Thunder Bars Introduction

Program Description

Thunder Bars is a program which generates **Bar Codes**, **Patch Codes** and **Postal Codes** that can be copied to the Windows Clipboard. Once in the clipboard, you can then insert the codes into any program that can paste graphics from the Windows Clipboard. The codes can then be pre-printed on your documents, labels, and envelopes.

Bar Code Introduction
Patch Code Introduction
Postal Code Introduction

Program **Operation**
Program **Glossary**
Thunder Bars **Source Code**

Program Operation

Thunder Bars user interface is made up of the Menu Bar, Command Buttons, and five areas called Code Orientation, Data & Messages, Code Options, Dimensions, and Drawing Area.

Menu Bar
Command Buttons
Code Orientation
Data & Messages
Code Options
Dimensions
Drawing Area

View or modify **Settings**
Ruler Options
Q&A regarding Display, Accuracy, Scaling, Speed and Hardware
Using **DDE** (Dynamic Data Exchange)

Dynamic Data Exchange

Dynamic Data Exchange (DDE) is a feature of Windows which makes it very easy to move data between applications. Thunder Bars design is such that it can optionally act as a DDE "Server". This means that you can establish a "link" to another application, (the "Client"). For example if you export a code from Thunder Bars to MS Word, any modifications that are made to a code in Thunder Bars will show up automatically in Word, but if you edit the code in Word, nothing will happen in Thunder Bars. This transfer of data is one-way only, from the server to the client.

In theory you can use Thunder Bars with any application that supports a DDE Paste Link. Some applications establish links through macros, or as menu options such as "Paste Link" or "Links", to name a few, while others will provide different methods. Though applications will differ in the procedures required to set up a DDE link, most will require some or all of the following items.

Link Name: This can usually be any name which is easily recognizable to the user.
Application Name: Thunder Bars application name is the word "Thunder".
Topic, Document, or File: Thunder Bars Topic is the word "Bars".
Item: Thunder Bars Item is "Thunder(0)" (the word Thunder followed by a zero enclosed by parentheses).

The following procedures for WordPerfect for Windows, and MS Word for Windows demonstrate two methods of setting up an automatic DDE link using Thunder Bars.

To use DDE with WordPerfect for Windows v5.1

- Start Thunder Bars and generate a code.
- Use the paper clip icon, or select Copy Code from the Edit menu to copy the code to the Clipboard.
- Start WordPerfect.
- Under the Edit menu, select Link, then choose Paste Link.
- If the code is not the expected size:
  Highlight the code by clicking it with the mouse.
  Under the Graphics menu, select Figure, then choose Position.
  In the Box Position and Size dialog, set Size to "Auto Both".

If the link was successful, any changes now made in Thunder Bars will result in the code immediately being updated in WordPerfect.

Optionally the following macro could be generated to establish the link:

```
Application (WP;WPWP;Default;"WPWPUS.WCD")
DDECreateLink
(
    LinkName:"Thunder(0)";
    Source:"THUNDER|Bars|Thunder(0)";
    UpdateMode:Automatic!;
    StoragePreference:Graphics!
)
```

To use DDE with MS Word for Windows v2.0

- Start Thunder Bars and generate a code.

- Use the paper clip icon, or select Copy Code from the Edit menu to copy the code to the Clipboard.

- Start MS Word.

- Under the Edit menu, select Paste Special, then choose Paste Link.

- If the code is not the expected size:
  Highlight the code by clicking it with the mouse.
  Under the Format menu, select Picture.
  In the Picture dialog, select "Reset".
  Note! You may need to use scaling percentages, which measure how much larger or smaller the graphic becomes to match the original code size generated by Thunder Bars. Examine the Drawing Area dialog to determine the exact size needed.

If the link was successful, any changes now made in Thunder Bars will result in the code immediately being updated in MS Word.

Optionally the following macro could be generated to establish the link:

```
Sub MAIN
EditPasteSpecial .Link = 1, .DataType = "Bitmap"
End Sub
```

Menu Options

File Menu

Save to File
: Saves the currently displayed code to disk as CODE.BMP in the startup directory.

Preferences
: View current settings and/or set defaults.

Exit
: Exits Thunder Bars.

Edit Menu

Redraw Code <F5>
: Redraws the code using the current settings.

Copy Code <Ctrl-C>
: Copies the currently displayed code along with DDE link information to the Clipboard.

Paste Text <Ctrl-V>
: Paste text from the Clipboard into any text box. If the Clipboard contents are not text, nothing is pasted.

Options Menu

Auto-Redraw
: Toggles whether or not the code is automatically redrawn immediately after changing most settings.

Save Settings on Exit
: Toggles whether or not to save settings to disk upon exit. Settings are saved to THUNDER.INI in the startup directory.

Show More
: Displays the full Thunder Bars interface.

Show Less
: Displays the smaller Thunder Bars interface.

Tools Menu

Clipboard <Ctrl-B>
: Calls the Windows Clipboard application. Use this to view and manipulate Clipboard contents, including Thunder Bars codes.

Character Map <Ctrl-M>
: Calls the Windows Character Map application. Use this to view installed fonts, and copy characters to the Clipboard, which can then be pasted into Thunder Bars text boxes.

Paintbrush <Ctrl-P>
: Calls the Windows Paintbrush application. Use this to modify, scale, flip, and enhance Thunder Bars codes copied from the Clipboard or retrieved from disk.

Write <Ctrl-W>
: Calls the Windows Write application. Use this to paste Thunder Bars codes from the clipboard into a document.

Help Menu

Contents <F1>

Loads Thunder Bars On-line Help files.

About

About Thunder Bars dialog. Displays mode, free memory, and version information.

Command Buttons

 Redraws the code

 Copies the code to the clipboard

 Runs Thunder Bars Help

 View the current settings and/or set defaults

 Sets ruler options

 Call Windows Character Map

 Call Windows Clipboard

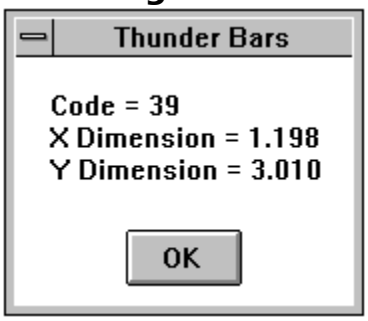
 Call Windows Paintbrush

 Call Windows Write

Display the full Thunder Bars interface

Display the smaller Thunder Bars interface

Drawing Area

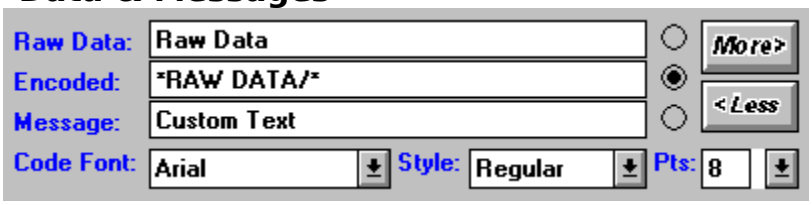


The code type and exact dimensions of a displayed code can be obtained using the above dialog. To display this dialog, move the mouse pointer over any part of the code until the cursor changes to a small box, then click once with the left mouse button. Click OK to clear the dialog.

Code Orientation

You can orient the code either vertically or horizontally. Vertical orientation is sometimes referred to as "Ladder", and is a 90° clockwise rotation. Ladder codes are drawn from top to bottom. "Picket Fence" refers to a horizontal, 0° non-rotated code. Picket codes are drawn from left to right. A single mouse click on either option button will redraw the code in the new orientation. Human readable is only available for horizontal codes.

Data & Messages



The Data & Messages area is where you input the data to be encoded, see what is actually encoded, and control the printed message (Human Readable).

The **Raw Data Box** is where you input the data to encode. For Zip+4 & Bar Codes, simply type in, or paste from the clipboard the string of data. Any invalid data will be ignored. For all other codes, you may manually input the "Code Type" here, or select the appropriate type from the Code Options areas. If an invalid type is manually input, a message will be generated indicating which types are valid.

The **Encoded Box** displays the actual data, or code type that was generated. Only valid data, valid code types, check digits and start/stop characters are displayed. This box represents the actual contents of the currently drawn code. The contents may go outside the boundaries, in which case you may scroll to the right to see the entire message. Manually changing the contents of the encoded box will have no effect on the displayed code.

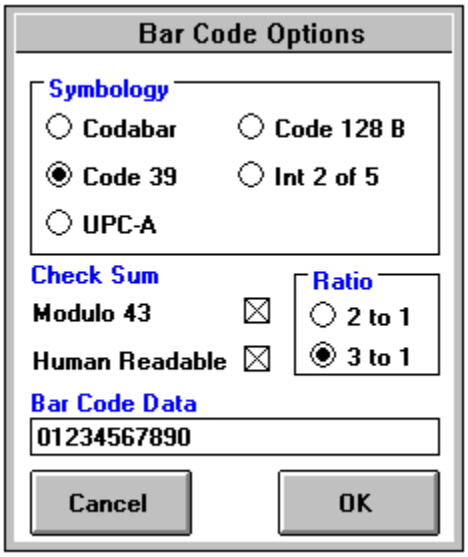
The option buttons located on the right hand side indicate which ot the text boxes will be used for the human readable. To customize the printed text (horizontal bar codes only), key the desired message in the **Message Box**. You will need to redraw the code after making any changes.

The pull down boxes are used to set the type of font, style, and point size used for the human readable. All currently installed fonts are available. Some point sizes and styles may not be applicable for a particular font.

Code Options

Bar Code Options
Patch Code Options
Postal Code Options

Bar Code Options



Symbology

Select from Codabar, Code 39, Code 128B, Interleaved 2 of 5, or UPC-A bar codes.

Check Sum

Check this box to have the check sum automatically calculated and encoded. The mathematical formula used to calculate the check sum is displayed to the left of the check box. If this option is disabled (grayed), then the use of a check sum is required for that particular bar code.

Ratio

Check the box next to the desired ratio. If these options are disabled (grayed), then that particular bar code requires a fixed ratio and may not be modified.

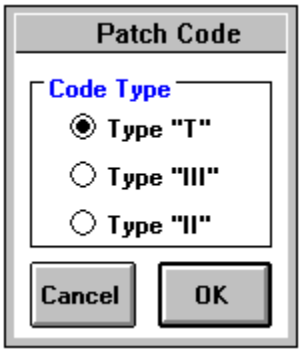
Human Readable

Checking this box enables or disables the automatic printing of the appropriate message below picket (horizontal) bar codes.

Bar Code Data

Optionally enter the desired data to encode here. This data may also be re-keyed in the raw data box in the main window.

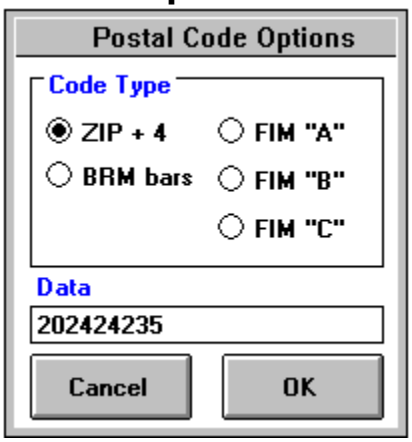
Patch Code Options



Code Type

Select from patch types T, II, and III.

Postal Options



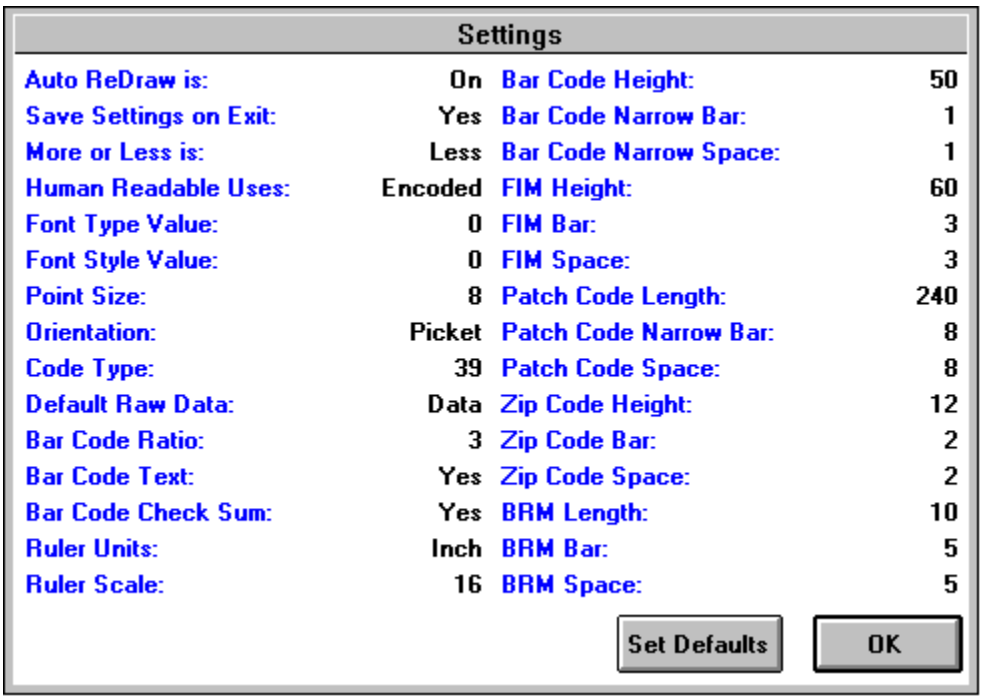
Code Type

Select from Postnet, FIM's A, B, & C, or BRM Bars.

Data

Optionally enter the Zip Code, FIM type or number of BRM Bars here. This data may also be re-keyed in the raw data box in the main window.

Preferences



Almost every Thunder Bars setting may be saved to an initialization file so that the program will look and act in the same fashion as the last time it was used. Use the "Save Settings on Exit" option in the Options Menu to enable or disable this feature. If this option is checked, the intialization file will be updated upon exiting Thunder Bars. Settings are stored in THUNDER.INI in the Thundar Bars directory. This file may be edited manually although this is not recommended. If THUNDER.INI can not be found in the startup directory, it will be re-created using default values upon exit if the save feature is enabled. These values can also be reset in the View Settings window.

Widths, Height & Length

This area allows you to change the dimensions of the code. All values are in pixels. The width of 1 pixel is dependant on the display type being used. A single mouse click on the "-", or "+" buttons will decrease or increase the values by one pixel. If the "Auto Redraw" option is enabled, the code will be redrawn after every mouse click.

If the "Auto Redraw" option is disabled, the code is not immediately redrawn after changing these settings, thereby allowing you to make significant adjustments without waiting for the code to be redrawn every pixel increase/decrease. If any of the "+" or "-" buttons have the focus (i.e., they have been clicked), AND the "Auto Redraw" option is disabled, the plus or minus keys on the keyboard will increment or decrement the values. Pressing and holding down either the plus or minus keys will enable you to change the values very rapidly.

Bars & Spaces

Changing the Narrow Bar setting will affect the width of a narrow bar (or dark element). Changing the Narrow Space setting will affect the width of a narrow space (white element or "gap"). Any changes made to the narrow elements will also affect the wide elements proportionally, according to the code's ratio setting. For example, changing the width of a narrow bar from an initial value of 1pixel to 2 pixels, for a bar code which has a 3 to 1 ratio, will result in the wide bars increasing to 6 pixels wide. Any change to a bar or space setting for a bar code will result in BOTH values being changed equally in order to maintain readability.

Height/Length

Changing the Height/Length settings affects the overall length of the elements, or put another way, the measurement of a bar or space from end to end.

Patch Code Introduction

A patch code is a pattern of parallel, alternating black bars and spaces that is printed on a document to be used in an Imaging application. Products which have a Patch Reader Accessory installed are capable of recognizing patch documents and automatically assigning a document Image Level and incrementing the document Image Address. This method of controlling document Image Level and document Image Addresses eliminates the need for an operator to manually set these values. Thunder Bars generates three different patch codes; **PATCH III**, **PATCH II**, and **PATCH T** (Transfer Patch). The transfer patch assigns a predefined Image Level to the *next* document. The predefined Image Level is based upon the Transfer Patch Definition which is defined for each application mode during installation. For example, if the Transfer Patch Definition defined for the current mode is Image Level 2, then use of a Transfer Patch assigns Image Level 2 to the *next* document.

Requirements & Specifications
Patch Code **Positioning**

Patch Code **Options**
Printing Guidelines

Patch Positioning

Horizontal and vertical placement of the patch code's are **critical** for proper operation. If the patch code is placed improperly on the document, the patch sensors may fail to sense the patch.

- Patches should appear with the bars parallel to the leading edge of the document (fed into the transport first).

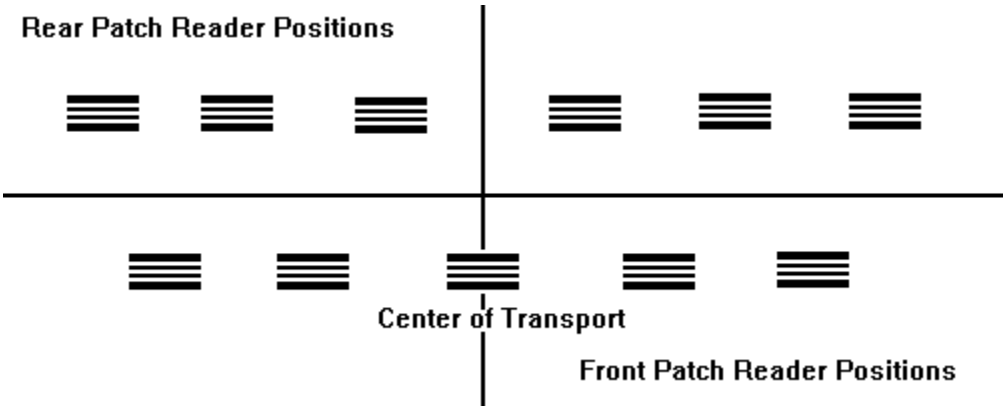
- There **must be at least** 0.20 inches (5 mm) of white space between the patch codes and any other printed information.

Horizontal Placement

- There are five front patch sensors and six rear patch sensors.

- Horizontal placement of patch codes is affected by the side guides. Make certain that the side guides are positioned so that the patch code will be transported directly past the window in which the patch reader is installed.

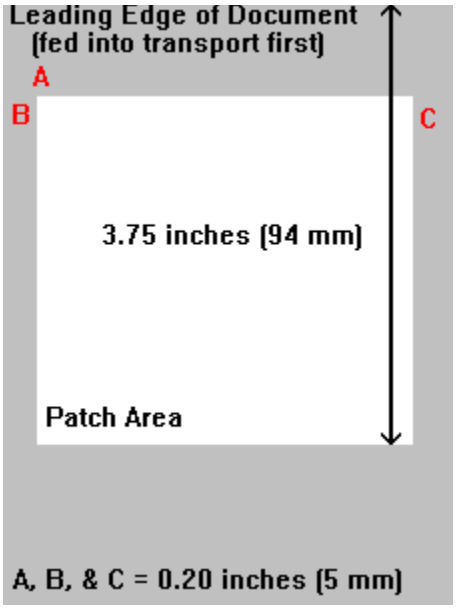
- Use of a standard 2.5 inch (62.5 mm) patch code is recomended to ensure that the patch code may still be read even if there is a slight variation in the positioning of the side guide(s).



Vertical Placement

- Patch codes **must appear at least** 0.20 inches (5 mm) from the right, left, and leading edges of the document.

- Patch codes **must appear no more than** 3.75 inches from the leading edge of the document.


Leading Edge of Document
(fed into transport first)
A
B
C
3.75 inches (94 mm)
Patch Area
A, B, & C = 0.20 inches (5 mm)

Patch Requirements & Specifications

The minimum overall length of the patch bars is 2 inches wide (50 mm). The maximum width of the patch code is 0.80 inches (20 mm) +/- 0.01 inches (0.25 mm).

The wide bars should be 0.20 inches (5mm) wide +/- 0.01 inches (0.25 mm).

The narrow bars and spaces should be 0.08 inches (2 mm) wide +/- 0.01 inches (0.25 mm).

Print the patch code only on the top page of multiple-part forms with carbon inserts. Smudged carbon in the patch code area can cause false readings.

If patch-coded forms are printed in pads, make sure that the torn edges are *not* fed into the machine first (not the leading edge). A ragged leading edge preceding the patch code may cause the distance to the first bar to be out of tolerance.

Thunder Bars generates variable length patch codes that adhere to the specifications outlined, with the two most common applications defined below:

Standard 2.5 inch (62.5 mm) long patch codes for preprinting patch code labels or when printing patch codes directly on application documents.

Full-page width and full-page length patch codes to be used as "batch headers"; interleaved among application documents. These full-page patches may be fed into the transport without regard to orientation.

Bar Code Introduction

A bar code symbol is a pattern of parallel, adjacent bars and spaces. The width's of the bars and spaces represent the actual data in the symbol. Thunder Bars generates five kinds of bar codes. Bar codes are used in a wide variety of applications, they're on your groceries, library books, invoices, etc. Almost anything can be bar coded to make identification faster and more accurate. When used with a check character, the substitution rate can often be better than 1 error in 1 million characters. Bar code is the automatic identification technology in a wide range of industries because they offer a simple and accurate, cost effective approach for identifying objects.

Bar Code Symbologies:

Codabar
Code 39
Code 128 B
Interleaved 2 of 5
UPC-A

Bar Code **Options**
Printing Guidelines

Postal Code Introduction

The Postal Service has developed a series of specifications which should be of interest to anyone who would like to reduce their company's mailing costs while benefiting from a much faster, more accurate and more efficient method of mail processing. Thunder Bars generates three types of postal codes to make it easier to produce "machinable" mailings. The United States Postal Service has representatives and volumes of documentation to assist you in meeting guidelines on preparing "Automation Compatible" and "Business Reply" mail. The dimension and positioning guidelines presented in this document are intended as a general introduction and quick reference to the Postal Service requirements and specifications for automation compatile mail. For additional, and more thorough information, contact your local account representative, automation readability specialist or a Postal Business Center.

Postnet Bar Codes **Postnet**
Facing Identification Marks **FIM**
Reply Mail Bars **BRM Bars**

Automation Compatible Letter Mail

An automation readable mailpiece is one that contains an accurate, correctly formatted, complete address or ZIP+4 code and is readable on an OCR (Optical Character Reader) and/or BCS (Bar Code Sorter).

Reply Mail

Business reply mail (BRM) enables mailers to receive First-Class mail back from customers by paying postage only on the mail which is returned to them from their original distribution of BRM pieces. The permit holder guarantees payment of the appropriate First-Class postage, plus a handling charge per piece. A permit is required annually. Courtesy Reply Mail (CRM) is the term applied to distributor-printed automation-readable mailpieces which are sent to clients for reply purposes. The correspondent must affix postage to the mailpiece prior to mailing. Credit unions, utility companies, banks and mortgage firms frequently utilize courtesy reply mail. These organizations anticipate a large response from their various mailings, and by inclding a courtesy reply card or envelope their customers show a higher response rate. The convenience of CRM increases the probability of a mail-back response.

Positioning Guidelines
Postal Code **Options**
Printing Guidelines

Postnet Bar Code

POSTal Numeric Encoding Technique (POSTNET): The **POSTNET** bar code was developed by the Postal Service to provide a system of encoding ZIP Code information on letter mail, which can be read reliably by relatively inexpensive bar code sorters. The code is made up of binary elements printed in the lower right corner of the mail piece as tall and short bars representing the ZIP Code or ZIP+4 code. The barcode may represent a five digit ZIP Code (32 bars), a nine-digit ZIP + 4 code (52 bars), or an eleven-digit delivery point code (62 bars). Thunder Bars will only generate the code if the input string contains 5, 9 or 11 valid numeric digits.

Dimensions

Height
Tall Bar 0.125" +/- 0.010"
Short Bar 0.050 +/- 0.010"

Width
0.020 +/- 0.005"

Pitch
21 +/- 1 bars per inch

Spacing
0.045" min. - 0.050" max. between centerlines of adjacent bars.

Skew
A variation of +/- 5 dgrees from the horizontal.

Total Length

Five-Digit ZIP Code (32 Bars)	1.245" - 1.625"
Nine-Digit ZIP+4 Code (52 Bars)	2.075" - 2.625"
Eleven-Digit Delivery Point Code (62 Bars)	2.495" - 3.125"

Reading the Postnet bar code
Reading and understanding the bar code is simple. There are 10 combinations of 5 bars, each consisting of 2 long (1's) and 3 short (0's) bars. The bar code consists of the digits of the ZIP code, plus a correction digit used to identify errors, contained between the frame bits represented by 1 bars. The digits 0 through 9 have been assigned to these combinations.

0 - 11000 1 - 00011 2 - 00101 3 - 00110 4 - 01001
5 - 01010 6 - 01100 7 - 10001 8 - 10010 9 - 10100

Within the group of 5 bars, each position has a different value. From the left to right, 7, 4, 2, 1, and 0. Addition of the values in the two positions occupied by 1 bars gives the value of the combination, except in the case of 11000, which totals 11 and has been assigned as zero. The sum of the digits in the bar code is always a multiple of 10.

Facing Identification Marks

Facing Identification Mark (FIM): **FIM** is another type of postal barcode used in automated processing. Currently three FIM patterns (A, B, and C) are in use. Coding certain types of letter mail with FIM patterns provides a method for the automatic facing and cancelling of letter mail, and a means of seperating business and courtesy reply mail from other letters and cards.

Dimensions

Height
5/8" +/- 1/8" (may be longer to wrap around top of envelope).

Width
0.031" +/- 0.008"

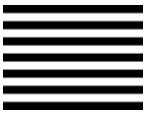
Pitch
1/16" nominal

Skew
A variation of +/- 5 dgrees from the horizontal.

Uses

FIM Type	Used On	Preprinted Postnet
A	Courtesy Reply Mail	YES
B	Business Reply, Penalty and Franked Mail	NO
C	Business Reply, Penalty and Franked Mail	YES

Business Reply Mail Bars



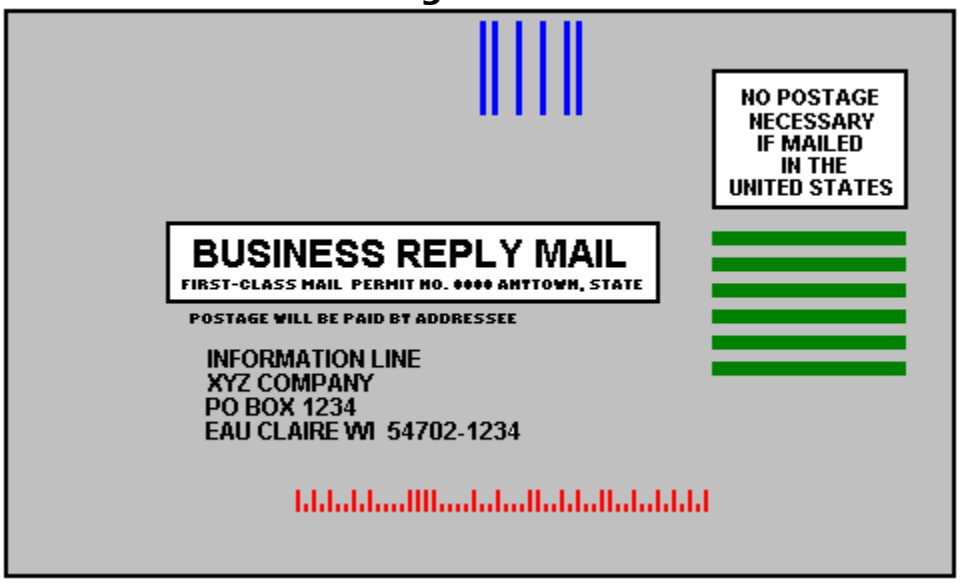
Business Reply Mail bars (BRM bars): **BRM** bars are a series of horizontal bars placed to the right of the delivery address line and directly below the "NO POSTAGE" endorsement box. BRM Bars facilitate rapid recognition of business reply mail.

Dimensions

Length
At least 1" in length (not longer than 1.5")

Thickness
1/16" to 3/16" in thickness and evenly spaced

Postal Code Positioning



Sample Business Reply Mail Piece

POSTNET Location.

Postnet bar codes are always located on the same side of the mail as the address. This area must be free of any printing other than the bar code. The bar code clear area extends up 5/8" from the bottom right edge and at least 4 3/4" leftward of the right edge of the mail piece. Within the barcode clear area, the leftmost bar of the barcode must be located 3 7/8" +/- 3/8" from the right edge of the mailpiece. The bottom, or baseline, of the barcode must be 1/4" +/- 1/16" from the bottom edge of the mailpiece. The bar code must be completely contained within the barcode read area.

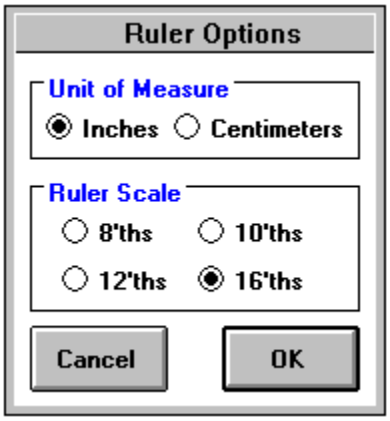
FIM Location.

A FIM clear zone must be maintained containing no printed matter other than the appropriate FIM pattern. The right boundary of this clear zone must be 1 3/4" from the right edge of the mail piece. The left boundary of this clear zone must be 3" from the right edge of the mailpiece. The top of the bars must be no lower than 1/8" from the top edge of the mailpiece, but they may touch the top of the mailpiece. The rightmost FIM bar must be 2" +/- 1/8" from the right edge of the mailpiece. The clear zone is 5/8" deep, measured from the top edge of the mailpiece.

BRM bars Location.

BRM Bars must be uniform in size, at least 1" in length (not longer than 1.5"), 1/16" to 3/16" in thickness and evenly spaced. The bars must not extend below the top of the delivery address line which is located directly above the city, state, ZIP Code line. There must be at least 1/2" clearance between the ZIP Code and the horizontal bars.

Ruler Options



Unit of Measure
Displays the ruler in either inchs or centimeter units.

Ruler Scale
Select from 1/8, 1/10, 1/12, or 1/16 increments.

Accuracy

The codes generated with this program are accurate to 1 pixel. However due to different display resolutions, and varying amount of "ink-spread" on different printers, "thickness" is adjustable. This allows you to add or subtract 1 pixel from the narrow elements (black or white). The code should be redrawn when you change any of these values. Keep experimenting until you find the values that work best for you.

Display Resolutions

The following figures illustrate how the display will affect printed output. For example if you have a VGA display, the narrowest element you can print (1 pixel) will be .010417" wide.

Output device	Resolution	x size	y size	x pixel	y pixel
CGA display	2:1	96	48	.010417"	.020833"
EGA display	1.33:1	96	72	.010417"	.013889"
VGA display	1:1	96	96	.010417"	.010417"
8514 display	1:1	120	120	.008333"	.008333"

Speed

Generating codes with very thick elements will cause a noticeable increase in the time it takes to draw the code. The method used to draw the individual lines was such that in order to offer the most flexibility and accuracy, drawing speed was sacrificed somewhat.

Bar Code Text

Bar code text (sometimes called Human Readable), is available only for horizontal codes. However many Windows applications will allow you to rotate the image after you have pasted it into your document.

Scaling

Any type of scaling (sometimes refferred to as Shrink & Grow, Stretching, or Sizing) can cause a code to be unreadable, or invalid. Some types of codes, like the 3 of 9 bar code, can be expanded or compressed in size and sometimes still be readable, while most others, UPC-A or Postnet for example, can become unreadable if just ONE PIXEL width is added or removed. The methods different applications will handle graphic images with regards to screen display and printed output will vary widely. IT IS AN ABSOLUTE REQUIREMENT THAT THE RECEIVING APPLICATION PROVIDES A METHOD IN WHICH THE CODE MAY BE PRINTED IN THE EXACT SIZE THAT IT WAS GENERATED FROM WITHIN THUNDER BARS! Some applications like Windows Paintbrush generally produce printed output which exactly matches screen display, while others will resize the graphic according to the printer resolution. In the latter types, you may need to use scaling percentages, which measure how much larger or smaller the graphic becomes to match the original code size generated by Thunder Bars. Examine the Drawing Area dialog to determine the exact size needed. Consult the documentation for your particular application to determine the appropriate settings, or methods to disable any automatic scaling or re-sizing.

Hardware Requirements

Thunder Bars system requirements & recommendations include:

Any IBM Compatible machine with an 80286 processor or higher.
2 megabytes of memory.
Microsoft Windows 3.1 or later.
A Microsoft compatible mouse.
A 5.25" or 3.5" high density floppy drive.
A hard disk with at least 2 megabytes free.
A VGA, 8514 or compatible display highly recommended.

Bar code ratio is the ratio of a wide element compared to a narrow element (bar or space); thus a 3 to 1 ratio means a wide bar is 3 times the width of a narrow bar. Note!, some symbologies do not allow variable ratios.

A code format which allows reading in either direction across the bars and spaces.

The error rate of a bar code can be dramatically reduced through the use of check characters. A check character (also referred to as Checksum or Check Digit) is an additional character that is mathematically based on the other characters in the encoded message. The bar code reader confirms that the check character is consistent with the data characters before accepting the data. Note!, some symbologies require that a check character be encoded, while with others the use of 1 or more check character methods are optional.

A discrete symbology is one in which an inter-character space is encoded as part of it's structure, which can serve as a self-checking mechanism. A continuous symbology does not encode any additional space between characters, thus the characters are continuous.

The narrowest element or module (bar or space) in a code. Sometimes referred to as the "Nominal Dimension".

Start/Stop bars are the special characters at the first and last positions in the code that identify the beginning and ending of the symbol. Start/Stop bars (sometimes called alignment bars) are required and the character(s) used vary depending on the symbology.

The quiet zone (sometimes referred to as Margin) is an area of white space immediately to the left and right of the outer bars that allows the reading device to distinguish the bars from other printed material.

Codabar

Codabar is commonly used in libraries, blood banks, parcel applications, inventory control and photo finishing. It is a discrete, variable length, self checking, bi-directional numeric symbology. Every character has four bars and three spaces. The characters A, B, C & D are used exclusively as start/stop alignment characters. The code has an inter-character space which is equal in width to a narrow element, that serves as a self checking mechanism. Optionally a modulus 16 check digit may be generated to increase data integrity.

Codabar characteristics:

Character Set:
10 digits
6 special characters (-$:/,+)

Input String & Encoded Data
Thunder Bars automatically generates the start/stop characters or they can be a part of the input string. If no start/stop characters are input, A will be assumed. Any alpha data not in the first or last position of the input string will be discarded.

Symbol Length:
Variable

Check Characters:
Modulo 16 (Optional)

Symbol Breakdown

1> Leading quiet zone
2> Start character
3> Data characters
4> Optional check character
5> Stop character
6> Trailing quiet zone

Code 39

Code 39 was the first alphanumeric symbology developed, and is the standard for non-retail, automotive, general industrial, and many government applications. It is a discrete, variable length, self checking, bi-directional alphanumeric symbology. Every character has five bars and four spaces; three of the elements in any given character are wide, and six are narrow. The asterisk character (*) is used exclusively as a start/stop alignment character. The code has an inter-character space which is equal in width to a narrow element, that serves as a self checking mechanism. Optionally a modulus 43 check digit may be generated to increase data integrity.

Code 39 characteristics:

Character Set:
26 uppercase letters
10 digits
7 special characters (-. $/+%)

Input String & Encoded Data
Thunder Bars automatically generates the start/stop characters and discards any invalid data in the input string.

Symbol Length:
Variable

Check Characters:
Modulo 43 (Optional)

Symbol Breakdown

1> Leading quiet zone
2> Start character
3> Data characters
4> Optional check character
5> Stop character
6> Trailing quiet zone

Code 128 B

Code 128 B is a continuous, variable length, bi-directional alphanumeric symbology. Applications include inventory control and general industrial. Every character has eleven modules containing three bars and three spaces. A start pattern of "B" is used to select the subset (character set). A modulus 103 check digit is generated to increase data integrity.

Code 128 B characteristics:

Character Set:
96 unique characters including:
The letters A-Z (uppercase)
The digits 0-9
The letters a-z (lowercase)
34 additional characters including space, !, ", #, $, %, &, ', (,), *, +, comma, -, ., /, :, ;, <, =, >, ?, @, [, \,], ^, _,null, {, |, }, ~, del

Input String & Encoded Data
Thunder Bars automatically generates the start/stop characters and discards any invalid data in the input string.

Symbol Length:
Variable

Check Characters:
Modulo 103

Symbol Breakdown

1> Leading quiet zone
2> Start character
3> Data characters
4> Check character
5> Stop character
6> Trailing quiet zone

Interleaved 2 of 5

Interleaved 2 of 5 was one of the earliest symbologies to become widely available, and is often used in industrial, warehousing, automotive and container identification applications. It is a continuous, self checking, bi-directional numeric symbology. Every character has five bars and five spaces; two of the elements in any given character are wide, and three are narrow. The 2 of 5 uses special patterns as start/stop alignment characters. An Interleaved 2 of 5 can only encode data containing an even number of digits. Optionally a modulus 10 check digit may be generated to increase data integrity.

Interleaved 2 of 5 characteristics:

Character Set:
10 digits

Input String & Encoded Data
Thunder Bars automatically generates the start/stop characters and discards any invalid data in the input string. If an odd number of characters are input, a leading zero will automatically be added.

Symbol Length:
Variable

Check Characters:
Modulo 10 (Optional)

Symbol Breakdown

1> Leading quiet zone
2> Start pattern
3> Data characters
4> Optional check character
5> Stop pattern
6> Trailing quiet zone

UPC-A

UPC (Universal Product Code) is the standard for the retail and supermarket industrys. It is a continuous, fixed length, bi-directional numeric symbology. Each digit is encoded as two bars and two spaces, within seven modules. The symbol is seperated into two halfs divided by two center guard bars. The halves are enclosed by two left guard bars and two right guard bars. The guard bars are similiar in function to start/stop alignment characters. The data encoded is used to uniquely identify a product and its manufacturer. A check digit is generated to increase data integrity.

UPC characteristics:

Character Set:
10 digits

Input String & Encoded Data
Thunder Bars automatically generates the guard bars and check character, and discards any invalid or extra data in the input string.
Only the first eleven digits of the input string will be encoded, if the input string is less than eleven digits, Thunder Bars will include the required leading zeros.

Symbol Length:
Fixed

Check Characters:
One

Symbol Breakdown

1> Leading quiet zone
2> Left guard bars
3> Number system character
4> Center guard bars
5> Product number
6> Center guard bars
7> Manufacturer number
8> Check Character
9> Right guard bars
10> Trailing quiet zone

Printing Guidelines

- Avoid photocopying codes.
- Voids (white areas within the bars) can cause the code to be misread.
- Avoid printing codes on glossy paper. Glare can cause the code to be misread.
- The ink used to print the black bars should be carbon based black or equivalent. The printed bars should reflect less than 20% of the infrared light source.
- Avoid background patterns or other printed matter within the code clear zone.
- The bond paper that forms the spaces should be white or a light pastel color that reflects at least 65% of the infrared light source.
- Excessive or extraneous inking should not cause any bar to exceed the recommended height or width limits.

Glossary

Aspect Ratio
The horizontal to vertical ratio of the screen display's resolution.

Bar Code
A bar code symbol is a pattern of parallel, adjacent bars and spaces. The width's of the bars and spaces represent the actual data in the symbol.

Bi-directional
A code format which allows reading in either direction across the bars and spaces.

BRM Bars
Business Reply Mail Bars. BRM Bars are a series of horizontal bars placed to the right of the delivery address line and directly below the "NO POSTAGE" endorsement box on a mailpiece.

Character Set
A group of letters, numbers, and symbols that have some relationship in common. For example, the ASCII character set contains characters that make up the ASCII coding scheme. Most bar codes have a limited character set.

Check Sum
A check character (also referred to as Checksum or Check Digit) is an additional character that is mathematically based on the other characters in the encoded message.

Client
An application whose documents can accept linked or embedded objects.

Continuous
A symbology which does not encode any additional space between characters, thus the characters are continuous.

DDE
Dynamic Data Exchange - an established protocol for exchanging data through active links between applications that run under Microsoft Windows.

Discrete
A symbology in which an inter-character space is encoded as part of it's structure, which can serve as a self-checking mechanism.

Element
The narrowest element or module (bar or space) in a code. Sometimes referred to as the "Nominal Dimension".

FIM
Facing Identification Marks. FIM is another type of postal barcode used in automated processing.

Human Readable
Printed message, usually representing the encoded data, which appears directly beneath the bar code.

Imaging
The capture, storage, retrieval, and manipulation of electronic images of documents.

Input String
The actual data input, irregardless of what is actually encoded.

Interface
The menu bar, command buttons, data areas and dialogs, which form the common boundary between the user the and internal operation of the program.

Ladder
Vertical orientation is sometimes referred to as "Ladder", and is a 90° clockwise rotation.

Link
To create a reference in a client application to an object in a server application. When you link a object, you are inserting a visual presentation of the object into the client application. The linked object can be edited directly from within the client application. When the object changes in the server application, the changes appear in the client application.

Macro
A series of recorded actions, which can be later played back to carry out all the recorded actions.

Module
The narrowest element or module (bar or space) in a code. Sometimes referred to as the "Nominal Dimension".

OCR
Optical Character Recognition.

Patch Code
A patch code is a pattern of parallel, alternating black bars and spaces that is printed on a document to be used in an Imaging application.

Picket
"Picket Fence" refers to a horizontal, 0° non-rotated code.

Pixel
The smallest graphic unit that can be displayed on the screen, usually a single colored dot. Also known as Pel, the abbreviation for picture element.

Postnet
POSTal Numeric Encoding Technique. The POSTNET bar code was developed by the Postal Service to provide a system of encoding ZIP Code information on letter mail.

Quiet Zone
The quiet zone (sometimes referred to as Margin) is an area of white space immediately to the left and right of the outer bars that allows the reading device to distinguish the bars from other printed material.

Ratio
The ratio of a wide element compared to a narrow element (bar or space); thus a 3 to 1 ratio means a wide bar is 3 times the width of a narrow bar.

Server
A Windows application that creates objects that can be linked or embedded into other documents.

Start/Stop
Start/Stop bars are the special characters at the first and last positions in a code that identify the beginning and ending of the symbol.

Void
Light areas in the bars of a symbol which are usually caused by printing errors.

Thunder Bars Source Code

Visual Basic source code is available for all of the routines used to generate the various codes in the Thunder Bars main program. Thunder Bars Source Code is available bundled with Thunder Bars, or can be purchased at a special price to registered Thunder Bars owners at a later time. If you did not purchase the bundled package and wish to order Thunder Bars Source Code, contact Thunder Island, or your local sales office.

Contents

Thunder Bars Source Files

TBARDEMO.BAS, TUTOR.BAS
Demo VB Basic module files, add one of these files to your VB project.

TBARDEMO.EXE, TUTORBAR.EXE
Demo executable programs, use these to experiment and test.

TBARDEMO.FRM, FORM1.FRM
Demo VB form files.

TBARDEMO.MAK, TUTORBAR.MAK
Demo VB project files.

Note, all demo files were saved with version 3.0, in an ASCII format, provided for reference, and as an aid for importing routines into other languages, such as Microsoft Access Basic, Visual Basic for DOS, etc.

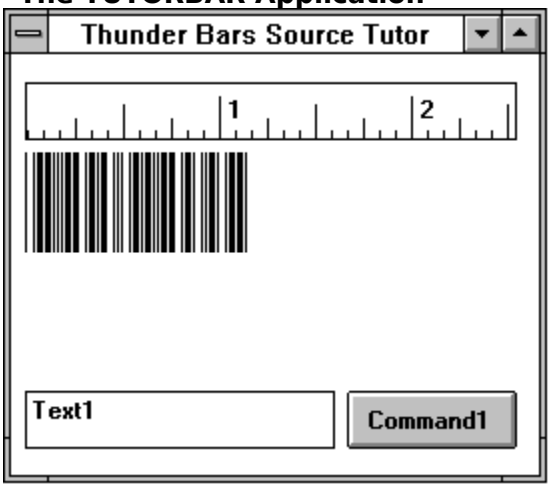
How to include THUNDER BARS routines in your project

To add the Thunder Bars code routines to your program, first make a copy of TBARDEMO.BAS called THUNDER.BAS (or a filename you prefer). For example, to make a copy using the DOS COPY program, at the DOS prompt type; COPY TBARDEMO.BAS THUNDER.BAS. Using a copy is strongly recommended as TBARDEMO.BAS should remain intact so that you can refer to it if you need the original unmodified code. Next, start Visual Basic and load your project file, or choose New Project from the File menu. Select the Add File command from the File menu. Then choose THUNDER.BAS in the Add File Dialog. All of the routines in THUNDER.BAS are now available from any form or module in your project.

Demo Applications

Thunder Bars Source includes source code for two fully functional applications. These demo applications demonstrate basic Thunder Bars programming methods, and introduce you to more advanced techniques for integrating code support into your own applications.

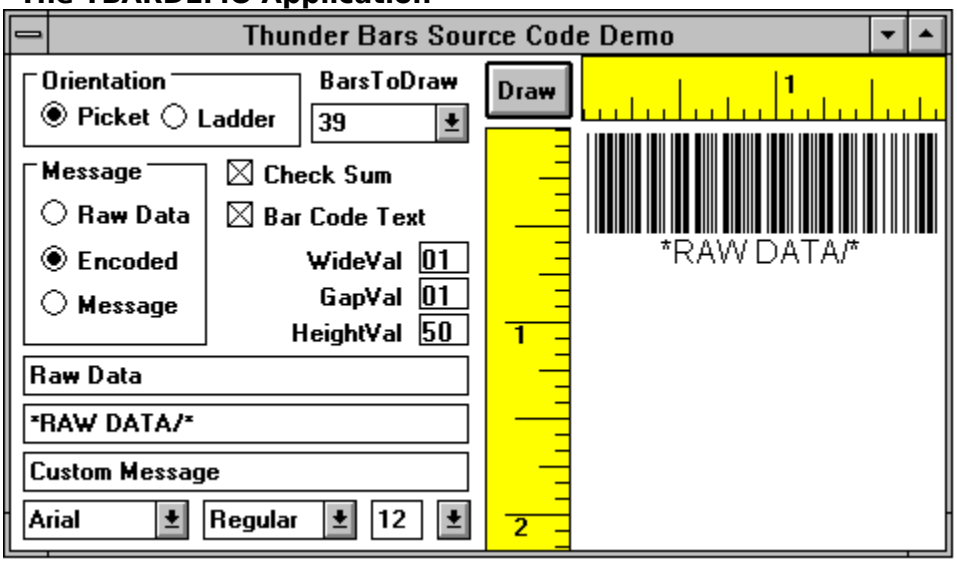
The TUTORBAR Application



The TUTORBAR application draws a horizontal ruler on startup, and encodes the contents of the text box as a horizontal 3 of 9 bar code when the command button is clicked. This application provides a good starting point for adding Thunder Bars routines to your own program.

It takes just a few minutes to re-create the TUTORBAR application. First you add the TUTOR.BAS file to your project. You then create the user interface by drawing the necessary controls, such as the picture boxes, text box and command button on the form. Next you set properties for the form and controls to specify such values as scale modes, control names and indexes. Finally, you write code to generate the ruler and the bar code.

The TBARDEMO Application



The TBARDEMO application draws horizontal & vertical rulers, encodes all of the code types

available in Thunder Bars, and demonstrates a sample user interface to provide for control over orientation, checksum, fonts and code dimensions. This demo application picks up where the TUTORBAR demo leaves off. TBARDEMO introduces the remaining Thunder Bars subroutines, variables and other Visual Basic functions.

Bar Code Programming

Variables

BCWideBarSize
Width of the bar code wide bars

BCNarrowBarSize
Width of the bar code narrow bars

BCNarrowSpaceSize
Width of the bar code narrow spaces

BCWideSpaceSize
Width of the bar code wide spaces

BCRatio
Wide to narrow ratio

BCCheckSum
Whether or not to calculate the bar code check sum

BCVerData
Actual encoded data

BCHeight
Bar code height

BCDoText
Whether or not to print the human readable

Primary Routines
These are the main bar code drawing routines you should call and pass the data for what you want to encode.

Action	Draws a Bar Code in a picture box.
Syntax1	DrawHorz128 (index, BCData$) DrawHorz25 (index, BCData$) DrawHorz39 (index, BCData$) DrawHorzCodabar (index, BCData$) DrawHorzUPCA (index, BCData$)
Syntax2	DrawVert128 (index, BCData$) DrawVert25 (index, BCData$) DrawVert39 (index, BCData$) DrawVertCodabar (index, BCData$) DrawVertUPCA (index, BCData$)
Remarks	Both arguments are REQUIRED
Argument	Description
index	The Index property of the picture box to contain the code.

BCData$ The input string to encode.

example:) DrawHorz39 0, "Data" will draw a horizontal 3 of 9 bar code on the control named Thunder(0) encoding the string "Data".

Secondary Routines
These routines are called by the bar code primary routines.

BCHNB ()
Draws a Bar Code Horizontal Narrow Bar

BCHNS ()
Draws a Bar Code Horizontal Narrow Space

BCHWB ()
Draws a Bar Code Horizontal Wide Bar

BCHWS ()
Draws a Bar Code Horizontal Wide Space

BCVNB ()
Draws a Bar Code Vertical Narrow Bar

BCVNS ()
Draws a Bar Code Vertical Narrow Space

BCVWB ()
Draws a Bar Code Vertical Wide Bar

BCVWS ()
Draws a Bar Code Vertical Wide Space

HumanReadable ()
Prints the bar code message

Mod128B (ModData$)
Checksum routine for Code 128-B

ModCodabar (ModData$)
Checksum routine for Codabar

ModCode39 (ModData$)
Checksum routine for Code 39

ModInt25 (ModData$)
Checksum routine for Interleaved 2 of 5

ModUPCA (ModData$)
Checksum routine for UPC-A

Other Considerations

PictureBox CtlName MUST be "Thunder" unless you modify thunder.bas.

PictureBox MUST have an Index value unless you modify thunder.bas.

ScaleMode property is automatically set to Pixels and reset after the primary routine. Other ScaleModes can result in an "Overflow" error.

If the form which contains the bar code(s) is NOT named Form1, you will need to modify thunder.bas.

You can add thunder.bas to your project and Visual Basic will find the bar code routines. Optionally you can add DrawHorz(bar type) & DrawVert(bar type) as subroutines in the general declarations section of your form.

BRM Programming

Variables

BRMBarSize
Width of a BRM bar

BRMSpaceSize
Width of a BRM space

BRMLength
BRM Length

Primary Routines
These are the main BRM drawing routines you should call and pass the type for what you want to encode.

Action Draws a BRM Code in a picture box.

Syntax1 DrawHorzBRM (index, BRMVal)
Syntax2 DrawVertBRM (index, BRMVal)

Remarks Both arguments are REQUIRED

Argument Description

index The Index property of the picture box to contain the code.

BRMVal The BRM Type to encode.

example:) DrawVertBRM 0, 6 will draw 6 vertical BRM bars on the control named Thunder(0).

Secondary Routines
These routines are called by the BRM primary routines.

BRMHBar ()
Draws a BRM Horizontal Bar

BRMHSpace ()
Draws a BRM Horizontal Space

BRMVBar ()
Draws a BRM Vertical Bar

BRMVSpace ()
Draws a BRM Vertical Space

Other Considerations

PictureBox CtlName MUST be "Thunder" unless you modify thunder.bas.

PictureBox MUST have an Index value unless you modify thunder.bas.

ScaleMode property is automatically set to Pixels and reset after the primary routine.

Other ScaleModes can result in an "Overflow" error.

If the form which contains the BRM code(s) is NOT named Form1, you will need to modify thunder.bas.

You can add thunder.bas to your project and Visual Basic will find the BRM routines. Optionally you can add DrawHorzBRM & DrawVertBRM as subroutines in the general declarations section of your form.

FIM Programming

Variables

FIMBarSize
Width of a FIM bar

FIMSpaceSize
Width of a FIM space

FIMHeight
FIM height

Primary Routines
These are the FIM drawing routines you should call and pass the type for what you want to encode.

Action Draws a FIM Code in a picture box.

Syntax1 DrawHorzFim (index, FimType$)
Syntax2 DrawVertFim (index, FimType$)

Remarks Both arguments are REQUIRED

Argument Description

index The Index property of the picture box to contain the code.

FimType$ The FIM Type to encode.

example:) DrawHorzFim 0, C will draw a horizontal Fim type "C" code on the control named Thunder(0).

Secondary Routines
These routines are called by the FIM primary routines.

FHBar ()
Draws a FIM Horizontal Bar

FHSpace()
Draws a FIM Horizontal Space

FVBar ()
Draws a Fim Vertical Bar

FVSpace ()
Draws a FIM Vertical Space

Other Considerations

PictureBox CtlName MUST be "Thunder" unless you modify thunder.bas.

PictureBox MUST have an Index value unless you modify thunder.bas.

ScaleMode property is automatically set to Pixels and reset after the primary routine.

Other ScaleModes can result in an "Overflow" error.

If the form which contains the FIM code(s) is NOT named Form1, you will need to modify thunder.bas.

You can add thunder.bas to your project and Visual Basic will find the FIM code routines. Optionally you can add DrawHorzFIM & DrawVertFIM as subroutines in the general declarations section of your form.

Patch Code Programming

Variables

PatchWideBarSize
Width of the patch code wide bars

PatchNarrowBarSize
Width of the patch code narrow bars

PatchSpaceSize
Width of the patch code spaces

PatchRatio
Wide to narrow ratio

PatchWidth
Overall patch code width(length)

Primary Routines
These are the patch code drawing routines you should call and pass the type for what you want to encode.

Action Draws a Patch Code in a picture box.

Syntax1 DrawHorzPatch (index, PType$)
Syntax2 DrawVertPatch (index, PType$)

Remarks Both arguments are REQUIRED

Argument Description

index The Index property of the picture box to contain the code.

PType The Patch Type to encode.

example:) DrawVertPatch 0, T will draw a vertical patch code type "T" on the control named Thunder(0).

Secondary Routines
These routines are called by the patch code primary routines.

PHNarrowBar ()
Draws a Patch Horizontal Narrow Bar

PHWideBar ()
Draws a Patch Horizontal Wide Bar

PVNarrowBar ()
Draws a Patch Vertical Narrow Bar

PVWideBar ()
Draws a Patch Vertical Wide Bar

Other Considerations

PictureBox CtlName MUST be "Thunder" unless you modify thunder.bas.

PictureBox MUST have an Index value unless you modify thunder.bas.

ScaleMode property is automatically set to Pixels and reset after the primary routine. Other ScaleModes can result in an "Overflow" error.

If the form which contains the patch code(s) is NOT named Form1, you will need to modify thunder.bas.

You can add thunder.bas to your project and Visual Basic will find the patch code routines. Optionally you can add DrawHorzPatch & DrawVertPatch as subroutines in the general declarations section of your form.

Postnet Programming

Variables

ZipBarSize
Width of a Zip+4 bar

ZipSpaceSize
Width of a Zip+4 space

ZipHeight
Height of the ZIP+4 code

Primary Routines
These are the postnet drawing routines you should call and pass the data for what you want to encode.

Action Draws a Postnet Code in a picture box.

Syntax1 DrawHorzZip (index, ZipData$)
Syntax2 DrawVertZip (index, ZipData$)

Remarks Both arguments are REQUIRED

Argument Description

index The Index property of the picture box to contain the code.

ZipData The ZIP Code input string to encode.

example:) DrawHorzZip 1, 54321 will draw a horizontal postnet code on the control named Thunder(1) encoding the string "54321".

Secondary Routines
These routines are called by the postnet primary routines.

ZHShortBar ()
Draws a ZIP+4 Horizontal Short Bar

ZHTallBar ()
Draws a ZIP+4 Horizontal Tall Bar

ZVShortBar ()
Draws a ZIP+4 Vertical Short Bar

ZVTallBar ()
Draws a ZIP+4 Vertical Tall Bar

Other Considerations

PictureBox CtlName MUST be "Thunder" unless you modify thunder.bas.

PictureBox MUST have an Index value unless you modify thunder.bas.

ScaleMode property is automatically set to Pixels and reset after the primary routine.

Other ScaleModes can result in an "Overflow" error.

If the form which contains the postnet code(s) is NOT named Form1, you will need to modify thunder.bas.

You can add thunder.bas to your project and Visual Basic will find the postnet code routines. Optionally you can add DrawHorzZip & DrawVertZip as subroutines in the general declarations section of your form.

Ruler Programming

Primary Routines

These are the ruler drawing routines you should call to generate a ruler.

Action Draws a Ruler in a picture box.

Syntax1 DrawHorizontalRuler (mRuleScale, index, mZoomVal)

Syntax2 DrawVerticalRuler (mRuleScale, index, mZoomVal)

Remarks All three arguments are REQUIRED

Argument	Description
mRuleScale	Value for the Scale of the ruler. ex.) 16 = 16'ths, 8 = 8'ths, etc.
index	The Index property of the picture box to contain the ruler.
mZoomVal	Value for the amount of Zoom to use. ex.) 1 = real size, .5 = 50%, 2 = 200%, etc.

example:) DrawHorizontalRuler 12, 0, 1 will draw a horizontal ruler, with 1/12 increments on the control named ruler(0) with a Zoom value of 1(real size).

Other Considerations

PictureBox CtlName MUST be "ruler" unless you modify thunder.bas.

PictureBox MUST have an Index value unless you modify thunder.bas.

Your form should have a ScaleMode property of 5 (inches), or 7 (centimeters). Other ScaleModes will work, but the ruler marks may be unreadable and will take a long time to draw. Note! The ScaleMode of the form AND the ruler will affect the drawing of the ruler, a mode of twips for example, will result in an "Overflow" error.

If the form which contains the ruler(s) is NOT named Form1, you will need to modify thunder.bas.

You can add thunder.bas to your project and Visual Basic will find the ruler routines. Optionally you can add DrawHorizontalRuler & DrawVerticalRuler as subroutines in the general declarations section of your form.

There are no minimum or maximum ranges for mRuleScale or mZoomVal, however very large or very small values will produce a sloppy looking ruler. This also applies to unusually large or small Font Size, Draw Width, and ScaleModes, etc.

About Visual Basic picture boxes

A picture box can display a graphic from a bitmap, icon, or metafile. It displays only as much of the graphic as fits into the rectangle you've drawn with the picture box tool.

To make the picture box automatically resize to display the whole graphic, set the AutoSize property to True. To create animation or simulation, you can manipulate the graphics properties and methods in your code. (Graphics properties and methods are indicated in the list below by an asterisk (*) to the right of the name.) These properties and events are also useful for run-time print operations, such as modifying the format of a screen form for printing.

Properties that have an effect on the appearance & functionality of the various codes and rulers.

*AutoRedraw	BackColor
BorderStyle	CtlName
*DrawMode	*DrawStyle
*DrawWidth	*FontBold
*FontItalic	*FontName
*FontSize	*FontStrikethru
*FontTransparent	*FontUnderline
*ForeColor	Height
Index	Picture
*ScaleMode	Width