

# ZPL II<sup>®</sup> Programming Guide

Volume One



Copyright 2003 ZIH Corp. All rights reserved.

The copyrights in this manual and the label printer described therein are owned by Zebra Technologies. All rights are reserved. Unauthorized reproduction of this manual or the software in the label printer may result in imprisonment of up to one year and fines of up to \$10,000 (17 U.S.C.506). Copyright violators may be subject to civil liability.

All trademarks and registered trademarks are property of their respective owners.

**Document Name: 45541LB-R3.pdf**

### **Proprietary Statement**

This manual contains proprietary information of Zebra Technologies Corporation and its subsidiaries ("Zebra Technologies"). It is intended solely for the information and use of parties operating and maintaining the equipment described herein. Such proprietary information may not be used, reproduced, or disclosed to any other parties for any other purpose without the expressed written permission of Zebra Technologies.

### **Product Improvements**

Continuous improvement of products is a policy of Zebra Technologies. All specifications and designs are subject to change without notice.

### **Liability Disclaimer**

Zebra Technologies takes steps to assure that its published Engineering specifications and manuals are correct; however, errors do occur. Zebra Technologies reserves the right to correct any such errors and disclaims liability resulting therefrom.

### **Limitation of Liability**

In no event shall Zebra Technologies or anyone else involved in the creation, production or delivery of the accompanying product (including hardware and software) be liable for any damages whatsoever (including, without limitation, consequential damages including loss of business profits, business interruption or loss of business information) arising out of the use of or the results of use of or inability to use such product, even if Zebra Technologies has been advised of the possibility of such damages. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

### **Copyrights**

The copyrights in this manual and the label printer described therein are owned by Zebra Technologies. Unauthorized reproduction of this manual or the software in the label printer may result in imprisonment of up to one year and fines of up to \$10,000 (17 U.S.C.506). Copyright violators may be subject to civil liability.

©2003 ZIH Corp. All trademarks and registered trademarks are property of their respective owners. All rights reserved.



Proprietary Statement

# Table of Contents

<b>Functional Table of Contents</b> .....	<b>xi</b>
<b>Preface</b> .....	<b>xvii</b>
Contacts .....	<b>xviii</b>
Support .....	<b>xviii</b>
About this Document .....	<b>xix</b>
Document Conventions .....	<b>xx</b>
Related Documents .....	<b>xxi</b>
.....	<b>xxii</b>
<b>Introduction</b> .....	<b>1</b>
.....	<b>2</b>
<b>ZPL II Commands</b> .....	<b>3</b>
Basic ZPL Exercises .....	<b>5</b>
Before you begin .....	<b>5</b>
^A .....	<b>13</b>
^A@ .....	<b>17</b>
^B1 .....	<b>20</b>
^B2 .....	<b>22</b>
^B3 .....	<b>24</b>
^B4 .....	<b>29</b>

<b>^B5</b>	<b>35</b>
<b>^B7</b>	<b>37</b>
<b>^B8</b>	<b>43</b>
<b>^B9</b>	<b>45</b>
<b>^BA</b>	<b>48</b>
<b>^BB</b>	<b>53</b>
<b>^BC</b>	<b>57</b>
<b>^BD</b>	<b>64</b>
<b>^BE</b>	<b>68</b>
<b>^BF</b>	<b>70</b>
<b>^BI</b>	<b>73</b>
<b>^BJ</b>	<b>75</b>
<b>^BK</b>	<b>77</b>
<b>^BL</b>	<b>79</b>
<b>^BM</b>	<b>81</b>
<b>^BP</b>	<b>84</b>
<b>^BQ</b>	<b>86</b>
<b>^BR</b>	<b>93</b>
<b>^BS</b>	<b>95</b>
<b>^BT</b>	<b>98</b>
<b>^BU</b>	<b>101</b>
<b>^BX</b>	<b>103</b>
<b>^BY</b>	<b>108</b>
<b>^BZ</b>	<b>110</b>
<b>^CC</b>	<b>112</b>
<b>~CC</b>	<b>113</b>
<b>^CD ~CD</b>	<b>114</b>
<b>^CF</b>	<b>115</b>
<b>^CI</b>	<b>117</b>
<b>^CM</b>	<b>121</b>
<b>^CO</b>	<b>123</b>
<b>^CT ~CT</b>	<b>126</b>
<b>^CV</b>	<b>127</b>
<b>^CW</b>	<b>130</b>
<b>~DB</b>	<b>132</b>
<b>^DE</b>	<b>135</b>

^DF	137
~DG	139
~DN	142
~DS	143
~DT	145
~DU	146
~DY	148
~EF	150
~EG	151
^FB	152
^FC	155
^FD	157
^FH	158
^FM	160
^FN	164
^FO	165
^FP	166
^FR	168
^FS	169
^FT	170
^FV	174
^FW	176
^FX	177
^GB	178
^GC	180
^GD	181
^GE	183
^GF	184
^GS	187
~HB	189
~HD	190
^HF	191
^HG	192
^HH	193
~HI	194
~HM	195

~HS	196
~HU	199
^HW	200
^HY	202
^HZ	203
^ID	205
^IL	207
^IM	209
^IS	211
~JA	213
^JB	214
~JB	215
~JC	216
~JD	217
~JE	218
~JF	219
~JG	221
^JI	222
~JI	224
^JJ	225
~JL	228
^JM	229
~JN	231
~JO	232
~JP	233
~JQ	234
~JR	235
^JS	236
~JS	238
^JT	240
~JU	242
^JW	243
~JX	244
^JZ	245
~KB	246
^KD	247



^KL	248
^KN	249
^KP	251
^LH	252
^LL	254
^LR	256
^LS	258
^LT	259
^MC	260
^MD	261
^MF	263
^ML	264
^MM	265
^MN	267
^MP	268
^MT	270
^MU	271
^MW	273
~NC	274
^NI	275
~NR	276
^NS	277
~NT	278
^PF	279
^PF ~PF	280
^PM	281
^PO	282
^PP ~PP	284
^PQ	285
^PR	287
~PR	290
~PS	291
^PW	292
~RO	293
^SC	295
~SD	297



^SE ..... 298  
 ^SF ..... 299  
 ^SL ..... 302  
 ^SN ..... 304  
 ^SO ..... 307  
 ^SP ..... 308  
 ^SQ ..... 310  
 ^SR ..... 313  
 ^SS ..... 314  
 ^ST ..... 316  
 ^SX ..... 317  
 ^SZ ..... 320  
 ~TA ..... 321  
 ^TO ..... 322  
 ~WC ..... 325  
 ^WD ..... 327  
 ^XA ..... 329  
 ^XB ..... 330  
 ^XF ..... 331  
 ^XG ..... 333  
 ^XZ ..... 335  
 ^ZZ ..... 336

**ZBI Commands ..... 337**

AND ..... 339  
 Arrays ..... 339  
 AUTONUM ..... 339  
 Boolean Expression ..... 340  
 BREAK ..... 341  
 Channels ..... 341  
 CLOSE ..... 341  
 CLRERR ..... 342  
 CTRL-C ..... 342  
 DEBUG ..... 342  
 Declaration ..... 343  
 DECLARE ..... 343  
 DELETE ..... 344  
 DIR ..... 344

DO-LOOP	345
ECHO	346
END	347
! (EXCLAMATION MARK)	347
EXIT	348
FOR-LOOP	348
Functions	349
Integer Functions	349
String Functions	356
GOTO	360
GOSUB-RETURN	361
IF Statements	362
INBYTE	363
INPUT	364
LET	365
LIST	366
LOAD	366
NEW	367
NOT	367
Numeric Expressions	368
ON ERROR	369
Open	370
OR	371
OUTBYTE	371
Ports <CHANNEL EXPRESSION>	372
PRINT	372
REM	373
RENUM	373
RESTART	374
RUN	374
SEARCHTO\$ (A,B\$)	375
SEARCHTO\$ (A,B\$,C)	376
SETERR	377
SLEEP	377
STEP	377
STORE	378
STRING CONCATENATION (&)	379
STRING VARIABLE	379
UB-STRINGS	380
TRACE	380



- ZBI — Tutorial . . . . . 383**
  - Starting ZBI . . . . . 384
  - ZBI Interpreter Modes . . . . . 385
  - Basic ZBI Examples . . . . . 387
- Index. . . . . 397**

# FUNCTIONAL TABLE OF CONTENTS

- Abort Download Graphic ..... 142
- ANSI Codabar Bar Code ..... 77
- Applicator Reprint ..... 290
- Bar Code Field Default ..... 108
- Battery Status ..... 189
- Cache On ..... 123
- Cancel All ..... 213
- Cancel Current Partially Input Format ..... 244
- Change Alphanumeric Default Font ..... 115
- Change Backfeed Sequence ..... 236
- Change Backfeed Sequence ..... 238
- Change Carets ..... 112
- Change Carets ..... 113
- Change Delimiter ..... 114
- Change International Font ..... 117
- Change Memory Letter Designation ..... 121
- Change Networking Settings ..... 277
- Change Tilde ..... 126
- CODABLOCK Bar Code ..... 53
- Code 11 Bar Code ..... 20
- Code 128 Bar Code (Subsets A, B, and C) ..... 57
- Code 39 Bar Code ..... 24
- Code 49 Bar Code ..... 29
- Code 93 Bar Code ..... 48

Code Validation	127
Comment	177
Configuration Label Return	193
Configuration Update	242
Data Matrix Bar Code	103
Define Language	248
Define Password	251
Define Printer Name	249
Disable Diagnostics	218
Display Description Information	203
Download Bitmap Font	132
Download Encoding	135
Download Format	137
Download Graphics	139
Download Graphics	148
Download Scalable Font	143
Download TrueType Font	145
Download Unbounded TrueType Font	146
EAN-13 Bar Code	68
EAN-8 Bar Code	43
Enable Communications Diagnostics	217
End Format	335
Erase Download Graphics	151
Erase Stored Formats	150
Field Block	152
Field Clock (for Real-Time Clock)	155
Field Data	157
Field Hexadecimal Indicator	158
Field Number	164
Field Orientation	176
Field Origin	165
Field Parameter	166
Field Reverse Print	168
Field Separator	169
Field Typeset	170
Field Variable	174

Font Identifier .....	130
Graphic Box .....	178
Graphic Circle .....	180
Graphic Diagonal Line .....	181
Graphic Ellipse .....	183
Graphic Field .....	184
Graphic Symbol .....	187
Graphic Symbol .....	191
Graphing Sensor Calibration .....	221
Halt ZebraNet Alert .....	310
Head Temperature Information .....	190
Head Test Fatal .....	231
Head Test Interval .....	240
Head Test Non fatal .....	232
Host Directory List .....	200
Host Graphic .....	192
Host Identification .....	194
Host RAM Status .....	195
Host Status Return .....	196
Image Load .....	207
Image Move .....	209
Image Save .....	211
Industrial 2 of 5 Bar Codes .....	73
Initialize Flash Memory .....	214
Interleaved 2 of 5 Bar Code .....	22
Kill Battery (Battery Discharge Mode) .....	246
Label Home .....	252
Label Length .....	254
Label Reverse Print .....	256
Label Shift .....	258
Label Top .....	259
LOGMARS Bar Code .....	79
Map Clear .....	260
Maximum Label Length .....	264
Media Darkness .....	261
Media Feed .....	263

Media Tracking .....	267
Media Type .....	270
Micro-PDF417 Bar Code .....	70
Mode Protection .....	268
Modify Heading Warning .....	273
MSI Bar Code .....	81
Multiple Field Origin Locations .....	160
Network Connect .....	274
Network ID Number .....	275
Object Delete .....	205
Pause and Cancel Format .....	233
PDF417 Bar Code .....	37
Planet Code bar code .....	35
Plessey Bar Code .....	84
POSTNET Bar Code .....	110
Power On Reset .....	235
Print Configuration Label .....	325
Print Directory Label .....	327
Print Mode .....	265
Print Orientation .....	282
Print Quantity .....	285
Print Rate .....	287
Print Start .....	291
Print Width .....	292
Printer Sleep .....	336
Printing Mirror Image of Label .....	281
Programmable Pause .....	284
QR Code Bar Code .....	86
Recall Format .....	331
Recall Graphic .....	333
Reprint After Error .....	245
Reset Advanced Counter .....	293
Reset Optional Memory .....	215
Return ZebraNet AlertConfiguration .....	199
RSS Bar Code .....	93
Scalable/Bitmapped Font .....	13



Select Date and Time Format (for Real Time Clock) .....	247
Select Encoding .....	298
Serialization Data .....	304
Serialization Field (with a Standard ^FD String) .....	299
Set All Network Printers Transparent .....	276
Set Auxiliary Port .....	225
Set Battery Condition .....	219
Set Currently Connected Printer Transparent .....	278
Set Darkness .....	297
Set Date and Time (for Real-Time Clock) .....	316
Set Dots per Millimeter .....	229
Set Label Length .....	228
Set Media Sensor Calibration .....	216
Set Media Sensors .....	314
Set Mode and Language (for Real-Time Clock) .....	302
Set Offset (for Real-Time Clock) .....	307
Set Printhead Resistance .....	313
Set Ribbon Tension .....	243
Set Serial Communications .....	295
Set Units of Measurement .....	271
Set ZebraNet Alert .....	317
Set ZPL .....	320
Slew Given Number of Dot Rows .....	279
Slew to Home Position .....	280
Standard 2 of 5 Bar Code .....	75
Start Format .....	329
Start Print .....	308
Start ZBI (Zebra BASIC Interpreter) .....	222
Start ZBI (Zebra BASIC Interpreter) .....	224
Suppress Backfeed .....	330
Tear-off Adjust Position .....	321
Terminate Zebra BASIC Interpreter .....	234
TLC39 bar code .....	98
Transfer Object .....	322
UPC/EAN Extensions .....	95

 **Table of Contents**

UPC-A Bar Code ..... 101  
UPC-E Bar Code ..... 45  
Upload Graphics ..... 202  
UPS MaxiCode Bar Code ..... 64  
Use Font Name to Call Font ..... 17

# Preface

The Preface discusses the topics and illustrates standards that are used throughout this guide.

## Contents

Contacts .....	<b>x</b>
About this Document.....	<b>xi</b>
Document Conventions.....	<b>xiii</b>
Related Documents.....	<b>xv</b>

## Contacts

You can contact Zebra at:

**Visit us at:** <http://www.zebra.com>

### **Our Mailing Addresses:**

#### **Zebra Technologies Corporation**

333 Corporate Woods Parkway  
Vernon Hills, Illinois 60061.3109 U.S.A  
Telephone +1 847.634.6700  
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

#### **Asia Pacific, LLC**

Bar Code & Card  
1 Sims Lane, #06-11  
Singapore 387355  
Telephone +6 6858 0722  
Facsimile +65 6885 0838

## Support

You can contact Zebra support with any questions or problems with Zebra Programming Language II (ZPL II):

**Support URL:** [http://www.zebra.com/SS/service\\_support.htm](http://www.zebra.com/SS/service_support.htm)

**Telephone:** +1.847.913.2259



# About this Document

The *ZPL II Programming Guide Volume One* consists of these chapters:

Title	Content Description
<i>Introduction</i>	Provides a high-level overview about this guide and Zebra Programming Language (ZPL).
<i>ZPL II Commands</i>	Provides an alphabetical, detailed description of each ZPL command.
<i>ZBI Commands</i>	Provides an alphabetical, detailed description of Zebra Basic Interpreter (ZBI).
<i>ZBI — Tutorial</i>	

# Document Conventions

These conventions are used throughout this document to convey certain information:

**About this Chapter Sections** These sections list and describe each main section of the chapter, including the initial page numbers of those sections. These sections primarily serve as hyperlink components for the Adobe Acrobat `.pdf` version of this guide.

**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.

**Commands** All commands appear in `Courier New` font. For example, all ZPL commands are displayed in `Courier New`.

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

These are the conventions and their meanings that you will see throughout this document.

**Important, Note, and Example** These topics are defined in this example:



**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.

## Related Documents

In addition to the *ZPL II® Programming Guide Volume One*, these documents might be helpful references:



**Note** • Most of these documents are available on our Web site.

- *The ZebraLink™ Solution Application white-paper*
- *ZBI Guide*
- *ZPL II® Programming Guide Volume Two: The X.10 Environment*





# 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.

**Note** • Firmware upgrades are available at: [www.zebra.com](http://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 that is earlier than X.10. Other commands have been redesigned and significantly enhanced to support innovations like:

- ZebraNet® ALERT
- Real-Time Clock
- Zebra BASIC Interpreter (ZBI)

Any word processor or text editor capable of creating ASCII-only files can be used to recreate the examples in this guide. Most of the examples are made up of a series of instruction lines. When you finish typing a line, press **Enter**. Continue this process for all of the lines in the example you are experimenting with.

To provide more information and convenient cross-referencing, commands that are directly related to features discussed in *Volume Two* have been noted under their Comments heading, pointing to the appendix or section that applies.



# ZPL II 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 (o, h, f, and g). The letters that follow `^B8` – o, h, f, and g – are 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

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.

- To write the ZPL files, use the DOS text editor.
- You can 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.
- That 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. All commands are not 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 using the following format as a model.



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

```
^XA
^FO50,50^ADN,36,20^xxxxxxxxxxFD^FS
^XZ
```

Send the above format to the printer.

^XA every format must start with this command

^XZ every format must end with this command

^FD field data

^FS field separator

^FO field origin

- 3 When the label prints correctly, alter the first number after the ^FOx and see how that effects the print position, then alter the second number after the ^FO50 , x and see how that effects the print position.

## Font instruction

```
^ADN
```

- 4 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, and the second 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.



**5** In the *ZPL II Programming Guide Volume Two*, see Appendix E to check the font matrices tables for other fonts to try.

**6** Try the zero scalable font `^A0N, x, x`.

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

## Rotation commands

**7** Change `^ADN` to `^ADR`, then `^ADI`, then `^ADB`.

See how the print position changes.

**8** Add more fields.

**9** 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

**10** You must add two more lines of code that start off with:

```
^XA
```

```
^FO50,50^ADN,36,20^FDxxxxxxxxxxxx^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:

```
^FO100,50^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 24](#).

```
^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 effects the print quality of the barcode. 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 the 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's 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%%d%%,1000
0^FS
^PQ15
^XZ
```

Notice how the field data character position aligns with ^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/decrement, d = decimal, 10000=increment value.

For more details on ^SF, see [^SF on page 299](#).

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

^IS and image save and image load



**Note •** There is a lot of data to type in this exercise, which puts you at risk of making a typo. It also becomes an exercise to 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.ZPL^FS
^XZ

```

**2** Send this format:

```

^XA
^ILR:SAMPLE.ZPL^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 case the file was saved in the printers R: memory, which is volatile.

## **DF and ^XF — Download format and recall format**

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

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

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

# ^A

## Scalable/Bitmapped Font

**Description** The ^A command is a scalable/bitmapped font that uses built-in or TrueType® fonts. ^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** ^Af $\circ$ ,h,w

This table identifies the parameters for this format:

Command	Details
f = font name	<p><i>Accepted Values:</i> A through Z, and 1 to 9</p> <p><i>Default Value:</i> A</p> <p>Any font in the printer (downloaded, EPROM, stored fonts, fonts A through Z and 1 to 9).</p>
o = font orientation	<p><i>Accepted Values:</i></p> <p>N = normal</p> <p>R = rotated 90 degrees (clockwise)</p> <p>I = inverted 180 degrees</p> <p>B = read from bottom up, 270 degrees</p> <p><i>Default Value:</i> the last accepted ^FW value or the ^FW default</p>

Command	Details
h = Character Height (in dots)	<p><b>Scalable</b></p> <p><i>Accepted Values:</i> 10 to 32000</p> <p><i>Default Value:</i> 15 or the last accepted ^CF value</p> <p><b>Bitmapped</b></p> <p><i>Accepted Values:</i> multiples of height from 2 to 10 times the standard height, in increments of 1</p> <p><i>Default Value:</i> the standard matrix height for a specified font</p>
w = width (in dots)	<p><b>Scalable</b></p> <p><i>Accepted Values:</i> 10 to 32000</p> <p><i>Default Value:</i> 12 or last accepted ^CF value</p> <p><b>Bitmapped</b></p> <p><i>Accepted Values:</i> multiples of width from 2 to 10 times the standard width, in increments of 1</p> <p><i>Default Value:</i> the standard matrix width for a specified font.</p>

### Scalable Font Command

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

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FO50,50 ^A0,32,25 ^FDZEBRA^FS ^FO50,150 ^A0,32,25 ^FDPROGRAMMING^FS ^FO50,250 ^A0,32,25^FDLANGUAGE^FS ^XZ                     </pre>	

## Bitmap Font Command

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

ZPL II CODE	GENERATED LABEL
<pre>^XA ^FO50,50 ^ADN,36,20 ^FDZEBRA^FS ^FO50,150 ^ADN,36,20 ^FDPROGRAMMING^FS ^FO50,250 ^ADN,36,20^FDLANGUAGE^FS ^XZ</pre>	<pre>ZEBRA  PROGRAMMING  LANGUAGE</pre>

➔ **Example** • This is an example of the P - V fonts:

```
FONT B -- ABCDwxyz 12345
FONT B -- ABCDWXYZ 12345
FONT D -- ABCDwxyz 12345
FONT E -- (OCR-B) ABCDwxyz 12345
FONT F -- ABCDwxyz 12345
FONT G -- Az 4
FONT H -- (OCR-A) UPPER CASE ONLY
FONT O -- (Scalable) ABCDwxyz 12345
FONT GS -- ©    ©
FONT P-- ABCDwxyz 12345
FONT Q-- ABCDwxyz 12345
FONT R-- ABCDwxyz 12345
FONT S-- ABCDwxyz 12345
FONT T-- ABCDwxyz 12345
FONT U-- ABCDwxyz 12345
FONT V-- ABCDwxyz 12345
```

**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 = font orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> N or the last ^FW value</p>
h = character height (in dots)	<p><i>Default Value:</i> magnification specified by <b>w</b> (character width) or the last accepted ^CF value. The base height is used if none is specified.</p> <p><b>Scalable</b> the value is the height in dots of the entire character block. Magnification factors are unnecessary, because characters are scaled.</p> <p><b>Bitmapped</b> 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.</p>

Parameters	Details
w = width (in dots)	<p><i>Default Value:</i> magnification specified by <b>h</b> (height) or the last accepted ^CF value. The base width is used if none is specified.</p> <p><b>Scalable</b> the value is the width in dots of the entire character block. Magnification factors are unnecessary, because characters are scaled.</p> <p><b>Bitmapped</b> the value is rounded 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.</p>
d = drive location of font	<p><i>Accepted Values:</i> R:, E:, B:, and A:</p> <p><i>Default Value:</i> R:</p>
o = font name	<p><i>Accepted Values:</i> any valid font</p> <p><i>Default Value:</i> 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.</p> <p>The font named carries over on all subsequent ^A@ commands without a font name.</p>
x = extension	<p><i>Fixed Value:</i> .FTN</p>

➔ **Example** • This example is followed by a table that identifies the callouts:

ZPL II CODE	GENERATED LABEL
1 — ^XA	<div style="border: 1px solid black; padding: 10px; margin: 0 auto; width: 80%;"> <p><b>Zebra Printer Fonts</b> This uses B:CYRI_UB.FNT</p> </div>
2 — ^A@N, 50, 50, B:CYRI_UB.FNT	
3 — ^FO100, 100	
4 — ^FDZebra Printer Fonts^FS	
5 — ^A@N, 40, 40	
6 — ^FO100, 150	
7 — ^FDThis uses B:CYRI_UB.FNT^FS	
8 — ^XZ	

- 
- 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.
- 

**Comments** For more information on scalable and bitmap fonts, see *ZPL II Programming Guide: Volume Two*.

# ^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



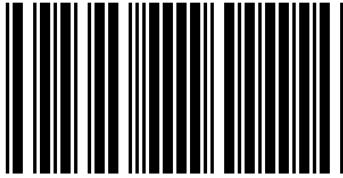
**Important** • If additional information about the Code 11 bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

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

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) Default Value: N

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

CODE 11 BAR CODE	ZPL II CODE
 <p>△12345611△</p>	<pre> ^XA ^FO100,100^BY3 ^B1N,N,150,Y,N ^FD123456^FS ^XZ                     </pre>
CODE 11 BAR CODE CHARACTERS	
<p>0    1    2    3    4    5    6    7    8    9    -</p> <p><b>Internal Start/Stop Character:△</b></p> <p><i>When used as a stop character:</i></p> <p>△ is used with 1 check digit  △ is used with 2 check digits</p>	

# <sup>B</sup>2

## Interleaved 2 of 5 Bar Code

**Description** The <sup>B</sup>2 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.

- <sup>B</sup>2 supports print ratios of 2.0:1 to 3.0:1.
- Field data (<sup>F</sup>D) is limited to the width (or length, if rotated) of the label.

**Format** <sup>B</sup>2 $\circ, h, f, g, e$



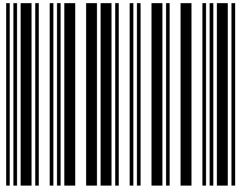
**Important** • If additional information about the Interleaved 2 of 5 bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = orientation	<p><i>Accepted Values:</i></p> <p>N = normal</p> <p>R = rotated 90 degrees (clockwise)</p> <p>I = nverted 180 degrees</p> <p>B = read from bottom up, 270 degrees</p> <p><i>Default Value:</i> current <sup>F</sup>W value</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 32000</p> <p><i>Default Value:</i> value set by <sup>B</sup>Y</p>

Parameters	Details
f = print interpretation line	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> Y
g = print interpretation line above code	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> N
e = calculate and print Mod 10 check digit	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> N

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

INTERLEAVED 2 OF 5 BAR CODE	ZPL II CODE
 123456	<pre> ^XA ^FO100,100^BY3 ^B2N,150,Y,N,N ^FD123456^FS ^XZ </pre>
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 Appendix C in *ZPL Programming Guide Volume Two*.

# ^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




**Important** • If additional information about the Code 39 bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.



This table identifies the parameters for this format:


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

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

CODE 39 BAR CODE	ZPL II CODE																																																		
	<pre> ^XA ^FO100,100^BY3 ^B3N,N,100,Y,N ^FD123ABC^FS ^XZ                     </pre>																																																		
CODE 39 BAR CODE CHARACTERS																																																			
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td><td style="width: 10%;">1</td><td style="width: 10%;">2</td><td style="width: 10%;">3</td><td style="width: 10%;">4</td><td style="width: 10%;">5</td><td style="width: 10%;">6</td><td style="width: 10%;">7</td><td style="width: 10%;">8</td><td style="width: 10%;">9</td> </tr> <tr> <td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td><td>H</td><td>I</td><td>J</td> </tr> <tr> <td>K</td><td>L</td><td>M</td><td>N</td><td>O</td><td>P</td><td>Q</td><td>R</td><td>S</td><td>T</td> </tr> <tr> <td>U</td><td>V</td><td>W</td><td>X</td><td>Y</td><td>Z</td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td></td><td></td><td style="text-align: center;">-</td><td style="text-align: center;">.</td><td style="text-align: center;">\$</td><td style="text-align: center;">/</td><td style="text-align: center;">+</td><td style="text-align: center;">%</td><td style="text-align: center;">Space</td> </tr> </table>	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	
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
<pre data-bbox="467 730 756 877">^XA ^FO20,20 ^B3N,N,100,Y ^FDTEST+\$\$M\$J-\$^FS ^XZ</pre>	

## Full ASCII Mode for Code 39

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

ASCII	Code 39	ASCII	Code 39
SOH	\$A	SP	Space
STX	\$B	!	/A
ETX	\$C	"	/B
EOT	\$D	#	/C
ENQ	\$E	\$	/D
ACK	\$F	%	/E
BEL	\$G	&	/F
BS	\$H	'	/G
HT	\$I	(	/H
LF	\$J	)	/I
VT	\$K	*	/J
FF	\$L	++	/K
CR	\$M	'	/L
SO	\$N	-	-
SI	\$O	.	.
DLE	\$P	/	/O
DC1	\$Q	0	O
DC2	\$R	1	1
DC3	\$S	2	2
DC4	\$T	3	3
NAK	\$U	4	4
SYN	\$V	5	5
ETB	\$W	6	6
CAN	\$X	7	7
EM	\$Y	8	8
SUB	\$Z	9	9
ESC	%A	:	/Z
FS	%B	:	%F
FS	%C	<	%G
RS	%D	=	%H
US	%E	>	%I
		?	%J

**Table A• Code 39 Full ASCII Mode**

ASCII	Code 39	ASCII	Code 39
@	%V	'	%W
A	A	a	+A
B	B	b	+B
C	C	c	+C
D	D	d	+D
E	E	e	+E
F	F	f	+F
G	G	g	+G
H	H	h	+H
I	I	l	+I
J	J	j	+J
K	K	k	+K
L	L	l	+L
M	M	m	+M
N	N	n	+N
O	O	o	+O
P	P	p	+P
Q	Q	q	+Q
R	R	r	+R
S	S	s	+S
T	T	t	+T
U	U	u	+U
V	V	v	+V
W	W	w	+W
X	X	x	+X
Y	Y	y	+Y
Z	Z	z	+Z
[	%K	{	%P
\	%L		%Q
]	%M	}	%R
^	%N	~	%S
_	%O	DEL	%T, %X

**Table B• Code 39 Full ASCII Mode**

# ^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. Rows are separated by a separator bar with a height of one module. 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

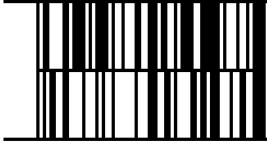


**Important** • If additional information about the Code 49 bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
h = height multiplier of individual rows	<p><i>Accepted Values:</i> 1 to height of label</p> <p><i>Default Value:</i> value set by ^BY</p> <p>This number multiplied by the module equals the height of the individual rows in dots. 1 is not a recommended value.</p>
f = print interpretation line	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = no line printed</li> <li>A = print interpretation line above code</li> <li>B = print interpretation line below code</li> </ul> <p><i>Default Value:</i> N</p> <p>When the field data exceeds two rows, expect the interpretation line to extend beyond the right edge of the bar code symbol.</p>
m = starting mode	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>0 = Regular Alphanumeric Mode</li> <li>1 = Multiple Read Alphanumeric</li> <li>2 = Regular Numeric Mode</li> <li>3 = Group Alphanumeric Mode</li> <li>4 = Regular Alphanumeric Shift 1</li> <li>5 = Regular Alphanumeric Shift 2</li> <li>A = Automatic Mode. The printer determines the starting mode by analyzing the field data.</li> </ul> <p><i>Default Value:</i> A</p>

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

CODE 49 BAR CODE	ZPL II CODE
<p data-bbox="532 485 727 512">12345ABCDE</p> 	<pre data-bbox="987 499 1243 642">^XA ^FO150,100^BY3 ^B4N,20,A,A ^FD12345ABCDE^FS ^XZ</pre>

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
-	-	(	ˆ
.	.	)	˘
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**

**Table C• Code 49**



## 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.

**Format** ^B5 $\circ$ , $h$ , $f$ , $g$

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = orientation code	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
$h$ = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 9999</p> <p><i>Default Value:</i> value set by ^BY</p>
$f$ = interpretation line	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = no default</li> <li>Y = yes</li> </ul>
$g$ = determines if the interpretation line is printed above the bar code	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = no default</li> <li>Y = yes</li> </ul>

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

ZPL II CODE	GENERATED LABEL
<pre data-bbox="431 480 704 625">^XA ^FO150,100^BY3 ^B5N,100,Y,0 ^FD12345678901\$FS ^XZ</pre>	

# ^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 160](#).

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

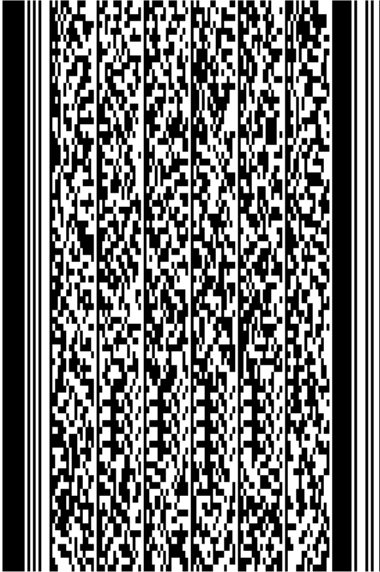
**Format** ^B7o,h,s,c,r,t

This table identifies the parameters for this format:

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

Parameters	Details
<p>s = security level</p>	<p><i>Accepted Values:</i> 1 to 8 (error detection and correction)  <i>Default Value:</i> 0 (error detection only)</p> <p>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.</p>
<p>c = number of data columns to encode</p>	<p><i>Accepted Values:</i> 1 to 30  <i>Default Value:</i> 1 : 2 (row-to-column aspect ratio)</p> <p>You can specify the number of code-word columns giving control over the width of the symbol.</p>
<p>r = number of rows to encode</p>	<p><i>Accepted Values:</i> 3 to 90  <i>Default Value:</i> 1 : 2 (row-to-column aspect ratio)</p> <p>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.</p>
<p>t = truncate right row indicators and stop pattern</p>	<p><i>Accepted Values:</i> Y (perform truncation) and N (no truncation)  <i>Default Value:</i> N</p>

➔ **Example 1** • This is an example of a PDF417 bar code:

PDF417 BAR CODE	ZPL II CODE
	<pre> ^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 </pre>

➔ **Example 2** • This is an example of a PDF417 without and with truncation selected:




**PDF417 without Truncation being selected**



**PDF417 with Truncation being selected**

➔ **Example 3** • This example shows the ^B7 command used with field hex (^FH) characters:

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FO50,50^BY3,3.0^B7N,8,5,7,21,N ^FH_ ^FD(&gt;_1E06_1DP12345678_1DQ160 _1D1JUN123456789A2B4C6D8E_1D20LA6-987 _1D21L54321_ZES_1D15KG1155 _1DBSC151208_1D7Q10GT_1E_04^FS ^XZ           </pre>	

**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 nontruncated and a truncated bar code is shown in [This is an example of a PDF417 without and with truncation selected: on page 39](#).

### 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.

## Page Number 850 table

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

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, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 32000</p> <p><i>Default Value:</i> value set by ^BY</p>

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) Default Value: N

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

EAN-8 BAR CODE		ZPL II CODE							
		<pre> ^XA ^FO100,100^BY3 ^B8N,100,Y,N ^FD1234567^FS ^XZ                     </pre>							
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** ^Bo,h,f,g,e




**Important** • If additional information about the UPC-E bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

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

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

UPC-E BAR CODE		ZPL II CODE							
		<pre> ^XA ^FO150,100^BY3 ^B9N,100,Y,N,Y ^FD1230000045^FS ^XZ                     </pre>							
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, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.



This table identifies the parameters for this format:

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

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

CODE 93 BAR CODE	ZPL II CODE																																																																																																																			
	<pre> ^XA ^FO100,75^BY3 ^BAN,100,Y,N,N ^FD12345ABCDE^FS ^XZ           </pre>																																																																																																																			
CODE 93 BAR CODE CHARACTERS																																																																																																																				
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">0</td> <td style="width: 10%;">1</td> <td style="width: 10%;">2</td> <td style="width: 10%;">3</td> <td style="width: 10%;">4</td> <td style="width: 10%;">5</td> <td style="width: 10%;">6</td> <td style="width: 10%;">7</td> <td style="width: 10%;">8</td> <td style="width: 10%;">9</td> </tr> <tr> <td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td><td>H</td><td>I</td><td>J</td><td>K</td><td>L</td><td>M</td><td>N</td><td>O</td><td>P</td><td>Q</td><td>R</td><td>S</td><td>T</td><td>U</td><td>V</td><td>W</td><td>X</td><td>Y</td><td>Z</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td>-</td><td>.</td><td>\$</td><td>/</td><td>+</td><td>%</td><td>&amp;</td><td>'</td><td>(</td><td>)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table>			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						-	.	\$	/	+	%	&	'	(	)																																																															
	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																																																																																											
					-	.	\$	/	+	%	&	'	(	)																																																																																																						
<p>□ Denotes an internal start/stop character that must precede and follow every bar code message.</p>																																																																																																																				

**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 D on page 51](#).

### Full ASCII Mode for Code 93

Code 93 can generate the full 128-character ASCII set using paired characters as shown in tables C and D.

ASCII	Code 93	ASCII	Code 93
NUL	'U	SP	Space
SOH	&A	!	(A
STX	&B	"	(B
ETX	&C	#	(C
EOT	&D	\$	(D
ENQ	&E	%	(E
ACK	&F	&	(F
BEL	&G	'	(G
BS	&H	(	(H
HT	&I	)	(I
LF	&J	*	(J
VT	&K	++	++
FF	&L	'	(L
CR	&M	-	-
SO	&N	.	.
SI	&O	/	/
DLE	&P	0	0
DC1	&Q	1	1
DC2	&R	2	2
DC3	&S	3	3
DC4	&T	4	4
NAK	&U	5	5
SYN	&V	6	6
ETB	&W	7	7
CAN	&X	8	8
EM	&Y	9	9
SUB	&Z	:	(Z
ESC	'A	;	'F
FS	'B	<	'G
FS	'C	=	'H
RS	'D	>	'I
US	'E	?	'J

Table D • Code 93 Full ASCII Mode

ASCII	Code 93	ASCII	Code 93
@	'V	'	'W
A	A	a	)A
B	B	b	)B
C	C	c	)C
D	D	d	)D
E	E	e	)E
F	F	f	)F
G	G	g	)G
H	H	h	)H
I	I	l	)I
J	J	j	)J
K	K	k	)K
L	L	l	)L
M	M	m	)M
N	N	n	)N
O	O	o	)O
P	P	p	)P
Q	Q	q	)Q
R	R	r	)R
S	S	s	)S
T	T	t	)T
U	U	u	)U
V	V	v	)V
W	W	w	)W
X	X	x	)X
Y	Y	y	)Y
Z	Z	z	)Z
[	'K	{	'P
\	'L		'Q
]	'M	}	'R
^	'N	~	'S
_	'O	DEL	'T

**Table E • Code 93 Full ASCII Mode**

# ^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** ^BB $\circ$ ,h,s,c,r,m



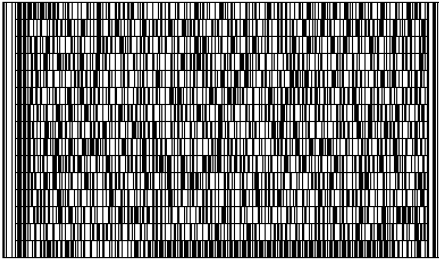
**Important** • If additional information about the CODABLOCK bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = orientation	<p><i>Accepted Values:</i></p> <p>N = normal</p> <p>R = rotated 90 degrees (clockwise)</p> <p>I = inverted 180 degrees</p> <p>B = read from bottom up, 270 degrees</p> <p><i>Default Value:</i> N</p>
h = bar code height for individual rows (in dots)	<p><i>Accepted Values:</i> 2 to 32000</p> <p><i>Default Value:</i> 8</p> <p>This number, multiplied by the module, equals the height of the individual row in dots.</p>

Parameters	Details
s = security level	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> Y</p> <p>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.</p>
c = number of characters per row (data columns)	<p><i>Accepted Values:</i> 2 to 62 characters</p> <p>This is used to encode a CODABLOCK symbol. It gives the you control over the width of the symbol.</p>
r = number of rows to encode	<p><i>Accepted Values:</i></p> <p>for CODABLOCK A: 1 to 22</p> <p>for CODABLOCK E and F: 2 to 4</p> <ul style="list-style-type: none"> <li>• If values for c and r are not specified, a single row is produced.</li> <li>• 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.</li> <li>• 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.</li> <li>• 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).</li> <li>• 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.</li> </ul>
m = mode	<p><i>Accepted Values:</i> A, E, F</p> <p>CODABLOCK A uses the Code 39 character set.</p> <p>CODABLOCK F uses the Code 128 character set.</p> <p>CODABLOCK E uses the Code 128 character set and automatically adds FNC1.</p> <p><i>Default Value:</i> F</p>

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

CODABLOCK BAR CODE	ZPL II CODE
	<pre> ^XA ^BY2,3 ^FO10,10^BBN,30,,30,44,E ^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 </pre>

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

The parameters for the ^BY $w, r, h$  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 through the use 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 through the use 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, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 32000</p> <p><i>Default Value:</i> value set by ^BY</p>

Parameters	Details
f = print interpretation line	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> Y</p> <p>The interpretation line can be printed in any font by placing the font command before the bar code command.</p>
g = print interpretation line above code	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> N</p>
e = UCC check digit	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> N</p>
m = mode	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = no selected mode</li> <li>U = UCC Case Mode</li> <li>A = Automatic Mode. 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 cause an automatic shift to Subset C.</li> </ul> <p><i>Default Value:</i> N</p>
d = insert data	

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

CODE 128 BAR CODE	ZPL II CODE
	<pre> ^XA ^FO100,100^BY3 ^BCN,100,Y,N,N ^FD123456^FS ^XZ                     </pre>

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)	

Table F • Code 128 character sets

**Special Conditions if UCC Case Mode is Selected**

- More than 19 digits in <sup>^</sup>FD or <sup>^</sup>SN are eliminated.
- Fewer than 19 digits in <sup>^</sup>FD or <sup>^</sup>SN add zeros to the right to bring the count to 19. This produces an invalid interpretation line.

**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 (<sup>^</sup>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.

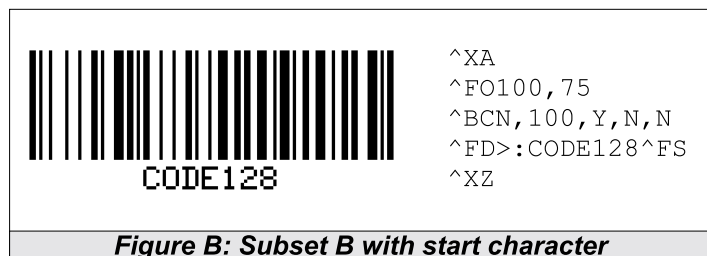
To change subsets within a bar code, place the Invocation Code at the appropriate points within the field data (<sup>^</sup>FD) string. The new subset stays in effect until changed with the Invocation Code. For example, in Subset C, >7 in the field data changes the Subset to A.

The table below shows the Code 128 Invocation Codes and Start Characters for the three subsets.

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
<b>Start Characters</b>				
>9	103	Start Code A	(Numeric Pairs give Alpha/Numerics)	
>:	104	Start Code B	(Normal Alpha/Numeric)	
		Start Code C		

**Table G • Code 128 Invocation Characters**

➔ **Example** • Figures A and B are examples of identical bar codes, as follows:



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.

### 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:

^ > ~

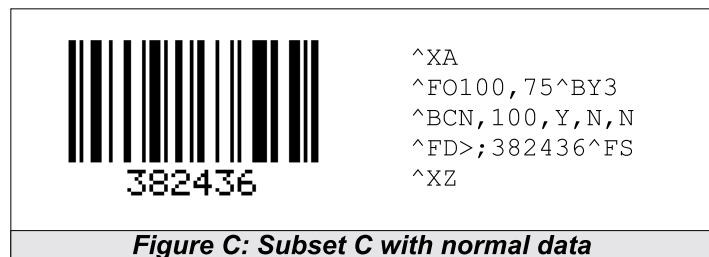
**➔ Example • Code 128 – Subsets A and C**

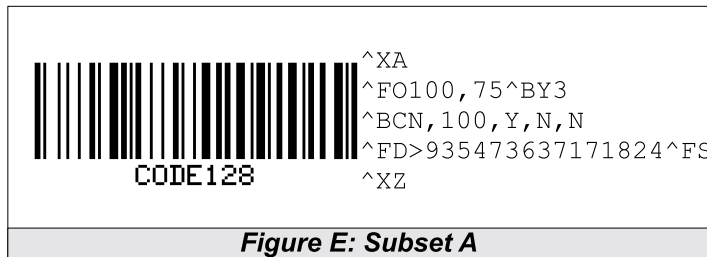
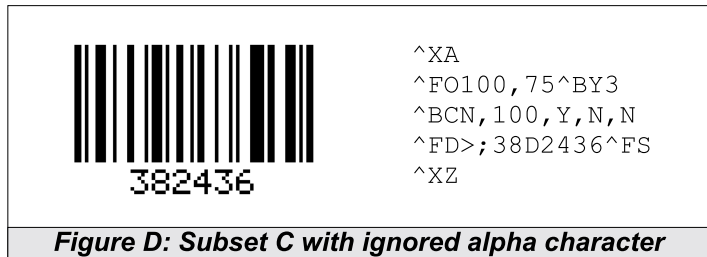
Code 128, Subsets A and C are programmed in pairs of digits, 00 to 99, in the field data string. For details see, *Table F on page 59*.

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.





# <sup>^</sup>BD

## UPS MaxiCode Bar Code

**Description** The <sup>^</sup>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 <sup>^</sup>BY command has no effect on the UPS MaxiCode bar code. However, the <sup>^</sup>CV command can be activated.

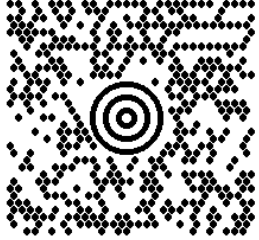
**Format** <sup>^</sup>BDm,n,t

This table identifies the parameters for this format:

Parameters	Details
m = mode	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>2 = structured carrier message: numeric postal code (U.S.)</li> <li>3 = structured carrier message: alphanumeric postal code (non-U.S.)</li> <li>4 = standard symbol, secretary</li> <li>5 = full EEC</li> <li>6 = reader program, secretary</li> </ul> <p><i>Default Value:</i> 2</p>
n = symbol number	<p><i>Accepted Values:</i> 1 to 8 can be added in a structured document</p> <p><i>Default Value:</i> 1</p>
t = total number of symbols	<p><i>Accepted Values:</i> 1 to 8, representing the total number of symbols in this sequence</p> <p><i>Default Value:</i> 1</p>



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

UPS MAXICODE - MODE 2	ZPL II CODE
	<pre> ^XA ^FO50,50 ^CVY ^BD^FH^FD001840152382802 [ ]&gt;_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 </pre>

### 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)	<p><i>Accepted Values:</i> 0 to 9, except where noted</p> <p><b>U.S. Style Postal Code (Mode 2)</b></p> <p>&lt;hpm&gt; = aaabbbccccdddd</p> <p>aaa = three-digit class of service</p> <p>bbb = three-digit country zip code</p> <p>cccc = five-digit zip code</p> <p>dddd = four-digit zip code extension (if none exists, four zeros (0000) must be entered)</p> <p><b>non-U.S. Style Postal Code (Mode 3)</b></p> <p>&lt;hpm&gt; = aaabbbcccccc</p> <p>aaa = three-digit class of service</p> <p>bbb = three-digit country zip code</p> <p>cccc = six-digit zip code (A through Z or 0 to 9)</p>																								
<lpm> = low priority message (only applicable in Modes 2 and 3)	<p>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.</p>																								
	<hr/> <table> <tr> <td>Message Header</td> <td>[ ]&gt;RS</td> </tr> <tr> <td>Transportation Data</td> <td></td> </tr> <tr> <td>Format Header</td> <td>01GS96</td> </tr> <tr> <td>Tracking Number*</td> <td>&lt;tracking number&gt;</td> </tr> <tr> <td>SCAC*</td> <td>GS&lt;SCAC&gt;</td> </tr> <tr> <td>UPS Shipper Number</td> <td>GS&lt;shipper number&gt;</td> </tr> <tr> <td>Julian Day of Pickup</td> <td>GS&lt;day of pickup&gt;</td> </tr> <tr> <td>Shipment ID Number</td> <td>GS&lt;shipment ID number&gt;</td> </tr> <tr> <td>Package n/x</td> <td>GS&lt;n/x&gt;</td> </tr> <tr> <td>Package Weight</td> <td>GS&lt;weight&gt;</td> </tr> <tr> <td>Address Validation</td> <td>GS&lt;validation&gt;</td> </tr> <tr> <td>Ship to Street Address</td> <td>GS&lt;street address&gt;</td> </tr> </table> <hr/>	Message Header	[ ]>RS	Transportation Data		Format Header	01GS96	Tracking Number*	<tracking number>	SCAC*	GS<SCAC>	UPS Shipper Number	GS<shipper number>	Julian Day of Pickup	GS<day of pickup>	Shipment ID Number	GS<shipment ID number>	Package n/x	GS<n/x>	Package Weight	GS<weight>	Address Validation	GS<validation>	Ship to Street Address	GS<street address>
Message Header	[ ]>RS																								
Transportation Data																									
Format Header	01GS96																								
Tracking Number*	<tracking number>																								
SCAC*	GS<SCAC>																								
UPS Shipper Number	GS<shipper number>																								
Julian Day of Pickup	GS<day of pickup>																								
Shipment ID Number	GS<shipment ID number>																								
Package n/x	GS<n/x>																								
Package Weight	GS<weight>																								
Address Validation	GS<validation>																								
Ship to Street Address	GS<street address>																								

---

Ship to City	GS<city>
Ship to State	GS<state>
RS	RS
End of Message	EOT

(\* 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 as 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




**Important** • If additional information about the EAN-13 bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

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

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

EAN-13 BAR CODE	ZPL II CODE								
 <p>0 000123 456784</p>	<pre> ^XA ^FO100,100^BY3 ^BEN,100,Y,N ^FD12345678^FS ^XZ </pre>								
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 Appendix C in *ZPL Programming Guide Volume Two*.

# <sup>^</sup>BF

## Micro-PDF417 Bar Code

**Description** The <sup>^</sup>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 (<sup>^</sup>FD) and field hexadecimal (<sup>^</sup>FH) are limited to:

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


**Format** <sup>^</sup>BF<sub>o, h, m</sub>

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<p><i>Accepted Values:</i></p> <p>N = normal</p> <p>R = rotated 90 degrees (clockwise)</p> <p>I = inverted 180 degrees</p> <p>B = read from bottom up, 270 degrees</p> <p><i>Default Value:</i> current <sup>^</sup>FW value</p>

Parameters	Details
h = bar code height (in dots)	<i>Accepted Values:</i> 1 to 9999 <i>Default Value:</i> value set by ^BY or 10 (if no ^BY value exists).
m = mode	<i>Accepted Values:</i> 0 to 33 ( <i>on page 72</i> ) <i>Default Value:</i> 0 ( <i>on page 72</i> ).

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

MICRO-PDF417 BAR CODE	ZPL II CODE
	<pre> ^XA ^FO100,100^BY6 ^BFN,8,3 ^FDABCDEF GHIJKLMNOPQRSTU V^FS ^XZ </pre>

### 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 *on page 72* with the information gathered from the questions above to select the mode of the bar code.

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

**Table H • Micro-PDF417 Mode**



# ^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** ^BI $\circ$ ,h,f,g



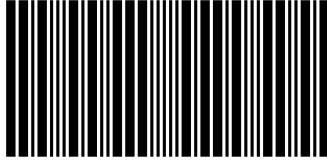
**Important** • If additional information about the Industrial 2 of 5 bar code, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = orientation	<p><i>Accepted Values:</i></p> <p>N = normal</p> <p>R = rotated 90 degrees (clockwise)</p> <p>I = inverted 180 degrees</p> <p>B = read from bottom up, 270 degrees</p> <p><i>Default Value:</i> current ^FW value</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 32000</p> <p><i>Default Value:</i> value set by ^BY</p>

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) Default Value: N

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

INDUSTRIAL 2 OF 5 BAR CODE	ZPL II CODE
 123456	<pre> ^XA ^FO100,100^BY3 ^BIN,150,Y,N ^FD123456^FS ^XZ           </pre>
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** ^BJ $\circ$ , h, f, g



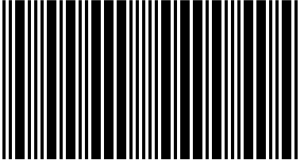
**Important** • If additional information about the Standard 2 of 5 bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 32000</p> <p><i>Default Value:</i> value set by ^BY</p>

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) 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	<pre> ^XA ^FO100,100^BY3 ^BJN,150,Y,N ^FD123456^FS ^XZ           </pre>
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** ^BK $\circ, e, h, f, g, k, l$




**Important** • If additional information about the ANSI Codabar bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = orientation	<p><i>Accepted Values:</i></p> <p>N = normal</p> <p>R = rotated 90 degrees (clockwise)</p> <p>I = inverted 180 degrees</p> <p>B = read from bottom up, 270 degrees</p> <p><i>Default Value:</i> current ^FW value</p>
e = check digit	<i>Fixed Value:</i> N
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 32000</p> <p><i>Default Value:</i> value set by ^BY</p>

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) Default Value: N
k = designates a start character	Accepted Values: A, B, C, D Default Value: A
l = designates stop character	Accepted Values: A, B, C, D Default Value: A

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

ANSI CODABAR BAR CODE		ZPL II CODE							
 <p>A123456A</p>		<pre> ^XA ^FO100,100^BY3 ^BKN,N,150,Y,N,A,A ^FD123456^FS ^XZ                     </pre>							
ANSI CODABAR BAR CODE CHARACTERS									
0	1	2	3	4	5	6	7	8	9
<b>Control Characters</b> - : . \$ / +									
<b>Start/Stop Characters</b> 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** ^BL $\circ$ ,h,g




**Important** • If additional information about the LOGMARS bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

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

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

LOGMARS BAR CODE	ZPL II CODE																																																																																							
	<pre> <sup>^</sup>XA <sup>^</sup>FO100,75<sup>^</sup>BY3 <sup>^</sup>BLN,100,N <sup>^</sup>FD12AB<sup>^</sup>FS <sup>^</sup>XZ                     </pre>																																																																																							
LOGMARS BAR CODE CHARACTERS																																																																																								
<table style="width: 100%; border: none;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">0</td> <td style="width: 10%;">1</td> <td style="width: 10%;">2</td> <td style="width: 10%;">3</td> <td style="width: 10%;">4</td> <td style="width: 10%;">5</td> <td style="width: 10%;">6</td> <td style="width: 10%;">7</td> <td style="width: 10%;">8</td> <td style="width: 10%;">9</td> </tr> <tr> <td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td><td>H</td><td>I</td><td>J</td><td>K</td><td>L</td><td>M</td><td>N</td><td>O</td><td>P</td><td>Q</td><td>R</td><td>S</td><td>T</td><td>U</td><td>V</td><td>W</td><td>X</td><td>Y</td><td>Z</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td>-</td><td>.</td><td>\$</td><td>/</td><td>+</td><td>%</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>SPACE</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table>			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													
	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 Appendix D in *ZPL Programming Guide Volume Two*.



# ^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** ^BM $o, e, h, f, g, e2$




**Important** • If additional information about the MSI bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
e = check digit selection	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>A = no check digits</li> <li>B = 1 Mod 10</li> <li>C = 2 Mod 10</li> <li>D = 1 Mod 10 and 1 Mod 11</li> </ul> <p><i>Default Value:</i> B</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>A = no check digits</li> <li>B = 1 Mod 10</li> <li>C = 2 Mod 10</li> <li>D = 1 Mod 10 and 1 Mod 11</li> </ul> <p><i>Default Value:</i> B</p>
f = print interpretation line	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> Y</p>
g = print interpretation line above code	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> N</p>
e2 = designates start character	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> N</p>



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

MSI BAR CODE					ZPL II CODE				
					<pre style="margin: 0;">^XA ^FO100,100^BY3 ^BMN,B,100,Y,N,N ^FD123456^FS ^XZ</pre>				
MSI BAR CODE CHARACTERS									
1	2	3	4	5	6	7	8	9	

# <sup>^</sup>BP

## Plessey Bar Code

**Description** The <sup>^</sup>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.

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

**Format** <sup>^</sup>BP<sub>o, e, h, f, g</sub>




**Important** • If additional information about the Plessey bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<i>Accepted Values:</i> N = normal R = rotated 90 degrees (clockwise) I = inverted 180 degrees B = read from bottom up, 270 degrees <i>Default Value:</i> current <sup>^</sup> FW value
e = print check digit	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> N
h = bar code height (in dots)	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> N

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) Default Value: N

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

PLESSEY BAR CODE	ZPL II CODE								
	<pre> ^XA ^FO100,100^BY3 ^BPN,N,100,Y,N ^FD12345^FS ^XZ </pre>								
PLESSEY BAR CODE CHARACTERS									
0	1	2	3	4	5	6	7	8	9
			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




**Important** • If additional information about the QR Code bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
a = field position	<i>Fixed Value:</i> normal (^FW has no effect on rotation)

Parameters	Details
b = model	<i>Accepted Values:</i> 1 (original) and 2 (enhanced – recommended) <i>Default Value:</i> 2
c = magnification factor	<i>Accepted Values:</i> 1 to 10 <i>Default Value:</i> 1 on 150 dpi printers 2 on 200 dpi printers 3 on 300 dpi printers 6 on 600 dpi printers

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

QR CODE BAR CODE	ZPL II CODE
	<pre> ^XA ^FO100,100 ^BQN,2,10 ^FDMM,AAC-42^FS ^XZ </pre>

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.

```
^XA
^FO20,20^BQ,2,10^FDQA,0123456789ABCD 2D
code^FS
^XZ
```

---

<b>&lt;error correction level&gt;</b>	Q	(high)
<b>&lt;input mode&gt;</b>	A,	(automatic setting)
<b>&lt;data character string&gt;</b>	0123456789ABCD 2D code	

---

## 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:

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

---

<b>&lt;error correction level&gt;</b>	H	(ultra-high reliability level)
<b>&lt;input mode&gt;</b>	M,	(manual input)

---

---

<character mode>	N	(numeric data)
<data character string>	123456789012345	

---

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

```

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

```

---

<error correction level>	M	(standard reliability level)
<input mode>	M,	(manual input)
<character mode>	A	(alphanumeric data)
<data character string>	AC-42	

---

## ^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:

```
^XA
^FO,20,20^BQ,2,10
^FDD03048F,LM,N0123456789,A12AABB,B0006qrco
de^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>	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>		qrco

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

```

^XA
^FO20,20^BQ,2,10
^FDD03048F,LA,012345678912AABBqr code^FS
^XZ

```

---

<b>&lt;mixed mode identifier&gt;</b>	D	(mixed)
<b>&lt;code No.&gt;</b>	M	(code number)
<b>&lt;No. of divisions&gt;</b>	D	(divisions)
<b>&lt;parity data&gt;</b>	M	(0x8F)
<b>&lt;error correction level&gt;</b>	L	(high-density level)
<b>&lt;input mode&gt;</b>	A	(automatic input)
<b>&lt;data character string&gt;</b>		012345678912AABBqr code

---

# ^BR

## RSS Bar Code

**Description** This is specific to CompactFLASH for the *XiIII*Plus printers.

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

This table identifies the parameters for this format:

Parameters	Details
a = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = Normal</li> <li>R = Rotated</li> <li>I = Inverted</li> <li>B = Bottom-up</li> </ul> <p><i>Default Value:</i> R</p>
b = symbology type in the RSS-14 family	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>1 = RSS14</li> <li>2 = RSS14 Truncated</li> <li>3 = RSS14 Stacked</li> <li>4 = RSS14 Stacked Omnidirectional</li> <li>5 = RSS Limited</li> <li>6 = RSS Expanded</li> <li>7 = UPC-A</li> <li>8 = UPC-E</li> <li>9 = EAN-13</li> <li>10 = EAN-8</li> <li>11 = UCC/EAN-128 &amp; CC-A/B</li> <li>12 = UCC/EAN-128 &amp; CC-C</li> </ul> <p><i>Default Value:</i> 1</p>

Parameters	Details
c = magnification factor	<p><i>Accepted Values:</i> 1 to 10</p> <p><i>Default Values:</i></p> <p>24 dot = 6, 12 dot is 3, 8 dot and lower is 2                      12 dot = 6, &gt; 8 dot is 3, 8 dot and less is 2)</p>
d = separator height	<p><i>Accepted Values:</i> 1 or 2</p> <p><i>Default Value:</i> 1</p>
e = bar code height	<p>The bar code height only affects the linear portion of the barcode. Only, UCC/EAN &amp; CC-A/B/C.</p> <p><i>Accepted Values:</i> 1 to 32000 dots</p> <p><i>Default Value:</i> 25</p>
f = the segment width (RSS expanded only)	<p><i>Accepted Values:</i> 2 to 22, even only, segs/line</p> <p><i>Default Value:</i> 22</p>

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

```

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

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

```

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

# ^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** ^BS $\circ$ ,h,f,g



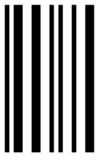
**Important** • If additional information about the UPC/EAN bar code is required, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:


Parameters	Details
$\circ$ = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 32000</p> <p><i>Default Value:</i> value set by ^BY</p>

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) 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
 <b>12</b>	<pre> ^XA ^FO100,100^BY3 ^BSN,100,Y,N ^FD12^FS ^XZ                     </pre>
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
 <b>12345</b>	<pre> ^XA ^FO100,100^BY3 ^BSN,100,Y,N ^FD12345^FS ^XZ                     </pre>
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
<i>Normal</i>	209 Dots	21 Dots
<i>Rotated</i>	0	209 Dots

This is an example of a UPC-E:

	Supplement Origin X - Offset	Adjustment Y - Offset
<i>Normal</i>	122 Dots	21 Dots
<i>Rotated</i>	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	ZPL II CODE
	<pre> ^XA ^FO100,100^BY3 ^BUN,137 ^FD07000002198^FS ^FO400,121 ^BSN,117 ^FD04414^FS ^XZ </pre>

# <sup>^</sup>BT

## TLC39 bar code

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

The TCIF CLEI codes 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** <sup>^</sup>BT<sub>o,w1,r1,h1,w2,h2</sub>

This table identifies the parameters for this format:

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

Parameters	Details
h2 = row height of the Micro-PDF417 bar code	<p><i>Accepted Values (in dots):</i> 1 to 255</p> <p><i>Default Value (600 dpi printer):</i> 8</p> <p><i>Default Value (200- and 300 dpi printers):</i> 4</p>
w2 = narrow bar width of the Micro-PDF417 bar code	<p><i>Accepted Values (in dots):</i> 1 to 10</p> <p><i>Default Value (600 dpi printer):</i> 4</p> <p><i>Default Value (200- and 300 dpi printers):</i> 2</p>

### ➔ Example • TLC39 Bar Code

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

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

```

^XA^FO100,100^BT^FD123456,ABCD12345678901234
5551212,888999^FS^XZ

```

**A — 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.

**B — 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.

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

**Note** • 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	GENERATED LABEL
<pre data-bbox="467 753 756 926">^XA^FO100 100^BT^FD123456 ABCd12345678901234 5551212 88899 ^FS^XZ</pre>	

# ^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** ^BU $\circ, h, f, g, e$



**Important** • If additional information about the UPC-A bar code is required, see *ZPL II Programming Guide Volume Two* for AIM, Inc. contact information.

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
h = bar code height (in dots)	<p><i>Accepted Values:</i> 1 to 9999</p> <p><i>Default Value:</i> value set by ^BY</p>

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: Y
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) Default Value: N
e = print check digit	Accepted Values: Y (yes) and N (no) Default Value: Y

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


**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, or 3 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
	<pre> ^XA ^FO100,100^BY3 ^BUN,137 ^FD07000002198^FS ^FO400,121 ^BSN,117 ^FD04414^FS ^XZ                     </pre>

**Comments** The UPC-A bar code uses the Mod 10 check digit scheme for error checking. For further information on Mod 10, see Appendix C in *ZPL Programming Guide Volume Two*.

# ^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.

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

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotated 90 degrees (clockwise)</li> <li>I = inverted 180 degrees</li> <li>B = read from bottom up, 270 degrees</li> </ul> <p><i>Default Value:</i> current ^FW value</p>
h = dimensional height of individual symbol elements	<p><i>Accepted Values:</i> 1 to the width of the label</p> <p>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.</p>
s = quality level	<p><i>Accepted Values:</i> 0, 50, 80, 100, 140, 200</p> <p><i>Default Value:</i> 0</p> <p><i>Quality</i> 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.</p>

Parameters	Details
c = columns to encode	<p><i>Accepted Values:</i> 9 to 49</p> <p>Odd values only for quality 0 to 140 (10 to 144); even values only for quality 200.</p>
r = rows to encode	<p><i>Accepted Values:</i> 9 to 49</p> <p>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.</p>



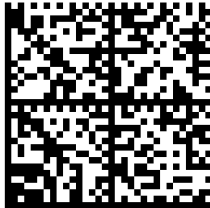
Parameters	Details
f = format ID (0 to 6) — not used with quality set at 200	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>1 = field data is numeric + space (0..9,") – No \&amp;"</li> <li>2 = field data is uppercase alphanumeric + space (A..Z,") – No \&amp;"</li> <li>3 = field data is uppercase alphanumeric + space, period, comma, dash, and slash (0..9,A..Z,“.-/”)</li> <li>4 = field data is upper-case alphanumeric + space (0..9,A..Z,") – no \&amp;"</li> <li>5 = field data is full 128 ASCII 7-bit set</li> <li>6 = field data is full 256 ISO 8-bit set</li> </ul> <p><i>Default Value:</i> 6</p>
g = escape sequence control character	<p><i>Accepted Values:</i> any character</p> <p><i>Default Value:</i> _ (underscore)</p> <p>This parameter is used only if quality 200 is specified. It is the escape character for embedding special control sequences within the field data. See <a href="#">Field Data (^FD)</a> for ^BX usage.</p>

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

**Table 1 • Maximum Field Sizes**

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

DATA MATRIX BAR CODE	ZPL II CODE
	<pre> ^XA ^FO100,100 ^BXN,10,200 ^FDZEBRA TECHNOLOGIES CORP 333 CORPORATE WOODS PKWY VERNON HILLS, ILLINOIS 60061-3109^FS ^XZ </pre>

### Effects of ^BY on ^BX

**w = module** (no effect — see dimensions of individual symbol elements)

**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 characters are supplied as field data, it is truncated to 3072 characters. 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 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 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)

# <sup>^</sup>BY

## Bar Code Field Default

**Description** The <sup>^</sup>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** <sup>^</sup>BYw, r, h

This table identifies the parameters for this format:

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

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

➔ **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.

Ratio Selected (r)	Module Width in Dots (w)									
	1	2	3	4	5	6	7	8	9	10
<b>2.0</b>	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1
<b>2.1</b>	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2.1:1
<b>2.2</b>	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
<b>2.3</b>	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
<b>2.4</b>	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
<b>2.5</b>	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
<b>2.6</b>	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
<b>2.7</b>	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
<b>2.8</b>	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
<b>2.9</b>	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
<b>3.0</b>	3:1	3:1	3:1	3:1	3:1	3:1	3:1	3:1	3:1	3:1

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

**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, see *ZPL Programming Guide Volume Two* for AIM, Inc. contact information, 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:

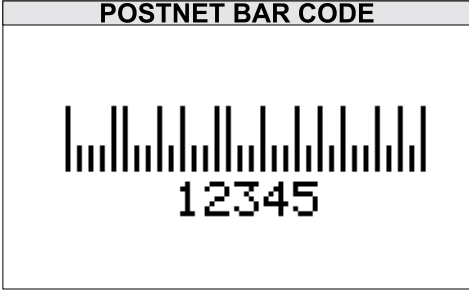
<http://pe.usps.gov/cpim/ftp/pubs/pub25/pub25.pdf>

This table identifies the parameters for this format:

Parameters	Details
o = orientation	<i>Accepted Values:</i> N = normal R = rotated 90 degrees (clockwise) I = inverted 180 degrees B = read from bottom up, 270 degrees <i>Default Value:</i> current ^FW value
h = bar code height (in dots)	<i>Accepted Values:</i> 1 to 32000 <i>Default Value:</i> value set by ^BY

Parameters	Details
f = print interpretation line	Accepted Values: Y (yes) or N (no) Default Value: N
g = print interpretation line above code	Accepted Values: Y (yes) or N (no) Default Value: N

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

POSTNET BAR CODE	ZPL II CODE								
	<pre> ^XA ^FO100,100^BY3 ^BZN,40,Y,N ^FD12345^FS ^XZ </pre>								
POSTNET BAR CODE CHARACTERS									
0	1	2	3	4	5	6	7	8	9

# ^CC

## Change Carets

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

**Format** ^CCx

This table identifies the parameters for this format:

Parameters	Details
x = caret character change	<i>Accepted Values:</i> any ASCII character <i>Default Value:</i> 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 ^ to a /:

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

**Comments** In the above example, the forward slash (/) is set at the new prefix. Note the /XZ ending tag uses the new designated prefix character (/).



# ~CC

## Change Carets

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

**Format** ~CCx

This table identifies the parameters for this format:

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

 **Example** • This is an example of how to change the command 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	<i>Accepted Values:</i> any ASCII character <i>Default Value:</i> 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 period ( . ):

```
^XA  
^CD.  
^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** ^CF $\mathit{f}$ ,  $\mathit{h}$ ,  $\mathit{w}$

This table identifies the parameters for this format:

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

Parameter  $\mathit{f}$  specifies the default font for every alphanumeric field. Parameter  $\mathit{h}$  is the default height for every alphanumeric field, and parameter  $\mathit{w}$  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 (^A $\mathit{f}$ ) 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	GENERATED LABEL
<pre>^XA ^CF0,89 ^FO20,50 ^FDA GUIDE TO^FS ^FO20,150 ^FDTHE ZPL II^FS ^FO20,250 ^FDPROGRAMMING^FS ^FO20,350 ^FDLANGUAGE^FS ^XZ</pre>	<p><b>A GUIDE TO THE ZPL II PROGRAMMING LANGUAGE</b></p>

**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

**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.

ZPL II follows the ISO standards for international characters.

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	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>0 = U.S.A. 1</li> <li>1 = U.S.A. 2</li> <li>2 = U.K.</li> <li>3 = Holland</li> <li>4 = Denmark/Norway</li> <li>5 = Sweden/Finland</li> <li>6 = Germany</li> <li>7 = France 1</li> <li>8 = France 2</li> <li>9 = Italy</li> <li>10 = Spain</li> <li>11 = miscellaneous</li> <li>12 = Japan (ASCII with Yen symbol)</li> <li>13 = IBM Code Page 850 (see <a href="#">page 42</a>)</li> <li>14 = 16-bit (Unicode) encoded scalable fonts*</li> <li>15 = Shift-JIS for scalable Japanese fonts**</li> <li>16 = EUC-Kanji for scalable fonts</li> <li>17 = Unicode (for Unicode-encoded fonts)</li> <li>18 to 23 = Reserved</li> <li>24 = 8-bit access to Unicode-encoded fonts</li> </ul> <p><i>Initial Value at power-up:</i> 0</p>
s1 = source 1 (character position to be remapped)	<i>Accepted Values:</i> decimals 0 to 255
d1 = destination 1 (new position for the character referred to in s1)	<i>Accepted Values:</i> decimals 0 to 255

Parameters	Details
s2 = source 2 (character position to be remapped)	<i>Accepted Values:</i> decimals 0 to 255
d2 = destination 2 (new position for the character referred to in s2)	<i>Accepted Values:</i> 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 font's 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.

- ➔ **Example 1** • This example remaps the Euro symbol (21) to the dollar sign value (36). When the dollar sign character is sent to the printer, the Euro symbol prints. The Euro symbol value, 15 hexadecimal, equals 21 decimal, and the dollar sign value, 24 hexadecimal, equals 36 decimal.

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

- ➔ **Example 2** • To print the Euro symbol, a hexadecimal value of 15 is placed in the field data (^FD) command. A field hexadecimal (^FH) command must precede the ^FD command.

ZPL II CODE	GENERATED LABEL
<pre>^XA ^FO10,10 ^A0,100 ^FH^FD_15^FS ^XZ</pre>	

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

**International Character Sets**

Hex	2	3	4	5	5	5	5	6	7	7	7	7
	3	0	0	B	C	D	E	0	B	C	D	E
CI0	#	0	@	[	Φ	]	^	'	{		}	~
CI1	#	0	@	¼	Φ	¾	^	'	¼	½	¾	~
CI2	£	0	@	[	Φ	]	^	'	{		}	~
CI3	f	0	§	[	ij	]	^	'	{	ij	}	~
CI4	#	0	@	Æ	Ø	Å	^	'	æ	ø	å	~
CI5	Ü	0	É	Ä	Ö	À	Ü	é	ä	ö	à	ü
CI6	#	0	§	Ä	Ö	Ü	^	'	ä	ö	ü	β
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	§	ì	Ñ	¿	^	'	{	ñ	ç	~
CI11	£	0	É	Ä	Ö	Ü	^	'	ä	ë	ï	ö
CI12	#	0	@	[	¥	]	^	'	{		}	~
CI13	#	0	@	[	\	]	^	'	{		}	~

**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 being forced to alter or recreate 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 letter designation	<i>Accepted Values:</i> B :, E :, R :, A :, and NONE <i>Default Value:</i> B :
b = memory alias letter designation	<i>Accepted Values:</i> B :, E :, R :, A :, and NONE <i>Default Value:</i> E :
c = memory alias letter designation	<i>Accepted Values:</i> B :, E :, R :, A :, and NONE <i>Default Value:</i> R :
d = memory alias letter designation	<i>Accepted Values:</i> B :, E :, R :, A :, and NONE <i>Default Value:</i> A :

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

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

➔ **Examples** • 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
```

This example resets all letter designations to point to themselves.

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

This example sets all letter designation to point to themselves.

```
^XA
^CM , , R , A
^JUS
^XZ
```

**Comments** 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.

# ^CO

## Cache On

**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 22K 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. Their sizes are selected by the user.

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 300K.

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	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> Y
b = amount of additional memory to be added to cache (in K)	<i>Accepted Values:</i> any size up to total memory available <i>Default Value:</i> 40
c = cache type	<i>Accepted Values:</i> 0 = cache buffer (normal fonts) 1 = internal buffer (recommended for Asian fonts) <i>Default Value:</i> 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	<i>Accepted Values:</i> any ASCII character <i>Default Value:</i> 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
+DGR:GRAPHIC.GRF,04412,010
```

# ^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:* Y (yes) or N (no)  
*Default Value:* N

➔ **Example** • The examples below show the error labels <sup>^</sup>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
<sup>^</sup> XA <sup>^</sup> CVY <sup>^</sup> FO50, 50 <sup>^</sup> BEN, 100, Y, N <sup>^</sup> FD97823456 890 <sup>^</sup> FS <sup>^</sup> XZ	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">INVALID - C</div>
<sup>^</sup> XA <sup>^</sup> CVY <sup>^</sup> FO50, 50 <sup>^</sup> BEN, 100, Y, N <sup>^</sup> FD9782345678907 <sup>^</sup> FS <sup>^</sup> XZ	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">INVALID - E</div>
<sup>^</sup> XA <sup>^</sup> CVY <sup>^</sup> FO50, 50 <sup>^</sup> BEN, 100, Y, N <sup>^</sup> FD97823456789081 <sup>^</sup> FS <sup>^</sup> XZ	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">INVALID - L</div>
<sup>^</sup> XA <sup>^</sup> CVY <sup>^</sup> FO50, 50 <sup>^</sup> BEN, 100, Y, N <sup>^</sup> FD97823456789 <sup>^</sup> FS <sup>^</sup> XZ	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">INVALID - S</div>
<sup>^</sup> XA <sup>^</sup> CVY <sup>^</sup> FO50, 50 <sup>^</sup> BQN2, 3 <sup>^</sup> FDHM, BQRCODE-22 <sup>^</sup> FS <sup>^</sup> XZ	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">INVALID - P</div>





**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.

# <sup>^</sup>CW

## Font Identifier

**Description** All built-in fonts are referenced using a one-character identifier. The <sup>^</sup>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** <sup>^</sup>CW $a, d: o . x$

This table identifies the parameters for this format:

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



**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 as additional 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
```

DIRECTORY OF R:*. *	
R:NEWFONT.FNT	65268
R:MYFONT.FNT	65268
582164 BYTES FREE R:	

**Label Listing Before Assignment**

DIRECTORY OF R:*. *	
F R:NEWFONT.FNT	65268
AQ R:MYFONT.FNT	65268
582164 BYTES FREE R:	

**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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = name of font	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .FNT
a = orientation of native font	<i>Fixed Value:</i> normal
h = maximum height of cell (in dots)	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> a value must be specified
w = maximum width of cell (in dots)	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> a value must be specified
base = dots from top of cell to character baseline	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> a value must be specified

Parameters	Details
space = width of space or non-existent characters	<p><i>Accepted Values:</i> 0 to 32000</p> <p><i>Default Value:</i> a value must be specified</p>
#char = number of characters in font	<p><i>Accepted Values:</i> 1 to 256 (must match the characters being downloaded)</p> <p><i>Default Value:</i> a value must be specified</p>
© = copyright holder	<p><i>Accepted Values:</i> 1 to 63 alphanumeric characters</p> <p><i>Default Value:</i> a value must be specified</p>
data = structured ASCII data that defines each character in the font	<p>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.</p> <p>The data structure is:</p> <p>#xxxx.h.w.x.y.i.data</p> <p>#xxxx = character code</p> <p>h = bitmap height (in dot rows)</p> <p>w = bitmap width (in dot rows)</p> <p>x = x-offset (in dots)</p> <p>y = y-offset (in dots)</p> <p>i = typesetting motion displacement (width, including inter character gap of a particular character in the font)</p> <p>data = hexadecimal bitmap description</p>

 **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.
```

```
O0FF
```

```
O0FF
```

```
F000
```

```
F000
```

```
FFFF
```

```
#0037.4.24.3.6.26.
```

```
O0FF00
```

```
OFOOFO
```

```
OFOOFO
```

```
O0FF00
```

# ^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 ZTools for Windows and downloaded with the ~DE command.

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

This table identifies the parameters for this format:

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



# ^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	<i>Accepted Value:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = image name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .ZPL

# ~DG

## Download Graphics

**Description** The ~DG command performs these functions:

- Puts the printer into graphics mode.
- Names the graphic (this name is used to recall it into a label).
- Defines the size of the graphic.
- Downloads the hexadecimal string to the printer.

For more saving and loading options when downloading graphics, see ~DY on [page 148](#).

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

This table identifies the parameters for this format:

Parameters	Details
d = device to store image	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = image name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .GRF
t = total number of bytes in graphic	See the formula below.
w = number of bytes per row	See the formula 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.

→ **Example** • The key for these examples is as follows:

$x$  = 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

**Example** • The  $t$  parameter can be determined using this formula:

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

**Example** • For example, to determine the correct  $t$  parameter for a graphic 8 mm wide, 16 mm high, and a print density of 8 dots/mm, the formula works this way:

$$\left(\frac{8 \times 8}{8}\right) \times (16 \times 8) = \text{bytes}$$

$$8 \times 128 = 1024$$

$$t = 1024$$

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

The  $w$  parameter (the width in terms of bytes per row) can be determined by using this formula:

$$\left(\frac{xz}{8}\right) = \text{totalbytes/row}$$

$$w = 8$$

For example, to determine the correct  $w$  parameter for a graphic 8 mm wide and a print density of 8 dots/mm, the formula works this way:

$$\left(\frac{8 \times 8}{8}\right) = 8 \text{ bytes}$$
$$w = 8$$

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

Parameter  $w$  is the first value in the  $\tau$  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 to be created should be white and the next four black, the dot-by-dot binary code would be 00001111. The hexadecimal representation of this binary value would be 0F. The entire graphic image is coded in this way. The complete graphic image is sent as one continuous string of hexadecimal values.

**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 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. ~DS is used for downloading Intellifont data to the printer. For downloading TrueType fonts, see the ~DT command on [page 145](#).

The ~DS command, and its associated parameters, is the result of converting a vendor-supplied font for use on a Zebra printer. The conversion is done using the Zebra utility program ZTools for Windows, available from Zebra Technologies.

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

This table identifies the parameters for this format:

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



**Example** • This example shows the first three lines of a scalable font that has been converted using the ZTools for Windows 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  
FFOAECB28FFF00FF
```

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

# ~DT


## Download TrueType Font

**Description** The ZTools for Windows program must be used to convert a TrueType font to a Zebra-downloadable format. ZTools for Windows creates a downloadable file that includes a ~DT command. For information on converting and downloading Intellifont information, see the ~DS command on [page 143](#).

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

This table identifies the parameters for this format:

Parameters	Details
d = font location	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = font name	<i>Accepted Values:</i> any valid TrueType name, up to 8 characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .DAT
s = font size	<i>Accepted Values:</i> the number of memory bytes required to hold the Zebra-downloadable format of the font <i>Default Value:</i> if an incorrect value or no value is entered, the command is ignored
data = data string	<i>Accepted Values:</i> a string of ASCII hexadecimal values (two hexadecimal digits/byte). The total number of two-digit values must match parameter s. <i>Default Value:</i> 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. The ZTools for Windows program must be used 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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = font name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .FNT
s = font size	<i>Accepted Values:</i> the number of memory bytes required to hold the Zebra-downloadable format of the font <i>Default Value:</i> if no data is entered, the command is ignored
data = data string	<i>Accepted Values:</i> a string of ASCII hexadecimal values (two hexadecimal digits/byte). The total number of two-digit values must match parameter s. <i>Default Value:</i> 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)

# ~DY

## Download Graphics

**Description** ~DY downloads to the printer graphic objects in any supported format. This command can be used in place of ~DG for more saving and loading options.

**Format** ~DYf,b,x,t,w,data

This table identifies the parameters for this format:

Parameters	Details
f = font name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
b = format downloaded in data field (f)	<i>Accepted Values:</i> A = uncompressed bitmap (.GRF, ASCII) B = uncompressed bitmap (.GRF, binary) C = AR-compressed bitmap (.GRF, compressed binary—used only by Zebra’s BAR-ONE <sup>®</sup> software) P = PNG image (.PNG) <i>Default Value:</i> a value must be specified
x = extension of stored graphic	<i>Accepted Values:</i> G = raw bitmap (.GRF) P = store as compressed (.PNG) <i>Default Value:</i> .GRF, unless parameter b is set to P (.PNG)
t = total number of bytes in graphic	.GRF images: the size after decompression into memory .PNG images: the size of the .PNG file

Parameters	Details
w = total number of bytes per row	.GRF images: number of bytes per row .PNG images: value ignored—data is encoded directly into .PNG data
data = data	ASCII hexadecimal encoding, ZB64, or binary data, depending on b a, p = ASCII hexadecimal or ZB64 b, c = binary  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.

**Comments** For more information on ZB64 encoding and compression, see Appendix I in *ZPL Programming Guide Volume Two*.

# ~EF

## Erase Stored Formats

**Description** The ~EF command erases all stored formats.

**Format** ~EF

**Comments** The ~EF command is no longer recommended for use. It is recommended that the ^ID (Object Delete) command be used to selectively delete stored formats.

# ~EG

## Erase Download Graphics

See *^ID on page 205*.

# ^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)	<p><i>Accepted Values:</i> 0 to the width of the label (or 9999)</p> <p><i>Default Value:</i> 0</p> <p>If the value is less than font width or not specified, text does not print.</p>
b = maximum number of lines in text block	<p><i>Accepted Values:</i> 1 to 9999</p> <p><i>Default Value:</i> 1</p> <p>Text exceeding the maximum number of lines overwrites the last line. Changing the font size automatically increases or decreases the size of the block.</p>
c = add or delete space between lines (in dots)	<p><i>Accepted Values:</i> -9999 to 9999</p> <p><i>Default Value:</i> 0</p> <p>Numbers are considered to be positive unless preceded by a minus sign. Positive values add space; negative values delete space.</p>

Parameters	Details
d = text justification	<p><i>Accepted Values:</i> L (left), C (center), R (right), J (justified)</p> <p><i>Default Value:</i> L</p> <p>Last line is left-justified if J is used.</p>
e = hanging indent (in dots) of the second and remaining lines	<p><i>Accepted Values:</i> 0 to 9999</p> <p><i>Default Value:</i> 0</p>

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

ZPL II CODE	GENERATED LABEL
<pre>^XA ^CF0,30,30^FO25,50 ^FB250,4,, ^FDFD command that IS preceded by an FB command.^FS ^XZ</pre>	<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p><b>FD command that IS preceded by an FB command.</b></p> </div>
<pre>^XA ^CF0,30,30^FO25,50 ^FDFD command that IS NOT preceded by an FB command.^FS ^XZ</pre>	<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p><b>FD command that IS NOT preceded by an FB cor</b></p> </div>

### 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 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. Placing a soft hyphen with a word controls 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.
- If ^SN is used instead of ^FD, the field does not print.
- ^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	<i>Accepted Values:</i> any ASCII character <i>Default Value:</i> %
b = secondary clock indicator character	<i>Accepted Values:</i> any ASCII character <i>Default Value:</i> none—this value cannot be the same as a or c
c = tertiary clock indicator character	<i>Accepted Values:</i> any ASCII character <i>Default Value:</i> none—this value cannot be the same as a or b

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

ZPL II CODE	GENERATED LABEL
<pre>^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%,{,# ^FDTertiary: #m/#d/#y^FS ^XZ</pre>	<p><b>Primary: 05/06/01</b></p> <p><b>Secondary: 05/06/01</b></p> <p><b>Tertiary: 05/06/01</b></p>

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

# ^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	<i>Accepted Values:</i> any ASCII string up to 3072 characters <i>Default Value:</i> none—a string of characters must be entered

**Comments** The ^ and ~ characters can be printed by changing the prefix characters—see the ~CC and ~CT commands. 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** ^FH`a`

This table identifies the parameters for this format:

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

➔ **Example** • This is an example of how to enter a hexadecimal value directly into a ^FD statement:

ZPL II CODE	GENERATED LABEL
<pre>^XA ^FO100,100 ^AD^FH ^FD~_7e used for HEX^FS ^XZ</pre>	<pre>Tilde ~ used for HEX</pre>
<pre>^XA ^FO100,100 ^AD^FH\ ^FD~\7E used for HEX^FS ^XZ</pre>	<pre>Tilde ~ used for HEX</pre>

# ^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:

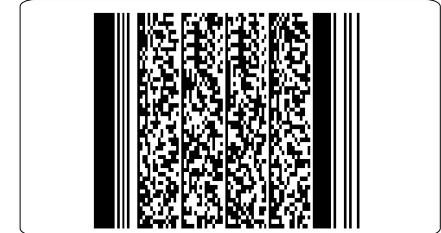
Parameters	Details
x1 = x-axis location of first symbol (in dots)	<p><i>Accepted Values:</i></p> <p>0 to 32000</p> <p>e = exclude this bar code from printing</p> <p><i>Default Value:</i> a value must be specified</p>
y1 = y-axis location of first symbol (in dots)	<p><i>Accepted Values:</i></p> <p>0 to 32000</p> <p>e = exclude this bar code from printing</p> <p><i>Default Value:</i> a value must be specified</p>

Parameters	Details
x2 = x-axis location of second symbol (in dots)	same as x1 parameter
y2 = y-axis location of second symbol (in dots)	same as y1 parameter
... = continuation of X,Y pairs	<i>Maximum number of pairs: 60</i>

➔ **Example 1** • This example assumes a maximum of three bar codes:

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FM100,100,200,200,300,300 ^B7N,5,5,,50,N^FD&lt;data&gt;^FS ^XZ                     </pre>	

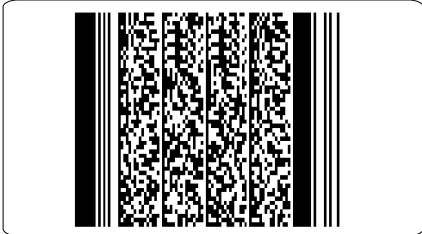
➔ **Example 2** • This example assumes a maximum of three bar codes, with bar code 2 of 3 omitted:

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FM100,100,e,e,300,300 ^B7N,5,5,,50,N^FD&lt;data&gt;^FS ^FS                     </pre>	

If e is entered for any of the x, y values, the bar code does not print.



➔ **Example 3** • Symbol 2 of 3 in this example is still excluded:

ZPL II CODE	GENERATED LABEL
<pre>^XA ^FM100,100,200,200,300,300 ^B7N,5,5,,50,N^FD&lt;data&gt;^FS ^XZ</pre>	

**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 is used to number data fields. This command is used in both ^DF (Store Format) and ^XF (Recall Format) commands.

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

**Format** ^FN#

This table identifies the parameters for this format:

Parameters	Details
# = number to be assigned to the field	<i>Accepted Values:</i> 0 to 9999 <i>Default Value:</i> 0

➔ **Example** • This example recalls the format (^XF) saved with ^DF and inserts field number data.

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^XFR:STOREFMT.ZPL^FS ^FN1^FDZEBRA^FS ^FN2^FDLABEL^FS ^XZ </pre>	<pre> ZEBRA LABEL BUILT BY ZEBRA </pre>

### Comments

- The same ^FN value can be stored with several different fields.
- If a label format contains a field with the ^FN command and the ^FD command, 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** ^FO $x, y$

This table identifies the parameters for this format:

Parameters	Details
$x$ = x-axis location (in dots)	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> 0
$y$ = y-axis location (in dots)	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> 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 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	<i>Accepted Values:</i> H = horizontal printing (left to right) V = vertical printing (top to bottom) R = reverse printing (right to left) <i>Default Value:</i> H
g = additional inter-character gap (in dots)	<i>Accepted Values:</i> 0 to 9999 <i>Default Value:</i> 0 if no value is entered

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

ZPL II CODE	GENERATED LABEL
<pre data-bbox="545 512 769 684">^XA ^FO100,50 ^FPV,10 ^AV ^FDvertical^FS ^XZ</pre>	<pre data-bbox="948 499 964 701">v e r t i c a l</pre>
<pre data-bbox="545 770 753 942">^XA ^FO350,50 ^FPR,10 ^AV ^FDreverse^FS ^XZ</pre>	<pre data-bbox="924 783 1016 804">esrever</pre>

**Comments** When using reverse printing, the origin specified in ^FT is the lower-left corner of the right-most text character.


# ^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	GENERATED LABEL
<pre> ^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 </pre>	

**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 also 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.

**Format** ^FT $x, y$

This table identifies the parameters for this format:

Parameters	Details
$x$ = x-axis location (in dots)	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> position after last formatted text field
$y$ = y-axis location (in dots)	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> position after last formatted text field

**Text** The origin is at the start of the character string, at the baseline of the font. Normally the baseline is the bottom of most characters, except for those with descenders, such as g, y, et cetera.

**Bar Codes** The origin is at the base of the bar code, even when an interpretation is present below the bar code, or if the bar code has guard bars.

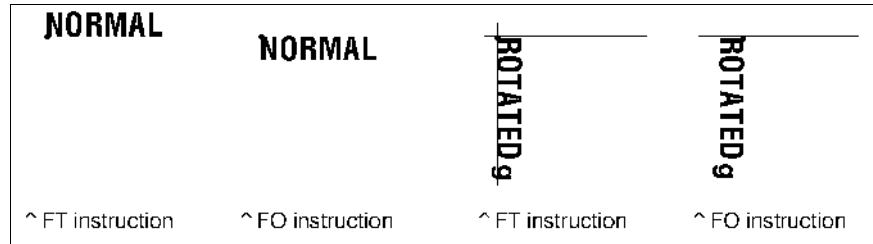
**Graphic Boxes** The origin is at the bottom-left corner of the box.

**Images** The origin is at the bottom-left corner of the rectangular image area.

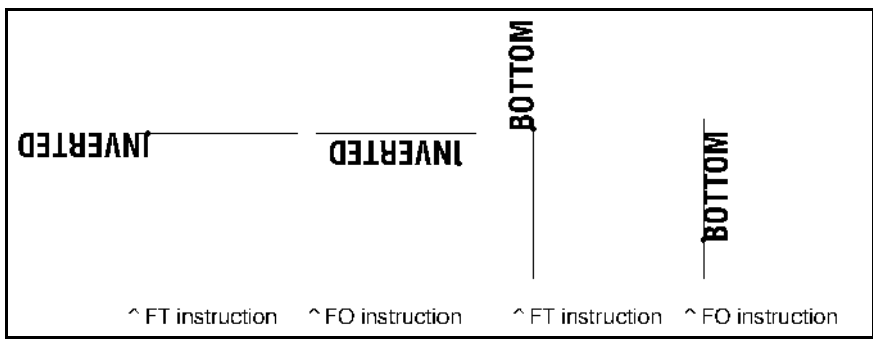
➔ **Example** • The example below shows the differences in font orientation when using <sup>^</sup>FT and <sup>^</sup>FO relative to their <sup>^</sup>LH position. The origin point of the font when using the <sup>^</sup>FT command is always at the left of the baseline position of the first element or character in the field.

In normal orientation, all characters rest on the baseline. In rotated orientation, all characters are drawn to the right of the label from the baseline. In inverted orientation, all characters are drawn down from the baseline and printed to the left. In bottom orientation, all characters are drawn towards the left of the label from the baseline and printed to the right. The *dot* shows the origin point for both the <sup>^</sup>FT and <sup>^</sup>FO font orientations.


**Scalable Normal Font Orientation, Scalable Rotated Font Orientation**



**Scalable Inverted Font Orientation, Scalable Bottom-up Font Orientation**



➔ **Example** • This is an example of the ^FT command and concatenation:

ZPL II CODE	GENERATED LABEL
<pre> ^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 </pre>	

**Comments** 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

# ^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	<i>Accepted Values:</i> 0 to 3072 character string <i>Default Value:</i> 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	GENERATED LABEL
<pre> ^XA ^FO40,40 ^GB300,203,8^FS ^FO55,60^CF0,25 ^FVVARIABLE DATA #1^FS ^FO80,150 ^FDFIXED DATA^FS ^MCN ^XZ </pre>	
<pre> ^XA ^FO55,60^CF0,25 ^FVVARIABLE DATA #2^FS ^MCY ^XZ </pre>	

**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** ^FW $\mathit{r}$

This table identifies the parameters for this format:

Parameters	Details
$\mathit{r}$ = rotate field	<p><i>Accepted Value:</i></p> <p>N = normal</p> <p>R = rotated 90 degrees</p> <p>I = inverted 180 degrees</p> <p>B = bottom-up 270 degrees, read from bottom up</p> <p><i>Initial Value at Power-up:</i> N</p>

**Comments** If the ^FW command is entered with the  $\mathit{r}$  parameter missing, the command is ignored.

^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

➔ **Example** • This is an example of how to use the ^FX command effectively.

ZPL II CODE	GENERATED LABEL
<pre> ^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 </pre>	

**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)	<i>Accepted Values:</i> value of t to 32000 <i>Default Value:</i> value used for thickness (t) or 1
h = box height (in dots)	<i>Accepted Values:</i> value of t to 32000 <i>Default Value:</i> value used for thickness (t) or 1
t = border thickness (in dots)	<i>Accepted Values:</i> 1 to 32000 <i>Default Value:</i> 1
c = line color	<i>Accepted Values:</i> B (black) or W (white) <i>Default Value:</i> B
r = degree of corner-rounding	<i>Accepted Values:</i> 0 (no rounding) to 8 (heaviest rounding) <i>Default Value:</i> 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.

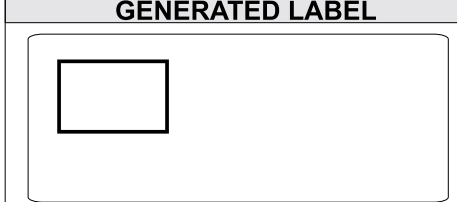
The roundness-index is used to determine a rounding-radius for each box. Formula:

$$\text{rounding-radius} = (\text{rounding-index} / 8) * (\text{shorter side} / 2)$$

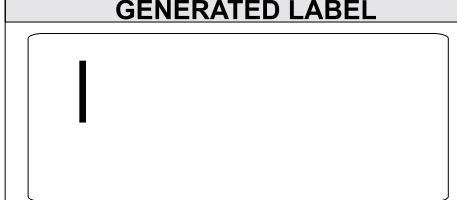
where the shorter side is the lesser of the width and height (after adjusting for minimum and default values).

**Examples** • Here are a few examples:

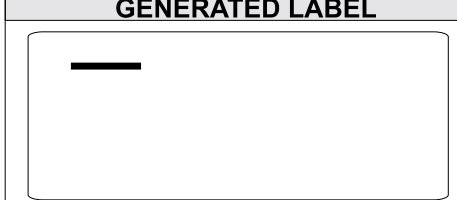
Width: 1.5 inch; Height: 1 inch; Thickness: 10; Color: default; Rounding: default

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FO50,50 ^GB300,200,10^FS ^XZ                     </pre>	

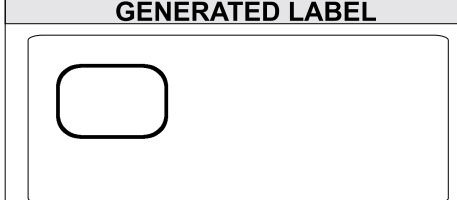
Width: 0 inch; Height: 1 inch; Thickness: 20; Color: default; Rounding: default:

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FO50,50 ^GB0,203,20^FS ^XZ                     </pre>	

Width: 1 inch; Height: 0 inch; Thickness: 30; Color: default; Rounding: default

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FO50,50 ^GB203,0,20^FS ^XZ                     </pre>	

Width: 1.5 inch; Height: 1 inch; Thickness: 10; Color: default; Rounding: 5

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FO50,50 ^GB300,200,10,,5^FS ^XZ                     </pre>	

# ^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)	<i>Accepted Values:</i> 3 to 4095 (larger values are replaced with 4095) <i>Default Value:</i> 3
t = border thickness (in dots)	<i>Accepted Values:</i> 2 to 4095 <i>Default Value:</i> 1
c = line color	<i>Accepted Values:</i> B (black) or W (white) <i>Default Value:</i> B

➔ **Example** • This is an example of how to create a circle on the printed label:

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^FO50,50 ^GC250,10,B^FS ^XZ </pre>	

# <sup>^</sup>GD

## Graphic Diagonal Line

**Description** The <sup>^</sup>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.

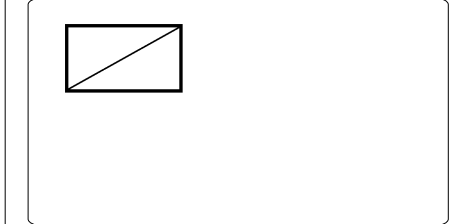
**Format** <sup>^</sup>GBw,h,t,c,o

This table identifies the parameters for this format:

Parameters	Details
w = box width (in dots)	<i>Accepted Values:</i> 3 to 32000 <i>Default Value:</i> value of t (thickness) or 1
h = box height (in dots)	<i>Accepted Values:</i> 3 to 32000 <i>Default Value:</i> value of t (thickness) or 1
t = border thickness (in dots)	<i>Accepted Values:</i> 1 to 32000 <i>Default Value:</i> 1
c = line color	<i>Accepted Values:</i> B (black) or W (white) <i>Default Value:</i> B
o = orientation (direction of the diagonal)	<i>Accepted Values:</i> R (or /) = right-leaning diagonal L (or \) = left-leaning diagonal <i>Default Value:</i> 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	GENERATED LABEL
<pre>^XA ^FO150,100 ^GB350,203,10^FS ^FO155,110 ^GD330,183,10,,R^FS ^XZ</pre>	

# <sup>^</sup>GE

## Graphic Ellipse


**Description** The <sup>^</sup>GE command produces an ellipse in the label format.

**Format** <sup>^</sup>GE $w, h, t, c$

This table identifies the parameters for this format:

Parameters	Details
w = ellipse width (in dots)	<i>Accepted Values:</i> 3 to 4095 (larger values are replaced with 4095) <i>Default Value:</i> value used for thickness (t) or 1
h = ellipse height (in dots)	<i>Accepted Values:</i> 3 to 4095 <i>Default Value:</i> value used for thickness (t) or 1
t = border thickness (in dots)	<i>Accepted Values:</i> 2 to 4095 <i>Default Value:</i> 1
c = line color	<i>Accepted Values:</i> B (black) or W (white) <i>Default Value:</i> B

➔ **Example** • This is an example of how to create a ellipse on a printed label:

ZPL II CODE	GENERATED LABEL
<pre> <sup>^</sup>XA <sup>^</sup>FO100,100 <sup>^</sup>GE300,100,10,B<sup>^</sup>FS <sup>^</sup>XZ </pre>	

# ^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	<p><i>Accepted Values:</i></p> <p>A = ASCII hexadecimal (follows the format for other download commands)</p> <p>B = binary (data sent after the c parameter is strictly binary)</p> <p>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.)</p> <p><i>Default Value:</i> A</p>
b = binary byte count	<p><i>Accepted Values:</i> 1 to 99999</p> <p>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.</p> <p><i>Default Value:</i> command is ignored if a value is not specified</p>

Parameters	Details
c = graphic field count	<p><i>Accepted Values:</i> 1 to 99999</p> <p>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).</p> <p><i>Default Value:</i> command is ignored if a value is not specified</p>
d = bytes per row	<p><i>Accepted Values:</i> 1 to 99999</p> <p>This is the number of bytes in the downloaded data that comprise one row of the image.</p> <p><i>Default Value:</i> command is ignored if a value is not specified</p>
data = data	<p><i>Accepted Values:</i></p> <p>ASCII hexadecimal data: 00 to FF</p> <p>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.</p> <p><b>Binary data:</b> 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.</p>

➔ **Example 1** • 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 2** • 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

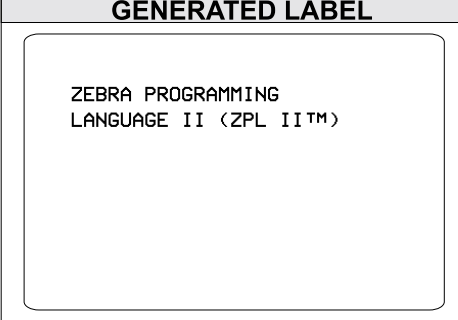
**Description** The ^GS command enables you to generate the registered trademark, copyright symbol, and other symbols.

**Format** ^GS $\circ$ , h, w

This table identifies the parameters for this format:

Parameters	Details
$\circ$ = font orientation	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>N = normal</li> <li>R = rotate 90 degrees clockwise</li> <li>I = inverted 180 degrees</li> <li>B = bottom-up, 270 degrees</li> </ul> <p><i>Default Value:</i> N or last ^FW value</p>
h = character height proportional to width (in dots)	<p><i>Accepted Values:</i> 0 to 32000</p> <p><i>Default Value:</i> last ^CF value</p>
w = character width proportional to height (in dots)	<p><i>Accepted Values:</i> 0 to 32000</p> <p><i>Default Value:</i> last ^CF value</p>

➔ **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	GENERATED LABEL
<pre> ^XA^CFD ^FO50,50 ^FDZEBRA PROGRAMMING^FS ^FO50,75 ^FDLANGUAGE II (ZPL II )^FS ^FO280,75 ^GS^FDC^FS ^XZ </pre>	

**A = ® (Registered Trade Mark)**

**B = © (Copyright)**

**C = ™ (Trade Mark)**

**D = U (Underwriters Laboratories approval)**

**E = S (Canadian Standards Association approval)**

# ~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 is terminated by an <ETX><CR><LF> control code sequence.

**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 Temperature Information

**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
```

# <sup>^</sup>HF

## Graphic Symbol

**Description** The <sup>^</sup>HF command sends an object of ZPL format instructions to the host, in ~DF.

**Format** <sup>^</sup>HF , o , h , w

This table identifies the parameters for this format:

Parameters	Details
o = font orientation	<i>Accepted Values:</i> N = normal R = rotate I = inverted 180 degrees B = bottom-up, 270 degrees <i>Default Value:</i> N or last <sup>^</sup> FW value.
h = character height proportional to width (in dots)	<i>Accepted Value:</i> 0 to 32000 <i>Default Value:</i> last <sup>^</sup> CF value
w = character width proportional to height (in dots)	<i>Accepted Value:</i> 0 to 32000 <i>Default Value:</i> last <sup>^</sup> CF value

# ^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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> search priority
o = object name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .GRF

**Comments** For more information on uploading graphics, see [^HY on page 202](#).

# ^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
--        RESOLUTION
864 8/MM FULL FIRMWARE
U32.10.2 <-  HARDWARE ID
U2.2.6.98.A CONFIGURATION
CUSTOMIZED
1024.....R: RAM
8192.....B: MEMORY CARD
0768.....E: ONBOARD FLASH
NONE      FORMAT CONUERT
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

**Parameters** 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  
|  
2 —|1024,0780,0780 |— 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 ~HS is sent to the printer, three data strings are sent back to the host. Each string starts with an <STX> control code and is terminated by an <ETX><CR><LF> control code sequence. To avoid confusion, each string is printed on a separate line by the host.

### 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 parameter 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 <sup>8</sup> a <sup>7</sup> a <sup>6</sup> a <sup>5</sup> a <sup>4</sup> a <sup>3</sup> a <sup>2</sup> a <sup>1</sup> a <sup>0</sup>	
a <sup>7</sup> = Handshake 0 = Xon/Xoff 1 = DTR	a <sup>8</sup> a <sup>2</sup> a <sup>1</sup> a <sup>0</sup> = Baud
a <sup>6</sup> = Parity Odd/Even 0 = Odd 1 = Even	0 000 = 110 0 001 = 300 0 010 = 600 0 011 = 1200 0 100 = 2400 0 101 = 4800 0 110 = 9600 0 111 = 19200
a <sup>5</sup> = Disable/Enable 0 = Disable 1 = Enable	1 000 = 28800 <i>(available only on certain printer models)</i> 1 001 = 38400 <i>(available only on certain printer models)</i> 1 010 = 57600 <i>(available only on certain printer models)</i> 1 011 = 14400
a <sup>4</sup> = Stop Bits 0 = 2 Bits 1 = 1 Bit	
a <sup>3</sup> = Data Bits 0 = 7 Bits 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

0 = Rewind

1 = Peel-Off

2 = Tear-Off

3 = Cutter

4 = Applicator

---

---

s = print width mode  
 t = *label waiting* flag (1 = label waiting in Peel-off Mode)  
 uuuuu = labels remaining in batch  
 uuu  
 v = *format while printing* flag (always 1)  
 www = number of graphic images stored in memory

---

\* This parameter 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 0 = Die-Cut 1 = Continuous				m4 m3 m2 m1 = Unused 0 = Off 1 = On			
m6 = Sensor Profile 0 = Off				m0 = Print Mode 0 = Direct Thermal 1 = Thermal Transfer			
m5 = Communications Diagnostics 0 = Off 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 AlertConfiguration

**Description** This command returns the table of configured ZebraNet Alertsettings to the host.

**Format** ~HU

→ **Example** • If the ~HU command is sent to the printer with existing Alertmessages set to go to e-mail and SNMP traps, the data returned would look something like the information below. See ^SX 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
```

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 Alertcondition as defined in the ^SX command.

# ^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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = object name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> asterisk (*). A question mark (?) can also be used.
x = extension	<i>Accepted Values:</i> any extension conforming to Zebra conventions <i>Default Value:</i> asterisk (*). A question mark (?) can also be used.

➔ **Example** • Listed is an example of the <sup>^</sup>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:○.x

This table identifies the parameters for this format:

Parameters	Details
d = location of object	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> search priority
○ = object name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> an object name must be specified
x = extension	<i>Accepted Values:</i> G = .GRF (raw bitmap format) P = .PNG (compressed bitmap format) <i>Default Value:</i> 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. For more information, see Chapter 6 (XML: Super Host Status) in *ZPL Programming Guide Volume Two*.

**Format** ^HZd

This table identifies the parameters for this format:

Parameters	Details
d = display description to return	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>a = display all information</li> <li>f = display printer format setting information</li> <li>l = display object directory listing information</li> <li>o = display individual object data information</li> </ul> <p>Object information can be recalled from R:, E:, B:, and A:. The object name and extension follow the standard Zebra naming conventions (see the Comments below). This parameter must be followed by a comma, the drive location, object name, and object extension:</p> <pre style="margin-left: 40px;">^XA ^HZO , R : SAMPLE . GRF ^XZ</pre> <ul style="list-style-type: none"> <li>r = display printer status information</li> </ul> <p><i>Default Value:</i> if the value is missing or invalid, the command is ignored</p>



**Comments** Supported extensions for objects (parameter o) include:

- .FNT — font
- .GRF — graphic
- .PNG — compressed graphic
- .ZPL — stored format
- .DAT — encoding table
- .ZOB — downloadable object
- .STO — Alertdata file
- .BAZ — security for all ZBI executables
  - For more details on .BAZ, see [Encrypting ZBI Files on page 386](#).

# ^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 prior to 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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = object name	<i>Accepted Values:</i> any 1 to 8 character name <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Accepted Values:</i> any extension conforming to Zebra conventions <i>Default Value:</i> .GRF

 **Example 1** • To delete stored formats from DRAM:

```
^XA
^IDR:* .ZPL^FS
^XZ
```

- **Example 2** • To delete formats and images named SAMPLE from DRAM, regardless of the extension:

```
XA
^IDR: SAMPLE.*^FS
^XZ
```

- **Example 3** • 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 4** • In this example, 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:.

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 ^FOO , 0.



**Important** • See [^IS on page 211](#).

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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = object name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .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 command.

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^ILR:SAMPLE2.GRF^FS ^CFD,36,20 ^FO15,210 ^FD900123^FS ^FO218,210 ^FDLINE 12^FS ^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 </pre>	

# ^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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> search priority
o = object name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Fixed Value:</i> .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 207](#).

**Format** ^ISd: o . x , p

This table identifies the parameters for this format:

Parameters	Details
d = location of stored object	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = object name	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension	<i>Accepted Values:</i> .GRF or .PNG <i>Default Value:</i> .GRF
p = print image after storing	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> 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	GENERATED LABEL
<pre> ^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 </pre>	

# ~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 the two 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	<p><i>Acceptable Values:</i></p> <p>B = Flash card (PCMCIA)</p> <p>E = internal Flash memory</p> <p><i>Default Value:</i> a device must be specified</p>

➔ **Example** • This is an example of initializing the different types of flash memory:

^JBB – initializes the optional Flash card when installed in the printer.

^JBE – initializes the optional Flash memory when installed in the printer.

^JBA – initializes initial Compact Flash memory when installed in 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*<sup>TM</sup> 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	<i>Accepted Values:</i> Y (pause on low voltage) or N (do not pause) N is suggested when the printer is powered by the Car Battery Adapter. <i>Default Value:</i> Y

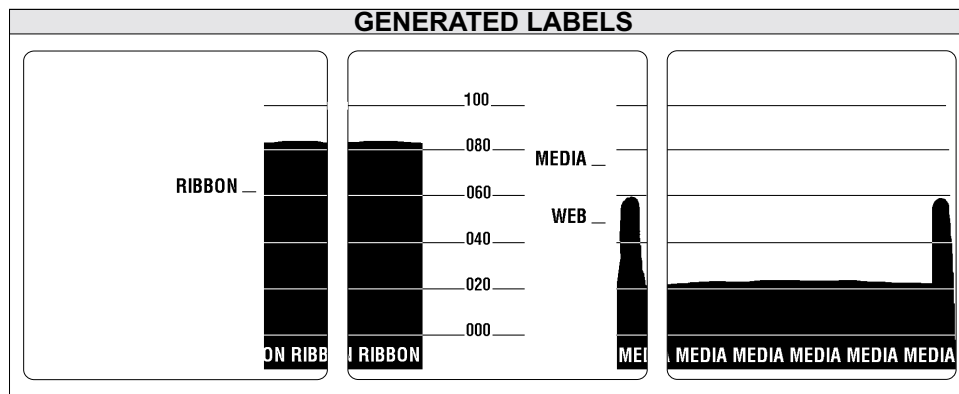
# ~JG

## 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:



# ^JI

## Start ZBI (Zebra BASIC Interpreter)

**Description** ^JI works much like the ~JI command. Both commands are sent to the printer to initialize the Zebra BASIC Interpreter.

In interactive mode, ^JI can be sent through one of the communication ports (serial, parallel, or Ethernet) to initialize the printer to receive ZBI commands. This command can be sent from one of the Zebra software utilities, such as ZTools, or from a standard PC program, such as Hyper terminal.

When the command is received, the printer responds by sending a ZBI header back to the console, along with the program version number. This indicates that the interpreter is active.

**Format** ^JI`d:o.x,b,c,d`

This table identifies the parameters for this format:

Parameters	Details
d = location of program to run after initialization	<i>Acceptable Values:</i> R:, E:, B:, and A: <i>Default Value:</i> location must be specified
o = name of program to run after initialization	<i>Accepted Values:</i> any valid program name <i>Default Value:</i> name must be specified
x = extension of program to run after initialization	<i>Fixed Value:</i> .BAS
b = console control	<i>Accepted Values:</i> Y (console on) or N (console off) <i>Default Value:</i> Y

Parameters	Details
c = echoing control	<i>Accepted Values:</i> Y (echo on) or N (echo off) <i>Default Value:</i> Y
d = memory allocation for ZBI	<i>Accepted Values:</i> 20K to 1024K <i>Default Value:</i> 50K

**Comments** When the printer is turned on, it can receive ZPL II commands and label formats. However, for the printer to recognize ZBI commands and programs, it must be initialized using ^JI or ~JI.

Only one ZBI interpreter can be active in the printer at a time. If a second ^JI or ~JI command is received while the interpreter is running, the command is ignored.

The interpreter is deactivated by entering one of two commands:

ZPL at the ZBI prompt

~JQ at an active ZPL port

# ~JI

## Start ZBI (Zebra BASIC Interpreter)

**Description** ~JI works much like the ^JI command. Both commands are sent to the printer to initialize the Zebra BASIC Interpreter.

In interactive mode, ~JI can be sent through one of the communication ports (serial, parallel, or Ethernet) to initialize the printer to receive ZBI commands. This command can be sent from one of the Zebra software utilities, such as ZTools, or from a standard PC program, such as Hypercritical.

When the command is received, the printer responds by sending a ZBI header back to the console, along with the program version number. This indicates that the interpreter is active.

**Format** ~JI

**Comments** While receiving commands, the printer *echoes* the received characters back to the source. This can be toggled on and off with the ZBI ECHO command.

When the printer is turned on, it can receive ZPL II commands and label formats. However, for the printer to recognize ZBI commands and formats, it must be initialized using ^JI or ~JI.

Only one ZBI interpreter can be active in the printer at a time. If a second ~JI or ^JI command is received while the interpreter is running, the command is ignored.

The interpreter is deactivated by entering one of these commands:

ZPL at the ZBI prompt

~JQ at an active ZPL port

# ^JJ

## Set Auxiliary Port

**Description** The ^JJ command allows you to control an online verifier or applicator device.

**Format** ^JJ*a,b,c,d,e,f*



This table identifies the parameters for this format:

Parameters	Details
a = Operational Mode for auxiliary port	<p><i>Accepted Values:</i></p> <p>0 = off</p> <p>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.</p> <p>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.</p> <p><i>Default Value:</i> 0</p>
b = Application Mode	<p><i>Accepted Values:</i></p> <p>0 = off</p> <p>1 = End Print signal normally high, and low only when the printer is moving the label forward.</p> <p>2 = End Print signal normally low, and high only when the printer is moving the label forward.</p> <p>3 = End Print signal normally high, and low for 20 ms when a label has been printed and positioned.</p> <p>4 = End Print signal normally low, and high for 20 ms when a label has been printed and positioned.</p> <p><i>Default Value:</i> 0</p>

Parameters	Details
c = Application Mode start signal print	<p><i>Accepted Values:</i></p> <p>p = Pulse Mode – Start Print signal must be deasserted before it can be asserted for the next label.</p> <p>l = Level Mode – Start Print signal does not need to be deasserted to print the next label. As long as the Start Print signal is low and a label is formatted, a label prints.</p> <p><i>Default Value:</i> 0</p>
d = Application Label Error Mode	<p><i>Accepted Values:</i></p> <p>e = error mode—the printer asserts the <i>Service Required</i> signal (svce_req - pin 10) on the application port, enters into Pause Mode, and displays an error message on the LCD.</p> <p>f = Feed Mode—a blank label prints when the web is not found where expected to sync the printer to the media.</p> <p><i>Default Value:</i> f</p>
e = Reprint Mode	<p><i>Accepted Values:</i></p> <p>e = enabled—printer ignores the Reprint signal.</p> <p>d = disabled—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.</p> <p><i>Default Value:</i> d</p>
f = Ribbon Low Mode	<p><i>Accepted Values:</i></p> <p>e = <i>enabled</i> – printer warning issued when ribbon low.</p> <p>d = <i>disabled</i> – printer warning not issued when ribbon low.</p> <p><i>Default Value:</i> e</p>

# ~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).

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	<i>Accepted Values:</i> A = 24 dots/mm, 12 dots/mm, 8 dots/mm or 6 dots/mm B = 12 dots/mm, 6 dots/mm, 4 dots/mm or 3 dots/mm <i>Default Value:</i> A

➔ **Example** • This example of the affects of alternating the dots per millimeter:

ZPL II CODE	GENERATED LABEL
<pre> ^XA ^JMA^FS ^FO100,100 ^B2N,50,Y,N,N ^FD1234567890^FS ^XZ                     </pre>	
<pre> ^XA ^JMB^FS ^FO100,100 ^B2N,50,Y,N,N ^FD1234567890^FS ^XZ                     </pre>	

**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



# ~JQ

## Terminate Zebra BASIC Interpreter

**Description** The ~JQ command is used when Zebra BASIC Interpreter is active. Sending ~JQ to the printer terminates the ZBI session.

**Format** ~JQ

**Comments** Entering ZPL at the command prompt also terminates a ZBI session.

# ~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

## 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 front panel. When a ~JS command is encountered, it overrides the current front 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	<p><i>Accepted Values:</i></p> <p>A = 100 percent backfeed after printing and cutting</p> <p>B = 0 percent backfeed after printing and cutting, and 100 percent before printing the next label</p> <p>N = normal — 90 percent backfeed after label is printed</p> <p>O = off — turn backfeed off completely</p> <p>10 to 90 = percentage value</p> <p style="text-align: right;">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.</p> <p style="text-align: center;"><i>Default Value:</i> N</p>

**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 DEFAULT

For ~JSA — or 100 the Backfeed parameter is listed as AFTER

For ~JSB — or 0 the Backfeed parameter is listed as BEFORE

For ~JS10 — to 90 the Backfeed parameter is listed as the value entered

# ~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 front panel. When a ~JS command is encountered, it overrides the current front 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	<p><i>Accepted Values:</i></p> <p>A = 100 percent backfeed after printing and cutting</p> <p>B = 0 percent backfeed after printing and cutting, and 100 percent before printing the next label</p> <p>N = normal — 90 percent backfeed after label is printed</p> <p>O = off — turn backfeed off completely</p> <p>10 to 90 = percentage value</p> <p style="padding-left: 40px;">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.</p> <p style="text-align: center;"><i>Default Value:</i> N</p>

**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 DEFAULT

For ~JSA — or 100 the Backfeed parameter is listed as AFTER

For ~JSB — or 0 the Backfeed parameter is listed as BEFORE

For ~JS10 — to 90 the Backfeed parameter is listed as the value entered

# ^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	<i>Accepted Values:</i> 0000 to 9999 If a value greater than 9999 is entered, it is ignored. <i>Default Value:</i> 0000 (off)
a = manually select range of elements to test	<i>Accepted Values:</i> Y (yes) or N (no) <i>Initial Value at Power-up:</i> N
b = first element to check when parameter a is Y	<i>Accepted Values:</i> 0 to 9999 <i>Initial Value at Power-up:</i> 0
c = last element to check when parameter a is Y	<i>Accepted Values:</i> 0 to 9999 <i>Initial Value at Power-up:</i> 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	<p><i>Accepted Values:</i></p> <p>F = reload factory values            These values are lost at power-off if not saved with ^JUS.</p> <p>R = recall last saved values</p> <p>S = save current settings            These values are used at power-on.</p> <p><i>Default Value:</i> a value must be specified</p>

# ^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	<i>Accepted Values:</i> L = low M = medium H = high <i>Default Value:</i> 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	<i>Accepted Values:</i> Y (yes) or N (no) <i>Initial Value at Power-up:</i> 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 front 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 front panel display, and displayed while setting the date and time.

**Format** ^KD*a*

This table identifies the parameters for this format:

Parameters	Details
<i>a</i> = value of date and time format	<i>Accepted Values:</i> 0 = normal, displays <i>Version Number</i> of firmware 1 = MM/DD/YY (24-hour clock) 2 = MM/DD/YY (12-hour clock) 3 = DD/MM/YY (24-hour clock) 4 = DD/MM/YY (12-hour clock)  <i>Default Value:</i> 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 front panel display is presented in format 1.

# ^KL

## Define Language

**Description** The ^KL command selects the language displayed on the front panel.

**Format** ^KL*a*

This table identifies the parameters for this format:

Parameters	Details
a = language	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>1 = English</li> <li>2 = Spanish</li> <li>3 = French</li> <li>4 = German</li> <li>5 = Italian</li> <li>6 = Norwegian</li> <li>7 = Portuguese</li> <li>8 = Swedish</li> <li>9 = Danish</li> <li>10 = Spanish2</li> <li>11 = Dutch</li> <li>12 = Finnish</li> <li>13 = Japanese</li> </ul> <p><i>Default Value:</i> 1</p>

# ^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	<i>Accepted Values:</i> up to 16 alphanumeric characters <i>Default Value:</i> if a value is not entered, the parameter is ignored If more than 16 characters are entered, only the first 16 are used.
b = printer description	<i>Accepted Values:</i> up to 35 alphanumeric characters <i>Default Value:</i> if a value is not entered, the parameter is ignored 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 and 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:**

PRINTER CONFIGURATION	
Zebra Technologies ZTC 105SL-200dpi	
+18.....	DARKNESS
-016.....	TEAR OFF
TEAR OFF.....	PRINT MODE
NON-CONTINUOUS.....	MEDIA TYPE
WEB.....	SENSOR TYPE
THERMAL-TRANS.....	PRINT METHOD
101 4/8 MM.....	PRINT WIDTH
1233.....	LABEL LENGTH
7.0IN 177MM.....	MAXIMUM LENGTH
PARALLEL.....	PARALLEL COMM.
RS232.....	SERIAL COMM.
9600.....	BAUD
8 BITS.....	DATA BITS
NONE.....	PARITY
XON/XOFF.....	HOST HANDSHAKE
NONE.....	PROTOCOL
000.....	NETWORK
NORMAL MODE.....	COMM.
<~> 7EH.....	
<^>.....	

**After using this command:**

PRINTER CONFIGURATION	
Zebra Technologies ZTC 105SL-200dpi Zebra1 desk_printer	
+18.....	DARKNESS
-016.....	TEAR OFF
TEAR OFF.....	PRINT MODE
NON-CONTINUOUS.....	MEDIA TYPE
WEB.....	SENSOR TYPE
THERMAL-TRANS.....	PRINT METHOD
101 4/8 MM.....	PRINT WIDTH
1233.....	LABEL LENGTH
7.0IN 177MM.....	MAXIMUM LENGTH
PARALLEL.....	PARALLEL COMM.
RS232.....	SERIAL COMM.
9600.....	BAUD
8 BITS.....	DATA BITS
NONE.....	PARITY
XON/XOFF.....	HOST HANDSHAKE
NONE.....	PROTOCOL
000.....	NETWORK
NORMAL MODE.....	COMM.
<~> 7EH.....	
<^>.....	

# ^KP

## Define Password

**Description** The ^KP command is used to define the password that must be entered to access the front panel switches and LCD Setup Mode.

**Format** ^KP####

This table identifies the parameters for this format:

Parameters	Details
#### = mandatory four-digit password	<i>Accepted Values:</i> any four-digit numeric sequence <i>Default Value:</i> 1234

➔ **Example** • This is an example of how to set a new front 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)	<i>Accepted Values:</i> 0 to 32000 <i>Initial Value at Power-up:</i> 0 or last permanently saved value
y = y-axis position (in dots)	<i>Accepted Values:</i> 0 to 32000 <i>Initial Value at Power-up:</i> 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

 **ZPL II Commands**  
^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** ^LL $y$

This table identifies the parameters for this format:

Parameters	Details
$y$ = y-axis position (in dots)	<p><i>Accepted Values:</i> 1 to 32000, not to exceed the maximum label size.</p> <p>While the printer accepts any value for this parameter, the amount of memory installed determines the maximum length of the label.</p> <p><i>Default Value:</i> typically set through the LCD (if applicable), or to the maximum label length capability of the printer.</p>

**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	<i>Accepted Values:</i> Y (yes) or N (no) <i>Initial Value at Power-up:</i> N or last permanently saved value



**Example** • This is an example that shows printing white over black and black over white:

ZPL II CODE	GENERATED LABEL
<pre> ^XA^LRY ^FO100,50 ^GB195,203,195^FS ^FO180,110^CFG ^FDLABEL^FS ^FO130,170 ^FDREVERSE^FS ^XZ </pre>	

**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.

 **ZPL II Commands**  
^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:

$$\begin{aligned} & \text{Z-130 and Z-220 values for } ^LHx + ^FOx \\ & \text{(distance from edge of label) = printer value for } ^LSa \end{aligned}$$

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)	<i>Accepted Values:</i> -9999 to 9999 <i>Initial Value at Power-up:</i> 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.

**Format** ^LTx

This table identifies the parameters for this format:

Parameters	Details
x = label top (in dot rows)	<i>Accepted Values:</i> -120 to 120 <i>Default Value:</i> 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	<i>Accepted Values:</i> Y (clear bitmap) or N (do not clear bitmap) <i>Initial Value at Power-up:</i> 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	<i>Accepted Values:</i> -30 to 30, depending on current value <i>Initial Value at Power-up:</i> 0 If no value is entered, this command is ignored.

➔ **Examples** • 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 XiIIIPlus is 0 to 30 in increments of .1

The firmware is setup so that the 2 Darkness ZPL commands (^MD, ~SD) accepts that range of settings.

➔ **Example** • These are examples of the XiIIIPlus 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.

# <sup>^</sup>MF

## Media Feed

**Description** The <sup>^</sup>MF command dictates what happens to the media at power-up and at head-close after the error clears.

**Format** <sup>^</sup>MF<sub>p, h</sub>

This table identifies the parameters for this format:

Parameters	Details
p = feed action at power-up	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>F = feed to the first web after sensor</li> <li>C = (see <a href="#">~JC</a> definition)</li> <li>L = (see <a href="#">~JL</a> definition)</li> <li>N = no media feed</li> </ul> <p><i>Default Value:</i> platform-dependent</p>
h = feed action after closing printhead	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>F = feed to the first web after sensor</li> <li>C = (see <a href="#">~JC</a> definition)</li> <li>L = (see <a href="#">~JL</a> definition)</li> <li>N = no media feed</li> </ul> <p><i>Default Value:</i> platform-dependent</p>

**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 <sup>^</sup>JU command to save changes.

# ^ML

## Maximum Label Length

**Description** The ^ML command lets you adjust the maximum label length.

**Format** ^MLa , b , c , d

This table identifies the parameters for this format:

Parameters	Details
a = maximum label length (in dot rows)	<i>Accepted Values:</i> 0 to maximum length of label <i>Default Value:</i> last permanently saved value
b = maximum logical paper out counter	<i>Accepted Values:</i> must be greater than the actual label length or the printer indicates a <b>paper out</b> error after each label <i>Default Value:</i> set to one inch longer than twice the label length
c = maximum physical paper out counter	<i>Accepted Values:</i> must be greater than the actual notch or hole or the printer indicates a <b>paper out</b> condition after each label <i>Default Value:</i> is set to one half an inch
d = maximum ribbon out counter	<i>Accepted Values:</i> allows for the ribbon sensors to occasionally get an incorrect ribbon reading without causing an error condition <i>Default Value:</i> set to one half of a millimeter

**Important** • The printer ignores ribbon indications that are less than one-half of a millimeter in length.

**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. 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.

**Format** ^MMa , b



This table identifies the parameters for this format:

Parameters	Details
a = desired mode	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>T = Tear-off</li> <li>P = Peel-off (not available on S-300)</li> <li>R = Rewind</li> <li>A = Applicator (depends on printer model)</li> <li>C = Cutter</li> </ul> <p><i>Default Value:</i> T</p> <p>The values available for parameter a are dependent on the printer being used and whether it supports the option.</p>
b = prepeel select	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> Y</p> <p>The command is ignored if parameters are missing or invalid. The current value of the command remains unchanged.</p>

**Comments** Be sure to select the appropriate value for the print mode being used to avoid unexpected results.

# ^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	<p><i>Accepted Values:</i></p> <p>N = continuous media</p> <p>*Y = non-continuous media web sensing</p> <p>*W = non-continuous media web sensing</p> <p>M = non-continuous media mark sensing</p> <p><i>Default Value:</i> a value must be entered or the command is ignored</p>

\* provides the same result.

# ^MP

## Mode Protection


**Description** The ^MP command is used to disable the various mode functions on the front 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	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>D = disable Darkness Mode</li> <li>P = disable Position Mode</li> <li>C = disable Calibration Mode</li> <li>E = enable all modes</li> <li>S = disable all mode saves (modes can be adjusted but values are not saved)</li> <li>W = disable Pause</li> <li>F = disable Feed</li> <li>X = disable Cancel</li> <li>M = disable menu changes</li> </ul> <p><i>Default Value:</i> a value must be entered or the command is ignored</p>

 **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	<p><i>Accepted Values:</i></p> <p>T = thermal transfer media</p> <p>D = direct thermal media</p> <p><i>Default Value:</i> a value must be entered or the command is ignored</p>

# ^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	<i>Accepted Values:</i> D = dots I = inches M = millimeters <i>Default Value:</i> D
b = format base in dots per inch	<i>Accepted Values:</i> 150, 200, 300 <i>Default Value:</i> a value must be entered or the command is ignored
c = desired dots-per-inch conversion	<i>Accepted Values:</i> 300, 600 <i>Default Value:</i> 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 2** • 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 Heading Warning

**Description** 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	<i>Accepted Values:</i> y = enable head cold warning 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)	<i>Accepted Values:</i> 001 to 999 <i>Default Value:</i> 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 ZNET RS-485 printer networking.

# ^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)	<i>Accepted Values:</i> 001 to 999 <i>Default Value:</i> 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 ZNET RS-485 printer networking.

# ~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 ZNET RS-485 printer networking.

# ^NS

## Change Networking Settings

**Description** The ^NS command is used to change network settings.

**Format** ^NSa , b , c , d

This table identifies the parameters for this format:

Parameters	Details
a = network setting	<i>Accepted Values:</i> IP Resolution. a (ALL) , b (BOOTP), c (DHCP and BOOTP), d (DHCP), g (GLEANNING ONLY), r (RARP), p (permanent)
b = IP Address	<i>Accepted Values:</i> 0 to 255
c = Subnet Mask	<i>Accepted Values:</i> 0 to 255
d = Default Gateway	<i>Accepted Values:</i> 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 ZNET RS-485 printer networking.

# ^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	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> a value must be entered or the command is ignored

# ^PF ~PF

## 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	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> N

➔ **Example** • This is an example of printing a mirror image on a label:

ZPL II CODE	GENERATED LABEL
<pre>^XA^PMY ^FO100,100 ^CFG ^FDMIRROR^FS ^FO100,160 ^FDIMAGE^FS ^XZ</pre>	

**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	<i>Accepted Values:</i> N (normal) or I (invert) <i>Default Value:</i> N

➔ **Example** • This is an example of printing a label at 180 degrees:

ZPL II CODE	GENERATED LABEL
<pre> ^XA^CFD ^POI ^LH330,10 ^FO50,50 ^FDZEBRA TECHNOLOGIES^FS ^FO50,75 ^FDVernon Hills, IL^FS ^XZ </pre>	

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.

 **ZPL II Commands**  
^PO

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 front 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	<i>Accepted Value:</i> 1 to 99,999,999 <i>Default Value:</i> 1
p = pause and cut value (labels between pauses)	<i>Accepted Value:</i> 1 to 99,999,999 <i>Default Value:</i> 0 (no pause)
r = replicates of each serial number	<i>Accepted Value:</i> 0 to 99,999,999 replicates <i>Default Value:</i> 0 (no replicates)
o = override pause count	<i>Accepted Value:</i> Y (yes) or N (no) <i>Default Value:</i> 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.

# <sup>^</sup>PR

## Print Rate

**Description** The <sup>^</sup>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** <sup>^</sup>PRp , s , b

This table identifies the parameters for this format:

Parameters	Details	
p = print speed	<i>Accepted Values:</i>	
A or 2	50.8 mm/sec.	(2 inches/sec.)
B or 3	76.2 mm/sec.	(3 inches/sec.)
C or 4	101.6 mm/sec.	(4 inches/sec.)
5	127 mm/sec.	(5 inches/sec.)
D or 6	152.4 mm/sec.	(6 inches/sec.)
E or 8	203.2 mm/sec.	(8 inches/sec.)
9	220.5 mm/sec.	(9 inches/sec.)

---

10	245 mm/sec.	(10 inches/sec.)
11	269.5 mm/sec.	(11 inches/sec.)
12	304.8 mm/sec.	(12 inches/sec.)

---

*Default Value: A*

s = slew speed

*Accepted Values:*

---

A or 2	50.8 mm/sec.	(2 inches/sec.)
B or 3	76.2 mm/sec.	(3 inches/sec.)
C or 4	101.6 mm/sec.	(4 inches/sec.)
5	127 mm/sec.	(5 inches/sec.)
D or 6	152.4 mm/sec.	(6 inches/sec.)
E or 8	203.2 mm/sec.	(8 inches/sec.)
9	220.5 mm/sec.	(9 inches/sec.)
10	245 mm/sec.	(10 inches/sec.)
11	269.5 mm/sec.	(11 inches/sec.)
12	304.8 mm/sec.	(12 inches/sec.)

---

*Default Value: D*

b = backfeed speed

*Accepted Values:*

---

A or 2	50.8 mm/sec.	(2 inches/sec.)
B or 3	76.2 mm/sec.	(3 inches/sec.)
C or 4	101.6 mm/sec.	(4 inches/sec.)
5	127 mm/sec.	(5 inches/sec.)
D or 6	152.4 mm/sec.	(6 inches/sec.)
E or 8	203.2 mm/sec.	(8 inches/sec.)

---

---

9	220.5 mm/sec.	(9 inches/sec.)
10	245 mm/sec.	(10 inches/sec.)
11	269.5 mm/sec.	(11 inches/sec.)
12	304.8 mm/sec.	(12 inches/sec.)

---

*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 users guide for specifics on performance.



# ~PR

## Applicator Reprint

**Description** The ~PR command is supported only by the *PAX* and *PAX2*-Series printers. If the ~PR command is enabled (see ^JJ), the last label printed reprint, similar to the applicator asserting the Reprint signal on the applicator port.

**Format** ~PR

**Comments** Pressing PREVIOUS on the front 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 front 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)	<i>Accepted Values:</i> 2, to the width of the label If the value exceeds the width of the label, the width is set to the label's maximum size. <i>Default Value:</i> 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	<i>Accepted Values:</i> 1 or 2 <i>Default Value:</i> a value must be specified or the command is ignored

➔ **Example** • This is an example of the ~RO command.

```
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 CNTR1
92928 LABLS..... RESET CNTR1
92928 LABLS..... RESET CNTR2
```

Before

```
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
```

After

PRINTER CONFIGURATION	
Zebra Technologies	
ZTC 140xiIIIPIde-200dpi	
Benny	
a	
19 S.....	DARKNESS
6 IPS.....	PRINT SPEED
028 OFF.....	TEAR OFF
.....	PRINT MODE
NON-CONTINUOUS.....	MEDIA TYPE
WEB.....	SENSOR TYPE
THERMAL-TRANS.....	PRINT METHOD
128 0/8 MM.....	PRINT WIDTH
128.....	LABEL LENGTH
39.0IN 988PM.....	MAXIMUM LENGTH
MEDIA DISABLED.....	EARLY WARNING
MAINT. OFF.....	EARLY WARNING
NOT CONNECTED.....	USB COMM.
PARALLEL.....	PARALLEL COMM.
RS232.....	SERIAL COMM.
38400.....	BAUD
8 BITS.....	DATA BITS
NONE.....	PARITY
XON/XOFF.....	HOST HANDSHAKE
NONE.....	PROTOCOL
000.....	NETWORK ID
NORMAL MODE.....	COMMUNICATIONS
<~> 7EH.....	CONTROL PREFIX
<~> 7EH.....	FORMAT PREFIX
<.> 2CH.....	DELIMITER CHAR
ZPL II.....	ZPL MODE
NO MOTION.....	MEDIA POWER UP
NO MOTION.....	HEAD CLOSE
DEFAULT.....	BACKFEED
+000.....	LABEL TOP
+0010.....	LEFT POSITION
0000.....	HEAD TEST COUNT
0526.....	HEAD RESISTOR
OFF.....	VERIFIER PORT
OFF.....	APPLICATOR PORT
PULSE MODE.....	START PRINT SIG
FEED MODE.....	RESYNCH MODE
051.....	WEB S.
079.....	MEDIA S.
071.....	RIBBON S.
050.....	MARK S.
000.....	MARK RED S.
084.....	MEDIA LED
007.....	RIBBON LED
027.....	MARK LED
+10.....	LCD ADJUST
DPSM/VA.....	MODES ENABLED
1024 8/PH FULL.....	MODES DISABLED
1024 8/PH FULL.....	RESOLUTION
V42.11.6.....	FIRMWARE
V16.0.0.30.....	HARDWARE ID
CUSTOMIZED.....	CONFIGURATION
1228B.....	RAM
NONE.....	MEMORY CARD
2048.....	ONBOARD FLASH
NONE.....	FORMAT CONVERT
005 DISPLAY.....	P32 INTERFACE
007 POWER SUPPLY.....	P34 INTERFACE
001 ADV. COUNTER.....	P35 INTERFACE
FW VERSION.....	TWIN/COAX ID
08/1/01.....	IDLE DISPLAY
20:05.....	RTC DATE
PERMANENT.....	RTC TIME
ALL.....	IP PROTOCOL
012.107.109.046.....	IP ADDRESS
255.255.255.224.....	SUBNET MASK
112.107.109.033.....	DEFAULT GATEWAY
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
AV12446.04-26-2002.33009.00.VH1	

FIRMWARE IN THIS PRINTER IS COPYRIGHTED

# <sup>^</sup>SC

## Set Serial Communications

**Description** The <sup>^</sup>SC command allows you to change the serial communications parameters you are using.

**Format** <sup>^</sup>SCa,b,c,d,e,f

This table identifies the parameters for this format:

Parameters	Details
a = baud rate	<i>Accepted Values:</i> 110; 600; 1,200; 2400; 4800; 9600; 14400; 19200; 28800; 38400; or 57600; 115000 <i>Default Value:</i> must be specified or the parameter is ignored
b = word length (in data bits)	<i>Accepted Values:</i> 7 or 8 <i>Default Value:</i> must be specified
c = parity	<i>Accepted Values:</i> N (none), E (even), or O (odd) <i>Default Value:</i> must be specified
d = stop bits	<i>Accepted Values:</i> 1 or 2 <i>Default Value:</i> must be specified
e = protocol mode	<i>Accepted Values:</i> X (XON/XOFF), D (DTR/DSR), or R (RTS) <i>Default Value:</i> must be specified
f = Zebra protocol	<i>Accepted Values:</i> A = ACK/NAK N = none Z = Zebra <i>Default Value:</i> 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 front panel display.

**Format** ~SD##

This table identifies the parameters for this format:

Parameters	Details
## = desired darkness setting (two-digit number)	<i>Accepted Values:</i> 00 to 30 <i>Default Value:</i> last permanently saved value

➔ **Example** • These are examples of the XiIIIPlus 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 was created 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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> R:
o = name of encoding table	<i>Accepted Value:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> a value must be specified
x = extension	<i>Fixed Value:</i> .DAT

# ^SF

## Serialization Field (with a Standard ^FD String)

**Description** The ^SF command allows you to serialize a standard ^FD string. Fields serialized with this command are right-justified or end with the last character of the string. The increment string is aligned with the mask, starting with the right-most position. 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	<p>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-most position.</p> <p><i>Mask String placeholders:</i></p> <ul style="list-style-type: none"> <li>D or d – Decimal numeric 0–9</li> <li>H or h – Hexadecimal 0–9 plus a-f or A-F</li> <li>O or o – Octal 0–7</li> <li>A or a – Alphabetic a–z or A–Z</li> <li>N or n – Alphanumeric 0–9 plus a–z or A–Z</li> <li>% – Ignore character or skip</li> </ul>
b = increment string	<p>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 character position.</p> <p>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.</p>

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 <sup>^</sup>FD string:

ZPL II CODE	GENERATED LABELS
<pre> <sup>^</sup>XA <sup>^</sup>FO100,100 <sup>^</sup>CF0,100 <sup>^</sup>FD12A<sup>^</sup>SFnnA,F<sup>^</sup>FS <sup>^</sup>PQ3 <sup>^</sup>XZ                     </pre> <p><i>Note: The ZPL II code above will generate three separate labels, seen to the right.</i></p>	<div style="border: 1px solid black; padding: 5px; text-align: center; font-size: 24pt; font-weight: bold;">12A</div> <div style="border: 1px solid black; padding: 5px; text-align: center; font-size: 24pt; font-weight: bold;">12F</div> <div style="border: 1px solid black; padding: 5px; text-align: center; font-size: 24pt; font-weight: bold;">12K</div>

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 <sup>^</sup>PQ command.

In a similar instance, the <sup>^</sup>FD string could be replaced with either of the <sup>^</sup>FD strings below to generate a series of label, determined by <sup>^</sup>PQ.

```
^FDBL0000^SFAAdddd,1
```

The print sequence on this series of labels is:

```
BL0000, BL0001, ...BL0009, BL0010, ...
```

```
BL0099, BL0100, ...BL9999, BM0000...
```

```
^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...
```

# ^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.

**Format** ^SLa ,b

This table identifies the parameters for this format:

Parameters	Details
a = mode	<p><i>Accepted Values:</i></p> <p>S = Start Time Mode. This is the time that is read from the Real-Time Clock when label formatting begins (when <sup>^</sup>XA is received). The first label has the same time placed on it as the last label.</p> <p>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. <i>Time Now</i> is similar to a serialized time or date field.</p> <p><i>Default Value:</i> S</p>
b = language	<p><i>Accepted Values:</i></p> <p>1 = English</p> <p>2 = Spanish</p> <p>3 = French</p> <p>4 = German</p> <p>5 = Italian</p> <p>6 = Norwegian</p> <p>7 = Portuguese</p> <p>8 = Swedish</p> <p>9 = Danish</p> <p>10 = Spanish 2</p> <p>11 = Dutch</p> <p>12 = Finnish</p> <p><i>Default Value:</i> the language selected with <sup>^</sup>KL or the front panel</p>

# ^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** ^SN $v, n, z$

This table identifies the parameters for this format:

Parameters	Details
$v$ = starting value	<i>Accepted Values:</i> 12-digits maximum for the portion to be indexed <i>Default Value:</i> 1
$n$ = increment or decrement value	<i>Accepted Values:</i> 12-digit maximum <i>Default Value:</i> 1 To indicate a decrement value, precede the value with a minus (-) sign.
$z$ = add leading zeros (if needed)	<i>Accepted Values:</i> Y (yes) or N (no) <i>Default Value:</i> N

➔ **Example** • This example shows incrementing by a specified value:

ZPL II CODE	GENERATED LABELS
<pre> ^XA ^FO260,110 ^CFG ^SN001,1,Y^FS ^PQ3 ^XZ </pre> <p><i>Note: The ZPL II code above will generate three separate labels, seen to the right.</i></p>	<div style="border: 1px solid black; padding: 5px; text-align: center; margin-bottom: 5px;">001</div> <div style="border: 1px solid black; padding: 5px; text-align: center; margin-bottom: 5px;">002</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">003</div>

**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.

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

This table identifies the parameters for this format:

Parameters	Details
a = clock set	<i>Accepted Values:</i> 2 (secondary) or 3 (tertiary) <i>Default Value:</i> value must be specified
b = months offset	<i>Accepted Values:</i> -32000 to 32000 <i>Default Value:</i> 0
c = days offset	<i>Accepted Values:</i> -32000 to 32000 <i>Default Value:</i> 0
d = years offset	<i>Accepted Values:</i> -32000 to 32000 <i>Default Value:</i> 0
e = hours offset	<i>Accepted Values:</i> -32000 to 32000 <i>Default Value:</i> 0
f = minutes offset	<i>Accepted Values:</i> -32000 to 32000 <i>Default Value:</i> 0
g = seconds offset	<i>Accepted Values:</i> -32000 to 32000 <i>Default Value:</i> 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:

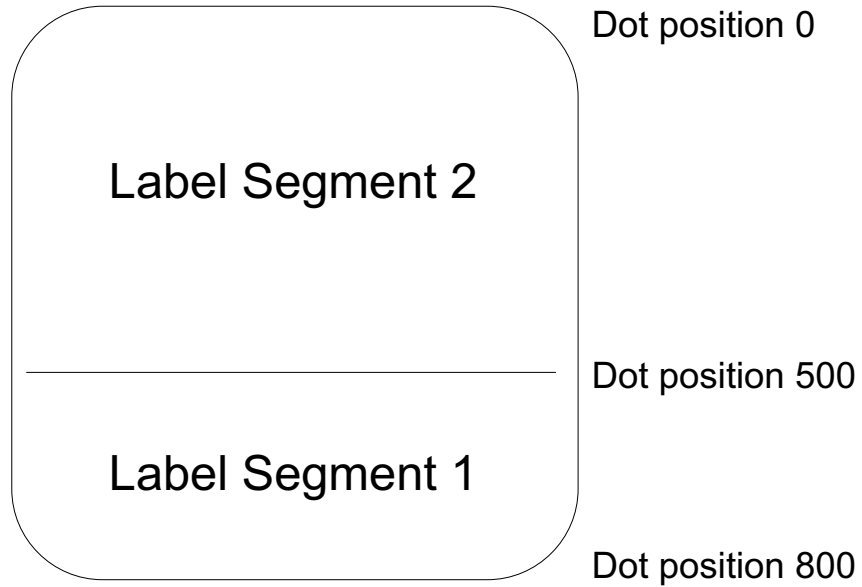
```
^SP#^FS
```

**Format** ^SPa

This table identifies the parameters for this format:

Parameters	Details
a = dot row to start printing	<i>Accepted Values:</i> 0 to 32000 <i>Default Value:</i> 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	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>A = paper out</li> <li>B = ribbon out</li> <li>C = printhead over-temp</li> <li>D = printhead under-temp</li> <li>E = head open</li> <li>F = power supply over-temp</li> <li>G = ribbon-in warning (Direct Thermal Mode)</li> <li>H = rewind full</li> <li>I = defaulted printer</li> <li>J = cut error</li> <li>K = printer paused</li> <li>L = PQ job completed</li> <li>M = label taken</li> <li>N = head element out</li> <li>O = ZBI (Zebra BASIC Interpreter) runtime error</li> <li>P = ZBI (Zebra BASIC Interpreter) forced error</li> <li>Q = power on</li> <li>R = all errors</li> <li>* = wild card to stop alerts for all conditions</li> </ul>

Parameters	Details
b = destination	<i>Accepted Values:</i> A = serial port B = parallel port C = e-mail address D = TCP/IP E = UDP/IP F = SNMP trap * = wild card to stop alerts for all destinations
c = halt messages	<i>Accepted Values:</i> Y (halt messages) or N (start messages) <i>Default Value:</i> 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)	<i>Accepted Value:</i> 0488 to 1175 <i>Default Value:</i> 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 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.

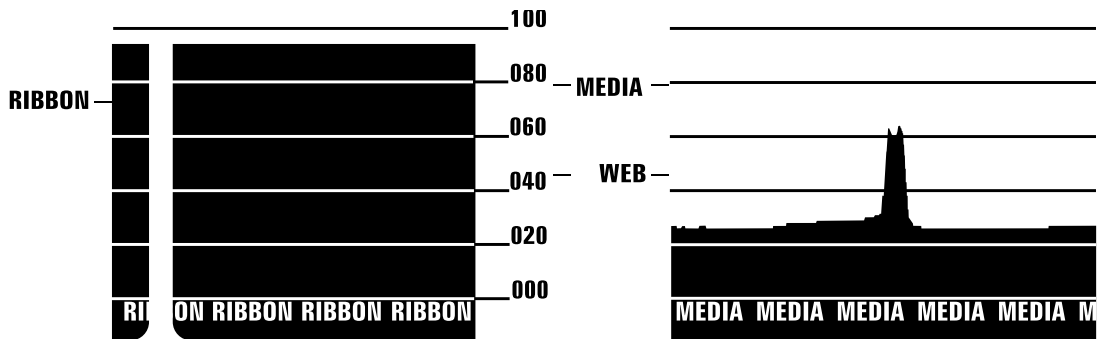
**Format** ^SSw,m,r,l,m2,r2,a,b,c

This table identifies the parameters for this format:

Parameters	Details
w = web (three-digit value)	<i>Accepted Values:</i> 000 to 100 <i>Default Value:</i> value shown on the media sensor profile or configuration label
m = media (three-digit value)	<i>Accepted Values:</i> 000 to 100 <i>Default Value:</i> value shown on the media sensor profile or configuration label
r = ribbon (three-digit value)	<i>Accepted Values:</i> 001 to 100 <i>Default Value:</i> value shown on the media sensor profile or configuration label
l = label length (in dots, four-digit value)	<i>Accepted Values:</i> 0001 to 32000 <i>Default Value:</i> value calculated in the calibration process
m2 = intensity of media LED (three-digit value)	<i>Accepted Values:</i> 000 to 100 <i>Default Value:</i> value calculated in the calibration process
r2 = intensity of ribbon LED (three-digit value)	<i>Accepted Values:</i> 000 to 100 <i>Default Value:</i> value calculated in the calibration process

Parameters	Details
a = mark sensing (three-digit value)	<i>Accepted Values:</i> 000 to 100 <i>Default Value:</i> value calculated in the calibration process
b = mark media sensing (three-digit value)	<i>Accepted Values:</i> 000 to 100 <i>Default Value:</i> value calculated in the calibration process
c = mark LED sensing (three-digit value)	<i>Accepted Values:</i> 000 to 100 <i>Default Value:</i> 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	<i>Accepted Values:</i> 01 to 12 <i>Default Value:</i> current month
b = day	<i>Accepted Values:</i> 01 to 31 <i>Default Value:</i> current day
c = year	<i>Accepted Values:</i> 1998 to 2097 <i>Default Value:</i> current year
d = hour	<i>Accepted Values:</i> 00 to 23 <i>Default Value:</i> current hour
e = minute	<i>Accepted Values:</i> 00 to 59 <i>Default Value:</i> current minute
f = second	<i>Accepted Values:</i> 00 to 59 <i>Default Value:</i> current second
g = format	<i>Accepted Values:</i> A = a.m. P = p.m. M = 24-hour military <i>Default Value:</i> 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:

Parameters	Details
a = condition type	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>A = paper out</li> <li>B = ribbon out</li> <li>C = printhead over-temp</li> <li>D = printhead under-temp</li> <li>E = head open</li> <li>F = power supply over-temp</li> <li>G = ribbon-in warning (Direct Thermal Mode)</li> <li>H = rewind full</li> <li>I = defaulted printer</li> <li>J = cut error</li> <li>K = printer paused</li> <li>L = PQ job completed</li> <li>M = label taken</li> <li>N = head element out</li> <li>O = ZBI (Zebra BASIC Interpreter) runtime error</li> <li>P = ZBI (Zebra BASIC Interpreter) forced error</li> <li>Q = power on</li> <li>R = all errors</li> </ul> <p><i>Default Value:</i> if the parameter is missing or invalid, the command is ignored</p>
b = destination for route alert	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>A = serial port</li> <li>B* = parallel port</li> <li>C = e-mail address</li> <li>D = TCP/IP</li> <li>E = UDP/IP</li> <li>F = SNMP trap</li> </ul> <p><i>Default Value:</i> if this parameter is missing or invalid, the command is ignored</p> <p>* <b>Requires</b> bidirectional communication.</p>
c = enable condition set alert to this destination	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> Y or previously configured value</p>

Parameters	Details
d = enable condition clear alert to this destination	<p><i>Accepted Values:</i> Y (yes) or N (no)</p> <p><i>Default Value:</i> N or previously configured value</p> <p>Parameters e and f are suboptions based on destination. If the sub-options are missing or invalid, these parameters is ignored.</p>
e = destination setting	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>Internet e-mail address (e.g. user@company.com)</li> <li>IP address (for example, 10.1.2.123)</li> <li>SNMP trap</li> <li>IP or IPX addresses</li> </ul>
f = port number	<p><i>Accepted Values:</i></p> <ul style="list-style-type: none"> <li>TCP port # (0 to 65535)</li> <li>UDP port # (0 to 65535)</li> </ul>

### Examples •

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	<i>Accepted Values:</i> 1 = ZPL 2 = ZPL II <i>Default Value:</i> 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	<i>Accepted Values:</i> -120 to 120 <i>Default Value:</i> last permanent value saved

**Important** • 3-digit value in dot rows must be used.

**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** ^TOd:o.x,s:o.x

This table identifies the parameters for this format:

Parameters	Details
d = source device of stored object	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> if a drive is not specified, all objects are transferred to the drive set in parameter s
o = stored object name	<i>Accepted Values:</i> any existing object conforming to Zebra conventions <i>Default Value:</i> if a name is not specified, * is used — all objects are selected
x = extension	<i>Accepted Values:</i> any extension conforming to Zebra conventions <i>Default Value:</i> if an extension is not specified, * is used — all extensions are selected
s = destination device of the stored object	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> a destination must be specified

Parameters	Details
o = name of the object at destination	<i>Accepted Values:</i> up to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, the name of the existing object is used
x = extension	<i>Accepted Values:</i> any extension conforming to Zebra conventions <i>Default Value:</i> 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.

**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
```

**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
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 ^Af 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:○.x

This table identifies the parameters for this format:

Parameters	Details
d = source device — optional	<i>Accepted Values:</i> R:, E:, B:, A: and Z: <i>Default Value:</i> R:
○ =object name — optional	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> * The use of a ? (question mark) is also allowed.
x = extension — optional	<i>Accepted Values:</i> any extension conforming to Zebra conventions .FNT = font .BAR = bar code .ZPL = stored ZPL format .GRF = GRF graphic .CO = memory cache .DAT = font encoding .BAS = ZBI program .STO = data storage .PNG = PNG graphic * = all objects <i>Default Value:</i> * The use of a ? (question mark) is also allowed.



➔ **Example 1** • To print a label listing all objects in DRAM, enter:

```
^XA  
^WDR: * . *  
^XZ
```

➔ **Example 2** • To print a label listing all resident bar codes, enter:

```
^XA  
^WDZ: * . BAR  
^XZ
```

➔ **Example 3** • 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)

# ^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:○.x

This table identifies the parameters for this format:

Parameters	Details
d = source device of stored image	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> search priority (R:, E:, B:, and A:)
○ = name of stored image	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension l	<i>Fixed Value:</i> .ZPL

→ **Example** • This is an example of using the ^XF command to recall the format STOREFMT.ZPL from DRAM and insert new reference data in the ^FN fields:

ZPL II CODE	GENERATED LABEL
<pre>^XA ^XFR:STOREFMT.ZPL^FS ^FN1^FDZEBRA^FS ^FN2^FDLABEL^FS ^XZ</pre>	<pre>ZEBRA PRINTER BUILT BY ZEBRA</pre>

# ^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	<i>Accepted Values:</i> R:, E:, B:, and A: <i>Default Value:</i> search priority (R:, E:, B:, and A:)
o = name of stored image	<i>Accepted Values:</i> 1 to 8 alphanumeric characters <i>Default Value:</i> if a name is not specified, UNKNOWN is used
x = extension l	<i>Fixed Value:</i> .GRF
mx = magnification factor on the x-axis	<i>Accepted Values:</i> 1 to 10 <i>Default Value:</i> 1
my = magnification factor on the y-axis	<i>Accepted Values:</i> 1 to 10 <i>Default Value:</i> 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	<i>Accepted Values:</i> 0 to 999999 – setting 0 disables automatic shutdown <i>Default Value:</i> last permanently saved value or 0
b =label status at shutdown	<i>Accepted Values:</i> Y = indicates to shutdown when labels are still queued N = indicates all labels must be printed before shutting down <i>Default Value:</i> N

**Comments** The ^ZZ command is only valid on the PA400 and PT400 battery-powered printers.





# ZBI Commands

This section contains an alphabetical listing of the ZBI commands. The version of firmware you are using can affect if the ZBI code is recognized.

The text in this section is arranged under these headings:


**Description** Under this heading is a description of how the command is used, what it is capable of, and any defining characteristics it has.

**Format** The format is how the command is arranged and what parameters it contains. For example, the AUTONUM command starts the auto-numbering option. The format for the command is AUTONUM <A>,<B>. The <A> and <B> are parameters of this command and are replaced with values determined by the user.

**Parameters** If a command has values that can be defined to make command function more specific, they are listed under this heading. Still using the AUTONUM example, the <A> parameter is defined as:

<A> = number used to start the auto-numbering sequence

**Example** When a command is best clarified in context, an example of the ZBI code is provided. Text indicating parameters, exact code to be entered, or data returned from the host is printed in the `Courier` font to be easily recognizable.

 **Example** • An example of AUTONUM code is:

```
AUTONUM 10,5  
10 PRINT "HELLO WORLD"  
15 GOTO 10
```

In an example, the > symbol indicates a line of code you enter.

**Comments** 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. An example comment could be: This is a program command and must be preceded by a line number.

## AND

**Description** The AND statement is a Boolean operator. If both of the expressions are true, the result is true; otherwise the result is false.

**Format** `<Boolean expression> AND <Boolean expression>`

**Comments** This is a Boolean operator that is used in conjunction with Boolean expressions.

## Arrays

**Description** An array is a collection of values used by a program. Arrays share these characteristics:

- arrays are allowed for both integer and string variables.
- indexes of arrays are accessed through parentheses.
- array indexes start at 1 and end at the length of an array (for example, variable 3 returns the value in the third location of the variable array).
- one- and two-dimensional arrays are allowed. Two-dimensional arrays are referenced with two indexes in parenthesis, separated by a comma.
- arrays can be re-dimensioned only by a call to DECLARE, which destroys the original array.
- array size is limited only by the size of the memory heap allocated.
- if an array cannot be allocated, an error message displays — `Error: Heap overflow`.
- if you attempt to access an array outside of its limits, an error message shows — `Error: Invalid array access`.

## AUTONUM

**Description** This command automatically generates sequential program line numbers. This feature is disabled by overwriting the current line number and entering the desired interactive mode commands, or leaving the line blank.

**Format** `AUTONUM A, B`

**Parameters:**

A = the number used to start the auto-numbering sequence

B = the automatic increment between the new line numbers

➔ **Example** • This example shows specifying the starting line number in the increment between new line number:

```
AUTONUM 10,5
10 PRINT "HELLO WORLD"
15 GOTO 10
```

**Comments** The two lines are automatically started with the AUTONUM parameters; in this case, the first line is started with 10 and each subsequent line is incremented by 5.

This is an interactive command that takes effect as soon as it is received by the printer.

## Boolean Expression

**Description** A Boolean expression holds 0 (zero) as false and non-zero as true.

**Formats** <string expression> <Boolean compare> <string expression>  
 <# expression> <Boolean compare> <# expression>  
 (<Boolean expression>)

**Comments** # indicates a numeric expression. A numeric expression cannot be compared to a string expression. If attempted, an error message displays — Error: Poorly formed expression.

A numeric expression can be substituted for a Boolean expression where a value of 0 (zero) represents false and a non-zero value represents true.

Order of precedence for <, <=, >, >=, = are all the same order followed by NOT, AND, and OR, in that order. Items with the same order of precedence are processed from left to right.

## BREAK

**Description** This command is available only when the DEBUG function has been activated. When DEBUG is on, BREAK halts processing. RUN starts the program from the beginning. RESTART allows the program to continue from where it left off.

**Format** BREAK

**Comments** This is a program command that is preceded by a line number.

## Channels

**Description** I/O commands can be issued to channels to direct the commands to specific ports. The data-input port is specified through the ZPL commands that start the ZBI interpreter. It is through this port that all the interactive communications take place.

**Format** #<channel expression>

**Parameters:**

#<channel expression>

*Accepted Values:* 0 to 9

*Default Value:* 0

## CLOSE

**Description** This command is implemented to close specific ports that are in use. If a port is open on a channel and the CLOSE command is entered, the port closes and returns to communicating with the ZPL buffer.

**Format** CLOSE <channel expression>

**Parameters:**

<channel expression>

*Accepted Values:* 0 through 9

*Default Value:* will not be used

→ **Example** • This example shows the closing of channel 1:

```
>CLOSE #1
```

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## CLRERR

**Description** This command sends a message to clear the error flag to the printer. The error flag resets if the error is nonpermanent.

**Format** 10 CLRERR

**Comments** This is a program command that is preceded by a line number.

## CTRL-C

**Description** Sending 03 to port 0 or using the Ctrl-C keystroke combination terminates any ZBI program currently running.

## DEBUG

**Description** When DEBUG is on, the TRACE and BREAK options are enabled. When DEBUG is off, the TRACE and BREAK options are disabled.

**Format** DEBUG <ON/OFF>

**Parameters:**

<ON/OFF> = toggles the debug mode on or off

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## Declaration

**Description** Some common characteristics of declarations are:

- The names of variables are typically established at the beginning of a ZBI program. When declared, integer variables are initialized to 0. When declared, string variables are initialized to empty strings (“ ”).
- An explicit declaration deletes and reassigns any previously declared variable.
- An implicit declaration uses a variable name that is not previously defined. The value of this variable is undefined.
- Nonarray string and integer variables might be declared implicitly or explicitly.
- Arrays must be explicitly defined.

## DECLARE

**Description** This is the explicit method of declaring a variable and is typically performed at the beginning of a program. Arrays are specified by placing a set of closed parentheses around a numeric expression of the array size to be allocated.

**Format** DECLARE <type> <variable name> [, <variable name>]\*

**Parameters:**

<type> = numeric or string

<variable name> = the variable being declared, which could be an array

➔ **Example** • This is an example of declaring a variable:

```
10 DECLARE NUMERIC A ! Declare an integer
20 DECLARE STRING A$ ! Declare a string
30 DECLARE NUMERIC My_Array(10) ! Declare an integer
array of size 10
40 DECLARE STRING My_Str_Array$(10) ! Declare a string
array with 10 strings
50 DECLARE NUMERIC A, B(10), C ! Declare multiple
integer variables
60 DECLARE STRING A$, B$, C$ ! Declare multiple string
variables
70 DECLARE STRING A2D$(5,5) ! Declare A2-D array
```


**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## DELETE

**Description** This command removes a specified file from the printer's memory.

**Format** DELETE <"filename">

**Parameters** <"filename"> = the name of the file to be deleted. Drive location and filename must be in quotation marks.

 **Example** • This is an example of deleting a specified file from printer memory:

```
>DELETE "E:PROGRAM1.BAS"
```

**Comments** This is an interactive command that takes effect as soon as it is received by the printer.

## DIR

**Description** This command, with no filter included, prompts the printer to list all the ZBI programs residing in all memory locations in the printer.

Including a filter signals the printer to limit the search; including a drive location signals the printer to search in only one location.

Asterisks (\*) are used as wild cards. A wild card (\*) finds every incidence of a particular request. The example here, DIR "B:\* .BAS", signals the printer to search for every file with a .BAS extension in B: memory.

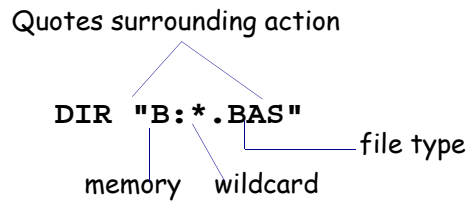
**Format** DIR ["filter"]

**Parameters** ["filter"] = the name of the file to be accessed (optional). Drive location and file name must be in quotation marks.





**Important** • Quotes must be around what you are doing. This shows you how to use the wildcard (\*) to search for all .BAS files in B: memory:



**Comments** This is an interactive command that takes effect as soon as it is received by the printer.

## DO-LOOP

**Description** Processing of the loop is controlled by a <WHILE/UNTIL> expression located on the DO or LOOP line.

Processing a WHILE statement is the same on either the DO or LOOP lines. The Boolean expression is evaluated and if the statement is true, the LOOP continues at the line after the DO statement. Otherwise, the line after the corresponding LOOP is the next in line to be processed.

Processing an UNTIL statement is the same on either the DO or LOOP lines. The Boolean expression is evaluated and if the statement is false, the LOOP continues at the line after the DO statement. Otherwise, the line after the corresponding LOOP is the next to be processed.

If <WHILE/UNTIL> is on the LOOP line, the BODY of the loop is executed before the Boolean expression is evaluated.

If neither the DO or LOOP line has a <WHILE/UNTIL> statement, the loop continues indefinitely.

DO-LOOPS could be nested but cannot overlap. The DO-LOOP command has two formats.

**Format 1**

```
DO <WHILE/UNTIL> <Boolean expression>
~~BODY~~
LOOP
```

➔ **Example Format 1** • This is an example of how to use the DO-LOOP command:

```
10 DO WHILE A$="70"
20 INPUT A$
30 LOOP
```

**Format 2**

```
DO
~~BODY~~
LOOP <WHILE/UNTIL> <Boolean expression>
```

➔ **Example Format 2** • This is an example of how to use the DO UNTIL LOOP command:

```
10 DO
20 INPUT A$
30 LOOP UNTIL A$="EXIT"
```

**Comments** This is a program command that is preceded by a line number.

**ECHO**

**Description** When Console Mode is enabled, this command controls whether the printer echoes the characters back to the communications port. If ECHO ON is entered, keystroke results return to the screen. If ECHO OFF is entered, keystroke results do not return to the screen.

**Format** ECHO <ON/OFF>

**Parameters** <ON/OFF> = toggles the ECHO command on or off

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## END

**Description** The END command terminates any program currently running. When the END command is received, the interpreter returns to Interactive Mode.

**Format** END

➔ **Example** • This is an example of how to use the END command:

```
10 PRINT "THIS PROGRAM WILL TERMINATE"  
20 PRINT "WHEN THE END COMMAND IS RECEIVED"  
30 END
```

**Comments** This is a program command and is preceded by a line number.

## ! (EXCLAMATION MARK)

**Description** The exclamation mark is the marker for adding comments to the end of numbered programming lines. Any text following the ! is ignored when the line or command is processed.

**Format** ![comment text]

➔ **Example** • This is an example of how to use the ! (comments) ommand:

```
10 LET A=10 ! Indicates number of labels to  
print
```

## EXIT

**Description** This command is used to exit the DO and FOR loops.

**Format** EXIT <DO/FOR>

**Comments** This is a program command that is preceded by a line number.

## FOR-LOOP

**Description** The FOR loop assigns the numeric variable to the value of the first expression. The NEXT line increments the numeric variable by the STEP amount.

If the numeric variable is less than the second expression after being incremented by the NEXT line, the program resumes running at the line following the FOR line.

When the STEP portion is omitted, the STEP value is 1. However, if the second expression is greater than the first expression, it defaults to -1.


### Format

```
FOR <# variable> = <# expression> TO <# expression>
STEP <# expression>]
~~BODY~~
NEXT <# variable>
```

### Parameters

<# variable> = indicates a numeric variable is used

<# expression> = indicates a numeric expression is used

 **Example** • This is an example of how to use the FOR LOOP command:

```
10 FOR X=1 TO 10 STEP 1
20 PRINT X; ":ZBI IS FUN"
30 NEXT X
```

**Comments** FOR-LOOPS could be nested but cannot overlap. Variables cannot be reused by the nested loops. This is a program command that is preceded by a line number.

## Functions

**Description** Functions built into this interpreter can be used in expressions only. The function names are not case sensitive.

If input parameters exist, they are enclosed in parentheses. If no parameters exist, no parentheses are used.

Variables referenced in the functions could be substituted by functions or expressions of the same type. If the function name ends with a \$, it returns a string value. Otherwise, it returns a numeric value.

Specifying the wrong type of parameter returns either Syntax Error or Poorly formed expression error.

## Integer Functions

The integer functions listed below return a numeric value:

### DATE

This function returns the current date in YYYYDDD format, where YYYY is the year and DDD is the number of days since the beginning of the year. If the Real-Time Clock is not installed, 0 is returned.

 **Example** • This example assumes the current date is January 1, 2003:

```
10 PRINT DATE  
RUN
```

The result is:

```
2003001
```

## DATAREADY(A)

This function returns the numeral 1 if data is ready on port A, and returns 0 if there is no data available.

➔ **Example** • This is an example of how to check if there is a data on a port:

```
10 PRINT DATAREADY(0)
```

```
RUN
```

The result, assuming no data is waiting, is:

```
0
```

## LEN(A\$)

This function returns the length of the string A\$.

➔ **Example** • This is an example tells you the length of a string in characters. Hello World is 10 characters, as follows:

```
10 LET A$="Hello World"
```

```
20 PRINT LEN(A$)
```

```
RUN
```

The result is:

```
11
```

## MAX(X,Y)

This function returns the maximum algebraic value of X or Y. If X is greater than Y, the value of X is returned. Otherwise, the value of Y is returned.

➔ **Example** • This is an example of how to use the MAX ( X , Y ) command:

```
10 LET A=-2
20 LET B=1
30 PRINT MAX(A,B)
RUN
```

The result is:

1

## MAXLEN(V\$)

This function returns the maximum length for the string V\$, which is always 255. This value is independent of V\$ but remains for compatibility with the ANSI specification.

➔ **Example** • This is an example of how to use the MAXLEN ( V\$ ) command:

```
10 LET A$="Hello"
20 PRINT MAXLEN(A$)
RUN
```

The result is:

255

## MAXNUM

This function returns the largest number represented by this machine: 2,147,483,647.

→ **Example** • This is an example of how to use the MAXNUM command:

```
10 PRINT MAXNUM  
RUN
```

The result is:

```
2147483647
```

## MIN(X,Y)

This function returns the minimum algebraic value of X or Y. If X is less than Y, the value of X is returned. Otherwise, the value of Y is returned.

→ **Example** • This is an example of how to use the MIN(X, Y) command:

```
10 LET A=-2  
20 LET B=0  
30 PRINT MIN(A, B)  
RUN
```

The result returned is:

```
-2
```



## MOD(X,Y)

This function returns X Modulo Y (same as the remainder of X/Y).

➔ **Example** • This is an example of how to use the MOD ( X , Y ) command:

```
10 LET A=9
20 LET B=2
30 LET C=-2
40 PRINT MOD(A,B)
50 PRINT MOD(C,A)
RUN
```

The result is:

```
1
-2
```

## ORD(A\$)

This function returns the ASCII value of the first character of string A\$.

➔ **Example** • This is an example of how to use the ORD ( A\$ ) command:

```
10 LET A$="ABC"
20 PRINT ORD(A$)
RUN
```

The result is:

```
65
POS(A$,B$)
```

This function returns the location of the first occurrence of B\$ in A\$. If there is no occurrence, the result is 0.

➔ **Example** • This is an example of how to use the ORD ( A\$ ) command:

```
10 LET A$="ABCDD"  
20 LET B$="D"  
30 PRINT POS ( A$ , B$ )  
RUN
```

The result is:

4

### **POS(A\$,B\$,M)**

This function returns the location of the first occurrence of B\$ in A\$ starting at the position of M. If there is no occurrence, the result is 0.

➔ **Example** • This is an example of how to use the POS ( A\$ , B\$ , M ) command:

```
10 LET A$="Hello World"  
20 LET B$="o"  
30 PRINT POS ( A$ , B$ , 6 )  
RUN
```

The result is:

8

## TIME

This function returns the time past midnight (2400h) in seconds. If the Real-Time Clock is not installed, 0 is returned.

➔ **Example** • This is an example of how to use the `TIME` command:

```
10 PRINT TIME
RUN
```

The result, assuming the time is one minute past midnight, is:

```
60
```

## VAL(A\$)

This function returns the numeric value represented by `A$`. The conversion is done in the same fashion as the `INPUT` command.

➔ **Example** • This is an example of how to use the `VAL(A$)` command:

```
10 LET A$="123"
20 LET C=VAL(A$)
30 PRINT C
RUN
```

The result is:

```
123
```

## String Functions

The string functions listed below return a string value:

### CHR\$(M)

This function returns the character at position M of the extended ASCII table. Using values of M greater than 255 value modulated by 255. M could not be 0. The numeral 1 is substituted.

➔ **Example** • This is an example of how to use the CHR\$(M) command:

```
10 LET A=97
20 PRINT CHR$(A)
30 PRINT CHR$(353) !Note that 353 is
modulated to 97
RUN
```

The result is:

```
a
a
```

### DATE\$

This function returns the current date in string form YYYYMMDD. If the Real-Time Clock is not installed, no data returns.

➔ **Example** • This is an example of how to use the DATE\$ command:

```
10 PRINT DATE$
RUN
```

The result, assuming the date is January 1, 2003 is:

```
20030101
```

## EXTRACT\$

This function returns a string.

- **Source** can be a port or a variable string name.
- **Beginning string** is the first string or character that is encountered to trigger the command.
- **Ending string** is the last string or character encountered to stop the command.



**Important** • If the EXTRACT\$ command encounters a carriage return line feed before encountering the beginning character or the ending character, it returns null.

The EXTRACT\$ command is used to get data from between two known points. For example, data in a comma-delimited file always has commas between records.



**Example** • This example shows how to extract the word Technologies from this string: Zebra,Technologies,Corporation.

This is what the program looks like to accomplish this:

```
10 LET A$ = Zebra,Technologies,Corporation,
20 LET DATA$ = EXTRACT$(A$,"","","")
```



**Example** • This example shows how the EXTRACT\$ command works from an open port:

```
10 OPEN #1: NAME "SER"
20 LET DATA$ = EXTRACT$(1,"","","")
```

Notice how the quotes are used to show a literal character, in this case a comma.



**Example** • This example shows how the start and stop points are variable; a variable name is used instead of the literal:

```
10 LET B$ = ","
20 LET A$ = Zebra,Technologies,Corporation
30 LET DATA$ = EXTRACT$(A$,B$,B$)
```

## LTRIM\$(A\$)

This function returns the string A\$ with all leading spaces removed.

→ **Example** • This is an example of how to use the LTRIM\$(A\$) command:

```
10 LET A$=" Hello"  
20 PRINT LTRIM$(A$)  
RUN
```

The result is:

```
Hello
```

## REPEAT\$(A\$,M)

This function returns a string containing M copies of A\$.

→ **Example** • This is an example of how to use the REPEAT\$(A\$,M) command:

```
10 LET X=3  
20 LET A$="Hello"  
30 PRINT REPEAT$(A$,X)  
RUN
```

The result is:

```
HelloHelloHello
```

## RTRIM\$(A\$)

This function returns a string created from A\$ with trailing spaces removed.

➔ **Example** • This is an example of how to use the RTRIM\$(A\$) command:

```
10 LET A$="Hello "  
20 LET B$="World"  
30 PRINT RTRIM$(A$);  
40 PRINT B$  
RUN
```

The result is:

```
HelloWorld
```

## STR\$(X)

This function converts numeric type X to a string.

➔ **Example** • This is an example of how to use the STR\$(X) command:

```
10 LET A=53  
20 PRINT STR$(A)  
RUN
```

The result is:

```
53
```

## TIME\$

This function returns the time of day in format HH:MM:SS (hours:minutes:seconds). If the Real-Time Clock is not installed, no data is returned.

➔ **Example** • This is an example of how to use the TIME\$ command:

```
10 PRINT TIME$
RUN
```

The result, assuming it is 10 a.m., is:

```
10:00:00
```

## UCASE\$(A\$)

This function returns a string created from A\$ with all characters in uppercase. Noncharacter values are not changed.

➔ **Example** • This is an example of how to use the UCASE\$(A\$) command:

```
10 LET A$="Zebra Technologies"
20 PRINT UCASE$(A$)
RUN
```

The result is:

```
ZEBRA TECHNOLOGIES
```

## GOTO

**Description** The GOTO statement is used to direct the interpreter to a specific line number. GOTO is followed by a line number that the program attempts to process next. Upon executing the GOTO statement, the interpreter continues running at the line number specified following GOTO. If the line number referenced does not exist, an error message displays — Error: Line does not exist.

**Format** GOTO



➔ **Example** • This is an example of how to use the GOTO command:

```
10 PRINT "Zebra Printers"  
20 GOTO 10
```

**Comments** The result displays `Zebra Printers` until the program is halted. This is a program command and must be preceded by a line number.

## GOSUB-RETURN

**Description** GOSUB is followed by a line number that the program attempts to process next. Upon executing the GOSUB statement, the interpreter continues running at the line number specified following GOSUB. If the line number referenced does not exist, an error message displays — Error: Line does not exist.

Before executing the next line, the GOSUB command stores the line number of the GOSUB line. When the RETURN statement is called, the program moves back to the next line following the GOSUB.

Executing a RETURN statement without a corresponding GOSUB statement causes an error message to display — Error: Invalid RETURN statement.

GOSUB statements can be nested.

### Format

```
GOSUB  
RETURN
```

➔ **Example** • This is an example of how to use the GOSUB-RETURN command:

```

10 PRINT "Call Subroutine"
20 GOSUB 1000
30 PRINT "Returned from Subroutine"
40 END

1000 PRINT "In Subroutine"
1010 RETURN

```

**Comments** These are program commands and must be preceded by line numbers.

## IF Statements

**Description** If the value of the <Boolean expression> in an IF statement is true and a program line follows the keyword THEN, this program line is executed. If the value of the Boolean expression is false and a program line follows the keyword ELSE, this program line is executed. If ELSE is not present, then execution continues in sequence, with the line following the END IF statement.

Nesting of blocks is permitted, subject to the same nesting constraints as DO-LOOPS (no overlapping blocks).

ELSE IF statements are treated as an ELSE line followed by an IF line, with the exception that the IF shares the END IF line of the original IF statement.

### Format

```

IF <Boolean expression> THEN
~~BODY~~
[ELSE IF <Boolean expression> THEN
~~BODY~~]*
[ELSE
~~BODY~~]
END IF

```

➔ **Example** • This is an example of how to use the IF statement command:

```

10 IF A$="0" THEN
20 PRINT "ZBI IS FUN"
30 ELSE IF A$="1" THEN
40 PRINT "ZBI IS EASY"
50 ELSE
60 PRINT "X=0"
70 END IF

```

**Comment** This is a program command that is preceded by a line number.

## INBYTE

**Description** This command forces the interpreter to pause until data is available. Use the DATAREADY function to determine if there is data on the port.

**Format** INBYTE [channel expression:] <A>

**Parameter** <A> = numeric or string expression. The received value replaces the current value held in the variable.

➔ **Example** • This is an example of how to use the INBYTE statement command:

```

10 INBYTE A$ !takes one byte (char) from
port #1
20 PRINT A$ !prints the character to the
console

```

**Comments** In this example, the interpreter pauses until the data is entered and then continues processing. This command enters all bytes in a string or integer, including control codes.

This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## INPUT

**Description** If the variable is numeric and the value entered cannot be converted to a number, it writes as 0. This operation scans the data from left to right, shifting any number into the variable. It ignores any other character except the return character, which terminates the input, or Ctrl-C (^C) which terminates the program. The variable can be in string or numeric form.

### Format

```
INPUT [<channel expression>:] <variable>
[ ,variable]*
```

If the [<channel expression>:] is omitted, the default port is 0. If an invalid port is specified, an error condition returns in the form of Error: Invalid port.

➔ **Example 1** • This is an example of how to use the INPUT command:

```
10 INPUT A
20 PRINT A
```

If you entered 1234567891011, the number sent to the display is modulated by the MAXNUM value.

➔ **Example 2** • This is an example of how to use the INPUT command:

```
10 OPEN #1: NAME "ZPL"
20 PRINT #1: "~HS"
30 FOR I = 1 TO 3
40 INPUT #1: A$
50 PRINT A$
60 NEXT I
```

In Example 2, a host status prints to the console after submitting the host status request ~HS to the ZPL port.

The Input/Output command of the ZBI interpreter is limited to the communications ports. File I/O is not supported.

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## LET

**Description** The LET command is used to assign value to a specific variable. The expression is evaluated and assigned to each variable in the variable list.

**Format** LET <variable> [, <variable>]\* = <expression>

The variable types must match the expression type or an error message displays — Error: Variable types must be the same.

When a value is assigned to a string variable with a substring qualifier, it replaces the value of the substring qualifier. The length of the value of the string variable might change as a result of this replacement.



**Examples** • This is an example of how to use the LET command:

```
10 LET A$= "1234"
```

```
15 LET A$(2:3)= "55" ! A$ NOW = 1554
```

```
20 LET A$(2:3)= "" ! A$ NOW = 14
```

```
10 LET A$= "1234"
```

```
15 LET A$(2:3)= A$(1:2) ! A$ NOW = 1124
```

```
10 LET A$= "1234"
```

```
20 LET A$(2:1)= "5" ! A$ NOW = 15234
```

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## LIST

**Description** This command numerically lists the program lines currently in memory.

**Format** LIST [RANGE]

### Parameters

[RANGE] = an optional request for specific lines (or a line) of code.  
[RANGE] is a single pair of numbers separated by a hyphen.

➔ **Examples** • This is an example of how to use the LIST command:

```
LIST 20
LIST 90-135
```

The printer returns line 20, or in the second example, lines 90 through 135.

**Comments** This is an interactive command that takes effect as soon as it is received by the printer.

## LOAD

**Description** This command transfers a program file previously stored in the printer's memory and opens it in the ZBI Program Memory.

If the program file does not exist, the ZBI Program Memory is cleared and no program is opened. An error message displays — Error: Invalid file name.

**Format** LOAD <"filename">

**Parameter** <"filename"> = the name of the file to be loaded into memory. Drive location and file name must be in quotation marks.

➔ **Example** • This is an example of how to use the LOAD command:

```
LOAD "PROGRAM1.BAS"
LOAD "E:PROGRAM1.BAS"
```

**Comments** This is an interactive command that takes effect as soon as it is received by the printer.

## NEW

**Description** This command clears the interpreter's memory, including the line buffer and variables, but not any open ports.

**Format** NEW

→ **Example** • This is an example of how to use the NEW command:

```
NEW
```

**Comments** This is an interactive command that takes effect as soon as it is received by the printer.

## NOT

**Description** The NOT statement is a Boolean operator. If the Boolean expression is true, the result is false. If the Boolean expression is false, the result is true.

**Format** NOT <Boolean expression>

→ **Example** • This is an example of how to use the NOT command:

```
10 LET A=1
20 IF NOT A=0 THEN
30 PRINT A
40 END IF
RUN
```

**Comments** This is a Boolean operator that is used in conjunction with Boolean expressions.

## Numeric Expressions

**Description:** Base numerical expression can be either a constant, variable, or another numerical expression closed in by parentheses. Overflow cannot be detected. The result is undefined. The five types used (addition, subtraction, multiplication, division, and exponentiation) are listed below.

**1** + (addition) Addition expressions use this format:

`<numerical expression>+<numerical expression>`

**2** - (subtraction) Subtraction expressions use this format:

`<numerical expression>-<numerical expression>`

**3** \* (multiplication) Multiplication expressions use this format:

`<numerical expression>*<numerical expression>`

**4** / (division) Division expressions use this format:

`<numerical expression>/<numerical expression>`

**5** ^ (exponentiation) Exponentiation expressions use this format:

`<numerical expression>^<numerical expression>`

In mathematics, order of precedence describes in what sequence items in an expression are processed. All expressions have a predefined order of precedence.

The order of precedence is ^, \*, /, +, -.

\* and / have the same precedence, and + and - have the same precedence. Items with the same order of precedence are processed from left to right.

For example, this expression  $5+(8+2)/5$  is processed as  $8+2=10$ , followed by  $10/5=2$ , then  $5+2$  to give a result of 7.

Functions and parenthesis always have the highest order of precedence, meaning that they are processed first. The remaining items are specific to the type of expression (Boolean, numeric, and string).

*Maximum value:* 2,147,483,647

*Minimum value:* -2,147,483,648



## Comments


- No floating point is supported.
- Variable names must start with a letter and can include any sequence of letters, digits, and underscore.
- Function and command names might not be used as variable names.
- Variable names are not case sensitive and are converted to uppercase by the interpreter.
- When using division, the number is always rounded down. For example,  $5/2=2$ .

## ON ERROR

**Description** The ON ERROR command can be used to prevent a program from halting in the event of an error. If an error occurs in a previous line during program execution, the ON ERROR statement calls the GOTO or GOSUB statement and allows the program to continue.

**Format** ON ERROR <GOTO/GOSUB> LN

If there is no error, this line is ignored.

 **Example** • This is an example of how to use the ON ERROR command:

```
30 LET A = B/C
40 ON ERROR GOTO 100
...
100 PRINT "DIVIDE BY ZERO OCCURRED"
110 LET A = 0
120 GOTO 50
...
```

**Comments** This is a program command that is preceded by a line number.

## Open

**Description** This command is used to open a port for transmitting and receiving data.

### Format

```
OPEN #<channel expression>: NAME <string expression>
[, ACCESS <ACCESS type>]
```

### Parameters

<channel expression> =

*Accepted Values: 0 to 9*

*Default Value: a port must be specified*

<string expression> = port name to open (SER, PAR, or ZPL)

<ACCESS type> =

INPUT for receiving only,

OUTPUT for transmitting only, and

OUTIN for transmitting and receiving.

A channel must be specified; the default value cannot be used. Access type OUTIN is used if access type is not specified.



**Example** • This is an example of how to use the OPEN command:

```
10 OPEN #1: NAME "ZPL"
```

If there are conflicts opening the port, an error message displays – Error: Unable to open port. If the port is already open, an error message displays — Error: Port already opened.

The port being opened no longer allows data to pass directly into its buffer, it disconnects, and the interpreter now controls the data flow.

Data already in the buffer stays in the buffer.

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## OR

**Description** The OR statement is a Boolean operator. If either of the expressions are true, the result is true; otherwise the result is false.

### Format

```
<Boolean expression> OR <Boolean expression>
```

**Comments** This is a Boolean operator that is used in conjunction with Boolean expressions.


## OUTBYTE

**Description** This command outputs all bytes in a string, including control codes.

**Format** OUTBYTE [<channel expression>:] <A>

**Parameters** <A> = a numeric or string expression.

If the parameter is a numeric expression, it must be a value of 0 through 255. If not, it is truncated. For a string, the first character is used. In case of a NULL string, 0 is sent.

 **Example** • This is an example of how to use the OUTBYTE command:

```
Let A$="Hello"  
OUTBYTE A$
```

The result is:

```
H
```

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## Ports <CHANNEL EXPRESSION>

**Description** Each printer has a list of ports that might be opened by the interpreter.

Each port is given a three-letter abbreviation that is used with the OPEN command to open that port. The standard port names are SER for the serial port and PAR for the parallel port. The port reference ZPL opens a channel to the printer's formatting engine that functions as a port not controlled by the ZBI interpreter.

## PRINT

**Description** This command sends data to the printer to be printed.

**Format** PRINT [channel expression:] <expression> [,or;  
<expression>]\* [;]

The expression can be either a string or a numeric expression.

Using a , to separate expressions adds a space between them.

Using a ; to separate expressions does not put a space between them.

Using a ; at the end of a line ends the print statement without a new line.

➔ **Example** • This is an example of how to use the PRINT command:

```
10 LET A$= "This is an example"
20 LET B$= "of the PRINT Command."
30 PRINT A$, B$ ! adds a space between
   expressions
40 PRINT A$; B$ ! no space added
RUN
```

The result is:

```
This is an example of the PRINT Command.
This is an exampleof the PRINT Command.
```

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.

## REM

**Description** A numbered **remark** line is started with REM and can include any text in any form after it. This line is ignored by the interpreter.

**Format** REM [ comment ]



**Example** • This is an example of how to use the REM command:

```
10 REM COMMAND LINES 20-100 PRINT A LABEL
```

**Comments** Remarks are used for program description and can be included as a separate program line or appended to the end of a program line. Also used for internal comments is the exclamation mark (!) statement.

## RENUM

**Description** This command renumbers the lines of the program being edited. RENUM can reorganize code when line numbers become over- or under-spaced. The line references following GOTO and GOSUB statements are renumbered if they are constant numeric values. Renumbering does not occur if the line number are outside of the range limits of 1 to 10000.

**Format** RENUM <A> , <B>

### Parameters

<A> = the number to start the renumbering sequence

<B> = the automatic increment between the new line numbers

→ **Example** • This is an example of how to use the RENUM command:

```
13 LET A=6
15 LET B=10
17 GOTO 13

>RENUM 10,5

0 LET A=6
15 LET B=10
20 GOTO 10
```

**Comments** This is an interactive command that takes effect as soon as it is received by the printer.

## RESTART

**Description** If a program was halted by Ctrl-C or a BREAK command, the RESTART command can be used to reactivate the program from the point it stopped. RESTART functions similarly to RUN, except the program attempts to restart from the point it was last terminated. It also works in conjunction with the STEP command, picking up where STEP ends.

**Format** RESTART

**Comments** If the program has not been run or is finished, RESTART runs the program from the beginning. This is an interactive command that takes effect as soon as it is received by the printer.

## RUN

**Description** This command allows the program being edited to be activated, starting with the lowest line number. If a line does not specify a new line to go to, the next line

in numeric order is processed. When a higher line number does not exist, the RUN command stops.

**Format** RUN

➔ **Example** • This is an example of how to use the RUN command:

```
10 PRINT "ZBI "  
15 PRINT "Programming"  
RUN
```

The result is:

```
ZBI  
Programming
```

**Comments** Ports that are open when the application is activated remains open after the application has terminated. Variables also remains after the application has terminated.

This is an interactive command that takes effect as soon as it is received by the printer.

## SEARCHTO\$(A,B\$)

**Description** This function performs a search up to a string, which is defined by B\$ on port A. The string the search yields is displayed.

**Format** SEARCHTO\$(A,B\$)

### Parameters

A = port number (0 to 9) to which requested data is sent.

B\$ = string variable or string array. If B\$ is an array, this command searches for all non-null strings in the B\$ array.

➔ **Example** • This is an example of how to use the SEARCHTO\$(A,B\$) command:

```
10 LET A$=SEARCHTO$(0,"Hello")
```

There are several ways to display that `Hello World` is the text string being searched in this example. `A$` then equals `Hello`, the program performs the search, and only `World` remains on the port.

## SEARCHTO\$(A,B\$,C)

**Description** This function works similarly to `SEARCHTO$(A,B$)` and performs a search up to a string, which is defined by `B$` on port `A`. The string the search yields is displayed. Unused characters up to the search string are sent to port `C`.

**Format** `SEARCHTO$(A,B$,C)`

### Parameters

`A` = port number (0 to 9) the requested string is sent to.

`B$` = string variable or string array. If `B$` is an array, this command searches for all non-null strings in the `B$` array.

`C` = port number (0 to 9) unused characters are sent to.

➔ **Example** • This is an example of how to use the `SEARCHTO$(A,B$,C)` command:

```
LET B$ = "Hello"
10 LET A$ = SEARCHTO$(0,"Hello",1)
```

There are several ways to display that `Hello World` is the text string being searched for this example. `A$` is then equal `Hello`, the program performs the search, and only `World` remains on the port. The unused characters are sent to port 1.

➔ **Example** • This example declares an array of variables, functions the same way only the list of possible variables is expanded:

```
DECLARE STRING B$(4)
LET B$(1) = "Hello"
LET B$(2) = "HELLO"
LET B$(3) = "hello"
LET B$(4) = "Hi"
10 LET A$ = SEARCHTO$(0,B$,0)
```



## SETERR

**Description** This command sends a message to the printer to set the error flag. A logical interpreter flag is triggered in the printer. This error is referenced as a BASIC Forced Error.

**Format** SETERR

**Comments** This is a program command and must be preceded by a line number.

## SLEEP

**Description** This command specifies the time that the interpreter pauses for label generation to receive greater priority. This command could be sent to the printer after sending a label format to be printed. The interpreter pauses in its processing for the amount of time specified.

**Format** SLEEP <A>

**Parameters** <A> = the time in seconds (0 to 500) the interpreter pauses.



**Example** • This is an example of how to use the SLEEP command:

```
10 SLEEP 450
```

**Comments** This is a program command and must be preceded by a line number.

## STEP

**Description** If a program was stopped by a BREAK command, STEP attempts to execute the program one line from where it last ended. If the program has not been run or has been completed, this executes the lowest numbered line.

**Format** STEP

➔ **Example** • This is an example of how to use the STEP command:

```
10 PRINT "Hello World"
20 BREAK
30 PRINT "30"
```

Entering STEP causes line 10 to be executed.

Entering RUN followed by STEP causes:

RUN — Execute to line 20 and stop.

STEP — Execute line 30 and stop.

**Comments** This is an interactive command that takes effect as soon as it is received by the printer.

**STORE**

**Description** This command saves the program currently in memory as the specified file name. This format is used:

```
STORE <"filename">
```

**Parameters** <"filename"> = the name of the file to be stored. Drive location and file name must be in quotation marks.

➔ **Example** • This is an example of how to use the STORE command:

```
STORE "E:ZEBRA1.BAS"
```

**Comments** For a file name to be valid, it must conform to the 8.3 Rule: each file must have no more than eight characters in the file name and have a three-character extension. Here the extension is always .BAS (for example, MAXIMUM8.BAS).

This is an interactive command that takes effect as soon as it is received by the printer.

## STRING CONCATENATION (&)

**Description** The basic string expression could be either a constant or a variable, and the concatenation (&) is supported.

Using the concatenation operator adds the second string to the first string.

**Format** <string expression> & <string expression>

→ **Example** • This is an example of how to use the STRING CONCATENATION (&) command:

```
10 LET A$= "ZBI "  
20 LET B$= "Programming"  
30 LET C$= A$ & B$  
40 PRINT C$
```

The result is:

```
ZBI Programming
```

**Comments** If the concatenation causes the string to be greater than the maximum possible length of a string (255 characters), the first string is returned and an error message displays — Error: String size limit exceeded.

## STRING VARIABLE

**Description** Maximum length: 255 characters

Variable names must start with a letter and can include any sequence of letters, digits, and underscore. The variable ends with a \$.

Function and command names might not be used as a variable name.

Variable names are not case sensitive and are converted to uppercase by the interpreter.

## UB-STRINGS

**Description** Using a substring operator on a string allows a specific portion of the string to be accessed. To determine the coordinates of the string portion to be used, count the spacing from the beginning to the end of the string, including spaces.

**Format** StringVariable\$(A:B)

### Parameters

A = the position of the first character in the desired string.

B = the position of the last character in the desired string.

➔ **Example** • This is an example of how to use the UB-STRINGS command:

```
10 LET A$="Zebra Quality Printers"
20 LET B$=A$(1:13)
30 PRINT B$
```

The result is:

```
Zebra Quality
```

To calculate the position of Zebra Quality Printers in line 10 above, Z occupies position 1, e occupies position 2, b occupies position 3, r occupies 4, a occupies 5, the space occupies 6, and so on.

**Comments** If the A parameter is less than 1, it is automatically assigned a value of 1. Because the string is calculated starting with 1, the A parameter cannot be less than 1.

If B is greater than the length of the string, it is replaced with the length of the string. In this example, the B parameter can be no greater than 22.

If A is greater than B, a NULL string (" "), which points to the location of the smaller of A or the end of the string, is returned. This is used when adding a string in the middle of another string without removing a character. Refer to the LET command.

## TRACE

**Description** This command is only valid when the DEBUG function is active.

**Format** TRACE <ON/OFF>

### Parameters

<ON/OFF> = controls whether TRACE is active (on) or disabled (off).

If DEBUG is activated and the TRACE command is on, trace details are displayed. When any variables are changed, the new value displays as follows:

Variable\$ = New Value

Every line processed has its line number printed.

If DEBUG is on, the line number prints before the command line is executed and variables print as their values are assigned.

```
10 LET A=5
20 GOTO 40
30 PRINT "Error"
40 PRINT A
RUN
```

The output is:

```
<TRACE> 10
<TRACE> A=5
<TRACE> 20
<TRACE> 40
5
```

**Comments** This can be an interactive command that takes effect as soon as it is received by the printer, or a program command that is preceded by a line number.



# ZBI — Tutorial

This tutorial provides you with examples of how Zebra BASIC Interpreter (ZBI) and Zebra Programming Language II (ZPL II) interact in bar code printing.

It is recommended that you run several examples to gain an understanding of how the interpreter works in conjunction with your printer.

ZBI allows you to maximize bar code printing options through custom programs written for a specific need.

To its advantage, ZBI bears a strong resemblance to many other BASIC interpreters and does not require you to learn an entirely new language from the ground up to take advantage of its features.

## Starting ZBI

ZBI is enabled by sending a ZPL II command to the printer via one of the communication ports (serial, parallel, or Ethernet) in an interactive mode. This communication port is referred to as the console. Programs like HyperTerminal for Windows can be used to communicate with your printer.

These ZPL II commands are available to start the ZBI Interpreter: `~JI` and `^JI`. When the printer is turned on, it accepts ZPL II commands and label formats. However, to receive ZBI commands the printer must be initialized using `~JI` or `^JI`.

With a console application open, send one of these two commands to the printer:

- 1** `~JI` — when this command is received, the printer responds by sending a ZBI header with the program version and a new line containing an input prompt (`>`) back to the screen. This is an indication that the ZBI program lines or commands can be sent to the printer. While receiving programming commands, the printer echoes the received characters back to the source. This echoing can be turned off with the ECHO command.
- 2** `^JI:d:o:x,b,c,d` — sending this command to the printer activates a ZBI session. The parameters have these functions:

`d` = location of program to run after initialization

*Accepted Values:* R:, E: and B:

*Default Value:* location must be specified

`o` = name of program to run after initialization

*Accepted Values:* any valid program name

*Default Value:* a name must be specified

`x` = extension of program to run after initialization

*Fixed Value:* .BAS

`b` = console control

*Accepted Values:* Y (console on) or N (console off)

*Default Value:* Y

`c` = echoing control

*Accepted Values:* Y (echo on) or N (echo off)

*Default Value:* Y



d = memory allocation for ZBI

*Accepted Values:* 20K to 1024K

*Default Value:* 50K

If a second ~JI or ^JI command is received while the interpreter is running, the command is ignored.

## ZBI Interpreter Modes

The ZBI Interpreter operates in these modes:

- The interactive mode allows commands to be entered and processed immediately. Command lines entered without line numbers are interactive and processed as soon as they are received by the printer.

➔ **Example** • This is an example of how to use the NEW command:

```
NEW
```

- The program mode allows a series of command lines to be stored in memory and processed in numerical order when a RUN command is entered. Line numbers must be greater than 0 and less than 10,000.

➔ **Example** • This is an example of

```
10 LET A=10
```

```
20 LET B=3
```

A program is activated from the first numbered line by entering the RUN command. While the program is running, it can be halted with the Control-C (^C) command, sent to the printer from the console. The interrupted program continues where it left off by entering the RESTART command.

## Ending A ZBI Session

An active ZBI session can be halted in one of these ways:

- Sending ZPL at the input prompt (>).
- Sending ~JQ at the input prompt (>).

## Encrypting ZBI Files

The `.baz` extension provides security for all ZBI executables. The `.baz` extension is created when the ZBI store command is used.

`^HZO` = uploads `.baz` files to the printer.



**Important** • For security purposes, users cannot:

- open `.baz` files
- upload or download `.baz` files to `.bas` file format



**Example** • This is an example on how to generate a `.baz` file.

- 1 Create basic program as in the single line program below.
- 2 Type `10 print "ZPL"`.
- 3 Save the file with the filename: `Store E:ZEBRA1.BAZ`
- 4 Recall the newly created file from the printer using the `^HZO` command, as follows:  
  
`^XA^HZO , E : ZEBRA1 . BAZ ^XZ`
- 5 To use the new file, download the data received that follows the `^HZO` command to any printer.

**Limitations** These commands do not work when loading a ZBI executable:

- List
- Trace
- Debug

## Basic ZBI Examples

This section provides you with basic examples of ZBI:

- ➔ **Example 1** • This example shows you the choices you have: print a configuration label, print text, or calibrate the printer. The choice you make corresponds with the ZPL that is sent to the printer:

```
 5 REM PORTS SHOULD BE CLOSED, BY ROUTINE, BEFORE OPENING
10 CLOSE # 1
20 CLOSE # 2
25 REM OPENING PORT
40 OPEN # 1 : NAME "ZPL"
45 REM THE FOLLOWING WILL BE DISPLAYED ON THE TERMINAL
50 PRINT "YOU HAVE THREE CHOICES"
51 PRINT "PRESS 1 TO PRINT CONFIG LABEL" !prints to the console
52 PRINT "PRESS 2 TO PRINT TEXT "
53 PRINT "PRESS 3 TO CALIBRATE"
54 PRINT "OR PRESS 7 TO EXIT"
58 REM THE FOLLOWING ARE THE POSSIBLE INPUTS
60 INPUT A !PRINTER IS EXPECTING AN INPUT
61 IF A = 1 THEN
62 GOTO 120
63 ELSE
65 IF A = 2 THEN
66 GOTO 130
67 ELSE
70 IF A = 3 THEN
71 GOTO 140
72 ELSE
75 IF A = 7 THEN
76 GOTO 300
80 END IF
81 GOTO 51
85 REM IF THERE IS AN IF STATEMENT ALWAYS HAVE AN ENDIF
95 REM THE FOLLOWING WILL PRINT A CONFIGURATION LABEL
```

```
120 PRINT # 1 : "~WC" !~WC TELLS THE PRINTER TO PRINT CONFIGURATION
    LABEL
125 GOTO 50
130 PRINT "THIS TASK WILL SEND SAMPLE TEXT TO THE PRINTER"
131 PRINT "INPUT YOUR SAMPLE TEXT"
132 INPUT Y$
133 REM THE FOLLOWING STATEMENT WILL PRINT THE Y$ STRING VALUE
135 PRINT # 1 : "^XA^FO20,20^A0N50,50^FD" ; Y$ ; "^XZ"
136 GOTO 50
139 REM THE FOLLOWING COMMAND WILL TELL THE PRINTER TO CALIBRATE
140 PRINT # 1 : "~JG" !~JG ZPL COMMAND BEING SENT TO THE PRINTER
141 GOTO 50
300 PRINT "BEFORE CLOSING THE PRINTER WILL NOW PAUSE FOR 10 SECONDS"
301 SLEEP 10 ! THE SLEEP COMMAND PAUSES THE PRINTER
310 END
```

➔ **Example 2** • This example shows ZBI code using extract and searchto commands:

```

10 REM ALWAYS CLOSE PORTS BEFORE OPENING (ROUTINE)
20 CLOSE # 1
30 CLOSE # 2
60 OPEN # 1 : NAME "ZPL"
69 REM SPECIFYING VALUE FOR STRING BELOW
70 LET B$ = "START"
79 REM USING SEARCHTO$ COMMAND TO SEARCH STREAMING DATA FOR B$ OR
  THE WORD "START"
80 LET Z$ = SEARCHTO$ ( 0 , B$ )
89 REM SPECIFYING VALUES FOR THE STRINGS F$ AND G$ ("T" AND "W ")
90 LET F$ = "T "
100 LET G$ = " W"
108 REM USING THE EXTRACT$ COMMAND TO SEARCH STREAMING DATA FOR F$
  AND G$
109 REM X$ WILL BE THE DATA BETWEEN F$ AND G$
110 LET X$ = EXTRACT$ ( 0 , F$ , G$ )
250 PRINT # 1 : "^XA^F025,25^A050,50^FD" & X$ & "^FS^XZ"
1000 END

```

➔ **Example 3** • This example shows ZBI code using ARRAYS:

```
5 REM PORTS SHOULD BE CLOSED, BY ROUTINE, PRIOR TO OPENING
10 CLOSE #1
20 CLOSE #2
40 OPEN #1: NAME "ZPL"
45 REM WHEN USING ARRAYS, THE ARRAY HAS TO BE DECLARED BEFORE
    USING
49 REM THIS ARRAY "NAME$" IS BEING DEFINED WITH 5 SPOTS FOR DATA
50 DECLARE STRING NAME$ (5)
57 REM THE FOLLOWING ALLOWS ONE TO INPUT 5 DIFFERENT VALUES AND
    THE VALUES
58 REM ARE PLACED INTO THE ARRAY IN ORDER (1-5)
59 PRINT "INPUT YOUR FIRST 5 NAMES"
60 FOR INDEX = 1 TO 5 STEP 1
70 INPUT NAME$(INDEX)
80 NEXT INDEX
89 REM THE FOLLOWING DECLARES AN ARRAY (NUMBERS$) AND ALLOWS 5
    SPOTS FOR DATA
90 DECLARE STRING NUMBERS$ (5)
99 PRINT "INPUT YOUR FIRST 5 NUMBERS"
100 FOR INDEX = 1 TO 5 STEP 1
110 INPUT NUMBERS$(INDEX)
120 NEXT INDEX
128 REM THE FOLLOWING PRINTS OUT THE VALUES THAT HAVE BEEN STORED
    IN THE ARRAYS
130 PRINT #1 : "^XA^FO60,60^A0N50,50^FD"&NAME$(1)&"^FS";
140 PRINT #1 : "^FO60,120^A0N50,50^FD"&NUMBERS$(1)&"^FS^XZ"
150 PRINT #1 : "^XA^FO60,60^A0N50,50^FD"&NAME$(2)&"^FS";
160 PRINT #1 : "^FO60,120^A0N50,50^FD"&NUMBERS$(2)&"^FS^XZ"
170 PRINT #1 : "^XA^FO60,60^A0N50,50^FD"&NAME$(3)&"^FS";
180 PRINT #1 : "^FO60,120^A0N50,50^FD"&NUMBERS$(3)&"^FS^XZ"
190 PRINT #1 : "^XA^FO60,60^A0N50,50^FD"&NAME$(4)&"^FS";
200 PRINT #1 : "^FO60,120^A0N50,50^FD"&NUMBERS$(4)&"^FS^XZ"
210 PRINT #1 : "^XA^FO60,60^A0N50,50^FD"&NAME$(5)&"^FS";
220 PRINT #1 : "^FO60,120^A0N50,50^FD"&NUMBERS$(5)&"^FS^XZ"
300 END
```

➔ **Example 4** • This ZBI code shows integer functions:

```

10 REM THIS PROGRAM WILL SHOW VARIOUS INTEGER FUNCTIONS
11 CLOSE # 1
12 CLOSE # 2
20 PRINT "ENTER YOUR LAST NAME"
25 INPUT A$ ! PRINTER IS EXPECTING A STRING INPUT
30 LET X = LEN ( A$ ) !THIS IS WHERE THE LENGTH IS DETERMINED
35 PRINT "YOUR LAST NAME ( " ; A$ ; " ) CONTAINS " ; X ; " CHARACTERS"
36 PRINT
40 PRINT "ENTER 2 DIFFERENT NUMBERS EACH FOLLOWED BY THE ENTER KEY"
45 INPUT A
46 INPUT B
55 LET Y = MIN ( A , B ) !THIS IS WHERE THE SMALLEST NUMBER IS
   DETERMINED
60 PRINT "THE SMALLEST NUMBER THAT YOU ENTERED WAS " ; Y
61 PRINT
65 PRINT "ENTER ANY ASCII CHARACTER FROM YOUR KEYBOARD"
66 INPUT B$
70 LET U = ORD ( B$ ) ! THIS IS WHERE THE ASCII EQUIVELANT IS
   DETERMINED
71 LET R = 34
75 PRINT "THE ASCII EQUIVELANT OF " , CHR$ ( R ) , B$ , CHR$ ( 290 )
   , "IS" , U !THIS LINE SHOWS TWO DIFFERENT WAYS TO PRINT THE "
   CHARACTER USING THE CHR$ FUNCTION
76 REM IN LINE 75 I AM USING " , " UNLIKE THE ";" THAT WAS USED IN LINE
   35
78 REM WHEN SEPERATING WITH THE " , " CHARACTER A SPACE WILL BE
   INCLUDED
79 REM IF USING THE ";" NO SPACE WILL BE INCLUDED AS IN LINE 34
80 END

```

➔ **Example 5** • This example shows ZBI code showing Numeric expressions:

```
10 REM This program will demonstrate the use of Numeric
    expressions
11 CLOSE # 1
12 CLOSE # 2
20 REM SIMPLE PROGRAM SHOWING HOW TO USE NUMERIC EXPRESSIONS
28 PRINT
30 PRINT "The first example will be addition, subtraction,
    mulitiplication , division and exponentiation"
31 PRINT " Input your first number"
32 INPUT A
33 PRINT " Input your second number"
35 INPUT B
38 LET X = A + B !simple math (addition)
39 LET Y = A - B !simple math (subtraction)
40 PRINT "If you add" , A , "plus" , B , "you get" , X
43 PRINT "If you subtract" , A , "minus" , B , "you get" , Y
50 LET Z = A * B !simple math (multiplication)
70 PRINT
71 PRINT "IF YOU MULTIPLY" , A , "AND" , B , "YOUR RESULT IS" , Z
75 LET J = A / B !simple math (division)
76 ON ERROR GOTO 100 !This statement allows the program to
    continue
78 PRINT "IF YOU DIVIDE" , A , "BY" , B , "YOU GET" , J
79 IF J = 0 THEN
80 GOSUB 400
82 GOTO 200
100 PRINT "THERE HAS BEEN A DIVIDE BY ZERO ERROR"
121 PRINT "THE SECOND NUMBER YOU ENTERED WAS A ZERO"
122 PRINT "THE ERROR WAS CAUSED BY TRYING TO DIVIDE" , A , "BY" , B
200 PRINT
201 PRINT " EXPONENTIATION USES THE ^ SYMBOL, X^2 WOULD BE X
    SQUARED"
202 LET G = A ^ B !exponentiation
205 PRINT A , "^" , B , "RESULTS IN" , G
300 END
400 PRINT
```



```
401 PRINT "YOUR RESULT WHEN YOU DIVIDED" , A , "BY" , B , "WAS ZERO"  
405 PRINT "REMEMBER WHEN USING DIVISION THE NUMBER WILL ALWAYS BE  
    ROUNDED DOWN"  
407 RETURN
```

➔ **Example 6 • ZBI example using E-mail server:**

```
10 CLOSE #1
20 CLOSE #2
30 OPEN #1: NAME "EML" ! PORT SET FOR EML FOR EMAIL SERVER
31 OPEN #2: NAME "ZPL"
32 PRINT "ENTER YOUR FIRST NAME"
35 INPUT A$
36 PRINT "ENTER YOUR LAST NAME"
37 INPUT B$
40 PRINT "ENTER YOUR EMPLOYEE NUMBER"
42 INPUT C$
45 PRINT "Hello ";A$; " We are now ready to proceed"
400 PRINT #1 : "JDOE@ZEBRA.COM";CHR$(4); !SENDING MESSAGE TO JDOE
405 PRINT #1 : "FROM: PRINTER #1" ! MAIL FROM: DATA ("PRINTER #1")
410 PRINT #1 : "TO: MAIN LOG SERVER (EMAIL)" ! MAIL TO: DATA ("MAIN
LOG SERVER")
500 PRINT #1 : "SUBJECT: EMPLOYEE "; B$ ;" ( ";C$;" ) LOGGED IN"!
SUBJECT : DATA
505 PRINT #1 : ""
510 PRINT #1 : "EMPLOYEE "; B$; " LOGGED IN" ! MAIL MESSAGE
550 PRINT #1 : CHR$(4) ! IMPORTANT TO END WITH EOT
600 CLOSE #1
700 CLOSE #2
```

In this program an e-mail message will be sent to  
JDOE@ZEBRA.COM <mailto:JDOE@ZEBRA.COM> that indicates that an  
employee has logged in

➔ **Example 7 • ZBI using TCP port:**

```

10 CLOSE #1
20 CLOSE #2
30 OPEN #1: NAME "TCP"
31 OPEN #2: NAME "ZPL"
32 PRINT "ENTER THE PRODUCT NAME"
35 INPUT A$
36 PRINT "ENTER QUANTITY"
37 INPUT B
40 PRINT "ENTER THE LOCATION AISLE/SHELF # EXAMPLE 3/4 "
42 INPUT C$
45 PRINT #2 : "^XA^FO100,100^A0N,50,50^FD";A$;"^FS"; !SENDING ZPL
    LOCALLY
46 PRINT #2 : "^FO100,200^A0N,50,50^FD";B;"^XZ"
400 PRINT #1 : "10.3.50.79 6101";CHR$(4); !OPENING TCP PORT
405 PRINT #1 : "! 0 200 200 400 1" !CPCL STARTS
410 PRINT #1 : "JOURNAL"
415 PRINT #1 : "PAGE-WIDTH 400"
420 PRINT #1 : "T 4 0 100 20 AISLE/SHELF"
500 PRINT #1 : "T 4 0 100 100 ";C$
501 PRINT #1 : "T 4 0 100 150 QUANTITY"
505 PRINT #1 : "T 4 0 100 200 ";B
510 PRINT #1 : "T 4 0 100 250 DESCRIPTION"
520 PRINT #1 : "T 4 0 100 300 ";A$
530 PRINT #1 : "PRINT" !CPCL ENDS WITH PRINT
550 PRINT #1 : CHR$(4);
600 CLOSE #1
700 CLOSE #2

```

User enters product name, quantity , aisle/shelf location and price. Local label prints description (product name) quantity and price, while the remote printer IP address 10.3.50.79 using TCP port 6101 (mobile printer using CPCL) prints the product description, quantity and the product location (aisle/shelf).

➔ **Example 8** • ZBI using UDP port. In this example Item,Quantity,Price PRINTER# are being sent from the printer. Log generates automatically the Month, Date, Time and IP address.

Text file from server below:

---

<b>Jun</b>	30	13:59:45	10.3.9.195	4	45	1.99	PRINTER5
<b>Jun</b>	30	14:00:01	10.3.9.195	3	23	8.69	PRINTER5
<b>Jun</b>	30	14:00:14	10.3.9.195	40	3	1.99	PRINTER5
<b>Jun</b>	30	14:00:24	10.3.9.195	5	65	5.99	PRINTER5

---

```
10 CLOSE #1
20 CLOSE #2
30 OPEN #1: NAME "ZPL"
40 OPEN #2: NAME "UDP"
45 LET C$ = "PRINTER5"
50 PRINT "ENTER PRODUCT NUMBER 1-45"
55 INPUT A
60 PRINT "ENTER QUANTITY"
65 INPUT B
70 PRINT "ENTER $ AMOUNT EXAMPLE 6.99"
75 INPUT A$
80 PRINT #2 : "10.3.50.79 514";CHR$(4);
90 PRINT #2 : A ;" ";B;" ";A$;" ";" " ";C$;
91 REM The extra spaces in above line are for
92 REM columns in ACCESS database where no data is sent or
   needed from the printer
95 PRINT #2: CHR$(4);
100 CLOSE #1
105 CLOSE #2
```

# INDEX

## A

- abort download graphic 142
- about this document *xix*
- advanced counter
  - reset 293
- alphanumeric default font
  - change 115
- applicator reprint 290
- auxiliary port
  - set 222, 225

## B

- backfeed sequence
  - change 236, 238
- bar code field default 108
- battery
  - set condition 219
- battery status 189
- bitmap font
  - download 132
- box 178

## C

- cache on 123
- cancel all 213
- cancel format 233
- caret
  - change 112, 113
- change alphanumeric default font 115
- change backfeed sequence 236, 238
- change caret 112, 113

- change delimiter 114
- change international font 117
- change memory letter designation 121
- change network settings 277
- change tilde 126
- circle 180
- CODABLOCK 53
  - considerations for ^FD character set 56
  - considerations for the ^BY 55
- code 11 20
- code 128
  - conditions with UCC 60
  - subsets 60
  - subsets A and C 62
  - subsets a, b, and c 57
- code 39 24
- code 49 29
  - automatic mode 33
  - field data character set 33
- code 93 48
  - full ASCII mode 50
- code validation 127
- comment 177
- communications diagnostics 217
  - enable 217
- configuration
  - update 242
- configuration label
  - print 325
- contacts
  - mailing address *xviii*
  - Web address *xviii*
- current partially input format
  - cancel 244
- currently connected printer
  - set transparent 278

## D

- darkness
  - set 297

data matrix 103  
date for real time clock  
    set 316  
define language 248  
define password 251  
define printer name 249  
delete object 205  
delimiter  
    change 114  
description information  
    display 203  
diagnostics  
    disable 218  
diagonal line 181  
directory label  
    print 327  
disable diagnostics 218  
discharge mode  
    battery 246  
display description information 203  
Document Conventions xx  
document conventions xx  
    command line xx  
    hot links xx  
download bitmap font 132  
download encoding 135  
download format 137  
download graphic  
    abort 142  
download graphics 139, 148  
download scalable font 143  
download true type font 145  
download unbounded true type font 146

## **E**

EAN-13 68  
EAN-8 43  
ellipse 183  
encoding  
    download 135  
    select 298  
end format 335

erase download graphics 151  
erase stored formats 150

## **F**

feedback  
    suppress 330  
field  
    field reverse 168  
    orientation 176  
    parameter 166  
    separator 169  
    typeset 170  
    variable 174  
field block 151, 152  
field clock  
    real time clock 155  
field data 157  
field hexadecimal indicator 158  
field number 164  
field orientation 176  
field origin 165  
field parameter 166  
field reverse print 168  
field separator 169  
field typeset 170  
field variable 174  
flash memory  
    initialize 214  
font identifier 130  
font name  
    to call font 17  
fonts  
    p-v 16  
format  
    cancel 233  
    download 137  
    end 335  
    pause 233  
    recall 331  
    set 329  
formats  
    erase stored 150

**G**

## graphic

- box 178
- circle 180
- diagonal line 181
- ellipse 183
- field 184
- recall 333
- symbol 187

## graphic field 184

## graphics

- download 139, 148
- erase download 151
- upload 202

## graphing sensor calibration 221

**H**

## head test

- fatal 231
- interval 240
- non-fatal 232

## head test fatal 231

## head test interval 240

## head test non-fatal 232

## host

- directory list 200
- graphic 192
- identification 194
- RAM status 195
- status return 196

## host directory list 200

## host graphic 192

## host identification 194

## host RAM status 195

## host status return 196

**I**

## image

- load 207
- move 209
- save 211

## image load 207

## image move 209

## image save 211

## industrial 2 of 5 73

## initialize Flash memory 214

## interleaved

- 2 of 5 22

## international font

- change 117

**K**

## kill battery 246

**L**

## label

- maximum length 264
- reverse print 256
- shift 258
- top 259

## label home 252

## label length 254

- set 228

## language

- define 248

## LOGMARS 79

**M**

## map clear 260

## maximum label length 264

## media

- darkness 261
- feed 263
- tracking 267
- type 270

## media darkness 261

## media sensor

- set 314

## media sensor calibration 216

- set 216

## media tracking 267

- media type 270
- memory letter designation
  - change 121
- mirror image
  - printing 281
- mode protection 268
- modify head warning 273
- MSI 81
- multiple field origin locations 160

## N

- network
  - change settings 277
  - connect 274
  - ID number 275
- network connect 274
- network ID number 275
- network printers
  - set all transparent 276

## O

- object delete 205
- offset for real time clock
  - set 307
- optional memory
  - reset 215

## P

- password
  - define 251
- pause
  - programmable 284
- pause format 233
- PDF417 37
- consideration for ^FD 41
- POSTNET 110
- power on
  - reset 235
- print
  - start 291

- width 292
- print mode 265
- print orientation 282
- print quantity 285
- print rate 287
- print start 291
- print width 292
- printer
  - sleep 336
- printer name
  - define 249
- printer sleep 336
- printhead resistance
  - set 313
- printing mirror image of label 281
- programmable pause 284

## Q

- QR code
  - normal mode 89
- quantity
  - print 285

## R

- real time clock
  - set language 302
  - set mode 302
- real time clock date format
  - select 247
- real time clock time format
  - select 247
- recall format 331
- recall graphic 333
- related documents *xxi*
- reprint
  - after error 245
  - applicator 290
- reset
  - power on 235
- reset advanced counter 293
- reset optional memory 215
- ribbon tension



set 243

## S

scalable font 13  
  download 143  
select encoding 298  
sensor calibration  
  graphing 221  
serial communications  
  set 295  
serialization data 304  
serialization field  
  standard ^FD string 299  
set all network printers transparent 276  
set auxiliary port 225  
set battery condition 219  
set darkness 297  
set dots  
  millimeter 229  
set dots per millimeter 229  
set label length 228  
set serial communications 295  
set units of measurements 271  
slew  
  home position 280  
slew given number  
  dot rows 279  
slew to home position 280  
start print 308  
start ZBI 222  
symbol 187, 191

## T

tear-off adjust position 321  
terminate ZBI 234  
tilde  
  change 126  
time for real time clock

set 316  
transfer object 322  
true type font  
  download 145

## U

unbounded true type font  
  download 146  
units of measurement  
  set 271  
UPC/EAN extensions 95  
UPC-A 101  
UPC-E 45  
update configuration 242  
upload graphics 202  
UPS maxicode 64  
  considerations for ^FD 65  
use font name to call font 17

## W

width  
  print 292

## Z

ZBI  
  start 222  
  terminate 234  
ZebraNet Alert  
  halt 310  
  set 317  
ZPL  
  set 320  
ZPL commands  
  ^B7 37  
ZPL II programming commands 3







**Zebra Technologies Corporation**

333 Corporate Woods Parkway  
Vernon Hills, Illinois 60061.3109 U. S. A.  
Telephone +1 847.634.6700  
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

