Version 2.6

WARNING – SERIALIZED COPY

The software BCF® is copyrighted and its usage is conditioned upon the acceptance by the user of the licence contract, which clearly states when and how the software can be used. **Under no circumstances the software should be distributed or transmitted to a third party in violation of the licence contract. Since every copy of the product is serialized, SZP is always able to determine the exact origin of an unauthorized copy thus individuating the original licensee, who will be prosecuted to the full extent for the violation of copyright and of the licence contract.**

SUMMARY

Introduction

BCF® is a software package including a collection of TrueType™ barcode fonts and a Dynamic Link Library (DLL), which is used to translate text to barcode.
An additional utility, BCF Assistant®, is also provided, for quickly creating barcodes to be pasted into other applications.

The fonts included are standard True Type ™ for Windows and allow the creation of barcodes in the following symbologies (the actual number of symbologies supported depends on the type of licence bought):

- Code 39
- Code 39 Full Ascii
- Code 93
- Code 93 Full Ascii
- ITF-14
- Interleaved 2/5
- Industrial 2/5
- 5 Bars 2/5
- 3 Bars 2/5
- BCD Matrix 2/5
- Code 11 Matrix
- 2/5 Inverted
- 2/5 Compressed
- MSI
- Plessey
- Delta-Distance-A
- EAN-13 (with and w/o add-on of 2 and 5 digits)
- EAN-8 (with and w/o add-on of 2 and 5 digits)
- UPC-A (with and w/o add-on of 2 and 5 digits)
- UPC-E (with and w/o add-on of 2 and 5 digits)
- Code 32 (with and w/o human readable characters in OCR font)
- Code C.I.P.
- Codabar/Monarch
- Code 128
- EAN 128
- US Postnet
- ISBN
- ISBN 13

The bar codes can be used in any Windows package that has the feature of allowing selection of font and sizes for text. The bar codes created following the directions in this manual can be read with traditional instruments, such as optical pen, CCD readers and laser scanners.

The DLL in the package is necessary to produce readable codes because it is not possible to simply apply the font to the information being encoded. A bar code is always characterized by:

- start bars
- body
- check digit, if needed
- end bars

Starting and ending bars are special symbols that instruct the decoder about which symbology is used for the bar code; they do not contain information useful to the user and are normally mapped to the characters "*" (start) and "#" (end) in every font of the package.

The body of the barcode, in the simplest symbologies, is more or less the information being encoded; however, a few symbologies like Interleaved 2/5, the EAN/UPC family and Code 128 are exceptions, in that they optimize the information being encoded with special strategies.

The check digit, which is always needed for a few symbologies (such as Code 128 and EAN/UPC family), reccomended for others (Interleaved 2/5) or optional (e.g. Code 39), is calculated with a different algorithm for each symbology and is determined from the contents being encoded; it supplies the decoder with an additional information that can be used to determine if the barcode was interpreted correctly, thus increasing the degree of security.

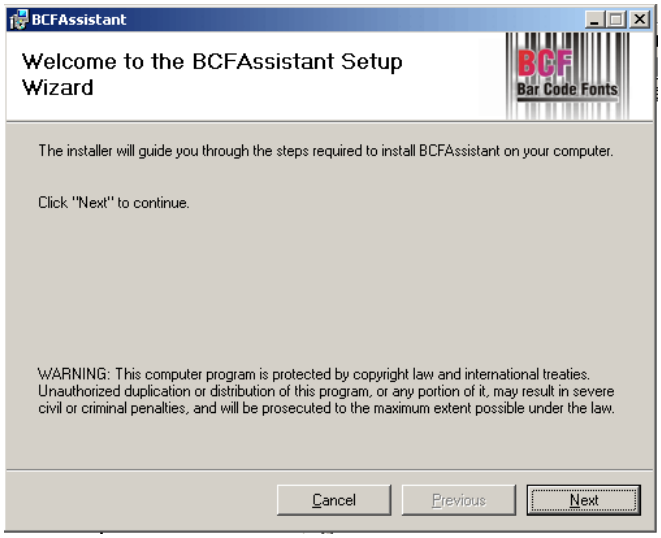
Every symbology is characterized by a few features, the most important of which are:

- alphabet – every symbology admits digits 0 to 9; a few symbologies let you use alphabetic characters and special characters, others include the full ASCII set;
- length – fixed or variable, limited only by the size of the resulting symbol;
- compactness – the same contents, expressed in different symbologies, create symbols of different lengths; moreover, a few symbologies, given the same length of text to be encoded, create bar codes of different size; this is because these symbologies optimize the information in different ways, depending on both contents and disposition (e.g. Code 128);
- security – a few symbologies should include a check digit, for being read with a high degree of security (e.g. Interleaved 2/5); more complex symbologies make use of bars of different thickness, thus it is necessary that both printing and reading devices can handle them correctly.

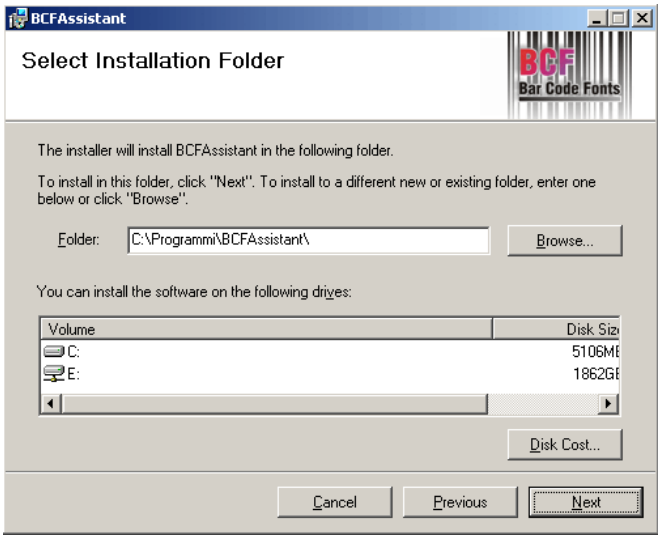
The field of application very often determines the symbology which should be used.

Installation

To install the package, insert your cd-rom into the reader of a Personal Computer equipped with Windows operating system. If the installation program does not start automatically, open the cd-drive using Explorer and double click on index.html; follow the link to install BCF Assistant®.



Press “Next”, read and agree the license terms by pressing “Next” (radio button on “I Agree”).



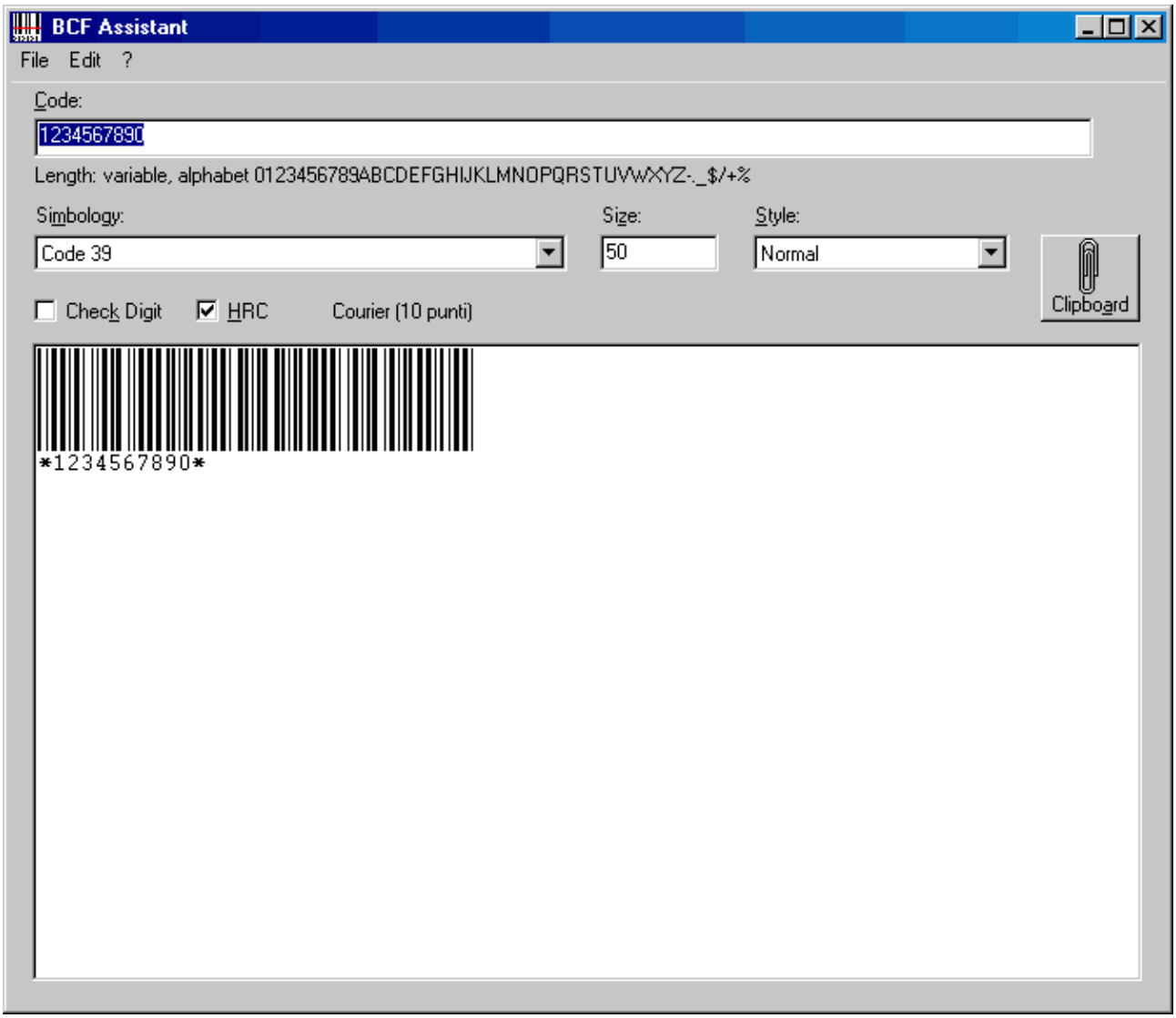
In the following form, press “Browse” button to modify the installation path; press Next to confirm and Next to install the application into the selected directory.
The installation program copies the required DLL to the system directory, together with BCF Assistant and this manual.
BCF TrueType fonts must be installed manually, by opening the applet "Fonts" in Windows Control Panel; select File, Install a new font, then browse to the folder TTF on the cd-rom, select all and confirm with Ok.
The 64-bit DLL (BCFDLL64.dll) must be copied manually to C:\Windows\Sysnative on 64-bit operating systems (if you need to invoke DLL functions from 64-bit applications).

Using BCF Assistant

BCF Assistant is an utility that lets you quickly and easily create bar codes in the supported symbologies, for pasting them inside other applications.

At the end of the installation, press the Windows Start button and select from the list of the programs “BCF Assistant”.

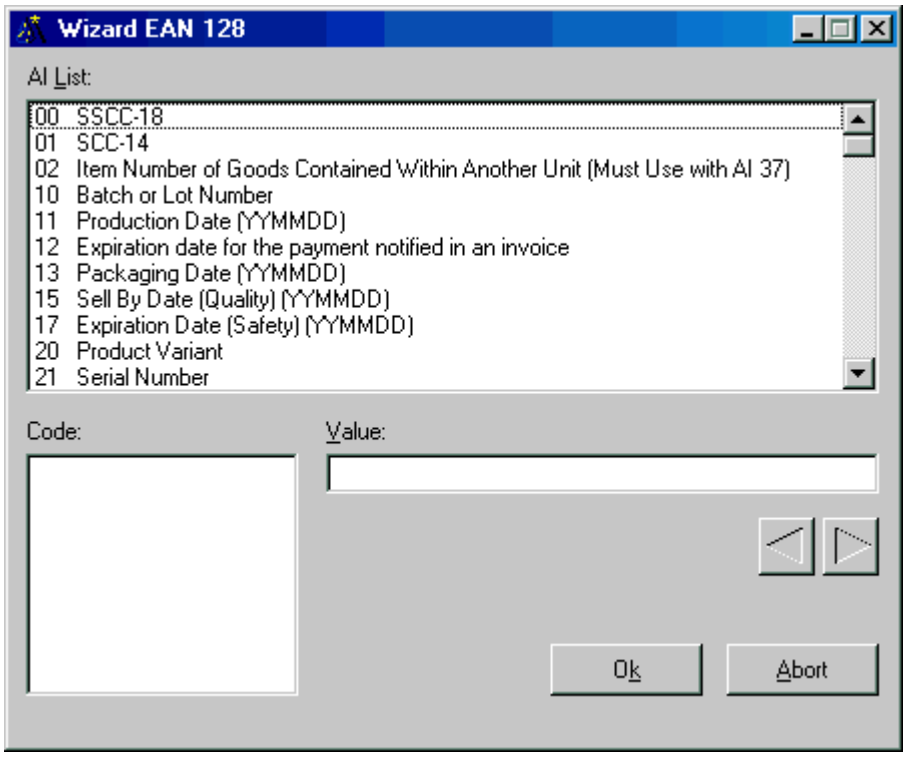
The following picture illustrates the interface of the program.



The software is very easy to use.

Just type in the field “Code” the information to be encoded; under the field there is a label that lists admitted characters and maximum length. When you press Enter, the corresponding bar code appears in the big area below, at least if the size of the bar code fits the window.
If the text cannot be encoded, a message will appear. For the symbology EAN 128, the picture of a small magic wand appears to the right of the input field; by clicking on this symbol a window (wizard) will open to help you build up the code. Choose from the list of AIs (Application Identifiers) the field to be included; type in the “Value” field the corresponding text, by following

the directions below for length and requested type. Press right arrow button to enter the field; use left arrow button to undo one field at a time. Press “Ok” to accept the newly composed EAN-128 barcode.



The field “Symbology” let you choose the type of barcode to create; a few symbologies limit the possible values for other settings (HRC, check digit).

The field “Size” let you choose the number of dots for the barcode character.

The “Style” field changes according to symbology; “normal” and “short” attributes refer to the height of the barcode (the short version is half the height of the normal version); “narrow” and “wide” refer to the horizontal size, with the narrow version being a little more compact and the wide version a little oversized.

The “Check Digit” checkbox enables or disables the check digit; for a few symbologies, this checkbox is unselectable. When the check digit is enabled, the resulting barcode is a little longer (if the symbology allows for variable length).

The checkbox “HRC” (Human Readable Code) enables or disables the printing of readable text under the barcode. The font used for this purpose can be selected with the menu option “Edit, HRC Font” and appears to the right of the checkbox. A few symbologies (like EAN/UPC family) always have HRC; others, like Code 32, have their own font and does not follow user settings.

Press the “Clipboard” button to copy to the Windows clipboard the contents of the preview.
In the applications that allow pasting, use “Edit, Paste” or press Ctrl + V to paste the contents of the clipboard in the document.

Font description

The following table illustrates all the fonts available in the package (the actual number depends on the type of licence):

File name	Font name	Description
szp39___.ttf	SZP39	Code 39 normal
szp39n__.ttf	SZP39 Narrow	Code 39 narrow
szp39w__.ttf	SZP39 Wide	Code 39 wide
szp39s__.ttf	SZP39 Short	Code 39 short
szp39ns_.ttf	SZP39 Narrow Short	Code 39 narrow and short
szp39ws_.ttf	SZP39 Wide Short	Code 39 wide and short
szp93___.ttf	SZP93	Code 93 normal
szp93n__.ttf	SZP93 Narrow	Code 93 narrow
szp93w__.ttf	SZP93 Wide	Code 93 wide
szp93s__.ttf	SZP93 Short	Code 93 short
szp93ns_.ttf	SZP93 Narrow Short	Code 93 narrow and short
szp93ws_.ttf	SZP93 Wide Short	Code 93 wide and short
szp25___.ttf	SZP25 ITF	Interleaved 2/5 normal
szp25s__.ttf	SZP25 ITF Short	Interleaved 2/5 short
szpind25.ttf	SZPIND25	Industrial 2/5 normal
szpin25s.ttf	SZPIND25 Short	Industrial 2/5 short
szp5bar_.ttf	SZP5BARS	5 bars 2/5 normal
szp5bars.ttf	SZP5BARS Short	5 bars 2/5 short
szp3bar_.ttf	SZP3BARS	3 bars 2/5 normal
szp3bars.ttf	SZP3BARS Short	3 bars 2/5 short
szpbcd__.ttf	SZPBCD	BCD Matrix 2/5 normal
szpbcds_.ttf	SZPBCD Short	BCD Matrix 2/5 short
szpmsi__.ttf	SZPMSI	MSI normal
szpmsis_.ttf	SZPMSI Short	MSI short
szp32___.ttf	SZP32	Code 32 (including HRC)
szpcbar_.ttf	SZPCODABAR	Codabar/Monarch normal
szpcbars.ttf	SZPCODABAR Short	Codabar/Monarch short
szpean__.ttf	SZPEAN	EAN/UPC/ADD-ON/ISBN normal
szpeans_.ttf	SZPEAN Short	EAN/UPC/ADD-ON/ISBN short
szp128__.ttf	SZP128	Code 128/EAN 128 normal
szp128s_.ttf	SZP128 Short	Code 128/EAN 128 short
szp11___.ttf	SZP11	Code 11 Matrix normal
szp11s__.ttf	SZP11 Short	Code 11 Matrix short
szp25com.ttf	SZP25 COMP	Code 2/5 Compressed normal
szp25cos.ttf	SZP25 COMP Short	Code 2/5 Compressed short
szp25inv.ttf	SZP25 INV	Code 2/5 Inverted normal
szp25ins.ttf	SZP25 INV Short	Code 2/5 Inverted short
szpdelta.ttf	SZPDELTA DA	Code Delta-Distance-A normal
szpdelts.ttf	SZPDELTA DA Short	Code Delta-Distance-A short
szpitf14.ttf	SZPITF14	Code ITF-14 normal
szpitfs_.ttf	SZPITF14 Short	Code ITF-14 short
szpplesy.ttf	SZPPLESSEY	Code Plessey normal

szppless.ttf	SZPPLESSEY Short	Code Plessey short
szppost.ttf	SZPPOSTNET	Code US Postnet

Features for different symbologies

In the following pages we will briefly discuss the symbologies supported by the package.

Code 39

It is one of the most widespread symbologies, thanks to its versatility and the security of the encoding, even though information density is not high (resulting bar codes are large).
The most interesting features are:
- variable length;
- alphabet: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ-._$/+% (space must be replaced by underscore, '_');
- Code 39 is usually represented **without** check digit, which is optional.

The DLL converts automatically to uppercase those characters that are lowercase before applying the encoding.

Code 93

It is a very versatile symbology; it was developed as a replacement for Code 39; it has the same alphabet, but is more compact.
Le caratteristiche salienti sono:
The most interesting features are:
- variable length;
- alphabet: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ-._$/+% (space must be replaced by underscore, '_');
- Code 93 is usually represented **with** check digit, which consists of two characters from the alphabet.

The DLL converts automatically to uppercase those characters that are lowercase before applying the encoding.

UCC-ITF 14

It is a symbology very widespread and compact; it is numeric only, with a fixed length of 14 digits. It is immediately recognizable because the barcode appears inside a thick black frame. It is very used for tracking goods in transit.
The most interesting features are:
- 13 (if check digit is used) or 14 digits; if the text to be encoded is not long enough, the DLL pads "0"s to the left;
- only digits 0 to 9 are allowed;
- check digit is always **needed**.

Interleaved 2 of 5 (0 padded)

It is a numeric-only symbology very widespread and compact. Since digits are encoded in pairs, they should be even in number if check digit is not used, should be odd otherwise.
The most interesting features are:

- variable length;
- only digits 0 to 9 are allowed;
- the check digit is **reccomended**, because the decoder can decode partial codes otherwise.

The DLL adds a "0" to the left, if necessary, to obtain even length for the text to be encoded.

Industrial 2 of 5

It is a symbology that uses the same encoding as Interleaved 2/5, but exploits only bars (not spaces). In this way the security of the decoding is greater, even though the resulting barcode is longer.
The most interesting features are:

- variable length;
- only digits 0 to 9 are allowed;
- check digit is optional.

5 bars 2 of 5

It is a numeric-only, low-density symbology, mostly used in photographic laboratories.
The most interesting features are:

- variable length;
- only digits 0 to 9 are allowed;
- check digit is optional.

3 bars Matrix 2 of 5

It is a numeric-only, medium-density symbology.
The most interesting features are:

- variable length;
- only digits 0 to 9 are allowed;
- check digit is optional.

Code 11 Matrix

It is a symbology with an 11-symbol alphabet and medium density of information.
The most interesting features are:

- variable length;
- only digits 0 to 9 are allowed and the special character "-";
- check digit is **optional**; if it is used, one check digit is added to codes of up to 10 characters, for longer codes the check digits become two.

BCD Matrix 2 of 5

It is a numeric-only, medium-density symbology.
The most interesting features are:

- variable length;
- only digits 0 to 9 are allowed;
- check digit is optional.

Code 2 of 5 Inverted

It is a numeric-only, low-density symbology; it was used for the possibility of being printed by chain printers; today is no longer used.
The most interesting features are:
- variable length;
- only digits 0 to 9 are allowed;
- check digit is optional.

Code 2 of 5 Compressed

It is a numeric-only, high-density symbology, based upon 2/5 3 bars Matrix.
The most interesting features are:
- variable length;
- only digits 0 to 9 are allowed;
- check digit is optional.

MSI

It is a numeric-only, low-density symbology.
The most interesting features are:
- variable length, up to 15 digits;
- only digits 0 to 9 are allowed;
- (double) check digit is **reccomended**.

Plessey

It is an alphanumeric, low-density symbology.
The most interesting features are:
- variable length, up to 15 characters;
- only digits 0 to 9 and the characters A, B, C, D, E, F are allowed;
- (double) check digit is **always needed**.

Delta-Distance-A

It is an alphanumeric, low-density symbology.
The most interesting features are:
- variable length, up to 15 characters (using check digit) or 16 characters (without check digit);
- only digits 0 to 9 and the characters K, L, M, O are allowed;
- check digit is **reccomended**.

EAN13

It is a numeric-only, high-density symbology, which is mainly used for labeling goods to be sold inside Europe.
The most interesting features are:
- fixed length: 12 digits (with check digit) or 13 digits (without check digit); if the text to be encoded is shorter, it is left-padded with “0”s by the DLL;
- only digits 0 to 9 are allowed;
- check digit is always **needed**.

ISBN

It is a numeric-only, high-density symbology, which is used for labeling books. An ISBN symbol is made of two lines; the first is the text “ISBN” followed by the code (including check digit), which is hyphenated according to specific rules; the second line is an EAN13 barcode.
The most interesting features are:

- fixed length: 9 digits; if the text to be encoded is shorter, it is left-padded with “0”s by the DLL;
- only digits 0 to 9 are allowed;
- check digit is always **needed** and is automatically added; the check digit can take the value “X”, besides ‘0’-‘9’;
- the DLL function returns two lines, corresponding to the text and the barcode; apply the EAN font to both lines to obtain a correct symbol.

ISBN 13

It is a numeric-only, high-density symbology, which is used for labeling books. An ISBN symbol is made of two lines; the first is the text “ISBN” followed by the code (including check digit), which is hyphenated according to specific rules; the second line is an EAN13 barcode.
The most interesting features are:

- fixed length: 12 digits; if the text to be encoded is shorter, it is left-padded with “0”s by the DLL;
- only digits 0 to 9 are allowed;
- check digit is always **needed** and is automatically added;
- the DLL function returns two lines, corresponding to the text and the barcode; apply the EAN font to both lines to obtain a correct symbol.

EAN 8

It is a numeric-only, high-density symbology, which is mainly used for labeling goods to be sold inside Europe; it is used in place of EAN 13 when the size of the object being labeled is very small.
The most interesting features are:

- fixed length: 7 digits (with check digit) or 8 digits (without check digit); if the text to be encoded is shorter, it is left-padded with “0”s by the DLL;
- only digits 0 to 9 are allowed;
- check digit is always **needed**.

UPC A

It is a numeric-only, high-density symbology, which is mainly used for labeling goods to be sold inside USA.
The most interesting features are:

- fixed length: 11 digits (with check digit) or 12 digits (without check digit); if the text to be encoded is shorter, it is left-padded with “0”s by the DLL;
- only digits 0 to 9 are allowed;
- check digit is always **needed**.

UPC E

It is a numeric-only, high-density symbology, which is mainly used for labeling goods to be sold inside USA; it is used in place of UPC A when the size of the object being labeled is very small.

The most interesting features are:
- fixed length: 7 digits (with check digit) or 8 digits (without check digit); if the text to be encoded is shorter, it is left-padded with "0"s by the DLL;
- only digits 0 to 9 are allowed;
- the text to be encoded **must** start with 0 or 1;
- check digit is always **needed**.

2 digit add on for EAN/UPC bar codes

EAN/UPC symbologies can be expanded by using a 2-digit add on positioned 5 modules to the right of the ending bars; the 2-digit add on is often used for labeling monthly magazines.
The most interesting features are:
- fixed length: 2 digits;
- only digits 0 to 9 are allowed;
- check digit is always included, regardless of user request;
- the string returned by the DLL starts with a blank character that is exactly 5 modules large; just concatenate this string to the one calculated for an EAN/UPC barcode and you will get a correctly formatted EAN/UPC barcode with add-on.

5 digit add on for EAN/UPC bar codes

EAN/UPC symbologies can be expanded by using a 5-digit add on positioned 5 modules to the right of the ending bars; the 5-digit add on is often used for labeling daily newspapers.
The most interesting features are:
- fixed length: 5 digits;
- only digits 0 to 9 are allowed;
- check digit is always included, regardless of user request;
- the string returned by the DLL starts with a blank character that is exactly 5 modules large; just concatenate this string to the one calculated for an EAN/UPC barcode and you will get a correctly formatted EAN/UPC barcode with add-on.

Code 32 (italian pharmaceutical)

It is a symbology developed for labeling pharmaceutical products sold in Italy, as requested by the law (D.L. 10 June 1983). It is based on Code 39.
The most interesting features are:
- fixed length: 8 digits (if check digit is used) or 9 digits (without check digit);
- only digits 0 to 9 are allowed;
- check digit is always **needed**;
- if HRC is requested, it is added below the barcode, prefixed by the "A" capital letter.

Code C.I.P. (french pharmaceutical)

It is a symbology used in France by the "Club Inter-Pharmaceutique".
It is based on Code 39.
The most interesting features are:
- fixed length: 6 digits (if check digit is used) or 7 digits (without check digit);
- only digits 0 to 9 are allowed;
- check digit is always **needed**;
- because of the special algorithm for calculating check digit (modulo 11), **not every sequence of 6 digits is encodable** – all the sequences that have a 10-checksum must be discarded.

Codabar/Monarch

The alphabet of this symbology includes the 10 digits, 6 special characters and 4 start/stop sequences. It is very common, especially in medical environments.
The most interesting features are:

- variable length, greater than or equal to 3;
- the text encoded must start and end with a character included in "ABCDEMT*";
- the remaining characters must belong to the following alphabet: "0123456789-$:/.+";
- the check digit is **optional**.

Code 128

This symbology derives its name from the fact that it can encode all 128 characters of the ASCII set; it is characterized by high density of information and high reliability. Information density is increased by encoding digits in pairs whenever it is possible.
The most interesting features are:

- variable length;
- every character belonging to the ASCII set (0-127) can be encoded;
- these additional characters can be encoded:
  128 = NUL
  129 = FNC1
  130 = FNC2
  131 = FNC3
  132 = FNC4
- to represent non-printable characters, the following C-like constants can be used:
  \a or \A = 7
  \b or \B = 8
  \f or \F = 12
  \n or \N = 10
  \r or \R = 13
  \t or \T = 9
  \v or \V = 11
  \\ = 92 = "\"
  \? = 93 = "?"
  \' = 39 = "'"
  \" = 34 = double quotes
  \dnnn = chr(nnn), where nnn is a decimal number between 000 and 255
  \ooo = character corresponding to the number ooo octal (digits 0..7), from 000 to 377
  \xhh = character corresponding to the number hh hexadecimal (0..9, A..F), from 00 to FF
  Thus, if the text being encoded includes the character “\”, it will need to be doubled for avoiding being misinterpreted as a C-like constant;
- the check digit is **always needed**, and is added regardless what the user specifies.

EAN 128

This symbology is based upon Code 128; it is used for encoding information divided into fields identified by a number (AI = Application Identifier); thus, it is possibile to extract automatically only the information needed, knowing exactly where that information is in the code.
AI calculation is left to the user; we only remember here that if the field following the AI code:

- is of **variable** length;
- is **shorter** than the maximum length allowed;

- is **not** at the end of the code,

it must be separated from the next AI by the character CHR(129) = FNC1 = "\201" = "\x81".
The check digit is **compulsory** and is always added automatically, regardless what the user requests.

Code 39 "Full Ascii" and Code 93 "Full Ascii"

These symbologies are based on Code 39 and Code 93, respectively. By using character combinations it is possible to encode the entire ASCII set.
The most interesting features are:

- variable length;
- every character belonging to the ASCII set (0-127) can be encoded;
- to represent non-printable characters, the following C-like constants can be used:
  \a or \A = 7
  \b or \B = 8
  \f or \F = 12
  \n or \N = 10
  \r or \R = 13
  \t or \T = 9
  \v or \V = 11
  \\ = 92 = "\"
  \? = 93 = "?"
  \' = 39 = "'"
  \" = 34 = double quotes
  \dnnn = chr(nnn), where nnn is a decimal number between 000 and 255
  \ooo = character corresponding to the number ooo octal (digits 0..7), from 000 to 377
  \xhh = character corresponding to the number hh hexadecimal (0..9, A..F), from 00 to FF
  Thus, if the text being encoded includes the character "\", it will need to be doubled for avoiding being misinterpreted as a C-like constant;

US Postnet

It is a symbology used in the United States for marking snail mail in transit; it differs from other symbologies in that it uses bars of two different heights, instead of two different widths.
The most interesting features are:

- fixed length: 5, 9 or 11 digits;
- only digits 0 to 9 are allowed;
- the check digit is **always needed**.

Using BCF with Visual Basic

BCF makes available a dynamic link library (BCFDLL.DLL) which is automatically copied to the system directory during installation. This library is needed to calculate the encoding of the text, before applying the font for obtaining a readable bar code.
BCFDLL exports two functions:

- SZPStringToBarcodeDLL, to calculate the encoding;
- SZPValidateStringDLL, to verify if the specified text can be encoded with the specified symbology.

To use these functions, they must be declared in a Visual Basic or VBA module (see also the folder `Examples\VisualBasic` on CD), in this way:

```
Declare Function SZPStringToBarcodeDLL Lib "BCFDLL" (ByVal code As String, _
ByVal symbology As String, ByVal IncludeCD As Integer, _
ByVal barcode As String) As Integer

Declare Function SZPValidateStringDLL Lib "BCFDLL" (ByVal code As String, _
ByVal symbology As String, ByVal IncludeCD As Integer) As Boolean
```

The function `SZPStringToBarcodeDLL` writes to the `barcode` buffer the encoding of `code` according to `symbology` and to the check digit flag (`IncludeCD`), returning the number of characters copied to `barcode`.

The function `SZPValidateStringDLL` verifies if it is possible to encode `code` according to `symbology` and to the check digit flag (`IncludeCD`), returning True if it is possible and False if it is impossible.

This is the meaning of the parameters:

`code`
represents text to be encoded; be sure to respect symbology conventions

`symbology`
alphanumeric string that specifies which symbology is to be used; the values admitted are:

C39	code 39
C39FULL	code 39 Full Ascii
C93	code 93
C93FULL	code 93 Full Ascii
ITF	UCC-ITF 14
C25	Interleaved 2/5
CIND25	Industrial 2/5
C5BARS	5 Bars 2/5
C3BARS	3 Bars 2/5
CBCD	BCD Matrix 2/5
C11	Code 11 Matrix
C25INV	Code 2/5 Invertito
C25COMP	Code 2/5 Compresso
CMSI	MSI
PLESSEY	Plessey

DELTA	Delta-Distance-A
EAN13	EAN 13
EAN8	EAN 8
UPCA	UPC A
UPCE	UPC E
ADD2	2-digit add-on for EAN/UPC
ADD5	5-digit add-on for EAN/UPC
C32	code 32 without HRC
C32HRC	code 32 with HRC
CIP	code C.I.P. (french pharmaceutical)
CBAR	codabar/monarch
C128	code 128
E128	EAN 128
POSTNET	US Postnet
ISBN	ISBN
ISBN13	ISBN with 13 digits

`IncludeCD`
specifies if the check digit for the symbology should be added automatically; 1 = add check digit, 0 = don't add

`Barcode` (only for `SZPStringToBarcodeDLL`)
string buffer to store encoding; the length of the buffer should be a minimum of 64 bytes, or a minimum of ***4 * n + 10*** bytes, where ***n*** is the length of the text being encoded. Clearly, to obtain a correctly formatted bar code, the corresponding True Type font will have to be applied to the encoded text, according to the table under "Font description".

Examples:

```
' This function returns the encoding with a single call

Function BCFCode (s As String, symb As String, with_cd As Boolean) As String

Dim result As String * 255, length As Integer

length = SZPStringToBarcodeDLL(s, symb, IIf(with_cd, 1, 0), result)
BCFCode = Left(result, length)

End Function
```

```
' Example of direct call for encoding the specified text to EAN13 with check digit;
' it is important to note that only 12 digits are specified, because the check
' digit is added automatically (IncludeCD = 1)

Dim result As String * 255, length As Integer
[..]
length = SZPStringToBarcodeDLL("123456789012", "EAN13", 1, result)

' Left(result, length) will contain the encoding
```

```
' This function verifies it it is possibile to encode s according to symbology symb
' and the with_cd flag

Function VerifyBCF(s As String, symb As String, with_cd As Boolean) As Boolean

VerifyBCF = SZPValidateStringDLL(s, symb, IIf(with_cd, 1, 0))

End Function
```

Using BCF with Visual Basic .Net

The dynamic link library (BCFDLL.DLL or BCFDLL64.DLL) can be used in Visual Basic .Net much like in Visual Basic.

To use these functions, they must be declared in this way (see also the folder `Examples\VisualBasicNet` on CD):

```
Imports System.Text
Imports System.Runtime.InteropServices

Module Module1

    ' 32-bit DLL
    Public Declare Function SZPStringToBarcodeDLL Lib "BCFDLL" _
    (ByVal code As String, ByVal simbology As String, _
     ByVal WithCD As Integer, <MarshalAs(UnmanagedType.LPStr)> _
     ByVal barcode As StringBuilder) As Integer

    Public Declare Function SZPValidateStringDLL Lib "BCFDLL" _
    (ByVal code As String, ByVal simbology As String, _
     ByVal WithCD As Integer) As Boolean

    ' 64-bit DLL
    Public Declare Function SZPStringToBarcodeDLL64 Lib "BCFDLL64" Alias _
    "SZPStringToBarcodeDLL" (ByVal codice As String, _
    ByVal simbologia As String, ByVal ConCD As Integer, _
    <MarshalAs(UnmanagedType.LPStr)> ByVal barcode As StringBuilder) As Integer

    Public Declare Function SZPValidateStringDLL64 Lib "BCFDLL64" Alias _
    "SZPValidateStringDLL" (ByVal codice As String, _
    ByVal simbologia As String, ByVal ConCD As Integer) As Boolean

[..]
End Module
```

Examples:

```
' This function returns the encoding with a single call

Function BCFCode(s As String, symb As String, with_cd As Boolean) As String

Dim result As StringBuilder = New StringBuilder(255)
Dim length As Integer, i As Integer

If with_cd Then
  i = 1
Else
  i = 0
End If
' 32-bit DLL
length = SZPStringToBarcodeDLL(s, symb, i, result)
' 64-bit DLL
' length = SZPStringToBarcodeDLL64(s, symb, i, result)
BCFCode = result.ToString(0, length)

End Function

' This function verifies it it is possibile to encode s according to symbology symb
' and the with_cd flag
```

```
Function VerifyBCF(s As String, symb As String, with_cd As Boolean) As Boolean

If with_cd Then
  i = 1
Else
  i = 0
End If
' 32-bit DLL
VerifyBCF = SZPValidateStringDLL(s, symb, i)
' 64-bit DLL
' VerifyBCF = SZPValidateStringDLL64(s, symb, i)

End Function
```

Using BCF with Visual C++

Most of the information detailed in the previous section apply; here we will only explain the differences. The prototypes for the functions in the DLL are the following (see also `Examples\VisualCPP` on the CD):

```
int APIENTRY SZPStringToBarcodeDLL(char *code, char *symbology, int with_cd,
                                   char *barcode);

int APIENTRY SZPValidateStringDLL(char *newcod, char *symbology, int with_cd);
```

To invoke dynamically the functions, you only need to load the library, assign the address of the function in the library to a function pointer and then use it, as showed in this example:

```
#include <windows.h>
#include <winbase.h>

typedef int (APIENTRY *BCF1)(char *, char *, int);
typedef int (APIENTRY *BCF2)(char *, char *, int, char *);

int main(int argc, char* argv[])
{
  HINSTANCE hLibrary;
  BCF1 lpFunc1;
  BCF2 lpFunc2;

  hLibrary = LoadLibrary("BCFDLL.DLL");
  if (hLibrary != NULL)
  {
    char result[255];
    int length;

    lpFunc1 = (BCF1) GetProcAddress(hLibrary, "SZPValidateStringDLL");
    lpFunc2 = (BCF2) GetProcAddress(hLibrary, "SZPStringToBarcodeDLL");

    if (lpFunc1 != NULL)
      if (lpFunc1("123456789012", "EAN13", 1))
      {
        // Encoding is possible
        if (lpFunc2 != NULL)
        {
          length = lpFunc2("123456789012", "EAN13", 1, result);
          result[length] = 0;
          printf("%s\r\n", result);
        }
      }
    FreeLibrary(hLibrary);
  }
  return 0;
}
```

Using BCF with Visual Objects

Most of the information detailed in the previous section apply; here we will only explain the differences. The external functions must be declared in this way (see also `Examples\VisualObjects` on the CD):

```
_DLL FUNCTION SZPStringToBarcodeDLL(code AS PSZ, symbology AS PSZ, CheckDigit AS
DWORD, barcode AS PSZ) AS DWORD PASCAL:BCFDLL.SZPStringToBarcodeDLL

_DLL FUNCTION SZPValidateStringDLL(code AS PSZ, symbology AS PSZ, CheckDigit AS
DWORD) AS DWORD PASCAL:BCFDLL.SZPValidateStringDLL
```

This is how they are supposed to be used:

```
LOCAL pszBuffer AS PSZ
LOCAL dwLen AS DWORD
LOCAL sValue AS STRING
LOCAL lSuccess AS LOGIC

IF SZPValidateStringDLL(PSZ("12345"), PSZ("C128"), 1)
  pszBuffer := PSZ(MemAlloc(255))
  dwLen := SZPStringToBarcodeDLL(PSZ("12345"), PSZ("C128"), 1, pszBuffer)
  sValue := Left(PSZ2String(pszBuffer), dwLen)
ENDIF
```

Using BCF with Crystal Reports

Crystal Reports® is a widespread report generation tool, which is used both as a stand-alone application and as a designer inside a development environment.

The easier way to transform data into a barcode is using the library `CRUFLbcf.dll`, a DLL/Active-X which is installed with BCFAssistant or is available in the `Examples\Crystal` directory on the CD; this DLL can also be manually installed by copying it to:

`C:\WINDOWS\SYSTEM` (Windows 95/98/ME)
`C:\WINNT\SYSTEM32` (Windows NT/2000)
`C:\WINDOWS\SYSTEM32` (Windows XP/2003)

The DLL needs to be registered when the file is manually copied; open a Dos prompt and type the following command (change the path as needed):

```
REGSVR32 C:\WINNT\SYSTEM32\CRUFLbcf.dll
```

To obtain a readable barcode follow these steps:
a) insert the field to be encoded in the report;
b) select “Omit” in the properties of the field;
c) create a new formula, pasting the following code:

```
Formula = IIF(bcfBarcodeVerify ({Articles.PN}, "C128", True),
              bcfBarcodeEncode({Articles.PN}, "C128", True), "")
```

or (if you know for sure that the information can be transformed into a barcode):

```
Formula = bcfBarcodeEncode({Articles.PN}, "C128", True)
```

Note: In our example the data to be rendered as a barcode is inside the field PN from the table Articles; the symbology to be used is Code 128, with check digit.

The formula can be used as a new field in the report; apply the BCF font to this field to obtain the corresponding barcode (change font size to increase/decrease barcode size).

Note: `CRUFLbcf.dll` uses `BCFDLL.DLL` for all the encoding operations; this DLL needs to be installed as well.

Using BCF with Crystal Reports® .Net

The easiest way to transform data into a barcode is using the library `CRUFL_VB_BCF.dll` (available in the `Examples\CrystalNet` directory on the CD), which must by registered by opening the Visual Studio 2005 command prompt and typing the following command (change the path as needed):

```
gacutil -if [path]CRUFL_VB_BCF.dll
```

To obtain a readable barcode follow these steps:
d) insert the field to be encoded in the report;
e) select “Omit” in the properties of the field;
f) create a new formula, pasting the following code:

```
Formula = IIF(VBBCFBCFVerify({Articles.PN}, "C128", True),
              VBBCFBCFEncode({Articles.PN}, "C128", True), "")
```

or (if you know for sure that the information can be transformed into a barcode):

```
Formula = VBBCFBCFEncode({Articles.PN}, "C128", True)
```

Note: In our example the data to be rendered as a barcode is inside the field PN from the table Articles; the symbology to be used is Code 128, with check digit.

The formula can be used as a new field in the report; apply the BCF font to this field to obtain the corresponding barcode (change font size to increase/decrease barcode size).

Note: `CRUFL_VB_BCF.dll` uses `BCFDLL.DLL` for all the encoding operations; this DLL needs to be installed as well.

Using BCF with other languages

If you need to use BCF with other languages the easiest way is using the Active-X `BCFActiveX.ocx`; this control is automatically installed with BCFAssistant, and is also available from the `Examples\ActiveX` directory on the CD; the control can be manually installed by copying its file to:

`C:\WINDOWS\SYSTEM` (Windows 95/98/ME)
`C:\WINNT\SYSTEM32` (Windows NT/2000)
`C:\WINDOWS\SYSTEM32` (Windows XP/2003)

The Active-X needs to be registered when the file is manually copied; open a Dos prompt and type the following command (change the path as needed):

```
REGSVR32 C:\WINNT\SYSTEM32\BCFActiveX.ocx
```

The Active-X can be referenced in your projects just like any other control and makes it available the following methods:

```
Encode(code As String, symbology As String, checkdigit As Boolean) As String
Verify(code As String, symbology As String, checkdigit As Boolean) As Boolean
```

To render “ABCDEF” into `barcode` by using Code128 with check digit, if `BCFActiveX1` is the name of the control in your project, use the following syntax:

```
barcode = BCFActiveX1.Encode("ABCDEF", "C128", True)
```

Note: `BCFActiveX.ocx` uses `BCFDLL.DLL` for all the encoding operations; this DLL needs to be installed as well.

Using BCF with Microsoft Word®

The following procedure explains how to use BCF with Microsoft Word® to create a merged document including barcodes.

The method detailed here consists of two steps:
- merge to a new document;
- run a macro to replace specific text with its corresponding barcode representation.

Proceed in this way:

1) **install a module (used for barcode translations):**
- if VBA (Visual Basic for Applications) support is not available, run Word setup and add this option;
- select *Tools*, then *Macro→ Visual Basic Editor* from Word menu;
- under Visual Basic Editor, check that the **Project** window is visible; if it is hidden, select *Project Explorer* from the *View* menu;
- highlight the item **Normal** in the Project window; click right mouse button and select *Import file*; when requested, select `BCFReplace.bas` (available in the `Examples\Word` folder on the BCF cd-rom);
- if you would later need to change the size of the resulting barcodes, modify the following line as required:

  ```
  Const SZPFontSize = 36
  ```

  (you can find it at the very beginning of the module BCFReplace, which can be opened with a double click in Project Explorer);
- save and close the editor; the macro will be available in all Word documents, since you imported the module `BCFReplace` in the **Normal** template;

2) **create the master document**: just take care to put every string which needs to be translated to a barcode between "[SBC]" and "[EBC]"; for example, if the field to be translated is `<<Code>>`, write it as:

   [SBC]<<Code>>[EBC]

3) **to obtain the final printout with the barcodes (the steps here might be slightly different depending on Word version)**:
- choose *Tools* → *Merge* from Word menu;
- click on **Merge**; select "*Merge to a new docment"* and click on **Merge** again; you will get a new document including all the data merged;
- select *Tools* → *Macro* and run the macro **BCFRender** from the **Normal** template; you will get yet another document with the barcodes required, which you can print.

Obviously steps 1-2 are needed only the first time.

Note: the module uses `BCFDLL.DLL` for all the encoding operations; this DLL needs to be installed.

Using BCF with Microsoft Access®

The following procedure explains how to use BCF with Microsoft Access® to create a query which includes a calculated field with the intermediate representation needed by BCF fonts.

Proceed in this way:

1) open the MDB file, click on the **Modules** tab and press the **New** button;

2) paste the following lines in a new module (you can find this source code in the database `demo.mdb`, which is located in the folder `Examples\Access` on the BCF cd-rom):

```
Option Compare Database
Option Explicit

Declare Function SZPStringToBarcodeDLL Lib "BCFDLL" (ByVal codice As String, _
ByVal simbologia As String, ByVal ConCD As Integer, _
ByVal barcode As String) As Integer

Function BCFEncode(s As String, symb As String, checkdigit As Boolean) As String

Dim result As String * 255, length As Integer

length = SZPStringToBarcodeDLL(s, symb, IIf(checkdigit, 1, 0), result)
BCFEncode = Mid(result, 1, length)

End Function
```

3) save and (optionally) rename the module to `BCF`;

4) click on the **Query** tab and press the button **New**; confirm *Design View* and press **Ok**;

5) add the fields needed by the query; find an empty column and, in the *Field* row, type the following expression:

   ```
   Barcode: BCFEncode([Code];"C128";True)
   ```

   This is the explanation for the formula:

   - `Barcode`: expression name, used as column heading in the query;
   - `[Code]`: “Code” is the name of the field to be transformed into a barcode; change as needed;
   - `"C128"`: symbology to be used; change as needed;
   - `True/False`: include check digit.

Now you can create reports including the calculated field; just apply the appropriate BCF font to obtain a readable barcode.

Note: the module uses `BCFDLL.DLL` for all the encoding operations; this DLL needs to be installed.

Using BCF with Microsoft Excel®

The following procedure explains how to use BCF with Microsoft Excel® to add barcode calculation formulas.

Proceed in this way:

1) if VBA (Visual Basic for Applications) support is not available, run Excel setup and add this option;

2) copy the file `bcf.xla` (available in the folder `Examples\Excel` on the BCF cd-rom) to the `XLStart` system folder (Excel add-ins); from Excel, click on *Tools* → *Add-ins*, click **Browse** and select `bcf.xla`; in this way the functions will be available in any new Excel worksheet; alternatively (and if the protection level blocks the automatic loading of the add-in), the functions can be imported into the worksheet by following these steps:

   - in Excel click on *Tools*, then *Macro*→ *Visual Basic Editor*;
   - under Visual Basic Editor, check that the **Project** window is visible; if it is hidden, select *Project Explorer* from the *View* menu;
   - highlight the current project in the Project window; click right mouse button and select *Import file*; when requested, select `BCF.bas` (available in the `Examples\Excel` folder on the BCF cd-rom);
   - save and close the editor;

3) if the data to be rendered as a barcode is in column A, select the first cell of an available column and type the following text in the formula field:

   ```
   =BCFEncode(A1;"C128";"True")
   ```

   This is the explanation for the formula:
   - `A1`: cell containing the value to be rendered as a barcode; change as needed;
   - `"C128"`: symbology to be used; change as needed;
   - `True/False`: include check digit.

   At this point, the formula can be applied to the whole column by clicking on the cell you selected at the beginning of step 3 and extending the selection by dragging the small black square in the lower right corner.
   To obtain the requested barcode, apply the appropriate BCF font to the cell format.

Note: the module uses `BCFDLL.DLL` for all the encoding operations; this DLL needs to be installed.

Software Licence Agreement

Carefully read the following Agreement before installing the software on your PC.

By installing the Software accompanying this document you acknowledge that you have read, understood and agree to abide by the terms and conditions of this Software Licence Agreement.

SOFTWARE LICENCE

This is a legal agreement (Agreement) between you (either an individual or an entity) and Simone Zanella Productions (SZP) that sets forth the licence terms and conditions for using the enclosed Software (Software). Updates of the Software shall also be subject to the terms and conditions of this Agreement. This Agreement is effective until terminated by destroying the Software and all of the diskettes and documentation provided in this package, together with all copies, tangible or intangible. In this Agreement, the term "use" means loading the Software into RAM, as well as installing it onto a hard disk or other storage device.
The Software is owned by SZP and is protected under Italian copyright laws as well as international treaty provisions. You must treat the software as you would any other copyrighted material.
The type of licence, specified on the product, determines the following restrictions.

SITE licence:

The purchase price for the Software grants you a non-exclusive licence to use the Software with the following restrictions: this licence allows a single company to install on an unlimited number of machines within a 100 km radius; all machines must be owned, leased, or used for the purpose of the purchasing company; an exception is the fonts can be installed on employees, or contractors personal machines if done so in order to perform work for the purchasing company (please inform all contractors that the product is a licensed product and not freeware). The end user can be the purchasing company or one of its customers (but not both). The purchasing company must notify to SZP (by using the appropriate form) end user details and places of installation. SZP retains title and ownership of the Software.
You may make one copy of the software solely for archival purposes.
You may not rent, sell, lease, sub-licence, time-share or lend the Software to a third party or otherwise transfer this Licence without written permission from SZP. You may not decompile, disassemble, reverse-engineer or modify the Software.
It is strictly and expressly prohibited the redistribution of the Software as part of a package whose main purpose is bar code generation and printing, or otherwise as part of a package that can be considered generally competitive with the Software.

CORPORATE licence:

The purchase price for the Software grants you a non-exclusive licence to use the Software with the following restrictions: this licence allows a single company to install on an unlimited number of machines within that corporation; all machines must be owned, leased, or used for the purpose of the purchasing company; an exception is the fonts can be installed on employees, or contractors personal machines if done so in order to perform work for the purchasing company (please inform all contractors that the product is a licensed product and not freeware); seperate companys that are owned by the purchasing company will need to purchase their own licence. The end user can be the purchasing company or one of its customers (but not both). The purchasing company must notify to SZP (by using the appropriate form) end user details and places of installation. SZP retains title and ownership of the Software.
You may make one copy of the software solely for archival purposes.

You may not rent, sell, lease, sub-licence, time-share or lend the Software to a third party or otherwise transfer this Licence without written permission from SZP. You may not decompile, disassemble, reverse-engineer or modify the Software.
It is strictly and expressly prohibited the redistribution of the Software as part of a package whose main purpose is bar code generation and printing, or otherwise as part of a package that can be considered generally competitive with the Software.

DEVELOPER licence:

The purchase price for the Software grants you a non-exclusive licence to use the Software with the following restrictions: the Software can be redistributed as part of a package developed by the purchasing company, provided that no more than 10% of the functions of the resulting product depends on BCF and provided that they do not represent a fundamental part of the package created.
The new package cannot be a derivative of BCF and cannot include more than 3 (three) symbologies from BCF. Whenever these conditions cannot be respected, for every package sold by the purchasing company it will be necessary to buy a new SITE or CORPORATE licence, according to the application. The end user of the new package will have to accept and obey the restrictions applying to a SITE licence.
The purchasing company must notify to SZP (by using the appropriate form) end user details and places of installation; SZP reserves the right to deny distribution of the BCF files with the applications registered. SZP retains title and ownership of the Software.
You may make one copy of the software solely for archival purposes.
You may not rent, sell, lease, sub-licence, time-share or lend the Software to a third party or otherwise transfer this Licence without written permission from SZP. You may not decompile, disassemble, reverse-engineer or modify the Software.
It is strictly and expressly prohibited the redistribution of the Software as part of a package whose main purpose is bar code generation and printing, or otherwise as part of a package that can be considered generally competitive with the Software.

For any type of licence:
if you fail to comply with any of the terms and conditions of this Agreement, this Licence will be terminated and you will be required to immediately return to SZP, the Software, diskettes and documentation provided in this package, together with all back-up copies. The provisions of this Agreement which protect the proprietary rights of SZP will continue in force after termination.

LIMITED LIABILITY

The software and documentation are sold AS IS. You assume responsibility for the selection of the Software to achieve your intended results, and for the installation, use and results obtained from the Software. SZP makes no representations or warranties with regard to the Software and documentation, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.
SZP shall not be liable for errors or omissions contained in software or manuals, any interruption of service, loss of business or anticipatory profits and/or for incidental or consequential damages in connection with the furnishing, performance or use of these materials.

LIMITED WARRANTY

For a period of twentyfour (24) months from date of purchase, SZP warrants to the original purchaser, that the disks on which the Software is recorded are free from defects in material and faulty workmanship when subject to normal use and service. If, during this twentyfour (24) month period, a defect should occur, the disk will be replaced free of charge after it is returned to SZP.
If a defect occurs after the expiration of this warranty period, certain charges may apply. SZP reserves the right to refuse repeated replacement requests.

This Limited Warranty gives you specific legal rights and you may also have other rights which vary from state to state. Some states do not allow the limitation or exclusion of implied warranties or of consequential damages, so the above limitations or exclusions may not apply to you.
You agree that this is the complete and exclusive statement of the Agreement between you and SZP which supercedes any proposal or prior agreement, oral or written, and any other communications between us regarding the subject matter of this Agreement. This Agreement shall be construed, interpreted and governed by the Italian laws and any controversy will be treated by the forum of Venice – Italy. If any provision of this Agreement is found unenforceable, it will not effect the validity of the balance of this Agreement, which shall remain valid and enforceable according to its terms.
You agree that this is the complete and exclusive statement of the Agreement between you and SZP which supercedes any proposal or prior agreement, oral or written, and any other communications between us regarding the subject matter of this Agreement. This Agreement shall be construed, interpreted and governed by the Italian laws and any controversy will be treated by the forum of Venice – Italy. If any provision of this Agreement is found unenforceable, it will not effect the validity of the balance of this Agreement, which shall remain valid and enforceable according to its terms.