Morovia DataBar Fontware 1.0 Reference Manual

Morovia DataBar Fontware 1.0 Reference Manual

(Docket: 2899)

Last updated on February 2, 2009

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Morovia DataBar Fontware is a font-based solution to print GS1 DataBar symbols. DataBar, formerly known as Reduced Space Symbology (RSS), is a GS1-endorsed solution to encode 14-digit *GTIN* numbers alongside other complementary information, such as sell by date and price. There are seven types of DataBar formats - *GS1 DataBar*, *GS1 DataBar Truncated*, *GS1 DataBar Limited*, *GS1 DataBar Stacked* and *GS1 DataBar Stacked Omnidirectional*. Morovia DataBar Fontware supports all of them [1]. It also support creating *Code128* and *EAN-128* (also referred as GS-128) barcodes.

This package includes the following contents:

- Two true type fonts targeting 300/600 dpi laser printers as well as 203-dpi barcode printers/fax machines - `mrvdatabar-13x.ttf` and `mrvdatabar-34x.ttf`.
- Two true type fonts targeting screen resolution: `mrvdatabar-13x-96dpi.ttf` and `mrvdatabar-34x-96dpi.ttf`.
- A special designed true type font, `mrvdatabar-em.ttf` to be used to create stacked symbols in some software that does not handle font height properly.
- The user manual, which you are reading on.
- The PCL package includes two *PCL* scalable fonts, `mrvdatabar-13x.sft` and `mrvdatabar-34x.sft`, as well as test file to work on Windows and Linux.
- *DataBar Encoder GUI*, a GUI program to create barcode strings based on data entered.
- A Windows native DLL that provides standard API that provides encoding functions to Microsoft Office programs, as well as programming environments that support calling DLL.
- A *Crystal Reports* extension DLL that adds DataBar printing functionality to *Crystal Reports*.

All files are packaged in a single zip file. The root directory contains the installer for Windows operating system. The `PCL` directory contains the font files and test files for work on PCL printers.

1.1. Installing Morovia DataBar Fontware

1.1.1. To Install from a CD

1. Insert the program CD into your CD drive. The setup starts automatically. Or if the auto-run feature isn't enabled on your system, click the Windows *Start* button and choose the *Run* command. Type `D:\Setup.exe` in the dialog box and click the *OK* button (Note that `D` represents the letter assigned to your CD-ROM drive. If your drive is assigned to a different letter, use it instead of `D`).
2. Follow the on-screen instructions.
3. Your will be prompted to enter the `License To` and `Registration Code`. The `License to` and `Registration Code` information are found on the back of the CD case.

[1] Morovia DataBar Fontware is designed to create standalone DataBar symbols. It does not support creating DataBar symbols with a 2D composite component.

1.1.2. To Install from direct download

1. Click the Download link to start the download.
2. When the browser prompts, do one of the following: A. To run setup immediately, click Open or Run This Program from Its Current Location. B. If you decide to run the setup at a later time, click `Save` or `Save This Program to Disk`.
3. If you choosed `Save This Program to Disk` in Step 2, locate the file where you saved it, and double click the .zip file to unzip the file.
4. Locate the .msi file under the root directory of the zip file, double click it to launch setup.
5. Follow the setup instructions.
6. Your will be prompted to enter the `License To/ Registration Code`. The `License To` and `Registration Code` information can be found in the email we send to you after order completes.

1.2. Limitations of Trial Version

A trial version is provided on our web site that can be downloaded freely. In the trial version, whenever an encoder function is called, a reminder dialog appears on the screen. This behavior does not appear in the full version. All other functionality are identical between the two.

Chapter 2. Overview: DataBar, Fonts, and Encoder

This chapter briefly reviews the symbologies (a.k.a. barcode formats) supported by DataBar Fontware. DataBar Fontware supports all DataBar symbologies, plus Code 128 and GS1-128 (formerly known as UCC/EAN-128).

2.1. GTIN and GS1-128

2.1.1. Global Trade Item Number (GTIN)

GTIN is the acronym for Global Trade Item Number, a 14-digit number that identifies trade items developed by GS1 organization. This number has many names, such as SCC-14 (Serial Container Code) and UCC-14.

GTIN can be derived from *UPC-A* or *EAN-13* numbers. The first digit is package indicator. Digit '0' and '9' have special meanings here - '0' often means that there is one item in the box, and '9' indicates a variable measure item. The package indicator is followed by GS1 company prefix (assigned by GS1) and item number (assigned by the company). They should be in total of 12 digits. This portion is the same as the first 12 digits in an EAN-13 number, or '0' plus the first 11 digits in a UPC-A number. The last digit is checksum, which is calculated based on Mod10 algorithm on previous 13 digits.

Because UPC-A and EAN-13 numbers can be thought as special cases of GTINs (the package indicator is '0'), a 14-digit GTIN uniquely identifies any trade item (a single item or a container).

Traditionally, GTIN is often depicted using Interleaved 2 of 5 or Code 128 symbologies. This is expected to change as GS1 is endorsing DataBar. The benefits of using DataBar is that it produces more compact barcodes, especially when comparing with UPC-A and EAN-13 symbols.

2.1.2. GS1-128

In addition to the GTIN, a trade item often communicates more information such as price, sell by date etc. In order to communicate those information in an unified method, GS1-128 was born. Each piece of data is assigned a numeric value, called *Application Identifier* (AI). The AI identifies the meaning and the format of the data that follows it (data field). For example, the AI for *batch number* is `10`, and the batch number AI is always followed by an alphanumeric batch code not to exceed 20-characters. In the barcode representation, both AI and value are encoded so that the information is understood throughout the logistics chain.

A typical GS1-128 data consists of multiple data fields concatenated together. For example,

```
(01)19421123450011(15)991231(10)101234
```

The data above contains multiple AIs and values:

- `01` indicates that the data followed `19421123450011` is the primary identification of the item, GTIN.

- `15` is the AI for Sell by Date. The value followed `991231` indicates that the Best Sell Date is December 31, 1999.
- `10` is the AI for Batch Number. According to the specification, it is a variable length AI. Here the value is `101234`.

GTIN can be thought as a special instance of GS1-128. In fact, there is an AI for GTIN numbers, `01`. When GTIN is coded in DataBar barcodes, the AI value 01 is required to transmitted to the application even it is not coded into the barcode.

DataBar-14, DataBar Stacked, DataBar Stacked Omnidirectional, DataBar Limited and DataBar Truncated encode GTIN only. DataBar Expanded and DataBar Expanded Stacked are capable of encoding any GS1-128 data, up to 74 numeric or 41 alphabetic characters.

2.2. DataBar Family Symbologies

DataBar family formerly referred to as Reduced Space Symbology, or RSS, adopted its official new name *GS1 DataBar* on February 12, 2007. The GS1 board, formerly known as UCC/EAN organization, has declared that "GS1 DataBar symbols and GS1 Application Identifiers shall be available in all trade item scanning systems beginning Jan 1, 2010."[1]

GS1 DataBar is really a family of bar code symbologies. Some are very small, intended for produce and small consumer packages. And some are larger, intended to carry more data needed for identifying variable-measure foods and the required content on coupons. Some can be read omnidirectional, which makes them perfectly suitable for POS applications.

Table 2.1. GS1 DataBar Family

Variant	Data Encoded	POS	Applications	Sample Barcode
DataBar Omnidirectional	14-digit GTIN	Yes	Packaged goods	
DataBar Stacked Omnidirectional	14-digit GTIN	Yes	Packaged goods, Produce	
DataBar Expanded	Any GS1-128 data, up to 74 digits or 41 alphanumeric	Yes	Variable-measure food, Coupons	
DataBar Expanded Stacked	Any GS1-128 data, up to 74 digits or 41 alphanumeric	Yes	Variable-measure food, Coupons	

[1] Dubbed as "GS1 DataBar Sunrise 2010." For more information, see http://www.gs1.org/databar/.

Variant	Data Encoded	POS	Applications	Sample Barcode
				
DataBar Truncated	14-digit GTIN	No	Health care item	
DataBar Stacked	14-digit GTIN	No	Health care item	
DataBar Limited	14-digit GTIN	No	Health care item	

Among the seven variants, four, DataBar-14, DataBar Stacked Omnidirectional, DataBar Expanded and DataBar Expanded Stacked were designed and specifically to work at retail POS because they can be omnidirectionally read. The remaining three, DataBar Truncated, DataBar Stacked, and DataBar Limited, are not recommended to work at retail POS and were designed for very very small products (such as health-care items).

2.3. Creating Barcodes Using DataBar Fontware

Creating barcodes using fonts involves two distinct processes: encoding and rendering. Rendering is to choose the appropriate font and font size and format the encoding results. Encoding is to convert the data into a special string, refereed as *Barcode String*, which becomes a barcode after being formatted with the font.

Note

In almost all circumstances, you can not just type your number and format with a barcode font to create a valid barcode. You must generate the barcode string first, and format the barcode string with the font.

```
amammmamaamaaaaaamaammmaaaaaaaamamaaaamammmmmamama
mammnpsqsumvsurvspspmpmumumuntnpsprvrunvnpmpnvamam
```

We will explain in detail the font and size selection in Chapter 3, *Choosing the Right Font and Size*.

2.3.1. Encoding

To create a valid databar barcode, you need to call an encoder to get a special string, and format this string with our databar font. To get this string, you need to call an encoder - you can run *DataBar Encoder GUI*, and obtain the result from this *GUI* program. Or if you need to bulk generate the results, using the programming interface exposed from the encoder DLL.

DataBar Encoder GUI
: This GUI program allows you to quickly create barcode string and transfer it to other programs. You are able to enter the data encoded, choose an appropriate databar font, and create the barcode on the fly. For more information, see Chapter 4, *Using Encoder GUI*.

MoroviaDataBarFontEncoder.dll
: This DLL is a standard Windows DLL that can be called by many programming environments, including *Microsoft Office*, *C++*, *Foxpro* and *.Net*. Most programming environments support Windows DLL directly. We provide a VBA module and a C# class that wrap around the DLL functionality.

  For more information on this VBA module, see Chapter 5, *Adding Barcodes in Microsoft Office*. Interoperability with other programming environments are discussed in Chapter 7, *Developing Application with DataBar Fontware*.

U25MoroviaDataBarFontEncoder.dll
: This DLL is specifically designed for *Crystal Reports*. With this special extension DLL, you can add barcodes in a Crystal Reports instantly.

  For more information about using the software with Crystal Reports, see Chapter 6, *Using DataBar Fonts in Crystal Reports*.

2.4. Input Format

When you call the encoder, either through its GUI interface, or its programming interface,you need to pass the data in a format required by the encoder. The following lists the requirement for each encoder.

2.4.1. Code128_Uni

This function accept any characters with value between 0 and 255. You can also add special symbols characters, such as FNC1, FNC2, FNC3 and FNC4. In addition to plain characters, you can also use the tilde codes to escape control characters, as below:

~dnnn
: When nnn corresponds to a numeric value between 0 and 255, the tilde code sequence represents a character with value equal to nnn. For example, ~d032 represents a space character.

~~
: Represents a tilde (~) character.

~1
: Represents a `FNC1` character. The tilde escape sequence can appear anywhere in the input.

~2
: Represents a `FNC2` character.

~3
: Represents a `FNC3` character.

~4

Represents a `FNC4` character. `FNC4` is used to encode extended *ASCII* characters. You do not need to enter the FNC4 in most circumstances. Just pass the *extended characters* you'd like to encode.

~X

Represents a character value from 0 to 26. Replace the X like in the following example `~@` means character 0 (the first character in the ASCII table), `~A` means character 1, `~B` means character 2, `~C` means character 3 and so on.

2.4.2. DataBar Encoding Functions (Except DataBarExpanded)

All databar symbols, except DataBar Expanded, encode a 14-digit number called GTIN. The last digit of GTIN is a modulo 10 checksum, and is not encoded in the barcode. For these functions, you can enter 13 or 14 digits in the `DataToEncode` parameter. If 14 digits are entered, the last one is discarded.

2.4.3. EAN128_Uni

A structured GS1-128 data is required for the input. Each GS1 data consists of two parts: Application Identifier (AI) and the value. The AI dictates the constraints of value. AI must be enclosed in parentheses in order for the program to correctly parse the input. This program understands most AIs in use. For a complete AI list, see Appendix A, *Known GS1-128 AIs*.

The input can contain spaces to make the data clear to understand for humans. Spaces are not encoded into barcode. For example, `(01)98898765432106 (3202)012345 (15)991231` produces the same barcode as `(01)98898765432106(3202)012345(15)991231`.

2.4.4. DataBar Expanded

The input requirement of DataBar Expanded encoder is identical to the one for EAN128 function.

In DataBar Expanded, several AI combinations are compressed to reduce the barcode length. You might want to consider them when spaces are a concern. In all these cases, the first digit of the GTIN number must be '9'.

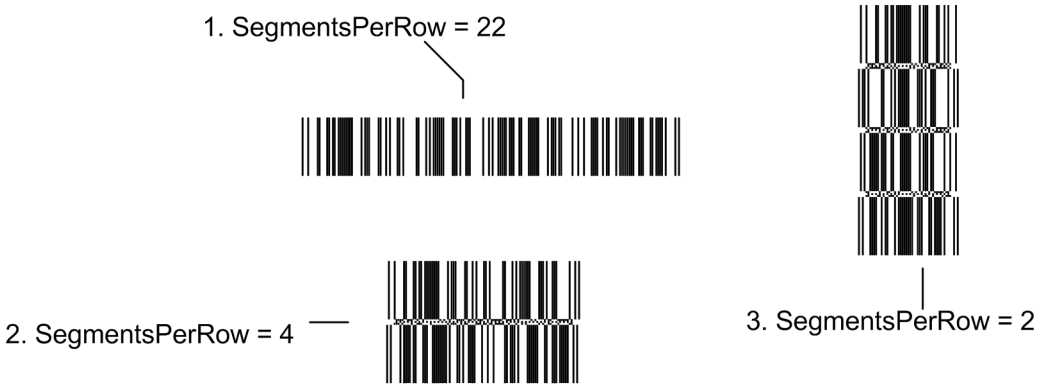
- Data consisting of only two AI elements: AI 01 followed by AI 3103. For example: `(01)90012345678908(3103)001750`
- Data consisting of only two AI elements: AI (01) followed by either 3202 or 3203. For example: `(01)90012345678908(3202)000156`.
- Data Consisting of the two or three A1 element strings for A1 01, A1 310x or 320x (x ranging from 0 to 9), and optionally A1 11, 13, 15 or 17. For example `(01)98898765432106 (3202)012345(15)991231`.
- Data starting with two element strings A1 01 followed by A1 392x. The A1 392x price may only have from zero to three digits to the right of the decimal point (x = 0 to 3).
- Data starting with two element strings A1 01 followed by A1 393x. 393x price may only have from zero to three digits to the right of the decimal point (x = 0 to 3). For example `(01)90012345678908(3932)0401234`.

	
optimized data `(01)98898765432106 (3202)012345 (15)991231`	non-optimized data. The indicator digit is not '9'. The resulted barcode is about 36% longer.

DataBar Expanded encoder function produces barcode strings for two types of barcodes: *GS1 DataBar Expanded* and *GS1 DataBar Expanded Stacked*. *GS1 DataBar Expanded* is a subset of DataBar Expanded - the number of rows is 1. This function requires you to specify the number of segment per row, which is an even number between 2 and 22. Because *GS1 DataBar Expanded* can have maximum 22 segments, setting this number to 22 will always produce non-stacked DataBar Expanded barcodes.

By changing SegmentsPerRow value, we can adjust the size of a DataBar Expanded Stacked barcode. Below lists three barcodes with the same data `(01)98898765432106 (3202)012345 (15)991231` encoded, but with different SegmentsPerRow values.

Figure 2.1. Barcode Sizes vs. SegmentPerRow



Because a DataBar Expanded Stacked barcode can grow vertically, it can encode a large amount of data in small space - a huge advantage over UCC/EAN-128. Not surprisingly, the new coupon format from GS1 is in DataBar Expanded Stacked format (with SegmentsPerRow=6). The barcode below is the new coupon format with `(81101)001414101234529011010O` encoded.

Chapter 3. Choosing the Right Font and Size

Vector graphics are often described as "can be scaled indefinitely without degrading". While this assertion holds true in many cases, it is not true when the dimension is small and the resolution of the drawing surface is low. All drawings are eventually converted to pixels, and the size of pixel imposes a severe restriction on barcode quality.

This release has taken the pixelization into count when the font was designed. As long as you follow the instruction below you should be able to create high quality barcodes, even at small dimensions on low resolution printers.

3.1. Font Selection

Unless special condition dictates otherwise, in general you should choose from the two fonts: `MRV DataBar 13X` and `MRV DataBar 34X`. The former produces barcodes at 13X (i.e. the bar height is 13 times the X dimension) and the second at 34X. In order for the barcode to read omnidirectionally, you should always use `MRV DataBar 34X` to creating omnidirectional capable barcodes, such as DataBar and DataBar Expanded.

Table 3.1. Font Selection Based on Barcode Type

Barcode Type	Total Height	Font to Use	Encode Function
DataBar14	34X	MRV DataBar 34X	DataBar
DataBar Truncated	13X	MRV DataBar 13X	DataBar14
DataBar Limited	13X	MRV DataBar 13X	DataBarLimited
DataBar Stacked	13X	MRV DataBar 13X	DataBarStacked
DataBar Stacked Omnidirectional	34X	MRV DataBar 34X	DataBarStackedOmni
DataBar Expanded	34X	MRV DataBar 34X	DataBarExpanded
DataBar Expanded Stacked	>=71X	MRV DataBar 34X	DataBarExpanded
Code128	34X	MRV DataBar 34X	Code128_Uni
GS1-128	34X	MRV DataBar 34X	EAN128_Uni

Because GS1 DataBar Expanded Stacked may have multiple rows, the total height depends on the number of rows: height = 34X + (number_of_rows)*37X

3.2. X Dimension

Although font characters scale linearly and print any sizes required, not all sizes will produce best quality barcodes. The reason is that printers can only address individual pixels instead of a length specified in inches or centimeters. For example, a printer dot measures 3.33 mils [1] on a 300-dpi

[1] 1 mil = 1/1000 inch.

printer. Certainly you can not print a size smaller than 3.33 mils. And you can not consistently print a length of 5 mils, because this printer either prints 3.33 mils, or 6.66 mils in this case. We call a font size `optimal` when the dots produced always remain constant. This is vital to small size barcodes where the barcode quality largely depends on the constant width of elements. On the other side, this is usually not a problem when X dimension is big enough (such as 15 and 20 mils).

The optimal sizes for DataBar fonts are integral times of 4 on a 300-dpi printer, integral times of 2 on a 600-dpi printer and integral times of 6 on a 203-dpi printer.

The following table lists all nominal X dimensions and font size correspondence for fonts `MRV DataBar 13X`, `MRV DataBar 30X` and `MRV DataBar 34X`.

Table 3.2. X Dimensions List

Font Size	X Dimension		Font Size	X Dimension	
6 pt[a]	5 mils	0.013 cm	28 pt[b]	23 mils	0.059 cm
8 pt[b]	7 mils	0.017 cm	30 pt[a]	25 mils	0.063 cm
10 pt	8 mils	0.021 cm	32 pt[b]	26 mils	0.065 cm
12 pt[a b]	10 mils	0.025 cm	34 pt	28 mils	0.071 cm
14 pt	12 mils	0.029 cm	36 pt[a b]	30 mils	0.076 cm
16 pt[b]	13 mils	0.034 cm	38 pt	31 mils	0.080 cm
18 pt[a]	15 mils	0.038 cm	40 pt[b]	33 mils	0.084 cm
20 pt[b]	17 mils	0.042 cm	42 pt[a]	35 mils	0.088 cm
22 pt	18 mils	0.046 cm	44 pt[b]	36 mils	0.092 cm
24 pt[a b]	20 mils	0.050 cm	46 pt	38 mils	0.097 cm
26 pt	22 mils	0.055 cm	48 pt[a b]	40 mils	0.101 cm

[a] Optimal size for 203-dpi printers
[b] Optimal size for 300-dpi printers

The characteristics for screen fonts (`MRV DataBar 13X 96dpi`) and `MRV DataBar 34X 96 dpi`)) are explained later in the chapter.

For example, supposing that X-dimension required is 15 mils, from Table 3.2, “X Dimensions List”, the optimal size is 18 points on a 600-dpi printer. [2]

3.3. MRV DataBar EM

In order to create a stacked barcode, the gap between two adjacent lines must be zero. Unfortunately, not all applications handle line gaps correctly, including some well-known programs, such as *Adobe InDesign* and *Microsoft Viso*.

When creating DataBar Stacked Ominidirectional and DataBar Expanded Stacked barcodes, the two rows overlap in those applications, as illustrated below.

[2] Note that 18 pt is not an optimal size for a 300-dpi printer. On the other side, it *is* an optimal size for a 203-dpi printer.

Some applications allow you to adjust line height. Others do not offer options to adjust line height. For the latter case you may attempt to replace the font with `MRV DataBar EM` and to see if the problem is solved. This font was design to have its character height match the point size.

However, the barcode created using this font is only the half size of the one created using other fonts. In order to match the size, you need to increase the font size to twice the original size. For example, to match a barcode created under 12 points using `MRV DataBar 34X`, format with 24 points with `MRV DataBar EM`.

Formatted with `MRV DataBar 34X` 12 points, line height at 100% (Microsoft Visio)

Formatted with `MRV DataBar EM` 24 points, line height at 100% (Microsoft Visio)

3.4. Screen Fonts

Two complementary fonts, `MRV DataBar 13X 96dpi` and `MRV DataBar 34X 96 dpi` are supplied for "on-screen" use. Some painting programs rasterize text under screen resolution. Some graphics APIs do not allow you to specify a resolution value when converting text to bitmaps.

Barcodes displayed on screen with the two screen fonts at designated font sizes are "screen copyable". You can use this method to quickly create a bitmap image for your web site while retaining high barcode quality. You can also use the fonts in programs such as *Microsoft Paint*.

Because the size of a screen pixel is relatively large (10 mils), the available selection of X dimensions is limited. See the table below for the font size list:

Table 3.3. X Dimensions List for Screen Fonts

Font Size	X Dimension		Font Size	X Dimension	
12 pt	10 mils	0.025 cm	24 pt	20 mils	0.050 cm

3.5. Other Considerations

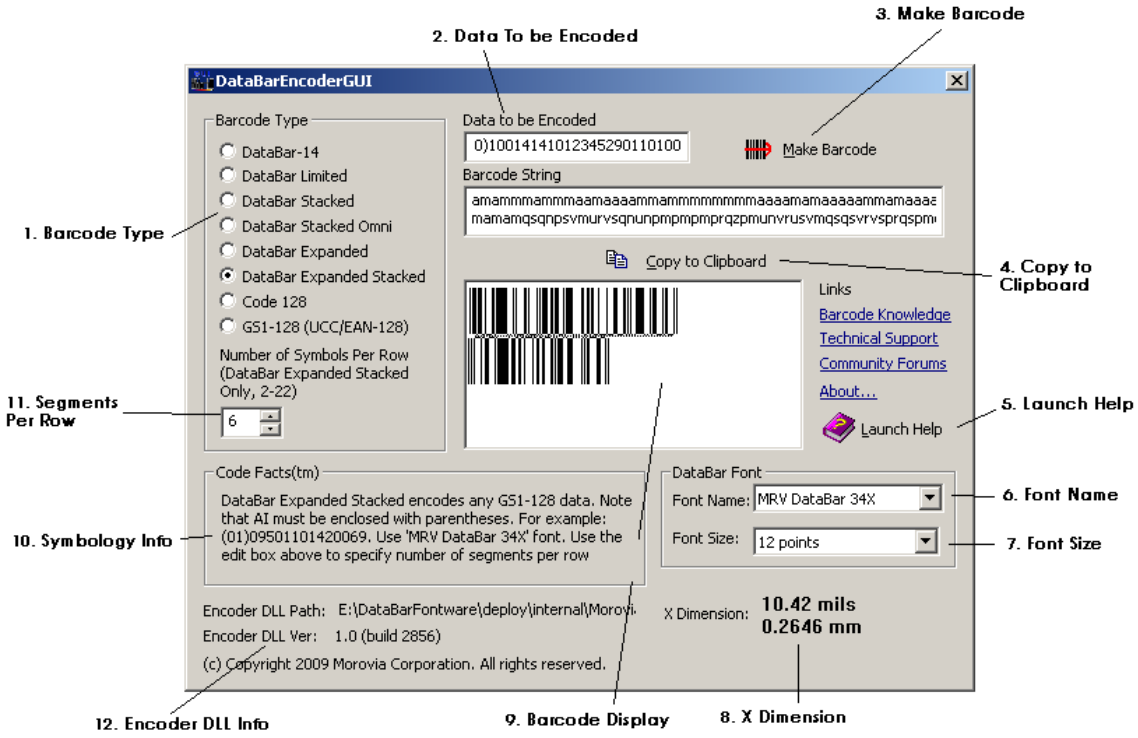
Fonts are easy to use and portable among systems. On the other side, font characters scale in both directions. That means it is impossible to get both sizes match exactly in some cases. The bar height always scales up and down at the same time X-dimension scales up and down.

Considering the potential customization requests to meet special needs, Morovia offers custom font services at reasonable cost. For example, we can create a font that meets *both* bar height and X dimension requirements accurately. Note that this service is offered only to customers who already purchased a license of the font, and the license term of the modified font follows the product to be replaced. If you have such needs, contact us at `<support@morovia.com>`.

Chapter 4. Using Encoder GUI

Morovia DataBar *Encoder GUI* is a GUI program that runs on Windows 2000 and above. It provides a fast way to create DataBar barcodes on the fly. Note that the program requires the fonts as well as the Encoder DLL to be present in the system.

Figure 4.1. DataBar Encoder GUI Options



Barcode Type

Choose the type of the barcode to make. The program lists eight options: DataBar-14, DataBar Limited, DataBar Stacked, DataBar Omni, DataBar Expanded, DataBar Expanded Stacked, Code128 and GS1-128. To create DataBar Truncated barcodes, select DataBar-14, and choose `MRV DataBar 13X`. DataBar truncated is a special type of DataBar-14 with its bar height at 13 times X dimension.

Data To Be Encoded

This is the place to enter the data to be encoded. The input format requirement varies from one symbology to other. Code128 can encode any bytes. GS1-128 and DataBar Expanded (including DataBar Expanded Stacked) encode structured GS-128 data (AI must be enclosed with parentheses). Other DataBar formats encode 14-digit GTIN (only the first 13 digits are required). DataBar Limited encode 14-digit GTIN numbers with the first digit '0' or '1'.

Make Barcode

Click on this button to refresh the barcode display.

Copy to Clipboard

Place the barcode string to the clipboard, so that you can subsequently paste them into another application such as *Microsoft Word*.

DataBar Encoder GUI places both text and RTF formats into the clipboard. In applications that are capable of processing RTF, you will see a barcode immediately after pressing the *paste* button. Otherwise, you will see the text string instead. If this is the case, highlight the whole string and format with an appropriate DataBar font.

Launch Help

Launch the HTML Help of this program.

Font Name

Select the font that applies on the barcode string. You can select any of the five DataBar fonts.

Font Size

Select the font size that applies on the barcode string.

X Dimension

Displays the X-dimension of the barcode. The X dimension is displayed in both mils and mm.

Barcode Display

This box allows you to visually inspect the barcode.

Symbology Info

Brief text about the barcode format. Read it once if you are not familiar with the symbology.

Segments Per Row

This entry only applies on DataBar Expanded Stacked. The value can be any even number between 2 and 22.

Encoder DLL Info

This section displays the absolute path and the file version of the encoder DLL. This is useful if you have multiple DLLs in the computer.

Chapter 5. Adding Barcodes in Microsoft Office

Because this is a font-based solution, you can always create barcodes in *Morovia DataBar Encoder GUI*, and paste the text into Microsoft Office. This works fine if you just want a couple of barcodes in the document. In many cases, however, it is desirable to have the barcode appear automatically and change when the data changes. This chapter explain how to.

Same as other barcode fonts, you can just enter your number and format it with the font to create a barcode. The barcode created is always unreadable. You need to get the barcode string first, and format the latter with the font in order to create a valid barcode.

5.1. VBA Functions

Microsoft Office programs can not call DLL directly. Therefore we provide a *VBA* module that can be integrated into the applications. The functions exported from this VBA module call the DLL at the back-end. In order for these functions to work, the encoder DLL (DataBarFontEncoder.dll) must locate in the search path.

Warning

On Microsoft Office 2003 and later versions, you must enable macro so that they can call VBA functions. In Office 2003, you need to set macro security to medium or low. In Office 2007, macros can be enabled on program basis.

The VBA module support the following functions:

Table 5.1. List of VBA Functions (`Morovia.DataBarFontDLL.bas`)

Function Name	Font to Use	Comment
Code128_Uni(DataToEncode As String) As String	MRV DataBar 34X	Returns the barcode string that becomes Code128 barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
EAN128_Uni(DataToEncode As String) As String	MRV DataBar 34X	Returns the barcode string that becomes GS1-128 (UCC/EAN-128) barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
DataBar14(DataToEncode As String) As String	MRV DataBar 34X [a]	Returns the barcode string that becomes DataBar-14 barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
DataBarLimited(DataToEncode As String) As String	MRV DataBar 13X	Returns the barcode string that becomes DataBar Limited barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
DataBarStacked(DataToEncode	MRV DataBar	Returns the barcode string that becomes

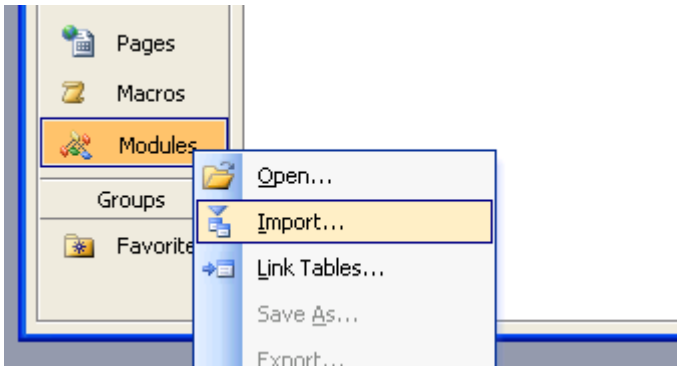
Function Name	Font to Use	Comment
As String) As String	13X	DataBar Stacked barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
DataBarStackedOmni (DataToEncode As String) As String	MRV DataBar 34X	Returns the barcode string that becomes DataBar Stacked Omnidirectional barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
DataBarExpanded(DataToEncode As String, symbolsPerRow As Long) As String	MRV DataBar 34X	Returns the barcode string that becomes DataBar Expanded or DataBar Expanded Stacked barcode for data `DataToEncode` after being formated with a Morovia DataBar Font. DataBar Expanded is a subset of DataBar Expanded Stacked. Set `SymbolsPerRow` to 0 or 22 creates DataBar Expanded barcodes.
Mod10(data As String) As String	N/A	This function calculates modulo 10 check digit based on the input `data`, and returns the original data with check digit appended. This function can be used to create full string for UPC-A, EAN-13 and GTIN numbers.

[a]To create a DataBar Truncated Barcode, use the same function but format the result with font `MRV DataBar 13X`.

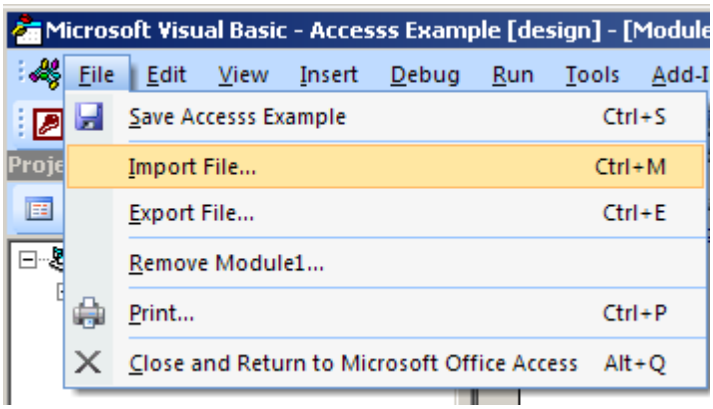
5.2. Microsoft Access

1. Before we can create barcodes in Access, we must import the required module.

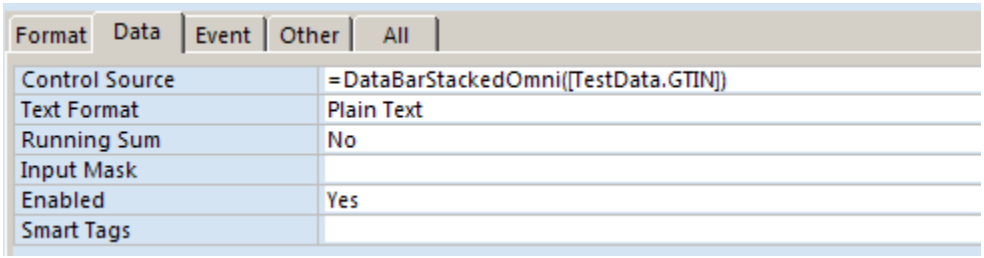
   On Access 2003 and earlier versions, choose *Modules* → *Import*. Navigate to the installation directory `c:\program files\Morovia\DataBarFontware1.0` and select `Accesss Example.mdb`.



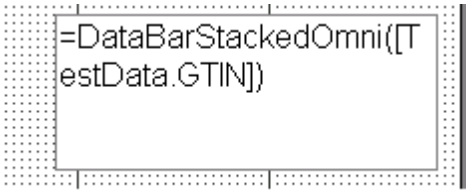
   On Access 2007, choose *Create*. On the toolbar, select *Modules* to open *Visual Basic Editor* window. Choose *File* → *Import File...*. Navigate to the installation directory `c:\program files\Morovia\DataBarFontware1.0` and select `Morovia.DataBarFontDLL.bas`. Close Visual Basic Editor after done.



2. Choose the module to import from the other database; this module should be named `Morovia_DataBarFontEncoder_Module`. After it is properly imported, it will appear as one of the modules in the database.
3. If you are working in Access 2007, the above
4. Open a report in design view and add a text box to your report. The text box will be modified to contain a barcode.
5. Right click on the text box and choose properties.
6. Place the formula `=DataBarStackedOmni([TestData.GTIN])` in the control source property of the text box where `TestData` is the table and `GTIN` is the field that contains the data you want to barcode.



7. Run the report. You should see that the formula changed the data from the database and appended additional characters at the beginning and ending of the text. You may notice that the string is completely meaningless with a series of low case letters. This is normal for DataBar barcode strings.
8. Open a report in design view, select the text box and choose `MRV DataBar 34X` as the font and choose 12 for the point size of the font.
9. Size the text box so it is large enough to contain the entire barcode. You will need to adjust both the height and width. Be sure to leave some extra space to the right and left of the barcode on the report.



10. Save the report. Click *Open* to run the report. You should see the barcodes appear on the report.

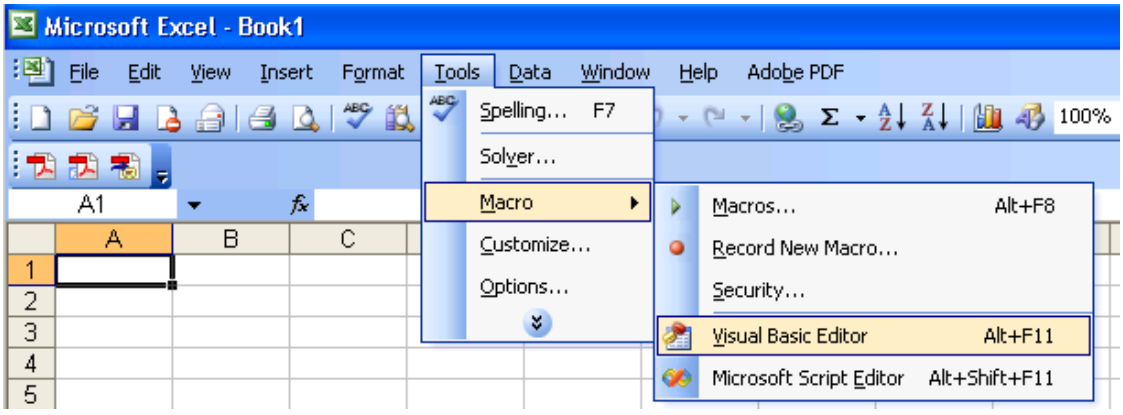
5.3. Microsoft Excel

Note

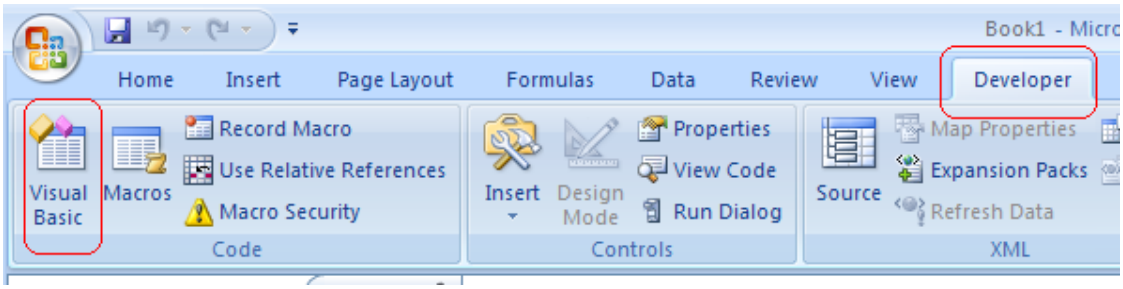
Microsoft Excel does not properly handle the line spacing required for creating DataBar Expanded Stacked and DataBar Stacked Omnidirectional symbols. You might want to consider using an ActiveX control based solution such as Morovia Barcode ActiveX [http://www.morovia.com/activex/barcode-activex.asp]. You can still use Excel as a back-end database and print the barcode using Mail Merge.

1. Before we can use the VBA functions, we must import them into the spreadsheet. First we need to open *Visual Basic Editor*.

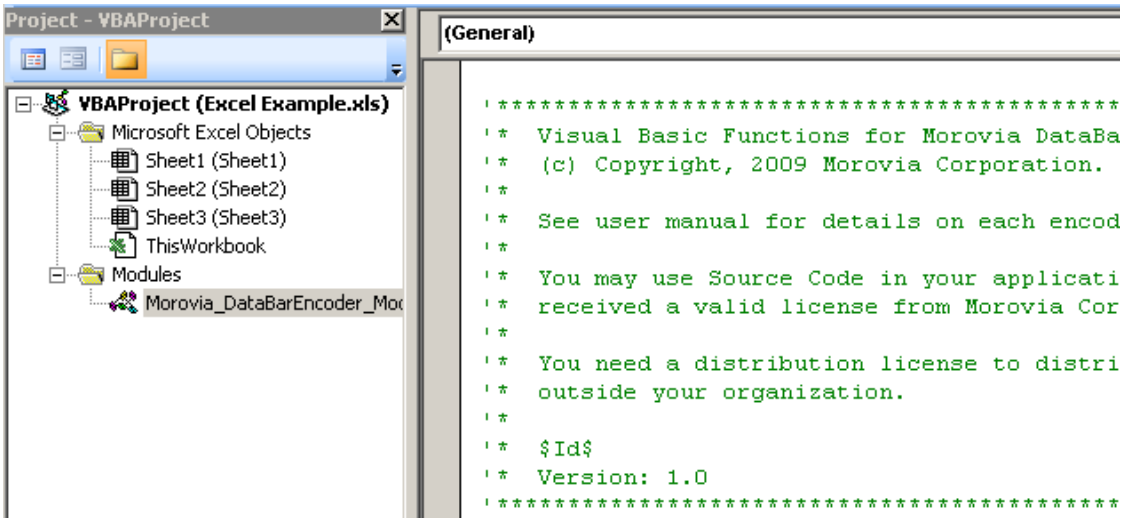
   In Excel 2003 and earlier version, chose *Tools* → Macro → *Visual Basic Editor*.



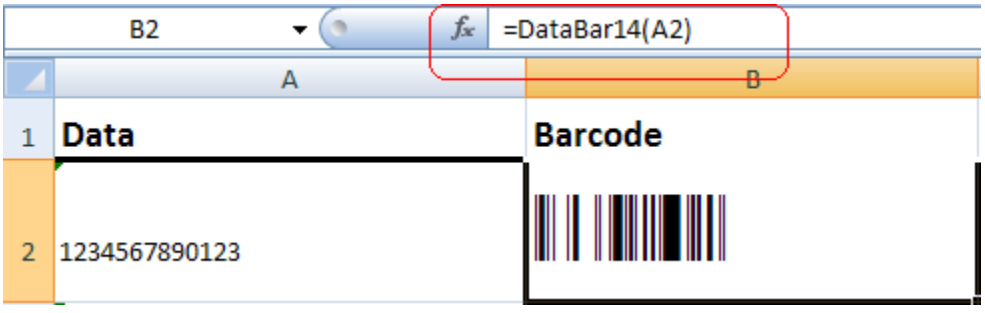
   In Excel 2007, select *Developer* and click on *Visual Basic* button on the toolbar.



2. In *Visual Basic Editor*, Choose *File* → *Import File...*and select `Morovia.DataBarFontDLL.bas` from the list of files. After this module is imported, it will be visible in the list of modules. Choose *File* → *Close*and return to Excel.

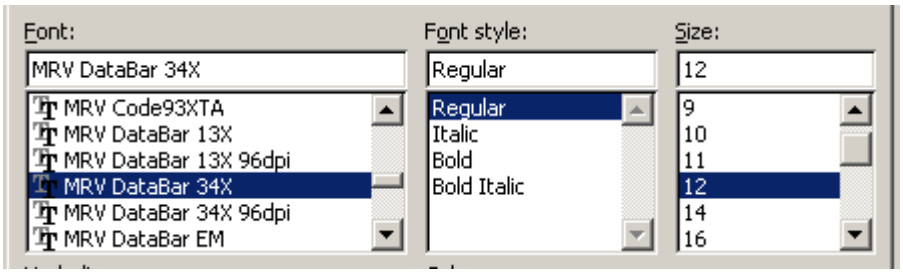


3. In the spreadsheet, choose the cell that you want the barcode appear, and enter the formula of `=DataBar14( A2)`. Here, `A2`is the cell that stores the data, and DataBar14 function indicates that the final barcode is a DataBar-14 one.



4. Now you should notice that the content of the cell turned into some illegible string such as `a1a3a1b1f1b3h1a1d1...`. This is normal.

5. Right click on the barcode cell, and choose *Format Cells...* Specify font name and size. For DataBar Limited, DataBar Truncated and DataBar Stacked barcodes, `MRV DataBar 13X` should be choosed. For rest of types, `MRV DataBar 34X` is the best choice.



6. After selecting the bar code font, you should see the barcode appear. Adjust the column width and height so that the whole barcode is visible.
7. To create an entire column of barcodes, choose *Edit* → *Copy* with the barcode cell highlighted. Highlight cells you wish to add barcodes to and choose *Edit* → *Paste*. The formula will automatically adjust for the other cells.

5.4. Microsoft Word Mail-Merge

1. To create a barcode in a Word mail-merge, we must insert a merge field from a data source that already formatted the text to the barcode font. In this example, we use Excel as the data source. The Excel spreadsheet data source must already be setup with barcodes just like the Excel Tutorial in this document.
2. In Word, Choose *Tools* → Letters and Mailings → *Mail Merge* select the Excel spreadsheet for your data source. Be sure to select the columns and range for the cells that contain the data formatted to the barcode font. You may have to go through the Word mail-merge tutorial for assistance if you are unsure of how to connect to a data source or perform a mail-merge.
3. When connected to the data source, we insert the merge field of `FormattedText` into the document. When we choose the *View Merged Data* option, we see the text formatted to the barcode font from the data source appear.
4. Select the text in the merged data and choose the `MRV DataBar 34X` font. Make the font 12 points in size.

Chapter 6. Using DataBar Fonts in Crystal Reports

Adding barcodes to *Crystal Reports* is straightforward. The example included in the software was authored in Crystal Reports 9. The fonts and UFL are compatible with Crystal version 6 and above.

6.1. The User Function DLL

The software includes a file called `U25MoroviaDataBar.dll`, which is specially designed to provide Crystal Reports with DataBar encoding functions. If you installed the software using the installer we provided, the DLL can be found under `c:\windows\system32` at default. When distributing your report, you can also copy it into the `bin` folder of Crystal Reports at `c:\program files\common files\Crystal Decisions\version\bin`. This file is *not* a COM DLL and does not require registration. After the DLL is copied, restart *Crystal Reports*.

Note

In the trial version, the DLL prompts a warning dialog when the encoder function is called. The full version does not have this limitation.

This DLL exports the following functions:

Table 6.1. List of Crystal Reports UFL Functions (`U25MoroviaDataBar.dll`)

Function Name	Font to Use	Comment
String Morovia_Code128_Uni (String)	MRV DataBar 34X	Returns the barcode string that becomes Code128 barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
String Morovia_EAN128_Uni (String)	MRV DataBar 34X	Returns the barcode string that becomes GS1-128 (UCC/EAN-128) barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
String Morovia_DataBar14 (String)	MRV DataBar 34X [a]	Returns the barcode string that becomes DataBar-14 barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
String Morovia_DataBarLimited (String)	MRV DataBar 13X	Returns the barcode string that becomes DataBar Limited barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
String Morovia_DataBarStacked (String)	MRV DataBar 13X	Returns the barcode string that becomes DataBar Stacked barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
String	MRV DataBar	Returns the barcode string that becomes

Function Name	Font to Use	Comment
Morovia_DataBarStackedOmni (String)	34X	DataBar Stacked Omnidirectional barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
String Morovia_DataBarExpanded (String, Number)	MRV DataBar 34X	Returns the barcode string that becomes DataBar Expanded or DataBar Expanded Stacked barcode for data `DataToEncode`after being formated with a Morovia DataBar Font. The second parameter specifies the number of segments per row (0-22). DataBar Expanded is a subset of DataBar Expanded Stacked.
String Morovia_Mod10(String)	N/A	This function calculates modulo 10 check digit based on the input `data`, and returns the original data with check digit appended. This function can be used to create full string for UPC-A, EAN-13 and GTIN numbers.

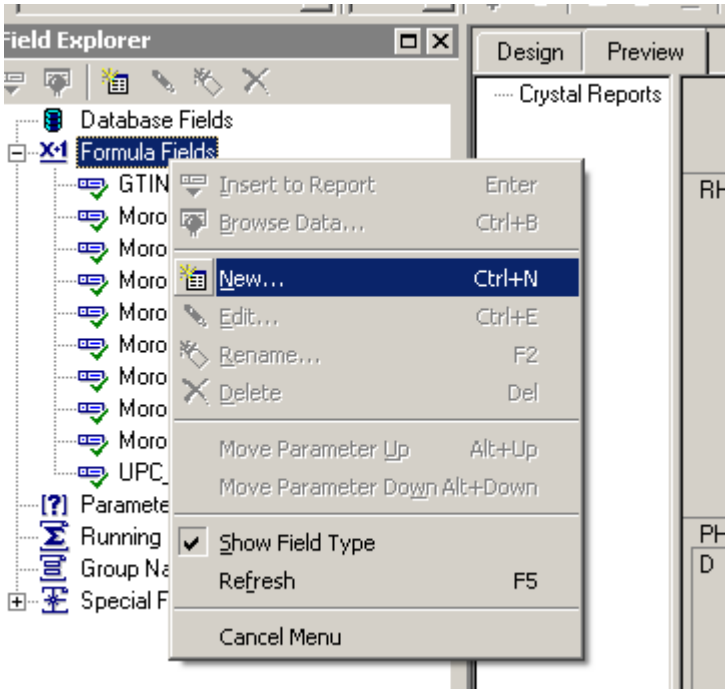
[a]To create a DataBar Truncated Barcode, use the same function but format the result with font `MRV DataBar 13X`.

6.2. Working with Crystal Reports

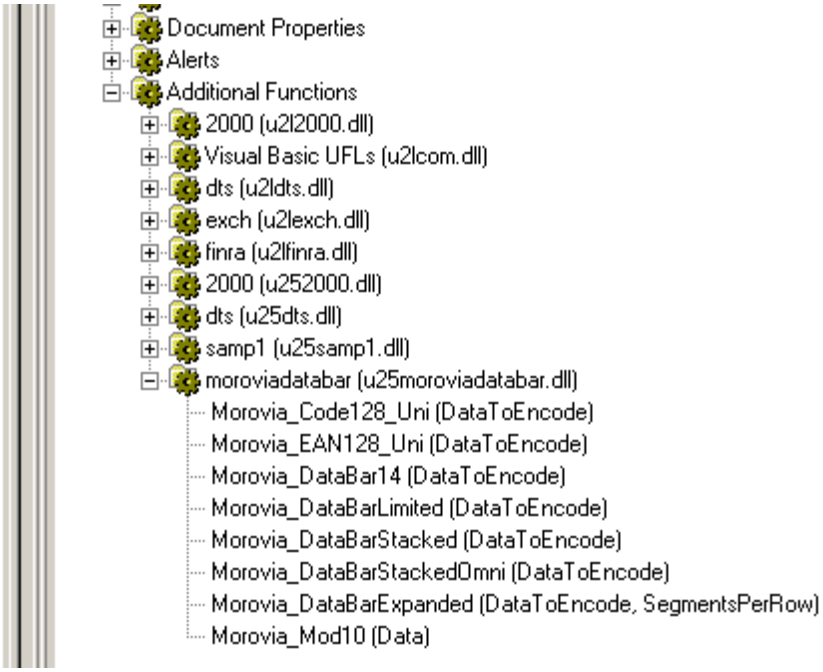
The screen shots in the following tutorial were taken in Crystal Reports 9. Interfaces in other versions may appear differently.

Assume that you have a database table with three fields: ID, GTIN and SerialNumber. You already put them into a report, and want to add a column to hold the DataBar14 barcodes for GTIN filed.

1. First switch to design mode. In version 8, choose *Insert* → *Formula Field* or in version 9 and above choose *Report* → *Formula Workshop*.
2. Right click on *Formula Fields* and choose *New*.
3. Give your formula field a name, in this example we will name it `Morovia_DataBar`. In versions 9 and above, you will be prompted to *use the editor* or *the expert*, choose *Use Editor*.



4. In the *Formula Editor,* choose *Functions* → *Additional Functions* → *moroviadatabar(u25moroviadatabar.dll)*. Select `Morovia_DataBar14` function.



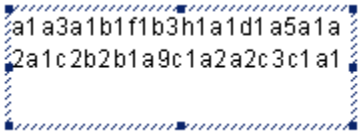
If you can not find the node `moroviadatabar` under `Additional Functions`, check if the dll is installed correctly.

5. Place the cursor between the parentheses in the formula and select the field you wish to encode in the barcode from the Report Fields area in the *Formula Editor*. A correct formula will appear

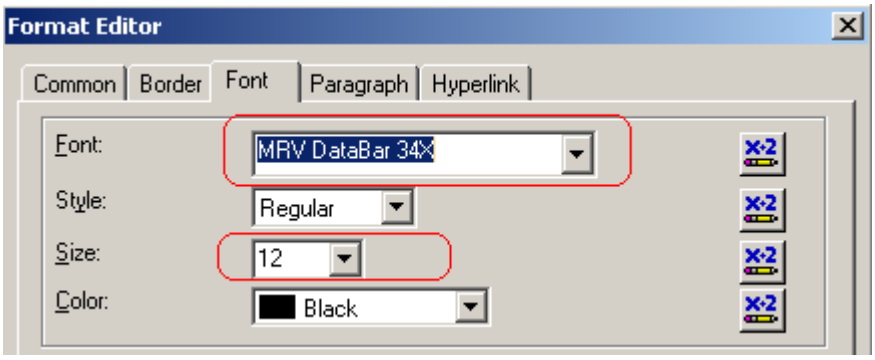
something like `Morovia_DataBar14 ({TestData.GTIN})` where `TestData.GTIN` is the source field.

```
Morovia_DataBar14 ({TestData.GTIN})
```

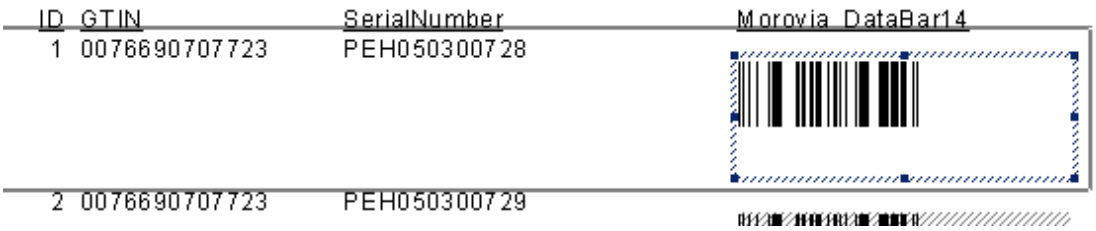
6. The tables and fields should be visible above in your database connection. Choose *Save* → *Close* to close *Formula Workshop*.
7. From the *Field Explorer*, drag the Morovia_DataBar14 Formula Field to the report.
8. Choose *File* → *Print Preview*. You should see a series of meaningless lower case letters and digits in the box. This is normal as the barcode string is completely difference from the data encoded in case of Databar.



9. Switch back to design mode. Right click on the field and choose *Format Field*. Click on the *Font* tab, and choose the `MRV DataBar 34X` font. Set the font size to 12 points or to the size appropriate for your environment. To find out how to select the font and size, see Chapter 3, *Choosing the Right Font and Size*



10. Drag the cursor to make the formula field big enough to contain the entire barcode. You may need to adjust both horizontally and vertically. Be sure to leave some spaces for quiet zone requirement.



11. Run the *Print Preview* again. The barcode should come up. Print a page and scan the barcode to make sure that the font size is appropriate and the bounding rectangle is big enough to hold the whole barcode.

ID	GTIN	SerialNumber	Morovia_DataBar14
1	0076690707723	PEH050300728	
2	0076690707723	PEH050300729	
3	0076690707725	PEH050300730	
4	0076690707724	PEH050300731	

6.3. Special Handling: DataBar Expanded

Crystal Reports restricts length of strings returned from a UFL function to 254 characters. This limit is not a problem for encoder function except `DataBarExpanded`. When a large concatenated GS1-128 is encoded, the resulted string may exceed the limit.

When the barcode string exceeds the limit, our UFL will report an error message "data is invalid, or too long". When this happens, you need to use the workaround outlined below.

Go back to the field definition and change it to the one below. Be sure to change the following variables: `DataToEncode` (data to be encoded) and `SegmentsPerRow`.

```
// Crystal Reports impose a constraint on UFL string length: max 254 characters.
// To get around this limit, use the code below
// This workaround require Crystal Report Version >= 9

StringVar DataToEncode := "(01)09501101420069(3922)995(3202)000100" +
        "(17)100101(422)123(21)123456";
// the number of segment per row
NumberVar SegmentsPerRow := 6;

StringVar BarcodeString;

// First find out how many chunks required to get the whole barcode string
NumberVar chunks := Morovia_DataBarExpanded_Set(DataToEncode, SegmentsPerRow);

NumberVar i:=0;

// Loop until all the data is retrieved
For i :=1 To chunks Step 1 Do
(
    BarcodeString := BarcodeString + Morovia_DataBarExpanded_Get;
);
```

```
// Returns the complete string
BarcodeString;
```

Note

The workaround requires Crystal Reports 9 and above because in earlier versions this "254" length limit also applies on formula fields.

The principle behind this workaround is to divide the barcode string into chunks. The first call `Morovia_DataBarExpanded_Set` returns the number of chunks needed, and in the subsequent loop `Morovia_DataBarExpanded_Get` retrieves a chunk of data each time. The data retrieved are combined into a string variable, which is returned as the value of the formula field.

6.4. Distributing UFL, Fonts with your report application

Once you finish the report design, you can distribute your report application with Crystal run time files, barcode fonts and the UFL library.

Note

Developer License is required to distribute font files and encoder UFL DLL outside your organization.

Two runtime files need to be included in your distribution package:

1. **Morovia DataBar Font Files**. The font files are located `C:\program files\morovia\DataBarFontware1.0`, and should be copied to the target computer's `fonts` folder.
2. **DataBar Encoder UFL**. This DLL is installed at `C:\windows\system32` by default. In the target computer, it can be placed in the same location, or the `bin` folder of Crystal Reports at `c:\program files\common files\Crystal Decisions\version\bin`.

Chapter 7. Developing Application with DataBar Fontware

DataBar Fontware simplifies the programming by breaking DataBar barcode creation into two distinct processes - encoding and rendering. The rendering is to format the barcode string from the encoding process with one of our font.

The encoding result is a string. If it is a stacked barcode (DataBar Expanded Stacked and DataBar Stacked Omnidirectional), the string contains two or more rows separated by line endings. On Windows, the line feed string consists of two control characters, expressed as `\r\n` in C language, and `vbCrLf` in Visual Basic. In ASCII values, they are the control character 10 followed by character 13.

DataBar Fontware includes two DLLs for integrating your applications with our software. One is a Crystal Reports extension UFL, and we have discussed it in Chapter 6, *Using DataBar Fonts in Crystal Reports*. This chapter addresses `DataBarFontEncoder.dll`, a Windows native DLL that exposes eight encoding functions. Each encoder function takes a string input and returns a string that becomes a barcode after being formatted with a DataBar font.

`DataBarFontEncoder.dll` is a regular Windows DLL, and does not require any registration. We recommend that you install the DLL in your own application's directory, which is usually searched first.

7.1. Encoder Functions

The table below lists the functions exported from this encoder DLL.

Table 7.1. Encoder Functions (`MoroviaDataBarEncoder.dll`)

Function Name	Barcode Format	Comment
int Code128_Uni(const char* DataToEncode, char* buffer, unsigned int maxSize)	Code128	Returns the barcode string that becomes Code128 barcode for data `DataToEncode` after being formated with a Morovia DataBar Font.
int EAN128_Uni(const char* DataToEncode, char* buffer, unsigned int maxSize)	GS1-128	Returns the barcode string that becomes GS1-128 (UCC/EAN-128) barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
int DataBar14(const char* DataToEncode, char* buffer, unsigned int maxSize)	GS1 DataBar, GS1 DataBar Truncated[a]	Returns the barcode string that becomes DataBar-14 barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
int DataBarLimited(const char* DataToEncode, char* buffer, unsigned int maxSize)	GS1 DataBar Limited	Returns the barcode string that becomes DataBar Limited barcode for data `DataToEncode`after being formated with a

Function Name	Barcode Format	Comment
		Morovia DataBar Font.
int DataBarStacked(const char* DataToEncode, char* buffer, unsigned int maxSize)	GS1 DataBar Stacked	Returns the barcode string that becomes DataBar Stacked barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
int DataBarStackedOmni(const char* DataToEncode, char* buffer, unsigned int maxSize)	GS1 DataBar Stacked Omnidirectional	Returns the barcode string that becomes DataBar Stacked Omnidirectional barcode for data `DataToEncode`after being formated with a Morovia DataBar Font.
int DataBarExpanded(const char* DataToEncode, unsigned int SegmentsPerRow, char* buffer, unsigned int maxSize)	DataBar Expanded, DataBar Expanded Omnidirectional	Returns the barcode string that becomes DataBar Expanded or DataBar Expanded Stacked barcode for data `DataToEncode` after being formated with a Morovia DataBar Font. DataBar Expanded is a subset of DataBar Expanded Stacked. Set `SymbolsPerRow` to 0 or 22 creates DataBar Expanded barcodes.
int getMod10CheckDigit(const char* data)	N/A	This function calculates modulo 10 check digit based on the input `data`, and returns the original data with check digit appended. This function can be used to create full string for UPC-A, EAN-13 and GTIN numbers.

[a]To create a DataBar Truncated Barcode, use the same function but format the result with font `MRV DataBar 13X`.

These are the functions directly exported from the DLL. We provide wrapper modules for some programming environments, such as *Visual Basic 6* and *.Net*. The interfaces exposed from these wrapper modules are different from the ones listed in this table.

Note

Some programing environments require the function name declared to exactly match the one in the DLL - that is, the function name is case-sensitive.

7.1.1. Parameters

DataToEncode

A pointer to a C string that specifies the data to be encoded. For `EAN128_Uni` and `DataBarExpanded`, the input must be structured GS1-128 data. For more information, see Section 2.4, “Input Format”.

buffer

A pointer to a variable that receives the encoded result (barcode string). The size of the storage area is denoted by `maxSize`.

maxSize

The maximum size, in bytes, that `buffer` can store.

SegmentsPerRow

This parameter specifies the number of segments in DataBar Expanded barcodes created. See Section 2.4.4, “DataBar Expanded” for more information.

7.1.2. Return Value

Every encoder function returns the number of bytes (excluding the terminating NUL character) of the encoding results if successful.

If the input is invalid, it return -1.

If the `buffer` is too small to hold the result, the function returns 0.

`getMod10CheckDigit` returns the check digit value (0-9).

7.2. C/C++

To use a DLL in a C/CC++ project, it is recommended that you load the DLL dynamically into the process, obtain address of the function, and invoke the function through that address. After calling the DLL function, your program unloads the DLL. In this way, it is not necessary to link your application with an import library.

The example below illustrates using run-time dynamic linking when calling the encoder DLL.

```
#include <windows.h>
#include <stdio.h>
int main(int argc, char* argv[])
{
  HINSTANCE hInstance = ::LoadLibrary("MoroviaDataBarFontEncoder.dll");
  int errorCode = 0;
  if (!hInstance) {
    fprintf(stderr, "Can not load MoroviaDataBarFontEncocer.dll.\n");
    errorCode = 100;
    goto LABEL_EXIT;
  }

  typedef int (__stdcall*ENCODE_FUNC_TYPE3)(const char*, char*, unsigned int);
  typedef int (__stdcall*ENCODE_FUCN_TYPE4)(const char*, unsigned int,
          char*, unsigned int);

  ENCODE_FUNC_TYPE3 pFuncDataBar14 = (ENCODE_FUNC_TYPE3)::GetProcAddress(hInstance,
                                          "DataBar14");
  if (!pFuncDataBar14) {
    fprintf(stderr, "Invalid DLL.");
    errorCode = 101;
    goto LABEL_EXIT;
  }

  const char* input = "0123456789123";
  char buffer[512];
  int rc = (*pFuncDataBar14)(input, buffer, 512);
  if ( rc > 0 ) {
    fprintf(stdout, "DataBar-14 barcode string for data (%s) is %s.\n",
      input, buffer);
  } else {
    fprintf(stderr, "Invalid data");
    errorCode = 101;
    goto LABEL_EXIT;
  }
```

```
  ENCODE_FUCN_TYPE4 pFuncDataBarExp =
              (ENCODE_FUCN_TYPE4)::GetProcAddress(hInstance, "DataBarExpanded");
  if (!pFuncDataBarExp) {
    fprintf(stderr, "Invalid DLL.");
    errorCode = 101;
    goto LABEL_EXIT;
  }
  input = "(01)1234567890(15)990210";
  rc = (*pFuncDataBarExp)(input, 4, buffer, 512);
  if ( rc > 0 ) {
    fprintf(stdout, "DataBar Expanded barcode string for data (%s) is %s.\n",
      input, buffer);
  } else {
    fprintf(stderr, "Invalid data");
    errorCode = 101;
    goto LABEL_EXIT;
  }

LABEL_EXIT:
  ::FreeLibrary(hInstance);
  return errorCode;
}
```

The code above first calls Win32 API `LoadLibrary` to load the DLL into the process memory. For improving readability, the code defines two function pointer types: `ENCODE_FUNC_TYPE3`, which takes three arguments, and `ENCODE_FUNC_TYPE4`, which takes four arguments. The address of the DLL exported function is then retrieved by calling `GetProcAddress` and casted to an corresponding function pointer instance. Subsequently the encoder function is invoked through this function pointer.

7.3. Visual Basic 6

You can use the same `Morovia.DataBarFontDLL.bas` module from the one in Office examples. Copy the file into your project directory, and add it to the project. For the function prototypes, see Table 5.1, “List of VBA Functions (`Morovia.DataBarFontDLL.bas`)”.

When printing GS1 DataBar Stacked Omnidirectional and GS1 DataBar Expanded Stacked barcodes, you need to call `Split` function to divide barcode string into an array of lines. Then print the lines one by one. Visual Basic does not handle line returns automatically.

```
'To print the barcode, we recommend using the built-in Printer object
'Do not print to a picture box and transfer it to printer. That results
'in a low quality barcode.

'You can change it to other values to adjust the barcode size.
Printer.FontSize = 12
Printer.FontName = "MRV DataBar 34X"

'We use inch as measurement unit here.
Printer.ScaleMode = vbInches

'We want to print the barcode at point (1.5 inch, 3 inch)
'Note - printer usually has a margin 0.25" at four directions.
'You may consider this fact when you set the printer cursor
Dim StartX, StartY As Double
StartX = 1.5
```

```
StartY = 3#
Printer.CurrentX = StartX
Printer.CurrentY = StartY

'The Priner always go back to the left edge of the page after
'executing a print statement. Unless we print the barcode to
'the left ledge of the page,
'we have to parse the barcode string into lines, and print
'them one by one.
'Fortunately, VB provides a nice helper function for this purpose.
Dim separator As String
Dim lines() As String
separator = vbCrLf
lines() = Split(lblBarcodeString.Caption, separator)

'Now we do the actual print line by line
Dim i As Integer

For i = 0 To UBound(lines)
  Printer.Print lines(i)
  'Note: the Printer always go back to the beginning of the next line
  'after executing Print. We need to set CurrentX every time after
  'printing a line
  Printer.CurrentX = StartX
Next

'EndDoc must be called to actual print
Printer.EndDoc
```

7.4. Perl

In Perl, you can use `Win32::API` [http://search.cpan.org/~cosimo/Win32-API-0.58/API.pm] module, the Perl Win32 API Import Facility to call a Windows DLL function. The code below demonstrates how you can call a DLL function from a Perl script:

```
#!/usr/bin/perl -w
use strict;
use Win32::API;

my $DataBarStacked = Win32::API->new('MoroviaDataBarFontEncoder',
                                     'DataBarStacked',
                                     'PPN', 'I');
my $DataToEncode = "1234567890123";
my $Buffer = " " x 512;  # allocate a 512-byte buffer

$DataBarStacked->Call($DataToEncode, $Buffer, 512);
printf("Barcode String is %s\n", $Buffer);
```

7.5. .Net Platform (All Versions)

For .Net programmers we prepared a wrapper class called `DataBarFontEncoder`. Add file `Morovia.DataBarFontEncoder.cs` into your project, and you can call the methods as if they are local. Note that the prototypes of the methods are different from the one listed in Table 7.1, “Encoder Functions (`MoroviaDataBarEncoder.dll`)”. They are very similiar to the VBA functions in Table 5.1, “List of VBA Functions (`Morovia.DataBarFontDLL.bas`)”.

In the following code snippet, `DataBarStacked` method is invoked to get the barcode string for DataBar Stacked barcode with data `1234567890123` encoded. The second part of the code gets the full string (with check digit) of a GTIN-14 number.

```
DataBarFontEncoder encoder = new DataBarFontEncoder();
string DataToEncode = @"1234567890123";
string result = encoder.DataBarStacked(DataToEncode);
Console.WriteLine("BarcodeString for {0} is {1}",
  DataToEncode,
  result);

result = encoder.Mod10(DataToEncode);
Console.WriteLine("Full GTIN number (with check digit) is {0}",
  result);
```

7.6. Other Programming Environments

Most programming environments support calling native Windows DLL directly. You usually declare the functions exported from the DLL first, then use them as if they are local functions.

- For Sybase *PowerBuilder*, see Prototyping API Calls for PowerBuilder [http://www.sybase.com/detail?id=44648] for more information on interoperability between *PowerBuilder* and `MoroviaDataBarEncoder.dll`.
- For *Visual FoxPro* users, see DECLARE - DLL Command [http://msdn.microsoft.com/en-us/library/aa977530(VS.71).aspx] in the language reference manual.

Chapter 8. Technical Support

Morovia offers a wide variety of support services. To help you save time and money when you encounter a problem, we suggest you try to resolve the problem by following the options below in the order shown.

- Consult the documentation. The quickest answer to many questions can be found in the Morovia product documentation.
- Review the tutorial and sample applications. The tutorial steps you through the development process for a typical barcode application. The sample applications provide working code examples in several programming languages. All sample applications are extensively commented.
- Access Morovia Online. Morovia Online provides a knowledge base which documents the frequently asked questions and a web forum.

  The web address for knowledge base is http://support.morovia.com. You can ask question at support forum at http://forum.morovia.com.
- Contact Morovia Technical Support Service. The Technical Support service is provided for free up to 180 days after the purchase. Email Morovia support engineers at <support@morovia.com>.

  Note

  If you purchased your software from our reseller, check to see if they provide support services before contacting Morovia.

Support services and policies are subject to change without notice.

Appendix A. Known GS1-128 AIs

Table A.1. List of Known AIs

AI	Name	Constraint	Short Name
00	SSCC (Serial Shipping Container Code)	n2+n18	SSCC
01	Global Trade Item Number	n2+n14	GTIN
02	GTIN of Trade Items Contained in a logistic unit	n2+n14	CONTENT
10	Batch or lot number	n2+an..20	BATCH/LOT
11	Production date (YYMMDD)	n2+n6	PROD DATE
12	Due date (YYMMDD)	n2+n6	DUE DATE
13	Packaging date (YYMMDD)	n2+n6	PACK DATE
15	Best before date (YYMMDD)	n2+n6	BEST BEFORE or SELL BY
17	Expiration date (YYMMDD)	n2+n6	USE BY OR EXPIRY
20	Product variant	n2+n2	VARIANT
21	Serial number	n2+an..20	SERIAL
22	Secondary data for specific health industry products	n2+an..29	QTY/DATE/BATCH
240	Additional product identification assigned by the manufacturer	n3+an..30	ADDITIONAL ID
241	Customer part number	n3+an..30	CUST. PART NO.
242	Made-to-Order Variation Number	n2+n..6	Variation Number
250	Secondary serial number	n3+an..30	SECONDARY SERIAL
251	Reference to source entity	n3+an..30	REF. TO SOURCE
253	Global Document Type Identifier	n3+n13+n..17	DOC. ID
254	GLN Extension component	n3+an..20	GLN EXTENSION
30	Variable count	n2+n..8	VAR. COUNT
310n-369n	(Trade and logistic measurements)	n4+n6	--
337n	Kilograms per square metre	n4+n6	KG PER m2
37	Count of trade items contained in a logistic unit	n2+n..8	COUNT
390(n)	Amount payable - single monetary area	n4+n..15	AMOUNT
391(n)	Amount payable - with ISO currency code	n4+n3+n..15	AMOUNT
392(n)	Amount payable for a Variable Measure Trade Item - single monetary unit	n4+n..15	PRICE

AI	Name	Constraint	Short Name
393(n)	Amount payable for a Variable Measure Trade Item - with ISO currency code	n4+n3+n..15	PRICE
400	Customer's purchase order number	n3+an..30	ORDER NUMBER
401	Consignment number	n3+an..30	CONSIGNMENT
402	Shipment Identification Number	n3+n17	SHIPMENT NO.
403	Routing code	n3+an..30	ROUTE
410	Ship to - deliver to Global Location Number	n3+n13	SHIP TO LOC
411	Bill to - invoice to Global Location Number	n3+n13	BILL TO
412	Purchased from Global Location Number	n3+n13	PURCHASE FROM
413	Ship for - deliver for - forward to Global Location Number	n3+n13	SHIP FOR LOC
414	Identification of a physical location Global Location Number	n3+n13	LOC No
415	Global Location Number of the Invoicing Party	n3+n13	PAY
420	Ship to - deliver to postal code within a single postal authority	n3+an..20	SHIP TO POST
421	Ship to - deliver to postal code with Three-Digit ISO country code	n3+n3+an..9	SHIP TO POST
422	Country of origin of a trade item	n3+n3	ORIGIN
423	Country of initial processing	n3+n3+n..12	COUNTRY - INITIAL PROCESS.
424	Country of processing	n3+n3	COUNTRY - PROCESS.
425	Country of disassembly	n3+n3	COUNTRY - DISASSEMBLY
426	Country covering full process chain	n3+n3	COUNTRY - FULL PROCESS
7001	NATO stock number	n4+n13	NSN
7002	UN/ECE meat carcasses and cuts classification	n4+an..30	MEAT CUT
703(s)	Approval number of processor with ISO country code	n4+n3+an..27	PROCESSOR # s4
7003	Expiration Date and Time	n4+n10	EXPIRY DATE/TIME
8001	Roll products - width, length, core diameter, direction, and splices	n4+n14	DIMENSIONS
8002	Electronic serial identifier for cellular mobile telephones	n4+an..20	CMT No

AI	Name	Constraint	Short Name
8003	Global Returnable Asset Identifier	n4+n14+an..16	GRAI
8004	Global Individual Asset Identifier	n4+an..30	GIAI
8005	Price per unit of measure	n4+n6	PRICE PER UNIT
8006	Identification of the component of a trade item	n4+n14+n2+n2	GCTIN
8007	International Bank Account Number	n4+an..30	IBAN
8008	Date and time of production	n4+n8+n..4	PROD TIME
8018	Global Service Relation Number	n4+n18	GSRN
8020	Payment Slip Reference Number	n4+an..25	REF No
8100	GS1-128 Coupon Extended Code - NSC + Offer Code	n4+n1+n5	-
8101	GS1-128 Coupon Extended Code - NSC + Offer Code + end of offer code	n4+n1+n5+n4	-
8102	GS1-128 Coupon Extended Code - NSC	n4+n1+n1	-
90	Information mutually agreed between trading partners (including FACT DIs)	n2+an..30	INTERNAL
91-99	Company internal information	n2+an..30	INTERNAL

Appendix B. Fontware License Agreement

By using or installing font software (referred as "Fontware" and "SOFTWARE" in this agreement, including fonts, components, source code, install program etc.) created by Morovia Corporation (referred as "Morovia" below), you (or you on behalf of your employer) are agreeing to be bound by the terms and conditions of this License Agreement. This License Agreement constitutes the complete agreement between you and Morovia. If you do not agree to the terms and condition of the agreement, discontinue use of the Fontware immediately.

License Grant

Number of Installation or Users: In consideration for the license fee paid, Morovia grants to you only, the License, the non-exclusive, non-transferable right to use the font in accordance with the license you purchase. If you are using this product for your employer, this agreement also includes your employer. You may only use the font on computers (CPUs) for which the Fontware is licensed.

The Single User License allows an individual to use the license Fontware on 1 CPU in your organization connected to any number of printers or other image producing devices. If you install or use the fonts on more than one CPU in your organization, or multiple individuals access the barcode printing functionality (e.g. printing from a printer server), multiple single user licenses must be purchased.

The Corporate License allows unlimited use of the licensed fonts in the organization that purchases it. After the Corporate License is purchased, the fonts may be installed and used on multiple CPUs in that organization without any additional fees to be paid to Morovia.

The Developer License allows 1 developer to install the fonts within his organization (Corporate License) as well as rent, lease or distribute the licensed fonts bundled with an application up to 10,000 users (formerly referred as Distribution License). The developer may not resell, rent, lease or distribute the fonts alone, they must be bundled with an application or with the application installation files. The developer may not resell, rent, lease or distribute the fonts in any way that would directly compete with Morovia. If use exceeds 10,000 concurrent users or installations, additional Developer License is required.

Copyright

All title and intellectual property rights in and to the Software Product (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the Software Product), the accompanying printed materials, and any copies of the Software Product are owned by Morovia. All title and intellectual property rights in and to the content that is not contained in the Software Product, but may be accessed through use of the Software Product, is the property of the respective content owners and may be protected by applicable copyright or other intellectual property laws and treaties. This Agreement grants you no rights to use such content. If this Software Product contains documentation that is provided only in electronic form, you may print one copy of such electronic documentation. You may not copy the printed materials accompanying the Software Product.

Distribution Limits

You must own a Developer License to distribute fonts outside your organization. You are allowed to distribute the software inside or outside your organization for up to 10,000 copies. When you distribute the software, you adhere to the following terms: (a) You may not resell, rent, lease or distribute the Software alone. The Software must be distributed as a component of an application and bundled with an application or with the application's installation files. The Software may only be used as part of, and in connection with, the bundled application. (b) You may not resell, rent, lease or distribute Software in any way that would compete with Morovia. (c) You must include the following MOROVIA copyright notice in your Developed Software documentation and/or in the "About Box" of your Developed Software, and wherever the copyright rights notice is located in the Developed Software ("Portions Copyright (c) Morovia Corporation 2004. All Rights Reserved."). (d) You agree to indemnify, hold harmless, and defend MOROVIA, its suppliers and resellers, from and against any claims or lawsuits, including attorney's fees that may arise from the use or distribution of your Developed Software. (e) you may use the SOFTWARE only to create Developed Software that is significantly different than the SOFTWARE. (f) You must use font embedding technology when you use the SOFTWARE in PDF and WORD documents; (g) You must specify the exact domain name when creating embedded fonts for web pages. (h) All GUI programs coming with the font package are non-distributable.

Termination

This Agreement is effective until terminated. This Agreement will terminate automatically without notice from Morovia if you fail to comply with any provision contained here. Upon termination, you must destroy the written materials, the Morovia product, and all copies of them, in part and in whole, including modified copies, if any.

Warranty & Risks

The fonts provided by Morovia are licensed to you as is and without warranties as to performance of merchantability, fitness for a particular purpose or any other warranties whether expressed or implied. You, your organization and all users of the font, assume all risks when using it. Morovia shall not be liable for any consequential, incidental, or special damages arising out of the use of or inability to use the font or the provision of or failure to provide support services, even if we have been advised of the possibility of such damages. In any case, the entire liability under any provision of this agreement shall be limited to the greater of the amount actually paid by you for the font or US $5.00.

Glossary

ASCII	The character set and code described in American National Standard Code for Information Interchange, ANSI X3.4-1977. Each ASCII character is encoded with seven bits.
EAN-13	EAN is designed by the International Article Numbering Association (EAN) in Europe. It is an extension to UPC-A to include the country information. EAN-13 encodes 12 digits of numeric data along with a trailing check digit, for a total of 13 digits of barcode data.
Extended character	A character other than a 7-bit ASCII character. An extended character is a 1-byte code point with the eighth bit set (ordinal 128 through 255).
GS1 DataBar	A family of bar code symbols, including GS1 DataBar-14, GS1 DataBar Limited, GS1 DataBar Expanded, and GS1 DataBar-14 Stacked. Any member of the GS1 DataBar family can be printed as a stand-alone linear symbol or as a composite symbol with an accompanying 2D Composite Component printed directly above the GS1 DataBar linear component. Formerly known as Reduced Space Symbology (RSS).
GS1 DataBar Expanded	A bar code symbol that encodes an GTIN-14 Identification Number plus supplementary AI Element Strings, such as weight and "best before" date, in a linear symbol that can be scanned omnidirectionally by suitably programmed Point-of-Sale scanners.
GS1 DataBar Expanded Stacked	A bar code symbol that is a variation of the GS1 DataBar Expanded Bar Code Symbol that is stacked in multiple rows and is used when the normal symbol would be too wide for the application.
GS1 DataBar Limited	A bar code symbol that encodes an GTIN-14 Identification Number with Indicators of zero or one in a linear symbol; for use on small items that will not be scanned at the Point-of-Sale.
GS1 DataBar Truncated	A truncated version of GS1 DataBar-14. The height is 10X. Used for small packaging and not fit for Point-of-Sale scanners.
GS1 DataBar Stacked	A bar code symbol that is a variation of the GS1 DataBar-14 Symbology that is stacked in two rows and is used when the normal symbol would be too wide for the application. It comes in two versions: a truncated version used for small item marking applications and a taller omni-directional version that is designed to be read by omni-directional scanners. GS1 DataBar Expanded can also be printed in multiple rows as a stacked symbol.
GTIN	Acronym for Global Trade Item Number. A 14-digit number that uniquely identifies a trade item.

GUI	Acronym for graphical user interface. A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use.
PCL	Acronym for Printer Control Language, the page description language (PDL) developed by Hewlett Packard and used in many of their laser and ink-jet printers.
UPC-A	The UPC-A barcode is the most common and well-known symbology in North America. UPC-A encodes 11 digits of numeric data along with a trailing check digit, for a total of 12 digits of barcode data.

Index