# ZPL II Programming

## Guide

For x.10 through x.13 Firmware

Zebra

**Part Number: 45541L-004 Rev. A**

# Contents

**Notes •**

# Functional Table of Contents

**Notes •**

# About This Document

This section provides you with contact information, document structure and organization, and additional reference documents.

## Contents

# Who Should Use This Document

This Guide is for programmers who are familiar working with programming languages.

# How This Document Is Organized

The Guide is set up as follows:

| Section | Description |
|---|---|
| *Introduction* | Provides a high-level overview about this guide and Zebra Programming Language (ZPL). |
| *ZPL Commands* | Provides an alphabetical, detailed description of each ZPL command. |
| *RFID Commands* | Provides an alphabetical, detailed description of each ZPL RFID command, and some examples of how to use them. |
| *Wireless Commands* | Provides new and modified ZPL commands for the wireless print server. |
| Appendixes | The appendixes include:<br>*Zebra Code Pages*<br>*ASCII*<br>*Fonts and Bar Codes*<br>*Mod 10 and Mod 43 Check Digits*<br>*ZB64 Encoding and Compression*<br>*Field Interactions* |
| *Glossary* | Glossary of terms. |

# Contacts

You can contact Zebra Technologies at the following:

## Web Site

http://www.zebra.com

Technical Support via the Internet is available 24 hours per day, 365 days per year. Go to http://www.zebra.com/support.

## The Americas

| Regional Headquarters | Technical Support | Customer Service Dept. |
|---|---|---|
| Zebra Technologies International, LLC<br>333 Corporate Woods Parkway<br>Vernon Hills, Illinois 60061.3109<br>U.S.A<br>T: +1 847 793 2600<br>Toll-free +1 800 423 0422<br>F: +1 847 913 8766 | T: +1 847 913 2259<br>F: +1 847 913 2578<br>Hardware: hwtsamerica@zebra.com<br>Software: swtsamerica@zebra.com | For printers, parts, media, and ribbon, please call your distributor, or contact us.<br>T: +1 866 230 9494<br>E: VHCustServ@zebra.com |

## Europe, Africa, Middle East, and India

| Regional Headquarters | Technical Support | Internal Sales Dept. |
|---|---|---|
| Zebra Technologies Europe Limited<br>Zebra House<br>The Valley Centre, Gordon Road<br>High Wycombe<br>Buckinghamshire HP13 6EQ, UK<br>T: +44 (0)1494 472872<br>F: +44 (0) 1494 450103 | T: +44 (0) 1494 768298<br>F: +44 (0) 1494 768210<br>Germany: Tsgermany@zebra.com<br>France: Tsfrance@zebra.com<br>Spain/Portugal: Tsspain@zebra.com<br>All other areas: Tseurope@zebra.com | For printers, parts, media, and ribbon, please call your distributor, or contact us.<br>T: +44 (0) 1494 768316<br>F: +44 (0) 1494 768244<br>E: mseurope@zebra.com |

## Asia Pacific

| Regional Headquarters | Technical Support | Customer Service |
|---|---|---|
| Zebra Technologies Asia Pacific, LLC<br>16 New Industrial Road<br>#05-03 Hudson TechnoCentre<br>Singapore 536204<br>T: +65 6858 0722<br>F: +65 6885 0838 | T: +65 6858 0722<br>F: +65 6885 0838<br>E: tsasiapacific@zebra.com | For printers, parts, media, and ribbon, please call your distributor, or contact us.<br>T: +65 6858 0722<br>F: +65 6885 0837 |

# Document Conventions

The following conventions are used throughout this document to convey certain information.

**Alternate Color** (online only) Cross-references contain hot links to other sections in this guide. If you are viewing this guide online in .pdf format, you can click the cross-reference (blue text) to jump directly to its location.

**Command Line Examples** Command line examples appear in `Courier New` font. For example, type `ZTools` to get to the Post-Install scripts in the `bin` directory.

**Files and Directories** File names and directories appear in `Courier New` font. For example, the `Zebra<version number>.tar` file and the `/root` directory.

**Icons Used**

**Important •** Advises you of information that is essential to complete a task.

**Note •** Indicates neutral or positive information that emphasizes or supplements important points of the main text.

**Example •** Provides an example, often a scenario, to better clarify a section of text.

# 1

# Introduction

This guide is the unabridged, alphabetical reference of programming commands supported in the firmware.

**Firmware** You can get the printer's firmware version by printing out a configuration label. For instructions to do so, see your printer's user guide.

**Note •** Firmware upgrades are available at: www.zebra.com.

If you are using a previous version of Zebra printer firmware, some of the commands are the same and function as they did before— but equally as many are new and are not recognized by firmware earlier than X.10.

Many word processors or a text editor capable of creating ASCII files can be used to recreate most examples in this guide. However, for other encodings such as Unicode, a text editor such as Microsoft Notepad is needed.

If there are any terms used in this guide that you need clarification on, please see the *Glossary* on page 379.

**Notes •**

# 2

# ZPL Commands

This section contains the complete alphabetical listing of ZPL II commands.

**Description** This heading provides an explanation of how the command is used, what it is capable of, and any defining characteristics it has.

**Format** Format explains how the command is syntactically arranged and what parameters it contains.

**For Example** The `^B8` command prints a EAN-8 bar code. The format of the `^B8` command is: `^B8o,h,f,g`. It is arranged with the caret symbol (`^`), the command code (`B8`), and the parameters and are replaced with supported values.

**Parameters** If a command has values that can be defined to make its function more specific, these are outlined as parameters. Parameters typically have *Accepted Values* and *Default Values*.

Still using the `^B8` example, the `h` parameter is defined as:

`h` = bar code height (in dots)

*Accepted Values:* `1 to 32000`

*Default Value:* value set by `^BY`

If the command has no parameters – for example `~JA` (Cancel All) – the parameter heading is removed, indicating that the format of the command (`~JA`) is acceptable ZPL II code.

**Example •** When the command is best clarified in context, an example of the ZPL II code is provided. Text indicating exact code entered is printed in an easily recognizable Courier font. An example of code using the `^B8` command looks like this:

```
^XA
^FO50,50
^B8N,100,Y,N
^FD1234567^FS
^XZ
```

Notice that the `^B8` parameter letters have been replaced with real values that apply to the command. In this example `N,100,Y,N` have been entered.

**Comment** This section is reserved for notes that are of value to a programmer, warnings of potential command interactions, or command-specific information that should be taken into consideration.

**Example •** An example comment is: **This command works only when the printer is idle,** or **This command is ignored if a value exceeds the parameter limits**.

Comments are also included next to parameters if they apply directly to a particular setting.

# Basic ZPL Exercises and Examples

The purpose of these exercises is to introduce basic ZPL commands to novice ZPL users.

### Make sure this checklist is complete:

- ❑ Load the printer with labels that are big enough to give you ample space to work with.
- ❑ Print a configuration label (CANCEL test).
- ❑ Look at the configuration label and make sure that the `LEFT POSITION` is set to `000` and `LABEL TOP` is set to `000`.
- ❑ Determine the printer's resolution. It is listed on the configuration label. 8/MM `=` 200 dpi, `12/MM = ` 300 dpi and `24/MM = ` 600 dpi.

#### Tips

These are some tips when using ZPL:

- Use the DOS text editor to write ZPL files.
- Save the file as a `.txt` file and copy it to the printer from DOS command line.

## Before You Begin

Some things that are important to understand before you begin are:

- 200 dpi means the resolution of the printhead is 200 dots per inch. If you program the printer to draw a line 100 dots long that equals a half inch. 100 dots on a 300 dpi printer prints a line 1/3 inch long.
- The home position that all your coordinates are referencing is at the left-hand trailing edge of the label as the label comes out of the printer. (There are some exceptions to this.)

### Exercises

The exercises start simple and gradually progress to give you an opportunity to try a variety of commonly used ZPL commands. Not all commands are covered, but this should be a good core of commands to learn. Some commands may not be supported due to the firmware version in your printer.

### Exercise 1 • This exercise shows you how to specify a location for an entered name.

1. Print your name on the label.
2. Start by printing just your name on the label. Use this format as a model:

**Important •** Your name goes where you see **`xxxxxxxxxxx`** in the second line of code.

3. Send this format to the printer:

```
1  ^XA
2  ^FO50,50^ADN,36,20^FDxxxxxxxxxxx
3  ^FS
4  ^XZ
                          5
```

| | |
|---|---|
| **1** | Every format starts with the `^XA` command |
| **2** | `^FO` (field origin) command |
| **3** | `^FS` (field separator) command |
| **4** | Every format ends with the `^XZ` command |
| **5** | `^FD` (field data) command |

4. When the label prints correctly, alter the first number after the `^FOx`. See how that change affects the print position. Alter the second number after the `^FO50,x` and see how that the print position.

## Font instruction

`^ADN`

1. Alter the numbers after the `^ADN,x,x` command.
   - 18,10 is the smallest size you can make the **`D`** font.
   - The first number is the height of the font in dots. The second number is the width in dots.
   - You can use direct multiples up to ten times that size as a maximum.

   **Example •** 180,100 is the largest you can make the **`D`** font.

   - 25,18 would not be a valid size. The printer rounds to the next recognizable size.
2. Check the font matrices tables for other fonts to try. See *Fonts and Bar Codes* on page 351.
3. Try the zero scalable font `^A0N,x,x`.

   This font is scalable, and you can choose any height and width.

### Rotation commands

1. Change `^ADN` to `^ADR`, and then `^ADI`, and then `^ADB`.

   See how the print position changes.

2. Add more fields.

3. Add two more fields to print directly under your name using the `^ADN,36,20` font and size:

   Your street address

   Your city, state, zip

4. You must add two more lines of code that start off with:

```
^XA
^FO50,50^ADN,36,20^FDxxxxxxxxxxx^FS
^FO    (fill in the rest)
^FO    (fill in the rest)
^XZ
```

Make sure all these fields print in the same font and size and left side of fields has same vertical alignment.

Your name

1200 W Main Street

Anytown, Il 60061

## Exercise 2 • Boxes and lines

1. Use the address format from **Exercise 1**.

2. Add this new line to your existing format:

   `^FO50,200^GB200,200,2^FS`

   This prints a box one wide by one inch long and the thickness of the line is 2 dots.

3. Reposition and resize the square so that it goes around the name and address uniformly.

4. Print a line by adding:

   `^FO50,300^GB400,0,4,^FS`

   This prints a horizontal line two inches wide by 4 dots thick.

5. Print a vertical line using this code:

   `^F0100,50^GBO,400,4^FS`

## Exercise 3 • Bar codes — ^B3 code 39 bar code

1. Write the following format and send to the printer:

   ```
   ^XA
   ^FO50,50^B3N,N,100,Y,N^FD123456^FS
   ^XZ
   ```

2. Try changing each of the parameters in the ^B3 string so you can see the effects.

**Important** • For valid parameter choices, see ^B3 on page 22.

   ```
   ^B3o,e,h,f,g
   ^BY
   ```

3. Insert the ^BY command just before the ^B3 to see how the narrow bar width can be altered.

   `^FO50,50^BY2^B3..etc ^BYx, acceptable values for x are 1 through 10`

4. Alter the ratio of the narrow to wide bar.

   `^FO50,50^BY2,3^B3..etc ^BY2,x acceptable values for x are 2.1 through 3 in .1 increments`

5. Print out a ^B3 bar code with the interpretation line on top of the bar code and the bar code rotated 90 degrees.

6. Add a ^PQ just before the ^XZ to print several labels.

   ```
   ^PQ4
   ^XZ
   ```

   ^PR Print rate (in inches per second)

7. Add a ^PR command after the ^XA at the beginning of the format to change the print rate (print speed).

   ```
   ^XA
   ^PR4 then try ^PR6 ^PRx acceptable values for x are 2 through 12 (check printer specs)
   ```

   See how the print speed affects the print quality of the bar code. You may need to increase the printer darkness setting at higher print speeds.

## Exercise 4 • ^SN — Serial Number command

1. Send this format to the printer:

```
^XA
^FO100,100^ADN,36,20^SN001,1,Y^FS
^PQ3
^XZ
```

To vary the `^SNv,n,z` to exercise increment/decrement and leading zeros functions, consult this guide.

If your serial number contains alpha and numeric characters, you can increment or decrement a specific segment of the data even if it is in the middle, as this sample sequence shows:

`ABCD1000EFGH, ABCD1001EFGH, ABCD1002EFGH`

2. Send this file to the printer and to see how it increments the serial number. The `^SF` command can also work with alpha characters.

```
^XA
^FO100,100^ADN,36,20^FDABCD1000EFGH^SF%%%%dddd%%%%,10000^FS
^PQ15
^XZ
```

Notice how the field data character position aligns with the `^SF` data string:

| ^ | F | D | A | B | C | D | 1 | 0 | 0 | 0 | E | F | G | H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ^ | S | F | % | % | % | % | d | d | d | d | % | % | % | % |
| | | | | | | | | | | 1 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | 2 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | 3 | 0 | 0 | 0 | 0 |

And on through…

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 1 | 0 | 1 | 4 | 0 | 0 | 0 | 0 |

The last label prints **ABCD1014EFGH**.

The % is placed in positions that you do not want to increment or decrement, `d = ` decimal, `10000 = ` increment value.

For more details on `^SF`, see *^SF* on page 254.

### Exercise 5 • Saving a template to memory. `^IS` and image save and image load.

**Note •** This exercise helps you troubleshoot your code against the errors you see on your labels.

1. Send this format to the printer:

```
^XA
^FO20,30^GB750,1100,4^FS
^FO20,30^GB750,200,4^FS
^FO20,30^GB750,400,4^FS
^FO20,30^GB750,700,4^FS
^FO20,226^GB325,204,4^FS
^FO30,40^ADN,36,20^FDShip to:^FS
^FO30,260^ADN,18,10^FDPart number #^FS
^FO360,260^ADN,18,10^FDDescription:^FS
^FO30,750^ADN,36,20^FDFrom:^FS
^ISR:SAMPLE.GRF^FS
^XZ
```

2. Send this format:

```
^XA
^ILR:SAMPLE.GRF^FS
^FO150,125^ADN,36,20^FDAcme Printing^FS
^FO60,330^ADN,36,20^FD14042^FS
^FO400,330^ADN,36,20^FDScrew^FS
^FO70,480^BY4^B3N,,200^FD12345678^FS
^FO150,800^ADN,36,20^FDMacks Fabricating^FS
^XZ
```

In this way the template only needs to be sent one time to the printer's memory. Subsequent formats can be sent recalling the template and merging variable data into the template. In this exercise, the file was saved in the printers `R:` memory, which is volatile.

## Exercise 6 • ^DF and ^XF — Download format and recall format

Similar concept to ^IS and ^IL command. ^IS and ^IL processes faster in the printer than ^DF and ^XF.

This is how the ^DF and ^XF format structure produces a label similar to the ^IS/^IL sample you just tried.

**Figure 1 • Download and Recall Format**

1

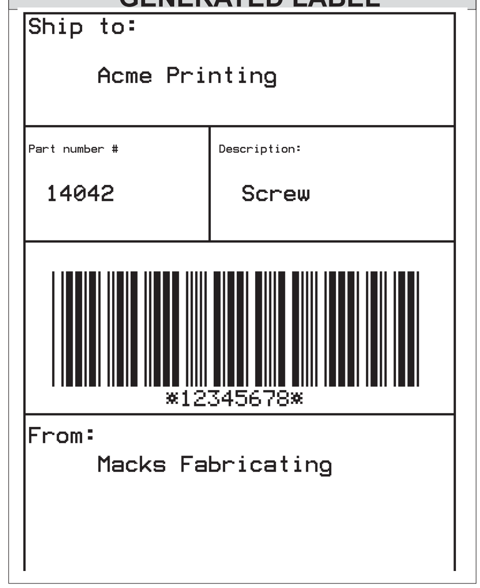
**ZPL II CODE**

```
^XA
^DFR:SAMPLE.GRF^FS
^FO20,30^GB750,1100,4^FS
^FO20,30^GB750,200,4^FS
^FO20,30^GB750,400,4^FS
^FO20,30^GB750,700,4^FS
^FO20,226^GB325,204,4^FS
^FO30,40^ADN,36,20^FDShip to:^FS
^FO30,260^ADN,18,10^FDPart number #^FS
^FO360,260^ADN,18,10^FDDescription:^FS
^FO30,750^ADN,36,20^FDFrom:^FS
^FO150,125^ADN,36,20^FN1^FS (ship to)
^FO60,330^ADN,36,20^FN2^FS(part num)
^FO400,330^ADN,36,20^FN3^FS(description)
^FO70,480^BY4^B3N,,200^FN4^FS(barcode)
^FO150,800^ADN,36,20^FN5^FS (from)
^XZ

^XA
^XFR:SAMPLE.GRF
^FN1^FDAcme Printing^FS
^FN2^FD14042^FS
^FN3^FDScrew^FS
^FN4^FD12345678^FS
^FN5^FDMacks Fabricating^FS
^XZ
```

2

**GENERATED LABEL**




| | |
|---|---|
| **1** | Download format code |
| **2** | Recall format call that generates the generated label in Figure 1. |

# ^A

## Scalable/Bitmapped Font

**Description** The ^A command specifies the font to use in a text field. ^A designates the font for the current ^FD statement or field. The font specified by ^A is used only once for that ^FD entry. If a value for ^A is not specified again, the default ^CF font is used for the next ^FD entry.

**Format** `^Afo,h,w`

**Important** • Parameter **f** is required. If **f** is omitted it defaults to the last value of the ^CF command.

This table identifies the parameters for this format:

| Command | Details |
|---|---|
| f = font name | *Accepted Values:* A through Z, and 0 to 9<br>Any font in the printer (downloaded, EPROM, stored fonts, fonts A through Z and 0 to 9). |
| o = field orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* the last accepted ^FW value or the ^FW default |
| h = Character Height (in dots) | **Scalable**<br>*Accepted Values:* 10 to 32000<br>*Default Value:* last accepted ^CF<br>**Bitmapped**<br>*Accepted Values:* multiples of height from 1 to 10 times the standard height, in increments of 1<br>*Default Value:* last accepted ^CF |
| w = width (in dots) | **Scalable**<br>*Accepted Values:* 10 to 32000<br>*Default Value:* last accepted ^CF<br>**Bitmapped**<br>*Accepted Values:* multiples of width from 1 to 10 times the standard width, in increments of 1<br>*Default Value:* last accepted ^CF |

## Scalable Font Command

**Example** • This is an example of a scalable font command:

**ZPL II CODE**

```
^XA
^FO50,50
^A0,32,25
^FDZEBRA^FS
^FO50,150
^A0,32,25
^FDPROGRAMMING^FS
^FO50,250
^A0,32,25^FDLANGUAGE^FS
^XZ
```

**GENERATED LABEL**

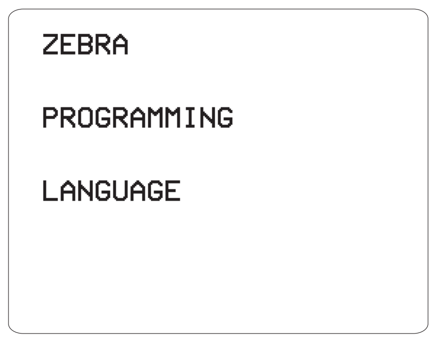

## Bitmap Font Command

**Example** • This is an example of a bitmap font command:

**ZPL II CODE**

```
^XA
^FO50,50
^ADN,36,20
^FDZEBRA^FS
^FO50,150
^ADN,36,20
^FDPROGRAMMING^FS
^FO50,250
^ADN,36,20^FDLANGUAGE^FS
^XZ
```

**GENERATED LABEL**




For reference, see *Standard Printer Fonts* on page 352**,** *Zebra Code Page 850* on page 342, *Fonts and Bar Codes* on page 351, and *ASCII* on page 347.

**Comments** Fonts are built using a matrix that defines standard height-to-width ratios. If you specify only the height or width value, the standard matrix for that font automatically determines the other value. If the value is not given or a 0 (zero) is entered, the height or width is determined by the standard font matrix.

# ^A@

## Use Font Name to Call Font

**Description** The ^A@ command uses the complete name of a font, rather than the character designation used in ^A. Once a value for ^A@ is defined, it represents that font until a new font name is specified by ^A@.

**Format** `^A@o,h,w,d:o.x`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = field orientation | *Accepted Values:*<br>N = normal<br>R = rotates 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* N or the last ^FW value |
| h = character height (in dots) | *Default Value:* specifies magnification by **w** (character width) or the last accepted ^CF value. Uses the base height if none is specified.<br>**Scalable** The value is the height in dots of the entire character block. Magnification factors are unnecessary, because characters are scaled.<br>**Bitmapped** The value is rounded to the nearest integer multiple of the font's base height, then divided by the font's base height to give a magnification nearest limit. |
| w = width (in dots) | *Default Value:* specifies magnification by **h** (height) or the last accepted ^CF value. Specifies the base width is used if none is specified.<br>**Scalable** The value is the width in dots of the entire character block. Magnification factors are unnecessary, because characters are scaled.<br>**Bitmapped** The value rounds to the nearest integer multiple of the font's base width, then divided by the font's base width to give a magnification nearest limit. |
| d = drive location of font | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |

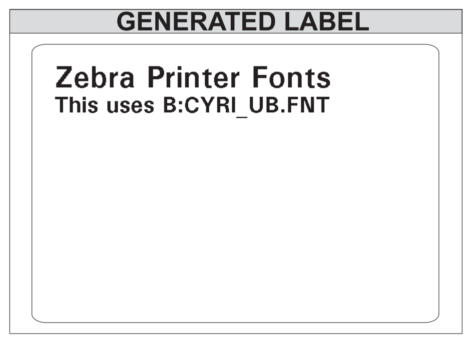
| Parameters | Details |
|---|---|
| `o` = font name | *Accepted Values:* any valid font<br>*Default Value:* if an invalid or no name is entered, the default set by `^CF` is used. If no font has been specified in `^CF`, font A is used.<br>The font named carries over on all subsequent `^A@` commands without a font name. |
| `x` = extension | *Accepted Values:*<br>`.FNT` = font<br>`.TTF` = TrueType Font |

**Example •** This example is followed by a table that identifies the called out lines of code:

**ZPL II CODE**

```
^XA
^A@N,50,50,B:CYRI_UB.FNT
^FO100,100
^FDZebra Printer Fonts^FS
^A@N,40,40
^FO100,150
^FDThis uses B:CYRI_UB.FNT^FS
^XZ
```

**GENERATED LABEL**




| | |
|---|---|
| **1** | Starts the label format. |
| **2** | Searches non-volatile printer memory (`B:`) for `CYRI_UB.FNT`. When the font is found, the `^A@` command sets the print orientation to normal and the character size to 50 dots by 50 dots. |
| **3** | Sets the field origin at 100,100. |
| **4** | Prints the field data, *Zebra Printer Fonts* on the label. |
| **5** | Calls the font again and character size is decreased to 40 dots by 40 dots. |
| **6** | Sets the new field origin at 100,150. |
| **7** | Prints the field data, *This uses the B:CYRI_UB.FNT* on the label. |
| **8** | Ends the label format. |

For reference, see *Zebra Code Page 850* on page 342, *Fonts and Bar Codes* on page 351, and *ASCII* on page 347.

# ^B0

## Aztec Bar Code Parameters

**Description** The ^B0 command creates a two-dimensional matrix symbology made up of square modules arranged around a bulls-eye pattern at the center.

**Note •** The Aztec bar code works with firmware V60.13.0.11A and higher.

**Format** ^B0a,b,c,d,e,f,g

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = orientation | *Accepted Values:*<br>N = normal<br>R = rotated<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| b = magnification factor | *Accepted Values:* 1 to 10<br>*Default Value:*<br>1 on 150 dpi printers<br>2 on 200 dpi printers<br>3 on 300 dpi printers<br>6 on 600 dpi printers |
| c = extended channel interpretation code indicator | *Accepted Values:*<br>Y = if data contains ECICs<br>N = if data does not contain ECICs<br>*Default Value:* N |
| d = error control and symbol size/type indicator | *Accepted Values:*<br>0 = default error correction level<br>01 to 99 = error correction percentage (minimum)<br>101 to 104 = 1 to 4-layer compact symbol<br>201 to 232 = 1 to 32-layer full-range symbol<br>300 = a simple Aztec “Rune”<br>*Default Value:* 0 |
| e = menu symbol indicator | *Accepted Values:*<br>Y = if this symbol is to be a menu (bar code reader initialization) symbol<br>N = if it is not a menu symbol<br>*Default Value:* N |

| Parameters | Details |
|---|---|
| `f` = number of symbols for structured append | *Accepted Values:* `1` through `26`<br>*Default Value:* `1` |
| `g` = *optional* ID field for structured append | The ID field is a text string with 24-character maximum<br>*Default Value:* no ID |

**Example** • This is an example of the `^B0` command:

**ZPL II CODE**

```
^XA
^B0R,7,N,0,N,1,0
^FD 7. This is testing label 7^FS
^XZ
```

**GENERATED LABEL**

# ^B1

## Code 11 Bar Code

**Description** The ^B1 command produces the Code 11 bar code, also known as USD-8 code. In a Code 11 bar code, each character is composed of three bars and two spaces, and the character set includes 10 digits and the hyphen (-).

- ^B1 supports print ratios of 2.0:1 to 3.0:1.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.

**Format** ^B1o,e,h,f,g

**Important •** If additional information about the Code 11 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value: current* ^FW *value* |
| e = check digit | *Accepted Values:*<br>Y = 1 digit<br>N = 2 digits<br>*Default Value:* N |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>Y = yes<br>N = no<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>Y = yes<br>N = no<br>*Default Value:* N |

**Example •** This is an example of the Code 11 bar code:

**ZPL II CODE**

```
^XA
^FO100,100^BY3
^B1N,N,150,Y,N
^FD123456^FS
^XZ
```

**CODE 11 BAR CODE**

△12345611△

**CODE 11 BAR CODE CHARACTERS**

0 1 2 3 4 5 6 7 8 9 -

**Internal Start/Stop Character:**△

*When used as a stop character:*

△ is used with 1 check digit
△ is used with 2 check digits

# ^B2

## Interleaved 2 of 5 Bar Code

**Description** The ^B2 command produces the Interleaved 2 of 5 bar code, a high-density, self-checking, continuous, numeric symbology.

Each data character for the Interleaved 2 of 5 bar code is composed of five elements: five bars or five spaces. Of the five elements, two are wide and three are narrow. The bar code is formed by interleaving characters formed with all spaces into characters formed with all bars.

- ^B2 supports print ratios of 2.0:1 to 3.0:1.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.

**Format** ^B2o,h,f,g,e,j

**Important •** If additional information about the Interleaved 2 of 5 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>Y = yes<br>N = no<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>Y = yes<br>N = no<br>*Default Value:* N |
| e = calculate and print Mod 10 check digit | *Accepted Values:*<br>Y = yes<br>N = no<br>*Default Value:* N |

**Example •** This is an example of an Interleaved 2 of 5 bar code:

**ZPL II CODE**

```
^XA
^FO100,100^BY3
^B2N,150,Y,N,N
^FD123456^FS
^XZ
```

**INTERLEAVED 2 OF 5 BAR CODE**

123456

| INTERLEAVED 2 OF 5 BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Start/Stop (internal)** | | | | | | | | | |

**Comments** The total number of digits in an Interleaved 2 of 5 bar code must be even. The printer automatically adds a leading 0 (zero) if an odd number of digits is received.

The Interleaved 2 of 5 bar code uses the Mod 10 check-digit scheme for error checking. For more information on Mod 10 check digits, see *Mod 10 Check Digit* on page 364.

# ^B3

## Code 39 Bar Code

**Description** The Code 39 bar code is the standard for many industries, including the U.S. Department of Defense. It is one of three symbologies identified in the American National Standards Institute (ANSI) standard MH10.8M-1983. Code 39 is also known as USD-3 Code and 3 of 9 Code.

Each character in a Code 39 bar code is composed of nine elements: five bars, four spaces, and an inter-character gap. Three of the nine elements are wide; the six remaining elements are narrow.

- ^B3 supports print ratios of 2.0:1 to 3.0:1.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.
- Code 39 automatically generates the start and stop character (*).
- Asterisk (*) for start and stop character prints in the interpretation line, if the interpretation line is turned on.
- Code 39 is capable of encoding the full 128-character ASCII set.

**Format** `^B3o,e,h,f,g,j,m,n`

**Important •** If additional information about the Code 39 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| e = Mod-43 check digit | *Accepted Values:*<br>Y = yes<br>N = no<br>*Default Value:* N |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |

| Parameters | Details |
|---|---|
| `f` = print interpretation line | *Accepted Values:*<br>`Y` = yes<br>`N` = no<br>*Default Value:* `Y` |
| `g` = print interpretation line above code | *Accepted Values:*<br>`Y` = yes<br>`N` = no<br>*Default Value:* `N` |

**Example** • This is an example of a Code 39 bar code:

| ZPL II CODE | CODE 39 BAR CODE |
|---|---|
| `^XA`<br>`^FO100,100^BY3`<br>`^B3N,N,100,Y,N`<br>`^FD123ABC^FS`<br>`^XZ` | *123ABC* |

**CODE 39 BAR CODE CHARACTERS**

0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- . $ / + % Space

**Comments** Extended ASCII is a function of the scanner, not of the bar code. Your scanner must have extended ASCII enabled for this feature to work. To enable extended ASCII in the Code 39, you must first encode *+$* in your `^FD` statement. To disable extended ASCII, you must encode *-$* in your `^FD` statement.

**Example** • This example encodes a carriage return with line feed into a Code 39 bar code:

| ZPL II CODE | GENERATED LABELS |
|---|---|
| `^XA`<br>`^FO20,20`<br>`^B3N,N,100,Y`<br>`^FDTEST+$$M$J-$^FS`<br>`^XZ` | *TEST+$$M$J-$* |

## Full ASCII Mode for Code 39

Code 39 can generate the full 128-character ASCII set using paired characters as shown in these tables:

**Table 1 • Code 39 Full ASCII Mode**

| ASCII | Code 39 |
|---|---|
| SOH | $A |
| STX | $B |
| ETX | $C |
| EOT | $D |
| ENQ | $E |
| ACK | $F |
| BEL | $G |
| BS | $H |
| HT | $I |
| LF | $J |
| VT | $K |
| FF | $L |
| CR | $M |
| SO | $N |
| SI | $O |
| DLE | $P |
| DC1 | $Q |
| DC2 | $R |
| DC3 | $S |
| DC4 | $T |
| NAK | $U |
| SYN | $V |
| ETB | $W |
| CAN | $X |
| EM | $Y |
| SUB | $Z |
| ESC | %A |
| FS | %B |
| FS | %C |
| RS | %D |
| US | %E |

| ASCII | Code 39 |
|---|---|
| SP | Space |
| ! | /A |
| “ | /B |
| # | /C |
| $ | /D |
| % | /E |
| & | /F |
| ‘ | /G |
| ( | /H |
| ) | /I |
| * | /J |
| ++ | /K |
| ‘ | /L |
| - | - |
| . | . |
| / | /O |
| 0 | O |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| : | /Z |
| ; | %F |
| < | %G |
| = | %H |
| > | %I |
| ? | %J |

**Table 2 • Code 39 Full ASCII Mode**

| ASCII | Code 39 |
|---|---|
| @ | %V |
| A | A |
| B | B |
| C | C |
| D | D |
| E | E |
| F | F |
| G | G |
| H | H |
| I | I |
| J | J |
| K | K |
| L | L |
| M | M |
| N | N |
| O | O |
| P | P |
| Q | Q |
| R | R |
| S | S |
| T | T |
| U | U |
| V | V |
| W | W |
| X | X |
| Y | Y |
| Z | Z |
| [ | %K |
| \ | %L |
| ] | %M |
| ^ | %N |
| _ | %O |

| ASCII | Code 39 |
|---|---|
| ‘ | %W |
| a | +A |
| b | +B |
| c | +C |
| d | +D |
| e | +E |
| f | +F |
| g | +G |
| h | +H |
| I | +I |
| j | +J |
| k | +K |
| l | +L |
| m | +M |
| n | +N |
| o | +O |
| p | +P |
| q | +Q |
| r | +R |
| s | +S |
| t | +T |
| u | +U |
| v | +V |
| w | +W |
| x | +X |
| y | +Y |
| z | +Z |
| { | %P |
| \| | %Q |
| } | %R |
| ~ | %S |
| DEL | %T, %X |

# ^B4

## Code 49 Bar Code

**Description** The ^B4 command creates a multi-row, continuous, variable-length symbology capable of encoding the full 128-character ASCII set. It is ideally suited for applications requiring large amounts of data in a small space.

The code consists of two to eight rows. A row consists of a leading quiet zone, four symbol characters encoding eight code characters, a stop pattern, and a trailing quiet zone. A separator bar with a height of one module separates each row. Each symbol character encodes two characters from a set of Code 49 characters.

- ^B4 has a fixed print ratio.
- Rows can be scanned in any order.

**Format** ^B4o,h,f,m

**Important •** For additional information about the Code 49 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = height multiplier of individual rows | *Accepted Values:* 1 to height of label<br>*Default Value:* value set by ^BY<br>This number multiplied by the module equals the height of the individual rows in dots. 1 is not a recommended value. |

| Parameters | Details |
|---|---|
| `f` = print interpretation line | *Accepted Values:*<br>`N` = no line printed<br>`A` = print interpretation line above code<br>`B` = print interpretation line below code<br>*Default Value:* `N`<br>When the field data exceeds two rows, expect the interpretation line to extend beyond the right edge of the bar code symbol. |
| `m` = starting mode | *Accepted Values:*<br>`0` = Regular Alphanumeric Mode<br>`1` = Multiple Read Alphanumeric<br>`2` = Regular Numeric Mode<br>`3` = Group Alphanumeric Mode<br>`4` = Regular Alphanumeric Shift 1<br>`5` = Regular Alphanumeric Shift 2<br>`A` = Automatic Mode. The printer determines the starting mode by analyzing the field data.<br>*Default Value:* `A` |

**Example** • This is an example of a Code 49 bar code:

**ZPL II CODE**

```
^XA
^FO150,100^BY3
^B4N,20,A,A
^FD12345ABCDE^FS
^XZ
```

**CODE 49 BAR CODE**

12345ABCDE

**Table 3 • Code 49**

| Field Data Set | Unshifted Character Set | Shift 1 Character Set | Shift 2 Character Set |
|---|---|---|---|
| 0 | 0 | ’ | |
| 1 | 1 | ESC | ; |
| 2 | 2 | FS | < |
| 3 | 3 | GS | = |
| 4 | 4 | RS | > |
| 5 | 5 | US | ? |
| 6 | 6 | ! | @ |
| 7 | 7 | “ | [ |
| 8 | 8 | # | \ |
| 9 | 9 | & | ] |
| A | A | SOH | a |
| B | B | STX | b |
| C | C | ETX | c |
| D | D | EOT | d |
| E | E | ENQ | e |
| F | F | ACK | f |
| G | G | BEL | g |
| H | H | BS | h |
| I | I | HT | I |
| J | J | LF | j |
| K | K | VT | k |
| L | L | FF | l |
| M | M | CR | m |
| N | N | SO | n |
| O | O | SI | o |
| P | P | DLE | p |
| Q | Q | DC1 | q |
| R | R | DC2 | r |
| S | S | DC3 | s |
| T | T | DC4 | t |
| U | U | NAK | u |
| V | V | SYN | v |
| W | W | ETB | w |
| X | X | CAN | x |
| Y | Y | EM | y |
| Z | Z | SUB | z |
| - | - | ( | _ |
| . | . | ) | ‘ |
| SPACE | SPACE | Null | DEL |
| $ | $ | * | { |
| / | / | , | \| |
| ++ | ++ | : | } |
| % | % | reserved | ~ |
| < (Shift 1) | | | |
| > (Shift 2) | | | |
| : (N.A.) | | | |
| ; (N.A.) | | | |
| ? (N.A.) | | | |
| = (Numeric Shift) | | | |

**Code 49 Shift 1 and 2 Character Substitutions**

## Code 49 Field Data Character Set

The ^FD data sent to the printer when using starting modes 0 to 5 is based on the Code 49 Internal Character Set. This is shown in the first column of the Code 49 table on the previous page. These characters are Code 49 control characters:

`: ; < = > ?`

Valid field data must be supplied when using modes 0 to 5. Shifted characters are sent as a two-character sequence of a shift character followed by a character in the unshifted character set.

**Example •** To encode a lowercase **a**, send **a >** (Shift 2) followed by an uppercase **A**. If interpretation line printing is selected, a lowercase *a* prints in the interpretation line. This reflects what the output from the scanner reads. Code 49 uses uppercase alphanumeric characters only.

If an invalid sequence is detected, the Code 49 formatter stops interpreting field data and prints a symbol with the data up to the invalid sequence. These are examples of invalid sequences:

- Terminating numeric mode with any characters other than 0 to 9 or a Numeric Space.
- Starting in Mode 4 (Regular Alphanumeric Shift 1) and the first field data character is not in the Shift 1 set.
- Starting in Mode 5 (Regular Alphanumeric Shift 2) and the first field data character is not in the Shift 2 set.
- Sending Shift 1 followed by a character not in the Shift 1 set.
- Sending Shift 2 followed by a character not in the Shift 2 set.
- Sending two Shift 1 or Shift 2 control characters.

## Advantages of Using the Code 49 Automatic Mode

Using the default (Automatic Mode) completely eliminates the need for selecting the starting mode or manually performing character shifts. The Automatic Mode analyzes the incoming ASCII string, determines the proper mode, performs all character shifts, and compacts the data for maximum efficiency.

Numeric Mode is selected or shifted only when five or more continuous digits are found. Numeric packaging provides no space advantage for numeric strings consisting of fewer than eight characters.

# ^B5

## Planet Code bar code

**Description** The ^B5 command is supported in all printers as a resident bar code.

**Note •** Accepted bar code characters are 0 - 9.

**Format** `^B5o,h,f,g`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `o` = orientation code | *Accepted Values:*<br>`N` = normal<br>`R` = rotated<br>`I` = inverted 180 degrees<br>`B` = read from bottom up, 270 degrees<br>*Default Value:* current `^FW` value |
| `h` = bar code height (in dots) | *Accepted Values:* `1` to `9999`<br>*Default Value:* value set by `^BY` |
| `f` = interpretation line | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |
| `g` = determines if the interpretation line is printed above the bar code | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |

**Example •** This is an example of a Planet Code bar code:

**ZPL II CODE**

```
^XA
^FO150,100^BY3
^B5N,100,Y,0
^FD12345678901^FS
^XZ
```

**GENERATED LABEL**

12345678901

# ^B7

## PDF417 Bar Code

**Description** The ^B7 command produces the PDF417 bar code, a two-dimensional, multirow, continuous, stacked symbology. PDF417 is capable of encoding over 1,000 characters per bar code. It is ideally suited for applications requiring large amounts of information at the time the bar code is read.

The bar code consists of three to 90 stacked rows. Each row consists of start and stop patterns and symbol characters called *code-words*. A code-word consists of four bars and four spaces. A three code-word minimum is required per row.

The PDF417 bar code is also capable of using the structured append option (^FM), which allows you to extend the field data limitations by printing multiple bar codes. For more information on using structured append, see *^FM* on page 142.

- PDF417 has a fixed print ratio.
- Field data (^FD) is limited to 3K of character data.

**Format** `^B7o,h,s,c,r,t`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height for individual rows (in dots) | *Accepted Values:* 1 to height of label<br>*Default Value:* value set by ^BY<br>This number multiplied by the module equals the height of the individual rows in dots. If this number is not specified, the overall bar code height, divided by the number of rows, equals the height of the individual rows in dots, where the overall bar code height is defined by the ^BY command. 1 is not a recommended value. |
| s = security level | *Accepted Values:* 1 to 8 (error detection and correction)<br>*Default Value:* 0 (error detection only)<br>This determines the number of error detection and correction code-words to be generated for the symbol. The default level provides only error detection without correction. Increasing the security level adds increasing levels of error correction and increases the symbol size. |

| Parameters | Details |
|---|---|
| `c` = number of data columns to encode | *Accepted Values:* `1` to `30`<br>*Default Value:* `1:2` (row-to-column aspect ratio)<br>You can specify the number of code-word columns giving control over the width of the symbol. |
| `r` = number of rows to encode | *Accepted Values:* `3` to `90`<br>*Default Value:* `1:2` (row-to-column aspect ratio)<br>You can specify the number of symbol rows giving control over the height of the symbol. For example, with no row or column values entered, 72 code-words would be encoded into a symbol of six columns and 12 rows. Depending on code-words, the aspect ratio is not always exact. |
| `t` = truncate right row indicators and stop pattern | *Accepted Values:*<br>`N` = no truncation<br>`Y` = perform truncation<br>*Default Value:* `N` |

**Example** • This is an example of a PDF417 bar code:

**ZPL II CODE**

```
^XA
^BY2,3
^FO10,10^B7N,5,5,,83,N
^FDZebra Technologies
Corporation strives to be
the expert supplier of
innovative solutions to
speciality demand labeling
and ticketing problems of
business and government.
We will attract and retain
the best people who will
understand our customer's
needs and provide them with
systems, hardware, software,
consumables and service
offering the best value,
high quality, and reliable
performance, all delivered
in a timely manner.
^FS^XZ
```

**PDF417 BAR CODE**

**Example •** This is an example of a PDF417 without and with truncation selected:

PDF417 without Truncation being selected

PDF417 with Truncation being selected

**Example •** This example shows the `^B7` command used with field hex (`^FH`) characters:

| ZPL II CODE | GENERATED LABEL |
|---|---|
| ^XA<br>^FO50,50^BY3,3.0^B7N,8,5,7,21,N<br>^FH_^FD[)>_1E06_1DP12345678_1DQ160<br>_1D1JUN123456789A2B4C6D8E_1D20LA6-987<br>_1D21L54321 ZES_1D15KG1155<br>_1DBSC151208_1D7Q10GT_1E_04^FS<br>^XZ | |

**Comments** Noted in this bulleted list:

- If both columns and rows are specified, their product must be less than 928.
- No symbol is printed if the product of columns and rows is greater than 928.
- No symbol is printed if total code-words are greater than the product of columns and rows.
- Serialization is not allowed with this bar code.
- The truncation feature can be used in situations where label damage is not likely. The right row indicators and stop pattern is reduced to a single module bar width. The difference between a non truncated and a truncated bar code is shown in the previous examples.

### Special Considerations for ^BY When Using PDF417

When used with ^B7, the parameters for the ^BY command are:

**w = module width (in dots)**

*Accepted Values:* 2 to 10

*Default Value:* 2

**r = ratio**

*Fixed Value:* 3 (ratio has no effect on PDF417)

**h = height of bars (in dots)**

*Accepted Values:* 1 to 32000

*Default Value:* 10

PDF417 uses this only when row height is not specified in the ^B7 h parameter.

### Special Considerations for ^FD When Using PDF417

The character set sent to the printer with the ^FD command includes the full ASCII set, except for those characters with special meaning to the printer.

See *Zebra Code Page 850* on page 342, *^CC ~CC* on page 107, and *^CT ~CT* on page 118.

- CR and LF are also valid characters for all ^FD statements. This scheme is used:

  \& = carriage return/line feed

  \\ = backslash (\)

- ^CI13 must be selected to print a backslash (\).

# ^B8

## EAN-8 Bar Code

**Description** The ^B8 command is the shortened version of the EAN-13 bar code. EAN is an acronym for European Article Numbering. Each character in the EAN-8 bar code is composed of four elements: two bars and two spaces.

- ^B8 supports a fixed ratio.
- Field data (^FD) is limited to exactly seven characters. ZPL II automatically pads or truncates on the left with zeros to achieve the required number of characters.
- When using JAN-8 (Japanese Article Numbering), a specialized application of EAN-8, the first two non-zero digits sent to the printer are always 49.

**Format** ^B8o,h,f,g

**Important** • If additional information about the EAN-8 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

**Example** • This is an example of an EAN-8 bar code:

**ZPL II CODE**

```
^XA
^FO100,100^BY3
^B8N,100,Y,N
^FD1234567^FS
^XZ
```

**EAN-8 BAR CODE**

| EAN-8 BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

# ^B9

## UPC-E Bar Code

**Description** The ^B9 command produces a variation of the UPC symbology used for number system 0. It is a shortened version of the UPC-A bar code, where zeros are suppressed, resulting in codes that require less printing space. The 6 dot/mm, 12 dot/mm, and 24 dot/mm printheads produce the UPC and EAN symbologies at 100 percent of their size. However, an 8 dot/mm printhead produces the UPC and EAN symbologies at a magnification factor of 77 percent.

Each character in a UPC-E bar code is composed of four elements: two bars and two spaces. The ^BY command must be used to specify the width of the narrow bar.

- ^B9 supports a fixed ratio.
- Field data (^FD) is limited to exactly 10 characters, requiring a five-digit manufacturer's code and five-digit product code.
- When using the zero-suppressed versions of UPC, you must enter the full 10-character sequence. ZPL II calculates and prints the shortened version.

**Format** `^B9,h,f,g,e`

**Important •** If additional information about the UPC-E bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |

| Parameters | Details |
|---|---|
| `g` = print interpretation line above code | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |
| `e` = print check digit | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `Y` |

**Example •** This is an example of a UPC-E bar code:

**ZPL II CODE**

```
^XA
^FO150,100^BY3
^B9N,100,Y,N,Y
^FD1230000045^FS
^XZ
```

**UPC-E BAR CODE**

0 123453 1

| UPC-E BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

### Rules for Proper Product Code Numbers

- If the last three digits in the manufacturer's number are 000, 100, or 200, valid product code numbers are 00000 to 00999.
- If the last three digits in the manufacturer's number are 300, 400, 500, 600, 700, 800, or 900, valid product code numbers are 00000 to 00099.
- If the last two digits in the manufacturer's number are 10, 20, 30, 40, 50, 60, 70, 80, or 90, valid product code numbers are 00000 to 00009.
- If the manufacturer's number does not end in zero (0), valid product code numbers are 00005 to 00009.

# ^BA

## Code 93 Bar Code

**Description** The ^BA command creates a variable length, continuous symbology. The Code 93 bar code is used in many of the same applications as Code 39. It uses the full 128-character ASCII set. ZPL II, however, does not support ASCII control codes or escape sequences. It uses the substitute characters shown below.

| Control Code | ZPL II Substitute |
|---|---|
| Ctrl $ | & |
| Ctrl % | ‘ |
| Ctrl / | ( |
| Ctrl + | ) |

Each character in the Code 93 bar code is composed of six elements: three bars and three spaces. Although invoked differently, the human-readable interpretation line prints as though the control code has been used.

- ^BA supports a fixed print ratio.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.

**Format** `^BAo,h,f,g,e`

**Important •** If additional information about the Code 93 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |

| Parameters | Details |
|---|---|
| `g` = print interpretation line above code | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |
| `e` = print check digit | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |

**Example •** This is an example of a Code 93 bar code:

| CODE 93 BAR CODE | ZPL II CODE |
|---|---|
| □12345ABCDE□ | `^XA`<br>`^FO100,75^BY3`<br>`^BAN,100,Y,N,N`<br>`^FD12345ABCDE^FS`<br>`^XZ` |

**CODE 93 BAR CODE CHARACTERS**

0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- . $ / + % & ’ ( )
SPACE

□ *Denotes an internal start/stop character that must precede and follow every bar code message.*

**Comments** All control codes are used in pairs.

Code 93 is also capable of encoding the full 128-character ASCII set. For more details, see *Table 4 on page 41*.

## Full ASCII Mode for Code 93

Code 93 can generate the full 128-character ASCII set using paired characters as shown in *Table 4 on page 41*.

**Table 4 • Code 93 Full ASCII Mode**

| ASCII | Code 93 |
|---|---|
| NUL | ‘U |
| SOH | &A |
| STX | &B |
| ETX | &C |
| EOT | &D |
| ENQ | &E |
| ACK | &F |
| BEL | &G |
| BS | &H |
| HT | &I |
| LF | &J |
| VT | &K |
| FF | &L |
| CR | &M |
| SO | &N |
| SI | &O |
| DLE | &P |
| DC1 | &Q |
| DC2 | &R |
| DC3 | &S |
| DC4 | &T |
| NAK | &U |
| SYN | &V |
| ETB | &W |
| CAN | &X |
| EM | &Y |
| SUB | &Z |
| ESC | ‘A |
| FS | ‘B |
| FS | ‘C |
| RS | ‘D |
| US | ‘E |

| ASCII | Code 93 |
|---|---|
| SP | Space |
| ! | (A |
| “ | (B |
| # | (C |
| $ | (D |
| % | (E |
| & | (F |
| ‘ | (G |
| ( | (H |
| ) | (I |
| * | (J |
| ++ | ++ |
| ‘ | (L |
| - | - |
| . | . |
| / | / |
| 0 | O |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| : | (Z |
| ; | ‘F |
| < | ‘G |
| = | ‘H |
| > | ‘I |
| ? | ‘J |

**Table 5 • Code 93 Full ASCII Mode**

| ASCII | Code 93 |
|---|---|
| @ | ‘V |
| A | A |
| B | B |
| C | C |
| D | D |
| E | E |
| F | F |
| G | G |
| H | H |
| I | I |
| J | J |
| K | K |
| L | L |
| M | M |
| N | N |
| O | O |
| P | P |
| Q | Q |
| R | R |
| S | S |
| T | T |
| U | U |
| V | V |
| W | W |
| X | X |
| Y | Y |
| Z | Z |
| [ | ‘K |
| \ | ‘L |
| ] | ‘M |
| ^ | ‘N |
| _ | ‘O |

| ASCII | Code 93 |
|---|---|
| ‘ | ‘W |
| a | )A |
| b | )B |
| c | )C |
| d | )D |
| e | )E |
| f | )F |
| g | )G |
| h | )H |
| I | )I |
| j | )J |
| k | )K |
| l | )L |
| m | )M |
| n | )N |
| o | )O |
| p | )P |
| q | )Q |
| r | )R |
| s | )S |
| t | )T |
| u | )U |
| v | )V |
| w | )W |
| x | )X |
| y | )Y |
| z | )Z |
| { | ‘P |
| \| | ‘Q |
| } | ‘R |
| ~ | ‘S |
| DEL | ‘T |

# ^BB

## CODABLOCK Bar Code

**Description** The `^BB` command produces a two-dimensional, multirow, stacked symbology. It is ideally suited for applications that require large amounts of information.

Depending on the mode selected, the code consists of one to 44 stacked rows. Each row begins and ends with a start and stop pattern.

- CODABLOCK A supports variable print ratios.
- CODABLOCK E and F support only fixed print ratios.

**Format** `^BBo,h,s,c,r,m`

**Important •** If additional information about the CODABLOCK bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `o` = orientation | *Accepted Values:*<br>`N` = normal<br>`R` = rotated 90 degrees (clockwise)<br>`I` = inverted 180 degrees<br>`B` = read from bottom up, 270 degrees<br>*Default Value:* `N` |
| `h` = bar code height for individual rows (in dots) | *Accepted Values:* `2` to `32000`<br>*Default Value:* `8`<br>This number, multiplied by the module, equals the height of the individual row in dots. |
| `s` = security level | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `Y`<br>Security level determines whether symbol check-sums are generated and added to the symbol. Check sums are never generated for single-row symbols. This can be turned off only if parameter `m` is set to `A`. |
| `c` = number of characters per row (data columns) | *Accepted Values:* `2` to `62` characters<br>This is used to encode a CODABLOCK symbol. It gives the you control over the width of the symbol. |

| Parameters | Details |
| --- | --- |
| r = number of rows to encode | *Accepted Values:*<br>for CODABLOCK A: 1 to 22<br>for CODABLOCK E and F: 2 to 4<br>• If values for c and r are not specified, a single row is produced.<br>• If a value for r is not specified, and c exceeds the maximum range, a single row equal to the field data length is produced.<br>• If a value for c is not specified, the number of characters per row is derived by dividing the field data by the value of r.<br>• If both parameters are specified, the amount of field data must be less than the product of the specified parameters. If the field data exceeds the value of the product, either no symbol or an error code is printed (if ^CV is active).<br>• If the data field contains primarily numeric data, fewer than the specified rows might be printed. If the field data contains several shift and code-switch characters, more than the specified number of rows might be printed. |
| m = mode | *Accepted Values:* A, E, F<br>CODABLOCK A uses the Code 39 character set.<br>CODABLOCK F uses the Code 128 character set.<br>CODABLOCK E uses the Code 128 character set and automatically adds FNC1.<br>*Default Value:* F |

**Example** • This is an example of a CODABLOCK bar code:

| CODABLOCK BAR CODE | ZPL II CODE |
|---|---|
| | ^XA<br>^BY2,3<br>^FO10,10^BBN,30,,30,44,E<br>^FDZebra Technologies<br>Corporation strives to be<br>the expert supplier of<br>innovative solutions to<br>speciality demand labeling<br>and ticketing problems of<br>business and government.<br>We will attract and retain<br>the best people who will<br>understand our customer's<br>needs and provide them with<br>systems, hardware, software,<br>consumables and service<br>offering the best value,<br>high quality, and reliable<br>performance, all delivered<br>in a timely manner.^FS<br>^XZ |

### Special Considerations for the ^BY Command When Using ^BB

The parameters for the ^BYw,r,h command, when used with a ^BB code, are as follows:

**w = module width (in dots)**

*Accepted Values:* 2 to 10 (CODABLOCK A only)

*Default Value:* 2

**r = ratio**

*Fixed Value:* 3 (ratio has no effect on CODABLOCK E or F)

**h = height of bars (in dots)**

*Accepted Values:* 1 to 32,32000

*Default Value:* 10

CODABLOCK uses this as the overall symbol height only when the row height is not specified in the ^BB h parameter.

### Special Considerations for ^FD Character Set When Using ^BB

The character set sent to the printer depends on the mode selected in parameter m.

**CODABLOCK A:** CODABLOCK A uses the same character set as Code 39. If any other character is used in the ^FD statement, either no bar code is printed or an error message is printed (if ^CV is active).

**CODABLOCK E:** The Automatic Mode includes the full ASCII set except for those characters with special meaning to the printer. Function codes or the Code 128 Subset A <nul> character can be inserted using of the ^FH command.

| | |
|---|---|
| **<fnc1> = 80 hex** | **<fnc3> = 82 hex** |
| **<fnc2> = 81 hex** | **<fnc4> = 83 hex** |
| **<nul> = 84 hex** | |

For any other character above 84 hex, either no bar code is printed or an error message is printed (if ^CV is active).

**CODABLOCK F:** CODABLOCK F uses the full ASCII set, except for those characters with special meaning to the printer. Function codes or the Code 128 Subset A <nul> character can be inserted using of the ^FH command.

| | |
|---|---|
| **<fnc1> = 80 hex** | **<fnc3> = 82 hex** |
| **<fnc2> = 81 hex** | **<fnc4> = 83 hex** |
| **<nul> = 84 hex** | |

# ^BC

## Code 128 Bar Code (Subsets A, B, and C)

**Description** The ^BC command creates the Code 128 bar code, a high-density, variable length, continuous, alphanumeric symbology. It was designed for complexly encoded product identification.

Code 128 has three subsets of characters. There are 106 encoded printing characters in each set, and each character can have up to three different meanings, depending on the character subset being used. Each Code 128 character consists of six elements: three bars and three spaces.

- ^BC supports a fixed print ratio.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.

**Format** ^BCo,h,f,g,e,m

**Important •** If additional information about the Code 128 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:* Y (yes) or N (no)<br>*Default Value:* Y<br>The interpretation line can be printed in any font by placing the font command before the bar code command. |
| g = print interpretation line above code | *Accepted Values:* Y (yes) or N (no)<br>*Default Value:* N |

| Parameters | Details |
|---|---|
| `e` = UCC check digit | *Accepted Values:* `Y` (turns on) or `N` (turns off)<br>Mod 103 check digit is always there. It cannot be turned on or off. Mod 10 and 103 appear together with `e` turned on.<br>*Default Value:* `N` |
| `m` = mode | *Accepted Values:*<br>`N` = no selected mode<br>`U` = UCC Case Mode<br>• More than 19 digits in `^FD` or **`^SN`** are eliminated.<br>• Fewer than 19 digits in `^FD` or ^SN add zeros to the right to bring the count to 19. This produces an invalid interpretation line.<br>`A` = Automatic Mode<br>This analyzes the data sent and automatically determines the best packing method. The full ASCII character set can be used in the `^FD` statement — the printer determines when to shift subsets. A string of four or more numeric digits causes an automatic shift to Subset C.<br>`D` = UCC/EAN Mode (x.11.x and newer firmware)<br>This allows dealing with UCC/EAN with and without chained application identifiers. The code starts in the appropriate subset followed by FNC1 to indicate a UCC/EAN 128 bar code. The printer automatically strips out parentheses and spaces for encoding, but prints them in the human-readable section. The printer automatically determines if a check digit is required, calculate it, and print it. Automatically sizes the human readable.<br>*Default Value:* `N` |

**Example •** This is an example of a Code 128 bar code:

| CODE 128 BAR CODE | ZPL II CODE |
|---|---|
| 123456 | `^XA`<br>`^FO100,100^BY3`<br>`^BCN,100,Y,N,N`<br>`^FD123456^FS`<br>`^XZ` |

## Code 128 Subsets

The Code 128 character subsets are referred to as Subset A, Subset B, and Subset C. A subset can be selected in these ways:

- A special Invocation Code can be included in the field data (`^FD`) string associated with that bar code.
- The desired Start Code can be placed at the beginning of the field data. If no Start Code is entered, Subset B are used.

To change subsets within a bar code, place the Invocation Code at the appropriate points within the field data (`^FD`) string. The new subset stays in effect until changed with the Invocation Code. For example, in Subset C, `>7` in the field data changes the Subset to A.

Table 6 shows the Code 128 Invocation Codes and Start Characters for the three subsets.

**Table 6 • Code 128 Invocation Characters**

| Invocation Code | Decimal Value | Subset A Character | Subset B Character | Subset C Character |
|---|---|---|---|---|
| >< | 62 | | | |
| >0 | 30 | > | > | |
| >= | 94 | | ~ | |
| >1 | 95 | USQ | DEL | |
| >2 | 96 | FNC 3 | FNC 3 | |
| >3 | 97 | FNC 2 | FNC 2 | |
| >4 | 98 | SHIFT | SHIFT | |
| >5 | 99 | CODE C | CODE C | |
| >6 | 100 | CODE B | FNC 4 | CODE B |
| >7 | 101 | FNC 4 | CODE A | CODE A |
| >8 | 102 | FNC 1 | FNC 1 | FNC 1 |
| **Start Characters** | | | | |
| >9 | 103 | Start Code A | (Numeric Pairs give Alpha/Numerics) | |
| >: | 104 | Start Code B | (Normal Alpha/Numeric) | |
| >; | 105 | Start Code C | (All numeric (00 - 99) | |

Table 7 shows the character sets for Code 128:

**Table 7 • Code 128 character sets**

| Value | Code A | Code B | Code C | Value | Code A | Code B | Code C |
|---|---|---|---|---|---|---|---|
| 0 | SP | SP | 00 | 53 | U | U | 53 |
| 1 | ! | ! | 01 | 54 | V | V | 54 |
| 2 | " | " | 02 | 55 | W | W | 55 |
| 3 | # | # | 03 | 56 | X | X | 56 |
| 4 | $ | $ | 04 | 57 | Y | Y | 57 |
| 5 | % | % | 05 | 58 | Z | Z | 58 |
| 6 | & | & | 06 | 59 | [ | [ | 59 |
| 7 | ' | ' | 07 | 60 | \ | \ | 60 |
| 8 | ( | ( | 08 | 61 | ] | ] | 61 |
| 9 | ) | ) | 09 | 62 | ^ | ^ | 62 |
| 10 | * | * | 10 | 63 | _ | _ | 63 |
| 11 | ++ | ++ | 11 | 64 | NUL | . | 64 |
| 12 | , | , | 12 | 65 | SOH | a | 65 |
| 13 | - | - | 13 | 66 | STX | b | 66 |
| 14 | . | . | 14 | 67 | ETX | c | 67 |
| 15 | / | / | 15 | 68 | EOT | d | 68 |
| 16 | 0 | 0 | 16 | 69 | ENQ | e | 69 |
| 17 | 1 | 1 | 17 | 70 | ACK | f | 70 |
| 18 | 2 | 2 | 18 | 71 | BEL | g | 71 |
| 19 | 3 | 3 | 19 | 72 | BS | h | 72 |
| 20 | 4 | 4 | 20 | 73 | HT | i | 73 |
| 21 | 5 | 5 | 21 | 74 | LF | j | 74 |
| 22 | 6 | 6 | 22 | 75 | VT | k | 75 |
| 23 | 7 | 7 | 23 | 76 | FF | l | 76 |
| 24 | 8 | 8 | 24 | 77 | CR | m | 77 |
| 25 | 9 | 9 | 25 | 78 | SO | n | 78 |
| 26 | : | : | 26 | 79 | SI | o | 79 |
| 27 | ; | ; | 27 | 80 | DLE | p | 80 |
| 28 | < | < | 28 | 81 | DC1 | q | 81 |
| 29 | = | = | 29 | 82 | DC2 | r | 82 |
| 30 | > | > | 30 | 83 | DC3 | s | 83 |
| 31 | ? | ? | 31 | 84 | DC4 | t | 84 |
| 32 | @ | @ | 32 | 85 | NAK | u | 85 |
| 33 | A | A | 33 | 86 | SYN | v | 86 |
| 34 | B | B | 34 | 87 | ETB | w | 87 |
| 35 | C | C | 35 | 88 | CAN | x | 88 |
| 36 | D | D | 36 | 89 | EM | y | 89 |
| 37 | E | E | 37 | 90 | SUB | z | 90 |
| 38 | F | F | 38 | 91 | ESC | { | 91 |
| 39 | G | G | 39 | 92 | FS | \| | 92 |
| 40 | H | H | 40 | 93 | GS | } | 93 |
| 41 | I | I | 41 | 94 | RS | ~ | 94 |
| 42 | J | J | 42 | 95 | US | DEL | 95 |
| 43 | K | K | 43 | 96 | FNC3 | FNC3 | 96 |
| 44 | L | L | 44 | 97 | FNC2 | FNC2 | 97 |
| 45 | M | M | 45 | 98 | SHIFT | SHIFT | 98 |
| 46 | N | N | 46 | 99 | Code C | Code C | 99 |
| 47 | O | O | 47 | 100 | Code B | FNC4 | Code B |
| 48 | P | P | 48 | 101 | FNC4 | Code A | Code A |
| 49 | Q | Q | 49 | 102 | FNC1 | FNC1 | FNC1 |
| 50 | R | R | 50 | 103 | | *START (Code A)* | |
| 51 | S | S | 51 | 104 | | *START (Code B)* | |
| 52 | T | T | 52 | 105 | | *START (Code C)* | |

**Example •** Figures A and B are examples of identical bar codes, and Figure C is an example of switching from Subset C to B to A, as follows:

```
^XA
^FO100,75
^BCN,100,Y,N,N
^FDCODE128^FS
^XZ
```

***Figure A: Subset B with no start character***

```
^XA
^FO100,75
^BCN,100,Y,N,N
^FD>:CODE128^FS
^XZ
```

***Figure B: Subset B with start character***

```
Because Code 128 Subset B is the most commonly used
subset, ZPL II defaults to Subset B if no start character
is specified in the data string.
```

```
^XA
^FO50,50
^BY3^BCN,100,Y,N,N
^FD>;382436>6CODE128>752375152^FS
^XZ
```

***Figure C: Switching from Subset C to B to A***

## How ^BC Works Within a ZPL II Script

**^XA** – the first command starts the label format.

**^FO100,75** – the second command sets the field origin at 100 dots across the x-axis and 75 dots down the y-axis from the upper-left corner.

**^BCN,100,Y,N,N** – the third command calls for a Code 128 bar code to be printed with no rotation (N) and a height of 100 dots. An interpretation line is printed (Y) below the bar code (N). No UCC check digit is used (N).

**^FDCODE128^FS** (Figure A) ^FD>:CODE128^FS (Figure B) – the field data command specifies the content of the bar code.

**^XZ** – the last command ends the field data and indicates the end of the label.

The interpretation line prints below the code with the UCC check digit turned off.

The ^FD command for Figure A does not specify any subset, so Subset B is used. In Figure B, the ^FD command specifically calls Subset B with the >: Start Code. Although ZPL II defaults to Code B, it is good practice to include the Invocation Codes in the command.

Code 128 – Subset B is programmed directly as ASCII text, except for values greater than 94 decimal and a few special characters that must be programmed using the invocation codes. Those characters are:

^ > ~

**Example • Code 128 – Subsets A and C**

Code 128, Subsets A and C are programmed in pairs of digits, 00 to 99, in the field data string. For details, see *Table 6 on page 49*.

In Subset A, each pair of digits results in a single character being encoded in the bar code; in Subset C, characters are printed as entered. Figure E below is an example of Subset A (>9 is the Start Code for Subset A).

Nonintegers programmed as the first character of a digit pair (D2) are ignored. However, nonintegers programmed as the second character of a digit pair (2D) invalidate the entire digit pair, and the pair is ignored. An extra unpaired digit in the field data string just before a code shift is also ignored.

Figure C and Figure D below are examples of Subset C. Notice that the bar codes are identical. In the program code for Figure D, the *D* is ignored and the 2 is paired with the 4.

382436

```
^XA
^FO100,75^BY3
^BCN,100,Y,N,N
^FD>;382436^FS
^XZ
```

***Figure C: Subset C with normal data***

382436

```
^XA
^FO100,75^BY3
^BCN,100,Y,N,N
^FD>;38D2436^FS
^XZ
```

***Figure D: Subset C with ignored alpha character***

CODE128

```
^XA
^FO100,75^BY3
^BCN,100,Y,N,N
^FD>935473637171824^FS
^XZ
```

***Figure E: Subset A***

## The UCC/EAN-128 Symbology

The symbology specified for the representation of Application Identifier data is UCC/EAN-128, a variant of Code 128, exclusively reserved to EAN International and the Uniform Code Council (UCC).

**Note •** It is not intended to be used for data to be scanned at the point of sales in retail outlets.

UCC/EAN-128 offers several advantages. It is one of the most complete, alphanumeric, one-dimensional symbologies available today. The use of three different character sets (A, B and C), facilitates the encoding of the full 128 ASCII character set. Code 128 is one of the most compact linear bar code symbologies. Character set C enables numeric data to be represented in a double density mode. In this mode, two digits are represented by only one symbol character saving valuable space. The code is concatenated. That means that multiple AIs and their fields may be combined into a single bar code. The code is also very reliable. Code 128 symbols use two independent self-checking features which improves printing and scanning reliability.

UCC/EAN-128 bar codes always contain a special non-data character known as function 1 (FNC 1), which follows the start character of the bar code. It enables scanners and processing software to auto-discriminate between UCC/EAN-128 and other bar code symbologies, and subsequently only process relevant data.

The UCC/EAN-128 bar code is made up of a leading quiet zone, a Code 128 start character A, B, or C, a FNC 1 character, Data (Application Identifier plus data field), a symbol check character, a stop character, and a trailing quiet zone.

UCC/EAN, UCC/128 are a couple of ways you'll hear someone refer to the code. This just indicates that the code is structured as dictated by the application identifiers that are used.

SSCC (Serial Shipping Container Code) formatted following the data structure layout for Application Identifier 00. See Table 8, *UCC Application Identifier Table* on page 57. It could be 00 which is the SSCC code. The customer needs to let us know what application identifiers are used for their bar code so we can help them.

There are several ways of writing the code to print the code to Application Identifier '00' structure.

## Using N for the mode (m) parameter

**Example** • This example shows with application identifier 00 structure:

| N FOR THE M PARAMETER | ZPL II CODE |
|---|---|
| 00123451234512345120 | ^XA<br>^FO90,200^BY4<br>^BCN,256,Y,N,Y,N<br>^FD>;>80012345123451234512^FS<br>^XZ |

- >;>8' sets it to subset C, function 1
- '00' is the application identifier followed by '17 characters', the check digit is selected using the 'Y' for the (e) parameter to automatically print the 20th character.
- you are not limited to 19 characters with mode set to N

## Using U for the mode (m) parameter

**Example** • The example shows the application identifier 00 format:

| U FOR THE M PARAMETER | ZPL II CODE |
|---|---|
| 00123451234512345120 | ^XA<br>^FO90,200<br>^BY4^BC,256,Y,N,,U<br>^FD0012345123451234512^FS<br>^XZ |

UCC Case Mode

- Choosing `U` selects UCC Case mode. You will have exactly 19 characters available in `^FD`.
- Subset C using FNC1 values are automatically selected.
- Check digit is automatically inserted.

## Using D for the mode (m) parameter

**Example** • This example shows application identifier 00 format ((x.11.x and above):

| D FOR THE M PARAMETER | ZPL II CODE |
|---|---|
| (00) 1 0084423 7449200941 | ^XA<br>^FO50,200^BCN,150,Y,N,,D<br>^FD(00)10084423 7449200940^FS<br>^XZ |

(0 at end of field data is a bogus character that is inserted as a place holder for the check digit the printer will automatically insert.

- Subset C using FNC1 values are automatically selected.
- Parentheses and spaces can be in the field data. '00' application identifier, followed by 17 characters, followed by bogus check digit place holder.
- Check digit is automatically inserted. The printer will automatically calculate the check digit and put it into the bar code and interpretation line.
- The interpretation line will also show the parentheses and spaces but will strip them out from the actual bar code.

## Printing the Interpretation Line

**Example** • This example shows printing the interpretation in a different font with firmware x.11.x and above:

| INTERPRETATION LINE | ZPL II CODE |
|---|---|
| 0012345123451234512 0 | ^XA<br>^FO50,200<br>^A0N,40,30^BCN,150,Y,N,Y<br>^FD>;>80012345123451234512^FS |

The font command (`^A0N,40,30`) can be added and changed to alter the font and size of the interpretation line.

### With firmware version older than x.10.x

- A separate text field needs to be written.
- The interpretation line needs to be turned off.
- `^A0N,50,40` is the font and size selection for the separate text field.

- You have to make sure you enter the correct check digit in the text field.
- Creating a separate text field allows you to format the interpretation line with parentheses and spaces.

| FIRMWARE OLDER THAN X.10.X | ZPL II CODE |
| --- | --- |
| (00) 1 2345123 451234512 0 | ^XA<br>^FO25,25<br>^BCN,150,N,N,Y<br>^FD>;>80012345123451234512^FS<br>^FO100,190<br>^A0N,50,40<br>^FD(00) 1 2345123 451234512 0^FS<br>^XZ |

## Application Identifiers — UCC/EAN APPLICATION IDENTIFIER

An Application Identifier is a prefix code used to identify the meaning and the format of the data that follows it (data field).

There are AIs for identification, traceability, dates, quantity, measurements, locations, and many other types of information.

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.

The UCC/EAN Application Identifiers provide an open standard that can be used and understood by all companies in the trading chain, regardless of the company that originally issued the codes.

**Table 8 • UCC Application Identifier Table**

| Data Content | AI | Plus The Following Data Structure |
|---|---|---|
| Serial Shipping Container Code (SSCC) | 00 | exactly 18 digits |
| Shipping Container Code | 01 | exactly 14 digits |
| Batch Numbers | 10 | up to 20 alpha numerics |
| Production Date (YYMMDD) | 11 | exactly 6 digits |
| Packaging Date (YYMMDD) | 13 | exactly 6 digits |
| Sell By Date (YYMMDD) | 15 | exactly 6 digits |
| Expiration Date (YYMMDD) | 17 | exactly 6 digits |
| Product Variant | 20 | exactly 2 digits |
| Serial Number | 21 | up to 20 alpha numerics |
| HIBCC Quantity, Date, Batch and Link | 22 | up to 29 alpha numerics |
| Lot Number | 23* | up to 19 alpha numerics |
| Quantity Each | 30 | |
| Net Weight (Kilograms) | 310** | exactly 6 digits |
| Length, Meters | 311** | exactly 6 digits |
| Width or Diameter (Meters) | 312** | exactly 6 digits |
| Depths (Meters) | 313** | exactly 6 digits |
| Area (Sq. Meters) | 314** | exactly 6 digits |
| Volume (Liters) | 315** | exactly 6 digits |
| Volume (Cubic Meters) | 316** | exactly 6 digits |
| Net Weight (Pounds) | 320** | exactly 6 digits |
| Customer PO Number | 400 | up to 29 alpha numerics |
| Ship To (Deliver To) Location Code using EAN 13 or DUNS Number with leading zeros | 410 | exactly 13 digits |
| Bill To (Invoice To) Location Code using EAN 13 or DUNS Number with leading zeros | 411 | exactly 13 digits |
| Purchase from | 412 | exactly 13 digits |
| Ship To (Deliver To) Postal Code within single postal authority | 420 | up to 9 alpha numerics |
| Ship To (Deliver To) Postal Code with 3-digit ISO Country Code Prefix | 421 | 3 digits plus up to 9 alpha numerics |
| Roll Products - width, length, core diameter, direction and splices | 8001 | exactly 14 digits |
| Electronic Serial number for cellular mobile phone | 8002 | up to 20 alpha numerics |

**Note •** Table 8 is a partial table showing the application identifiers. For more current and complete information, search the Internet for **UCC Application Identifier**.

For date fields that only need to indicate a year and month, the day field is set to `00`.

* Plus one digit for length indication.

** Plus one digit for decimal point indication.

### Chaining several application identifiers (firmware x.11.x and later)

The FNC1, which is invoked by `>8`, is inserted just before the AI's so that the scanners reading the code sees the FNC1 and knows that an AI follows.

**Example •** This is an example with the mode parameter set to `A` (automatic):

```
^XA
^BY2,2.5,193
^FO33,400
^BCN,,N,N,N,A
^FD>;>80204017773003486100008535>8910001>837252^FS
^FT33,625^AEN,0,0^FD(02)04017773003486(10)0008535(91)
0001(37)252^FS
^XZ
```

**Example •** This is an example with the mode parameter set to `U`:

```
^XA
^BY3,2.5,193
^FO33,200
^BCN,,N,N,N,U
^FD>;>80204017773003486>8100008535>8910001>837252^FS
^FT33,455^A0N,30,30^FD(02)04017773003486(10)0008535(9
1)0001(37)252^FS
^XZ
```

**Example** • This is an example with the mode parameter set to D*:

```
^XA
^PON
^LH0,0
^BY2,2.5,145
^FO218,343
^BCB,,Y,N,N,D
^FD(91)0005886>8(10)0000410549>8(99)05^FS
^XZ
```

D* — When trying to print the last Application Identifier with an odd number of characters, a problem existed when printing EAN128 bar codes using Mode D. The problem was fixed in firmware V60.13.0.6.

# ^BD

## UPS MaxiCode Bar Code

**Description** The ^BD command creates a two-dimensional, optically read (not scanned) code. This symbology was developed by UPS (United Parcel Service).

Notice that there are no additional parameters for this code and it does not generate an interpretation line. The ^BY command has no effect on the UPS MaxiCode bar code. However, the ^CV command can be activated.

**Format** `^BDm,n,t`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| m = mode | *Accepted Values:*<br>2 = structured carrier message: numeric postal code (U.S.)<br>3 = structured carrier message: alphanumeric postal code (non-U.S.)<br>4 = standard symbol, secretary<br>5 = full EEC<br>6 = reader program, secretary<br>*Default Value:* 2 |
| n = symbol number | *Accepted Values:* 1 to 8 can be added in a structured document<br>*Default Value:* 1 |
| t = total number of symbols | *Accepted Values:* 1 to 8, representing the total number of symbols in this sequence<br>*Default Value:* 1 |

**Example •** This is an example of UPS MAXICODE - MODE 2 bar code:

**UPS MAXICODE - MODE 2**

**ZPL II CODE**

```
^XA
^FO50,50
^CVY
^BD^FH^FD001840152382802
[)>_1E01_1D961Z00004951_1DUPSN_
1D_06X610_1D159_1D1234567_1D1/1_
1D_1DY_1D634 ALPHA DR_
1DPITTSBURGH_1DPA_1E_04^FS
^FO30,300^A0,30,30^FDMode2^FS
^XZ
```

## Special Considerations for ^FD when Using ^BD

The ^FD statement is divided into two parts: a high priority message (hpm) and a low priority message (lpm). There are two types of high priority messages. One is for a U.S. Style Postal Code; the other is for a non-U.S. Style Postal Code. The syntax for either of these high priority messages must be exactly as shown or an error message is generated.

**Format** ^FD <hpm><lpm>

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| <hpm> = high priority message (applicable only in Modes 2 and 3) | *Accepted Values:* 0 to 9, except where noted<br>U.S. Style Postal Code (Mode 2)<br>**<hpm>** = aaabbbcccccdddd<br>aaa = three-digit class of service<br>bbb = three-digit country zip code<br>ccccc = five-digit zip code<br>dddd = four-digit zip code extension (if none exists, four zeros (0000) must be entered)<br>**non-U.S. Style Postal Code (Mode 3)**<br>**<hpm>** = aaabbbcccccc<br>aaa = three-digit class of service<br>bbb = three-digit country zip code<br>ccccc = six-digit zip code (A through Z or 0 to 9) |
| <lpm> = low priority message (only applicable in Modes 2 and 3) | GS is used to separate fields in a message (0x1D). RS is used to separate format types (0x1E). EOT is the end of transmission characters.<br>**Message Header** [)>RS<br>**Transportation Data**<br>**Format Header** 01GS96<br>**Tracking Number*** <tracking number><br>**SCAC*** GS<SCAC><br>**UPS Shipper Number** GS<shipper number><br>**Julian Day of Pickup** GS<day of pickup><br>**Shipment ID Number** GS<shipment ID number><br>**Package n/x** GS<n/x><br>**Package Weight** GS<weight><br>**Address Validation** GS<validation><br>**Ship to Street Address** GS<street address><br>**Ship to City** GS<city><br>**Ship to State** GS<state><br>**RS** RS<br>**End of Message** EOT<br>(* Mandatory Data for UPS) |

### Comments

- The formatting of **<hpm>** and **<lpm>** apply only when using Modes 2 and 3. Mode 4, for example, takes whatever data is defined in the `^FD` command and places it in the symbol.
- UPS requires that certain data be present in a defined manner. When formatting MaxiCode data for UPS, always use uppercase characters. When filling in the *fields* in the **<lpm>** for UPS, follow the data size and types specified in *Guide to Bar Coding with UPS.*
- If you do not choose a mode, the default is Mode 2. If you use non-U.S. Postal Codes, you probably get an error message (invalid character or message too short). When using non-U.S. codes, use Mode 3.
- ZPL II doesn't automatically change your mode based on the zip code format.
- When using special characters, such as GS, RS, or EOT, use the `^FH` command to tell ZPL II to use the hexadecimal value following the underscore character ( _ ).

# ^BE

## EAN-13 Bar Code

**Description** The ^BE command is similar to the UPC-A bar code. It is widely used throughout Europe and Japan in the retail marketplace.

The EAN-13 bar code has 12 data characters, one more data character than the UPC-A code. An EAN-13 symbol contains the same number of bars as the UPC-A, but encodes a 13th digit into a parity pattern of the left-hand six digits. This 13th digit, in combination with the 12th digit, represents a country code.

- ^BE supports fixed print ratios.
- Field data (^FD) is limited to exactly 12 characters. ZPL II automatically truncates or pads on the left with zeros to achieve the required number of characters.
- When using JAN-13 (Japanese Article Numbering), a specialized application of EAN-13, the first two non-zero digits sent to the printer must be 49.

**Format** `^BEo,h,f,g`

**Note** • Use Interleaved 2 of 5 for UCC and EAN 14.

**Important** • If additional information about the EAN-13 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

**Example •** This is an example of an EAN-13 bar code:

| EAN-13 BAR CODE | ZPL II CODE |
|---|---|
| 0 000123 456784 | ^XA<br>^FO100,100^BY3<br>^BEN,100,Y,N<br>^FD12345678^FS<br>^XZ |

| EAN-13 BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**Comments** The EAN-13 bar code uses the Mod 10 check-digit scheme for error checking. For more information on Mod 10, see *Mod 10 Check Digit* on page 364.

# ^BF

## Micro-PDF417 Bar Code

**Description** The ^BF command creates a two-dimensional, multi-row, continuous, stacked symbology identical to PDF417, except it replaces the 17-module-wide start and stop patterns and left/right row indicators with a unique set of 10-module-wide row address patterns. These reduce overall symbol width and allow linear scanning at row heights as low as 2X.

Micro-PDF417 is designed for applications with a need for improved area efficiency but without the requirement for PDF417's maximum data capacity. It can be printed only in specific combinations of rows and columns up to a maximum of four data columns by 44 rows.

Field data (^FD) and field hexadecimal (^FH) are limited to:

- 250 7-bit characters
- 150 8-bit characters
- 366 4-bit numeric characters

**Format** ^BFo,h,m

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 9999<br>*Default Value:* value set by ^BY or 10 (if no ^BY value exists). |
| m = mode | *Accepted Values:* 0 to 33 (see *Micro-PDF417 Mode* on page 67)<br>*Default Value:* 0 (see *Micro-PDF417 Mode* on page 67) |

**Example** • This is an example of a Micro-PDF417 bar code:

**MICRO-PDF417 BAR CODE**

**ZPL II CODE**

```
^XA
^FO100,100^BY6
^BFN,8,3
^FDABCDEFGHIJKLMNOPQRSTUV^FS
^XZ
```

### To encode data into a Micro-PDF417 Bar Code, complete these steps:

1. Determine the type of data to be encoded (for example, ASCII characters, numbers, 8-bit data, or a combination).
2. Determine the maximum amount of data to be encoded within the bar code (for example, number of ASCII characters, quantity of numbers, or quantity of 8-bit data characters).
3. Determine the percentage of check digits that are used within the bar code. The higher the percentage of check digits that are used, the more resistant the bar code is to damage — however, the size of the bar code increases.
4. Use the chart *Micro-PDF417 Mode* on page 67 with the information gathered from the questions above to select the mode of the bar code.

**Table 9 • Micro-PDF417 Mode**

| Mode (M) | Number of Data Columns | Number of Data Rows | % of Cws for EC | Max Alpha Characters | Max Digits |
|---|---|---|---|---|---|
| 0 | 1 | 11 | 64 | 6 | 8 |
| 1 | 1 | 14 | 50 | 12 | 17 |
| 2 | 1 | 17 | 41 | 18 | 26 |
| 3 | 1 | 20 | 40 | 22 | 32 |
| 4 | 1 | 24 | 33 | 30 | 44 |
| 5 | 1 | 28 | 29 | 38 | 55 |
| 6 | 2 | 8 | 50 | 14 | 20 |
| 7 | 2 | 11 | 41 | 24 | 35 |
| 8 | 2 | 14 | 32 | 36 | 52 |
| 9 | 2 | 17 | 29 | 46 | 67 |
| 10 | 2 | 20 | 28 | 56 | 82 |
| 11 | 2 | 23 | 28 | 64 | 93 |
| 12 | 2 | 26 | 29 | 72 | 105 |
| 13 | 3 | 6 | 67 | 10 | 14 |
| 14 | 3 | 8 | 58 | 18 | 26 |
| 15 | 3 | 10 | 53 | 26 | 38 |
| 16 | 3 | 12 | 50 | 34 | 49 |
| 17 | 3 | 15 | 47 | 46 | 67 |
| 18 | 3 | 20 | 43 | 66 | 96 |
| 19 | 3 | 26 | 41 | 90 | 132 |
| 20 | 3 | 32 | 40 | 114 | 167 |
| 21 | 3 | 38 | 39 | 138 | 202 |
| 22 | 3 | 44 | 38 | 162 | 237 |
| 23 | 4 | 6 | 50 | 22 | 32 |
| 24 | 4 | 8 | 44 | 34 | 49 |
| 25 | 4 | 10 | 40 | 46 | 67 |
| 26 | 4 | 12 | 38 | 58 | 85 |
| 27 | 4 | 15 | 35 | 76 | 111 |
| 28 | 4 | 20 | 33 | 106 | 155 |
| 29 | 4 | 26 | 31 | 142 | 208 |
| 30 | 4 | 32 | 30 | 178 | 261 |
| 31 | 4 | 38 | 29 | 214 | 313 |
| 32 | 4 | 44 | 28 | 250 | 366 |
| 33 | 4 | 4 | 50 | 14 | 20 |

# ^BI

## Industrial 2 of 5 Bar Codes

**Description** The ^BI command is a discrete, self-checking, continuous numeric symbology. The Industrial 2 of 5 bar code has been in use the longest of the 2 of 5 family of bar codes. Of that family, the Standard 2 of 5 (^BJ) and Interleaved 2 of 5 (^B2) bar codes are also available in ZPL II.

With Industrial 2 of 5, all of the information is contained in the bars. Two bar widths are employed in this code, the wide bar measuring three times the width of the narrow bar.

- ^BI supports a print ratio of 2.0:1 to 3.0:1.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.

**Format** ^BIo,h,f,g

**Important •** If additional information about the Industrial 2 of 5 bar code, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

**Example** • This is an example of an Industrial 2 of 5 bar code:

**INDUSTRIAL 2 OF 5 BAR CODE**

123456

**ZPL II CODE**

```
^XA
^FO100,100^BY3
^BIN,150,Y,N
^FD123456^FS
^XZ
```

| INDUSTRIAL 2 OF 5 BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**Start/Stop (internal)**

# ^BJ

## Standard 2 of 5 Bar Code

**Description** The ^BJ command is a discrete, self-checking, continuous numeric symbology.

With Standard 2 of 5, all of the information is contained in the bars. Two bar widths are employed in this code, the wide bar measuring three times the width of the narrow bar.

- ^BJ supports a print ratio of 2.0:1 to 3.0:1.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.

**Format** ^BJo,h,f,g

**Important •** If additional information about the Standard 2 of 5 bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

**Example** • This is an example of a Standard 2 of 5 bar code:

| STANDARD 2 OF 5 BAR CODE | ZPL II CODE |
|---|---|
| 123456 | ^XA<br>^FO100,100^BY3<br>^BJN,150,Y,N<br>^FD123456^FS<br>^XZ |

| STANDARD 2 OF 5 BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**Start/Stop (automatic)**

# ^BK

## ANSI Codabar Bar Code

**Description** The ANSI Codabar bar code is used in a variety of information processing applications such as libraries, the medical industry, and overnight package delivery companies. This bar code is also known as USD-4 code, NW-7, and 2 of 7 code. It was originally developed for retail price labeling.

Each character in this code is composed of seven elements: four bars and three spaces. Codabar bar codes use two character sets, numeric and control (start and stop) characters.

- `^BK` supports a print ratio of 2.0:1 to 3.0:1.
- Field data (`^FD`) is limited to the width (or length, if rotated) of the label.

**Format** `^BKo,e,h,f,g,k,l`

**Important •** If additional information about the ANSI Codabar bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `o` = orientation | *Accepted Values:*<br>`N` = normal<br>`R` = rotated 90 degrees (clockwise)<br>`I` = inverted 180 degrees<br>`B` = read from bottom up, 270 degrees<br>*Default Value:* current `^FW` value |
| `e` = check digit | *Fixed Value:* `N` |
| `h` = bar code height (in dots) | *Accepted Values:* `1` to `32000`<br>*Default Value:* value set by `^BY` |
| `f` = print interpretation line | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `Y` |
| `g` = print interpretation line above code | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |
| `k` = designates a start character | *Accepted Values:* `A`, `B`, `C`, `D`<br>*Default Value:* `A` |
| `l` = designates stop character | *Accepted Values:* `A`, `B`, `C`, `D`<br>*Default Value:* `A` |

**Example •** This is an example of an ANSI Codabar bar code:

| ANSI CODABAR BAR CODE | ZPL II CODE |
|---|---|
| A123456A | ^XA<br>^FO100,100^BY3<br>^BKN,N,150,Y,N,A,A<br>^FD123456^FS<br>^XZ |

**ANSI CODABAR BAR CODE CHARACTERS**

0 1 2 3 4 5 6 7 8 9

**Control Characters**

\- : . $ / +

**Start/Stop Characters**

A B C D

# ^BL

## LOGMARS Bar Code

**Description** The ^BL command is a special application of Code 39 used by the Department of Defense. LOGMARS is an acronym for Logistics Applications of Automated Marking and Reading Symbols.

- ^BL supports a print ratio of 2.0:1 to 3.0:1.
- Field data (^FD) is limited to the width (or length, if rotated) of the label. Lowercase letters in the ^FD string are converted to the supported uppercase LOGMARS characters.

**Format** ^BLo,h,g

**Important •** If additional information about the LOGMARS bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

**Example** • This is an example of a LOGMARS bar code:

**LOGMARS BAR CODE**

12AB0

**ZPL II CODE**

```
^XA
^FO100,75^BY3
^BLN,100,N
^FD12AB^FS
^XZ
```

**LOGMARS BAR CODE CHARACTERS**

0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- . $ / + %
SPACE

**Comments** The LOGMARS bar code produces a *mandatory* check digit using Mod 43 calculations. For further information on the Mod 43 check digit, see *Mod 43 Check Digit* on page 365.

# ^BM

## MSI Bar Code

**Description** The ^BM command is a pulse-width modulated, continuous, non-self- checking symbology. It is a variant of the Plessey bar code (^BP).

Each character in the MSI bar code is composed of eight elements: four bars and four adjacent spaces.

- ^BM supports a print ratio of 2.0:1 to 3.0:1.
- For the bar code to be valid, field data (^FD) is limited to 1 to 14 digits when parameter e is B, C, or D. ^FD is limited to 1 to 13 digits when parameter e is A, plus a quiet zone.

**Format** ^BMo,e,h,f,g,e2

**Important •** If additional information about the MSI bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| e = check digit selection | *Accepted Values:*<br>A = no check digits<br>B = 1 Mod 10<br>C = 2 Mod 10<br>D = 1 Mod 11 and 1 Mod 10<br>*Default Value:* B |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |

| Parameters | Details |
|---|---|
| `g` = print interpretation line above code | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |
| `e2` = inserts check digit into the interpretation line | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |

**Example** • This is an example of a MSI bar code:

**MSI BAR CODE**

123456

**ZPL II CODE**

```
^XA
^FO100,100^BY3
^BMN,B,100,Y,N,N
^FD123456^FS
^XZ
```

| MSI BAR CODE CHARACTERS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

# ^BO

## Aztec Bar Code Parameters

**Description** The ^BO command creates a two-dimensional matrix symbology made up of square modules arranged around a bulls-eye pattern at the center.

**Note •** The Aztec bar code works with firmware V60.13.0.11A and higher.

**Format** `^BOa,b,c,d,e,f,g`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = orientation | *Accepted Values:*<br>`N` = normal<br>`R` = rotated<br>`I` = inverted 180 degrees<br>`B` = read from bottom up, 270 degrees<br>*Default Value:* current `^FW` value |
| `b` = magnification factor | *Accepted Values:* `1` to `10`<br>*Default Value:*<br>`1` on `150` dpi printers<br>`2` on `200` dpi printers<br>`3` on `300` dpi printers<br>`6` on `600` dpi printers |
| `c` = extended channel interpretation code indicator | *Accepted Values:*<br>`Y` = if data contains ECICs<br>`N` = if data does not contain ECICs.<br>*Default Value:* `N` |
| `d` = error control and symbol size/type indicator | *Accepted Values:*<br>`0` = default error correction level<br>`01` to `99` = error correction percentage (minimum)<br>`101` to `104` = 1 to 4-layer compact symbol<br>`201` to `232` = 1 to 32-layer full-range symbol<br>`300` = a simple Aztec “Rune”<br>*Default Value:* `0` |
| `e` = menu symbol indicator | *Accepted Values:*<br>`Y` = if this symbol is to be a menu (bar code reader initialization) symbol<br>`N` = if it is not a menu symbol<br>*Default Value:* `N` |

| Parameters | Details |
| --- | --- |
| `f` = number of symbols for structured append | *Accepted Values:* `1` through `26`<br>*Default Value:* `1` |
| `g` = *optional* ID field for structured append | The ID field is a text string with 24-character maximum<br>*Default Value:* no ID |

**Example** • This is an example of the `^B0` command:

**ZPL II CODE**

```
^XA
^B0R,7,N,0,N,1,0
^FD 7. This is testing label 7^FS
^XZ
```

**GENERATED LABEL**

# ^BP

## Plessey Bar Code

**Description** The ^BP command is a pulse-width modulated, continuous, non-self- checking symbology.

Each character in the Plessey bar code is composed of eight elements: four bars and four adjacent spaces.

- ^BP supports a print ratio of 2.0:1 to 3.0:1.
- Field data (^FD) is limited to the width (or length, if rotated) of the label.

**Format** `^BPo,e,h,f,g`

**Important •** If additional information about the Plessey bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| e = print check digit | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |
| h = bar code height (in dots) | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

**Example** • This is an example of a Plessey bar code:

| PLESSEY BAR CODE | ZPL II CODE |
|---|---|
| 12345 | ^XA<br>^FO100,100^BY3<br>^BPN,N,100,Y,N<br>^FD12345^FS<br>^XZ |

| PLESSEY BAR CODE CHARACTERS |
|---|
| 0 1 2 3 4 5 6 7 8 9<br>A B C D E F |

# ^BQ

## QR Code Bar Code

**Description** The ^BQ command produces a matrix symbology consisting of an array of nominally square modules arranged in an overall square pattern. A unique pattern at three of the symbol's four corners assists in determining bar code size, position, and inclination.

A wide range of symbol sizes is possible, along with four levels of error correction. User-specified module dimensions provide a wide variety of symbol production techniques.

QR Code Model 1 is the original specification, while QR Code Model 2 is an enhanced form of the symbology. Model 2 provides additional features and can be automatically differentiated from Model 1.

Model 2 is the recommended model and should normally be used.

This bar code is printed using field data specified in a subsequent ^FD string.

Encodable character sets include numeric data, alphanumeric data, 8-bit byte data, and Kanji characters.

**Format** `^BQa,b,c,d,e`

**Important •** If additional information about the QR Code bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = field orientation | *Fixed Value:* normal (^FW has no effect on rotation) |
| b = model | *Accepted Values:* 1 (original) and 2 (enhanced – recommended)<br>*Default Value:* 2 |
| c = magnification factor | *Accepted Values:* 1 to 10<br>*Default Value:*<br>1 on 150 dpi printers<br>2 on 200 dpi printers<br>3 on 300 dpi printers<br>6 on 600 dpi printers |

| Parameters | Details |
|---|---|
| d = H,Q,M,L | *Accepted Values:*<br>H = ultra-high reliability level<br>Q = high reliability level<br>M = standard level<br>L = high density level<br>*Default Value:*<br>Q = if empty<br>M = invalid values |
| e = N,A,B.K | *Accepted Values:* 1 - 7<br>*Default Value:* 7 |

**Example** • This is an example of a QR Code bar code:

**QR CODE BAR CODE**

**ZPL II CODE**

```
^XA
^FO100,100
^BQN,2,10
^FDMM,AAC-42^FS
^XZ
```

On the pages that follow are specific commands for formatting the ^BQ command with the ^FD statements that contain the information to be coded.

### Considerations for ^FD When Using the QR Code:

### QR Switches (formatted into the ^FD field data)

**mixed mode <D>**

D = allows mixing of different types of character modes in one code.

**code No. <01 16>**

Value = subtracted from the Nth number of the divided code (must be two digits).

**No. of divisions <02 16>**

Number of divisions (must be two digits).

**parity data <1 byte>**

Parity data value is obtained by calculating at the input data (the original input data before divided byte-by-byte through the EX-OR operation).

**error correction level <H, Q, M, L>**

H = ultra-high reliability level

Q = high reliability level

M = standard level (default)

L = high density level

**character Mode <N, A, B, K>**

N = numeric

A = alphanumeric

Bxxxx = 8-bit byte mode. This handles the 8-bit Latin/Kana character set in accordance with JIS X 0201 (character values 0x00 to 0xFF).

xxxx = number of data characters is represented by two bytes of BCD code.

K = Kanji — handles only Kanji characters in accordance with the Shift JIS system based on JIS X 0208. This means that all parameters after the character mode *K* should be 16-bit characters. If there are any 8-bit characters (such as ASCII code), an error occurs.

**data character string <Data>**

Follows character mode or it is the last switch in the ^FD statement.

**data input <A, M>**

A = Automatic Input (default). Data character string JIS8 unit, Shift JIS. When the input mode is Automatic Input, the binary codes of 0x80 to 0x9F and 0xE0 to 0xFF cannot be set.

M = Manual Input

Two types of data input mode exist: Automatic (A) and Manual (M). If A is specified, the character mode does not need to be specified. If M is specified, the character mode must be specified.

### ^FD Field Data (Normal Mode)

Automatic Data Input (A) with Switches

```
^FD
<error correction level>A,
<data character string>
^FS
```

**Example •** QR Code, normal mode with automatic data input.

| GENERATED LABEL | ZPL II CODE |
|---|---|
| | `^XA` <br> `^FO20,20^BQ,2,10^FDQA,0123456789ABCD 2D code^FS` <br> `^XZ` |

| | |
|---|---|
| 1 | Q = error correction level |
| 2 | A, = automatic setting |
| 3 | data string character |

## Manual Data Input (M) with Switches

```
^FD
<error correction level>M,
<character mode><data character string>
^FS
```

**Example •** QR Code, normal mode with manual data input:

**GENERATED LABEL**

**ZPL II CODE**

```
^XA
^FO20,20^BQ,2,10
^FDHM,N123456789012345^FS
^XZ
```

| | |
|---|---|
| **1** | H = error correction level (ultra-high reliability level |
| **2** | M, = input mode (manual input) |
| **3** | N = character mode (numeric data) |
| **4** | data character string |

**Example •** QR Code, normal mode with standard reliability and manual data input:

**GENERATED LABEL**

**ZPL II CODE**

```
^XA
^FO20,20^BQ,2,10^FDMM,AAC-42^FS
^XZ
```

| | |
|---|---|
| **1** | M = error correction level (standard-high reliability level |
| **2** | M, = manual input |
| **3** | AC-42 = data character string |
| **4** | A = alphanumeric data |

## ^FD Field Data (Mixed Mode – requires more switches)

### Automatic Data Input (A) with Switches

```
^FD
<D><code No.> <No. of divisions> <parity data>,
<error correction level> A,
<data character string>,
<data character string>,
<  :  >,
<data character string n**>
^FS
```

### Manual Data Input (M) with Switches

```
^FD
<code No.> <No. of divisions> <parity data>,
<error correction level> M,
<character mode 1> <data character string 1>,
<character mode 2> <data character string 2>,
< :  > <  :  >,
<character mode n> <data character string n**>
^FS
```

n** up to 200 in mixed mode

**Example •** QR Code, mixed mode with manual data input:

| GENERATED LABEL | ZPL II CODE |
|---|---|
| | ^XA<br>^FO,20,20^BQ,2,10<br>^FDD03048F,LM,N0123456789,A12AABB,B0006qrcode^FS<br>^XZ |

| | | |
|---|---|---|
| **<mixed mode identifier>** | D | (mixed) |
| **<code No.>** | M | (code number) |
| **<No. of divisions>** | D | (divisions) |
| **<parity data>** | M | (0x8F) |
| | ‘ | |
| **<error correction level>** | L | (high-density level) |
| **<input mode>** | M | (manual input) |
| | ‘ | |
| **<character mode>** | N | (numeric data) |
| **<data character string>** | | 0123456789 |
| | ‘ | |
| **<character mode>** | A | (alphanumeric data) |
| **<data character string>** | | 12AABB |
| | ‘ | |
| **<character mode>** | B | (8-bit byte data) |
| | 0006 | (number of bytes) |
| **<data character string>** | | qrcode |

**Example •** This is an example of QR Code, mixed mode with automatic data input:

```
^XA
^FO20,20^BQ,2,10
^FDD03048F,LA,012345678912AABBqrcode^FS
^XZ
```

| | | |
|---|---|---|
| **<mixed mode identifier>** | D | (mixed) |
| **<code No.>** | M | (code number) |
| **<No. of divisions>** | D | (divisions) |
| **<parity data>** | M | (0x8F) |
| **<error correction level>** | L | (high-density level) |
| **<input mode>** | A | (automatic input) |
| **<data character string>** | | 012345678912AABBqrcode |

# ^BR

## RSS (Reduced Space Symbology) Bar Code

**Description** The ^BR command is bar code types for space-constrained identification from EAN International and the Uniform Code Council, Inc.

**Format** ^BRa,b,c,d,e,f

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = orientation | *Accepted Values:*<br>N = Normal<br>R = Rotated<br>I = Inverted<br>B = Bottom-up<br>*Default Value:* R |
| b = symbology type in the RSS-14 family | *Accepted Values:*<br>1 = RSS14<br>2 = RSS14 Truncated<br>3 = RSS14 Stacked<br>4 = RSS14 Stacked Omnidirectional<br>5 = RSS Limited<br>6 = RSS Expanded<br>7 = UPC-A<br>8 = UPC-E<br>9 = EAN-13<br>10 = EAN-8<br>11 = UCC/EAN-128 & CC-A/B<br>12 = UCC/EAN-128 & CC-C<br>*Default Value:* 1 |
| c = magnification factor | *Accepted Values:* 1 to 10<br>*Default Value:*<br>24 dot = 6, 12 dot is 3, 8 dot and lower is 2<br>12 dot = 6, > 8 dot is 3, 8 dot and less is 2) |
| d = separator height | *Accepted Values:* 1 or 2<br>*Default Value:* 1 |

| Parameters | Details |
|---|---|
| e = bar code height | The bar code height only affects the linear portion of the bar code. Only UCC/EAN & CC-A/B/C.<br>*Accepted Values:* 1 to 32000 dots<br>*Default Value:* 25 |
| f = the segment width (RSS expanded only) | *Accepted Values:* 2 to 22, even numbers only, in segments per line<br>*Default Value:* 22 |

**Example •** This is an example of Symbology Type 7 - UPC-A:

ZPL II CODE

```
^XA
^FO10,10^BRN,7,5,2,100
^FD12345678901|this is composite info^FS
^XZ
```

GENERATED LABEL

**Example •** This is an example of Symbology Type 1 - RSS14:

ZPL II CODE

```
^XA
^FO10,10^BRN,1,5,2,100
^FD12345678901|this is composite info^FS
^XZ
```

GENERATED LABEL

# ^BS

## UPC/EAN Extensions

**Description** The ^BS command is the two-digit and five-digit add-on used primarily by publishers to create bar codes for ISBNs (International Standard Book Numbers). These extensions are handled as separate bar codes.

The ^BS command is designed to be used with the UPC-A bar code (^BU) and the UPC-E bar code (^B9).

- ^BS supports a fixed print ratio.
- Field data (^FD) is limited to exactly two or five characters. ZPL II automatically truncates or pads on the left with zeros to achieve the required number of characters.

**Format** ^BSo,h,f,g

**Important** • If additional information about the UPC/EAN bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |

**Example •** This is an example of a UPC/EAN Two-digit bar code:

| UPC/EAN 2-DIGIT BAR CODE | ZPL II CODE |
|---|---|
| 12 | `^XA`<br>`^FO100,100^BY3`<br>`^BSN,100,Y,N`<br>`^FD12^FS`<br>`^XZ` |

| UPC/EAN 2-DIGIT BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**Example •** This is an example of a UPC/EAN Five-digit bar code:

| UPC/EAN 5-DIGIT BAR CODE | ZPL II CODE |
|---|---|
| 12345 | `^XA`<br>`^FO100,100^BY3`<br>`^BSN,100,Y,N`<br>`^FD12345^FS`<br>`^XZ` |

| UPC/EAN 5-DIGIT BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Care should be taken in positioning the UPC/EAN extension with respect to the UPC-A or UPC-E code to ensure the resulting composite code is within the UPC specification.

For UPC codes, with a module width of **2** (default), the field origin offsets for the extension are:

**Example** • This is an example of a UPC-A:

| | Supplement Origin X - Offset | Adjustment Y - Offset |
|---|---|---|
| ***Normal*** | 209 Dots | 21 Dots |
| ***Rotated*** | 0 | 209 Dots |

This is an example of a UPC-E:

| | Supplement Origin X - Offset | Adjustment Y - Offset |
|---|---|---|
| ***Normal*** | 122 Dots | 21 Dots |
| ***Rotated*** | 0 | 122 Dots |

Additionally, the bar code height for the extension should be 27 dots (0.135 inches) shorter than that of the primary code. A primary UPC code height of 183 dots (0.900 inches) requires an extension height of 155 dots (0.765 inches).

**Example** • This example illustrates how to create a normal UPC-A bar code for the value 7000002198 with an extension equal to 04414:

**UPC-A BAR CODE WITH EXTENSION**

04414

0 70000 02198 5

**ZPL II CODE**

```
^XA
^FO100,100^BY3
^BUN,137
^FD07000002198^FS
^FO400,121
^BSN,117
^FD04414^FS
^XZ
```

# ^BT

## TLC39 bar code

**Description** The ^BT bar code is the standard for the TCIF can tag telecommunications equipment.

The TCIF CLEI code, which is the Micro-PDF417 bar code, is always four columns. The firmware must determine what mode to use based on the number of characters to be encoded.

**Format** ^BTo,w1,r1,h1,w2,h2

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated<br>I = inverted<br>B = bottom up |
| w1 = width of the Code 39 bar code | *Accepted Value (in dots):* 1 to 10<br>*Default Value (600 dpi printers):* 4<br>*Default Value (200- and 300 dpi printer):* 2 |
| r1 = wide to narrow bar width ratio the Code 39 bar code | *Accepted Values:* 2.0 to 3.0(increments of 0.1)<br>*Default Value:* 2.0 |
| h1 = height of the Code 39 bar code | *Accepted Values (in dots):* 1 to 9999<br>*Default Value (600 dpi printer):* 120<br>*Default Value (300 dpi printer):* 60<br>*Default Value (200 dpi printer):* 40 |
| h2 = row height of the Micro-PDF417 bar ode | *Accepted Values (in dots):* 1 to 255<br>*Default Value (600 dpi printer):* 8<br>*Default Value (200- and 300 dpi printers):* 4 |
| w2 = narrow bar width of the Micro-PDF417 bar code | *Accepted Values (in dots):* 1 to 10<br>*Default Value (600 dpi printer):* 4<br>*Default Value (200- and 300 dpi printers):* 2 |

**Example • TLC39 Bar Code**

This is an example on how to print TLC39 bar code. The callouts identify the key components and are followed by a detailed description below:

Use the command defaults to get results that are in compliance with TCIF industry standards; regardless of printhead density.

```
                   1              2
^XA^FO100,100^BT^FD123456,ABCD12345678901234,
5551212,888999^FS^XZ
       3
```

**1** — **ECI Number.** If the seventh character is **not** a comma, only Code 39 prints. This means if more than 6 digits are present, Code 39 prints for the first six digits (and no Micro-PDF symbol is printed).

- Must be 6 digits.
- Firmware generates invalid character error if the firmware sees anything but 6 digits.
- This number is not padded.

**2** — **Serial number.** The serial number can contain up to 25 characters and is variable length. The serial number is stored in the Micro-PDF symbol. If a comma follows the serial number, then additional data is used below.

- If present, must be alphanumeric (letters and numbers, no punctuation).

This value is used if a comma follows the ECI number.

**3** — **Additional data.** If present, it is used for things such as a country code.

Data cannot exceed 150 bytes. This includes serial number commas.

- Additional data is stored in the Micro-PDF symbol and appended after the serial number. A comma must exist between each maximum of 25 characters in the additional fields.
- Additional data fields can contain up to 25 alphanumeric characters per field.

The result is:

**ZPL II CODE**

```
^XA^FO100,
100^BT^FD123456,
ABCd12345678901234,
5551212,
88899
^FS^XZ
```

**GENERATED LABEL**

# ^BU

## UPC-A Bar Code

**Description** The ^BU command produces a fixed length, numeric symbology. It is primarily used in the retail industry for labeling packages. The UPC-A bar code has 11 data characters. The 6 dot/mm, 12 dot/mm, and 24 dot/mm printheads produce the UPC-A bar code (UPC/EAN symbologies) at 100 percent size. However, an 8 dot/mm printhead produces the UPC/EAN symbologies at a magnification factor of 77 percent.

- ^BU supports a fixed print ratio.
- Field data (^FD) is limited to exactly 11 characters. ZPL II automatically truncates or pads on the left with zeros to achieve required number of characters.

**Format** ^BUo,h,f,g,e

**Important •** If additional information about the UPC-A bar code is required, go to www.aimglobal.org.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = bar code height (in dots) | *Accepted Values:* 1 to 9999<br>*Default Value:* value set by ^BY |
| f = print interpretation line | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| g = print interpretation line above code | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |
| e = print check digit | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |

The font style of the interpretation line depends on the modulus (width of narrow bar) selected in ^BY:

**Note •** Zero is not allowed.

- **6 dot/mm printer:** a modulus of 2 dots or greater prints with an OCR-B interpretation line; a modulus of 1 dot prints font A.
- **8 dot/mm printer:** a modulus of 3 dots or greater prints with an OCR-B interpretation line; a modulus of 1 or 2 dots prints font A.
- **12 dot/mm printer:** a modulus of 5 dots or greater prints with an OCR-B interpretation line; a modulus of 1, 2, 3, or 4 dots prints font A.
- **24 dot/mm printer:** a modulus of 9 dots or greater prints with an OCR-B interpretation line; a modulus of 1 to 8 dots prints font A.

**Example •** This is an example of a UPC-A bar code with extension:

**UPC-A BAR CODE WITH EXTENSION**

**ZPL II CODE**

```
^XA
^FO100,100^BY3
^BUN,137
^FD07000002198^FS
^FO400,121
^BSN,117
^FD04414^FS
^XZ
```

**Comments** The UPC-A bar code uses the Mod 10 check digit scheme for error checking. For further information on Mod 10, see *Mod 10 Check Digit* on page 364.

# ^BX

## Data Matrix Bar Code

**Description** The ^BX command creates a two-dimensional matrix symbology made up of square modules arranged within a perimeter finder pattern.

The ability to create a rectangular Datamatrix bar code is not available as a ZPL coding option.

**Format** `^BXo,h,s,c,r,f,g`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| o = orientation | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees (clockwise)<br>I = inverted 180 degrees<br>B = read from bottom up, 270 degrees<br>*Default Value:* current ^FW value |
| h = dimensional height of individual symbol elements | *Accepted Values:* 1 to the width of the label<br>The individual elements are square — this parameter specifies both module and row height. If this parameter is zero (or not given), the h parameter (bar height) in ^BY is used as the approximate symbol height. |
| s = quality level | *Accepted Values:* 0, 50, 80, 100, 140, 200<br>*Default Value:* 0<br>*Quality* refers to the amount of data that is added to the symbol for error correction. The AIM specification refers to it as the ECC value. ECC 50, ECC 80, ECC 100, and ECC 140 use convolution encoding; ECC 200 uses Reed-Solomon encoding. For new applications, ECC 200 is recommended. ECC 000-140 should be used only in closed applications where a single party controls both the production and reading of the symbols and is responsible for overall system performance. |

| Parameters | Details |
|---|---|
| `c` = columns to encode | *Accepted Values:* `9` to `49`<br>Odd values only for quality 0 to 140 (10 to 144); even values only for quality 200.<br>Odd values only for quality 0 to 140 (10 to 144); even values only for quality 200. The number of rows and columns in the symbol is automatically determined. You might want to force the number of rows and columns to a larger value to achieve uniform symbol size. In the current implementation, quality 0 to 140 symbols are square, so the larger of the rows or columns supplied are used to force a symbol to that size. If you attempt to force the data into too small of a symbol, no symbol is printed. If a value greater than 49 is entered, the rows or columns value is set to zero and the size is determined normally. If an even value is entered, it generates INVALID-P (invalid parameter). If a value less than 9 but not 0, or if the data is too large for the forced size, no symbol prints; if `^CV` is active, INVALID-L prints. |
| `r` = rows to encode | *Accepted Values:* `9` to `49` |
| `f` = format ID (0 to 6) — not used with quality set at 200 | *Accepted Values:*<br>`1` = field data is numeric + space (0..9,") – No \&''<br>`2` = field data is uppercase alphanumeric + space (A..Z,'') – No \&''<br>`3` = field data is uppercase alphanumeric + space, period, comma, dash, and slash (0..9,A..Z,".-/")<br>`4` = field data is upper-case alphanumeric + space (0..9,A..Z,'') – no \&''<br>`5` = field data is full 128 ASCII 7-bit set<br>`6` = field data is full 256 ISO 8-bit set<br>*Default Value:* `6` |
| `g` = escape sequence control character | *Accepted Values:* any character<br>*Default Value:* `~` (tilde)<br>This parameter is used only if quality 200 is specified. It is the escape character for embedding special control sequences within the field data.<br>**Note •** The *default value* of **`g`** will continue to be underscore (_) for anyone with firmware versions: V60.13.0.12, V60.13.0.12Z, V60.13.0.12B, or V60.13.0.12ZB. |

**Table 10 • Maximum Field Sizes**

| ECC LEVEL | ID = 1 | ID = 2 | ID = 3 | ID = 4 | ID = 5 | ID = 6 |
|---|---|---|---|---|---|---|
| **0** | 596 | 452 | 394 | 413 | 310 | 271 |
| **50** | 457 | 333 | 291 | 305 | 228 | 200 |
| **80** | 402 | 293 | 256 | 268 | 201 | 176 |
| **100** | 300 | 218 | 190 | 200 | 150 | 131 |
| **140** | 144 | 105 | 91 | 96 | 72 | 63 |

Maximum Field Sizes

**Example •** This is an example of a Data Matrix bar code:

**DATA MATRIX BAR CODE**

**ZPL II CODE**

```
^XA
^FO100,100
^BXN,10,200
^FDZEBRA TECHNOLOGIES CORP
333 CORPORATE WOODS PKWY
VERNON HILLS, ILLINOIS
60061-3109^FS
^XZ
```

## Effects of ^BY on ^BX

**w = module width** (no effect)

**r = ratio** (no effect)

**h = height of symbol**

If the dimensions of individual symbol elements are not specified in the ^BD command, the height of symbol value is divided by the required rows/columns, rounded, limited to a minimum value of one, and used as the dimensions of individual symbol elements.

## Field Data (^FD) for ^BX

### Quality 000 to 140

- The \& and || can be used to insert carriage returns, line feeds, and the backslash, similar to the PDF417. Other characters in the control character range can be inserted only by using ^FH. Field data is limited to 596 characters for quality *0* to *140*. Excess field data causes no symbol to print; if ^CV is active, INVALID-L prints. The field data must correspond to a user-specified format ID or no symbol prints; if ^CV is active, INVALID-C prints.
- The maximum field sizes for quality *0* to *140* symbols are shown in the table in the **g** parameter.

### Quality 200

- If more than 3072 bytes are supplied as field data, it is truncated to 3072 bytes. This limits the maximum size of a numeric Data Matrix symbol to less than the 3116 numeric characters that the specification would allow. The maximum alphanumeric capacity is 2335 and the maximum 8-bit byte capacity is 1556.
- If `^FH` is used, field hexadecimal processing takes place before the escape sequence processing described below.
- The underscore is the default escape sequence control character for quality 200 field data. A different escape sequence control character can be selected by using parameter g in the `^BX` command.

The information that follows applies to these versions of firmware: V60.13.0.12, V60.13.0.12Z, V60.13.0.12B, or V60.13.0.12ZB. The input string escape sequences can be embedded in quality 200 field data using the ASCII 95 underscore character ( _ ) or the character entered in parameter `g`:

- `_X` is the shift character for control characters (e.g., `_@=NUL,_G=BEL,_0 is PAD`)
- `_1` to `_3` for FNC characters 1 to 3 (explicit FNC4, upper shift, is not allowed)
- FNC2 (Structured Append) must be followed by nine digits, composed of three-digit numbers with values between 1 and 254, that represent the symbol sequence and file identifier (for example, symbol 3 of 7 with file ID 1001 is represented by `_2214001001`)
- 5NNN is code page NNN where NNN is a three-digit code page value (for example, Code Page 9 is represented by `_5009`)
- `_dNNN` creates ASCII decimal value NNN for a code word (must be three digits)
- _ in data is encoded by __ (two underscores)

The information that follows applies to all other versions of firmware. The input string escape sequences can be embedded in quality 200 field data using the ASCII 7E tilde character (~) or the character entered in parameter `g`:

- `~X` is the shift character for control characters (e.g., `~@=NUL,~G=BEL,~0 is PAD`)
- `~1` to `~3` for FNC characters 1 to 3 (explicit FNC4, upper shift, is not allowed)
- FNC2 (Structured Append) must be followed by nine digits, composed of three-digit numbers with values between 1 and 254, that represent the symbol sequence and file identifier (for example, symbol 3 of 7 with file ID 1001 is represented by `~2214001001`)
- 5NNN is code page NNN where NNN is a three-digit code page value (for example, Code Page 9 is represented by `~5009`)
- `~dNNN` creates ASCII decimal value NNN for a code word (must be three digits)
- ~ in data is encoded by a ~ (tilde)

# ^BY

## Bar Code Field Default

**Description** The ^BY command is used to change the default values for the module width (in dots), the wide bar to narrow bar width ratio and the bar code height (in dots). It can be used as often as necessary within a label format.

**Format** ^BYw,r,h

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| w = module width (in dots) | *Accepted Values:* 1 to 10<br>*Initial Value at power-up:* 2 |
| r = wide bar to narrow bar width ratio | *Accepted Values:* 2.0 to 3.0, in 0.1 increments<br>This parameter has no effect on fixed-ratio bar codes.<br>*Default Value:* 3.0 |
| h = bar code height (in dots) | *Initial Value at power-up:* 10 |

For parameter r, the actual ratio generated is a function of the number of dots in parameter w, module width. See Table 11 on page 104.

**Example** • Set module width (w) to 9 and the ratio (r) to 2.4. The width of the narrow bar is 9 dots wide and the wide bar is 9 by 2.4, or 21.6 dots. However, since the printer rounds out to the nearest dot, the wide bar is actually printed at 22 dots.

This produces a bar code with a ratio of 2.44 (22 divided by 9). This ratio is as close to 2.4 as possible, since only full dots are printed.

Module width and height (w and h) can be changed at anytime with the ^BY command, regardless of the symbology selected.

**Table 11 • Shows module width ratios in dots**

| Ratio Selected (r) | Module Width in Dots (w) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **2.0** | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 |
| **2.1** | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2:1 | 2.1:1 |
| **2.2** | 2:1 | 2:1 | 2:1 | 2:1 | 2.2:1 | 2.16:1 | 2.1:1 | 2.12:1 | 2.1:1 | 2.2:1 |
| **2.3** | 2:1 | 2:1 | 2.3:1 | 2.25:1 | 2.2:1 | 2.16:1 | 2.28:1 | 2.25:1 | 2.2:1 | 2.3:1 |
| **2.4** | 2:1 | 2:1 | 2.3:1 | 2.25:1 | 2.4:1 | 2.3:1 | 2.28:1 | 2.37:1 | 2.3:1 | 2.4:1 |
| **2.5** | 2:1 | 2.5:1 | 2.3:1 | 2.5:1 | 2.4:1 | 2.5:1 | 2.4:1 | 2.5:1 | 2.4:1 | 2.5:1 |
| **2.6** | 2:1 | 2.5:1 | 2.3:1 | 2.5:1 | 2.6:1 | 2.5:1 | 2.57:1 | 2.5:1 | 2.5:1 | 2.6:1 |
| **2.7** | 2:1 | 2.5:1 | 2.6:1 | 2.5:1 | 2.6:1 | 2.6:1 | 2.57:1 | 2.65:1 | 2.6:1 | 2.7:1 |
| **2.8** | 2:1 | 2.5:1 | 2.6:1 | 2.75:1 | 2.8:1 | 2.6:1 | 2.7:1 | 2.75:1 | 2.7:1 | 2.8:1 |
| **2.9** | 2:1 | 2.5:1 | 2.6:1 | 2.75:1 | 2.8:1 | 2.8:1 | 2.85:1 | 2.87:1 | 2.8:1 | 2.9:1 |
| **3.0** | 3:1 | 3:1 | 3:1 | 3:1 | 3:1 | 3:1 | 3:1 | 3:1 | 3:1 | 3:1 |

**Comments** Once a ^BY command is entered into a label format, it stays in effect until another ^BY command is encountered.

# ^BZ

## POSTNET Bar Code

**Description** The POSTNET bar code is used to automate the handling of mail. POSTNET uses a series of five bars, two tall and three short, to represent the digits 0 to 9.

- `^BZ` supports a print ratio of 2.0:1 to 3.0:1.
- Field data (`^FD`) is limited to the width (or length, if rotated) of the label.

**Format** `^BZo,h,f,g`

**Important •** If additional information about the POSTNET bar code is required, go to www.aimglobal.org, or contact the United States Postal Service and ask for Publication 25 — Designing Letter Mail, which includes a full specification for POSTNET. You can also download Publication 25 from:

http://pe.usps.gov/cpim/ftp/pubs/pub25/pub25.pdf

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `o` = orientation | *Accepted Values:*<br>`N` = normal<br>`R` = rotated 90 degrees (clockwise)<br>`I` = inverted 180 degrees<br>`B` = read from bottom up, 270 degrees<br>*Default Value:* current `^FW` value |
| `h` = bar code height (in dots) | *Accepted Values:* `1` to `32000`<br>*Default Value:* value set by `^BY` |
| `f` = print interpretation line | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |
| `g` = print interpretation line above code | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |

**Example** • This is an example of a POSTNET bar code:

| POSTNET BAR CODE | ZPL II CODE |
|---|---|
| 12345 | ^XA<br>^FO100,100^BY3<br>^BZN,40,Y,N<br>^FD12345^FS<br>^XZ |

| POSTNET BAR CODE CHARACTERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

# ^CC ~CC

## Change Carets

**Description** The `^CC` command is used to change the format command prefix. The default prefix is the caret (^).

**Format** `^CCx` or `~CCx`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| x = caret character change | *Accepted Values:* any ASCII character<br>*Default Value:* a parameter is required. If a parameter is not entered, the next character received is the new prefix character. |

**Example •** This is an example of how to change the format prefix to / from a ::

```
^XA
^CC/
/XZ
```

The forward slash (/) is set at the new prefix. Note the `/XZ` ending tag uses the new designated prefix character (/).

**Example •** This is an example of how to change the format prefix from ~ to a `/`:

```
~CC/
/XA/JUS/XZ
```

# ^CD ~CD

## Change Delimiter

**Description** The ^CD and ~CD commands are used to change the delimiter character. This character is used to separate parameter values associated with several ZPL II commands. The default delimiter is a comma (,).

**Format** `^CDa or ~CDa`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = delimiter character change | *Accepted Values:* any ASCII character<br>*Default Value:* a parameter is required. If a parameter is not entered, the next character received is the new prefix character. |

**Example** • This shows how to change the character delimiter to a semi-colon (;):

```
^XA
^FO10,10
^GB10,10,3
^XZ
^XA
^CD;
^FO10;10
^GB10;10;3
^XZ
```

- To save, the JUS command is required. Here is an example using JUS:

```
~CD;
^XA^JUS^XZ
```

# ^CF

## Change Alphanumeric Default Font

**Description** The ^CF command sets the default font used in your printer. You can use the ^CF command to simplify your programs.

**Format** `^CFf,h,w`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| f = specified default font | *Accepted Values:* A through Z and 0 to 9<br>*Initial Value at power-up:* A |
| h = individual character height (in dots) | *Accepted Values:* 0 to 32000<br>*Initial Value at power-up:* 9 |
| w = individual character width (in dots) | *Accepted Values:* 0 to 32000<br>*Initial Value at power-up:* 5 or last permanent saved value |

Parameter f specifies the default font for every alphanumeric field. Parameter h is the default height for every alphanumeric field, and parameter w is the default width value for every alphanumeric field.

The default alphanumeric font is A. If you do not change the alphanumeric default font and do not use any alphanumeric field command (^AF) or enter an invalid font value, any data you specify prints in font A.

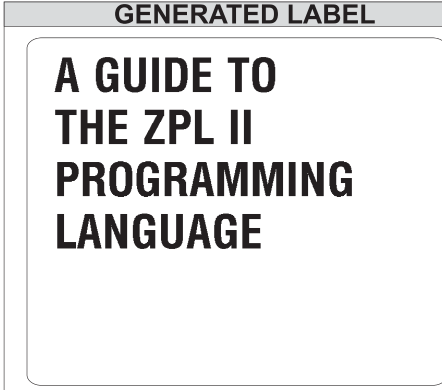
Defining only the height or width forces the magnification to be proportional to the parameter defined. If neither value is defined, the last ^CF values given or the default ^CF values for height and width are used.

**Example** • This is an example of ^CF code and the result of the code:

**ZPL II CODE**

```
^XA
^CF0,89
^FO20,50
^FDA GUIDE TO^FS
^FO20,150
^FDTHE ZPL II^FS
^FO20,250
^FDPROGRAMMING^FS
^FO20,350
^FDLANGUAGE^FS
^XZ
```

**GENERATED LABEL**



**Comments** Any font in the printer, including downloaded fonts, EPROM stored fonts, and fonts A through Z and 0 to 9, can also be selected with `^CW`.

# ^CI

## Change International Font/Encoding

**Description** Zebra printers can print fonts using international character sets: U.S.A.1, U.S.A.2, UK, Holland, Denmark/Norway, Sweden/Finland, Germany, France 1, France 2, Italy, Spain, and several other sets, ..

The `^CI` command enables you to call up the international character set you want to use for printing. You can mix character sets on a label.

This command allows character remapping. Any character within a font can be remapped to a different numerical position.

**Format** `^CIa,s1,d1,s2,d2,...`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = desired character set | Accepted values `0` - `12` are Zebra Code Page 850 with specific character replacements. For details, see *International Character Sets* on page 114 and/or *Zebra Code Page 850* on page 342.<br>*Accepted Values:*<br>`0` = Single Byte Encoding - U.S.A. 1 Character Set<br>`1` = Single Byte Encoding - U.S.A. 2 Character Set<br>`2` = Single Byte Encoding - U.K. Character Set<br>`3` = Single Byte Encoding - Holland Character Set<br>`4` = Single Byte Encoding - Denmark/Norway Character Set<br>`5` = Single Byte Encoding - Sweden/Finland Character Set<br>`6` = Single Byte Encoding - Germany Character Set<br>`7` = Single Byte Encoding - France 1 Character Set<br>`8` = Single Byte Encoding - France 2 Character Set<br>`9` = Single Byte Encoding - Italy Character Set<br>`10` = Single Byte Encoding - Spain Character Set<br>`11` = Single Byte Encoding - Miscellaneous Character Set<br>`12` = Single Byte Encoding - Japan (ASCII with Yen symbol) Character Set<br>`13` = Zebra Code Page 850 (see page 342)<br>`14` = Double Byte Asian Encodings *<br>`15` = Shift-JIS **<br>`16` = EUC-JP and EUC-CN *<br>`17` = Deprecated - UCS-2 Big Endian ****<br>`18` to `23` = Reserved<br>`24` = Single Byte Asian Encodings *<br>`25` = Reserved<br>`26` = Multibyte Asian Encodings with ASCII Transparency * and ***<br>`27` = Zebra Code Page 1252 (see page 344)<br>*Initial Value at power-up:* `0` |
| `s1` = source 1 (character output image) | *Accepted Values:* decimals `0` to `255` |
| `d1` = destination 1 (character input) | *Accepted Values:* decimals `0` to `255` |
| `s2` = source 2 (character output image) | *Accepted Values:* decimals `0` to `255` |

| Parameters | Details |
|---|---|
| d2 = destination 2 (character input) | *Accepted Values:* decimals 0 to 255 |
| ... = continuation of pattern | Up to 256 source and destination pairs can be entered in this command. |

* The encoding is controlled by the conversion table (*.DAT). The table generated by ZTools™ is the TrueType fonts internal encoding (Unicode).

** Shift-JIS encoding converts Shift-JIS to JIS and then looks up the JIS conversion in JIS.DAT. This table must be present for Shift-JIS to function.

*** Supports ASCII transparency for Asian . 7F and less are treated as single byte characters. 80 to FE is treated as the first byte of a 2 byte character 8000 to FEFF in the encoding table for Unicode.

**Example** • This example remaps the Euro symbol (36) decimal to the dollar sign value (21) decimal. When the dollar sign character is sent to the printer, the Euro symbol prints.

```
^CI0,36,21
```

The font selected determines the shape and resolution of the printed symbol.

## International Character Sets

| Hex | 23 | 30 | 40 | 5B | 5C | 5D | 5E | 60 | 7B | 7C | 7D | 7E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CI0 | # | 0 | @ | [ | ¢ | ] | ^ | ` | { | ¦ | } | ~ |
| CI1 | # | 0 | @ | ⅓ | ¢ | ⅔ | ^ | ` | ¼ | ½ | ¾ | ~ |
| CI2 | £ | 0 | @ | [ | ¢ | ] | ^ | ` | { | ¦ | } | ~ |
| CI3 | ƒ | 0 | § | [ | Ĳ | ] | ^ | ` | { | ĳ | } | ~ |
| CI4 | # | 0 | @ | Æ | Ø | Å | ^ | ` | æ | ø | å | ~ |
| CI5 | Ü | 0 | É | Ä | Ö | Å | Ü | é | ä | ö | å | ü |
| CI6 | # | 0 | § | Ä | Ö | Ü | ^ | ` | ä | ö | ü | ß |
| CI7 | £ | 0 | à | [ | ç | ] | ^ | ` | é | ¦ | ù | è |
| CI8 | # | 0 | à | â | ç | ê | î | ô | é | ù | è | û |
| CI9 | £ | 0 | § | [ | ç | é | ^ | ù | à | ò | è | ì |
| CI10 | # | 0 | § | ¡ | Ñ | ¿ | ^ | ` | { | ñ | ç | ~ |
| CI11 | £ | 0 | É | Ä | Ö | Ü | ^ | ä | ë | ï | ö | ü |
| CI12 | # | 0 | @ | [ | ¥ | ] | ^ | ` | { | ¦ | } | ~ |
| CI13 | # | 0 | @ | [ | \ | ] | ^ | ` | { | ¦ | } | ~ |

**Note •** `^CI 13 =` US keyboard

**Comments** The *space* character cannot be remapped for any font.

# ^CM

## Change Memory Letter Designation

**Description** The ^CM command allows you to reassign a letter designation to the printer's memory devices. If a format already exists, you can reassign the memory device to the corresponding letter without forcing, altering, or recreating the format itself.

Using this command affects every subsequent command that refers to specific memory locations.

**Format** ^CMa,b,c,d

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = memory alias for B: | *Accepted Values:* B:, E:,R:, A:, and NONE<br>*Default Value:* B: |
| b = memory alias for E: | *Accepted Values:* B:, E:,R:, A:, and NONE<br>*Default Value:* E: |
| c = memory alias for R: | *Accepted Values:* B:, E:,R:, A:, and NONE<br>*Default Value:* R: |
| d = memory alias for A: | *Accepted Values:* B:, E:,R:, A:, and NONE<br>*Default Value:* A: |

**Comments** If two or more parameters specify the same letter designator, all letter designators are set to their default values.

It is recommended that after entering the ^CM command, ^JUS is entered to save changes to EEPROM. Any duplicate parameters entered reset the letter designations back to the default.

If any of the parameters are out of specification, the command is ignored.

**Example •** This example designates letter E: to point to the B: memory device, and the letter B: to point to the E: memory device.

```
^XA
^CME,B,R,A
^JUS
^XA
```

# ^CO

## Cache On

**Note •** If you have firmware x.12 or greater this command is not required because the printer firmware automatically expands the size of the character cache as needed.

**Description** The `^CO` command is used to change the size of the character cache. By definition, a *character cache* (referred to as cache) is a portion of the DRAM reserved for storing scalable characters. All printers have a default 40K cache that is always turned on. The maximum single character size that can be stored, without changing the size of the cache, is 450 dots by 450 dots.

There are two types of fonts used in Zebra printers: bitmapped and scalable. Letters, numbers, and symbols in a bitmapped font have a fixed size (for example: 10 points, 12 points, 14 points). By comparison, scalable fonts are not fixed in size.

Because their size is fixed, bitmapped fonts can be moved quickly to the label. In contrast, scalable fonts are much slower because each character is built on an as-needed basis before it is moved to the label. By storing scaled characters in a cache, they can be recalled at a much faster speed.

The number of characters that can be stored in the cache depends on two factors: the size of the cache (memory) and the size of the character (in points) being saved. The larger the point size, the more space in the cache it uses. The default cache stores every scalable character that is requested for use on a label. If the same character, with the same rotation and size is used again, it is quickly retrieved from cache.

It is possible that after a while the print cache could become full. Once the cache is full, space for new characters is obtained by eliminating an existing character from the print cache. Existing characters are eliminated by determining how often they have been used. This is done automatically. For example, a 28-point *Q* that was used only once would be a good candidate for elimination from the cache.

Maximum size of a single print cache character is 1500 dots by 1500 dots. This would require a cache of 274K.

When the cache is too small for the desired style, smaller characters might appear but larger characters do not. If possible, increase the size of the cache.

**Format** `^COa,b,c`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = cache on | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |
| b = amount of additional memory to be added to cache (in K) | *Accepted Values:* any size up to total memory available<br>*Default Value:* 40 |
| c = cache type | *Accepted Values:*<br>0 = cache buffer (normal fonts)<br>1 = internal buffer (recommended for Asian fonts)<br>*Default Value:* 0 |

**Example** • To resize the print cache to 62K, assuming a 22K existing cache:

```
^COY,40
```

To resize the print cache to 100K, assuming a 22K existing cache:

```
^COY,78
```

## Print Cache Performance

For printing large characters, memory added to the cache by the ^CO command is not physically added to the 22K cache already in the printer. In the second example above, the resulting 100K cache is actually two separate blocks of memory, 22K and 78K.

Because large characters need contiguous blocks of memory, a character requiring a cache of 90K would not be completely stored because neither portion of the 100K cache is big enough. Therefore, if large characters are needed, the ^CO command should reflect the actual size of the cache you need.

Increasing the size of the cache improves the performance in printing scalable fonts. However, the performance decreases if the size of the cache becomes large and contains too many characters. The performance gained is lost because of the time involved searching cache for each character.

**Comments** The cache can be resized as often as needed. Any characters in the cache when it is resized are lost. Memory used for the cache reduces the space available for label bitmaps, graphic, downloaded fonts, et cetera.

Some Asian fonts require an internal working buffer that is much larger than the normal cache. Since most fonts do not require this larger buffer, it is now a selectable configuration option. Printing with the Asian fonts greatly reduces the printer memory available for labels, graphics, fonts, and formats.

# ^CT ~CT

## Change Tilde

**Description** The ^CT and ~CT commands are used to change the control command prefix. The default prefix is the tilde (~).

**Format** `^CTa or ~CTa`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = change control command character | *Accepted Values:* any ASCII character<br>*Default Value:* a parameter is required. If a parameter is not entered, the next character received is the new control command character. |

**Example** • This is an example of how to change the control command prefix from a ^ to a ~:

```
^XA
^CT+
^XZ
+HS
```

# ^CV

## Code Validation

**Description** The ^CV command acts as a switch to turn the code validation function on and off. When this command is turned on, all bar code data is checked for these error conditions:

- character not in character set
- check-digit incorrect
- data field too long (too many characters)
- data field too short (too few characters)
- parameter string contains incorrect data or missing parameter

When invalid data is detected, an error message and code is printed in reverse image in place of the bar code. The message reads `INVALID - X` where `X` is one of these error codes:

`C` = character not in character set
`E` = check-digit incorrect
`L` = data field too long
`S` = data field too short
`P` = parameter string contains incorrect data
(occurs only on select bar codes)

Once turned on, the ^CV command remains active from format to format until turned off by another ^CV command or the printer is turned off. The command is not permanently saved.

**Format** `^CVa`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = code validation | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |

**Example •** The examples below show the error labels `^CVY` generates when incorrect field data is entered. Compare the letter following *INVALID* – to the listing on the previous page.

| ZPL II CODE | GENERATED LABEL |
|---|---|
| `^XA`<br>`^CVY`<br>`^FO50,50`<br>`^BEN,100,Y,N`<br>`^FD97823456 890^FS`<br>`^XZ` | INVALID - C |
| `^XA`<br>`^CVY`<br>`^FO50,50`<br>`^BEN,100,Y,N`<br>`^FD9782345678907^FS`<br>`^XZ` | INVALID - E |
| `^XA`<br>`^CVY`<br>`^FO50,50`<br>`^BEN,100,Y,N`<br>`^FD97823456789081^FS`<br>`^XZ` | INVALID - L |
| `^XA`<br>`^CVY`<br>`^FO50,50`<br>`^BEN,100,Y,N`<br>`^FD97823456789^FS`<br>`^XZ` | INVALID - S |
| `^XA`<br>`^CVY`<br>`^FO50,50`<br>`^BQN2,3`<br>`^FDHM,BQRCODE-22^FS`<br>`^XZ` | INVALID - P |

**Comments** If more than one error exists, the first error detected is the one displayed.

The `^CV` command tests the integrity of the data encoded into the bar code. It is not used for (or to be confused with) testing the scan-integrity of an image or bar code.

# ^CW

## Font Identifier

**Description** All built-in fonts are referenced using a one-character identifier. The `^CW` command assigns a single alphanumeric character to a font stored in DRAM, memory card, EPROM, or Flash.

If the assigned character is the same as that of a built-in font, the downloaded font is used in place of the built-in font. The new font is printed on the label wherever the format calls for the built-in font. If used in place of a built-in font, the change is in effect only until power is turned off.

If the assigned character is different, the downloaded font is used as an additional font. The assignment remains in effect until a new command is issued or the printer is turned off.

**Format** `^CWa,d:o.x`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = letter of existing font to be substituted, or new font to be added | *Accepted Values:* `A` through `Z` and `0` to `9`<br>*Default Value:* a one-character entry is required |
| `d` = device to store font in (optional) | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* `R:` |
| `o` = name of the downloaded font to be substituted for the built-in, or as an additional font | *Accepted Values:* any name up to 8 characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `x` = extension | *Accepted Values:*<br>`.FNT` = Font<br>`.TTF` = TrueType Font |

**Example •** These examples show how to use:

- `MYFONT.FNT` stored in DRAM whenever a format calls for Font A:

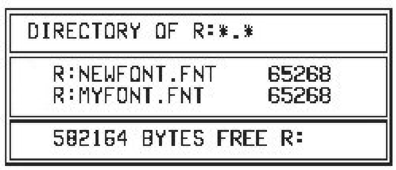
```
^XA
^CWA,R:MYFONT.FNT
^XZ
```

- `MYFONT.FNT` stored in DRAM additionally as Font Q:

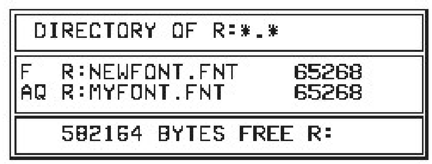
```
^XA
^CWQ,R:MYFONT.FNT
^XZ
```

- `NEWFONT.FNT` stored in DRAM whenever a format calls for font F:

```
^XA
^CWF,R:NEWFONT.FNT
^XZ
```




**Label Listing Before Assignment**




**Label Listing After Assignment**

# ~DB

## Download Bitmap Font

**Description** The `~DB` command sets the printer to receive a downloaded bitmap font and defines native cell size, baseline, space size, and copyright.

This command consists of two portions, a ZPL II command defining the font and a structured data segment that defines each character of the font.

**Format** `~DBd:o.x,a,h,w,base,space,#char,©,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = drive to store font | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* `R:` |
| `o` = name of font | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `x` = extension | *Fixed Value:* `.FNT` |
| `a` = orientation of native font | *Fixed Value:* normal |
| `h` = maximum height of cell (in dots) | *Accepted Values:* `1` to `32000`<br>*Default Value:* a value must be specified |
| `w` = maximum width of cell (in dots) | *Accepted Values:* `1` to `32000`<br>*Default Value:* a value must be specified |
| `base` = dots from top of cell to character baseline | *Accepted Values:* `1` to `32000`<br>*Default Value:* a value must be specified |
| `space` = width of space or non-existent characters | *Accepted Values:* `1` to `32000`<br>*Default Value:* a value must be specified |
| `#char` = number of characters in font | *Accepted Values:* `1` to `256` (must match the characters being downloaded)<br>*Default Value:* a value must be specified |

| Parameters | Details |
|---|---|
| © = copyright holder | *Accepted Values:* `1` to `63` alphanumeric characters<br>*Default Value:* a value must be specified |
| `data` = structured ASCII data that defines each character in the font | The # symbol signifies character code parameters, which are separated with periods. The character code is from 1 to 4 characters to allow for large international character sets to be downloaded to the printer.<br>The data structure is:<br>#xxxx.h.w.x.y.i.data<br>`#xxxx` = character code<br>`h` = bitmap height (in dot rows)<br>`w` = bitmap width (in dot rows)<br>`x` = x-offset (in dots)<br>`y` = y-offset (in dots)<br>`i` = typesetting motion displacement (width, including inter character gap of a particular character in the font)<br>`data` = hexadecimal bitmap description |

**Example •** This is an example of how to use the `~DB` command. It shows the first two characters of a font being downloaded to DRAM.

```
~DBR:TIMES.FNT,N,5,24,3,10,2,ZEBRA 1992,
#0025.5.16.2.5.18.
OOFF
OOFF
FFOO
FFOO
FFFF
#0037.4.24.3.6.26.
OOFFOO
OFOOFO
OFOOFO
OOFFOO
```

# ~DE

## Download Encoding

**Description** The standard encoding for TrueType Windows® fonts is always Unicode. The ZPL II field data must be converted from some other encoding to Unicode that the Zebra printer understands. The required translation tables are provided with font packs. Some tables can be downloaded from www.zebra.com.

**Format** `~DEd:o.x,s,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = location of table | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* `R:` |
| `o` = name of table | *Accepted Values:* any valid name, up to 8 characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `x` = extension | *Fixed Value:* `.DAT` |
| `s` = table size | *Accepted Values:* the number of memory bytes required to hold the Zebra downloadable format of the font<br>*Default Value:* if an incorrect value or no value is entered, the command is ignored |
| `data` = data string | *Accepted Values:* a string of ASCII hexadecimal values<br>*Default Value:* if no data is entered, the command is ignored |

**Example •** This is an example of how to download the required translation table:

```
~DER:JIS.DAT,27848,300021213001...
```

(27848 two-digit hexadecimal values)

**Comments** For more information on ZTools or ZebraNet Bridge, see the program documentation included with the software.

For assistance with editing or adding mappings to `.DAT` tables, ZebraNet Bridge includes a `.DAT` table editor in the font wizard.

Encoding scheme for the data sent to the printer is the second four character and the encoding scheme for the font is the first four characters throughout the `.DAT` file. The data must be ordered by the second four characters (the encoding table).

# ^DF

## Download Format

**Description** The ^DF command saves ZPL II format commands as text strings to be later merged using ^XF with variable data. The format to be stored might contain field number (^FN) commands to be referenced when recalled.

While use of stored formats reduces transmission time, no formatting time is saved—this command saves ZPL II as text strings formatted at print time.

Enter the ^DF stored format command immediately after the ^XA command, then enter the format commands to be saved.

**Format** `^DFd:o.x`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = device to store image | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = image name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Fixed Value:* .ZPL |

For a complete example of the ^DF and ^XF command, see *^DF and ^XF — Download format and recall format* on page 11.

**Example •** This example is generated using the ^XF command to recall this format:

**ZPL II CODE**

```
^XA
^DFR:STOREFMT.ZPL^FS
^FO25,25
^AD,36,20^FN1^FS
^FO165,25
^AD,36,20^FN2^FS
^FO25,75
^AB,22,14^FDBUILT BY^FS
^FO25,125
^AE,28,15^FN1
^XZ
^XA
^XFR:STOREFMT.ZPL^FS
^FN1^FDZEBRA^FS
^XZ
```

**GENERATED LABEL**

ZEBRA PRINTER
BUILT BY
ZEBRA

# ~DG

## Download Graphics

**Description** The ~DG command downloads an ASCII Hex representation of a graphic image. If .GRF is not the specified file extension, .GRF is automatically appended.

For more saving and loading options when downloading files, see *~DY* on page 134.

**Format** `~DGd:o.x,t,w,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = device to store image | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = image name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Fixed Value:* .GRF |
| t = total number of bytes in graphic | See the formula in the examples below. |
| w = number of bytes per row | See the formula in the examples below. |
| data = ASCII hexadecimal string defining image | The data string defines the image and is an ASCII hexadecimal representation of the image. Each character represents a horizontal nibble of four dots. |

This is the key for the examples that follow:

x = width of the graphic in millimeters
y = height of the graphic in millimeters
z = dots/mm = print density of the printer being programmed
8 = bits/byte

**Examples •** These are some example related to the `~DG` command:

To determine the `t` parameter use this formula:

$$\frac{xz}{8} \times yz = \text{totalbytes}$$

To determine the correct t parameter for a graphic 8 mm wide, 16 mm high, and a print density of 8 dots/mm, use this formula:

$$8 \times 128 = 1024$$
$$t = 1024$$

**Raise any portion of a byte to the next whole byte.**

To determine the `w` parameter (the width in terms of bytes per row) use this formula:

$$w = 8$$

To determine the correct `w` parameter for a graphic 8 mm wide and a print density of 8 dots/mm, use this formula:

$$w = 8$$

**Raise any portion of a byte to the next whole byte.**

Parameter `w` is the first value in the `t` calculation.

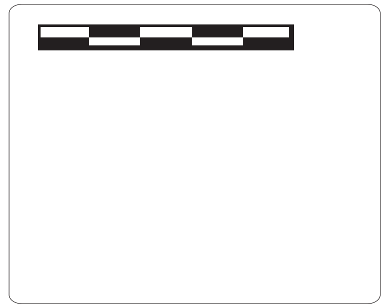
The data parameter is a string of hexadecimal numbers sent as a representation of the graphic image. Each hexadecimal character represents a horizontal nibble of four dots. For example, if the first four dots of the graphic image are white and the next four black, the dot-by-dot binary code is 00001111. The hexadecimal representation of this binary value is 0F. The entire graphic image is coded in this way, and the complete graphic image is sent as one continuous string of hexadecimal values.

This is an example of using the ~DG command to load a checkerboard pattern into DRAM. The name used to store the graphic is SAMPLE.GRF:

**ZPL II CODE**

```
~DGR:SAMPLE.GRF,00080,010,
FFFFFFFFFFFFFFFFFFFF
8000FFFF0000FFFF0001
8000FFFF0000FFFF0001
8000FFFF0000FFFF0001
FFFF0000FFFF0000FFFF
FFFF0000FFFF0000FFFF
FFFF0000FFFF0000FFFF
FFFFFFFFFFFFFFFFFFFF
^XA
^F020,20^XGR:SAMPLE.GRF,1,1^FS
^XZ
```

**GENERATED LABEL**



**Comments** Do not use spaces or periods when naming your graphics. Always use different names for different graphics.

If two graphics with the same name are sent to the printer, the first graphic is erased and replaced by the second graphic.

# ~DN

## Abort Download Graphic

**Description** After decoding and printing the number of bytes in parameter `t` of the `~DG` command, the printer returns to normal Print Mode. Graphics Mode can be aborted and normal printer operation resumed by using the `~DN` command.

**Format** `~DN`

**Comments** If you need to stop a graphic from downloading, you should abort the transmission from the host device. To clear the `~DG` command, however, you must send a `~DN` command.

# ~DS

## Download Intellifont (Scalable Font)

**Description** The ~DS command is used to set the printer to receive a downloadable scalable font and defines the size of the font in bytes.

The ~DS command, and its associated parameters, is the result of converting a vendor-supplied font for use on a Zebra printer. To convert this font use the ZTools utility.

**Format** `~DSd:o.x,s,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = device to store image | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = image name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Fixed Value:* .FNT |
| s = size of font in bytes | *Fixed Value:* this number is generated by ZTools and should not be changed |
| data = ASCII hexadecimal string that defines font | *Fixed Value:* this number is generated by ZTools and should not be changed |

**Example** • This example shows the first three lines of a scalable font that was converted using the ZTools program and is ready to be downloaded to the printer. If necessary, the destination and object name can be changed.

```
~DSB:CGTIMES.FNT,37080,
OOFFOOFFOOFFOOFF
FFOAECB28FFFOOFF
```

**Comments** Downloaded scalable fonts are not checked for integrity. If they are corrupt, they cause unpredictable results at the printer.

# ~DT

## Download Bounded TrueType Font

**Description** Use ZTools to convert a TrueType font to a Zebra-downloadable format. ZTools creates a downloadable file that includes a `~DT` command. For information on converting and downloading Intellifont information, see *~DS* on page 131.

**Format** `~DTd:o.x,s,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = font location | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* `R:` |
| `o` = font name | *Accepted Values:* any valid TrueType name, up to 8 characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `x` = extension | *Fixed Value:* `.DAT` |
| `s` = font size | *Accepted Values:* the number of memory bytes required to hold the Zebra-downloadable format of the font<br>*Default Value:* if an incorrect value or no value is entered, the command is ignored |
| `data` = data string | *Accepted Values:* a string of ASCII hexadecimal values (two hexadecimal digits/byte). The total number of two-digit values must match parameter `s`.<br>*Default Value:* if no data is entered, the command is ignored |

**Example •** This is an example of how to download a true type font:

```
~DTR:FONT,52010,00AF01B0C65E...
```

(52010 two-digit hexadecimal values)

# ~DU

## Download Unbounded TrueType Font

**Description** Some international fonts, such as Asian fonts, have more than 256 printable characters. These fonts are supported as *large TrueType fonts* and are downloaded to the printer with the `~DU` command. Use ZTools to convert the large TrueType fonts to a Zebra-downloadable format.

The Field Block (`^FB`) command cannot support the large TrueType fonts.

**Format** `~DUd:o.x,s,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = font location | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* `R:` |
| `o` = font name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `x` = extension | *Fixed Value:* `.FNT` |
| `s` = font size | *Accepted Values:* the number of memory bytes required to hold the Zebra-downloadable format of the font<br>*Default Value:* if no data is entered, the command is ignored |
| `data` = data string | *Accepted Values:* a string of ASCII hexadecimal values (two hexadecimal digits/byte). The total number of two-digit values must match parameter `s`.<br>*Default Value:* if no data is entered, the command is ignored |

**Example** • This is an example of how to download an unbounded true type font:

```
~DUR:KANJI,86753,60CA017B0CE7...
```

(86753 two-digit hexadecimal values)

For similar commands, see *~DS* on page 131, *~DT* on page 132, and *~DY* on page 134.

# ~DY

## Download Graphics / Native TrueType or OpenType Font

**Description** The `~DY` command downloads to the printer graphic objects or fonts in any supported format. This command can be used in place of `~DG` for more saving and loading options. `~DY` is the preferred command to download TrueType fonts on printers with firmware greater than X.13. It is faster than `~DU`.

**Format** `~DYd:f,b,x,t,w,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = file location | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* `R:` |
| `f` = file name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `b` = format downloaded in data field | *Accepted Values:*<br>`A` = uncompressed (ZB64, ASCII)<br>`B` = uncompressed (binary)<br>`C` = AR-compressed (used only by Zebra's BAR-ONE® v5)<br>`P` = portable network graphic (`.PNG`) - ZB64 encoded<br>*Default Value:* a value must be specified |
| `x` = extension of stored file | *Accepted Values:*<br>`B` = bitmap<br>`G` = raw bitmap (`.GRF`)<br>`P` = store as compressed (`.PNG`)<br>`T` = TrueType (`.TTF`) or `X` = Paintbrush (`.PCX`)<br>*Default Value:* a value other than the accepted values defaults to `.GRF` |

| Parameters | Details |
|---|---|
| t = total number of bytes in file | *Accepted Values:*<br>`.BMP`<br>This parameter refers to the actual size of the file, not the amount of disk space.<br>`.GRF` images: the size after decompression into memory<br>This parameter refers to the actual size of the file, not the amount of disk space.<br>`.PCX`<br>This parameter refers to the actual size of the file, not the amount of disk space.<br>`.PNG` images:<br>This parameter refers to the actual size of the file, not the amount of disk space.<br>`.TTF`<br>This parameter refers to the actual size of the file, not the amount of disk space. |
| `w` = total number of bytes per row | *Accepted Values:*<br>`.GRF` images: number of bytes per row<br>`.PNG` images: value ignored<br>`.TTF`images: value ignored<br>images: value ignored |
| `data` = data | ASCII hexadecimal encoding, ZB64, or binary data, depending on `b`.<br>`A, P` = ASCII hexadecimal or ZB64<br>`B, C` = binary<br>When binary data is sent, all control prefixes and flow control characters are ignored until the total number of bytes needed for the graphic format is received. |

**Note •** The `~DY` command and the binary font file or binary graphic file can be sent as two separate files. If the files are sent to the printer separately, the data light might remain lit until the printer receives the TrueType font file. **SIZE number** of bytes must be received for the download to complete successfully. The graphic files must be monochrome (black and white).

**Example •** This is an example of how to download a binary TrueType Font file of Size bytes using the name fontfile.ttf and storing it to permanent flash memory on the printer:

```
~DYE:FONTFILE.TTF,B,T,SIZE,,
```

**Comments** For more information on ZB64 encoding and compression, see *ZB64 Encoding and Compression* on page 367.

# ~EG

## Erase Download Graphics

See ^ID on page 181.

# ^FB

## Field Block

**Description** The ^FB command allows you to print text into a defined *block type* format. This command formats an ^FD or ^SN string into a block of text using the origin, font, and rotation specified for the text string. The ^FB command also contains an automatic word-wrap function.

**Format** `^FBa,b,c,d,e`

This table identifies the parameters for this format:

| Parameters | Details |
| --- | --- |
| a = width of text block line (in dots) | *Accepted Values:* 0 to the width of the label<br>*Default Value:* 0<br>If the value is less than font width or not specified, text does not print. |
| b = maximum number of lines in text block | *Accepted Values:* 1 to 9999<br>*Default Value:* 1<br>Text exceeding the maximum number of lines overwrites the last line. Changing the font size automatically increases or decreases the size of the block. |
| c = add or delete space between lines (in dots) | *Accepted Values:* -9999 to 9999<br>*Default Value:* 0<br>Numbers are considered to be positive unless preceded by a minus sign. Positive values add space; negative values delete space. |
| d = text justification | *Accepted Values:*<br>L = left<br>C = center<br>R = right<br>J = justified<br>*Default Value:* L<br>If J is used the last line is left-justified. |
| e = hanging indent (in dots) of the second and remaining lines | *Accepted Values:* 0 to 9999<br>*Default Value:* 0 |

**Example •** These are examples of how the ^FB command affects field data.

| ZPL II CODE | GENERATED LABEL |
|---|---|
| ^XA<br>^CF0,30,30^FO25,50<br>^FB250,4,,<br>^FDFD command that IS<br>preceded by an FB command.^FS<br>^XZ | FD command that IS<br>preceded by an FB<br>command. |
| ^XA<br>^CF0,30,30^FO25,50<br>^FDFD command that IS NOT<br>preceded by an FB command.^FS<br>^XZ | FD command that IS NOT preceded by an FB cor |

## Comments on the ^FB Command

This scheme can be used to facilitate special functions:

`\&` = carriage return/line feed
`\(*)` = soft hyphen (word break with a dash)
`\\` = backslash (\)

**Item 1:** ^CI13 must be selected to print a backslash (\).

**Item 2:** If a soft hyphen escape sequence is placed near the end of a line, the hyphen is printed. If it is not placed near the end of the line, it is ignored.

(*) = any alphanumeric character

- If a word is too long to print on one line by itself (and no soft hyphen is specified), a hyphen is automatically placed in the word at the right edge of the block. The remainder of the word is on the next line. The position of the hyphen depends on word length, not a syllable boundary. Use a soft hyphen within a word to control where the hyphenation occurs.
- Maximum data-string length is 3K, including control characters, carriage returns, and line feeds.
- Normal carriage returns, line feeds, and *word spaces* at line breaks are discarded.
- When using ^FT (Field Typeset), ^FT uses the baseline origin of the last possible line of text. Increasing the font size causes the text block to increase in size from bottom to top. This could cause a label to print past its top margin.
- When using ^FO (Field Origin), increasing the font size causes the text block to increase in size from top to bottom.
- ^FS terminates an ^FB command. Each block requires its own ^FB command.

# ^FC

## Field Clock (for Real-Time Clock)

**Description** The ^FC command is used to set the clock-indicators (delimiters) and the clock mode for use with the Real-Time Clock hardware. This command must be included within each label field command string each time the Real-Time Clock values are required within the field.

**Format** ^FCa,b,c

This table identifies the parameters for this format:

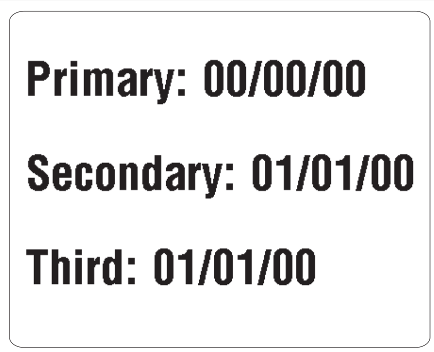
| Parameters | Details |
|---|---|
| a = primary clock indicator character | *Accepted Values:* any ASCII character<br>*Default Value:* % |
| b = secondary clock indicator character | *Accepted Values:* any ASCII character<br>*Default Value:* none—this value cannot be the same as a or c |
| c = third clock indicator character | *Accepted Values:* any ASCII character<br>*Default Value:* none—this value cannot be the same as a or b |

**Example** • Entering these ZPL commands sets the primary clock indicator to %, the secondary clock indicator to {, and the third clock indicator to #. The results are printed on a label with Primary, Secondary, and Third as field data.

**ZPL II CODE**

```
^XA
^FO10,100^A0N,50,50
^FC%,{,#
^FDPrimary: %m/%d/%y^FS
^FO10,200^A0N,50,50
^FC%,{,#
^FDSecondary: {m/{d/{y^FS
^FO10,300^A0N,50,50
^FC%,{,#
^FDThird: #m/#d/#y^FS
^XZ
```

**GENERATED LABEL**




**Comments** The ^FC command is ignored if the Real-Time Clock hardware is not present.

For more details on the Real Time Clock, see the *Zebra Real Time Clock Guide*.

# ^FD

## Field Data

**Description** The ^FD command defines the data string for the field. The field data can be any printable character except those used as command prefixes (^ and ~).

**Format** ^FDa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = data to be printed | *Accepted Values:* any data string up to 3072 bytes<br>*Default Value:* none—a string of characters must be entered |

**Comments** The ^ and ~ characters can be printed by changing the prefix characters—see *^CD ~CD* on page 108 and *^CT ~CT* on page 118. The new prefix characters cannot be printed.

Characters with codes above 127, or the ^ and ~ characters, can be printed using the ^FH and ^FD commands.

- ^CI13 must be selected to print a backslash (\).

# ^FH

## Field Hexadecimal Indicator

**Description** The ^FH command allows you to enter the hexadecimal value for any character directly into the ^FD statement. The ^FH command must precede each ^FD command that uses hexadecimals in its field.

Within the ^FD statement, the hexadecimal indicator must precede each hexadecimal value. The default hexadecimal indicator is _ (underscore). There must be a minimum of two characters designated to follow the underscore. The a parameter can be added when a different hexadecimal indicator is needed.

This command can be used with any of the commands that have field data (that is ^FD, ^FV (Field Variable), and ^SN (Serialized Data)).

Valid hexadecimal characters are:

0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

**Format** ^FHa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = hexadecimal indicator | *Accepted Values:* any character except current format and control prefix (^ and ~ by default)<br>*Default Value:* _ (underscore) |

**Example •** This is an example of how to enter a hexadecimal value directly into a ^FD statement: This is an example for ascii data using ^CI0.

| ZPL II CODE | GENERATED LABEL |
|---|---|
| ^XA<br>^FO100,100<br>^AD^FH<br>^FDTilde _7e used for HEX^FS<br>^XZ | Tilde ~ used for HEX |
| ^XA<br>^FO100,100<br>^AD^FH\<br>^FDTilde \7E used for HEX^FS<br>^XZ | Tilde ~ used for HEX |

The font linking remains until the link is broken or the printer is turned off. To permanently save the font linking, use the ^JUS command.

# ^FM

## Multiple Field Origin Locations

**Description** The `^FM` command allows you to control the placement of bar code symbols.

It designates field locations for the PDF417 (`^B7`) and Micro-PDF417 (`^BF`) bar codes when the structured append capabilities are used. This allows printing multiple bar codes from the same set of text information.

The structured append capability is a way of extending the text printing capacity of both bar codes. If a string extends beyond what the data limitations of the bar code are, it can be printed as a series: 1 of 3, 2 of 3, 3 of 3. Scanners read the information and reconcile it into the original, unsegmented text.

The `^FM` command triggers multiple bar code printing on the same label with `^B7` and `^BF` only. When used with any other commands, it is ignored.

**Format** `^FMx1,y1,x2,y2,...`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `x1` = x-axis location of first symbol (in dots) | *Accepted Values:*<br>`0` to `32000`<br>`e` = exclude this bar code from printing<br>*Default Value:* a value must be specified |
| `y1` = y-axis location of first symbol (in dots) | *Accepted Values:*<br>`0` to `32000`<br>`e` = exclude this bar code from printing<br>*Default Value:* a value must be specified |
| `x2` = x-axis location of second symbol (in dots) | *Accepted Values:*<br>`0` to `32000`<br>`e` = exclude this bar code from printing<br>*Default Value:* a value must be specified |
| `y2` = y-axis location of second symbol (in dots) | *Accepted Values:*<br>`0` to `32000`<br>`e` = exclude this bar code from printing<br>*Default Value:* a value must be specified |
| ... = continuation of X,Y pairs | *Maximum number of pairs:* `60` |

**Example •** This example shows you how to generate three bar codes with the text "Zebra Technologies Corporation strives to be…" would need to be repeated seven times, which includes 2870 characters of data (including spaces) between ^FD and ^FS:

**ZPL II CODE** | **GENERATED LABEL**

```
^XA
^FM100,100,100,600,100,1200
^BY2,3
^B7N,5,5,9,83,N
^FDZebra Technologies
Corporation strives to be
the expert supplier of
innovative solutions to
specialty demand labeling
and ticketing problems of
business and government.
We will attract and retain
the best people who will
understand our customer's
needs and provide them with
systems, hardware, software,
consumables and service
offering the best value,
high quality, and reliable
performance, all delivered
in a timely manner
…
^FS^XZ
```

1

| | |
|---|---|
| 1 | The ellipse is not part of the code. It indicates that the text needs to be repeated seven times, as mentioned in the example description. |

**Example** • This example assumes a maximum of three bar codes, with bar code 2 of 3 omitted:

**ZPL II CODE**

```
^XA
^FM100,100,e,e,100,1200
^BY2,3
^B7N,5,5,9,83,N
^FDZebra Technologies
Corporation strives to be
the expert supplier of
innovative solutions to
specialty demand labeling
and ticketing problems of
business and government.
We will attract and retain
the best people who will
understand our customer's
needs and provide them with
systems, hardware, software,
consumables and service
offering the best value,
high quality, and reliable
performance, all delivered
in a timely manner
…
^FS^XZ
```

1

**GENERATED LABEL**

| | |
|---|---|
| 1 | The ellipse is not part of the code. It indicates that the text needs to be repeated seven times, as mentioned in the example description. |

**Comments** Subsequent bar codes print once the data limitations of the previous bar code have been exceeded. For example, bar code 2 of 3 prints once 1 of 3 has reached the maximum amount of data it can hold. Specifying three fields does not ensure that three bar codes print; enough field data to fill three bar code fields has to be provided.

The number of the `x,y` pairs can exceed the number of bar codes generated. However, if too few are designated, no symbols print.

# ^FN

## Field Number

**Description** The ^FN command numbers the data fields. This command is used in both ^DF (Store Format) and ^XF (Recall Format) commands.

In a stored format, use the ^FN command where you would normally use the ^FD (Field Data) command. In recalling the stored format, use ^FN in conjunction with the ^FD command.

The number of fields and data that can be stored is dependent in the available printer memory.

**Note •** The maximum number of ^FN commands that can be used depends on the amount of data that is placed in the fields on the label. It is recommended to use 400 or fewer fields.

**Format** ^FN#

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| # = number to be assigned to the field | *Accepted Values:* 0 to 9999<br>*Default Value:* 0 |

For a complete example of the ^DF and ^XF command, see *^DF and ^XF — Download format and recall format* on page 11.

### Comments

- The same ^FN value can be stored with several different fields.
- If a label format contains a field with ^FN and ^FD, the data in that field prints for any other field containing the same ^FN value.

# ^FO

## Field Origin

**Description** The ^FO command sets a field origin, relative to the label home (^LH) position. ^FO sets the upper-left corner of the field area by defining points along the x-axis and y-axis independent of the rotation.

**Format** ^FOx,y

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| x = x-axis location (in dots) | *Accepted Values:* 0 to 32000<br>*Default Value:* 0 |
| y = y-axis location (in dots) | *Accepted Values:* 0 to 32000<br>*Default Value:* 0 |

**Comments** If the value entered for the x or y parameter is too high, it could position the field origin completely off the label.

# ^FP

## Field Parameter

**Description** The ^FP command allows vertical and reverse formatting of the font field, commonly used for printing Asian fonts.

**Format** ^FPd,g

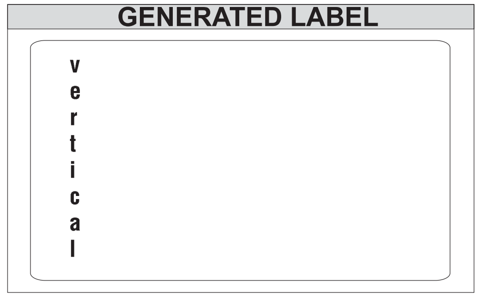
This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = direction | *Accepted Values:*<br>H = horizontal printing (left to right)<br>V = vertical printing (top to bottom)<br>R = reverse printing (right to left)<br>*Default Value:* H |
| g = additional inter-character gap (in dots) | *Accepted Values:* 0 to 9999<br>*Default Value:* 0 if no value is entered |

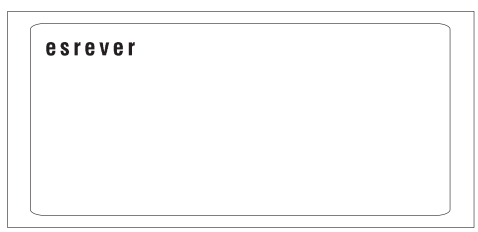
**Example •** This is an example of how to implement reverse and vertical print:

| ZPL II CODE | GENERATED LABEL |
|---|---|

```
^XA
^FO100,50
^FPV,10
^AV
^FDvertical^FS
^XZ
```




```
^XA
^FO350,50
^FPR,10
^AV
^FDreverse^FS
^XZ
```



# ^FR

## Field Reverse Print

**Description** The ^FR command allows a field to appear as white over black or black over white. When printing a field and the ^FR command has been used, the color of the output is the reverse of its background.

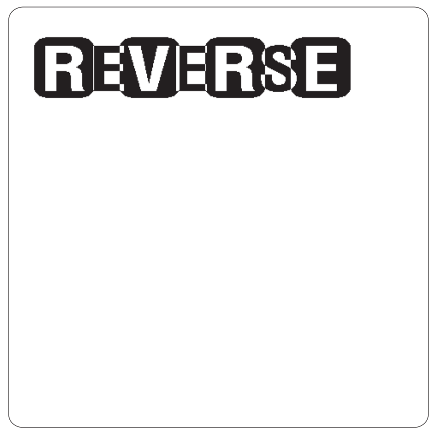
**Format** ^FR

**Example** • In this example, the ^GB command creates areas of black allowing the printing to appear white:

**ZPL II CODE**

```
^XA
^PR1
^FO100,100
^GB70,70,70,,3^FS
^FO200,100
^GB70,70,70,,3^FS
^FO300,100
^GB70,70,70,,3^FS
^FO400,100
^GB70,70,70,,3^FS
^FO107,110^CF0,70,93
^FR^FDREVERSE^FS
^XZ
```

**GENERATED LABEL**




**Comments** The ^FR command applies to only one field and has to be specified each time. When multiple ^FR commands are going to be used, it might be more convenient to use the ^LR command.

# ^FS

## Field Separator

**Description** The ^FS command denotes the end of the field definition. Alternatively, ^FS command can also be issued as a single ASCII control code SI (Control-O, hexadecimal 0F).

**Format** ^FS

# ^FT

## Field Typeset

**Description** The `^FT` command sets the field position, relative to the home position of the label designated by the `^LH` command. The typesetting origin of the field is fixed with respect to the contents of the field and does not change with rotation.

**Note •** The `^FT` command is capable of concantination of fields.

**Format** `^FTx,y`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `x` = x-axis location (in dots) | *Accepted Values:* `0` to `32000`<br>*Default Value:* position after last formatted text field |
| `y` = y-axis location (in dots) | *Accepted Values:* `0` to `32000`<br>*Default Value:* position after last formatted text field |

**Table 12 • Typeset Justification**

| | | |
|---|---|---|
| **Left Justified** | Text | For examples, see *Field Interactions* on page 373. |
| | Bar Codes | Origin is base of bar code, at left edge |
| | Graphic Boxes | Origin is bottom-left corner of the box |
| | Images | Origin is bottom-left corner of the image area |

**Example •** This is an example of the ^FT command and concatenation:

**ZPL II CODE**

```
^XA
^FT10,200^A0N,30,20^FDACME ^FS
^FT^GS^FDC^FS
^FT^A0N,30,20^FDSummer ^FS
^FT^A0N,60,50^FDClearance ^FS
^FT^A0N,120,100^FDSale ^FS
^XZ
```

**GENERATED LABEL**


When a coordinate is missing, the position following the last formatted field is assumed. This *remembering* simplifies field positioning with respect to other fields. Once the first field is positioned, other fields follow automatically.

There are several instances where using the ^FT command without specifying x and y parameters is not recommended:

- when positioning the first field in a label format
- at any time with the ^FN (Field Number) command
- following an ^SN (Serialization Data) command
- variable data
- bidirectional text

# ^FV

## Field Variable

**Description** ^FV replaces the ^FD (field data) command in a label format when the field is variable.

**Format** ^FVa

This table identifies the parameters for this format:

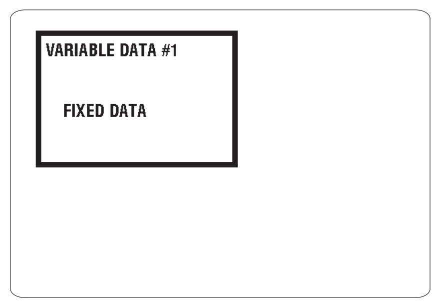
| Parameters | Details |
|---|---|
| a = variable field data to be printed | *Accepted Values:* 0 to 3072 byte string<br>*Default Value:* if no data is entered, the command is ignored |

**Example •** This is an example of how to use the ^MC and ^FV command:

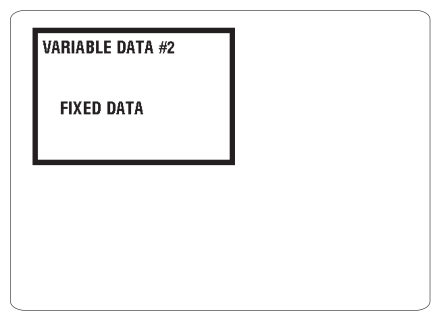
**ZPL II CODE**

```
^XA
^FO40,40
^GB300,203,8^FS
^FO55,60^CF0,25
^FVVARIABLE DATA #1^FS
^FO80,150
^FDFIXED DATA^FS
^MCN
^XZ
```

**GENERATED LABEL**




```
^XA
^FO55,60^CF0,25
^FVVARIABLE DATA #2^FS
^MCY
^XZ
```




**Comments** ^FV fields are always cleared after the label is printed. ^FD fields are not cleared.

# ^FW

## Field Orientation

**Description** The ^FW command sets the default orientation for all command fields that have an orientation (rotation) parameter. Fields can be rotated 0, 90, 180, or 270 degrees clockwise by using this command.

The ^FW command affects only fields that follow it. Once you have issued a ^FW command, the setting is retained until you turn off the printer or send a new ^FW command to the printer.

**Format** ^FWr

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| r = rotate field | *Accepted Values:*<br>N = normal<br>R = rotated 90 degrees<br>I = inverted 180 degrees<br>B = bottom-up 270 degrees, read from bottom up<br>*Initial Value at Power-up:* N |

**Example** • This example shows how ^FW rotation works in conjunction with ^FO. In this example note that:

- the fields using A0N print the field in normal rotation
- the fields with no rotation indicated (A0) follow the rotation used in the ^FW command (^FWR).

| ZPL II CODE | GENERATED LABEL |
|---|---|

```
^XA
^FWR
^FO150,100^A0N,70,60^FDZebra Technologies^FS
^FO50,75^A0,70,60^FD0123456789^FS
^FO150,200^A0N,70,60^FD333 Corporate Woods
 Parkway^FS
^FO900,75^A0,70,60^FDXXXXXXXXX^FS
^XZ
```


**Comments** ^FW affects only the orientation in commands where the rotation parameter has not been specifically set. If a command has a specific rotation parameter, that value is used.

# ^FX

## Comment

**Description** The ^FX command is useful when you want to add *non-printing* informational comments or statements within a label format. Any data after the ^FX command up to the next caret (^) or tilde (~) command does not have any effect on the label format. Therefore, you should avoid using the caret (^) or tilde (~) commands within the ^FX statement.

**Format** ^FXc

This table identifies the parameters for this format:

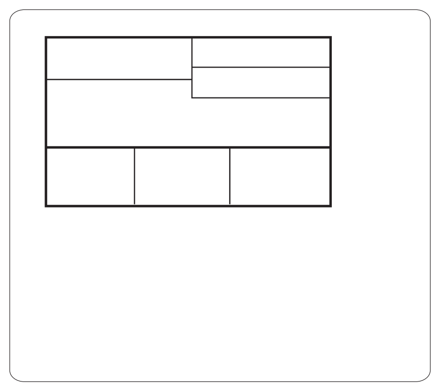
| Parameters | Details |
|---|---|
| c = non printing comment | Creates a non-printable comment. |

**Example** • This is an example of how to use the ^FX command effectively:

**ZPL II CODE**

```
^XA
^LH100,100^FS
^FXSHIPPING LABEL^FS
^FO10,10^GB470,280,4^FS
^FO10,190^GB470,4,4^FS
^FO10,80^GB240,2,2^FS
^FO250,10^GB2,100,2^FS
^FO250,110^GB226,2,2^FS
^FO250,60^GB226,2,2^FS
^FO156,190^GB2,95,2^FS
^FO312,190^GB2,95,2^FS
^XZ
```

**GENERATED LABEL**



**Comments** Correct usage of the ^FX command includes following it with the ^FS command.

# ^GB

## Graphic Box

**Description** The `^GB` command is used to draw boxes and lines as part of a label format. Boxes and lines are used to highlight important information, divide labels into distinct areas, or to improve the appearance of a label. The same format command is used for drawing either boxes or lines.

**Format** `^GBw,h,t,c,r`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `w` = box width (in dots) | *Accepted Values:* value of `t` to `32000`<br>*Default Value:* value used for thickness (t) or 1 |
| `h` = box height (in dots) | *Accepted Values:* value of `t` to `32000`<br>*Default Value:* value used for thickness (t) or 1 |
| `t` = border thickness (in dots) | *Accepted Values:* `1` to `32000`<br>*Default Value:* `1` |
| `c` = line color | *Accepted Values:*<br>`B` = black<br>`W` = white<br>*Default Value:* `B` |
| `r` = degree of corner-rounding | *Accepted Values:* `0` (no rounding) to `8` (heaviest rounding)<br>*Default Value:* `0` |

For the `w` and `h` parameters, keep in mind that printers have a default of 6, 8, 12, or 24 dots/millimeter. This comes out to 153, 203, 300, or 600 dots per inch. To determine the values for w and h, calculate the dimensions in millimeters and multiply by 6, 8, 12, or 24.

If the width and height are not specified, you get a solid box with its width and height as specified by value `t`.

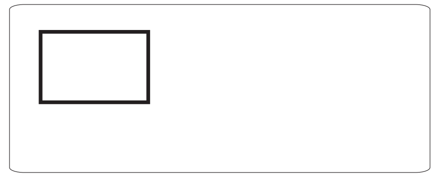
The roundness-index is used to determine a rounding-radius for each box. Formula:

**rounding-radius = (rounding-index / 8) * (shorter side / 2)**

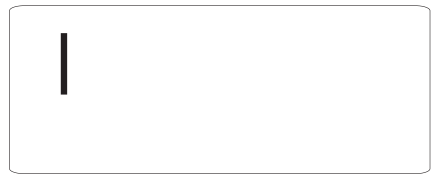
where the shorter side is the lesser of the width and height (after adjusting for minimum and default values).

**Examples** • Here are a few examples of graphic boxes:

**Width: 1.5 inch; Height: 1 inch; Thickness: 10; Color: default; Rounding: default**

| ZPL II CODE | GENERATED LABEL |
|---|---|
| `^XA`<br>`^FO50,50`<br>`^GB300,200,10^FS`<br>`^XZ` |  |

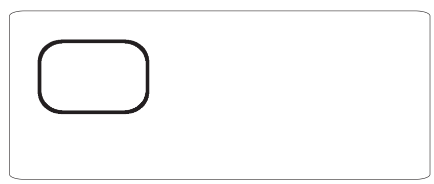
**Width: 0 inch; Height: 1 inch; Thickness: 20; Color: default; Rounding: default:**

| ZPL II CODE | GENERATED LABEL |
|---|---|
| `^XA`<br>`^FO50,50`<br>`^GB0,203,20^FS`<br>`^XZ` |  |

**Width: 1 inch; Height: 0 inch; Thickness: 30; Color: default; Rounding: default**

| ZPL II CODE | GENERATED LABEL |
|---|---|
| `^XA`<br>`^FO50,50`<br>`^GB203,0,20^FS`<br>`^XZ` |  |

**Width: 1.5 inch; Height: 1 inch; Thickness: 10; Color: default; Rounding: 5**

| ZPL II CODE | GENERATED LABEL |
|---|---|
| `^XA`<br>`^FO50,50`<br>`^GB300,200,10,,5^FS`<br>`^XZ` |  |

# ^GC

## Graphic Circle

**Description** The `^GC` command produces a circle on the printed label. The command parameters specify the diameter (width) of the circle, outline thickness, and color. Thickness extends inward from the outline.

**Format** `^GCd,t,c`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = circle diameter (in dots) | *Accepted Values:* `3` to `4095` (larger values are replaced with 4095)<br>*Default Value:* `3` |
| `t` = border thickness (in dots) | *Accepted Values:* `2` to `4095`<br>*Default Value:* `1` |
| `c` = line color | *Accepted Values:*<br>`B` = black<br>`W` = white<br>*Default Value:* `B` |

**Example** • This is an example of how to create a circle on the printed label:

**ZPL II CODE**

```
^XA
^FO50,50
^GC250,10,B^FS
^XZ
```

**GENERATED LABEL**

# ^GD

## Graphic Diagonal Line

**Description** The ^GD command produces a straight diagonal line on a label. This can be used in conjunction with other graphic commands to create a more complex figure.

**Format** `^GDw,h,t,c,o`

This table identifies the parameters for this format:

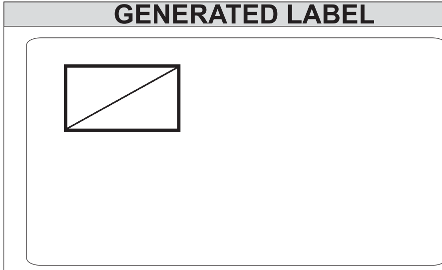
| Parameters | Details |
| --- | --- |
| `w` = box width (in dots) | *Accepted Values:* `3` to `32000`<br>*Default Value:* value of `t` (thickness) or `1` |
| `h` = box height (in dots) | *Accepted Values:* `3` to `32000`<br>*Default Value:* value of `t` (thickness) or `1` |
| `t` = border thickness (in dots) | *Accepted Values:* 1 to 32000<br>*Default Value:* 1 |
| `c` = line color | *Accepted Values:*<br>`B` = black<br>`W` = white<br>*Default Value:* `B` |
| `o` = orientation (direction of the diagonal) | *Accepted Values:*<br>R (or /) = right-leaning diagonal<br>L (or \) = left-leaning diagonal<br>*Default Value:* R |

**Example** • This is an example of how to create a diagonal line connecting one corner with the opposite corner of a box on a printed label:

**ZPL II CODE**

```
^XA
^FO150,100
^GB350,203,10^FS
^FO155,110
^GD330,183,10,,R^FS
^XZ
```

**GENERATED LABEL**



# ^GE

## Graphic Ellipse

**Description** The ^GE command produces an ellipse in the label format.

**Format** `^GEw,h,t,c`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `w` = ellipse width (in dots) | *Accepted Values:* 3 to 4095 (larger values are replaced with 4095)<br>*Default Value:* value used for thickness (`t`) or `1` |
| `h` = ellipse height (in dots) | *Accepted Values:* 3 to 4095<br>*Default Value:* value used for thickness (`t`) or `1` |
| `t` = border thickness (in dots) | *Accepted Values:* 2 to 4095<br>*Default Value:* 1 |
| `c` = line color | *Accepted Values:*<br>`B` = black<br>`W` = white<br>*Default Value:* `B` |

**Example** • This is an example of how to create a ellipse on a printed label:

**ZPL II CODE**

```
^XA
^FO100,100
^GE300,100,10,B^FS
^XZ
```

**GENERATED LABEL**

# ^GF

## Graphic Field

**Description** The ^GF command allows you to download graphic field data directly into the printer's bitmap storage area. This command follows the conventions for any other field, meaning a field orientation is included. The graphic field data can be placed at any location within the bitmap space.

**Format** `^GFa,b,c,d,data`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = compression type | *Accepted Values:*<br>A = ASCII hexadecimal (follows the format for other download commands)<br>B = binary (data sent after the `c` parameter is strictly binary)<br>C = compressed binary (data sent after the `c` parameter is in compressed binary format. The data is compressed on the host side using Zebra's compression algorithm. The data is then decompressed and placed directly into the bitmap.)<br>*Default Value:* A |
| `b` = binary byte count | *Accepted Values:* 1 to 99999<br>This is the total number of bytes to be transmitted for the total image or the total number of bytes that follow parameter `d`. For ASCII download, the parameter should match parameter `c`. Out-of-range values are set to the nearest limit.<br>*Default Value:* command is ignored if a value is not specified |
| `c` = graphic field count | *Accepted Values:* 1 to 99999<br>This is the total number of bytes comprising the graphic format (width x height), which is sent as parameter `d`. Count divided by bytes per row gives the number of lines in the image. This number represents the size of the image, not necessarily the size of the data stream (see `d`).<br>*Default Value:* command is ignored if a value is not specified |

| Parameters | Details |
|---|---|
| `d` = bytes per row | *Accepted Values:* 1 to 99999<br>This is the number of bytes in the downloaded data that comprise one row of the image.<br>*Default Value:* command is ignored if a value is not specified |
| `data` = data | *Accepted Values:*<br>ASCII hexadecimal data: 00 to FF<br>A string of ASCII hexadecimal numbers, two digits per image byte. CR and LF can be inserted as needed for readability. The number of two-digit number pairs must match the above count. Any numbers sent after count is satisfied are ignored. A comma in the data pads the current line with *00* (white space), minimizing the data sent. `~DN` or any caret or tilde character prematurely aborts the download.<br>**Binary data:** Strictly binary data is sent from the host. All control prefixes are ignored until the total number of bytes needed for the graphic format is sent. |

**Example •** This example downloads 8,000 total bytes of data and places the graphic data at location 100,100 of the bitmap. The data sent to the printer is in ASCII form.

```
^FO100,100^GFA,8000,8000,80,ASCII data
```

**Example •** This example downloads 8,000 total bytes of data and places the graphic data at location 100,100 of the bitmap. The data sent to the printer is in binary form.

```
^FO100,100^GFB,8000,8000,80,Binary data
```

# ^GS

## Graphic Symbol

**Description** The ^GS command enables you to generate the registered trademark, copyright symbol, and other symbols.

**Format** `^GSo,h,w`

This table identifies the parameters for this format:

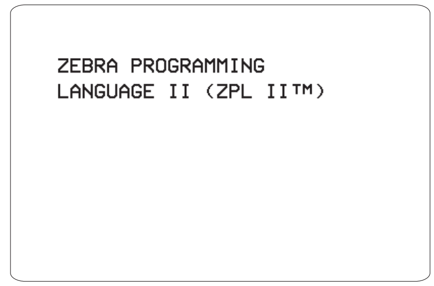
| Parameters | Details |
|---|---|
| o = field orientation | *Accepted Values:*<br>N = normal<br>R = rotate 90 degrees clockwise<br>I = inverted 180 degrees<br>B = bottom-up, 270 degrees<br>*Default Value:* N or last ^FW value |
| h = character height proportional to width (in dots) | *Accepted Values:* 0 to 32000<br>*Default Value:* last ^CF value |
| w = character width proportional to height (in dots) | *Accepted Values:* 0 to 32000<br>*Default Value:* last ^CF value |

**Example** • Use the ^GS command followed by ^FD and the appropriate character (A through E) within the field data to generate the desired character:

**ZPL II CODE**

```
^XA^CFD
^FO50,50
^FDZEBRA PROGRAMMING^FS
^FO50,75
^FDLANGUAGE II (ZPL II )^FS
^FO280,75
^GS^FDC^FS
^XZ
```

**GENERATED LABEL**




# ~HB

## Battery Status

**Description** When the ~HB command is sent to the printer, a data string is sent back to the host. The string starts with an <STX> control code sequence and terminates by an <ETX><CR><LF> control code sequence.

**Important •** This command only responds to mobile printers.

**Format** ~HB

**Parameters:** when the printer receives the command, it returns:

```
<STX>bb.bb,hh.hh,bt<ETX><CR><LF>
```

| | | |
|---|---|---|
| <STX> | = | ASCII start-of-text character |
| bb.bb | = | current battery voltage reading to the nearest 1/4 volt |
| hh.hh | = | current head voltage reading to the nearest 1/4 volt |
| bt | = | battery temperature in Celsius |
| <ETX> | = | ASCII end-of-text character |
| <CR> | = | ASCII carriage return |
| <LF> | = | ASCII line feed character |

**Comments** This command is used for the power-supply battery of the printer and should not be confused with the battery backed-up RAM.

# ~HD

## Head Diagnostic

**Description** The ~HD command echoes printer status information that includes the power supply and head temperature using the terminal emulator.

**Format** ~HD

**Example** • This is an example of the ~HD command:

```
Head Temp = 29
Ambient Temp = 00
Head Test = Passed
Darkness Adjust = 23
Print Speed = 2
Slew Speed = 6
Backfeed Speed = 2
Static_pitch_length = 0521
Dynamic_pitch_length = 0540
Max_dynamic_pitch_length = 0540
Min_dynamic_pitch_length = 0537
COMMAND PFX = ~ : FORMAT PFX = ^ : DELIMITER = ,
P30 INTERFACE = None
P31 INTERFACE = None
P32 INTERFACE = Front Panel             Revision 5
P33 INTERFACE = None
P34 INTERFACE = None
P35 INTERFACE = None
Dynamic_top_position = 0008

No ribbon A/D = 0000
```

# ^HF

## Host Format

**Description** The ^HF command sends stored formats to the host.

**Format** ^HFd,o,x

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = device to recall image | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = image name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Fixed Value:* .ZPL |

**Example •** This example shows the sequence and results.

Using a terminal emulator, you download this code to the printer:

```
^XA
^DFB:FILE1.ZPL
^FO100,100^A0,100
^FDTEST^FS
^XZ
```

Then you send this code to the printer:

```
^XA
^HFB:FILE1.ZPL
^XZ
```

The terminal emulator returns this code:

```
^XA^DFFILE1,
^FO100,100^A0,100^FDTEST^FS
^XZ
```

# ^HG

## Host Graphic

**Description** The ^HG command is used to upload graphics to the host. The graphic image can be stored for future use, or it can be downloaded to any Zebra printer.

**Format** ^HGd:o.x

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = device location of object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* search priority |
| o = object name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Fixed Value:* .GRF |

**Comments** For more information on uploading graphics, see *^HY* on page 178.

# ^HH

## Configuration Label Return

**Description** The ^HH command echoes printer configuration back to the host, using a terminal emulator.

**Format** ^HH

**Example •** This is an example of the ^HH command:

```
+10                   DARKNESS
+000                  TEAR OFF
TEAR OFF              PRINT MODE
NON-CONTINUOUS        MEDIA TYPE
WEB                   SENSOR TYPE
DIRECT-THERMAL        PRINT METHOD
050 6/8 MM            PRINT  WIDTH
0622                  LABEL LENGTH
22.0IN   557MM        MAXIMUM LENGTH
9600                  BAUD
8 BITS                DATA BITS
NONE                  PARITY
XON/XOFF              HOST HANDSHAKE
NONE                  PROTOCOL
000                   NETWORK ID
NORMAL MODE           COMMUNICATIONS
<~>  7EH              CONTROL PREFIX
<^>  5EH              FORMAT PREFIX
<,>  2CH              DELIMITER CHAR
ZPL II                ZPL MODE
NO MOTION             MEDIA POWER UP
NO MOTION             HEAD CLOSE
DEFAULT               BACKFEED
+000                  LABEL TOP
+0000                 LEFT POSITION
026                   WEB S.
068                   MEDIA S.
050                   MARK S.
001                   MARK MED S.
CS                    MODES ENABLED
..                    MODES DISABLED
864 8/MM FULL         RESOLUTION
V32.10.2    <-        FIRMWARE
V2.2.6.98.A           HARDWARE ID
CUSTOMIZED            CONFIGURATION
1024.............R:   RAM
8192.............B:   MEMORY CARD
0768.............E:   ONBOARD FLASH
NONE                  FORMAT CONVERT
NONE                  OPTION
05/14/03              RTC DATE
02:23                 RTC TIME
DYNAMIC               IP RESOLUTION
ALL                   IP PROTOCOL
010.003.005.090       IP ADDRESS
255.255.255.000       SUBNET MASK
010.003.005.001       DEFAULT GATEWAY
```

# ~HI

## Host Identification

**Description** The `~HI` command is designed to be sent from the host to the Zebra printer to retrieve information. Upon receipt, the printer responds with information on the model, software version, dots-per-millimeter setting, memory size, and any detected objects.

**Format** `~HI`

When the printer receives this command, it returns:

`XXXXXX,V1.0.0,dpm,000KB,X`

**XXXXXX = model of Zebra printer**

**V1.0.0** = version of software

**dpm = dots/mm**

6, 8, 12, or 24 dots/mm printheads

**000KB = memory**

`512KB` = 1/2 MB
`1024KB` = 1 MB
`2048KB` = 2 MB
`4096KB` = 4 MB
`8192KB` = 8 MB

**x = recognizable objects**

only options specific to printer are shown (cutter, options, et cetera.)

# ~HM

## Host RAM Status

**Description** Sending `~HM` to the printer immediately returns a memory status message to the host. Use this command whenever you need to know the printer's RAM status.

When `~HM` is sent to the Zebra printer, a line of data containing information on the total amount, maximum amount, and available amount of memory is sent back to the host.

**Format** `~HM`

**Example •** This example shows when the `~HM` is sent to the printer, a line of data containing three numbers are sent back to the host. Each set of numbers is identified and explained in the table that follows:

```
               2
1   3584,3586,3586   3
```

| | |
|---|---|
| **1** | The total amount of RAM (in kilobytes) installed in the printer. In this example, the printer has 1024K RAM installed. |
| **2** | The maximum amount of RAM (in kilobytes) available to the user. In this example, the printer has a maximum of 780K RAM available. |
| **3** | The amount of RAM (in kilobytes) currently available to the user. In this example, there is 780K of RAM in the printer currently available to the user. |

**Comments** Memory taken up by bitmaps is included in the currently available memory value (due to `^MCN`).

Downloading a graphic image, fonts, or saving a bitmap affects only the amount of RAM. The total amount of RAM and maximum amount of RAM does not change after the printer is turned on.

# ~HS

## Host Status Return

**Description** When the host sends `~HS` to the printer, the printer sends three data strings back. Each string starts with an <STX> control code and is terminated by an <ETX><CR><LF> control code sequence. To avoid confusion, the host prints each string on a separate line.

**Note •** When the parallel port is in an error condition, the host might not send data to the printer until the error condition is corrected.

**String 1**

```
<STX>aaa,b,c,dddd,eee,f,g,h,iii,j,k,l<ETX><CR><LF>
```

| | | |
|---|---|---|
| **aaa** | = | communication (interface) settings* |
| **b** | = | paper out flag (1 = paper out) |
| **c** | = | pause flag (1 = pause active) |
| **dddd** | = | label length (value in number of dots) |
| **eee** | = | number of formats in receive buffer |
| **f** | = | *buffer full* flag (1 = receive buffer full) |
| **g** | = | *communications diagnostic mode* flag (1 = diagnostic mode active) |
| **h** | = | *partial format* flag (1 = partial format in progress) |
| **iii** | = | unused (always 000) |
| **j** | = | *corrupt RAM* flag (1 = configuration data lost) |
| **k** | = | temperature range (1 = under temperature) |
| **l** | = | temperature range (1 = over temperature) |

* This string specifies the printer's baud rate, number of data bits, number of stop bits, parity setting, and type of handshaking. This value is a three-digit decimal representation of an eight-bit binary number. To evaluate this parameter, first convert the decimal number to a binary number.

The nine-digit binary number is read according to this table:

| aaa = $a^8$ $a^7$ $a^6$ $a^5$ $a^4$ $a^3$ $a^2$ $a^1$ $a^0$ | |
|---|---|
| $a^7$ = Handshake<br>0 = Xon/Xoff<br>1 = DTR | $a^8$ $a^2 a^1 a^0$ = Baud<br><br>0 000 = 110<br>0 001 = 300<br>0 010 = 600<br>0 011 = 1200<br>0 100 = 2400<br>0 101 = 4800<br>0 110 = 9600<br>0 111 = 19200<br>1 000 = 28800 *(available only on certain printer models)*<br>1 001 = 38400 *(available only on certain printer models)*<br>1 010 = 57600 *(available only on certain printer models)*<br>1 011 = 14400 |
| $a^6$ = Parity Odd/Even<br>0 = Odd<br>1 = Even | |
| $a^5$ = Disable/Enable<br>0 = Disable<br>1 = Enable | |
| $a^4$ = Stop Bits<br>0 = 2 Bits<br>1 = 1 Bit | |
| $a^3$ = Data Bits<br>0 = 7 Bits<br>1 = 8 Bits | |

**String 2**

```
<STX>mmm,n,o,p,q,r,s,t,uuuuuuuu,v,www<ETX><CR><LF>
```

| | | |
|---|---|---|
| `mmm` | = | function settings* |
| `n` | = | unused |
| `o` | = | *head up* flag (1 = head in up position) |
| `p` | = | *ribbon out* flag (1 = ribbon out) |
| `q` | = | *thermal transfer mode* flag (1 = Thermal Transfer Mode selected) |
| `r` | = | Print Mode<br>`0` = Rewind<br>`1` = Peel-Off<br>`2` = Tear-Off<br>`3` = Cutter<br>`4` = Applicator |
| `s` | = | print width mode |
| `t` | = | *label waiting* flag (1 = label waiting in Peel-off Mode) |
| `uuuuuuuu` | = | labels remaining in batch |
| `v` | = | *format while printing* flag (always 1) |
| `www` | = | number of graphic images stored in memory |

* This string specifies the printer's media type, sensor profile status, and communication diagnostics status. As in String 1, this is a three-digit decimal representation of an eight-bit binary number. First, convert the decimal number to a binary number.

The eight-digit binary number is read according to this table:

| mmm = m7 m6 m5 m4 m3 m2 m1 m0 | |
|---|---|
| m7 = Media Type<br>0 = Die-Cut<br>1 = Continuous | m4 m3 m2 m1 = Unused<br>0 = Off<br>1 = On |
| m6 = Sensor Profile<br>0 = Off | m0 = Print Mode<br>0 = Direct Thermal<br>1 = Thermal Transfer |
| m5 = Communications Diagnostics<br>0 = Off<br>1 = On | |

**String 3**

```
<STX>xxxx,y<ETX><CR><LF>
```

| | | |
|---|---|---|
| xxxx | = | password |
| y | = | 0 (static RAM not installed) |
| | | 1 (static RAM installed) |

# ~HU

## Return ZebraNet Alert Configuration

**Description** This command returns the table of configured ZebraNet Alert settings to the host.

**Format** `~HU`

**Example** • If the `~HU` command is sent to the printer with existing Alert messages set to go to e-mail and SNMP traps, the data returned would look something like the information below. See ^SX on page 270 for complete information on the individual parameter settings.

```
B,C,Y,Y,ADMIN@COMPANY.COM,0
J,F,Y,Y,,0
C,F,Y,Y,,0
D,F,Y,Y,,0
E,F,Y,N,,0
F,F,Y,N,,0
H,C,Y,N,ADMIN@COMPANY.COM,0
N,C,Y,Y,ADMIN@COMPANY.COM,0
O,C,Y,Y,ADMIN@COMPANY.COM,0
P,C,Y,Y,ADMIN@COMPANY.COM,0
```

**Important** • If there are no `^SX` (alerts) set, the printer will not respond to the `~HU` command.

The first line indicates that condition B (ribbon out) is routed to destination C (e-mail address).

The next two characters, Y and Y, indicate that the *condition set* and *condition clear* options have been set to `yes`.

The following entry is the destination that the Alert e-mail should be sent to; in this example it is `admin@company.com`.

The last figure seen in the first line is 0, which is the port number.

Each line shows the settings for a different Alert condition as defined in the `^SX` command.

# ^HV

## Host Verification

**Description** This command is used to return data from specified fields, along with an optional ASCII header, to the host computer. The command can be used with any field that has been assigned a number with the ^RT command or the ^FN and ^RF commands.

**Format** ^HV#,n,h

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| # = field number specified with another command | The value assigned to this parameter should be the same as the one used in another command.<br>*Accepted Values:* 0 to 9999<br>*Default Value:* 0 |
| n = number of bytes to be returned | *Accepted Values:* 1 to 256<br>*Default Value:* 64 |
| h = header | Header to be returned with the data.<br>*Accepted Values:* 0 to 3072 bytes<br>*Default Value:* no header |

# ^HW

## Host Directory List

**Description** `^HW` is used to transmit a directory listing of objects in a specific memory area (storage device) back to the host device. This command returns a formatted ASCII string of object names to the host.

Each object is listed on a line and has a fixed length. The total length of a line is also fixed. Each line listing an object begins with the asterisk (*) followed by a blank space. There are eight spaces for the object name, followed by a period and three spaces for the extension. The extension is followed by two blank spaces, six spaces for the object size, two blank spaces, and three spaces for option flags (reserved for future use). The format looks like this:

```
<STX><CR><LF>
DIR R: <CR><LF>
*Name.ext(2sp.)(6 obj. sz.)(2sp.)(3 option flags)
*Name.ext(2sp.)(6 obj. sz.)(2sp.)(3 option flags)
<CR><LF>
-xxxxxxx bytes free
<CR><LF>
<ETX>
<STX> = start of text
<CR><LR> = carriage return/line feed
<ETX> = end on text
```

The command might be used in a stand-alone file to be issued to the printer at any time. The printer returns the directory listing as soon as possible, based on other tasks it might be performing when the command is received.

This command, like all ^ (caret) commands, is processed in the order that it is received by the printer.

**Format** `^HWd:o.x`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = location to retrieve object listing | *Accepted Values:* `R:`, `E:`, `B:`, `A:` and `Z:`<br>*Default Value:* `R:` |
| `o` = object name | *Accepted Values:* `1` to `8` alphanumeric characters<br>*Default Value:* asterisk (`*`). A question mark (`?`) can also be used. |
| `x` = extension | *Accepted Values:* any extension conforming to Zebra conventions<br>*Default Value:* asterisk (`*`). A question mark (`?`) can also be used. |

**Example** • Listed is an example of the `^HW` command to retrieve from information `R:`

```
^XA
^HWR:*.*
^XZ
```

**Example** • The printer returned this information as the Host Directory Listing:`-DIR R:*.*`

```
*R:ARIALN1.FNT 49140
*R:ARIALN2.FNT 49140
*R:ARIALN3.FNT 49140
*R:ARIALN4.FNT 49140
*R:ARIALN.FNT 49140
*R:ZEBRA.GRF 8420
-794292 bytes free R:RAM
```

# ^HY

## Upload Graphics

**Description** The ^HY command is an extension of the ^HG command. ^HY is used to upload graphic objects from the printer in any supported format.

**Format** ^HYd:o.x

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = location of object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* search priority |
| o = object name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* an object name must be specified |
| x = extension | *Accepted Values:*<br>G = .GRF (raw bitmap format)<br>P = .PNG (compressed bitmap format)<br>*Default Value:* format of stored image |

**Comments** The image is uploaded in the form of a ~DY command. The data field of the returned ~DY command is always encoded in the ZB64 format.

# ^HZ

## Display Description Information

**Description** The ^HZ command is used for returning printer description information in XML format. The printer returns information on format parameters, object directories, individual object data, and print status information.

**Format** ^HZb

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| b = display description to return | *Accepted Values:*<br>a = display all information<br>f = display printer format setting information<br>l = display object directory listing information<br>o = display individual object data information<br>r = display printer status information<br>*Default Value:* if the value is missing or invalid, the command is ignored |

**Format** ^HZO,d:o.x,l

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = location of stored object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = object name | *Accepted Values:* 1 to 8, or 1 to 16 alphanumeric characters based on parameter **l**.<br>*Default Value:* if a name is not specified, UNKNOWN is used. |

| Parameters | Details |
|---|---|
| `x` = extension | Supported extensions for objects (parameter `o`) include:<br>`.FNT` — font<br>`.GRF` — graphic<br>`.PNG` — compressed graphic<br>`.ZPL` — stored format<br>`.DAT` — encoding table<br>`.ZOB` — downloadable object<br>`.STO` — Alert data file |
| `l` = long filename support | *Accepted Values:*<br>`Y` = Yes<br>If `Y`, the object data stores the filename as 16 characters. The data is only compatible with firmware V60.13.0.5 and above.<br>`N` = No<br>If `N`, the object data stores the filename as 8 characters. The data is forward and backward compatible with all versions of firmware.<br>*Default Value:* `N` |

**Example** • This example shows the object data information for the object `SAMPLE.GRF` located on `R:`.

```
^XA
^HZO,R:SAMPLE.GRF
^XZ
```

# ^ID

## Object Delete

**Description** The ^ID command deletes objects, graphics, fonts, and stored formats from storage areas. Objects can be deleted selectively or in groups. This command can be used within a printing format to delete objects before saving new ones, or in a stand-alone format to delete objects.

The image name and extension support the use of the asterisk (*) as a wild card. This allows you to easily delete a selected groups of objects.

**Format** `^IDd:o.x`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = location of stored object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = object name | *Accepted Values:* any 1 to 8 character name<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Accepted Values:* any extension conforming to Zebra conventions<br>*Default Value:* .GRF |

**Example** • To delete stored formats from DRAM:

```
^XA
^IDR:*.ZPL^FS
^XZ
```

**Example** • To delete formats and images named SAMPLE from DRAM, regardless of the extension:

```
XA
^IDR:SAMPLE.*^FS
^XZ
```

**Example** • To delete the image SAMPLE1.GRF prior to storing SAMPLE2.GRF:

```
^XA
^FO25,25^AD,18,10
^FDDelete^FS
^FO25,45^AD,18,10
^FDthen Save^FS
^IDR:SAMPLE1.GRF^FS
^ISR:SAMPLE2.GRF^FS^XZ
```

**Example** • In this the * is a wild card, indicating that all objects with the .GRF extension are deleted:

```
^XA
^IDR:*.GRF^FS
^XZ
```

**Comments** When an object is deleted from R:, the object can no longer be used and memory is available for storage. This applies only to R: memory. With the other memory types (A:, B:, E:) the deleted object is no longer available. The memory space recovers when an automatic defragmentation or initiallization occurs.

The ^ID command also frees up the uncompressed version of the object in DRAM.

If the name is specified as *.ZOB, all downloaded bar code fonts (or other objects) are deleted.

If the named downloadable object cannot be found in the R:, E:, B:, and A: device, the ^ID command is ignored.

# ^IL

## Image Load

**Description** The ^IL command is used at the beginning of a label format to load a stored image of a format and merge it with additional data. The image is always positioned at ^FO0,0.

**Important •** See *^IS* on page 186.

Using this technique to overlay the image of constant information with variable data greatly increases the throughput of the label format.

**Format** `^ILd:o.x`

This table identifies the parameters for this format:

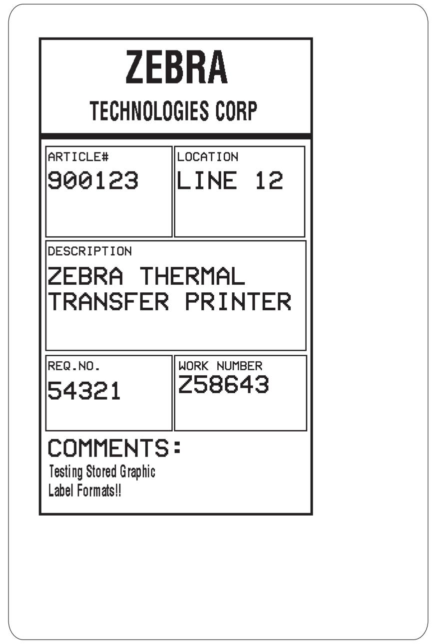
| Parameters | Details |
|---|---|
| d = location of stored object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = object name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Fixed Value:* .GRF |

**Example •** This example recalls the stored image `SAMPLE2.GRF` from DRAM and overlays it with the additional data. The graphic was stored using the `^IS` command. For the stored label format, see the *^IS* on page 186 command.

**ZPL II CODE**

```
^XA
^ILR:SAMPLE2.GRF^FS
^CFD,36,20
^FO15,210
^FD900123^FS
^FO218,210
^FDLINE 12^FS
^FO15,360^AD
^FDZEBRA THERMAL^FS
^FO15,400^AD
^FDTRANSFER PRINTER^FS
^FO15,540
^FD54321^FS
^FO220,530
^FDZ58643^FS
^FO15,670^A0,27,18
^FDTesting Stored Graphic^FS
^FO15,700^A0,27,18
^FDLabel Formats!!^FS
^XZ
```

**GENERATED LABEL**



# ^IM

## Image Move

**Description** The ^IM command performs a direct move of an image from storage area into the bitmap. The command is identical to the ^XG command (Recall Graphic), except there are no sizing parameters.

**Format** ^IMd:o.x

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = location of stored object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* search priority |
| o = object name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Fixed Value:* .GRF |

**Example** • This example moves the image SAMPLE.GRF from DRAM and prints it in several locations in its original size.

```
^XA
^FO100,100^IMR:SAMPLE.GRF^FS
^FO100,200^IMR:SAMPLE.GRF^FS
^FO100,300^IMR:SAMPLE.GRF^FS
^FO100,400^IMR:SAMPLE.GRF^FS
^FO100,500^IMR:SAMPLE.GRF^FS
^XZ
```

**Comments** By using the ^FO command, the graphic image can be positioned anywhere on the label.

The difference between ^IM and ^XG: ^IM does not have magnification, and therefore might require less formatting time. However, to take advantage of this, the image must be at a 8-, 16-, or 32-bit boundary.

# ^IS

## Image Save

**Description** The ^IS command is used within a label format to save that format as a graphic image, rather than as a ZPL II script. It is typically used toward the end of a script. The saved image can later be recalled with virtually no formatting time and overlaid with variable data to form a complete label.

Using this technique to overlay the image of constant information with the variable data greatly increases the throughput of the label format.

**Important** • See *^IL* on page 183.

**Format** ^ISd:o.x,p

This table identifies the parameters for this format:

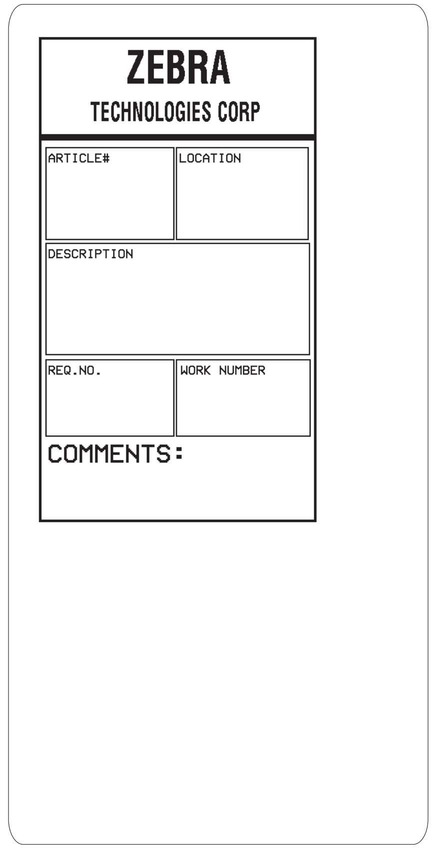
| Parameters | Details |
|---|---|
| d = location of stored object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = object name | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| x = extension | *Accepted Values:* .GRF or .PNG<br>*Default Value:* .GRF |
| p = print image after storing | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y |

**Example •** This is an example of using the `^IS` command to save a label format to DRAM. The name used to store the graphic is `SAMPLE2.GRF`.

**ZPL II CODE**

```
^XA
^LH10,15^FWN^BY3,3,85^CFD,36
^GB430,750,4^FS
^FO10,170^GB200,144,2^FS
^FO10,318^GB410,174,2^FS
^FO212,170^GB206,144,2^FS
^FO10,498^GB200,120,2^FSR
^FO212,498^GB209,120,2^FS
^FO4,150^GB422,10,10^FS
^FO135,20^A0,70,60
^FDZEBRA^FS
^FO80,100^A0,40,30
^FDTECHNOLOGIES CORP^FS
^FO15,180^CFD,18,10^FS
^FDARTICLE#^FS
^FO218,180
^FDLOCATION^FS
^FO15,328
^FDDESCRIPTION^FS
^FO15,508
^FDREQ.NO.^FS
^FO220,508
^FDWORK NUMBER^FS
^FO15,630^AD,36,20
^FDCOMMENTS:^FS
^ISR:SAMPLE2.GRF,Y
^XZ
```

**GENERATED LABEL**



# ~JA

## Cancel All

**Description** The `~JA` command cancels all format commands in the buffer. It also cancels any batches that are printing.

The printer stops after the current label is finished printing. All internal buffers are cleared of data and the DATA LED turn off.

Submitting this command to the printer scans the buffer and deletes only the data before the `~JA` in the input buffer — it does not scan the remainder of the buffer for additional `~JA` commands.

**Format** `~JA`

# ^JB

## Initialize Flash Memory

**Description** The ^JB command is used to initialize various types of Flash memory available in the Zebra printers.

**Format** ^JBa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = device to initialize | *Acceptable Values:*<br>A = Compact Flash memory<br>B = Flash card (PCMCIA)<br>E = internal Flash memory<br>*Default Value:* a device must be specified |

**Example** • This is an example of initializing the different types of flash memory:

^JBA – initializes initial Compact Flash memory when installed in the printer.

^JBB – initializes the optional Flash card when installed in the printer.

^JBE – initializes the optional Flash memory when installed in the printer.

**Note** • Initializing memory can take several minutes. Be sure to allow sufficient time for the initialization to complete before power cycling the printer.

# ~JB

## Reset Optional Memory

**Description** The `~JB` command is used for these conditions:

- The `~JB` command must be sent to the printer if the battery supplying power to the battery powered memory card fails and is replaced. A bad battery shows a *battery dead* condition on the Printer Configuration Label.
- The `~JB` command can also be used to intentionally clear (reinitialize) the `B:` memory card. The card must **not** be write protected.

**Format** `~JB`

**Comments** If the battery is replaced and this command is not sent to the printer, the memory card cannot function.

# ~JC

## Set Media Sensor Calibration

**Description** The `~JC` command is used to force a label length measurement and adjust the media and ribbon sensor values.

**Format** `~JC`

**Comments** In Continuous Mode, only the media and ribbon sensors are calibrated.

# ~JD

## Enable Communications Diagnostics

**Description** The `~JD` command initiates Diagnostic Mode, which produces an ASCII printout (using current label length and full width of printer) of all characters received by the printer. This printout includes the ASCII characters, the hexadecimal value, and any communication errors.

**Format** `~JD`

# ~JE

## Disable Diagnostics

**Description** The `~JE` command cancels Diagnostic Mode and returns the printer to normal label printing.

**Format** `~JE`

# ~JF

## Set Battery Condition

**Description** There are two low battery voltage levels sensed by the *PA*/*PT400*™ printers. When battery voltage goes below the first level, the green LED begins flashing as a warning but printing continues. When this warning occurs, it is recommended to recharge the battery.

As printing continues, a second low voltage level is reached. At this point, both green and orange LEDs flash as a warning, and printing automatically pauses.

When pause on low voltage is active (`~JFY`) and the battery voltage level falls below the second *low voltage* level, printing pauses and an error condition is displayed as an indication that the printer should be plugged into the battery charger. By pressing FEED, printing continues on a label-by-label basis, but there is a high risk of losing label format information due to the continued decrease of battery voltage.

When pause on low voltage is not active (`~JFN`), and the battery voltage level falls below the second *low voltage* level, printing continues and the orange LED remains off. If the battery voltage continues to decrease, label information could be lost and cause the printer to stop operating. This option should be selected only when the printer is connected to the Car Battery Adapter. From time to time the printer might sense that battery voltage is below the first *low voltage* level, but due to the continuous recharging of the car battery, further loss of battery voltage is not a concern and printing continues.

If this option is not selected when using the Car Battery Adapter, you might need to press FEED to take the printer out of Pause Mode and print each label.

**Format** `~JFp`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `p` = pause on low voltage | *Accepted Values:* `Y` (pause on low voltage) or `N` (do not pause)<br>`N` is suggested when the printer is powered by the Car Battery Adapter.<br>*Default Value:* `Y` |

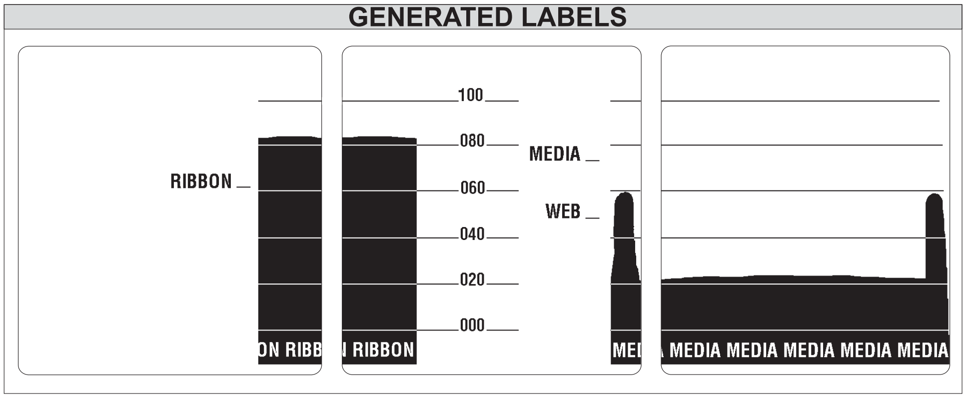
# ~JG

## Graphing Sensor Calibration

**Description** The `~JG` command is used to force a label length measurement, recalibrate the media and ribbon sensors, and print a graph (media sensor profile) of the sensor values.

**Format** `~JG`

**Example •** Sending the `~JG` command to the printer produces a series of labels resembling this image:



# ^JJ

## Set Auxiliary Port

**Description** The ^JJ command allows you to control an online verifier or applicator device.

**Format** ^JJa,b,c,d,e,f

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = operational mode for auxiliary port | *Accepted Values:*<br>0 = off<br>1 = reprint on error—the printer stops on a label with a verification error. When PAUSE is pressed, the label reprints (if ^JZ is set to reprint). If a bar code is near the upper edge of a label, the label feeds out far enough for the bar code to be verified and then backfeeds to allow the next label to be printed and verified.<br>2 = maximum throughput—the printer stops when a verification error is detected. The printer starts printing the next label while the verifier is still checking the previous label. This mode provides maximum throughput, but does not allow the printer to stop immediately on a label with a verification error.<br>*Default Value:* 0 |
| b = application mode | *Accepted Values:*<br>0 = off<br>1 = End Print signal normally high, and low only when the printer is moving the label forward.<br>2 = End Print signal normally low, and high only when the printer is moving the label forward.<br>3 = End Print signal normally high, and low for 20 ms when a label has been printed and positioned.<br>4 = End Print signal normally low, and high for 20 ms when a label has been printed and positioned.<br>*Default Value:* 0 |
| c = application mode start signal print | *Accepted Values:*<br>p = Pulse Mode – Start Print signal must be de-asserted before it can be asserted for the next label.<br>l = Level Mode – Start Print signal does not need to be de-asserted to print the next label. As long as the Start Print signal is low and a label is formatted, a label prints.<br>*Default Value:* 0 |

| Parameters | Details |
| --- | --- |
| `d` = application label error mode | *Accepted Values:*<br>`e` = error mode—the printer asserts the *Service Required* signal (svce_req - pin 10) on the application port, enters into Pause Mode, and displays an error message on the LCD.<br>`f` = Feed Mode—a blank label prints when the web is not found where expected to sync the printer to the media.<br>*Default Value:* `f` |
| `e` = reprint mode | *Accepted Values:*<br>`e` = enabled—the last label reprints after the signal is asserted. If a label is canceled, the label to be reprinted is also canceled. This mode consumes more memory because the last printed label is not released until it reprints.<br>`d` = disabled—printer ignores the Reprint signal.<br>*Default Value:* `d` |
| `f` = ribbon low mode | *Accepted Values:*<br>`e` = *enabled* – printer warning issued when ribbon low.<br>`d` = *disabled* – printer warning not issued when ribbon low.<br>*Default Value:* `e` |

# ~JL

## Set Label Length

**Description** The `~JL` command is used to set the label length. Depending on the size of the label, the printer feeds one or more blank labels.

**Format** `~JL`

# ^JM

## Set Dots per Millimeter

**Description** The ^JM command lowers the density of the print—24 dots/mm becomes 12, 12 dots/mm becomes 6, 8 dots/mm becomes 4, and 6 dots/mm becomes 3. ^JM also affects the field origin (^FO) placement on the label (see example below).

When sent to the printer, the ^JM command doubles the format size of the label. Depending on the printhead, normal dot-per-millimeter capabilities for a Zebra printer are 12 dots/mm (304 dots/inch), 8 dots/mm (203 dots/inch) or 6 dots/mm (153 dots/inch).

This command must be entered before the first ^FS command in a format. The effects of ^JM are persistent.

**Format** ^JMn

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| n = set dots per millimeter | *Accepted Values:*<br>A = 24 dots/mm, 12 dots/mm, 8 dots/mm or 6 dots/mm<br>B = 12 dots/mm, 6 dots/mm, 4 dots/mm or 3 dots/mm<br>*Default Value:* A |

**Example •** This example of the affects of alternating the dots per millimeter:

| ZPL II CODE | GENERATED LABEL |
|---|---|
| `^XA`<br>`^JMA^FS`<br>`^FO100,100`<br>`^B2N,50,Y,N,N`<br>`^FD1234567890^FS`<br>`^XZ` | 1234567890 |
| `^XA`<br>`^JMB^FS`<br>`^FO100,100`<br>`^B2N,50,Y,N,N`<br>`^FD1234567890^FS`<br>`^XZ` | 1234567890 |

**Comments** If ^JMB is used, the UPS MaxiCode bar code becomes out of specification.

# ~JN

## Head Test Fatal

**Description** The ~JN command turns on the head test option. When activated, ~JN causes the printer to halt when a head test failure is encountered.

Once an error is encountered the printer remains in error mode until the head test is turned off (~JO) or power is cycled.

**Format** ~JN

**Comments** If the communications buffer is full, the printer is not able to receive data. In this condition, the ~JO command is not received by the printer.

# ~JO

## Head Test Non-Fatal

**Description** The `~JO` command turns off the head test option. `~JO` is the default printhead test condition and overrides a failure of printhead element status check. This state is changed when the printer receives a `~JN` (Head Test Fatal) command. The printhead test does not produce an error when `~JO` is active.

**Format** `~JO`

# ~JP

## Pause and Cancel Format

**Description** The `~JP` command clears the format currently being processed and places the printer into Pause Mode.

The command clears the next format that would print, or the oldest format from the buffer. Each subsequent `~JP` command clears the next buffered format until the buffer is empty. The DATA indicator turns off when the buffer is empty and no data is being transmitted.

Issuing the `~JP` command is identical to using CANCEL on the printer, but the printer does not have to be in Pause Mode first.

**Format** `~JP`

# ~JR

## Power On Reset

**Description** The `~JR` command resets all of the printer's internal software, performs a power-on self-test (POST), clears the buffer and DRAM, and resets communication parameters and default values. Issuing a `~JR` command performs the same function as a manual power-on reset.

**Format** `~JR`

# ^JS

## Sensor Select

**Format** `^JSa`

**Note •** This command is only for the Z-Series and S4M printers, only.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = sensor selection | *Accepted Values:*<br>`A` = auto select<br>`R` = reflective sensor<br>`T` = transmissive sensor<br>*Default Value:* Z-Series = `A` and the S4M = `R` |

# ~JS

## Change Backfeed Sequence

**Description** The `~JS` command is used to control the backfeed sequence. This command can be used on printers with or without built-in cutters.

These are the primary applications:

- to allow programming of the *rest point* of the cut edge of continuous media.
- provide immediate backfeed after peel-off when the printer is used in a print/apply application configuration.

This command stays in effect only until the printer is turned off, a new `~JS` command is sent, or the setting is changed on the control panel. When a `~JS` command is encountered, it overrides the current control panel setting for the Backfeed Sequence.

The most common way of eliminating backfeed is to operate in Rewind Mode. Rewind Mode does not backfeed at all. After a label prints, the leading edge of the next label is placed at the print line. This eliminates the need to backfeed and does not introduce a non printable area at the leading edge or bottom of the label. It also does not allow the label to be taken from the printer because it is not fed out from under the printhead.

Running in another mode with backfeed turned off allows the label to be removed and eliminates the time-reduction of the backfeed sequence.

**Format** `~JSb`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `b` = backfeed order in relation to printing | *Accepted Values:*<br>`A` = 100 percent backfeed after printing and cutting<br>`B` = 0 percent backfeed after printing and cutting, and 100 percent before printing the next label<br>`N` = normal — 90 percent backfeed after label is printed<br>`O` = off — turn backfeed off completely<br>`10` to `90` = percentage value<br>The value entered must be a multiple of 10. Values not divisible by 10 are rounded to the nearest acceptable value. For example, `~JS55` is accepted as 50 percent backfeed.<br>*Default Value:* `N` |

**Comments** When using a specific value, the difference between the value entered and 100 percent is calculated before the next label is printed. For example, a value of 40 means 40 percent of the backfeed takes place after the label is cut or removed. The remaining 60 percent takes place before the next label is printed.

The value for this command is also reflected in the Backfeed parameter on the printer configuration label.

For `~JSN` — the Backfeed parameter is listed as DEDAULT

For `~JSA` — or 100% the Backfeed parameter is listed as AFTER

For `~JSB` — or 0% the Backfeed parameter is listed as BEFORE

For `~JS10` — 10% of the backfeed takes place after the label is cut or removed. The remaining 90% takes place before the next label is printed.

# ^JT

## Head Test Interval

**Description** The ^JT command allows you to change the printhead test interval from every 100 labels to any desired interval. With the ^JT command, the printer is allowed to run the test after printing a label. When a parameter is defined, the printer runs the test after printing a set amount of labels.

The printer's default head test state is off. Parameters for running the printhead test are defined by the user.

**Format** `^JT####,a,b,c`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `####` = four-digit number of labels printed between head tests | *Accepted Values:* `0000` to `9999`<br>If a value greater than `9999` is entered, it is ignored.<br>*Default Value:* `0000` (off) |
| `a` = manually select range of elements to test | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Initial Value at Power-up:* `N` |
| `b` = first element to check when parameter `a` is Y | *Accepted Values:* `0` to `9999`<br>*Initial Value at Power-up:* `0` |
| `c` = last element to check when parameter `a` is Y | *Accepted Values:* `0` to `9999`<br>*Initial Value at Power-up:* `9999` |

**Comments** The ^JT command supports testing a range of print elements. The printer automatically selects the test range by tracking which elements have been used since the previous test.

^JT also turns on Automatic Mode to specify the first and last elements for the head test. This makes it possible to select any specific area of the label or the entire print width.

If the last element selected is greater than the print width selected, the test stops at the selected print width.

Whenever the head test command is received, a head test is performed on the next label unless the count is set to 0 (zero).

# ^JU

## Configuration Update

**Description** The ^JU command sets the active configuration for the printer.

**Format** ^JUa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = active configuration | *Accepted Values:*<br>F = reload factory values<br>These values are lost at power-off if not saved with ^JUS.<br>R = recall last saved values<br>S = save current settings<br>These values are used at power-on.<br>*Default Value:* a value must be specified |

# ^JW

## Set Ribbon Tension

**Description** ^JW sets the ribbon tension for the printer it is sent to.

**Format** ^JWt

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| t = tension | *Accepted Values:*<br>L = low<br>M = medium<br>H = high<br>*Default Value:* a value must be specified |

**Comments** ^JW is used only for *PAX* series printers.

# ~JX

## Cancel Current Partially Input Format

**Description** The `~JX` command cancels a format currently being sent to the printer. It does not affect any formats currently being printed, or any subsequent formats that might be sent.

**Format** `~JX`

# ^JZ

## Reprint After Error

**Description** The ^JZ command reprints a partially printed label caused by a Ribbon Out, Media Out, or Head Open error condition. The label is reprinted as soon as the error condition is corrected.

This command remains active until another ^JZ command is sent to the printer or the printer is turned off.

**Format** ^JZa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = reprint after error | *Accepted Values:*<br>N = no<br>Y = yes<br>*Initial Value at Power-up:* Y |

**Comments** ^JZ sets the error mode for the printer. If ^JZ changes, only labels printed after the change are affected.

If the parameter is missing or incorrect, the command is ignored.

# ~KB

## Kill Battery (Battery Discharge Mode)

**Description** To maintain performance of the rechargeable battery in the portable printers, the battery must be fully discharged and recharged regularly. The `~KB` command places the printer in battery discharge mode. This allows the battery to be drained without actually printing.

**Format** `~KB`

**Comments** While the printer is in Discharge Mode, the green power LED flashes in groups of three flashes.

Discharge Mode might be terminated by sending a printing format to the printer or by pressing either of the control panel keys.

If the battery charger is plugged into the printer, the battery is automatically recharged once the discharge process is completed.

# ^KD

## Select Date and Time Format (for Real Time Clock)

**Description** The ^KD command selects the format that the Real-Time Clock's date and time information presents as on a configuration label. This is also displayed on the *Printer Idle* LCD control panel display, and displayed while setting the date and time.

**Format** ^KDa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = value of date and time format | *Accepted Values:*<br>0 = normal, displays *Version Number* of firmware<br>1 = MM/DD/YY (24-hour clock)<br>2 = MM/DD/YY (12-hour clock)<br>3 = DD/MM/YY (24-hour clock)<br>4 = DD/MM/YY (12-hour clock)<br>*Default Value:* 0 |

**Comments** If the Real-Time Clock hardware is not present, Display Mode is set to 0 (Version Number).

If Display Mode is set to 0 (Version Number) and the Real-Time Clock hardware is present, the date and time format on the configuration label is presented in format 1.

If Display Mode is set to 0 (Version Number) and the Real-Time Clock hardware is present, the date and time format on the control panel display is presented in format 1.

# ^KL

## Define Language

**Description** The ^KL command selects the language displayed on the control panel.

**Format** ^KLa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = language | *Accepted Values:*<br>1 = English<br>2 = Spanish<br>3 = French<br>4 = German<br>5 = Italian<br>6 = Norwegian<br>7 = Portuguese<br>8 = Swedish<br>9 = Danish<br>10 = Spanish2<br>11 = Dutch<br>12 = Finnish<br>13 = Japanese<br>*Default Value:* 1 |

# ^KN

## Define Printer Name

**Description** The printer's network name and description can be set using the `^KN` command. `^KN` is designed to make your Zebra printer easy for users to identify. The name the administrator designates is listed on the configuration label and on the Web page generated by the printer.

**Format** `^KNa,b`

This table identifies the parameters for this format:

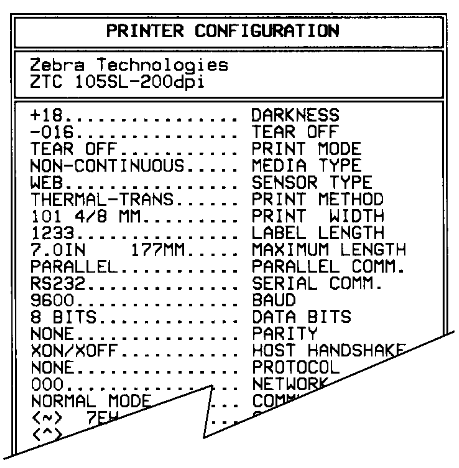
| Parameters | Details |
|---|---|
| a = printer name | *Accepted Values:* up to 16 alphanumeric characters<br>*Default Value:* if a value is not entered, the parameter is ignored<br>If more than 16 characters are entered, only the first 16 are used. |
| b = printer description | *Accepted Values:* up to 35 alphanumeric characters<br>*Default Value:* if a value is not entered, the parameter is ignored<br>If more than 35 characters are entered, only the first 35 are used. |

**Example •** This is an example of how to change the printer's network name an description:

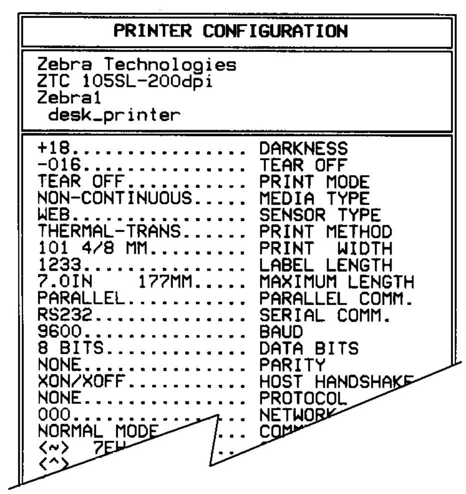
This shows how a configuration looks before using this command and after using this command:

```
^XA
^KNZebra1,desk_printer
^XZ
```

**Before using this command:**




**After using this command:**



# ^KP

## Define Password

**Description** The ^KP command is used to define the password that must be entered to access the control panel switches and LCD Setup Mode.

**Format** ^KPa,b

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = mandatory four-digit password | *Accepted Values:* any four-digit numeric sequence<br>*Default Value:* 1234 |
| b = password level | *Accepted Values:* 1, 2, 3, 4<br>*Default Value:* 3 |

**Example** • This is an example of how to set a new control panel password:

```
^XA
^KP5678
^XZ
```

**Comments** If you forget your password, the printer can be returned to a default Setup Mode and the default password *1234* is valid again. Caution should be used, however — this also sets the printer configuration values back to their defaults.

To return the printer to the default factory settings using ZPL, send this:

```
^XA
^JUF
^XZ
```

To return the printer to the default factory settings using the control panel keys, see your printer's User Guide for the procedure.

# ^LH

## Label Home

**Description** The ^LH command sets the label home position.

The default home position of a label is the upper-left corner (position 0,0 along the x and y axis). This is the axis reference point for labels. Any area below and to the right of this point is available for printing. The ^LH command changes this reference point. For instance, when working with preprinted labels, use this command to move the reference point below the preprinted area.

This command affects only fields that come after it. It is recommended to use ^LH as one of the first commands in the label format.

**Format** `^LHx,y`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `x` = x-axis position (in dots) | *Accepted Values:* `0` to `32000`<br>*Initial Value at Power-up:* 0 or last permanently saved value |
| `y` = y-axis position (in dots) | *Accepted Values:* `0` to `32000`<br>*Initial Value at Power-up:* `0` or last permanently saved value |

Depending on the printhead used in your printer, use one of these when figuring the values for `x` and `y`:

6 dots = 1 mm, 152 dots = 1 inch
8 dots = 1 mm, 203 dots = 1 inch
11.8 dots = 1 mm, 300 dots = 1 inch
24 dots = 1 mm, 608 dots = 1 inch

**Comments** To be compatible with existing printers, this command must come before the first ^FS (Field Separator) command. Once you have issued an ^LH command, the setting is retained until you turn off the printer or send a new ^LH command to the printer.

# ^LL

## Label Length

**Description** The ^LL command defines the length of the label. This command is necessary when using continuous media (media not divided into separate labels by gaps, spaces, notches, slots, or holes).

To affect the current label and be compatible with existing printers, ^LL must come before the first ^FS (Field Separator) command. Once you have issued ^LL, the setting is retained until you turn off the printer or send a new ^LL command.

**Format** `^LLy`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `y` = y-axis position (in dots) | *Accepted Values:* `1` to `32000`, not to exceed the maximum label size.<br>While the printer accepts any value for this parameter, the amount of memory installed determines the maximum length of the label.<br>*Default Value:* typically set through the LCD (if applicable), or to the maximum label length capability of the printer. |

**Comments** These formulas can be used to determine the value of y:

| | |
|---|---|
| **For 6 dot/mm printheads...** | Label length in inches x 152.4 (dots/inch) = y |
| **For 8 dot/mm printheads...** | Label length in inches x 203.2 (dots/inch) = y |
| **For 12 dot/mm printheads...** | Label length in inches x 304.8 (dots/inch) = y |
| **For 24 dot/mm printheads...** | Label length in inches x 609.6 (dots/inch) = y |

Values for `y` depend on the memory size. If the entered value for `y` exceeds the acceptable limits, the bottom of the label is cut off. The label also shifts down from top to bottom.

If multiple ^LL commands are issued in the same label format, the last ^LL command affects the next label unless it is prior to the first ^FS.

# ^LR

## Label Reverse Print

**Description** The ^LR command reverses the printing of all fields in the label format. It allows a field to appear as white over black or black over white.

Using the ^LR is identical to placing an ^FR command in all current and subsequent fields.

**Format** ^LRa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = reverse print all fields | *Accepted Values:*<br>N = no<br>Y = yes<br>*Initial Value at Power-up:* N or last permanently saved value |

**Example** • This is an example that shows printing white over black and black over white. The ^GB command is used to create the black background.

**ZPL II CODE**

```
^XA^LRY
^FO100,50
^GB195,203,195^FS
^FO180,110^CFG
^FDLABEL^FS
^FO130,170
^FDREVERSE^FS
^XZ
```

**GENERATED LABEL**


**Comments** The ^LR setting remains active unless turned off by ^LRN or the printer is turned off.

**Note** • ^GB needs to be used together with ^LR.

Only fields following this command are affected.

# ^LS

## Label Shift

**Description** The ^LS command allows for compatibility with Z-130 printer formats that are set for less than full label width. It is used to shift all field positions to the left so the same commands used on a Z-130 or Z-220 Printer can be used on other Zebra printers.

To determine the value for the ^LS command, use this formula:

`Z-130 and Z-220 values for ^LHx + ^FOx`

(distance from edge of label) = printer value for ^LSa

If the print position is less than 0, set ^LS to 0.

**Format** ^LSa

**Important •** The ability to save the ^LS command depends on the version of firmware.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = shift left value (in dots) | *Accepted Values:* -9999 to 9999<br>*Initial Value at Power-up:* 0 |

**Comments** When entering positive values, it is not necessary to use the + sign. The value is assumed to be positive unless preceded by a negative sign (-).

To be compatible with existing Zebra printers, this command must come before the first ^FS (Field Separator) command. Once you have issued an ^LS command, the setting is retained until you turn off the printer or send a new ^LS command to the printer.

# ^LT

## Label Top

**Description** The ^LT command moves the entire label format a maximum of 120 dot rows up or down from its current position, in relation to the top edge of the label. A negative value moves the format towards the top of the label; a positive value moves the format away from the top of the label.

This command can be used to fine-tune the position of the finished label without having to change any of the existing parameters.

**Important** • For some printer models, it is possible to request a negative value large enough to cause the media to backup into the printer and become unthreaded from the platen. This condition can result in a printer error or unpredictable results.

**Format** ^LTx

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| x = label top (in dot rows) | *Accepted Values:* -120 to 120<br>*Default Value:* a value must be specified or the command is ignored |

**Comments** The Accepted Value range for x might be smaller depending on the printer platform.

The ^LT command does not change the media rest position.

# ^MC

## Map Clear

**Description** In normal operation, the bitmap is cleared after the format has been printed. The ^MC command is used to retain the current bitmap. This applies to current and subsequent labels until cleared with ^MCY.

**Format** ^MCa

**Important •** To produce a label template, ^MC must be used with ^FV.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = map clear | *Accepted Values:* Y (clear bitmap) or N (do not clear bitmap)<br>*Initial Value at Power-up:* Y |

**Comments** The ^MC command retains the image of the current label after formatting. It appears in the background of the next label printed.

# ^MD

## Media Darkness

**Description** The ^MD command adjusts the darkness relative to the current darkness setting.

**Format** ^MDa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = media darkness level | *Accepted Values:* -30 to 30, depending on current value<br>*Initial Value at Power-up:* 0<br>If no value is entered, this command is ignored. |

**Example •** These examples show setting the printer to different darkness levels:

- If the current value (value on configuration label) is 16, entering the command ^MD-9 decreases the value to 7.
- If the current value (value on configuration label) is 1, entering the command ^MD15 increases the value to 16.
- If the current value (value on configuration label) is 25, entering the command ^MD10 increases only the value to 30, which is the maximum value allowed.

Each ^MD command is treated separately in relation to the current value as printed on the configuration label.

**Important •** The darkness setting range for the *Xi*III*Plus* is 0 to 30 in increments of 0.1. The firmware is setup so that the ^MD and ~SD commands (ZPL darkness commands) accepts that range of settings.

**Example •** These are examples of the *Xi*III*Plus* Darkness Setting:

```
^MD8.3
```

```
~SD8.3
```

**Example •** For example, this is what would happen if two ^MD commands were received:

Assume the current value is 15. An ^MD-6 command is received that changes the current value to 9. Another command, ^MD2, is received. The current value changes to 17.

The two ^MD commands are treated individually in relation to the current value of 15.

**Comments** The ~SD command value, if applicable, is added to the ^MD command.

# ^MF

## Media Feed

**Description** The ^MF command dictates what happens to the media at power-up and at head-close after the error clears.

**Format** ^MFp,h

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| p = feed action at power-up | *Accepted Values:*<br>F = feed to the first web after sensor<br>C = (see ~JC on page 191 definition)<br>L = (see ~JL on page 198 definition)<br>N = no media feed<br>*Default Value:* platform-dependent |
| h = feed action after closing printhead | *Accepted Values:*<br>F = feed to the first web after sensor<br>C = (see ~JC on page 191 definition)<br>L = (see ~JL on page 198 definition)<br>N = no media feed<br>*Default Value:* platform-dependent |

**Comments** It is important to remember that if you choose the N setting, the printer assumes that the media and its position relative to the printhead are the same as before power was turned off or the printhead was opened. Use the ^JU command to save changes.

# ^ML

## Maximum Label Length

**Description** The ^ML command lets you adjust the maximum label length.

**Format** ^MLa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = maximum label length (in dot rows) | *Accepted Values:* 0 to maximum length of label<br>*Default Value:* last permanently saved value |

**Comments** For calibration to work properly, you must set the maximum label length equal to or greater than your actual label length.

# ^MM

## Print Mode

**Description** The ^MM command determines the action the printer takes after a label or group of labels has printed.

**Format** ^MMa,b

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = desired mode | *Accepted Values:*<br>T = Tear-off<br>P = Peel-off (not available on *S*-300)<br>R = Rewind (depends on printer model)<br>A = Applicator (depends on printer model)<br>C = Cutter (depends on printer model)<br>D = Delayed cutter<br>*Default Value:* T<br>The values available for parameter a are dependent on the printer being used and whether it supports the option. |
| b = prepeel select | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y<br>The command is ignored if parameters are missing or invalid. The current value of the command remains unchanged. |

This bulleted list identifies the different modes of operation:

- Tear-off — after printing, the label advances so the web is over the tear bar. The label, with liner attached, can be torn off manually.
- Peel-off — after printing, the label moves forward and activates a Label Available Sensor. Printing stops until the label is manually removed from the printer.

  *Power Peel* – liner automatically rewinds using an optional internal rewind spindle.

  *Value Peel* – liner feeds down the front of the printer and is manually removed.

  *Prepeel* – after each label is manually removed, the printer feeds the next label forward to prepeel a small portion of the label away from the liner material. The printer then backfeeds and prints the label. The prepeel feature assists in the proper peel operation of some media types.
- Rewind — the label and backing are rewound on an (optional) external rewind device. The next label is positioned under the printhead (no backfeed motion).
- Applicator — when used with an application device, the label move far enough forward to be removed by the applicator and applied to an item.

Cutter — after printing, the media feeds forward and is automatically cut into predetermined lengths.

**Comments** Be sure to select the appropriate value for the print mode being used to avoid unexpected results.

The Delayed Cut feature can be activated as follows:

- through PRINT MODE on the printer's control panel
- with a `^MMD` command

**Note •** When the printer is in the Delayed Cut PRINT MODE, it will cut the label when it receives the `~JK` (Delayed Cut) command. To activate the `~JK` command, the printer's PRINT MODE must be set to Delayed Cut and there must be a label waiting to be cut.

When the printer is not in the Delayed Cut PRINT MODE, the printer will not cut the label when it receives the `~JK` command.

# ^MN

## Media Tracking

**Description** The ^MN command relays to the printer what type of media is being used (continuous or non-continuous) for purposes of tracking. This bulleted list shows the types of media associated with this command:

- Continuous Media – this media has no physical characteristic (web, notch, perforation, mark, et cetera) to separate labels. Label length is determined by the ^LL command.
- Non-continuous Media – this media has some type of physical characteristic (web, notch, perforation, mark, et cetera) to separate the labels.

**Format** ^MNa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = media being used | *Accepted Values:*<br>N = continuous media<br>Y = non-continuous media web sensing *<br>W = non-continuous media web sensing *<br>M = non-continuous media mark sensing<br>*Default Value:* a value must be entered or the command is ignored |

* provides the same result.

# ^MP

## Mode Protection

**Description** The ^MP command is used to disable the various mode functions on the control panel. Once disabled, the settings for the particular mode function can no longer be changed and the LED associated with the function does not light.

Because this command has only one parameter, each mode must be disabled with an individual ^MP command.

**Format** ^MPa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = mode to protect | *Accepted Values:*<br>D = disable Darkness Mode<br>P = disable Position Mode<br>C = disable Calibration Mode<br>E = enable all modes<br>S = disable all mode saves (modes can be adjusted but values are not saved)<br>W = disable Pause<br>F = disable Feed<br>X = disable Cancel<br>M = disable menu changes<br>*Default Value:* a value must be entered or the command is ignored |

**Example •** This example disables these modes, D and C.

```
^XA
^MPD
^MPC
^XZ
```

# ^MT

## Media Type

**Description** The ^MT command selects the type of media being used in the printer. There are two choices for this command:

- Thermal Transfer Media – this media uses a high-carbon black or colored ribbon. The ink on the ribbon is bonded to the media.
- Direct Thermal Media – this media is heat sensitive and requires no ribbon.

**Format** ^MTa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = media type used | *Accepted Values:*<br>T = thermal transfer media<br>D = direct thermal media<br>*Default Value:* a value must be entered or the command is ignored |

# ^MU

## Set Units of Measurement

**Description** The ^MU command sets the units of measurement the printer uses. ^MU works on a field-by-field basis. Once the mode of units is set, it carries over from field to field until a new mode of units is entered.

^MU also allows for printing at lower resolutions — 600 dpi printers are capable of printing at 300, 200, and 150 dpi; 300 dpi printers are capable of printing at 150 dpi.

**Format** `^MUa,b,c`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = units | *Accepted Values:*<br>`D` = dots<br>`I` = inches<br>`M` = millimeters<br>*Default Value:* `D` |
| `b` = format base in dots per inch | *Accepted Values:* `150`, `200`, `300`<br>*Default Value:* a value must be entered or the command is ignored |
| `c` = desired dots-per-inch conversion | *Accepted Values:* `300`, `600`<br>*Default Value:* a value must be entered or the command is ignored |

**Example** • This is an example of Setting Units:

Assume 8 dot/millimeter (203 dot/inch) printer.

*Field based on dots:*

```
^MUd^FO100,100^GB1024,128,128^FS
```

*Field based on millimeters:*

```
^MUm^FO12.5,12.5^GB128,16,16^FS
```

*Field based on inches:*

```
^MUi^FO.493,.493^GB5.044,.631,.631^FS
```

**Example** • This is an example of Converting dpi Values.

*Convert a 150 dpi format to a 300 dpi format with a base in dots:*

```
^MUd,150,300
```

*Convert a 150 dpi format to a 600 dpi format with a base in dots:*

```
^MUd,150,600
```

*Convert a 200 dpi format to a 600 dpi format with a base in dots:*

```
^MUd,200,600
```

*To reset the conversion factor to the original format, enter matching values for parameters* `b` *and* `c`*:*

```
^MUd,150,150
^MUd,200,200
^MUd,300,300
^MUd,600,600
```

**Comments** This command should appear at the beginning of the label format to be in proper ZPL II format.

To turn the conversion off, enter matching values for parameter `b` and `c`.

# ^MW

## Modify Head Cold Warning

**Description** The ^MW command allows you to set the head cold warning indicator based on the operating environment.

**Format** ^MWy

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = enable head cold warning | *Accepted Values:*<br>y = enable head cold warning<br>n = disable head cold warning |

**Important •** When a parameter is **not** given, the instruction is *ignored*.

# ~NC

## Network Connect

**Description** The ~NC command is used to connect a particular printer to a network by calling up the printer's network ID number.

**Format** `~NC###`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| ### = network ID number assigned (must be a three-digit entry) | *Accepted Values:* 001 to 999<br>*Default Value:* 000 (none) |

**Comments** Use this command at the beginning of any label format to specify which printer on the network is going to be used. Once the printer is established, it continues to be used until it is changed by another ~NC command. This command must be included in the label format to *wake up the printer.*

The commands ^MW, ~NC, ^NI, ~NR, and ~NT are used only with RS-485 printer communications.

# ^NI

## Network ID Number

**Description** The ^NI command is used to assign a network ID number to the printer. This must be done before the printer can be used in a network.

**Format** ^NI###

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| ### = network ID number assigned (must be a three-digit entry) | *Accepted Values:* 001 to 999<br>*Default Value:* 000 (none) |

**Comments** The last network ID number set is the one recognized by the system.

The commands ~NC, ^NI, ~NR, and ~NT are used only with RS-485 printer communications.

# ~NR

## Set All Network Printers Transparent

**Description** The `~NR` command sets all printers in the network to be transparent, regardless of ID or current mode.

**Format** `~NR`

**Comments** The commands `~NC`, `^NI`, `~NR`, and `~NT` are used only with RS-485 printer communications.

# ^NS

## Change Networking Settings

**Description** The ^NS command is used to change Ethernet network settings.

**Format** ^NSa,b,c,d

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = network setting | *Accepted Values:*<br>IP Resolution.<br>a = all<br>b = BOOTP<br>c = DHCP and BOOTP<br>d = DHCP<br>g = gleaning only<br>r = RARP<br>p = permanent |
| b = IP Address | *Accepted Values:* 0 to 255 |
| c = Subnet Mask | *Accepted Values:* 0 to 255 |
| d = Default Gateway | *Accepted Values:* 0 to 255 |

# ~NT

## Set Currently Connected Printer Transparent

**Description** The `~NT` command sets the currently connected network printer to be transparent.

**Format** `~NT`

**Comments** With Z Series® printers, the `~NT` command functions the same as the `~NR` command. All Z Series printers on a network receive the transmission.

The commands `~NC`, `^NI`, `~NR`, and `~NT` are used only with RS-485 printer communications.

# ^PF

## Slew Given Number of Dot Rows

**Description** The `^PF` command causes the printer to slew labels (move labels at a high speed without printing) a specified number of dot rows from the bottom of the label. This allows faster printing when the bottom portion of a label is blank.

**Format** `^PF#`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| # = number of dots rows to slew | *Accepted Values:* `0` to `32000`<br>*Default Value:* a value must be entered or the command is ignored |

# ^PH ~PH

## Slew to Home Position

**Description** The ^PH or ~PH command causes the printer to feed one blank label.

The ~PH command feeds one label after the format currently being printed is done or when the printer is placed in pause.

The ^PH command feeds one blank label after the current format prints.

**Format** `^PH or ~PH`

# ^PM

## Printing Mirror Image of Label

**Description** The ^PM command prints the entire printable area of the label as a mirror image. This command flips the image from left to right.

**Format** ^PMa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = print mirror image of entire label | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

**Example •** This is an example of printing a mirror image on a label:

**ZPL II CODE**

```
^XA^PMY
^FO100,100
^CFG
^FDMIRROR^FS
^FO100,160
^FDIMAGE^FS
^XZ
```

**GENERATED LABEL**


**Comments** If the parameter is missing or invalid, the command is ignored. Once entered, the ^PM command remains active until ^PMN is received or the printer is turned off.

# ^PO

## Print Orientation

**Description** The ^PO command inverts the label format 180 degrees. The label appears to be printed upside down. If the original label contains commands such as ^LL, ^LS, ^LT and ^PF, the inverted label output is affected differently.

**Format** ^POa

This table identifies the parameters for this format:

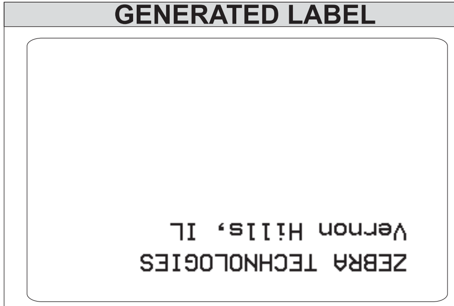
| Parameters | Details |
|---|---|
| a = invert label 180 degrees | *Accepted Values:* N (normal) or I (invert)<br>*Default Value:* N |

**Example •** This is an example of printing a label at 180 degrees:

**ZPL II CODE**

```
^XA^CFD
^POI
^LH330,10
^FO50,50
^FDZEBRA TECHNOLOGIES^FS
^FO50,75
^FDVernon Hills, IL^FS
^XZ
```

**GENERATED LABEL**




The ^POI command inverts the x, y coordinates. All image placement is relative to these inverted coordinates. Therefore, a different ^LH (Label Home) can be used to move the print back onto the label.

**Comments** If multiple ^PO commands are issued in the same label format, only the last command sent to the printer is used.

Once the ^PO command is sent, the setting is retained until another ^PO command is received or the printer is turned off.

# ^PP ~PP

## Programmable Pause

**Description** The `~PP` command stops printing after the current label is complete (if one is printing) and places the printer in Pause Mode.

The `^PP` command is not immediate. Therefore, several labels might print before a pause is performed. This command pauses the printer after the current format prints.

The operation is identical to pressing PAUSE on the control panel of the printer. The printer remains paused until PAUSE is pressed or a `~PS` (Print Start) command is sent to the printer.

**Format** `^PP or ~PP`

# ^PQ

## Print Quantity

**Description** The ^PQ command gives control over several printing operations. It controls the number of labels to print, the number of labels printed before printer pauses, and the number of replications of each serial number.

**Format** ^PQq,p,r,o

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| q = total quantity of labels to print | *Accepted Value:* 1 to 99,999,999<br>*Default Value:* 1 |
| p = pause and cut value (labels between pauses) | *Accepted Value:* 1 to 99,999,999<br>*Default Value:* 0 (no pause) |
| r = replicates of each serial number | *Accepted Value:* 0 to 99,999,999 replicates<br>*Default Value:* 0 (no replicates) |
| o = override pause count | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N |

If the o parameter is set to Y, the printer cuts but does not pause, and the printer does **not** pause after every group count of labels has been printed. With the o parameter set to N (default), the printer pauses after every group count of labels has been printed.

**Example** • This example shows the control over print operations:

**^PQ50,10,1,Y**: This example prints a total of 50 labels with one replicate of each serial number. It prints the total quantity in groups of 10, but does not pause after every group.

**^PQ50,10,1,N**: This example prints a total of 50 labels with one replicate of each serial number. It prints the total quantity in groups of 10, pausing after every group.

# ^PR

## Print Rate

**Description** The ^PR command determines the media and slew speed (feeding a blank label) during printing.

The printer operates with the selected speeds until the setting is reissued or the printer is turned off.

The print speed is application-specific. Because print quality is affected by media, ribbon, printing speeds, and printer operating modes, it is very important to run tests for your applications.

**Important •** Some models go to default print speed when power is turned off.

**Format** ^PRp,s,b

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| p = print speed | *Accepted Values:*<br>A or 2 = 50.8 mm/sec. (2 inches/sec.)<br>B or 3 = 76.2 mm/sec. (3 inches/sec.)<br>C or 4 = 101.6 mm/sec. (4 inches/sec.)<br>5 = 127 mm/sec.(5 inches/sec.)<br>D or 6 = 152.4 mm/sec. (6 inches/sec.)<br>E or 8 = 203.2 mm/sec. (8 inches/sec.)<br>9 = 220.5 mm/sec. 9 inches/sec.)<br>10 = 245 mm/sec.(10 inches/sec.)<br>11 = 269.5 mm/sec.(11 inches/sec.)<br>12 = 304.8 mm/sec. 12 inches/sec.)<br>*Default Value:* A |

| Parameters | Details |
|---|---|
| s = slew speed | *Accepted Values:*<br>A or 2 = 50.8 mm/sec. (2 inches/sec.)<br>B or 3 = 76.2 mm/sec. (3 inches/sec.)<br>C or 4 = 101.6 mm/sec. (4 inches/sec.)<br>5 = 127 mm/sec. 5 inches/sec.)<br>D or 6 = 152.4 mm/sec. (6 inches/sec.)<br>E or 8 = 203.2 mm/sec. (8 inches/sec.)<br>9 = 220.5 mm/sec. (9 inches/sec.)<br>10 = 245 mm/sec. (10 inches/sec.)<br>11 = 269.5 mm/sec. 11 inches/sec.)<br>12 = 304.8 mm/sec. 12 inches/sec.)<br>*Default Value:* D |
| b = backfeed speed | *Accepted Values:*<br>A or 2 = 50.8 mm/sec. (2 inches/sec.)<br>B or 3 = 76.2 mm/sec. (3 inches/sec.)<br>C or 4 = 101.6 mm/sec. (4 inches/sec.)<br>5 = 127 mm/sec.(5 inches/sec.)<br>D or 6 = 152.4 mm/sec. (6 inches/sec.)<br>E or 8 = 203.2 mm/sec. (8 inches/sec.)<br>9 = 220.5 mm/sec. 9 inches/sec.)<br>10 = 245 mm/sec. 10 inches/sec.)<br>11 = 269.5 mm/sec. 11 inches/sec.)<br>12 = 304.8 mm/sec. 12 inches/sec.)<br>*Default Value:* A |

**Comments** The speed setting for p, s, and b is dependent on the limitations of the printer. If a particular printer is limited to a rate of 6 ips (inches per second), a value of 12 can be entered but the printer performs only at a 6 ips rate. See your printer's User Guide for specifics on performance.

# ~PR

## Applicator Reprint

**Description** The ~PR command is supported only by the *PAX* and *PAX*2-Series printers. If the ~PR command is enabled (see ^JJ on page 196), the last label printed reprint, similar to the applicator asserting the Reprint signal on the applicator port.

**Format** ~PR

**Comments** Pressing PREVIOUS on the control panel also causes the last label to reprint.

# ~PS

## Print Start

**Description** The `~PS` command causes a printer in Pause Mode to resume printing. The operation is identical to pressing PAUSE on the control panel of the printer when the printer is already in Pause Mode.

**Format** `~PS`

# ^PW

## Print Width

**Description** The ^PW command allows you set the print width.

**Format** ^PWa

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = label width (in dots) | *Accepted Values:* 2, to the width of the label<br>If the value exceeds the width of the label, the width is set to the label's maximum size.<br>*Default Value:* last permanently saved value |

**Limitation** Not all Zebra printers support the ^PW command.

# ~RO

## Reset Advanced Counter

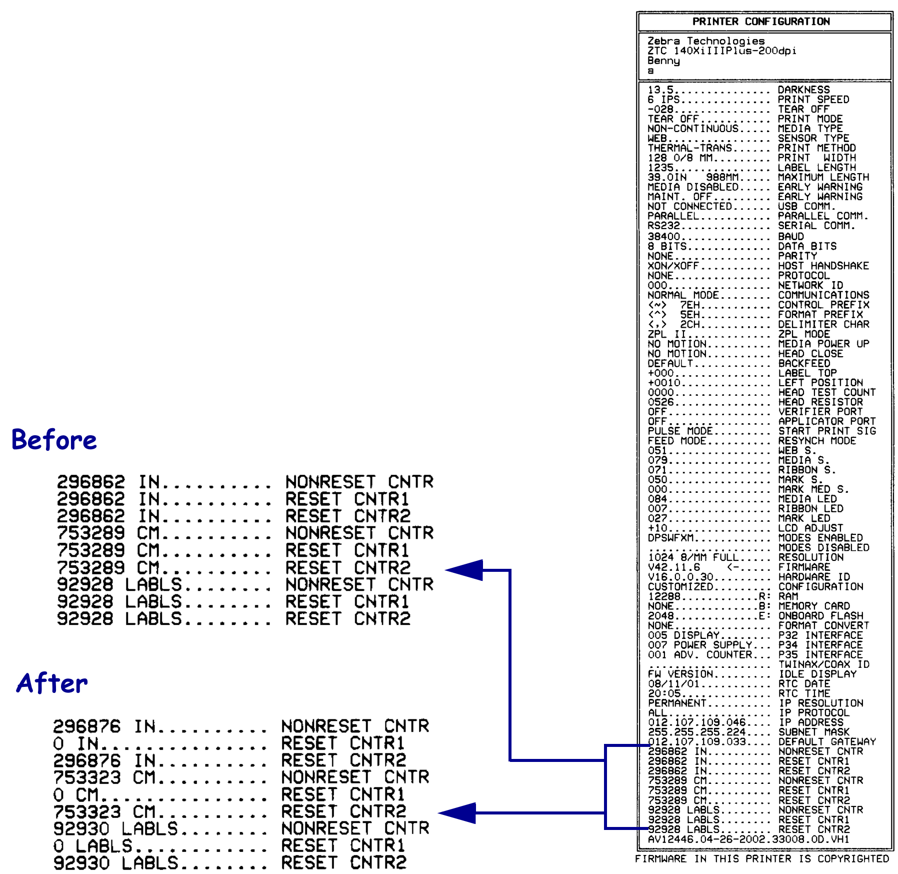
**Description** The `~RO` command resets the advanced counters used by the printer to monitor label generation in inches, centimeters, and number of labels. Two resettable counters are available and can be reset.

**Format** `~ROc`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `c` = counter number | *Accepted Values:* `1` or `2`<br>*Default Value:* a value must be specified or the command is ignored |

**Example •** This is an example of the `~RO` command.



# ^SC

## Set Serial Communications

**Description** The ^SC command allows you to change the serial communications parameters you are using.

**Format** ^SCa,b,c,d,e,f

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = baud rate | *Accepted Values:* 110; 600; 1,200; 2400; 4800; 9600; 14400; 19200; 28800; 38400; or 57600; 115200<br>*Default Value:* must be specified or the parameter is ignored |
| b = word length (in data bits) | *Accepted Values:* 7 or 8<br>*Default Value:* must be specified |
| c = parity | *Accepted Values:* N (none), E (even), or O (odd)<br>*Default Value:* must be specified |
| d = stop bits | *Accepted Values:* 1 or 2<br>*Default Value:* must be specified |
| e = protocol mode | *Accepted Values:*<br>X = XON/XOFF<br>D = DTR/DSR<br>R = RTS<br>*Default Value:* must be specified |
| f = Zebra protocol | *Accepted Values:*<br>A = ACK/NAK<br>N = none<br>Z = Zebra<br>*Default Value:* must be specified |

**Comments** If any of the parameters are missing, out of specification, not supported by a particular printer, or have a ZPL-override DIP switch set, the command is ignored.

A ^JUS command causes the changes in Communications Mode to persist through power-up and software resets.

# ~SD

## Set Darkness

**Description** The ~SD command allows you to set the darkness of printing. ~SD is the equivalent of the darkness setting parameter on the control panel display.

**Format** ~SD##

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| ## = desired darkness setting (two-digit number) | *Accepted Values:* 00 to 30<br>*Default Value:* last permanently saved value |

**Example** • These are examples of the *Xi*III*Plus* Darkness Setting:

```
^MD8.3
```

```
~SD8.3
```

**Comments** The ^MD command value, if applicable, is added to the ~SD command.

# ^SE

## Select Encoding

**Description** The ^SE command is used to select the desired ZPL or ZPL II encoding table.

**Format** ^SEd:o.x

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = location of encoding table | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* R: |
| o = name of encoding table | *Accepted Value:* 1 to 8 alphanumeric characters<br>*Default Value:* a value must be specified |
| x = extension | *Fixed Value:* .DAT |

The encoding tables are provided with the font card or downloaded in flash with the font. The table appears as XXXXXXX.DAT in a directory label printed by the ZPL commands.

The most active encoding table is indicated by the * on the directory label.

# ^SF

## Serialization Field (with a Standard ^FD String)

**Description** The ^SF command allows you to serialize a standard ^FD string. The maximum size of the mask and increment string is 3K combined.

**Format** ^SFa,b

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = mask string | The mask string sets the serialization scheme. The length of the string mask defines the number of characters in the current ^FD string to be serialized. The mask is aligned to the characters in the ^FD string starting with the right-mostposition.<br>*Mask String placeholders:*<br>D or d – Decimal numeric 0–9<br>H or h – Hexadecimal 0–9 plus a-f or A-F<br>O or o – Octal 0–7<br>A or a – Alphabetic A–Z or a–z<br>N or n – Alphanumeric 0–9 plus A–Z or a–z<br>% – Ignore character or skip |
| b = increment string | The increment string is the value to be added to the field on each label. The default value is equivalent to a decimal value of one. The string is composed of any characters defined in the serial string. Invalid characters are assumed to be equal to a value of zero in that characters position.<br>The increment value for alphabetic strings start with 'A' or 'a' as the zero placeholder. This means to increment an alphabetic character by one, a value of 'B' or 'b' must be in the increment string. |

For characters that do not get incremented, the % character needs to be added to the increment string.

**Example •** This is an example of serializing a ^FD string. The ZPL II code generates three separate labels as seen in Generated Labels:

**ZPL II CODE**

```
^XA
^FO100,100
^CF0,100
^FD12A^SFnnA,F^FS
^PQ3
^XZ
```

**GENERATED LABELS**

12K

12F

12A

This mask has the first characters as alphanumeric (nn = 12) and the last digit as uppercase alphabetic (A). The decimal value of the increment number is equivalent to 5 (F). The number of labels generated depends on the number specified by the ^PQ command.

In a similar instance, the ^FD string could be replaced with either of the ^FD strings below to generate a series of label, determined by ^PQ.

Using this ZPL code:

```
^FDBL0000^SFAAdddd,1
```

The print sequence on this series of labels is:

```
BL0000, BL0001,...BL0009, BL0010,...
BL0099, BL0100,...BL9999, BM0000...
```

Using this ZPL code:

```
^FDBL00-0^SFAAdd%d,1%1
```

The print sequence on this series of labels is:

```
BL00-0, BL01-1, BL02-2,...BL09-9,
BL11-0, BL12-1...
```

**Examples •** These examples show the importance of capitalization and location within the mask.

In this example, the printer cycles with every two printed labels and alternates between H (position 18), and then Z (position 36). With n or N, the serial number increments from 0 - 9 and a–z or A–Z (36 positions overall). With each completed cycle, the second cluster (nn) increments one position (from 00, 01, 02 …) per cycle:

**ZPL II CODE**

```
^XA
^FO100,50^A0N,50,50^FDzzZ^SFnnN,I^FS
^PQ10
^XZ
```

**GENERATED LABELS**


In this example, lower case i increments with a mask string of nnN. Nothing changes because the first cluster (Z) never triggers the second cluster (zz) to change.

**ZPL II CODE**

```
^XA
^FO100,50^A0N,50,50^FDzzZ^SFnnN,i^FS
^PQ10
^XZ
```

**GENERATED LABELS**

# ^SL

## Set Mode and Language (for Real-Time Clock)

**Description** The ^SL command is used to specify the Real-Time Clock's mode of operation and language for printing information.

**Important •**

- Time is read when the image is created. If the image stays in the queue longer than the specified time the image will be recreated with a new time.
- There are incidents when the same time or a larger space of time may be printed on labels. This is due to the format complexity and print speed.

**Format** ^SLa,b

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = mode | *Accepted Values:*<br>S = Start Time Mode. This is the time that is read from the Real-Time Clock when label formatting begins (when ^XA is received). The first label has the same time placed on it as the last label.<br>T = Time Now Mode. This is the time that is read from the Real-Time Clock when the label to be printed is placed in print queue. *Time Now* is similar to a serialized time or date field.<br>Numeric Value = With the Enhanced Real Time Clock (V60.13.0.10 and above) a time accuracy tolerance can be specified. Range = 1 to 999 seconds, 0 = one second tolerance<br>Example: SL30,1 = Accuracy tolerance of 30 seconds and use English.<br>*Default Value:* S |

| Parameters | Details |
|---|---|
| b = language | *Accepted Values:*<br>1 = English<br>2 = Spanish<br>3 = French<br>4 = German<br>5 = Italian<br>6 = Norwegian<br>7 = Portuguese<br>8 = Swedish<br>9 = Danish<br>10 = Spanish 2<br>11 = Dutch<br>12 = Finnish<br>= *Default Value:* the language selected with ^KL or the control panel |

**Comments** These are some comments to be aware of:

- The **^SL** command must be placed before the first **^FO** command.
- As of V60.13.0.10 all supported printers have Enhanced Real Time Clock capabilities the RTC will not print time fields that are more than sixty seconds old, rather it will update the time prior to printing (**^SLT** or **^SL60**). To control time with increments other than sixty seconds the **^SL** command can be used with a numeric value (**^SL30**).
  **^SLS** can keep times longer than sixty seconds.

# ^SN

## Serialization Data

**Description** The `^SN` command allows the printer to index data fields by a selected increment or decrement value, making the data fields increase or decrease by a specified value each time a label is printed. This can be performed on 100 to 150 fields in a given format and can be performed on both alphanumeric and bar code fields. A maximum of 12 of the right-most integers are subject to indexing. The first integer found when scanning from right to left starts the indexing portion of the data field.

If the alphanumeric field to be indexed ends with an alpha character, the data is scanned, character by character, from right to left until a numeric character is encountered. Serialization takes place using the value of the first number found.

**Format** `^SNv,n,z`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `v` = starting value | *Accepted Values:* 12-digits maximum for the portion to be indexed<br>*Default Value:* `1` |
| `n` = increment or decrement value | *Accepted Values:* 12-digit maximum<br>*Default Value:* `1`<br>To indicate a decrement value, precede the value with a minus (–) sign. |
| `z` = add leading zeros (if needed) | *Accepted Values:*<br>`N` = no<br>`Y` = yes<br>*Default Value:* `N` |

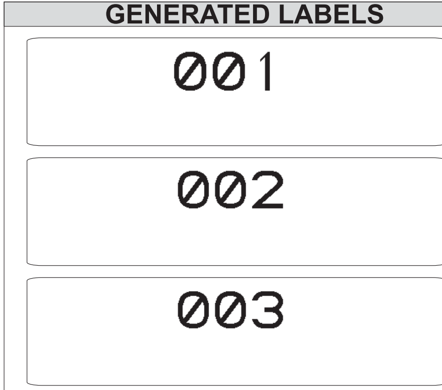
**Example** • This example shows incrementing by a specified value:

**ZPL II CODE**

```
^XA
^FO260,110
^CFG
^SN001,1,Y^FS
^PQ3
^XZ
```

***Note: The ZPL II code above will generate three separate labels, seen to the right.***

**GENERATED LABELS**



**Comments** Incrementing and decrementing takes place for each serial-numbered field when all replicates for each serial number have been printed, as specified in parameter `r` of the `^PQ` (print quality) command.

If, during the course of printing serialized labels, the printer runs out of either paper or ribbon, the first label printed (after the media or ribbon has been replaced and calibration completed) has the same serial number as the *partial* label printed before the *out* condition occurred. This is done in case the last label before the *out* condition did not fully print. This is controlled by the `^JZ` command.

## Using Leading Zeros

In the `^SN` command, the `z` parameter determines if leading zeros are printed or suppressed. Depending on which value is used (`Y =` print leading zeros; `N =` do not print leading zeros), the printer either prints or suppresses the leading zeros.

The default value for this parameter is `N` (do not print leading zeros).

## Print Leading Zeros

The starting value consists of the right-most consecutive sequence of digits. The width (number of digits in the sequence) is determined by scanning from right to left until the first non-digit (space or alpha character) is encountered. To create a specific width, manually place leading zeros as necessary.

## Suppressing Leading Zeros

The starting value consists of the right-most consecutive sequence of digits, including any leading spaces. The width (number of digits in the sequence) is determined by scanning from right to left until the first alpha character (except a space) is encountered. To create a specific width, manually place leading spaces or zeros as necessary. Suppressed zeros are replaced by spaces. During the serialization process, when the entire number contains all zeros, the last zero is not suppressed.

The `^SN` command replaces the Field Data (`^FD`) command within a label formatting program.

# ^SO

## Set Offset (for Real-Time Clock)

**Description** The ^SO command is used to set the secondary and the tertiary offset from the primary Real-Time Clock.

**Note** • For each label only one SO2 command can be used. If more than one offset is required, SO3 must be used.

**Format** ^SOa,b,c,d,e,f,g

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = clock set | *Accepted Values:*<br>2 = secondary<br>3 = third<br>*Default Value:* value must be specified |
| b = months offset | *Accepted Values:* −32000 to 32000<br>*Default Value:* 0 |
| c = days offset | *Accepted Values:* −32000 to 32000<br>*Default Value:* 0 |
| d = years offset | *Accepted Values:* −32000 to 32000<br>*Default Value:* 0 |
| e = hours offset | *Accepted Values:* −32000 to 32000<br>*Default Value:* 0 |
| f = minutes offset | *Accepted Values:* −32000 to 32000<br>*Default Value:* 0 |
| g = seconds offset | *Accepted Values:* −32000 to 32000<br>*Default Value:* 0 |

# ^SP

## Start Print

**Description** The ^SP command allows a label to start printing at a specified point before the entire label has been completely formatted. On extremely complex labels, this command can increase the overall throughput of the print.

The command works as follows: Specify the dot row at which the ^SP command is to begin. This creates a label *segment*. Once the ^SP command is processed, all information in that segment prints. During the printing process, all of the commands after the ^SP continue to be received and processed by the printer.

If the segment after the ^SP command (or the remainder of the label) is ready for printing, media motion does not stop. If the next segment is not ready, the printer stops mid-label and wait for the next segment to be completed. Precise positioning of the ^SP command requires a trial-and-error process, as it depends primarily on print speed and label complexity.

The ^SP command can be effectively used to determine the worst possible print quality. You can determine whether using the ^SP command is appropriate for the particular application by using this procedure.

If you send the label format up to the first ^SP command and then wait for printing to stop before sending the next segment, the printed label is a sample of the worst possible print quality. It drops any field that is out of order.

If the procedure above is used, the end of the label format must be:

```
^SP#^FS
```

**Format** `^SPa`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = dot row to start printing | *Accepted Values:* 0 to 32000<br>*Default Value:* 0 |

**Example •** In this example, a label 800 dot rows in length uses `^SP500`. Segment 1 prints while commands in Segment 2 are being received and formatted.

# ^SQ

## Halt ZebraNet Alert

**Description** The ^SQ command is used to stop the ZebraNet Alert option.

**Format** ^SQa,b,c

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = condition type | *Accepted Values:*<br>A = paper out<br>B = ribbon out<br>C = printhead over-temp<br>D = printhead under-temp<br>E = head open<br>F = power supply over-temp<br>G = ribbon-in warning (Direct Thermal Mode)<br>H = rewind full<br>I = cut error<br>J = printer paused<br>K = PQ job completed<br>L = label ready<br>M = head element out<br>N = reserved<br>O = reserved<br>P = power on<br>Q = clean printhead<br>R = media low<br>S = ribbon low<br>T = replace head<br>U = battery low<br>V = RFID error (in RFID printers only)<br>W = all errors (in RFID printers only)<br>* = all errors (in non-RFID printers) |

| Parameters | Details |
|---|---|
| b = destination | *Accepted Values:*<br>A = serial port<br>B = parallel port<br>C = e-mail address<br>D = TCP/IP<br>E = UDP/IP<br>F = SNMP trap<br>* = wild card to stop alerts for all destinations |
| c = halt messages | *Accepted Values:*<br>Y = halt messages<br>N = start messages<br>*Default Value:* Y |

# ^SR

## Set Printhead Resistance

**Description** The ^SR command allows you to set the printhead resistance.

**Format** ^SR####

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| #### = resistance value (four-digit numeric value) | *Accepted Value:* 0488 to 1175<br>*Default Value:* last permanently saved value |

**Comments** To avoid damaging the printhead, this value should be less than or equal to the value shown on the printhead being used. Setting a higher value could damage the printhead.

**Note •** New printer models automatically set head resistance.

# ^SS

## Set Media Sensors

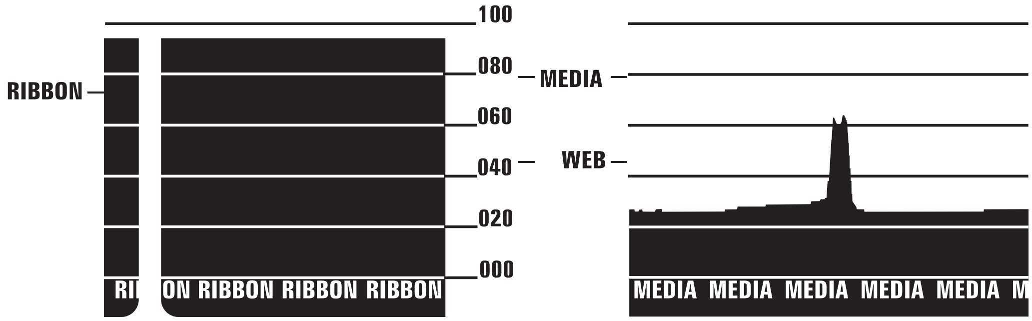
**Description** The ^SS command is used to change the values for media, web, ribbon, and label length set during the media calibration process. The media calibration process is described in your specific printer's user's guide.

**Format** `^SSw,m,r,l,m2,r2,a,b,c`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `w` = web (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value shown on the media sensor profile or configuration label |
| `m` = media (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value shown on the media sensor profile or configuration label |
| `r` = ribbon (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value shown on the media sensor profile or configuration label |
| `l` = label length (in dots, four-digit value) | *Accepted Values:* `0001` to `32000`<br>*Default Value:* value calculated in the calibration process |
| `m2` = intensity of media LED (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value calculated in the calibration process |
| `r2` = intensity of ribbon LED (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value calculated in the calibration process |
| `a` = mark sensing (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value calculated in the calibration process |
| `b` = mark media sensing (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value calculated in the calibration process |
| `c` = mark LED sensing (three-digit value) | *Accepted Values:* `000` to `100`<br>*Default Value:* value calculated in the calibration process |

**Example •** Below is an example of a media sensor profile. Notice the numbers from 000 to 100 and where the words WEB, MEDIA, and RIBBON appear in relation to those numbers. Also notice the black vertical spike. This represents where the printer sensed the transition from media-to-web-to-media.




The media and sensor profiles produced vary in appearance from printer to printer.

**Comments** The `m2` and `r2` parameters have no effect in Stripe® *S*-300 and *S*-500 printers.

Maximum values for parameters depend on which printer platform is being used.

# ^ST

## Set Date and Time (for Real-Time Clock)

**Description** The ^ST command sets the date and time of the Real-Time Clock.

**Format** ^STa,b,c,d,e,f,g

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| a = month | *Accepted Values:* 01 to 12<br>*Default Value:* current month |
| b = day | *Accepted Values:* 01 to 31<br>*Default Value:* current day |
| c = year | *Accepted Values:* 1998 to 2097<br>*Default Value:* current year |
| d = hour | *Accepted Values:* 00 to 23<br>*Default Value:* current hour |
| e = minute | *Accepted Values:* 00 to 59<br>*Default Value:* current minute |
| f = second | *Accepted Values:* 00 to 59<br>*Default Value:* current second |
| g = format | *Accepted Values:*<br>A = a.m.<br>P = p.m.<br>M = 24-hour military<br>*Default Value:* M |

# ^SX

## Set ZebraNet Alert

**Description** The `^SX` command is used to configure the ZebraNet Alert System.

**Format** `^SXa,b,c,d,e,f`

This table identifies the parameters for this format:

**Note •** The values in this table apply to firmware V48_12_4 and above.

| Parameters | Details |
|---|---|
| `a` = condition type | *Accepted Values:*<br>`A` = paper out<br>`B` = ribbon out<br>`C` = printhead over-temp<br>`D` = printhead under-temp<br>`E` = head open<br>`F` = power supply over-temp<br>`G` = ribbon-in warning (Direct Thermal Mode)<br>`H` = rewind full<br>`I` = cut error<br>`J` = printer paused<br>`K` = PQ job completed<br>`L` = label ready<br>`M` = head element out<br>`N` = reserved<br>`O` = reserved<br>`P` = power on<br>`Q` = clean printhead<br>`R` = media low<br>`S` = ribbon low<br>`T` = replace head<br>`U` = battery low<br>`V` = RFID error (in RFID printers only)<br>`W` = all errors (in RFID printers only)<br>`*` = all errors (in non-RFID printers)<br>*Default Value:* if the parameter is missing or invalid, the command is ignored |

| Parameters | Details |
|---|---|
| b = destination for route alert | *Accepted Values:*<br>A = serial port<br>B* = parallel port<br>C = e-mail address<br>D = TCP/IP<br>E = UDP/IP<br>F = SNMP trap<br>*Default Value:* if this parameter is missing or invalid, the command is ignored<br>* **Requires** bidirectional communication. |
| c = enable *condition set* alert to this destination | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* Y or previously configured value |
| d = enable condition clear alert to this destination | *Accepted Values:*<br>N = no<br>Y = yes<br>*Default Value:* N or previously configured value<br>Parameters e and f are sub-options based on destination. If the sub-options are missing or invalid, these parameters are ignored. |
| e = destination setting | *Accepted Values:*<br>Internet e-mail address (e.g. user@company.com)<br>IP address (for example, 10.1.2.123)<br>SNMP trap<br>IP or IPX addresses |
| f = port number | *Accepted Values:*<br>TCP port # (0 to 65535)<br>UPD port # (0 to 65535) |

**Example** • This is an example of the different (b) destinations that you can send for the condition type (a):

Serial: ^SXA,A,Y,Y
Parallel: ^SXA,B,Y,Y
E-Mail: ^SXA,C,Y,Y,admin@company.com
TCP: ^SXA,D,Y,Y,123.45.67.89,1234
UDP: ^SXA,E,Y,Y,123.45.67.89,1234
SNMP Trap: ^SXA,F,Y,Y,255.255.255.255

**Comments** In the example above for SNMP Trap, entering 255.255.255.255 broadcasts the notification to every SNMP manager on the network. To route the device to a single SNMP manager, enter a specific address (123.45.67.89).

# ^SZ

## Set ZPL

**Description** The `^SZ` command is used to select the programming language used by the printer. This command gives you the ability to print labels formatted in both ZPL and ZPL II.

This command remains active until another `^SZ` command is sent to the printer or the printer is turned off.

**Format** `^SZa`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = ZPL version | *Accepted Values:*<br>`1` = ZPL<br>`2` = ZPL II<br>*Default Value:* 2 |

**Comments** If the parameter is missing or invalid, the command is ignored.

# ~TA

## Tear-off Adjust Position

**Description** The `~TA` command lets you adjust the rest position of the media after a label is printed, which changes the position at which the label is torn or cut.

**Format** `~TA###`

**Important •** These are some important facts about this command:

- For 600 dpi printers, the step size doubles.
- If the number of characters is **less than** 3, the command is ignored.

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `###` = change in media rest position (3-digit value in dot rows must be used.) | *Accepted Values:* `–120` to `120`<br>*Default Value:* last permanent value saved |

**Comments** If the parameter is missing or invalid, the command is ignored.

# ^TO

## Transfer Object

**Description** The ^TO command is used to copy an object or group of objects from one storage device to another. It is similar to the copy function used in PCs.

Source and destination devices must be supplied and must be different and valid for the action specified. Invalid parameters cause the command to be ignored.

The asterisk (*) can be used as a wild card for object names and extensions. For instance, ZEBRA.* or *.GRF are acceptable forms for use with the ^TO command.

At least one source parameter (d, o, or x) and one destination parameter (s, o, or x) must be specified. If only ^TO is entered, the command is ignored.

**Format** ^TOs:o.x,d:o.x

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| s = source device of stored object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* if a drive is not specified, all objects are transferred to the drive set in parameter s |
| o = stored object name | *Accepted Values:* any existing object conforming to Zebra conventions<br>*Default Value:* if a name is not specified, * is used — all objects are selected |
| x = extension | *Accepted Values:* any extension conforming to Zebra conventions<br>*Default Value:* if an extension is not specified, * is used — all extensions are selected |
| d = destination device of the stored object | *Accepted Values:* R:, E:, B:, and A:<br>*Default Value:* a destination must be specified |
| o = name of the object at destination | *Accepted Values:* up to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, the name of the existing object is used |
| x = extension | *Accepted Values:* any extension conforming to Zebra conventions<br>*Default Value:* if an extension is not specified, the extension of the existing object is used |

**Comments** Parameters o, x, and s support the use of the wild card (*).

If the destination device does not have enough free space to store the object being copied, the command is canceled.

Zebra files (Z:*.*) cannot be transferred. These files are copyrighted by Zebra Technologies.

### Transferring Objects

These are some examples of using the ^TO command.

**Example •** To copy the object **ZLOGO.GRF** from DRAM to an optional Memory Card and rename it **ZLOGO1.GRF**, write the following format:

```
^XA
^TOR:ZLOGO.GRF,B:ZLOGO1.GRF
^XZ
```

**Example •** To copy the object **SAMPLE.GRF** from an optional Memory Card to DRAM and keep the same name, write this format:

```
^XA
^TOB:SAMPLE.GRF,R:SAMPLE.GRF
^XZ
```

### Transferring Multiple Objects

The asterisk (*) can be used to transfer multiple object files (except *.FNT) from DRAM to the Memory Card. For example, assume you have several object files that contain logos. These files are named LOGO1.GRF, LOGO2.GRF, and LOGO3.GRF.

To transfer all these files to the memory card using the name NEW instead of LOGO, place an asterisk after the names NEW and LOGO in the transfer command. This copies all files beginning with LOGO in one command.

```
^XA
^TOR:LOGO*.GRF,B:NEW*.GRF
^XZ
```

During a multiple transfer, if a file is too big to be stored on the memory card, that file is skipped. All remaining files attempt to be transferred. All files that can be stored within the space limitations are transferred, while other files are ignored.

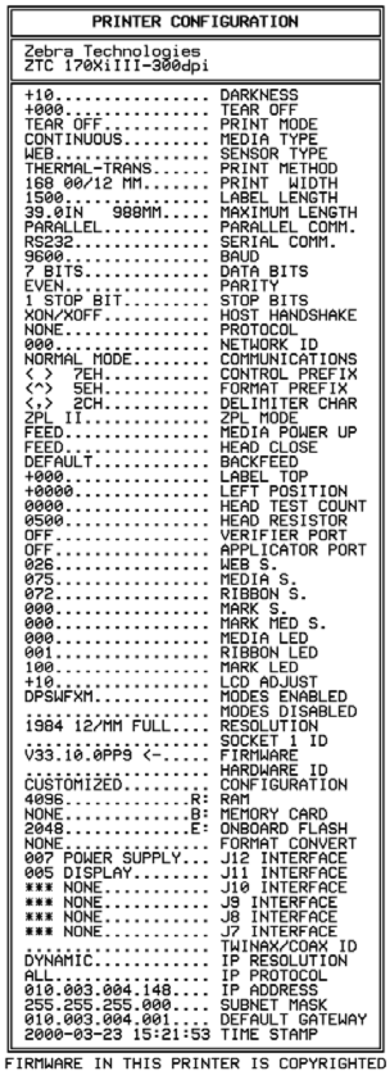
# ~WC

## Print Configuration Label

**Description** The `~WC` command is used to generate a printer configuration label. The printer configuration label contains information about the printer setup, such as sensor type, network ID, ZPL mode, firmware version, and descriptive data on the `R:`, `E:`, `B:`, and `A:` devices.

**Format** `~WC`

**Comments** This command works only when the printer is idle.



```
PRINTER CONFIGURATION

Zebra Technologies
ZTC 170XiIII-300dpi

+10................. DARKNESS
+000................ TEAR OFF
TEAR OFF............ PRINT MODE
CONTINUOUS.......... MEDIA TYPE
WEB................. SENSOR TYPE
THERMAL-TRANS....... PRINT METHOD
168 00/12 MM........ PRINT  WIDTH
1500................ LABEL LENGTH
39.0IN   988MM...... MAXIMUM LENGTH
PARALLEL............ PARALLEL COMM.
RS232............... SERIAL COMM.
9600................ BAUD
7 BITS.............. DATA BITS
EVEN................ PARITY
1 STOP BIT.......... STOP BITS
XON/XOFF............ HOST HANDSHAKE
NONE................ PROTOCOL
000................. NETWORK ID
NORMAL MODE......... COMMUNICATIONS
< >  7EH............ CONTROL PREFIX
<^>  5EH............ FORMAT PREFIX
<,>  2CH............ DELIMITER CHAR
ZPL II.............. ZPL MODE
FEED................ MEDIA POWER UP
FEED................ HEAD CLOSE
DEFAULT............. BACKFEED
+000................ LABEL TOP
+0000............... LEFT POSITION
0000................ HEAD TEST COUNT
0500................ HEAD RESISTOR
OFF................. VERIFIER PORT
OFF................. APPLICATOR PORT
026................. WEB S.
075................. MEDIA S.
072................. RIBBON S.
000................. MARK S.
000................. MARK MED S.
000................. MEDIA LED
001................. RIBBON LED
100................. MARK LED
+10................. LCD ADJUST
DPSWFXM............. MODES ENABLED
.................... MODES DISABLED
1984 12/MM FULL..... RESOLUTION
.................... SOCKET 1 ID
V33.10.0PP9 <-...... FIRMWARE
.................... HARDWARE ID
CUSTOMIZED.......... CONFIGURATION
4096..............R: RAM
NONE..............B: MEMORY CARD
2048..............E: ONBOARD FLASH
NONE................ FORMAT CONVERT
007 POWER SUPPLY.... J12 INTERFACE
005 DISPLAY......... J11 INTERFACE
*** NONE............ J10 INTERFACE
*** NONE............ J9 INTERFACE
*** NONE............ J8 INTERFACE
*** NONE............ J7 INTERFACE
.................... TWINAX/COAX ID
DYNAMIC............. IP RESOLUTION
ALL................. IP PROTOCOL
010.003.004.148..... IP ADDRESS
255.255.255.000..... SUBNET MASK
010.003.004.001..... DEFAULT GATEWAY
2000-03-23 15:21:53 TIME STAMP
FIRMWARE IN THIS PRINTER IS COPYRIGHTED
```

# ^WD

## Print Directory Label

**Description** The ^WD command is used to print a label listing bar codes, objects stored in DRAM, or fonts.

For bar codes, the list shows the name of the bar code. For fonts, the list shows the name of the font, the number to use with ^A command, and size. For objects stored in DRAM, the list shows the name of the object, extension, size, and option flags. All lists are enclosed in a double-line box.

**Format** `~WDd:o.x`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| d = source device — optional | *Accepted Values:* R:, E:, B:, A: and Z:<br>*Default Value:* R: |
| o = object name — optional | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* *<br>The use of a ? (question mark) is also allowed. |
| x = extension — optional | *Accepted Values:* any extension conforming to Zebra conventions<br>.FNT = font<br>.BAR = bar code<br>.ZPL = stored ZPL format<br>.GRF = GRF graphic<br>.CO = memory cache<br>.DAT = font encoding<br>.STO = data storage<br>.PNG = PNG graphic<br>* = all objects<br>*Default Value:* *<br>The use of a ? (question mark) is also allowed. |

**Example** • To print a label listing all objects in DRAM, enter:

```
^XA
^WDR:*.*
^XZ
```

**Example** • To print a label listing all resident bar codes, enter:

```
^XA
^WDZ:*.BAR
^XZ
```

**Example** • To print a label listing all resident fonts, enter:

```
^XA
^WDZ:*.FNT
^XZ
```

# ^XA

## Start Format

**Description** The `^XA` command is used at the beginning of ZPL II code. It is the opening bracket and indicates the start of a new label format. This command is substituted with a single ASCII control character STX (control-B, hexadecimal 02).

**Format** `^XA`

**Comments** Valid ZPL II format requires that label formats should start with the `^XA` command and end with the `^XZ` command.

# ^XB

## Suppress Backfeed

**Description** The ^XB command suppresses forward feed of media to tear-off position depending on the current printer mode. Because no forward feed occurs, a backfeed before printing of the next label is not necessary; this improves throughput. When printing a batch of labels, the last label should not contain this command.

**Format** ^XB

### ^XB in the Tear-off Mode

*Normal Operation:* backfeed, print, and feed to rest

^XB *Operation:* print (Rewind Mode)

### ^XB in Peel-off Mode

*Normal Operation:* backfeed, print, and feed to rest

^XB *Operation:* print (Rewind Mode)

**Note •** To prevent jamming in cutter mode, ^XB suppresses backfeed and cutting.

# ^XF

## Recall Format

**Description** The ^XF command recalls a stored format to be merged with variable data. There can be multiple ^XF commands in one format, and they can be located anywhere within the code.

When recalling a stored format and merging data using the ^FN (Field Number) function, the calling format must contain the ^FN command to merge the data properly.

While using stored formats reduces transmission time, no formatting time is saved. The ZPL II format being recalled is saved as text strings that need to be formatted at print time.

**Format** `^XFd:o.x`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = source device of stored image | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* search priority (`R:`, `E:`, `B:`, and `A:`) |
| `o` = name of stored image | *Accepted Values:* 1 to 8 alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `x` = extension l | *Fixed Value:* `.ZPL` |

For a complete example of the ^DF and ^XF command, see *^DF and ^XF — Download format and recall format* on page 11.

# ^XG

## Recall Graphic

**Description** The ^XG command is used to recall one or more graphic images for printing. This command is used in a label format to merge graphics, such as company logos and piece parts, with text data to form a complete label.

An image can be recalled and resized as many times as needed in each format. Other images and data might be added to the format.

**Format** `^XGd:o.x,mx,my`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `d` = source device of stored image | *Accepted Values:* `R:`, `E:`, `B:`, and `A:`<br>*Default Value:* search priority (`R:`, `E:`, `B:`, and `A:`) |
| `o` = name of stored image | *Accepted Values:* `1` to `8` alphanumeric characters<br>*Default Value:* if a name is not specified, UNKNOWN is used |
| `x` = extension l | *Fixed Value:* `.GRF` |
| `mx` = magnification factor on the x-axis | *Accepted Values:* `1` to `10`<br>*Default Value:* `1` |
| `my` = magnification factor on the y-axis | *Accepted Values:* `1` to `10`<br>*Default Value:* `1` |

**Example** • This is an example of using the ^XG command to recall the image SAMPLE.GRF from DRAM and print it in five different sizes in five different locations on the same label:

```
^XA
^FO100,100^XGR:SAMPLE.GRF,1,1^FS
^FO100,200^XGR:SAMPLE.GRF,2,2^FS
^FO100,300^XGR:SAMPLE.GRF,3,3^FS
^FO100,400^XGR:SAMPLE.GRF,4,4^FS
^FO100,500^XGR:SAMPLE.GRF,5,5^FS
^XZ
```

# ^XZ

## End Format

**Description** The ^XZ command is the ending (closing) bracket. It indicates the end of a label format. When this command is received, a label prints. This command can also be issued as a single ASCII control character ETX (Control-C, hexadecimal 03).

**Format** ^XZ

**Comments** Label formats must start with the ^XA command and end with the ^XZ command to be in valid ZPL II format.

# ^ZZ

## Printer Sleep

**Description** The `^ZZ` command places the printer in an idle or shutdown mode.

**Format** `^ZZt,b`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `t` = number of second (idle time) prior to shutdown | *Accepted Values:* `0` to `999999` – setting `0` disables automatic shutdown<br>*Default Value:* last permanently saved value or `0` |
| `b` = label status at shutdown | *Accepted Values:*<br>`Y` = indicates to shutdown when labels are still queued<br>`N` = indicates all labels must be printed before shutting down<br>*Default Value:* `N` |

**Comments** The `^ZZ` command is only valid on the PA400 and PT400 battery-powered printers.

# 3

# RFID Commands

This section contains the ZPL II commands for RFID-specific applications.

For more information about the RFID commands, refer to the *RFID Programming Guide*. A copy is available on the User CD provided with your printer and online at http://www.zebra.com/manuals.

# RFID Command Overview

In addition to reading or encoding RFID tags, the RFID ZPL commands also provide for RFID exception handling, such as setting the number of read/write retries before declaring a transponder defective (set with `^RR`, `^RT`, and `^WT`) or setting the number of labels that will be attempted if an error occurs (set with `^RS`).

For example, if an RFID label fails to program correctly or if the transponder cannot be detected, the printer ejects the label and prints VOID across it. The printer will try to print another label with the same data and format for the number of RFID labels specified by the `^RS` command. If the problem persists, the printer follows the error handling instructions specified by the `^RS` command: the printer may remove the problematic format from the print queue and proceed with the next format (if one exists in the buffer), or it may place the printer in Pause or Error mode.

**Important •** Consider the following before using any command in this section:

- Before using a particular command, verify that it is compatible with your printer and firmware version. See Table 13 on page 287.
- If a parameter in the following tables is designated as *not applicable* for a particular printer, any value entered for the parameter will be ignored, but the place holder for the field is required.

# Printer and Firmware Compatibility

Table 13 shows which RFID ZPL commands you can use with different printers and firmware versions.

### Table 13 • Supported Commands Based on Printer and Firmware

| Command | Function | UHF Printers | | | | | | HF Printers | |
|---|---|---|---|---|---|---|---|---|---|
| | | R60.13.X | R62.13.X | R63.13.X | SP920X | SP994X, SP999X, SP1027X | SP1056X | | |
| | | R110*Xi* R170*Xi* | R110*PAX4* | | R4Mplus | | | R110*Xi* HF | R2844-Z |
| *^HL or ~HL* on page 288 | Return RFID Data Log to Host | * | * | * | — | * | * | * | — |
| *^HR* on page 289 | Calibrate RFID Transponder Position | * | * | * | — | * | * | * | — |
| *^RA* on page 292 | Read AFI or DSFID Byte | — | — | — | — | — | — | * | — |
| *^RB* on page 294 | Define EPC Data Structure | * | * | * | — | * | * | * | — |
| *^RE* on page 296 | Enable/Disable E.A.S. Bit | — | — | — | — | — | — | * | * |
| *^RF* on page 297 | Read or Write RFID Format | * | * | * | — | * | * | * | — |
| *^RI* on page 299 | Get RFID Tag ID | *a | — | — | — | * | — | * | * |
| *^RM* on page 300 | Enable RFID Motion | * | * | * | — | * | * | * | — |
| *^RN* on page 301 | Detect Multiple RFID Tags in Encoding Field | * b | * | * | — | — | — | * | — |
| *~RO* on page 302 | Reset Advanced Counters | * | * | * | — | * | * | * | — |
| *^RR* on page 304 | Specify RFID Retries for a Block | * | * | * | — | * | * | * | — |
| *^RS* on page 305 | Set Up RFID Parameters | * | * | * | * | * | * | * | * |
| *^RT* on page 311 | Read RFID Tag | * c | * c | * c | * | * c | * c | * c | * |
| *~RV* on page 313 | Report RFID Encoding Results | * | * | * | — | — | — | — | — |
| *^RW* on page 314 | Set RFID Read and Write Power Levels | * | * | * | — | * | * | * | — |
| *^RZ* on page 315 | Set RFID Tag Password and Lock Tag | * | * | * | — | * | * | — | — |
| on page 316 | Encode AFI or DSFID Byte | — | — | — | — | — | — | * | — |
| *^WT* on page 319 | Write (Encode) Tag | * d | * d | * d | * | * d | * d | * d | * |
| *^WV* on page 321 | Verify RFID Encoding Operation | * | * | * | — | * | * | — | — |

a. Requires R60.13.0.13ZD or higher.

b. Requires R60.13.0.3 or higher.

c. Use the `^RF`, `^RM`, and `^RR` commands rather than the `^RT` command. The `^RT` command is provided only for backward-compatibility with label formats that were developed for older Zebra RFID printers.

d. Use the `^RF`, `^RM`, `^RR`, and `^WV` commands rather than the `^WT` command. The `^WT` command is provided only for backward-compatibility with label formats that were developed for older Zebra RFID printers.

# ^HL or ~HL

## Return RFID Data Log to Host

**Description** The printer continually logs RFID data and stores it in the printer's RAM. Use this command to request that the RFID data log be returned to the host computer, to clear the current data log, and to restart data recording. The data returned show the status of the RFID read, write, and lock commands and show any resulting error codes.

**Format** `^HL` or `~HL`

### Comments

- Data is shown in the format sent by the `^RFW` command (ASCII, Hex, or EPC).
- In the log, the data displays in this manner:

  `C,EEEE,DDDDDDDDDDDDDDDDDDDDDDDD`

  where

  `C` = the RFID operation (R = read, W = write, L = lock)

  `EEEE` = the RFID error code

  `DDDDDDDDDDDDDDDDDDDDDDDD` = data read or written
- If the log exceeds 64K (approximately 2000 operations), the data log is cleared automatically, and data recording restarts. When this happens, the following appears in the log:

  `Logfile automatically reset`
- If the printer loses power, the log is lost. If the log results are important to you, retrieve the log frequently.

# ^HR

## Calibrate RFID Transponder Position

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to initiate an RFID transponder calibration for a specific RFID label. Results are returned to the host computer. This calibration is used to determine the optimal programming position for RFID media that may not meet the transponder placement specifications for the printer.

Do not perform transponder calibration for RFID media that meets the transponder placement specifications for your printer. Doing so will slow the printer's throughput unnecessarily. To order media that is designed for use with your RFID printer, contact your authorized Zebra reseller.

During transponder calibration, the printer feeds the RFID label one-dot row at a time while taking readings (via the READ TAG command and the WRITE TAG commands) to profile the RFID transponder. Based on the results, the printer determines the optimal programming position for the label and returns a results table to the host. The calibrated value is used as the programming position for the `^RS` command, can be overwritten by the `^RS` command, and is saved to nonvolatile memory (the value is saved even if the power is turned off).

This calibration takes into account the print mode, backfeed mode, and tear off position. The `RUN` option in the `RFID TAG CALIB` control panel parameter performs the same calibration but does not create a results table.

**Important •** If a label format specifies a value for parameter `p` (read/write position of the transponder) in the `^RS` command, that value will be used for the programming position for all RFID labels until a new position is specified or until the printer is turned Off (**O**) and then back On (**I**).

**Format** `^HRa,b`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = start string | User text to appear before the results table.<br>*Accepted values:* any string less than 65 characters<br>*Default value:* `start` |
| `b` = end string | User text to appear after the results table.<br>*Accepted values:* any string less than 65 characters<br>*Default value:* `end` |

### Comments

- Based on the recommended transponder placement position for most RFID labels, the printer's default RFID programming position is zero for the R110*PAX*4. For other RFID printers, the default programming position is the label length minus 1 mm (0.04 in.). To return to the default programming position at any time, use the RESTORE option in the RFID TAG CALIB control panel parameter.
- At the end of calibration, a results table is returned to the host. Each line in the results table appears as:

  Row, Read Result, Write Result

  where

  Row = the dot row where calibration occurred

  Read Result = results of calibration (R = read, “ ” = unable to read)

  Write Result = results of calibration (W = write, “ ” = unable to write)

**Example** • If the following command is sent to the printer:

```
^XA^HR^XZ
```

The printer starts the transponder calibration and returns a results table such as the following:

```
start
position=195
215, ,
214, ,
213, ,
212, ,
211, ,
210, ,W
209,R,
208, ,
207, ,
206, ,W
205,R,
204, ,
203, ,
202, ,W
201,R,W
200,R,W
199,R,W
198,R,W
197,R,W
196,R,W
195,R,W <---****
194,R,W
193,R,W
192,R,W
191,R,W
190,R,W
189,R,
188, ,
187, ,
186, ,
185, ,
.
.
.
end
```

In this example, the optimal programming position is 195. This is identified at the top of the table (`position=195`) and with an the arrow (`<---****`) in the table.

# ^RA

## Read AFI or DSFID Byte

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to read the AFI or DSFID byte. The data can be returned to the host via the `^HV` command.

**Format** `^RA#,f,r,m,b`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `#` = field number specified with another command | The value assigned to this parameter should be the same as the one used in the `^RT` command.<br>*Accepted values:* 0 to 9999<br>*Default value:* 0 |
| `f` = format | *Accepted values:*<br>`0` = ASCII<br>`1` = Hexadecimal<br>*Default value:* 0 |
| `r` = number of retries | *Accepted values:* 0 to 10<br>*Default value:* 0 |
| `m` = motion | *Accepted values:*<br>`0` = Feed label after writing.<br>`1` = No Feed after writing. Other ZPL may cause a feed.<br>*Default value:* 0 |
| b = type of byte to read | *Accepted values:*<br>`A` = AFI byte<br>`D` = DSFID byte<br>*Default value:* A |

**Example 1 •** This example reads the AFI byte in ASCII format and returns `AFI Byte:x` to the host. The printer will retry the command five times if necessary. A voided label is generated if the read is unsuccessful after these retries. The data read will go into the `^FN1` location of the recalled format.

```
^XA
^FO20,120^A0N,60^FN1^FS
^RA1,0,5,0^FS
^HV1,,AFI Byte:^FS
^XZ
```

**Example 2** • This example reads the DSFID byte in ASCII format and returns `DSFID Byte:x` to the host. The printer will retry the command three times if necessary. A voided label is generated if the read is unsuccessful after these retries. The data read will go into the `^FN1` location of the recalled format.

```
^XA
^FO20,120^A0N,60^FN1^FS
^RA1,0,3,0,D^FS
^HV1,,DSFID Byte:^FS
^XZ
```

# ^RB

## Define EPC Data Structure

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to define the structure of EPC data, which can be read from or written to an RFID transponder. For more information about EPC specifications, refer to the EPC Global web site. All parameters in this command are persistent and will be used in subsequent formats if not provided. The values are initially set to the default values.

RFID transponders can have different partitions defined. This command specifies the number of partitions and how many bits are in each partition.

**Format** `^RBn,p0,p1,p2, ..., p15`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `n` = total bit size of the partitions | Specify the number of bits to include in the partitions.<br>*Accepted values:* 1 to *n*, where *n* is the bit size of the tag.<br>*Default value:* `96` |
| `p1 ... p15` = partition sizes | Specify the number of bits to include in the individual partitions. The partition sizes must add up to the bit size specified for the previous parameter. The largest individual partition size is 64 bits.<br>*Accepted values:* 1 to 64<br>*Default value:* `1` |

**Example 1 •** The following command specifies that there are 96 bits used with three fields. Fields 1, 2, and 3 contain 10, 26, and 60 bits, respectively.

```
^RB96,10,26,60
```

The ZPL code to encode a tag with this format would look like this:

```
^RFW,E^FD1000.67108000.1122921504606846976^FS
```

When the tag is being encoded, the tag stores the data in the following way:

- Field 1 contains `1000`. This value is stored in the first 10 bits
- Field 2 contains `67108000`. This value is stored in the next 26 bits.
- Field 3 contains `1122921504606846976`. This value is stored in the remaining 60 bits.

**Example 2 •** The following command specifies that there are 64 bits used with eight 8-bit fields.

```
^RB64,8,8,8,8,8,8,8,8^FS
```

The ZPL code to encode a tag with this format would look like this:

```
^RFW,E^FD1.123.160.200.249.6.1.0^FS
```

When writing to the tag, each set of data is written in its respective 8-bit field.

**Example 3 •** This example uses the SGTIN-64 standard, which defines 64-bit structure in the following way:

| | Header | Filter Value | Company Prefix Index | Item Reference | Serial Number |
|---|---|---|---|---|---|
| SGTIN-64 | 2 bits | 3 bits | 14 bits | 20 bits | 25 bits |
| | 10 (binary value) | 8 (decimal capacity) | 16,383 (decimal capacity) | 9 to 1,048,575 (decimal capacity*) | 33,554,431 (decimal capacity) |

* Capacity of Item Reference field varies with the length of the company prefix.

The ZPL code to encode a tag with this format would look like this:

```
^XA
^RB64,2,3,14,20,25
^RFW,E^FD0,3,12345,544332,22335221^FS
^XZ
```

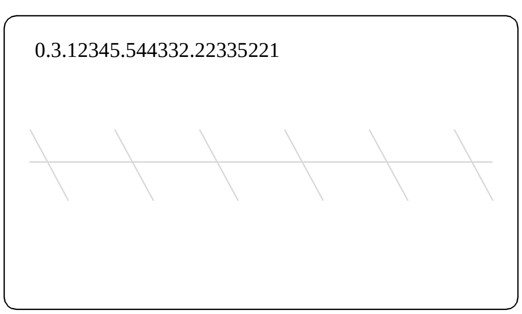
These commands would put

- `0` in the header
- `3` as the filter value
- `12345` as the company prefix
- `544332` as the item reference
- `22335221` as the serial number

To read this EPC data and print the results on the label, you would use the following code:

```
^XA
^RB64,2,3,14,20,25
^FO50,50^A0N,40^FN0^FS
^FN0^RFR,E^FS
^XZ
```

The resulting label would look like this:



# ^RE

## Enable/Disable E.A.S. Bit

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to enable or disable the Electronic Article Surveillance (E.A.S.) bit that is available in some ISO15693 tags (such as Philips). This command works only on those ISO15693 transponders and will be ignored if the tag does not support E.A.S.

**Format** `^REt,r`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `t` = Enable/disable the E.A.S. bit in the ISO15693 transponder | *Accepted values:*<br>`N` = Disable E.A.S.<br>`Y` = Enable E.A.S.<br>*Default value:* N |
| `r` = number of retries | *Accepted values:* 0 to 10<br>*Default value:* 0 |

**Example •** This example enables the E.A.S. bit in the transponder. It will retry the command five times if necessary.

```
^XA
^REy,5
^XZ
```

# ^RF

## Read or Write RFID Format

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to read or write to (encode) an RFID tag. When using this command to read a tag, you may use a field variable to print the tag data on the label or to return the data to the host.

**Format** `^RFo,f,b,n`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `o` = operation | The action to be performed.<br>*Accepted values:*<br>`W` = write to (encode) the tag<br>`L` = write with LOCK (if supported by tag type; Gen 2 does not use this locking function)<br>`R` = read the tag<br>`P` = read password (Gen 2 only)<br>*Default value:* W |
| `f` = format | *Accepted values:*<br>`A` = ASCII<br>`H` = Hexadecimal<br>`E` = EPC (ensure proper setup with the `^RB` command)<br>*Default value:* `H` |
| `b` = starting block number | **Gen 2 tags**<br>Which password to read when `P` is specified for the operation.<br>*Accepted values:*<br>`K` = kill password<br>`A` = access password<br>**Other Tag Types**<br>*Accepted values:* 0 to *n*, where *n* is the maximum number of blocks for the tag.<br>*Default value:* `0` |
| `n` = number of bytes to read or write | **Note •** This parameter is used only by the R4Mplus with firmware version SP994*X* (R4Mplus European version).<br>This parameter applies only when the starting block number is 1.<br>*Accepted values:* 1 to *n*, where *n* is the maximum number of bytes for the tag. For UCODE EPC 1.19, *n* is 32.<br>*Default value:* 1 |

**Examples •**

### Encode ASCII

This example encodes 96-bit data.

```
^XA
^RS4
^RFw,a^FD00 my data^FS
^XZ
```

### Encode Hex

This example encodes 64-bit data.

```
^XA
^RS3
^RFW,H^FD1122334455667788^FS
^XZ
```

### Encode EPC

This example encodes 96-bit data, as specified by the `^RB` command.

```
^XA
^RB96,8,3,3,20,24,38
^RFw,e^FD16,3,5,78742,146165,1234567891^FS
^XZ
```

# ^RI

## Get RFID Tag ID

**Important** • This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to get the unique serial number of the tag and return it in hexadecimal format. The data can be sent back to the host via the ^HV command.

**Format** `^RI#,s,r,m`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| # = number to be assigned to the field | *Accepted values:* 0 to 9999<br>*Default value:* 0 |
| s = specify data order | **Note** • This parameter applies only to the R110*Xi* HF and R2844-Z printers.<br>*Accepted values:*<br>0 = Most significant byte first for Tag*It and PicoTag. Least significant byte first for I*code and ISO15693.<br>1 = Reverse the data order<br>*Default value:* 0 |
| r = number of retries | *Accepted values:* 0 to 10<br>*Default value:* 0 |
| m = motion | *Accepted values:*<br>0 = Feed label after writing.<br>1 = No Feed after writing. Other ZPL may cause a feed.<br>*Default value:* 0 |

**Example** • This example reads a tag ID, prints it on a label, and sends string `Tag ID:xxxxxxxx` to the host. The data read will go into the ^FN0 location of the format. The printer will retry the command five times, if necessary.

```
^XA
^FO20,120^A0N,60^FN0^FS
^RI0,,5^FS
^HV0,,Tag ID:^FS
^XZ
```

# ^RM

## Enable RFID Motion

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to enable or disable RFID motion. By default, labels automatically print at the end of the format. This command allows you to inhibit the label from actually moving when it reaches the program position, which is useful for debugging, setup, and custom applications. This parameter is not persistent (carried over from label to label).

**Format** `^RMe`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `e` = enable | *Accepted values:*<br>`Y` = Yes, move the label<br>`N` = No, do not move the label<br>*Default value:* Y |

# ^RN

## Detect Multiple RFID Tags in Encoding Field

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to enable or disable detection of multiple RFID tags in the encoding field. By default, the printer checks for more than one tag in the field before attempting to read or write. If more than one tag is found, the label over the antenna support is voided, and the RFID ERR STATUS parameter on the control panel displays MULTIPLE TAGS. To speed up printing and encoding by up to 200 ms, the check may be disabled. This parameter is persistent (carried over from label to label).

**Format** `^RNe`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `e` = enable | *Accepted Values:*<br>`Y` = Yes, check for multiple tags<br>`N` = No, do not check for multiple tags<br>*Default Value:* `Y` |

# ~RO

## Reset Advanced Counters

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to reset the advanced counters used by the printer to monitor label generation in inches and centimeters, the number of labels printed, and the number of valid and voided RFID labels. Any single error during programming of an RFID tag will result in that label being considered “void” by the counter.

Four resettable counters are available. The values for the counters are displayed on the printer configuration label.

**Note •** For the R4Mplus, the counter values are not saved, so power cycling the printer resets all counters to zero.

**Format** `~ROc`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `c` = counter to reset | *Accepted Values:*<br>`1` = counter 1<br>`2` = counter 2<br>`3` = valid RFID label counter<br>`4` = voided RFID label counter<br>*Default Value:* None. If a value is not specified, the command is ignored. |

**Example 1 •** This example shows how the counter portion of the printer configuration labels looks when counter 1 is reset by sending `~RO1`.

**Before**

```
296862 IN.......... NONRESET CNTR
296862 IN.......... RESET CNTR1
296862 IN.......... RESET CNTR2
753289 CM.......... NONRESET CNTR
753289 CM.......... RESET CNTR1
753289 CM.......... RESET CNTR2
92928 LABLS........ NONRESET CNTR
92928 LABLS........ RESET CNTR1
92928 LABLS........ RESET CNTR2
```

**After**

```
296876 IN.......... NONRESET CNTR
0 IN............... RESET CNTR1
296876 IN.......... RESET CNTR2
753323 CM.......... NONRESET CNTR
0 CM............... RESET CNTR1
753323 CM.......... RESET CNTR2
92930 LABLS........ NONRESET CNTR
0 LABLS............ RESET CNTR1
92930 LABLS........ RESET CNTR2
```

**Example 2 •** This example shows how the counter portion of the printer configuration labels looks when the RFID counters are reset by sending `~RO3` and `~RO4`.

**Before**

```
02/10/05........... RTC DATE
07:21.............. RTC TIME
507................ RFID VALID CTR
4.................. RFID VOID CTR
HIGH............... RFID READ PWR
HIGH............... RFID WRITE PWR
RFID OK............ RFID ERR STATUS
Class 0............ RFID TAG TYPE
TM: 20050201....... RFID VERSION
1950 IN............ NONRESET CNTR
1950 IN............ RESET CNTR1
1950 IN............ RESET CNTR2
```

**After**

```
02/10/05........... RTC DATE
07:21.............. RTC TIME
0.................. RFID VALID CTR
0.................. RFID VOID CTR
HIGH............... RFID READ PWR
HIGH............... RFID WRITE PWR
RFID OK............ RFID ERR STATUS
Class 0............ RFID TAG TYPE
TM: 20050201....... RFID VERSION
1951 IN............ NONRESET CNTR
1951 IN............ RESET CNTR1
1951 IN............ RESET CNTR2
```

# ^RR

## Specify RFID Retries for a Block

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to change the number of times that the printer attempts to read or write to a particular block of a single RFID tag. By default, the printer will attempt six retries. This command is persistent and will be used in subsequent formats if not provided.

**Note •** This command's function is different than the "number of labels" parameter in the ^RS command.

**Format** ^RRn

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| n = number of retries | *Accepted values:* 0 to 10<br>*Default value:* 0 |

**Examples •**

**Set read block retries to 5**

```
^XA
^FN1^RR5^RFR,H^FS
^HV1^FS
^XZ
```

**Set write block retries to 2**

```
^XA
^RR2^RFW,H^FD1234^FS
^XZ
```

# ^RS

## Set Up RFID Parameters

**Description** Use this command to set up RFID parameters including tag type, read/write position of the transponder, and error handling.

**Important •** Use care when using this command in combination with `^RT` or `^RF` for reading tag data. Problems can occur if the data read from the tag is going to be printed on the label. Any data read from the transponder must be positioned to be printed above the read/write position. Failure to do this will prevent read data from being printed on the label.

**Format** `^RSt,p,v,n,e,a,c,s`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `t` = tag type | Tells the printer/print engine which tag type you are using. If you specify a tag type that is not supported by your printer or firmware, the printer uses the default value. For the supported tag types and defaults, see Table 14 on page 308.<br>**UHF Printers**<br>`0` = None<br>`1` = EPC Class 0<br>`2` = EPC Class 0 Plus<br>`3` = EPC Class 1 64-bit<br>`4` = EPC Class 1 96-bit<br>`5` = UCODE EPC 1.19<br>`6` = Impinj Class 0 Plus<br>`7` = ISO 18000-06A<br>`8` = EPC Class 1, Generation 2 (Gen 2)<br>`9` = ISO 18000-06B<br>**HF Printers**<br>**Note •** Only the R110*Xi* HF printer (firmware version R65.X.X) supports the use of letters for this parameter. All other printers use the numbers.<br>`A` or `0` = None<br>`B` or `1` = Auto detect (query tag to determine)<br>`C` or `2` = Tag*It (Texas Instruments Tagit tags)<br>`D` or `3` = `I`*code (Phillips Icode tags)<br>`E` or `4` = Pico Tag (Inside Technology's)<br>`F` or `5` = ISO 15693<br>`G` or `6` = EPC tag (13.56 MHz)<br>`H` or `7` = UID Tag<br>`I` or `8` = Mifare UltraLight |

| Parameters | Details |
|---|---|
| `p` = read/write position of the transponder | Sets the read/write position of the transponder in vertical (Y axis) dot rows from the top of the label. Set to 0 (no movement) if the transponder is already in the effective area without moving the media.<br>**Important •** If a label format specifies a value for this parameter, this value will be used for the programming position for all labels until a new position is specified or until the printer is turned off (**O**) and then back on (**I**).<br>*Accepted values:* 0 to label length<br>*Default value:*<br>**For the R110*PAX*4 and R2844-Z:** 0<br>**For all other supported printers:** label length minus 1 mm (1/16 in.) |
| `v` = length of void printout | Sets the length of the void printout in vertical (Y axis) dot rows.<br>*Accepted values:* 0 to label length<br>*Default value:* label length |
| `n` = number of labels | The number of labels that will be attempted in case of read/encode failure.<br>*Accepted values:* 1 to 10<br>*Default value:* 3 |
| `e` = error handling | If an error persists after the specified number of labels are tried, perform this error handling action.<br>*Accepted values:*<br>`N` = No action (printer drops the label format causing the error and moves to the next queued label)<br>`P` = Place printer in Pause mode (label format stays in the queue until the user cancels)<br>`E` = Place printer in Error mode (label format stays in the queue until the user cancels)<br>*Default value:* N<br>**Note •** You can set the printer to send an error message to the host for each failure. To enable or disable this unsolicited error message, refer to the `^SX` and `^SQ` ZPL commands. Use `V` for the condition type for an RFID error. |

| Parameters | Details |
| --- | --- |
| `a` = signals on applicator | **Note •** This parameter does not apply to the R2844-Z. For the R4Mplus, this parameter applies only to printers with firmware version SP994*X* (R4Mplus European version).<br><br>When the value for parameter `p` (read/write position of the transponder) is nonzero, this parameter changes the number of start and stop print signals required for printing.<br>In Single mode, one start print command is required. In Double mode, two are required, so the printer will resume printing only after the second start print command is received.<br>*Accepted values:*<br>`S` = single signal<br>`D` = double signal (For the R110*PAX*4, Double mode will work only if the read/write position is changed from the default of zero.)<br>*Default value:* `S` |
| `c` = certify tag with a pre-read | Not applicable. |
| `s` = void print speed | **Note •** This parameter does not apply to the R2844-Z. For the R4Mplus, this parameter applies only to printers with firmware version SP994*X* (R4Mplus European version).<br><br>If a label is voided, the speed at which “VOID” will be printed across the label.<br>*Accepted values:* any valid print speed<br>*Default value:* the printer’s maximum print speed |

**Supported Tag Types** Table 14 shows the tag types supported by different RFID printers/print engines and firmware versions. Depending on your country or on the firmware version that you are using, your printer may not support all of the tag types listed. If you specify an unsupported tag type in the `^RS` command, the printer uses the default value. If a tag type is shown as supported but does not work with your printer, you may need to upgrade the printer's firmware (see http://www.zebra.com/firmware).

### Table 14 • Supported Tag Types and Default Values

| Tag Type | UHF Printers | | | | | | HF Printers | |
|---|---|---|---|---|---|---|---|---|
| | R60.13.X | R62.13.X | R63.13.X | SP920X | SP994X, SP999X, SP1027X | SP1056X | R110*Xi* HF | R2844-Z |
| | R110*Xi* R170*Xi* | R110*PAX4* | | R4Mplus | | | | |
| **UHF Tag Types and Options** | | | | | | | | |
| None (no tag type specified) | * | * | — | — | — | — | — | — |
| EPC Class 0 | * | * | — | — | — | — | — | — |
| EPC Class 0 Plus | * | * | — | — | — | — | — | — |
| EPC Class 1 64-bit | * | * | — | * | * | — | — | — |
| EPC Class 1 96-bit | # | # | — | # | * | — | — | — |
| UCODE EPC 1.19 | *[a] | — | — | — | — | * | — | — |
| Impinj Class 0 Plus | * | * | — | — | — | — | — | — |
| ISO 18000-06A | — | * | — | — | * | — | — | — |
| EPC Class 1, Generation 2 (Gen 2) | * | * | — | — | * | # | — | — |
| ISO 18000-06B | * | * | — | — | # | — | — | — |
| **HF Tag Types and Options** | | | | | | | | |
| Auto-detect the tag type by querying the tag | — | — | — | — | — | — | — | # |
| Tag*It (Texas Instruments Tagit tags) | — | — | — | — | — | — | — | * |
| I*code (Phillips Icode tags) | — | — | — | — | — | — | — | * |
| Pico Tag (Inside Technology's) | — | — | — | — | — | — | — | * |
| ISO 15693 | — | — | — | — | — | — | # | * |
| EPC tag | — | — | — | — | — | — | — | * |
| UID Tag | — | — | — | — | — | — | — | — |
| Mifare UltraLight | — | — | — | — | — | — | * | — |

# = Default value
* = Accepted value

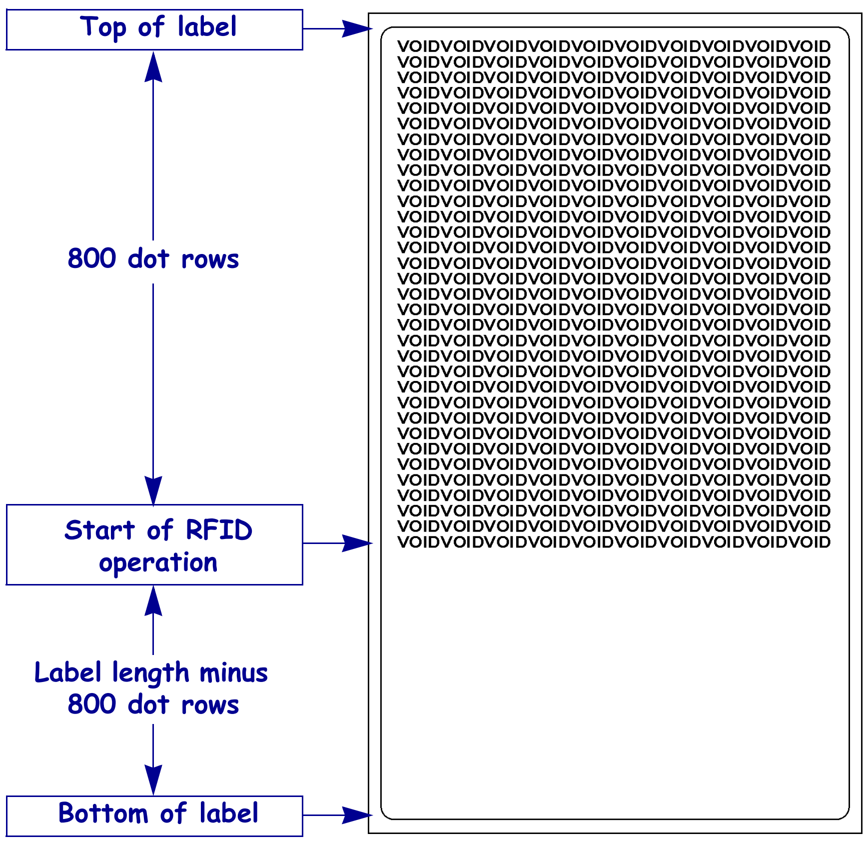
a. Requires R60.13.0.13ZD or higher.

**Example 1** • This example sets the printer to move the media to 800 dots from the top of the media [or label length minus 800 from the bottom (leading edge) of the media] and voids the rest of the media in case of an error. The printer will try to print two labels and then will pause if printing and encoding fail.

```
^XA
^RS,800,,2,P^FS
^XZ
```

Figure 2 shows the resulting voided label. Note where the void starts. The media has been moved 800 dot rows from the top of the label (label length minus 800 dot rows from the bottom (leading edge) of a label) to bring the transponder into the effective area to read/write a tag. If the printer fails the operation, the rest of the media is voided.

**Figure 2 • Sample Void Label, Remainder of Label Voided**

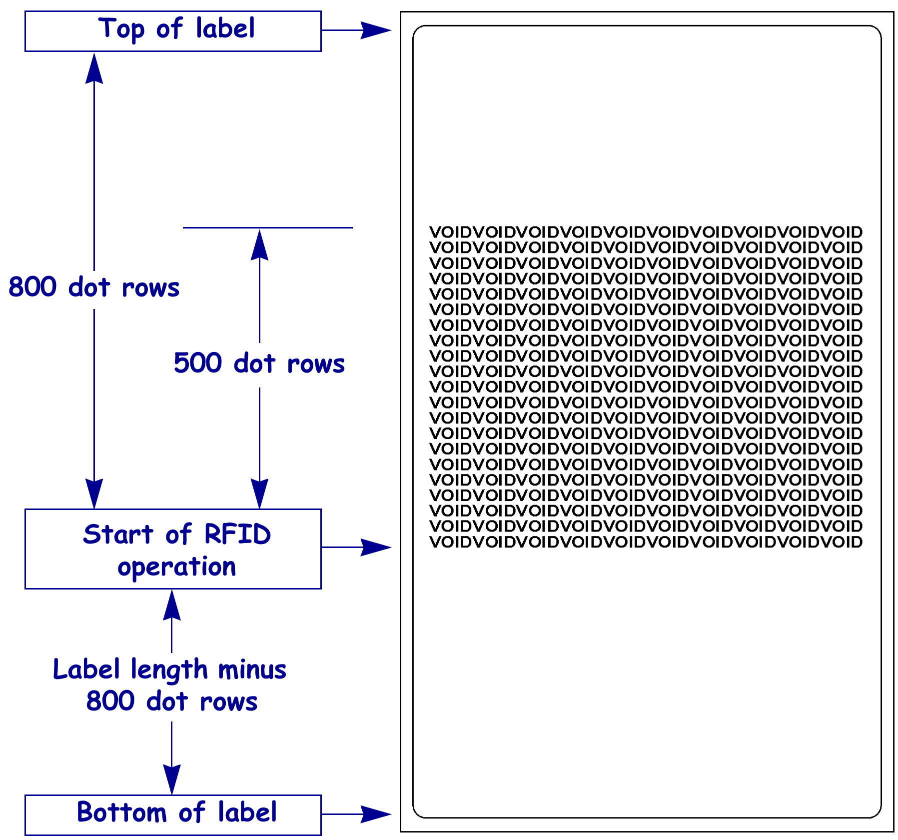


**Example 2 •** This example sets the printer to move the media to 800 dots from the top of the media [or label length - 500 from the bottom (leading edge) of the media] and prints “VOID” 500 dots in vertical length (Y axis) in case of an error.

```
^XA
^RS,800,500,2,P^FS
^XZ
```

Figure 3 shows the resulting voided label. Note where the void starts. The media has been moved 800 dot rows from the top of the label [label length minus 800 dot rows from the bottom (leading edge) of a label] to bring the transponder into the effective area to read/write a tag. If the printer fails the operation, an area that is 500 dot rows of the media is voided instead of the entire rest of the media.

**Figure 3 • Sample Void Label, 500 Dot Row Area Voided**



# ^RT

## Read RFID Tag

**Note •** The ^RT command is provided only for backward-compatibility with label formats that were developed for older Zebra RFID printers. See Table 13 on page 287 for the list of printers and firmware with which you should not use this command.

**Description** Use this command to tell the printer to read the current RFID tag data. The data can be returned to the host via the ^HV command.

**Format** `^RT#,b,n,f,r,m,s`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| # = number to be assigned to the field | *Accepted values:* 0 to 9999<br>*Default value:* 0 |
| b = starting block number | *Accepted values:* 0 to *n*, where *n* is the maximum number of blocks for the tag.<br>*Default value:* 0 |
| n = number of blocks/bytes to read | **Note •** This parameter does NOT apply to R4Mplus printers with firmware version SP920*X* (R4Mplus U.S. version).<br>**For R4Mplus printers with firmware version SP994*X* (European version):**<br>This parameter applies only when the starting block number (parameter b) is 1.<br>*Accepted values:* 1 to *n*, where *n* is the maximum number of bytes for the tag. For UCODE EPC 1.19, *n* is 32.<br>*Default value:* 1<br>**For all other supported printers:**<br>*Accepted values:* 1 to *n*, where *n* is the maximum number of blocks for the tag type minus the starting block number. For example, if the tag has 8 blocks (starting with block 0) and you start with block 6, *n* can be 2. This would return block 6 and block 7 information.<br>*Default value:* 1 |
| f = format | *Accepted values:*<br>0 = ASCII<br>1 = Hexadecimal<br>*Default value:* 0 |
| r = number of retries | Changes the number of times that the printer attempts to read a particular block of a single RFID tag. (Same retry rules as the ^RR command.)<br>*Accepted values:* 0 to 10<br>*Default value:* 0 |

| Parameters | Details |
|---|---|
| m = motion | Enables or disables RFID motion for the current field.<br>*Accepted values:*<br>`0` = Feed label after writing.<br>`1` = No feed after writing. Other ZPL may cause a feed.<br>*Default value:* 0 |
| s = special mode | **Note •** This parameter is used only for the printers referenced here.<br>**For R4Mplus printers with firmware version SP920*X* (U.S. version):**<br>Specify actions for mismatched checksums. For EPC Class 1 (Alien reader) only. Not applicable for EPC class 0.<br>*Default value:* 0<br>*Accepted values:*<br>`0` = Do not read if mismatched checksum<br>`1` = Read even if mismatched checksum<br>**For R110*Xi* HF and R2844-Z printers:**<br>Specify data order.<br>*Default value:* 0<br>*Accepted values:*<br>`0` = least significant byte first<br>`1` = most significant byte first |

**Example 1 •** This example reads a tag, prints the data on a label, and sends the string `Tag Data:`*`xxxxxxxx`* back to the host. The data read will go into the `^FN1` location of the format. The printer will retry the command five times, if necessary.

```
^XA
^FO20,120^A0N,60^FN1^FS
^RT1,,,,5^FS
^HV1,,Tag Data:^FS
^XZ
```

**Example 2 •** This example reads from a tag twice and prints the results on a label.

```
^XA
^FO20,120^A0N,60^FN1^FS
^FO20,100^A0N,20^FN2^FS
^RT1,7,3,,5^FS
^RT2,3,2,,5^FS
^XZ
```

The first `^RT` command starts at block 7 and reads three blocks of data in ASCII format. The data read will go into the `^FN1` location of the format. The printer will retry the command five times, if necessary.

The second `^RT` command starts at block 2 and reads two blocks of data in ASCII format. The data read will go into the `^FN2` location of the format. The printer will retry the command five times, if necessary.

# ~RV

## Report RFID Encoding Results

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to tell the printer to send RFID encoding success or failure results to the host computer after each label format completes.

**Format** `~RVa`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `a` = enable/disable | Enables or disables the results reporting feature.<br>*Accepted Values:*<br>`E` = Enable<br>`D` = Disable<br>*Default Value:* E |

**Example 1 •** Assume that the following code is sent and that there is no RFID tag in the field.

```
~RVE
^XA
^RS8,0,,3
^RMY
^RFR,H
^XZ
```

The printer attempts to program a tag three times and then returns the following to the host:

```
_-,3_
```

The minus sign indicates that the programming attempt failed entirely and voided three labels.

**Example 2 •** Assume that the same code is sent and that the first two attempts at programming a tag are unsuccessful. The third attempt succeeds.

```
~RVE
^XA
^RS8,0,,3
^RMY
^RFR,H
^XZ
```

The printer attempts to program a tag three times and then returns the following to the host:

```
_+,2_
```

The plus sign indicates that the programming attempt was successful and voided two labels.

# ^RW

## Set RFID Read and Write Power Levels

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to set the read and write power levels. This function is useful when using different tag types or transponders that require different power levels to obtain the best read and write abilities. If not enough power is applied, the transponder may not have sufficient power for programming, and tag data will fail to encode. If too much power is applied, the extra power may disable the tag may or cause data communication errors.

**Format** `^RWr,w`

This table identifies the parameters for this format:

| Parameters | Details |
|---|---|
| `r` = read power | Sets the power level to match the desired output as calibrated in the factory.<br>*Accepted Values:*<br>`H` = high<br>`M` = medium<br>`L` = low<br>*Default Value:* H |
| `w` = write power | **Note •** This parameter is ignored on the R110*Xi* HF printer because read and write powers cannot be specified separately. The printer uses the value that you specified for read power for both the read and write power settings.<br>Sets the power level to match the desired output as calibrated in the factory.<br>*Accepted Values:*<br>`H` = high<br>`M` = medium<br>`L` = low<br>*Default Value:* H |
| `a` = RFID antenna selection | **Note •** This parameter applies only to the R110*Xi* HF printer.<br>Selects the antenna port that provides the best results for reading and writing.<br>*Accepted Values:*<br>`1` = antenna port 1<br>`2` = antenna port 2<br>*Default Value:* 1 |

# ^RZ

## Set RFID Tag Password and Lock Tag

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to define a password for a tag during writing.

With Gen 2 tags, you can lock a tag's memory bank with an access password or define a kill password that can be used to permanently disable the tag. If you do not set access or kill passwords for a tag, the tag ignores any read or write commands that try to use these functions.

**Note •** The printer can set a kill password, but the printer cannot kill a tag.

**Format** `^RZp,m,l`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `p` = password | Sets a password. Gen 2 tags use a 32-bit password and specify the memory bank and lock style. Other tags use 8 bits and ignore the memory bank and lock style. Use *^RF* on page 297 to read the passwords.<br>*Accepted values:* 00 to FF (hexadecimal)<br>*Default value:* 00 |
| `m` = memory bank | This parameter applies to Gen 2 tags only.<br>*Accepted values:*<br>`K` = kill password<br>`A` = access password<br>`E` = EPC<br>`T` = tag identifier (TID)<br>`U` = user<br>*Default value:* none |
| `l` = lock style | This parameter applies to Gen 2 tags only.<br>*Accepted values:*<br>`U` = unlocked<br>`L` = locked<br>`O` = permanently unlocked (open)<br>`P` = permanently locked (protected)<br>`W` = write value (used only for the kill password memory bank)<br>*Default value:* none |

**Example 1** • On a tag that is not Gen 2, this example encodes 5A as the tag password.

```
^XA
^RZ5A^FS
^XZ
```

**Example 2** • On a Gen 2 tag, this example encodes EPC data 112233445566778899001122 to the tag in Hex format, write protects the tag's EPC data with password 1234ABCD, and leaves the tag's access password unlocked.

```
^XA
^RFW,H^FD112233445566778899001122^FS
^RZ1234ABCD,E,L^FS
^XZ
```

**Example 3** • On a Gen 2 tag, this example encodes EPC data 112233445566778899001122 to the tag in Hex format, write protects the tag's EPC data with password 1234ABCD, and makes the tag's access password unreadable.

```
^XA
^RFW,H^FD112233445566778899001122^FS
^RZ1234ABCD,E,L^FS
^RZ1234ABCD,A,L^FS
^XZ
```

The following code unprotects EPC data 112233445566778899001122 using the password 1234ABCD, encodes EPC data newdata to the tag in ASCII format, and then write protects the tag's new EPC data. The access password and its lock state are not changed, so the access password remains unreadable.

```
^XA
^RZ1234ABCD,E,U^FS
^RFW,A^FDnewdata^FS
^RZ1234ABCD,E,L^FS
^xz
```

**Example 4** • On a Gen 2 tag, this example unlocks the locked access password from the previous example.

```
^XA
^RZ1234ABCD,A,U^FS
^XZ
```

# ^WF

## Encode AFI or DSFID Byte

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to encode the AFI or DSFID byte to a tag. Error handling is set by the ^RS command.

**Format** `^WFr,m,w,f,b`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `r` = number of retries | *Accepted values:* 0 to 10<br>*Default value:* 0 |
| `m` = motion | *Accepted values:*<br>0 = Feed label after writing.<br>1 = No Feed after writing. Other ZPL may cause a feed.<br>*Default value:* 0 |
| `w` = write protect | *Accepted values:*<br>0 = Not write protected<br>1 = Write protect<br>*Default value:* 0 |
| `f` = data format | *Accepted values:*<br>0 = ASCII<br>1 = Hexadecimal<br>*Default value:* 0 |
| b = type of byte to read | *Accepted values:*<br>`A` = AFI byte<br>`D` = DSFID byte<br>*Default value:* A |

**Example 1 •** This example encodes data "R" (hex 52) as the AFI byte. The printer will try the command up to five times, if necessary.

```
^XA
^WF5^FDR
^XZ
```

**Example 2 •** This example encodes data hex 66 as the AFI byte. The printer will try the command up to three times, if necessary.

```
^XA
^WF3,,,1^FD66
^XZ
```

**Example 3 •** This example encodes data hex 77 as the DSFID byte. The printer will try the command up to four times, if necessary.

```
^XA
^WF4,,,1,D^FD77
^XZ
```

# ^WT

## Write (Encode) Tag

**Note •** The `^WT` command is provided only for backward-compatibility with label formats that were developed for older Zebra RFID printers. See Table 13 on page 287 for the list of printers and firmware with which you should not use this command.

**Description** Use this command to encode the current RFID tag. Check the amount of data memory available for the tag that you will be using. If you send more data than the memory can hold, the printer truncates the data.

**Format** `^WTb,r,m,w,f,v`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `b` = block number | Specifies the block number to encode. This parameter is tag-dependent.<br>• For most tags, use block 0.<br>• For EPC Class 0 Plus, block 0 is EPC data, and block 1 is user data.<br>• For the R4Mplus, this parameter does not apply to printers with firmware version SP902*X*. With other versions of firmware in this printer, you can encode 12 bytes (96 bits) to block 0, but you can encode only the first byte of block 1.<br>*Accepted values:* 0 to *n*, where *n* is the maximum number of blocks for the tag.<br>*Default value:* 0 |
| `r` = number of retries | Changes the number of times that the printer attempts to encode a particular block of a single RFID tag. (Same function as the `^RR` command.)<br>*Accepted values:* 0 to 10<br>*Default value:* 0 |
| `m` = motion | Enables or disables RFID motion. (Same function as the `^RM` command.)<br>*Accepted values:*<br>`0` = Feed label after writing<br>`1` = No feed after writing (other ZPL may cause a feed)<br>*Default value:* `0` |
| `w` = write protect | *Accepted values:*<br>`0` = Not write protected<br>`1` = Write protected<br>*Default value:* 0 |
| `f` = data format | *Accepted values:*<br>`0` = ASCII<br>`1` = Hexadecimal<br>*Default value:* 0 |

| Parameters | Details |
|---|---|
| | This parameter is not used in the R2844-Z. |
| **For the R110*Xi* HF:**<br>`v` = reverse the data order | **For the R110*Xi* HF:**<br>Reverses the data order.<br>*Accepted values:*<br>`N` = Do not reverse the data order (Most significant byte first for Tag*It and PicoTag. Last significant byte first for I*code and ISO 15693)<br>`Y` = Reverse the data order<br>*Default value:* N |
| **For other supported printers:**<br>`v` = verify valid data | **For other supported printers:**<br>For reliability, some manufacturers encode tags with known data (such as `A5A5`). This parameter flags whether the preprogrammed data is verified. (Same function as the `^WV` command.)<br>*Accepted values:*<br>`N` = Do not verify<br>`Y` = Verify valid data before writing<br>*Default value:* N |

**Example** • This sample encodes data "RFIDRFID" and will try writing up to five times, if necessary.

```
^XA
^WT,5^FDRFIDRFID^FS
^XZ
```

# ^WV

## Verify RFID Encoding Operation

**Important •** This command is not supported by all printers or firmware. See Table 13 on page 287 for the list of printers and firmware with which you can use this command.

**Description** Use this command to enable or disable the write verify function. When write verify is enabled, this command verifies the RFID encoding operation to ensure that the tag about to be programmed contains the hex data “A5A5” in the first two bytes. This parameter is not persistent (carried over from label to label).

**Format** `^WVe`

This table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `e` = enable | *Accepted values:* Y or N<br>*Default value:* N |

**Notes •**

# 4

# Wireless Commands

This section contains new or modified ZPL commands for the Wireless and Wireless Plus print servers.

# ^NB

## Search for Wired Print Server during Network Boot

**Description** Use this command to tell the printer whether to search for a wired print server at bootup. Table 15 shows the results of this check.

**Table 15 • Results of Check for Wired Print Server**

| Wired Print Server Connected? | Check for Wired Print Server? | Results |
|---|---|---|
| Yes | Skip | The printer does not check for or acknowledge the wired print server. The Wireless or Wireless Plus print server is used as the primary print server. |
| Yes | Check | The printer checks for a wired print server. If the printer detects the wired print server, it uses it as the primary print server. If the printer does not detect the wired print server, the Wireless or Wireless Plus print server is used as the primary print server. |
| No | Skip | The printer does not check for a wired print server. The Wireless or Wireless Plus print server is used as the primary print server. |
| No | Check | During bootup, the printer tries for 70 seconds to detect a wired print server. Because a wired print server is not connected, the printer uses the Wireless or Wireless Plus print server as the primary print server. |

**Format** `^NBa`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = check for wired print server at boot time | *Accepted Values:* C (CHECK), S (SKIP CHECK)<br>*Default Value:* S |

# ^NN

## Set SNMP

**Note •** Firmware versions SP1071 and SP1073 support SNMP traps only. Other SNMP parameters are not supported.

**Description** Use this command to set the Simple Network Management Protocol (SNMP) parameters.

**Format** `^NNa,b,c,d,e,f`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = system name | Same as printer name.<br>*Accepted Values:* Up to 17 alphanumeric characters |
| `b` = system contact | Any contact information as desired (such as a name or phrase)<br>*Accepted Values:* Up to 50 alphanumeric characters |
| `c` = system location | The printer's model information.<br>*Accepted Values:* Up to 50 alphanumeric characters |
| `d` = get community name | *Accepted Values:* Up to 19 alphanumeric characters<br>*Default Value:* public |
| `e` = set community name | *Accepted Values:* Up to 19 alphanumeric characters<br>*Default Value:* public |
| `f` = trap community name | *Accepted Values:* Up to 20 alphanumeric characters<br>*Default Value:* public |

# ^NP

## Set Primary/Secondary Device

**Description** Use this command to specify whether to use the printer's or the print server's LAN/WLAN settings at boot time. The default is to use the printer's settings.

When the printer is set as the primary device, you can set it up using ZPL commands or the SetWLAN utility, and any wired print server inserted into the printer will get these settings.

**Format** `^NPa`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = device to use as primary | *Accepted Values:* P (PRINTER), M (MPS/PRINTSERVER) |

# ^NS

## Change Wired Networking Settings

**Description** Use this command to change the wired print server network settings.

**Format** `^NSa,b,c,d,e,f,g,h,i`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = IP resolution | *Accepted Values:*<br>A (ALL)<br>B (BOOTP)<br>C (DHCP AND BOOTP)<br>D (DHCP)<br>G (GLEANING ONLY)<br>R (RARP)<br>P (PERMANENT)<br>**Note •** Use of GLEANING ONLY is not recommended when the Wireless Print Server is installed. |
| `b` = IP address | *Accepted Values:* Any properly formatted IP address in the xxx.xxx.xxx.xxx format. |
| `c` = subnet mask | *Accepted Values:* Any properly formatted subnet mask in the xxx.xxx.xxx.xxx format. |
| `d` = default gateway | *Accepted Values:* Any properly formatted gateway in the xxx.xxx.xxx.xxx format. |
| `e` = WINS server address | *Accepted Values:* Any properly formatted WINS server in the xxx.xxx.xxx.xxx format. |
| `f` = connection timeout checking | *Accepted Values:* Y (YES), N (NO) |
| `g` = timeout value | Time, in seconds, before the connection times out.<br>*Accepted Values:* 0 through 9999 |
| `h` = ARP broadcast interval | Time, in minutes, that the broadcast is sent to update the device's ARP cache. |
| `i` = base raw port number | The port number that the printer should use for its RAW data.<br>*Accepted Values:* 0 through 99999<br>*Default Value:* 9100 |

**Example •**

```
^XA
^NSa,192.168.0.1,255.255.255.0,192.168.0.2
```

# ^NT

## Set SMTP

**Description** Use this command to set the Simple Mail Transfer Protocol (SMTP) parameters. This allows you to set the e-mail settings for alerts.

**Format** `^NTa,b`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = SMTP server address | *Accepted Values:* Any properly formatted server address in the xxx.xxx.xxx.xxx format |
| `b` = print server domain | *Accepted Values:* Any properly formatted print server domain name. A domain name is one or more labels separated by a period (“dot”), and a label consists of letters, numbers, and hyphens. An example of a domain name is zebra.com |

# ^NW

## Set Web Authentication Timeout Value

**Description** Use this command to set the timeout value for the printer home page. The printer will prompt for the printer password only the first time that certain screens are accessed until 1) the web authentication timeout value is reached (default value is 5 minutes) or 2) the printer is reset. At that time, the printer will prompt for the password again.

**Format** `^NWa`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = timeout value | The timeout value in minutes for an IP address to be authenticated to the printer web pages.<br>*Accepted Values:* 0 (no secure pages can be accessed without entering the printer password) to 255 minutes<br>*Default Value:* 5 |

# ^WA

## Set Antenna Parameters

**Description** Use this command to set the values for the receive and transmit antenna.

**Format** `^WAa,b`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = receive antenna | *Accepted Values:* D (Diversity), L (Left), R (Right)<br>*Default Value:* D |
| `b` = transmit antenna | *Accepted Values:* D (Diversity), L (Left), R (Right)<br>*Default Value:* D |

# ^WE

## Set WEP Mode

**Description** Use this command to command enable Wired Equivalent Privacy (WEP) mode and set WEP values. WEP is a security protocol for wireless local area networks (WLANs).

**Important •** To use WEP, WPA must be disabled.

Use care to include the exact number of commas required for this command when setting encryption keys (parameters `e` through `h`). A missing or extra comma will cause the keys to be stored in the wrong slots and can disable the wireless system.

**Format** `^WEa,b,c,d,e,f,g,h`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = encryption mode | *Accepted Values:* OFF, 40 (40-bit encryption), 128 (128-bit encryption)<br>*Default Value:* OFF |
| `b` = encryption index | Tells the printer which encryption key to use.<br>*Accepted Values:* 1 (Key 1), 2 (Key 2), 3 (Key 3), 4 (Key 4)<br>*Default Value:* 1 |
| `c` = authentication type | *Accepted Values:* O (Open System), S (Shared Key)<br>*Default Value:* O<br>**Note •** If you enable Shared Key authentication with Encryption Mode set to OFF, this value resets to Open. |
| `d` = encryption key storage | *Accepted Values:* H (Hex key storage), S (string key storage)<br>*Default Value:* H |
| e, f, g, h = encryption keys 1 through 4 | *Accepted Values:* The actual value for the encryption key<br>The encryption mode affects what can be entered for the encryption keys:<br>• For 40-bit, encryption keys can be set to any 5 hex pairs or any 10 alphanumeric characters.<br>• For 128-bit, encryption keys can be set to any 13 hex pairs or any 26 alphanumeric characters.<br>**Note •** When using hex storage, do not add a leading 0x on the WEP key. |

**Example 1 •** This example sets encryption to 40-bit, activates encryption key 1, and sets encryption key 1 to the string `12345`.

```
^WE40,,,,12345
```

In this example, the Encryption Index, Authentication Type, and Encryption Key Storage parameters are left blank with commas as placeholders for the fields. The printer uses the default values for these parameters.

**Example 2** • This example sets encryption to 128-bit, activates encryption key 2, and sets encryption keys 1 and 2 to hex values.

```
^WE128,2,,H,12345678901234567890123456,98765432109876543
210987654
```

The value for encryption key 1 is stored and can be activated in the future by the following command:

```
^WE128,1
```

**Example 3** • This example sets encryption to 128-bit, activates encryption key 4, and sets encryption key 4 to a hex value.

```
^WE128,4,,H,,,,98765432109876543210987654
```

Values are not required for encryption keys 1 through 3 when setting encryption key 4. In this example, commas are used as placeholders for the fields for encryption keys 1 through 3. Any previously stored values for these encryption keys do not change.

**Important** • Make sure that you include the exact number of commas required to get to the slot for encryption key 4 (parameter `h`).

# ^WI

## Change Wireless Network Settings

**Description** Use this command to change the wireless network settings.

**Format** `^WIa,b,c,d,e,f,g,h,i`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = IP resolution | *Accepted Values:*<br>A (ALL)<br>B (BOOTP)<br>C (DHCP AND BOOTP)<br>D (DHCP)<br>G (GLEANING ONLY)<br>R (RARP)<br>P (PERMANENT)<br>**Note •** Use of GLEANING ONLY is not recommended when the Wireless Print Server is installed. |
| `b` = IP address | *Accepted Values:* Any properly formatted IP address in the xxx.xxx.xxx.xxx format. |
| `c` = subnet mask | *Accepted Values:* Any properly formatted subnet mask in the xxx.xxx.xxx.xxx format. |
| `d` = default gateway | *Accepted Values:* Any properly formatted gateway in the xxx.xxx.xxx.xxx format. |
| `e` = WINS server address | *Accepted Values:* Any properly formatted WINS server in the xxx.xxx.xxx.xxx format. |
| `f` = connection timeout checking | *Accepted Values:* Y (YES), N (NO) |
| `g` = timeout value | Time, in seconds, before the connection times out.<br>*Accepted Values:* 0 through 9999 |
| `h` = ARP broadcast interval | Time, in minutes, that the broadcast is sent to update devices ARP cache. |
| `i` = base raw port number | The port number that the printer should use for its RAW data.<br>*Accepted Values:* 0 through 99999<br>*Default Value:* 9100 |

# ^WL

## Set LEAP Parameters

**Description** Use this command to enable Cisco® Lightweight Extensible Authentication Protocol (LEAP) mode and set parameters. LEAP is user authentication method that is available with some wireless cards.

**Format** `^WLa,b,c`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = mode | *Accepted Values:* OFF, ON<br>*Default Value:* OFF |
| `b` = user name | *Default Value:* Any 4 to 40 alphanumeric characters user |
| `c` = password | *Default Value:* Any 4 to 40 alphanumeric characters password |

# ~WL

## Print Network Configuration Label

**Description** Use this command to generate a network configuration label (Figure 4).

**Format** `~WL`

**Figure 4 • Network Configuration Label**

```
Network Configuration

Zebra Technologies
PRINTER MODEL XXXdpi
USER-DEFINED TEXT

NO................. WIRED PS CHECK?
Printer............ LOAD LAN FROM?

Wired
ALL................ IP PROTOCOL
000.000.000.000.... IP ADDRESS
000.000.000.000.... SUBNET MASK
000.000.000.000.... DEFAULT GATEWAY
000.000.000.000.... WINS SERVER IP
YES................ TIMEOUT CHECKING
0300............... TIMEOUT VALUE
0000............... ARP INTERVAL
9100............... BASE RAW PORT

Wireless*
ALL................ IP PROTOCOL
010.003.015.089.... IP ADDRESS
255.255.255.000.... SUBNET MASK
010.003.015.001.... DEFAULT GATEWAY
010.003.001.015.... WINS SERVER IP
YES................ TIMEOUT CHECKING
0300............... TIMEOUT VALUE
0000............... ARP INTERVAL
9100............... BASE RAW PORT
YES................ CARD INSERTED
015FH.............. CARD MFG ID
000AH.............. CARD PRODUCT ID
000e83df3bc7....... MAC ADDRESS
YES................ DRIVER INSTALLED
INFRASTRUCTURE..... OPERATING MODE
vh-CTC-PRD......... ESSID
100................ TX POWER
ON................. 1 Mb/s
ON................. 2 Mb/s
ON................. 5.5 Mb/s
ON................. 11 Mb/s
11 Mb/s............ CURRENT TX RATE
DIVERSITY.......... RECEIVE ANTENNA
DIVERSITY.......... XMIT ANTENNA
OPEN............... AUTH. TYPE
OFF................ LEAP MODE
128-bit............ ENCRYPTION MODE
1.................. ENCRYPT. INDEX
020................ POOR SIGNAL
LONG............... PREAMBLE
YES................ ASSOCIATED
FIRMWARE IN THIS PRINTER IS COPYRIGHTED
```

# ^WP

## Set Wireless Password

**Description** Use this command to set the four-digit wireless password (not the same as the general printer password). If the wireless password is **0000**, the Wireless and Wireless Plus print servers run in an "unprotected" mode, which means that you do not need to enter the wireless password through the control panel to view or modify wireless settings.

If a wireless password is set, the values for the following parameters will not appear through the control panel until the wireless password is entered:

- MAC Address
- ESSID
- Leap Mode (if applicable)
- Reset Network

**Format** `^WPa,b`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = old wireless password | *Accepted Values:* 0000 through 9999<br>*Default Value:* 0000 |
| `b` = new wireless password | *Accepted Values:* 0000 through 9999<br>*Default Value:* 0000 |

# ^WR

## Set Transmit Rate

**Description** Use this command to change the transmission parameters.

**Format** `^WRa,b,c,d,e`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `a` = rate 1 | Sets the 1 Mb/s transmit rate.<br>*Accepted Values:* Y (On), N (Off) |
| `b` = rate 2 | Sets the 2 Mb/s transmit rate.<br>*Accepted Values:* Y (On), N (Off) |
| `c` = rate 5.5 | Sets the 5.5 Mb/s transmit rate.<br>*Accepted Values:* Y (On), N (Off) |
| `d` = rate 11 | Sets the 11 Mb/s transmit rate.<br>*Accepted Values:* Y (On), N (Off) |
| `e` = transmit power | *Accepted Values:* 1, 5, 20, 30, 50, 100 |

# ~WR

## Reset Wireless Card

**Description** Use this command to reinitialize the wireless card and the print server (wired or wireless) when the Wireless or Wireless Plus print server is running. The command also causes any wireless card in the printer to reassociate to the wireless network. Same function as the `RESET NETWORK` control panel parameter.

**Format** `~WR`

# ^WS

## Set Wireless Card Values

**Description** Use this command to set the wireless card values for ESSID, Operating Mode, and Card Preamble.

**Format** `^WSe,o,p`

The following table identifies the parameters for this format.

| Parameters | Details |
|---|---|
| `e` = ESSID value | *Accepted Values:* Any value up to 32 characters, including all ASCII and Extended ASCII characters, including the space character. When this parameter is left blank, the ESSID is not changed.<br>*Default Value:* 125 |
| `o` = operating mode | *Accepted Values:* I (Infrastructure), A (Adhoc)<br>*Default Value:* I |
| `p` = wireless card preamble | *Accepted Values:* L (Long), S (Short)<br>*Default Value:* L |

**Notes •**

# A

# Zebra Code Pages

This section provides you with a visual of the different Zebra Code pages.

# Zebra Code Page 850

This is the Zebra Code Page 850:

**Note •** For hex 5C, a cent sign prints for all printer resident fonts. A backslash prints for downloaded fonts.

| CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | 20 | 32 | 0 | 30 | 48 | @ | 40 | 64 | P | 50 | 80 | ` | 60 | 96 | p | 70 | 112 | Ç | 80 | 128 |
| ! | 21 | 33 | 1 | 31 | 49 | A | 41 | 65 | Q | 51 | 81 | a | 61 | 97 | q | 71 | 113 | ü | 81 | 129 |
| " | 22 | 34 | 2 | 32 | 50 | B | 42 | 66 | R | 52 | 82 | b | 62 | 98 | r | 72 | 114 | é | 82 | 130 |
| # | 23 | 35 | 3 | 33 | 51 | C | 43 | 67 | S | 53 | 83 | c | 63 | 99 | s | 73 | 115 | â | 83 | 131 |
| $ | 24 | 36 | 4 | 34 | 52 | D | 44 | 68 | T | 54 | 84 | d | 64 | 100 | t | 74 | 116 | ä | 84 | 132 |
| % | 25 | 37 | 5 | 35 | 53 | E | 45 | 69 | U | 55 | 85 | e | 65 | 101 | u | 75 | 117 | à | 85 | 133 |
| & | 26 | 38 | 6 | 36 | 54 | F | 46 | 70 | V | 56 | 86 | f | 66 | 102 | v | 76 | 118 | å | 86 | 134 |
| ' | 27 | 39 | 7 | 37 | 55 | G | 47 | 71 | W | 57 | 87 | g | 67 | 103 | w | 77 | 119 | ç | 87 | 135 |
| ( | 28 | 40 | 8 | 38 | 56 | H | 48 | 72 | X | 58 | 88 | h | 68 | 104 | x | 78 | 120 | ê | 88 | 136 |
| ) | 29 | 41 | 9 | 39 | 57 | I | 49 | 73 | Y | 59 | 89 | i | 69 | 105 | y | 79 | 121 | ë | 89 | 137 |
| * | 2a | 42 | : | 3a | 58 | J | 4a | 74 | Z | 5a | 90 | j | 6a | 106 | z | 7a | 122 | è | 8a | 138 |
| + | 2b | 43 | ; | 3b | 59 | K | 4b | 75 | [ | 5b | 91 | k | 6b | 107 | { | 7b | 123 | ï | 8b | 139 |
| , | 2c | 44 | < | 3c | 60 | L | 4c | 76 | ¢ / \ | 5c | 92 | l | 6c | 108 | \| | 7c | 124 | î | 8c | 140 |
| - | 2d | 45 | = | 3d | 61 | M | 4d | 77 | ] | 5d | 93 | m | 6d | 109 | } | 7d | 125 | ì | 8d | 141 |
| . | 2e | 46 | > | 3e | 62 | N | 4e | 78 | ^ | 5e | 94 | n | 6e | 110 | ~ | 7e | 126 | Ä | 8e | 142 |
| / | 2f | 47 | ? | 3f | 63 | O | 4f | 79 | _ | 5f | 95 | o | 6f | 111 | ⌂ | 7f | 127 | Å | 8f | 143 |

| CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| É | 90 | 144 | á | a0 | 160 | ░ | b0 | 176 | └ | c0 | 192 | ð | d0 | 208 | Ó | e0 | 224 | - | f0 | 240 |
| æ | 91 | 145 | í | a1 | 161 | ▒ | b1 | 177 | ┴ | c1 | 193 | Ð | d1 | 209 | ß | e1 | 225 | ± | f1 | 241 |
| Æ | 92 | 146 | ó | a2 | 162 | ▓ | b2 | 178 | ┬ | c2 | 194 | Ê | d2 | 210 | Ô | e2 | 226 | ‗ | f2 | 242 |
| ô | 93 | 147 | ú | a3 | 163 | │ | b3 | 179 | ├ | c3 | 195 | Ë | d3 | 211 | Ò | e3 | 227 | ¾ | f3 | 243 |
| ö | 94 | 148 | ñ | a4 | 164 | ┤ | b4 | 180 | ─ | c4 | 196 | È | d4 | 212 | õ | e4 | 228 | ¶ | f4 | 244 |
| ò | 95 | 149 | Ñ | a5 | 165 | Á | b5 | 181 | ┼ | c5 | 197 | ı | d5 | 213 | Õ | e5 | 229 | § | f5 | 245 |
| û | 96 | 150 | ª | a6 | 166 | Â | b6 | 182 | ã | c6 | 198 | Í | d6 | 214 | µ | e6 | 230 | ÷ | f6 | 246 |
| ù | 97 | 151 | º | a7 | 167 | À | b7 | 183 | Ã | c7 | 199 | Î | d7 | 215 | þ | e7 | 231 | ¸ | f7 | 247 |
| ÿ | 98 | 152 | ¿ | a8 | 168 | © | b8 | 184 | ╚ | c8 | 200 | Ï | d8 | 216 | Þ | e8 | 232 | ° | f8 | 248 |
| Ö | 99 | 153 | ® | a9 | 169 | ╣ | b9 | 185 | ╔ | c9 | 201 | ┘ | d9 | 217 | Ú | e9 | 233 | ¨ | f9 | 249 |
| Ü | 9a | 154 | ¬ | aa | 170 | ║ | ba | 186 | ╩ | ca | 202 | ┌ | da | 218 | Û | ea | 234 | · | fa | 250 |
| ø | 9b | 155 | ½ | ab | 171 | ╗ | bb | 187 | ╦ | cb | 203 | █ | db | 219 | Ù | eb | 235 | ¹ | fb | 251 |
| £ | 9c | 156 | ¼ | ac | 172 | ╝ | bc | 188 | ╠ | cc | 204 | ▄ | dc | 220 | ý | ec | 236 | ³ | fc | 252 |
| Ø | 9d | 157 | ¡ | ad | 173 | ¢ | bd | 189 | ═ | cd | 205 | ¦ | dd | 221 | Ý | ed | 237 | ² | fd | 253 |
| × | 9e | 158 | « | ae | 174 | ¥ | be | 190 | ╬ | ce | 206 | Ì | de | 222 | ¯ | ee | 238 | ■ | fe | 254 |
| ƒ | 9f | 159 | » | af | 175 | ┐ | bf | 191 | ¤ | cf | 207 | ▀ | df | 223 | ´ | ef | 239 |  | ff | 255 |

# Zebra Code Page 1252

This is the Zebra Code Page 1252:

| CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 20 | 32 | 0 | 30 | 48 | @ | 40 | 64 | P | 50 | 80 | ` | 60 | 96 | p | 70 | 112 | € | 80 | 128 |
| ! | 21 | 33 | 1 | 31 | 49 | A | 41 | 65 | Q | 51 | 81 | a | 61 | 97 | q | 71 | 113 | | 81 | 129 |
| " | 22 | 34 | 2 | 32 | 50 | B | 42 | 66 | R | 52 | 82 | b | 62 | 98 | r | 72 | 114 | ‚ | 82 | 130 |
| # | 23 | 35 | 3 | 33 | 51 | C | 43 | 67 | S | 53 | 83 | c | 63 | 99 | s | 73 | 115 | ƒ | 83 | 131 |
| $ | 24 | 36 | 4 | 34 | 52 | D | 44 | 68 | T | 54 | 84 | d | 64 | 100 | t | 74 | 116 | „ | 84 | 132 |
| % | 25 | 37 | 5 | 35 | 53 | E | 45 | 69 | U | 55 | 85 | e | 65 | 101 | u | 75 | 117 | … | 85 | 133 |
| & | 26 | 38 | 6 | 36 | 54 | F | 46 | 70 | V | 56 | 86 | f | 66 | 102 | v | 76 | 118 | † | 86 | 134 |
| ' | 27 | 39 | 7 | 37 | 55 | G | 47 | 71 | W | 57 | 87 | g | 67 | 103 | w | 77 | 119 | ‡ | 87 | 135 |
| ( | 28 | 40 | 8 | 38 | 56 | H | 48 | 72 | X | 58 | 88 | h | 68 | 104 | x | 78 | 120 | ˆ | 88 | 136 |
| ) | 29 | 41 | 9 | 39 | 57 | I | 49 | 73 | Y | 59 | 89 | i | 69 | 105 | y | 79 | 121 | ‰ | 89 | 137 |
| * | 2a | 42 | : | 3a | 58 | J | 4a | 74 | Z | 5a | 90 | j | 6a | 106 | z | 7a | 122 | Š | 8a | 138 |
| + | 2b | 43 | ; | 3b | 59 | K | 4b | 75 | [ | 5b | 91 | k | 6b | 107 | { | 7b | 123 | ‹ | 8b | 139 |
| , | 2c | 44 | < | 3c | 60 | L | 4c | 76 | \ | 5c | 92 | l | 6c | 108 | \| | 7c | 124 | Œ | 8c | 140 |
| - | 2d | 45 | = | 3d | 61 | M | 4d | 77 | ] | 5d | 93 | m | 6d | 109 | } | 7d | 125 | | 8d | 141 |
| . | 2e | 46 | > | 3e | 62 | N | 4e | 78 | ^ | 5e | 94 | n | 6e | 110 | ~ | 7e | 126 | Ž | 8e | 142 |
| / | 2f | 47 | ? | 3f | 63 | O | 4f | 79 | _ | 5f | 95 | o | 6f | 111 | | 7f | 127 | | 8f | 143 |

| CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC | CHR | HEX | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 90 | 144 | | a0 | 160 | ° | b0 | 176 | À | c0 | 192 | Ð | d0 | 208 | à | e0 | 224 | ð | f0 | 240 |
| ‘ | 91 | 145 | ¡ | a1 | 161 | ± | b1 | 177 | Á | c1 | 193 | Ñ | d1 | 209 | á | e1 | 225 | ñ | f1 | 241 |
| ’ | 92 | 146 | ¢ | a2 | 162 | ² | b2 | 178 | Â | c2 | 194 | Ò | d2 | 210 | â | e2 | 226 | ò | f2 | 242 |
| “ | 93 | 147 | £ | a3 | 163 | ³ | b3 | 179 | Ã | c3 | 195 | Ó | d3 | 211 | ã | e3 | 227 | ó | f3 | 243 |
| ” | 94 | 148 | ¤ | a4 | 164 | ´ | b4 | 180 | Ä | c4 | 196 | Ô | d4 | 212 | ä | e4 | 228 | ô | f4 | 244 |
| • | 95 | 149 | ¥ | a5 | 165 | µ | b5 | 181 | Å | c5 | 197 | Õ | d5 | 213 | å | e5 | 229 | õ | f5 | 245 |
| – | 96 | 150 | ¦ | a6 | 166 | ¶ | b6 | 182 | Æ | c6 | 198 | Ö | d6 | 214 | æ | e6 | 230 | ö | f6 | 246 |
| — | 97 | 151 | § | a7 | 167 | · | b7 | 183 | Ç | c7 | 199 | × | d7 | 215 | ç | e7 | 231 | ÷ | f7 | 247 |
| ˜ | 98 | 152 | ¨ | a8 | 168 | ¸ | b8 | 184 | È | c8 | 200 | Ø | d8 | 216 | è | e8 | 232 | ø | f8 | 248 |
| ™ | 99 | 153 | © | a9 | 169 | ¹ | b9 | 185 | É | c9 | 201 | Ù | d9 | 217 | é | e9 | 233 | ù | f9 | 249 |
| š | 9a | 154 | ª | aa | 170 | º | ba | 186 | Ê | ca | 202 | Ú | da | 218 | ê | ea | 234 | ú | fa | 250 |
| › | 9b | 155 | « | ab | 171 | » | bb | 187 | Ë | cb | 203 | Û | db | 219 | ë | eb | 235 | û | fb | 251 |
| œ | 9c | 156 | ¬ | ac | 172 | ¼ | bc | 188 | Ì | cc | 204 | Ü | dc | 220 | ì | ec | 236 | ü | fc | 252 |
| | 9d | 157 | - | ad | 173 | ½ | bd | 189 | Í | cd | 205 | Ý | dd | 221 | í | ed | 237 | ý | fd | 253 |
| ž | 9e | 158 | ® | ae | 174 | ¾ | be | 190 | Î | ce | 206 | Þ | de | 222 | î | ee | 238 | þ | fe | 254 |
| Ÿ | 9f | 159 | ¯ | af | 175 | ¿ | bf | 191 | Ï | cf | 207 | ß | df | 223 | ï | ef | 239 | ÿ | ff | 255 |

**Notes •**

# B

# ASCII

This section shows the American Standard Code for Information Interchange (ASCII) code used by Zebra printers.

# ASCII Code Chart

Shaded areas in Table 16 indicate characters not recommended for command prefix, format prefix, or delimiter characters.

**Table 16 • Zebra Used ASCII Code**

| HEX | Character | HEX | Character | HEX | Character | HEX | Character |
|---|---|---|---|---|---|---|---|
| 00 | NUL | 20 | Space | 40 | @ | 60 | ‘ |
| 01 | SOH | 21 | ! | 41 | A | 61 | a |
| 02 | STX | 22 | “ | 42 | B | 62 | b |
| 03 | ETX | 23 | # | 43 | C | 63 | c |
| 04 | EOT | 24 | $ | 44 | D | 64 | d |
| 05 | ENQ | 25 | % | 45 | E | 65 | e |
| 06 | ACK | 26 | & | 46 | F | 66 | f |
| 07 | BEL | 27 | ‘ | 47 | G | 67 | g |
| 08 | BS | 28 | ( | 48 | H | 68 | h |
| 09 | HT | 29 | ) | 49 | I | 69 | i |
| 0A | LF | 2A | * | 4A | J | 6A | j |
| 0B | VT | 2B | + | 4B | K | 6B | k |
| 0C | FF | 2C | , | 4C | L | 6C | l |
| 0D | CR | 2D | - | 4D | M | 6D | m |
| 0E | SO | 2E | . | 4E | N | 6E | n |
| 0F | SI | 2F | / | 4F | O | 6F | o |
| 10 | DLE | 30 | 0 | 50 | P | 70 | p |
| 11 | DC1 | 31 | 1 | 51 | Q | 71 | q |
| 12 | DC2 | 32 | 2 | 52 | R | 72 | r |
| 13 | DC3 | 33 | 3 | 53 | S | 73 | s |
| 14 | DC4 | 34 | 4 | 54 | T | 74 | t |
| 15 | NAK | 35 | 5 | 55 | U | 75 | u |
| 16 | SYN | 36 | 6 | 56 | V | 76 | v |
| 17 | ETB | 37 | 7 | 57 | W | 77 | w |
| 18 | CAN | 38 | 8 | 58 | X | 78 | x |
| 19 | EM | 39 | 9 | 59 | Y | 79 | y |
| 1A | SUB | 3A | : | 5A | Z | 7A | z |
| 1B | ESC | 3B | ; | 5B | [ | 7B | { |
| 1C | FS | 3C | < | 5C | \ | 7C | \| |
| 1D | GS | 3D | = | 5D | ] | 7D | } |

**Table 16 • Zebra Used ASCII Code (Continued)**

| HEX | Character | HEX | Character | HEX | Character | HEX | Character |
|---|---|---|---|---|---|---|---|
| 1E | RS | 3E | > | 5E | ^ | 7E | ~ |
| 1F | US | 3F | ? | 5F | _ | 7F | DEL |

**Notes •**

# C

# Fonts and Bar Codes

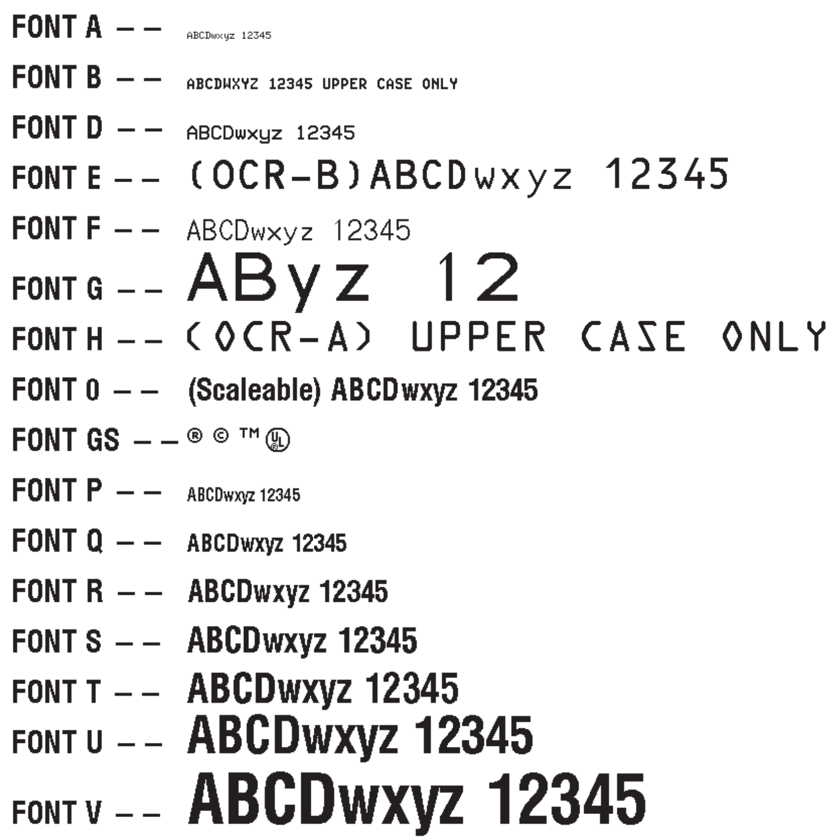
This section provides information about different fonts (type faces) and bar codes that can be used with the printer.

## Contents

# Standard Printer Fonts

Most Zebra printers come standard with 15 bitmapped fonts and one scalable font, see Figure 5. Additional downloadable bitmapped and scalable fonts are also available. Character size and density (how dark it appears) depend on the density of the printhead and the media used.

**Figure 5 • Examples of the Standard Printer Fonts**




To use one of these fonts, you must either use the change alphanumeric default font command (`^CF`) or specify an alphanumeric field command (`^A`).

The standard Zebra character set is Code 850 for character values greater than 20 HEX. There are six HEX character values below 20 HEX that are also recognized. Figure 6 shows how these character values are printed.

**Note •** Unidentified characters should default to a space.

**Figure 6 • Recognized HEX Values below 20 HEX**

```
A HEX  1a  will print a   0 (numeric)
A HEX  1b  will print a   ⅓
A HEX  1c  will print a   ⅔
A HEX  1d  will print a   Ĳ
A HEX  1e  will print a   ĳ
A HEX  1f  will print a   \
```

# Proportional and Fixed Spacing

Proportional spacing is different than fixed spacing. In Table 17, the intercharacter gap (space between characters) is constant for fonts A through H, which means that the spacing between all characters is the same. For example, the spacing between the letters `MW` is the same as between the letters `IE`.

The baseline is the imaginary line on which the bottom (base) of all characters (except any descenders) rest. The area between the baseline and the bottom of the matrix is used for any character “descenders.” Baseline numbers define where the baseline is located in relationship to the top of the matrix. For example, the baseline for font “E” is 23 dots down from the top of the matrix.

**Table 17 • Intercharacter Gap and Baseline Parameters**

| Font | H x W (in dots) | Type | Intercharacter Gap (in dots) | Baseline (in dots) |
|---|---|---|---|---|
| A | 9 x 5 | U-L-D | 1 | 7 |
| B | 11 x 7 | U | 2 | 11 |
| C,D | 18 x 10 | U-L-D | 2 | 14 |
| E | 28 x 15 | OCR-B | 5 | 23 |
| F | 26 x 13 | U-L-D | 3 | 21 |
| G | 60 x 40 | U-L-D | 8 | 48 |
| H | 21 x 13 | OCR-A | 6 | 21 |
| GS | 24 x 24 | SYMBOL | PROPORTIONAL | 3 x HEIGHT/4 |
| 0 | DEFAULT: 15 x 12 | | PROPORTIONAL | 3 x HEIGHT/4 |

# Scalable Versus Bitmapped Fonts

For scalable fonts, setting the height and width equally produces characters that appear the most balanced. Balanced characters are pleasing to the eye because actual height and width are approximately equal to each other. This is achieved through the use of a smooth-scaling algorithm in the printer.

For bitmapped fonts, this balancing is built into the font. In actuality, the height of a bitmap font is slightly larger than the width. Bitmap fonts are always at the maximum size of the character's cell.

## Scalable Fonts

All dot parameters used in the commands to create scalable fonts are translated into a point size because scalable fonts are measured in point sizes, not dots. To determine how many dots to enter to obtain a particular point size, use the following formula. The actual point size will be an approximate value.

$$\text{Dots} = \frac{\text{(Point size) x (Dots per inch of Printer)}}{72}$$

- For printers using a 6 dot/mm printhead the "dots per inch of printer" value is 152.4
- For printers using a 8 dot/mm printhead the "dots per inch of printer" value is 203.2
- For printers using a 12 dot/mm printhead the "dots per inch of printer" value is 304.8
- For printers using a 24 dot/mm printhead the "dots per inch of printer" value is 609.6

The actual height and width of the character in dots will vary, depending on the font style and the particular character. Therefore, some characters will be smaller and some will be larger than the actual dot size requested. The baselines for all scalable fonts are calculated against the dot size of the cell. The baseline is 3/4 down from the top of the cell. For example, if the size of the cell is 80 dots, the baseline will be 60 dots (3/4) down from the top of the cell.

For more information concerning fonts and related commands, see *~DB* on page 123 and *~DS* on page 131.

## Bitmapped Fonts

Internal bitmapped fonts can be magnified from 1 to 10 times their normal (default) size. The magnification factor is in whole numbers. Therefore, if the normal size of a bitmapped font is 9 dots high and 5 dots wide, a magnification factor of 3 would produce a character of 27 dots high and 15 dots wide. Height and width can be magnified independently.

### Magnification Factor

The font commands contain parameters for entering the height and width of printed characters. The values are always entered in dots. When entering these values for bitmapped fonts, use the following formula:

Base Height x Magnification Factor = Height Parameter Value

The same principle applies when calculating width.

Example:

Base height = 9 dots

Base width = 5 dots

To magnify a bitmapped character with the above specifics 3 times its size:

Height parameter = 27 [9 x 3]

Width parameter = 15 [5 x 3]

## Changing Bitmapped Font Size

Alphanumeric field command (`^A`) parameters h and w control the magnification and, therefore, the ultimate size of the font. The parameter is specified in dots, but ZPL II actually uses an integer multiplier times the original height/width of the font. For example, if you specify

```
^AD,54
```

you get characters three times their normal size (54 dots high), but if you specify

```
^AD,52
```

you receive the same result, not characters 52 dots high.

Defining only the height or width of a bitmapped font forces the magnification to be proportional to the parameter defined. If neither is defined, the `^CF` height and width are used. For example, if the height is twice the standard height, the width will be twice the standard width.

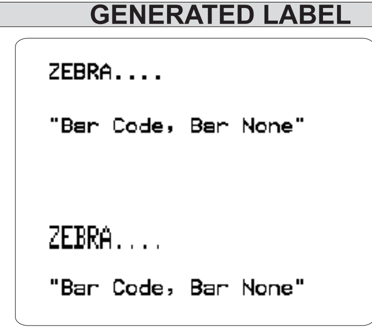
**Example •** If a `^CF` command, with height and width parameters defined, is used to set the first font, any `^A` commands (to select a different font) that follow must have the height and width parameter filled in.

If this is not done, the newly selected font will be magnified using values for the `^CF` height and width parameters. This is an example of what happens:.

**ZPL II CODE**

```
^XA^LL1800
^FO50,50^CFD,26,10^FDZEBRA....^FS
^FO50,100^FD"Bar Code, Bar None"^FS
^FO50,200^AA^FDZEBRA....^FS
^FO50,250^FD"Bar Code, Bar None"^FS
^XZ
```

**GENERATED LABEL**



# Font Matrices

**Type Key** **U** = Uppercase, **L** = Lowercase, **D** = Descenders

## 6 dot/mm printhead

| Font | Matrix | Type | Character Size | | | |
|---|---|---|---|---|---|---|
| | HxW (in dots) | | HxW (in in.) | Char./in. | HxW (in mm) | Char. /mm |
| A | 9 x 5 | U-L-D | 0.059 x 0.039 | 25.4 | 1.50 x 0.99 | 1.01 |
| B | 11 x 7 | U | 0.072 x 0.059 | 16.9 | 1.82 x 1.50 | 0.066 |
| C, D | 18 x 10 | U-L-D | 0.118 x 0.079 | 12.7 | 2.99 x 2.00 | 0.05 |
| E | 21 x 10 | OCR-B | 0.138 x 0.085 | 11.7 | 3.50 x 2.16 | 0.46 |
| F | 26 x 13 | U-L-D | 0.170 x 0.105 | 9.53 | 4.32 x 2.67 | 0.37 |
| G | 60 x 40 | U-L-D | 0.394 x 0.315 | 3.18 | 10.0 x 8.00 | 0.125 |
| H | 17 x 11 | OCR-A | 0.111 x 0.098 | 10.2 | 2.81 x 2.48 | 0.40 |
| GS | 24 x 24 | SYMBOL | 0.157 x 0.157 | 6.35 | 3.98 x 3.98 | 0.251 |
| 0 | Default: 15 x 12 | | | | | |

## 8 dot/mm (203 dpi) printhead

| Font | Matrix | Type | Character Size | | | |
|---|---|---|---|---|---|---|
| | HxW (in dots) | | HxW (in in.) | Char./in. | HxW (in mm) | Char. /mm |
| A | 9 X 5 | U-L-D | 0.044 x 0.030 | 33.3 | 1.12 x 0.76 | 1.31 |
| B | 11 X 7 | U | 0.054 x 0.044 | 22.7 | 1.37 x 1.12 | 0.89 |
| C, D | 18 X 10 | U-L-D | 0.089 x 0.059 | 16.9 | 2.26 x 1.12 | 0.66 |
| E | 28 x 15 | OCR-B | 0.138 x 0.098 | 10.2 | 3.50 x 2.49 | 0.40 |
| F | 26 x 13 | U-L-D | 0.128 x 0.079 | 12.7 | 3.25 x 2.00 | 0.50 |
| G | 60 x 40 | U-L-D | 0.295 x 0.197 | 4.2 | 7.49 x 5.00 | 0.167 |
| H | 21 x 13 | OCR-A | 0.103 x 0.093 | 10.8 | 2.61 x 2.36 | 0.423 |
| GS | 24 x 24 | SYMBOL | 0.118 x 0.118 | 8.5 | 2.99 x 2.99 | 0.334 |
| P | 20 x 18 | U-L-D | 0.098 x 0.089 | N/A | 2.50 x 2.25 | N/A |
| Q | 28 x 24 | U-L-D | 0.138 x 0.118 | N/A | 3.50 x 3.00 | N/A |
| R | 35 x 31 | U-L-D | 0.172 x 0.153 | N/A | 4.38 x 3.88 | N/A |
| S | 40 x 35 | U-L-D | 0.197 x 0.172 | N/A | 5.00 x 4.38 | N/A |
| T | 48 x 42 | U-L-D | 0.236 x 0.207 | N/A | 6.00 x 5.25 | N/A |
| U | 59 x 53 | U-L-D | 0.290 x 0.261 | N/A | 7.38 x 6.63 | N/A |
| V | 80 x 71 | U-L-D | 0.394 x 0.349 | N/A | 10.00 x 8.88 | N/A |
| 0 | Default: 15 x 12 | U-L-D | Scalable | | Scalable | |

## 12 dot/mm (300 dpi) printhead

| Font | Matrix | Type | Character Size | | | |
|---|---|---|---|---|---|---|
| | HxW (in dots) | | HxW (in in.) | Char./in. | HxW (in mm) | Char. /mm |
| A | 9 X 5 | U-L-D | 0.030 x 0.020 | 50.8 | 0.75 x 0.50 | 2.02 |
| B | 11 X 7 | U | 0.036 x 0.030 | 33.8 | 0.91 x 0.75 | 1.32 |
| C, D | 18 X 10 | U-L-D | 0.059 x 0.040 | 25.4 | 1.50 x 1.00 | 1.00 |
| E | 42 x 20 | OCR-B | 0.138 x 0.085 | 23.4 | 1.75 x 1.08 | 0.92 |
| F | 26 x 13 | U-L-D | 0.085 x 0.053 | 19.06 | 2.16 x 1.34 | 0.74 |
| G | 60 x 40 | U-L-D | 0.197 x 0.158 | 6.36 | 5.00 x 4.00 | 0.25 |
| H | 34 x 22 | OCR-A | 0.111 x 0.098 | 10.20 | 2.81 x 2.48 | 0.40 |
| GS | 24 x 24 | SYMBOL | 0.079 x 0.079 | 12.70 | 1.99 x 1.99 | 0.52 |
| P | 20 x 18 | U-L-D | 0.067 x 0.060 | N/A | 1.69 x 1.52 | N/A |
| Q | 28 x 24 | U-L-D | 0.093 x 0.080 | N/A | 2.37 x 2.03 | N/A |
| R | 35 x 31 | U-L-D | 0.117 x 0.103 | N/A | 2.96 x 2.62 | N/A |
| S | 40 x 35 | U-L-D | 0.133 x 0.177 | N/A | 3.39 x 2.96 | N/A |
| T | 48 x 42 | U-L-D | 0.160 x 0.140 | N/A | 4.06 x 3.56 | N/A |
| U | 59 x 53 | U-L-D | 0.197 x 0.177 | N/A | 5.00 x 4.49 | N/A |
| V | 80 x 71 | U-L-D | 0.267 x 0.237 | N/A | 6.77 x 6.01 | N/A |
| 0 | Default: 15 x 12 | U-L-D | Scalable | | Scalable | |

## 24 dot/mm (600 dpi) printhead

| Font | Matrix | Type | Character Size | | | |
|---|---|---|---|---|---|---|
| | HxW (in dots) | | HxW (in in.) | Char./in. | HxW (in mm) | Char. /mm |
| A | 9 X 5 | U-L-D | 0.015 x 0.010 | 100.00 | 0.38 x 0.25 | 4.00 |
| B | 11 X 7 | U | 0.018 x 0.015 | 66.66 | 0.46 x 0.38 | 2.60 |
| C, D | 18 X 10 | U-L-D | 0.030 x 0.020 | 50.00 | 0.77 x 0.51 | 2.0 |
| E | 42 x 20 | OCR-B | 0.137 x 0.087 | 11.54 | 3.47 x 2.20 | 0.45 |
| F | 26 x 13 | U-L-D | 0.043 x 0.027 | 37.5 | 1.10 x 0.68 | 1.50 |
| G | 60 x 40 | U-L-D | 0.100 x 0.080 | 12.50 | 2.54 x 2.04 | 0.50 |
| H | 34 x 22 | OCR-A | 0.100 x 0.093 | 10.71 | 2.54 x 2.37 | 0.42 |
| GS | 24 x 24 | SYMBOL | 0.040 x 0.040 | 25.00 | 1.02 x 1.02 | 1.00 |
| P | 20 x 18 | U-L-D | 0.067 x 0.060 | N/A | 1.69 x 1.52 | N/A |
| Q | 28 x 24 | U-L-D | 0.093 x 0.080 | N/A | 2.37 x 2.03 | N/A |
| R | 35 x 31 | U-L-D | 0.117 x 0.103 | N/A | 2.96 x 2.62 | N/A |
| S | 40 x 35 | U-L-D | 0.133 x 0.117 | N/A | 3.39 x 2.96 | N/A |
| T | 48 x 42 | U-L-D | 0.160 x 0.140 | N/A | 4.06 x 3.56 | N/A |
| U | 59 x 53 | U-L-D | 0.197 x 0.177 | N/A | 5.00 x 4.49 | N/A |
| V | 80 x 71 | U-L-D | 0.267 x 0.237 | N/A | 6.77 x 6.01 | N/A |
| 0 | Default: 15 x 12 | U-L-D | Scalable | | Scalable | |

# Bar Codes

Every bar code contains data made up of a sequence of light spaces and dark bars that represent letters, numbers, or other graphic characters. The usable characters differ among the various kinds of bar codes. Each bar code section in the Command Reference provides a table of applicable characters. Start and stop characters and check digits are used by many, but not all, bar codes. These will be indicated in the specific bar code explanations.

Zebra printers can print the following kinds of bar codes:

| **Bar code modulus "X" dimensions** | **Linear bar codes** |
|---|---|
| • Picket fence (non-rotated) orientation:<br>  • 203 dpi = 0.0049 in. mil to 0.049 in.<br>  • 300 dpi = 0.0033 in. mil to 0.033 in.<br>• Ladder (rotated) orientation:<br>  • 203 dpi = 0.0049 in. mil to 0.049 in.<br>  • 300 dpi = 0.0039 in. mil to 0.039 in. | • Code 11<br>• Code 39<br>• Code 93<br>• Code 128 with subsets A/B C and UCC Case Codes<br>• ISBT-128<br>• UPC-A |
| **Two-dimensional bar codes**<br>• Code 49<br>• Maxi Code<br>• PDF-417<br>• QR Code<br>• Codablock<br>• DataMatrix<br>• Micro-PDF417 | • UPC-E<br>• EAN-8<br>• EAN-13<br>• UPC and EAN 2 or 5 digit extensions<br>• Plessey<br>• Postnet<br>• Standard 2 of 5<br>• Industrial 2 of 5 |
| **Bar code ratios**<br>• 2:1<br>• 7:3<br>• 5:2<br>• 3:1 | • Interleaved 2 of 5<br>• LOGMARS<br>• MSI<br>• Codabar<br>• RSS-14 |

## Basic Format for Bar Codes

The basic format for bar codes is quiet zone, start character, data, check digit, stop character, and quiet zone. Not all bar codes require each of these elements.

Every bar code requires a quiet zone. A quiet zone (sometimes called a "clear area") is an area adjacent to the machine-readable symbols that ensure proper reading (decoding) of the symbols. No printing is permissible within this area. Preprinted characters, borders, and background color are acceptable if they are invisible to the reading device; these are used in some applications but restrict the type of reading device that can be used. The size of the quiet zone depends on the size of bar widths (usually 10 times the width of the narrow bar).

**Figure 7 • Quiet Zone in a Bar Code**


## Bar Code Field Instructions

To create a bar code, a bar code field command must be contained in the label format. Table 18 shows the bar code field commands. The number in brackets denotes the print ratio. Each command produces a unique bar code.

**Important •** (*) for Fixed Printing Ratio means that the ratio between the width of the bars in the code is a fixed standard and cannot be changed.

**Table 18 • Bar Code Field Commands**

| | | |
|---|---|---|
| **^B1** | Code 11 (USD-8) | [2.0 - 3.0] |
| **^B2** | Interleaved 2 of 5 | [2.0 - 3.0] |
| **^B3** | Code 39 (USD-3 & 3 of 9) | [2.0 - 3.0] |
| **^B4** | Code 49 (*) | [Fixed] |
| **^B7** | PDF417 (*) | [Fixed] |
| **^B8** | EAN-8 (*) | [Fixed] |
| **^B9** | UPC-E | [Fixed] |
| **^BA** | Code 93 (USS-93)(*) | [Fixed] |
| **^BB** | CODABLOCK A, E, F (*) | [Fixed] |
| **^BC** | Code 128 (USD-6) (*) | [Fixed] |
| **^BD** | UPS MaxiCode (*) | [Fixed] |
| **^BE** | EAN-13 | [Fixed] |
| **^BF** | Micro-PDF417 | [Fixed] |
| **^BI** | Industrial 2 of 5 | [2.0 - 3.0] |
| **^BJ** | Standard 2 of 5 | [2.0 - 3.0] |
| **^BK** | ANSI Codabar (USD-4 & 2 of 7) | [2.0 - 3.0] |
| **^BL** | LOGMARS | [2.0 - 3.0] |

**Table 18 • Bar Code Field Commands (Continued)**

| | | |
|---|---|---|
| **^BM** | MSI | [2.0 - 3.0] |
| **^BP** | Plessey | [2.0 - 3.0] |
| **^BQ** | QR Code (*) | [Fixed] |
| ^BS | UPC/EAN Extensions (*) | [Fixed] |
| ^BU | UPC-A (*) | [Fixed] |
| ^BX | Data Matrix (*) | [Fixed] |
| ^BZ | PostNet (*) | [Fixed] |

Additionally, each bar code field command can be issued with a definition parameter string. The parameter string defines field rotation, height, and interpretation line status for all bar codes. For some bar codes, the parameter string also sets a check digit, start character, and/or stop character. Use the definition parameter string to command the printer to print bar codes of appropriate heights and densities that conform to the specifications of the application.

The use of the parameter string is optional because all parameters have default values. If the default values for all of the bar code parameters suit the application, then only the bar code command needs to be entered.

Parameters in bar code field commands are "position specific." If a value (other than the default value) is manually entered for one parameter, a comma **,** the ZPL II delimiter character, must be used to mark the position of the preceding parameters in the string.

To change just the third parameter, enter two commas and then the value for the third parameter. The default values will be automatically used for the first and second parameters.

In the *ZPL II Programming Guide Volume Two*, see the Saving Label Formats as Graphic Images topic in the Programming Exercises chapter, the ^B3 bar code field command has five parameters. The third parameter defines the height of the bar in dots. The bar code is to be printed using default values for the first two parameters EXCEPT the height of the bar. This is to be 20 dots. Finally, the "N" indicates that a print interpretation line will not print with the bar code. The command would be entered as follows:

```
^B3,,20,N
```

Delimiters (commas) are not required for parameters between a manually entered value and the end of the parameter string.

## Bar Code Command Groups

Bar code commands are organized into four groups. Each group represents a type of bar code. Table 19 through Table 22 identify the groups and the bar codes they contain:

**Table 19 • Numeric Only Bar Codes**

| | |
|---|---|
| **^B1** | Code 11 |
| **^B2** | Interleaved 2 of 5 |
| **^BI** | Industrial 2 of 5 |

**Table 19 • Numeric Only Bar Codes**

| | |
|---|---|
| **^BJ** | Standard 2 of 5 |
| **^BK** | ANSI Codabar (or NW-7) |
| **^BM** | MSI |
| **^BP** | Plessey |
| **^BZ** | POSTNET |

**Table 20 • Retail Labeling Bar Codes**

| | |
|---|---|
| **^B8** | EAN-8 |
| **^B9** | UPC-E |
| **^BE** | EAM-13 |
| **^BS** | UPC/EAN extensions |
| **^BU** | UPC-A |

**Table 21 • Alphanumeric Bar Code**

| | |
|---|---|
| **^B3** | Code 39 |
| **^BA** | Code 93 |
| **^BC** | Code 128 |
| **^BL** | LOGMARS |

**Table 22 • Two-Dimensional Bar Codes**

| | |
|---|---|
| **^B4** | Code 49 |
| **^B7** | PDF417 |
| **^BB** | CODABLOCK |
| **^BD** | UPS MaxiCode |
| **^BF** | MicroPDF417 |
| **^BQ** | QR Code |
| **^BX** | Data Matrix |

**Notes •**

# D

# Mod 10 and Mod 43 Check Digits

This section provides information about Mod 10 and Mod 43 check digits.

**Contents**

# Mod 10 Check Digit

The calculations for determining the Mod 10 Check Digit character are as follows:

1. Start at the first position and add the value of every other position together.

   0 + 2 + 4 + 6 + 8 + 0 = 20

2. The result of Step 1 is multiplied by 3.

   20 x 3 = 60

3. Start at the second position and add the value of every other position together.

   1 + 3 + 5 + 7 + 9 = 25

4. The results of steps 1 and 3 are added together.

   60 + 25 = 85

5. The check character (12th character) is the smallest number which, when added to the result in step 4, produces a multiple of 10.

   85 + X = 90 (next higher multiple of 10)

   X = 5 Check Character

This bar code illustrates the above example. The digit on the right (5) is the check digit.

0 12345 67890 5

# Mod 43 Check Digit

The calculations for determining the Mod 43 check Digit character are as follows:

Each character in the Code 39 character set has a specific value, as follows:

| | | | |
|---|---|---|---|
| 0=0 | B=11 | M=22 | X=33 |
| 1=1 | C=12 | N=23 | Y=34 |
| 2=2 | D=13 | O=24 | Z=35 |
| 3=3 | E=14 | P=25 | - =36 |
| 4=4 | F=15 | Q=26 | . = 37 |
| 5=5 | G=16 | R=27 | Space=38 |
| 6=6 | H=17 | S=28 | $=39 |
| 7=7 | I=18 | T=29 | /=40 |
| 8=8 | J=19 | U=30 | +=41 |
| 9=9 | K=20 | V=31 | %=42 |
| A=10 | L=21 | W=32 | |

**Example** • Data string 2345ABCDE/

1. Add the sum of all the character values in the data string. Using the chart above, the sum of the character values is as follows:

   1 + 2 + 3 + 4 + 5 + 10 + 11 + 12 + 13 + 14 + 40 =115

2. Divide the total by 43. Keep track of the remainder.

   115/43 = 2 Remainder is 29

3. The “check digit” is the character that corresponds to the value of the remainder.

   Remainder = 29

   29 is the value for the letter T.

   T is the check digit.

Below is a bar code that illustrates the example. The character on the right, T, is the check digit.

*12345ABCDE/T*

```
^FO125,100^B3N,Y,150,Y,N^FD12345ABCDE/^FS
```

**Notes •**

# E

# ZB64 Encoding and Compression

This section describes the Base 64 MIME (ZB64) encoding and compression. This is the same type of MIME encoding that is used in e-mail.

## Contents

# Introduction to B64 and Z64

The first encoding, known as B64, encodes the data using the MIME Base64 scheme. Base64 is used to encode e-mail attachments and is specifically designed to address communications path limitations, such as control characters and 7-bit data links. It encodes the data using only the printable ASCII characters:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h I j k l m n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9 + / =

With the use of ZPL, this has the added benefit of avoiding the caret (^) and tilde (~) characters. Base64 encodes six bits to the byte, for an expansion of 33 percent over the un-enclosed data. This is much better than the 100 percent expansion given by the existing ASCII hexadecimal encoding.

The second encoding, known as Z64, first compresses the data using the LZ77 algorithm to reduce its size. (This algorithm is used by the PKWARE® compression program PKZIP™ and is integral to the PNG graphics format.) The compressed data is then encoded using the MIME Base64 scheme as described above.

A CRC is calculated across the Base64-encoded data. If the CRC-check fails or the download is aborted, the object can be invalidated by the printer.

The robust encodings can be piggybacked on the existing download commands with full backward compatibility. This is done by prefacing the new encodings with a header that uniquely identifies them. The download routines in the printer firmware can key-off the header to determine whether the data is in the old ASCII hexadecimal encoding or one of the new encodings. This allows existing downloadable objects to be used in their present format, while new objects can be created using the same download commands with the new encodings for increased integrity and reduced download times.

For easy reference, B64 and Z64 are referred to as ZB64. In any reference to the ZB64 encoding, assume that both Base64-only (B64) and LZ77/Base64 (Z64) encodings are accepted.

**Example •** The following is an example of an existing download command using the new encoding:

```
~DTARIAL,59494,:Z64:H4sICMB8+DMAC0FSSUFMLlRURgDsmnd8VEW7x5+ZO
edsyibZNNJhlyWhbEJIwYSwJDGNkmwghJIgJYEEEhQIPSggKAjEAiIiVaSoIJ
YNBAkIGgGxUBVUUCGU0JQSC0WFnPvbE+SF18+9H+8f973X+3Jm93umzzNznvn
NSSFGRJ6ARAVZvXK7XDaXLyTiR5B7ontuZPQ824I5RKIa6ew+aba8+pU1rVDZ
iciv
```

[multiple lines deleted]

```
/O6DU5wZ7ie2+g4xzDPwCpwm3nqW2GAPcdclxF4fIP66jHjncmKvKzh/ZUNCx
l9/QQx2HXHYB4m/PkQcdCdx2G7OYt+mszkMh4iZxoifvkh89BFipo87kwD/Bf
/dOcycAAEA:a1b2
```

The parameters are identical to the existing `~DT` command:

**Table 23 • ~DT Command Parameters**

| Parameter | Details |
|---|---|
| `o` = font name | *Accepted Values:* any valid TrueType name, up to 8 characters<br>*Default Value:* if a name is not specified, UNKNOWN is used<br>In this example, Arial is the specified font. |
| `s` = font size | *Accepted Values:* the number of memory bytes required to hold the Zebra-downloadable format of the font<br>*Default Value:* if an incorrect value or no value is entered, the command is ignored<br>In this example, 59494 is the size.<br>To maintain compatibility with the existing ASCII hexadecimal encoding, this field must contain the size of the un-enclosed and uncompressed object — the number of bytes that are finally placed into the printer's memory, not the number of bytes downloaded. |
| `data` = data string | *Accepted Values:* a string of ASCII hexadecimal values (two hexadecimal digits/byte). The total number of two-digit values must match parameter `s`.<br>*Default Value:* if no data is entered, the command is ignored<br>Everything following the size field is data. The new encoding imposes a header with a unique signature. The new encoding must start with the characters :B64: (data encoded in Base-64 only) or :Z64: (data compressed with LZ77, then encoded in Base-64) followed by the encoded data.<br>After the data is presented, another colon (:) and four hexadecimal digits comprise the CRC. The Base64 standard allows new-line characters (carriage returns and line feeds) to be inserted into the encoded data for clarity. These characters are ignored by the printer. |

When downloading graphics, the colon is used in the current ASCII hexadecimal encoding indicate "repeat the previous dot row." Since this shorthand is invalid for the first character of data (no previous dot row has been downloaded), it is safe for the printer to detect the leading colon character as the lead-in for the new encodings.

# B64 and Z64 Encoding

These download encodings, B64 and Z64, are created as drop-in replacements for the existing ASCII hexadecimal encoding.

B64 encoding do the following:

- Encode the compressed data using the MIME Base64 algorithm.
- Calculate a CRC across the encoded data.
- Add a unique header to differentiate the new format from the existing ASCII hex encoding.

Z64 encoding do the following:

- Compress the data using the LZ77 algorithm.
- Encode the compressed data using the MIME Base64 algorithm.
- Calculate a CRC across the encoded data.
- Add a unique header to differentiate the new format from the existing ASCII hexadecimal encoding.

The data field have this format:

**:id:encoded_data:crc**

This table identifies the parameters for this format:

**Table 24 • Format Parameters**

| Parameter | Details |
|---|---|
| **:id** | the identifying string B64 or Z64 |
| **:iencoded_data** | data to download, compressed with LZ77 (if the id parameter is set to Z64) and encoded with Base64. |
| **:crc** | four hexadecimal digits representing the CRC calculated over the :encoded_data field. |

The printer calculates a CRC across the received data bytes and compare this to the CRC in the header. A CRC mismatch is treated as an aborted download.

The B64 and Z64 encodings can be used in place of the ASCII hexadecimal encoding in any download command. The commands are:

**~DB** – Download Bitmap Font

**~DE** – Download Encoding

**~DG** – Download Graphic

**~DL** – Download Unicode Bitmap Font

**~DS** – Download Scalable Font

**~DT** – Download TrueType Font

**~DU** – Download Unbounded TrueType Font

**^GF** – Graphic Field (with compression type set to “ASCII hex”)

The ~DB (Download Bitmap Font) command can use the new encodings in place of the ASCII hexadecimal encoding in data sub-fields. Each character is encoded individually. However, for small amounts of data, the identifying B64 or Z64 header and trailing CRC may negate any gains made by using the new format.

For backward compatibility, the ^HG (Host Graphic) command uses the ASCII hexadecimal encoding. It does not use the new encodings.

**Notes •**

# F

# Field Interactions

This section provides you with examples that show how commands interact with various justification parameters. The examples are in charts for these orientations:

- Normal
- Rotated
- Bottom-up
- Inverted

These charts are designed so that you can identify the location of the field origin and interactions between the rotation, formatting and justification commands.

# Normal Orientation

Table 25 shows you the various normal orientation outputs:

**Table 25 • Normal Orientation Examples**

| | ^FPH | ^FPV | ^FPR |
|---|---|---|---|
| ^FO Left Justified |   |   |   |
| ^FT Left Justified |   |   |   |
| ^FO Right Justified |   |   |   |
| ^FT Right Justified |   |   |   |

# Rotated Orientation

Table 26 shows you the various rotated orientation outputs:

**Table 26 • Rotated Orientation Examples**

| | ^FPH | ^FPV | ^FPR |
|---|---|---|---|
| ^FO Left Justified |  |  |  |
| ^FT Left Justified |  |  |  |
| ^FO Right Justified |  |  |  |
| ^FT Right Justified |  |  |  |

# Bottom Up Orientation

Table 27 shows you the various bottom up orientation outputs:

**Table 27 • Bottom Orientation Examples**

| | ^FPH | ^FPV | ^FPR |
|---|---|---|---|
| **^FO Left Justified** | ABCDE | A B C D E | EDCBA |
| **^FT Left Justified** | ABCDE | A B C D E | EDCBA |
| **^FO Right Justified** | ABCDE | A B C D E | EDCBA |
| **^FT Right Justified** | ABCDE | A B C D E | EDCBA |

# Inverted Orientation

Table 28 shows you the various inverted orientation outputs:

**Table 28 • Inverted Orientation Examples**

| | ^FPH | ^FPV | ^FPR |
|---|---|---|---|
| **^FO Left Justified** |  |  |  |
| **^FT Left Justified** |  |  |  |
| **^FO Right Justified** |  |  |  |
| **^FT Right Justified** |  |  |  |

**Notes •**

# Glossary

This is a glossary of terms.

# Glossary of Terms

**ASCII** American Standard Code for Information Interchange. A 7-bit character set that includes Latin characters, as well as control and punctuation characters.

**bidirectional text layout** The arrangement of characters with mixed directionality on a single line. For example, English and Arabic.

**big-endian** In the encoding, the most significant byte is stored first.

**byte order mark** BOM; indicates big-endian (BE) or little-endian (LE).

**character** The smallest component of a writing system that has understanding, refers to the abstract meaning not the specific shape.

**character code** Another term for code point.

**character set** A collection of characters.

**coded character set** An ordered collection of characters that are assigned an integral reference number.

**code point** An integral reference number assigned to a character.

**coded character set** An ordered collection of characters that are assigned an integral reference number.

**combining semantic cluster** Consists of an atomic character, a combining character sequence consisting of a base character plus one or more nonspacing marks, enclosing marks, combining word joiner, such as an Indic virama, or a sequence of Hangul jamos equivalent to a Hangual syllable. .

**diacritic** A mark that is added to a letter or other character to change its value or meaning.

**encoding** The mapping of a characters code points to a sequence of bytes used to represent the data.

**font** An electronic collection of glyphs used for the visual representation of characters.

**GB18030** A standard required by the People's Republic of China for operating systems of non-handheld computers.

**GB 2312** A Chinese character set and encoding.

**glyph** The visual representation of a character, such as a shape or image.

**grapheme cluster** Consists of an atomic character, a combining character sequence consisting of a base character plus one or more nonspacing marks or enclosing marks, or a sequence of Hangul jamos equivalent to a Hangul syllable.

**ISO 10646** An international standard that defines the Universal Character Set (UCS) and a character encoding. The UCS matches Unicode.

**ISO 8859-1** An encoding standard for representing Western European languages using the Latin Alphabet.

**language** A system of sounds and scripts used to represent and communicate concepts, ideas, meanings, and thoughts.

**ligature** A glyph representing a combination of two or more characters.

**little-endian** In the encoding, the least significant byte is stored first.

**open type** A file format for scalable font files that extends the existing TrueType font file format used by Microsoft Windows and Apple Macintosh operating systems.OpenType tables support advanced layout features.

**presentation form** A pre-combined character, ligature or variant glyph that has a separate code point; used for compatibility.

**script** A collection of symbols used to represent textual information in one or more writing systems.

**Shift-JIS** A shifted encoding of the Japanese character encoding standard, JIS X 0208, heavily deployed in PCs.

**True type** An outline font format that is scalable without quality loss.

**Unicode** The universal character set and encoding maintained by the Unicode Consortium.

**UTF-8** The Unicode encoding that assigns each character code point to a sequence of one to four bytes.

**UTF-16** A Unicode encoding form that represents Unicode code point values in the BMP with 16-bit code units and Unicode code point values in the supplementary planes with pairs of 16-bit code units.

**UTF-16BE** A Unicode encoding scheme based on UTF-16 that serializes the bytes in each code unit in big-endian order.

**UTF-16LE** A Unicode encoding scheme based on UTF-16 that serializes the bytes in each code unit in little-endian order.

**Notes •**

# Index

## Symbols



## A



## B



## C

## D



## E



## F

## L



## M



## N



## O



## P

**Zebra Technologies Corporation**
333 Corporate Woods Parkway
Vernon Hills, Illinois 60061.3109 U.S.A.
Telephone: +1 847 793 2600
Facsimile: +1 847 913 8766

**Zebra Technologies Europe Limited**
Zebra House
The Valley Centre, Gordon Road
High Wycombe
Buckinghamshire HP13 6EQ, UK
Telephone: +44 (0) 1494 472872
Facsimile: +44 (0) 1494 450103

**Zebra Technologies Asia Pacific, LLC**
16 New Industrial Road
#05-03 Hudson TechnoCentre
Singapore 536204
Telephone: +65 6858 0722
Facsimile: +65 6885 0838

Part Number: 45541L-004 Rev. A

Change variables below to match your book, and set conditional text settings so only the appropriate tag(s) shows in Table 1. Okay to import variables, conditional text, paragraph tags, character tags, page layouts, table formats, color definitions, cross-reference formats, and reference pages from this file to create most VH TechPubs books. Some differences may be required for Maintenance Manuals

13979L-001 - x.14 and greater

45541L-002 - x.1- through x.13..

**Part #:** 45541L-004

**Revision Number:** Rev. A

**Printer Name:** ZPL II Programming

**Book Title:** Guide

**Firmware:** For x.10 through x.13 Firmware

**Type of printer (lower case):** printer

**Type of printer (initial caps):** Printer

**Company Name Short:** Zebra

**Company Name (initial caps):** Zebra Technologies

**Company Name (all caps):** ZEBRA TECHNOLOGIES

**Copyright:** © 2006 ZIH Corp.

**Language abbreviation:** EN-

**Table 1 • Conditional Text Tags**

| Printer | UG/PG | QRG | MM | Kits |
|---|---|---|---|---|
| 105SL | | | | |
| Brady 200 M-e | | | — | — |
| Brady MVP | | | — | — |
| Brady X-Plus II | | | — | — |
| PAX4 110 | | | | |
| PAX4 170 | | | | |
| RFID Programming Guide | | — | — | — |
| RPAX | | | | |
| RXi | | | | |
| RXiIIIPlus HF | | | — | — |
| RXMplus Europe | | | — | — |
| RXMplus U.S. | | | | |
| S4M | | | | |
| Tyco | | | — | — |
| UPS | — | | — | — |
| XiIIIPlus | | | | |
| Z4M/Z6M | | | | |
| Z4Mplus/Z6Mplus | | | | |
| Z4Mplus DT | | | | |
| ZPL II Programming Guide | | — | — | — |
| ZPL x.13 | ZPL | — | — | — |
| ZPL x.14 -Pinon | ZPL | — | — | — |
| ZPL x.15 - Tarkus | ZPL | — | — | — |