CC-A or CC-B).	

81	**RSS Expanded Stacked Composite Symbology** Valid characters RSS ES: ASCII characters between 0..127 Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: None (eCDNone). Possible check-digits: Modulo 10 (eCDMod10). EAN-14 (eCDEAN14)	
	Encoded data: ABCabc123+|TEC-IT	
	Notes: RSS Expanded Stacked barcode with an attached 2D component (CC-A or CC-B).	
30	**RSS Limited Composite Symbology** Valid characters RSS Lim.: "0".."9", 13 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:5:6:7:8:9:1:2:3:4:5:6:7:8:9 Ratio format string: 1B:2B:3B:4B:5B:6B:7B:8B:9B: 1S:2S:3S:4S:5S:6S:7S:8S:9S Default check-digit: EAN 14 (eCDEAN14) Possible check-digits: User supplied	 (01)12345678901231
	Encoded data: 1234567890123|TEC-IT	
	Notes: RSS Limited barcode with an attached 2D component (CC-A or CC-B).	
16	**UCC/EAN128 Composite Symbology** Valid characters EAN 128: ASCII-characters between 0..127 Valid characters CC-A/B/C: ISO 646 character set, up to 2361 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B1S:2S:3S:4S Default check-digit: None (eCDNone). Possible check-digits: Modulo 10 (eCDMod10). EAN-14 (eCDEAN14)	 1234567890
	Encoded data: 1234567890|TEC-IT	
	Notes: EAN128 barcode with an attached 2D component (CC-A, CC-B or CC-C).	
10	**EAN-8 Composite Symbology** Valid characters EAN 8: "0".."9", 7 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-8 (eCDEAN8) Possible check-digits: User supplied	 1234 5670
	Encoded data: 1234567|TEC-IT	
	Notes: EAN-8 barcode with an attached 2D component (CC-A or CC-B).	
13	**EAN-13 Composite Symbology** Valid characters EAN 13: "0".."9", 12 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: EAN-13 (eCDEAN13) Possible check-digits: User supplied	 1 234567 890128
	Encoded data: 123456789012|TEC-IT	
	Notes: EAN-13 barcode with an attached 2D component (CC-A or CC-B).	

34	**UPC-A Composite Symbology** Valid characters UPC-A: "0".."9", 11 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: UPC-A (eCDUPCA) Possible check-digits: User supplied	
	Encoded data:	12345678901|TEC-IT
	Notes:	UPC-A barcode with an attached 2D component (CC-A or CC-B).
37	**UPC-E Composite Symbology** Valid characters UPC-A: "0".."9", 7 digits + 1 check digit Valid characters CC-A/B: ISO 646 character set, up to 338 characters Standard print ratio: 1:2:3:4:1:2:3:4 Ratio format string: 1B:2B:3B:4B:1S:2S:3S:4S Default check-digit: UPC-E (eCDUPCE) Possible check-digits: User supplied	
	Encoded data:	1234567|TEC-IT
	Notes:	UPC-E barcode with an attached 2D component (CC-A or CC-B).

Table 44: Supported Composite Symbologies and Enumerators

19 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