

PRELIMINARY

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 propriety 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.

FCC Compliance Statement

Digital apparatus has been tested and found to comply with the limits for a Class B Digital Device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the product manuals, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

The user is cautioned that any changes or modifications not expressly approved by Zebra Technologies could void the user's authority to operate the equipment. This unit was tested with shielded cables on the peripheral devices. Shielded cables must be used with the unit to insure compliance.

WARNING: EXPOSURE TO RADIO FREQUENCY RADIATION ON SOME MODELS. TO CONFORM TO FCC RF EXPOSURE REQUIREMENTS THIS DEVICE SHALL BE USED IN ACCORDANCE WITH THE OPERATING CONDITIONS AND INSTRUCTIONS LISTED IN THE UNIT'S USER'S MANUAL.

Canadian DOC Compliance Statement

Digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus as set out in the radio interference regulations of the Canadian Department of Communications.

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. 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.

© 2005 ZIH Corp. All trademarks and registered trademarks are property of their respective owners. All rights reserved.

TABLE OF CONTENTS

PROGRAMMING INTRODUCTION	P1-1
PROGRAMMING LANGUAGE EMULATION	P1-1
PROGRAMMING LANGUAGE RECOMMENDATIONS	P1-2
GETTING PRINTER INFORMATION	P1-3
GETTING PRINTER INFORMATION EXAMPLE	P1-4
COMMUNICATIONS DIAGNOSTICS MODE	P1-7
LABEL COORDINATE SYSTEM	P1-8
LABEL VISTA™	P1-9
PRINTER COMMANDS	P2-1
PRINTER COMMANDS	P2-2
PRINT COMMAND	P2-3
FORM COMMAND	P2-4
JOURNAL COMMAND	P2-5
UNITS COMMANDS	P2-6
USING COMMENTS	P2-8
TEXT	P3-1
RESIDENT FONT SAMPLES	P3-1
TEXT COMMANDS	P3-4
USING FONT GROUPS	P3-6
TEXT CONCATENATION COMMANDS	P3-8
MULTILINE (ML) COMMANDS	P3-10
COUNT COMMAND	P3-12
SETMAG COMMAND	P3-14
SCALABLE TEXT	P4-1
SCALE-TEXT COMMANDS	P4-2
SCALE-TO-FIT COMMANDS	P4-4
SCALABLE CONCATENATION COMMANDS	P4-6
ROTATE COMMANDS	P4-8
LINEAR BAR CODES	P5-1
INTRODUCTION	P5-1
UPC AND EAN/JAN BAR CODES	P5-3
CODE 39 OR CODE 3 OF 9 BAR CODES	P5-6
CODE 93 OR CODE 9 OF 3 BAR CODES	P5-8
INTERLEAVED 2 OF 5 BAR CODES	P5-8
CODE 128 AND THE UCC-128 SHIPPING STANDARD	P5-9
CODABAR	P5-11
MSI PLESSEY BAR CODES	P5-12
POSTNET AND FACING IDENTIFICATION MARKS	P5-13
BAR CODE COMMANDS	P5-15
BARCODE COMMAND	P5-16
BARCODE-TEXT COMMAND	P5-18

TABLE OF CONTENTS (continued)

COUNT COMMAND	P5-20
TWO-DIMENSIONAL BAR CODES	P6-1
INTRODUCTION	P6-1
PDF417	P6-2
MAXICODE	P6-2
TWO-DIMENSIONAL BAR CODE COMMANDS	P6-3
PDF417 (PORTABLE DATA FILE)	P6-4
GRAPHICS	P7-1
BOX COMMAND	P7-1
LINE COMMANDS	P7-2
INVERSE-LINE COMMANDS	P7-4
PATTERN COMMAND	P7-6
GRAPHICS COMMANDS	P7-8
PCX COMMANDS	P7-10
ADVANCED COMMANDS	P8-1
CONTRAST COMMAND	P8-1
TONE COMMAND	P8-2
JUSTIFICATION COMMANDS	P8-3
PAGE-WIDTH COMMAND	P8-4
PACE COMMAND	P8-6
NO-PACE COMMAND	P8-9
WAIT COMMAND	P8-10
REWIND COMMAND	P8-11
TENSION COMMANDS	P8-12
SPEED COMMAND	P8-13
SETSP COMMAND	P8-14
ON-OUT-OF-PAPER COMMAND	P8-15
ON-FEED COMMAND	P8-16
PREFEED COMMAND	P8-18
POSTFEED COMMAND	P8-19
PRESENT-AT COMMAND	P8-20
COUNTRY / CODE PAGE COMMAND	P8-22
USING FORMAT FILES	P8-24
BEEP COMMAND	P8-27
CUT COMMAND	P8-28
PARTIAL-CUT COMMAND	P8-29
CUT-AT COMMAND	P8-30
MCR COMMANDS	P8-31
LINE PRINT MODE	P9-1
INTRODUCTION	P9-1
SPECIAL COMMANDS USING THE UTILITY FUNCTION	P9-2
UNITS COMMANDS	P9-3
SETLP COMMAND	P9-4

SETLF COMMAND	P9-5
MOVING WITH X AND Y COORDINATES	P9-6
LMARGIN COMMAND	P9-7
SETBOLD COMMAND	P9-8
SETSP COMMAND	P9-9
PAGE-WIDTH COMMAND & PAGE-HEIGHT COMMAND	P9-10
SPECIAL ASCII CHARACTERS	P9-10
SETFF COMMAND	P9-11
SET-TOF COMMAND	P9-12
TEARING OR CUTTING THE PAPER	P9-14
BAR CODES, GRAPHICS AND LINES	P9-15
SETLP-TIMEOUT COMMAND	P9-16
DESIGNING A RECEIPT	P9-16

ADVANCED UTILITIES P10-1

VERSION UTILITY	P10-2
CHECKSUM UTILITY	P10-3
DEL UTILITY	P10-4
DIR UTILITY	P10-5
DEFINE-FILE (DF) UTILITY	P10-6
TYPE UTILITY	P10-8
BAUD UTILITY	P10-9
COUNTRY / CODE-PAGE UTILITY OR	P10-10
CHAR-SET/CODE PAGE UTILITY	P10-10
ANNOUNCE UTILITY	P10-12
TIMEOUT UTILITY	P10-14
TIMEOUT UTILITY EXAMPLE	P10-14
BEEP COMMAND	P10-15
BEEP COMMAND EXAMPLE	P10-15
ON-LOW-BATTERY COMMAND	P10-16
LT COMMAND	P10-17
LT COMMAND EXAMPLES	P10-17
SET-TIME UTILITY	P10-18
GET-TIME UTILITY	P10-19
SET-DATE UTILITY	P10-20
GET-DATE UTILITY	P10-21
PRINTING A TIME STAMP	P10-22
PRINTING A DATE STAMP	P10-23
PAPER-JAM UTILITY	P10-24
MAGNETIC CARD READER (MCR) COMMAND	P10-25
MCR-QUERY COMMAND	P10-31
MCR-CAN COMMAND	P10-33
S-CARD COMMAND	P10-34
S-CARD COMMAND EXAMPLES	P10-37

DENSO BHT COMMANDS P10-38

SETTING THE DATA FORMAT	P10-38
-------------------------------	--------

TABLE OF CONTENTS (continued)

IR OR CABLE INTERFACE SELECTION	P10-40
BHT-BAUD COMMAND	P10-40
SETTING THE MODE FOR IR DATA TRANSMISSION	P10-40
CONFIGURING THE BHT FOR IR TRANSMISSION	P10-41
BHT-IR FILE TRANSMISSION	P10-42
CONFIGURING THE BHT FOR CABLE TRANSMISSION	P10-43
LOADING A .DAT FILE INTO THE BHT	P10-44
INTERRUPTED TRANSMISSIONS	P10-45
EXAMPLE .DAT FILE	P10-46
SRF-ACCESS AND END-SRF-ACCESS COMMANDS	P10-47
BHT-BAUD COMMAND	P10-48
BHT MODE COMMANDS	P10-49
BHT PROTOCOL COMMAND	P10-53
PRINTER ESCAPE COMMANDS	P11-1
SET AND READ CODE COMMAND	P11-1
PRINTER ESCAPE COMMANDS FORMAT	P11-2
STATUS/INFORMATION	P11-3
USER LABEL COUNT	P11-5
POWER OFF COMMAND	P11-5
WIRELESS NETWORK PRINTERS	P12-1
INTRODUCTION	P12-1
NETWORK PRINTER SAFETY CONSIDERATIONS	P12-1
LAN COMMAND	P12-1
SETTING THE IP ADDRESS FOR NETWORK PRINTERS	P12-7
LAN COMMAND RESPONSE	P12-8
NETWORK PRINTER TROUBLESHOOTING	P12-9
WIRELESS LAN REPORT EXAMPLE	P12-10
PROGRAMMING FOR THE DISPLAY OPTION	P13-1
INTRODUCTION:	P13-1
EXAMPLE 1:	P13-1
EXAMPLE 2:	P13-5
CONFIGURATION/CONTROL COMMANDS	P14-1
INTRODUCTION	P14-1
COMMAND FORMAT	P14-1
COMMANDS / PARAMETERS	P14-3
BLUETOOTH® PARAMETERS	P14-5
COMM PORT PARAMETERS	P14-11
DEVICE PARAMETERS	P14-13
DISPLAY PARAMETERS	P14-15
FILE PARAMETERS	P14-17
PRINTER MECHANISM PARAMETERS	P14-19
INPUT PARAMETER	P14-22

MEDIA PARAMETERS	P14-24
MEMORY PARAMETERS	P14-27
ODOMETER PARAMETERS	P14-28
POWER PARAMETERS	P14-32
TEST FUNCTION PARAMETERS	P14-36
NETWORKING PARAMETERS	P14-37
USB PARAMETERS	P14-59
WLAN PARAMETERS	P14-63
WLAN.XXX FHSS COMPATIBILITY	P14-63
ZEBRA PRINTER MIRROR PROCESS	P14-86
PRINTER CONFIGURATION AND SETUP	P15-1
USING LABEL VISTA FOR PRINTER CONFIGURATION	P15-1
USING LABEL VISTA FOR WIRELESS CONFIGURATION	P15-3
POWER MANAGEMENT	P15-4
BATCH FILES	P15-7
RUN.BAT COMMAND/FILE	P15-8
RE-RUN COMMAND	P15-9
GAP-SENSE & BAR-SENSE COMMANDS	P15-10
INDEX	A-1
APPENDIX A-FREQUENTLY ASKED QUESTIONS .	A-14
APPENDIX B- INTERFACE CABLES	A-19
APPENDIX C- CHARACTER TABLES	A-24
APPENDIX D - FONT INFORMATION	A-28
APPENDIX E- BAR CODE QUICK REFERENCE ...	A-32
APPENDIX F - PRODUCT SUPPORT	A-34

Programming Section

PROGRAMMING INTRODUCTION

This manual details the various commands in the CPCL language which allow the programmer to utilize the built in text, graphics, bar code printing and communications capabilities of Zebra mobile printers. The following notation conventions are used throughout this manual::

- `{ }` Required item
- `[]` Optional item
- `()` Abbreviated command
- `< >` Literal item

A space character is used to delimit each field in a command line.

Many commands are accompanied by examples of the command in use. After the word 'Input' in each example, the file name of that example will appear in parentheses.

This manual can also be found on the Accessories CD (Part Number 46984) available from Zebra.

PROGRAMMING LANGUAGE EMULATION

Zebra Mobile Printers can emulate the EPL2™ and ZPL® programming languages used by other types of Zebra printers. Printers using emulation must be configured with more memory and have a special emulation program loaded. For more information on the appropriate uses of these languages, refer to the following language comparison chart:

continued on next page

PROGRAMMING LANGUAGE RECOMMENDATIONS

Language	Native in	Recommended Use
CPCL	Cameo®, Encore® QL, RW and older Comtec® models	<ul style="list-style-type: none"> • In new installations of mobiles, where CPCL is easily integrated into the host application • When older Comtec models are being upgraded- so that the customer can use existing application without code modifications
ZPL (emulation available on QL series, std. on QL plus & RW Series)	PA/PT Series mobile printers, Zebra High Performance/ Industrial /Commercial printers, R-140 RFID printer, LP/TLP 2844-Z and PAX applicator printers	<ul style="list-style-type: none"> • When an installation already uses ZPL as a standard language & needs to maintain a consistent language for all thermal printers. • When certain printer functions are not available in CPCL or EPL, such as: (ZBI, Datamatrix, Code 11, Micro PDF) • When replacing a PA/PT 40x, or another Zebra printer using ZPL, with a Zebra mobile printer
EPL (emulation available on QL, Encore, or Cameo series and std. on RW & QL plus series)	Zebra Desktops, R402 (RFID printer), TR 220, PS 21xx Series	<ul style="list-style-type: none"> • When an installation already uses EPL- the most commonly used Zebra language- as a standard language and needs to maintain a consistent language for all thermal printers. • If you are replacing Eltron Transport or Xport mobile printers, a Zebra mobile with an EPL emulation will ease the transition.

Note: QL Plus and RW Series printers have EPL and ZPL emulation built into their operating system. See the "device.languages" command under the "Device Parameters" topic in Section 14 for more information on setting programming languages with these products.

GETTING PRINTER INFORMATION

The printer can produce a report containing information about the application resident in printer memory. A report similar to the example printouts shown on the following pages can be obtained from your printer by doing the following:

1. Turn the printer OFF.
2. While holding the FEED key down, turn the printer ON.
3. When printing begins, release the FEED key.

The printer prints a line of interlocking “x” characters and then produces two reports. The first report indicates the printer model, ROM version, serial number, baud rate, etc.

The second report contains application information. The last digits in the application number indicate the software version.(e.g. “Software: HTLK**40**d” indicates a software version of 40.) If no second report appears, there is no application loaded.

The Wireless Communications report will appear if a Short Range Radio (SRRF), infrared (IrDA) or wireless LAN (RF LAN) option has been installed in the printer. If no wireless options are installed, the Wireless Communications Section will consist of a blank line.

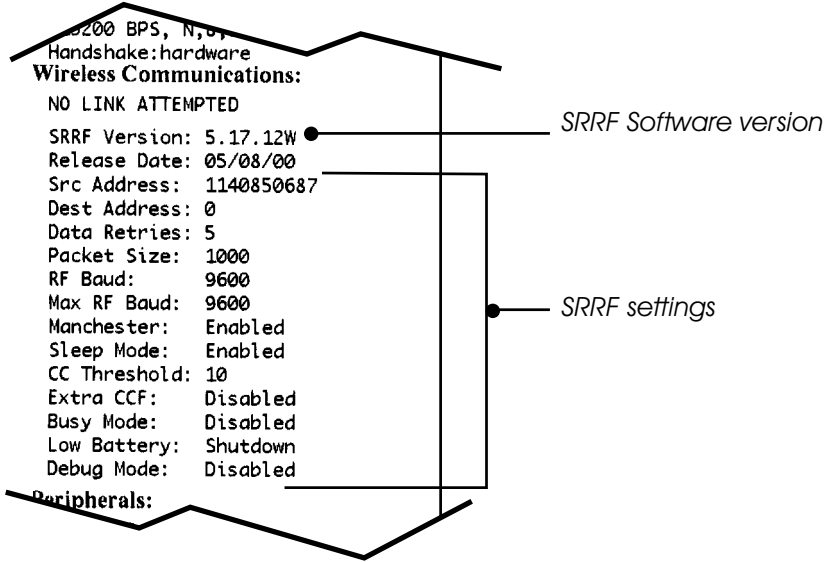
The RF LAN Information section will only appear on Network Printers (units equipped with a WLAN card). Network printers are covered in detail in Section 12 of this manual.

The Label section shown in the second report reports the maximum size label that can be printed, based on a printer resolution of 203 dots/inch (8 dots/mm).

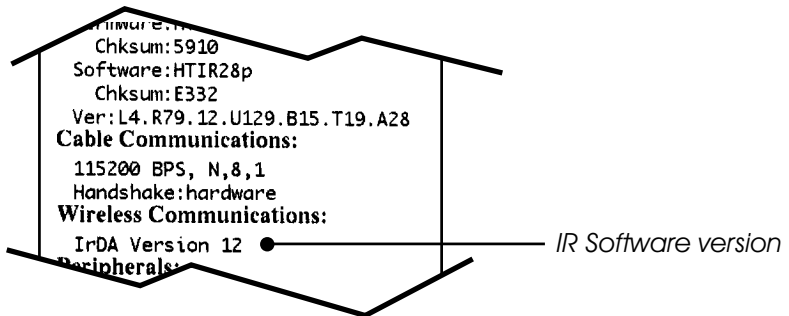
In the example, the Label Height is 2224 dots, which means for a label width of 576 dots (2.8 inches or 72 mm), you can print labels up to 10.9 inches (278 mm) long. Reducing the label width results in a corresponding increase in the maximum label length.

GETTING PRINTER INFORMATION EXAMPLE, CONT.

*Wireless Communications Section with Short Range Radio (SRRF)
Wireless Option Installed*

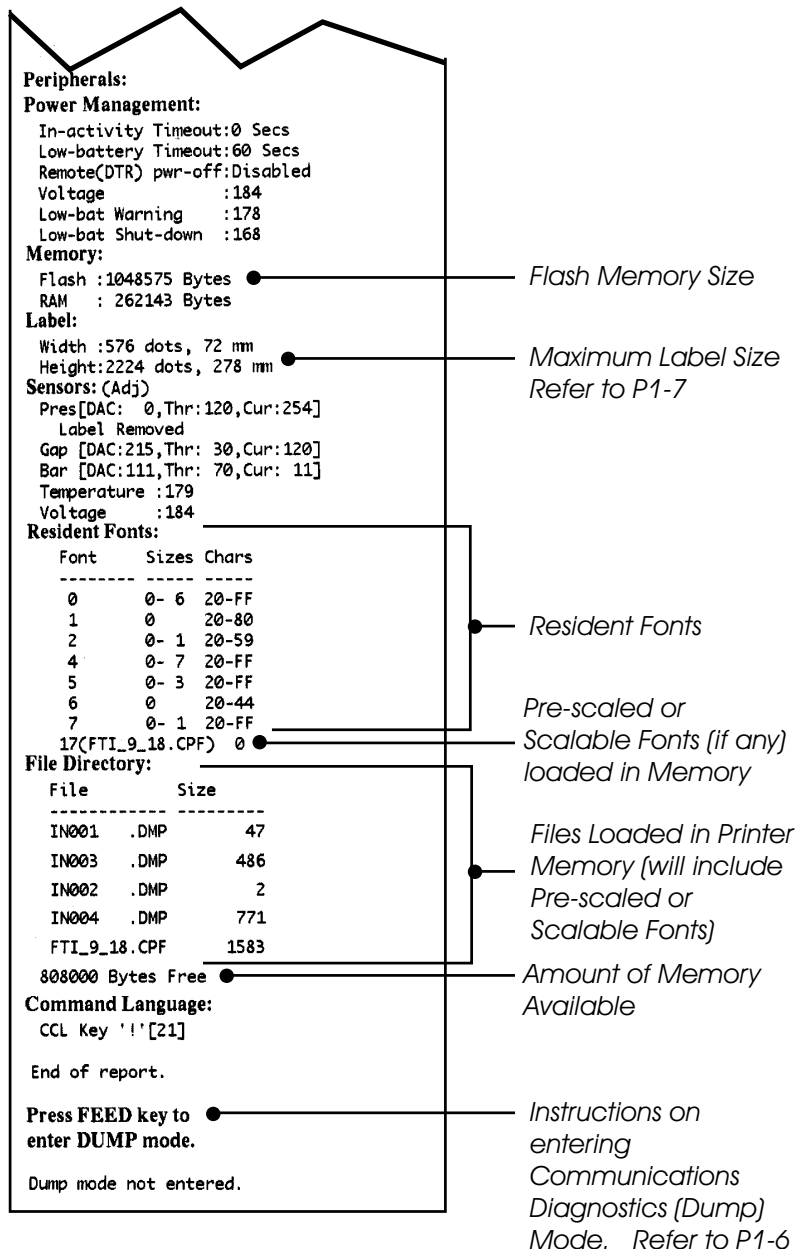


Wireless Communications Section with IR Wireless Option Installed



GETTING PRINTER INFORMATION EXAMPLE, CONT.

Continuation of Second Report



COMMUNICATIONS DIAGNOSTICS MODE

To aid the user in diagnosing communications problems, the printer features a Communications Diagnostics Mode (Dump Mode). In the dump mode, the printer will print the ASCII hex codes of the data sent to it, and their text representation (or the period '.', if not a printable character). As a test of the printer the "ALL CHR.S.LBL" file on the MPU disk may be sent.

To enter Communications Diagnostics Mode:

1. Turn the printer OFF.
2. Hold FEED key down.
3. Turn the printer ON.
4. Release FEED key when printer starts printing the diagnostics.
5. At the end of 2nd diagnostics report, the printer will print: "Press FEED key to enter DUMP mode".
6. Now press the FEED key. The printer will print: "Entering DUMP mode".

Note: If the FEED key is not pressed within 3 seconds, the printer will print "DUMP mode not entered" and will resume normal operation.

7. At this point, the printer is in DUMP mode and will print the ASCII hex codes of the data sent to it, and their text representation (or "." if not a printable character).

Additionally, a file with a ".dmp" extension containing the ASCII information will be created and stored in the printer's memory. It can be viewed, "cloned" or deleted using the Label Vista application. (Refer to pg. P1-8 and the Label Vista documentation for more information.)

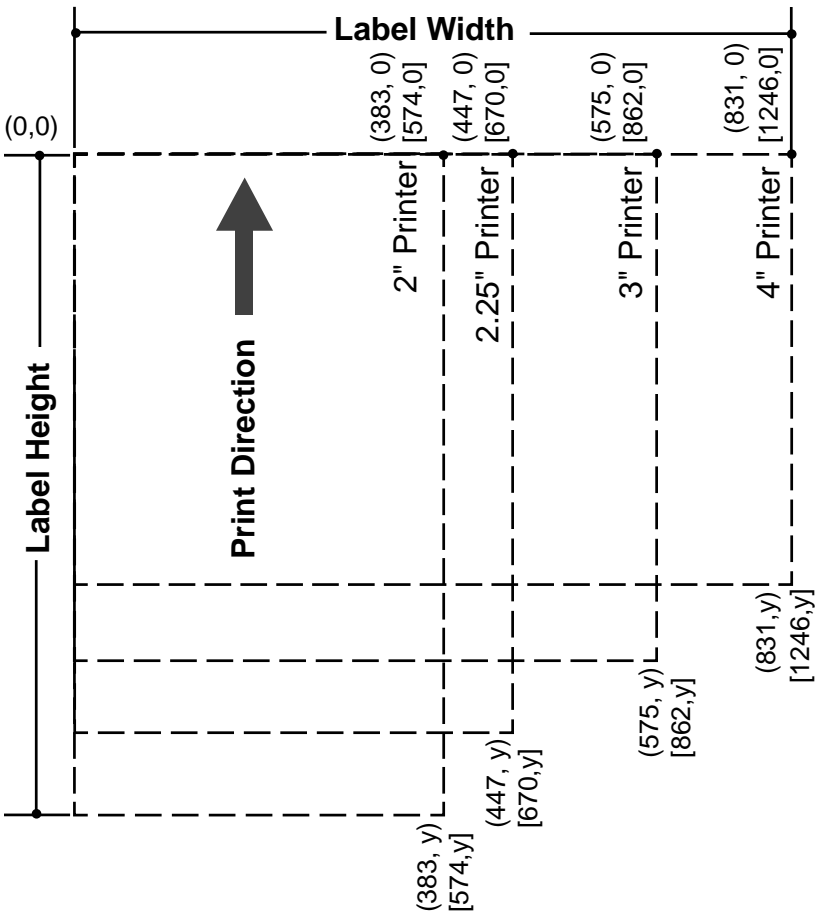
To cancel Communications Diagnostics Mode:

1. Turn the printer OFF.
2. Wait 5 seconds.
3. Turn the printer ON.

LABEL COORDINATE SYSTEM

The x and y coordinates are expressed here in terms of dots. Coordinates in () are for 200 dot per inch printers. On 200 d.p.i. printers, 8 dots (either horizontally or vertically) equal 1 millimeter and 203 dots equate approximately to 1 inch.

Coordinates in [] are for 300 dot per inch printers. On 300 d.p.i. printers 12 dots equal 1 millimeter, and 305 dots equate approximately to 1 inch.



Notes:

1. Coordinates refer to the actual printing area of the printers.
2. "y" = the available label height which can vary with the resident application. (See Getting Printer Information, Page P1-3.)

LABEL VISTA™

Label Vista is a stand-alone program running in Windows® 95/NT that allows users with little or no programming background to design labels which can be printed on certain model Zebra portable printers. It combines an intuitive graphically based user environment with powerful, but easily mastered, editing tools.

Label Vista allows the creation of printable, fixed-size (pre-scaled) fonts derived from an included library of TrueType™ fonts, which greatly enhances the versatility of this program.

In addition, Label Vista allows the easy creation of format files which can remain resident in the printer and be merged with variable data files sent from the host. This provides a very efficient method of printing labels that have a mixture of data fields that change from label to label and elements that remain constant. Refer to Section 8 of this Manual for more information on format files.

Label Vista also provides a powerful set of diagnostics tools. It is recommended that the Label Vista documentation package be consulted for a more detailed description of the printer diagnostics available in this program.

Label Vista utilizes a subset of the full Mobile Printing Systems Programming Language described in this manual. Files created in Label Vista are fully compatible with any other label files created using the complete set of Mobile Printer commands.

Label Vista requires an IBM compatible personal computer, running Windows 95 or later. A PC with the minimum configuration to run Windows 95 will have sufficient memory to run Label Vista.

Note: Label Vista has proven to be compatible with Windows XP in informal testing, however, compatibility problems with certain unusual combinations of hardware and software may arise.

PRINTER COMMANDS

A label file always begins with the “!” character followed by an “x” offset parameter, “x” and “y” axis resolutions, a label length and finally a quantity of labels to print. The line containing these parameters is referred to as the Command Start Line.

A label file always begins with the Command Start Line and ends with the “PRINT” command. The commands that build specific labels are placed between these two commands.

A space character is used to delimit each field in a command line.

Note: Every line in the command session must be terminated with both carriage-return and line-feed characters. All Printer Commands must be in uppercase characters ONLY.

PRINTER Commands

Format:

`<!/ > {offset} <200> <200> {height} {qty}`

where:

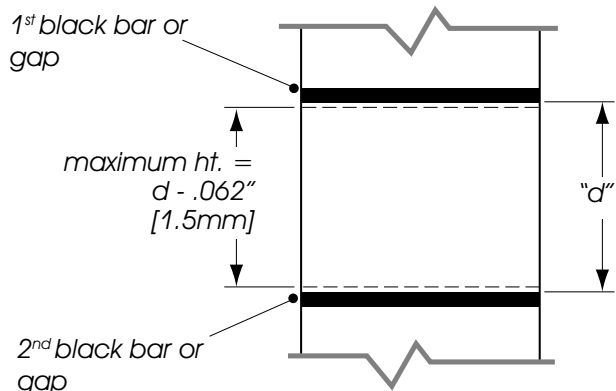
`<!/ >`: Use '!' to begin a control session.

`{offset}`: The horizontal offset for the entire label. This value causes all fields to be offset horizontally by the specified number of UNITS.

`<200>`: Horizontal resolution (in dots-per-inch).

`<200>`: Vertical resolution (in dots-per-inch).

`{height}`: The maximum height of the label. The maximum label height is calculated by measuring from the bottom of the first black bar (or label gap) to the top of the next black bar (or label gap). Then 1/16" [1.5mm] is subtracted from this distance to obtain the maximum height. (In dots: subtract 12 dots on 203 d.p.i printers; 18 dots on 306 d.p.i. printers)



`{qty}`: Quantity of labels to be printed.
Maximum = 1024.

PRINTER Command Example

Input (HELLO.LBL):

```
! 0 200 200 210 1
TEXT 4 0 30 40 Hello World
FORM
PRINT
```

Output:

Hello World

PRINT Command

The PRINT command terminates and prints the file. This must always be the last command (except when in Line Print Mode). Upon execution of the PRINT command, the printer will exit from a control session. Be sure to terminate this and all commands with both carriage-return and line-feed characters.

Format:

{command}

where:

{command}: PRINT

FORM Command

The FORM command instructs the printer to feed to top of form after printing.

Format:

{command}

where:

{command}: FORM

In the following example, the printer will execute a form feed after the label is printed. See the SETFF (set form feed) command in the section on designing receipts and lists for information on setting printer behavior when the FORM command is executed.

```
Input: FORM.LBL

! 0 200 200 3 1
IN-CENTIMETERS
CENTER
TEXT 4 1 0 .5 Form Command
FORM
PRINT
```

JOURNAL Command

By default, the printer will check for correct media alignment if it encounters the eye-sense mark (black horizontal bars on back of media) during a print cycle (LABEL mode). If necessary, the JOURNAL command can be used to disable this automatic correction feature. The user's program is responsible for checking and assuring presence of paper in JOURNAL mode. Please refer to the status inquiry command for details on checking for out-of-paper condition.

Format:

{command}

where:

{command}: JOURNAL

UNITS Commands

The units commands are used to specify a measurement system for all subsequent command fields in a control session. Coordinates, widths, and heights for all control commands can be entered with precision to four decimal places. By placing a units command immediately after the first line in a control session, the specified measurement system will also apply to the offset and height fields. The printer measurement system will default to dots until a units command is issued.

Format:

{command}

where:

{command}: Choose from the following:

IN-INCHES Measurement in inches.

IN-CENTIMETERS Measurement in centimeters.

IN-MILLIMETERS Measurement in millimeters.

IN-DOTS Measurement in dots.

UNITS Examples

Input 1 (UNITS1.LBL):

```
! 0.3937 200 200 1 1
IN-INCHES
T 4 0 0 0 1 cm = 0.3937"
IN-DOTS
T 4 0 0 48 1 mm = 8 dots
B 128 1 1 48 16 112 UNITS
T 4 0 48 160 UNITS
FORM
PRINT
```

Output 1:

1 cm = 0.3937"
1 mm = 8 dots



Input 2 (UNITS2.LBL):

```
! 0 200 200 2.54 1
IN-CENTIMETERS
T 4 0 1 0 1" = 2.54 cm
IN-MILLIMETERS
T 4 0 0 6 203 dots = 25.4 mm
B 128 0.125 1 6 12 14 UNITS
T 4 0 16 20 UNITS
FORM
PRINT
```

Output 2:

1" = 2.54 cm
203 dots = 25.4 mm



USING COMMENTS

Comments can be added between the first line of a command session and the “PRINT” command.

A comment is placed in the file by starting a line with the ';' character in the first column. Any remaining text to the end of the line will be ignored. Comments are illegal between the CONCAT and ENDCONCAT commands.

COMMENTS Example

Input (COMMENT.LBL):

```
! 0 200 200 25 1
IN-MILLIMETERS
JOURNAL
; Center justify text
CENTER
; Print the words 'A COMMENT'
TEXT 5 1 0 5 A COMMENT
; Print the label and go to top of next form
FORM
PRINT
```

Output:

A COMMENT

TEXT

RESIDENT FONT SAMPLES

Font#: Size:
0 0

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo
\$01.23456789¢ \$0123456789 \$0123456789 \$0123

Font#: Size:
0 5

Aa Bb Cc Dd Ee Ff Gg
\$01.23456789¢ \$0123

Font#: Size:
0 1

Aa Bb Cc Dd Ee Ff Gg
\$01.23456789¢ \$0123

Font#: Size:
0 6

Aa Bb Cc Dd
\$01.2345678¢

Font#: Size:
0 2

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo
\$01.23456789¢ \$0123456789 \$0123456789 \$0123

Font#: Size:
1 0

Aa Bb Cc Dd Ee Ff Gg
~~\$01.23456789¢~~ \$0123

Font#: Size:
0 3

Aa Bb Cc Dd Ee Ff Gg
\$01.23456789¢ \$0123

Font#: Size:
2 0

A C D
~~\$01.23456789~~

Font#: Size:
0 4

Aa Bb Cc Dd
\$01.2345678¢

Font#: Size:
2 1

A C D
\$01.23456789

RESIDENT FONT SAMPLES (continued)

Font#: Size:
4 0

Aa Bb Cc Dd Ee Ff
\$01.23456789¢

Font#: Size:
4 1

Aa Bb Cc \$123

Font#: Size:
4 2

\$120.34

Font#: Size:
4 3

\$120.34

Font#: Size:
4 4

\$120.34

Font#: Size:
4 5

\$120.34

Font#: Size:
4 6

\$120.34

Font#: Size:
4 7

\$120.34

RESIDENT FONT SAMPLES (continued)

Font#: Size:

5 0

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj

\$01.23456789¢ \$01.234567890

Font#: Size:

6 0

ı ı' ıı' ııı'

0 1 2 3 4 5 6 7 8 9

Font#: Size:

5 1

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj

\$01.23456789¢ \$01.234567890

Font#: Size:

7 0

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj

\$01.23456789¢ \$01.234567890

Font#: Size:

5 2

Aa Bb Cc Dd Ee Ff

\$01.23456789¢

Font#: Size:

7 1

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj

\$01.23456789¢ \$01.234567890

Font#: Size:

5 3

Aa Bb Cc \$1.23 10¢

TEXT Commands

The TEXT command is used to place text on a label. This command and its variants control the specific font number and size used, the location of the text on the label, and the orientation of this text. Standard resident fonts can be rotated to 90° increments as shown in the example.

Format:

{command} {font} {size} {x} {y} {data}

where:

<i>{command}</i> :	Choose from the following:
TEXT (or T)	Prints text horizontally.
VTEXT (or VT)	Prints text (vertically) rotated 90 degrees counterclockwise.
TEXT90 (or T90)	(Same as VTEXT above.)
TEXT180 (or T180)	Prints text (upside down) rotated 180 degrees counterclockwise.
TEXT270 (or T270)	Prints text (vertically) rotated 270 degrees counterclockwise.
<i>{font}</i> :	Name/number of the font.
<i>{size}</i> :	Size identifier for the font.
<i>{x}</i> :	Horizontal starting position.
<i>{y}</i> :	Vertical starting position.
<i>{data}</i> :	The text to be printed.

TEXT Example

Input (TEXT.LBL):

```
! 0 200 200 210 1
TEXT 4 0 200 100 TEXT
TEXT90 4 0 200 100 T90
TEXT180 4 0 200 100 T180
TEXT270 4 0 200 100 T270
FORM
PRINT
```

Output:

```
0811190
1111111
T180
T270
TEXT
```

USING FONT GROUPS

FONT-GROUP (FG) Command

The FG command gives a user the ability to group up to 10 pre-scaled font files into a single group. A user can later specify the font group in a TEXT command. If a font group is used in a text command, the printer will use the *largest* font specified in the font group that will produce the required text data and still remain within the available width of the label for the text. When specified in the TEXT command, the {font} parameter is specified as FG, and the {size} parameter is specified as the {fg}. Note that a user can also specify an FG command within a CONCAT/ENCONCAT command.

Format:

{command} {fg fn fs} [fn fs] ...

where:

{command}: FG

{fg}: Font group number. Up to 10 font groups can be specified. Valid font groups range from 0 to 9.

{fn}: Name/number of the font.

{fs}: Size identifier for the font.

NOTE: Up to 10 font number/font size pairs can be assigned to a font group.

FG Example

In the example, the descriptions will be printed with the largest font in the specified font group that is capable of fitting the requested text in a 250 dot label field.

Input (FG.LBL):

```
! 0 200 200 250 1
; Specify fonts 0-0, 7-0, 5-0, 4-0 as members
; of font group 3.
FG 3 0 0 7 0 5 0 4 0
VT FG 3 10 250 Ketchup
VT FG 3 70 250 Fancy Ketchup
VT FG 3 120 250 Extra Fancy Ketchup
VT FG 3 180 250 Large Size Extra Fancy Ketchup
FORM
PRINT
```

Output:

Ketchup
Fancy Ketchup
Extra Fancy Ketchup
Large Size Extra Fancy Ketchup

TEXT CONCATENATION COMMANDS

Text concatenation allows you to assign different character styles to strings, printing them with uniform spacing on the same text line. This command can be used in combination with scalable fonts. See Scalable Concatenation Commands

Format:

```
{command} {x} {y}  
{font} {size} {offset} {data}  
    "      "      "      "  
{font} {size} {offset} {data}  
<ENDCONCAT>
```

where:

{command}: Choose from the following:

CONCAT	Horizontal concatenation.
VCONCAT	Vertical concatenation.

{x}: Horizontal starting position.

{y}: Vertical starting position.

{font}: Name/number of the font.

{size}: Size identifier for the font.

{offset}: Unit-value to offset text from the starting position. Used to align individual text strings or create superscript/subscript characters.

{data}: Text to be printed.

<ENDCONCAT>: Terminates concatenation.

TEXT CONCATENATION Example

Input (CONCAT.LBL):

```
! 0 200 200 210 1
CONCAT 75 75
4 2 5 $
4 3 0 12
4 2 5 34
ENDCONCAT
FORM
PRINT
```

Output:

\$1234

MULTILINE (ML) Commands

MULTILINE (ML) allows you to print multiple lines of text using the same font and line-height.

Format:

```
{command} {height}  
{text} {font} {size} {x} {y}  
{data}  
"  
{data}  
<ENDMULTILINE>
```

where:

{command}: Choose from the following:

MULTILINE Prints multiple lines of text.
(or ML)

{height}: Unit-height for each line of text.

{text}: Text command (TEXT, VTEXT, etc.).

{font}: Name/number of the font.

{size}: Size identifier for the font.

{x}: Horizontal starting position.

{y}: Vertical starting position.

{data}: Text to be printed.

<ENDMULTILINE>: Terminates MULTILINE.
(or ENDML)

MULTILINE Example

Input (MULTIL.LBL):

```
! 0 200 200 210 1
ML 47
TEXT 4 0 10 20
1st line of text
2nd line of text
:
Nth line of text
ENDML
FORM
PRINT
```

Output:

```
1st line of text
2nd line of text
:
Nth line of text
```

COUNT Command

The COUNT command is used for printing multiple labels where a numeric text field or numeric data encoded in a bar code is to be incremented or decremented for each label. The TEXT/BARCODE command string must contain this numeric data as the last characters of the string. The numeric data portion can be up to 20 characters, and can be preceded by the '-' sign. Incrementing or decrementing the numeric data thru '0' is not allowed. Leading zeros will be retained. Up to 3 COUNT commands can be used in a label file.

The numeric data incremented/decremented is contained in the TEXT or BARCODE command that immediately preceded the COUNT command.

Format:

{command} {numeric value}

where:

{command}: COUNT

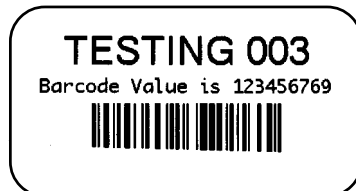
{numeric value}: Any integer value up to 20 characters. The value can be preceded by a '-' sign if decrementing of the TEXT/BARCODE value is desired. Leading zeros will be retained in the output.

COUNT Example

Input (COUNT.LBL):

```
! 0 200 200 210 3
; Print 3 labels
CENTER
TEXT 4 0 0 50 TESTING 001
COUNT 1
TEXT 7 0 0 100 Barcode Value is 123456789
COUNT -10
BARCODE 128 1 1 50 0 130 123456789
COUNT -10
FORM
PRINT
```

Output:



SETMAG Command

The SETMAG command magnifies a resident font to the magnification factor specified.

Format:

{command} {w} {h}

where:

{command}: SETMAG

{w}: Width magnification of the font.
Valid magnifications are 1 thru 16.

{h}: Height magnification of the font.
Valid magnifications are 1 thru 16.

NOTE: The SETMAG command stays in effect after printing a label. This means that the next label printed will use the most recently set SETMAG values. To cancel any SETMAG values and allow the printer to use its default font sizes, use the SETMAG command with magnifications of 0,0.

SETMAG Example

Input (SETMAG.LBL):

```
! 0 200 200 210 1
CENTER
SETMAG 1 1
TEXT 0 0 0 10 Font 0-0 at SETMAG 1 1
SETMAG 1 2
TEXT 0 0 0 40 Font 0-0 at SETMAG 1 2
SETMAG 2 1
TEXT 0 0 0 80 Font 0-0 at SETMAG 2 1
SETMAG 2 2
TEXT 0 0 0 110 Font 0-0 at SETMAG 2 2
SETMAG 2 4
TEXT 0 0 0 145 Font 0-0 at SETMAG 2 4
; Restore default font sizes
SETMAG 0 0
FORM
PRINT
```

Output:

```
Font 0-0 at SETMAG 1 1
Font 0-0 at SETMAG 1 2
Font 0-0 at SETMAG 2 1
Font 0-0 at SETMAG 2 2
Font 0-0 at SETMAG 2 4
```


SCALABLE TEXT

Scalable text allows a user to print text at any point size. Point size can be specified for both the X and Y directions to produce characters that are "stretched" in either their width or height. Point sizes specified and text produced will print at 72 points equating to one inch (25.4mm).

The printer can contain scalable font files as part of the application, or scalable font files can be downloaded to the printer using one of the utilities on the supplied disk. A scalable text file must be present in your printer's memory in order to use scalable text features.

SCALE-TEXT Commands

The SCALE-TEXT commands allow the user to specify the point size of *both* the width and height of the font.

Format:

{command} {name} {width} {height} {x} {y} {data}

where:

{command}: Choose from the following:

SCALE-TEXT: Prints scaled text horizontally.
(or ST)

VSCALE-TEXT: Prints scaled text vertically.
(or VST)

{name}: Font name.

{width}: Font width (point size).

{height}: Font height (point size).

{x}: Horizontal starting position.

{y}: Vertical starting position.

{data}: Text to be printed.

SCALE-TEXT Example

Input (SCALE.LBL):

```
! 0 200 200 300 1
CENTER
; Print using x and y scales of 10 points
SCALE-TEXT PLL_LAT.CSF 10 10 0 10 10 POINT FONT
; Print using x scale of 20 points and y scale
; of 10 points
SCALE-TEXT PLL_LAT.CSF 20 10 0 80 WIDER FONT
; Print using x scale of 10 points and y scale
; of 20 points
SCALE-TEXT PLL_LAT.CSF 10 20 0 150 TALLER FONT
FORM
PRINT
```

Output:

10 POINT FONT

WIDER FONT

TALLER FONT

SCALE-TO-FIT Commands

The SCALE-TO-FIT commands automatically calculate the scale in order to fit text inside a window.

Format:

{command} {name} {width} {height} {x} {y} {data}

where:

{command}: Choose from the following:

SCALE-TO-FIT: Prints scaled text horizontally.
(or STF)

VSCALE-TO-FIT: Prints scaled text vertically.
(or VSTF)

{name}: Font name.

{width}: Unit-width of the window.

{height}: Unit-height of the window.

{x}: Horizontal starting position.

{y}: Vertical starting position.

{data}: Text to be printed.

SCALE-TO-FIT Example

Input (STF.LBL):

```
! 0 200 200 100 1
IN-MILLIMETERS
CENTER
; Fit a text string into an area 40mm wide by 10mm
; high
SCALE-TO-FIT PLL_LAT.CSF 40 10 0 10 SALE
; Fit a longer text string into the same 40mm wide
; by 10mm high area
SCALE-TO-FIT PLL_LAT.CSF 40 10 0 20 SALE PRICE
; Fit "SALE" text into a 40mm wide by 20mm high
; area
SCALE-TO-FIT PLL_LAT.CSF 40 20 0 30 SALE
FORM
PRINT
```

Output:

SALE
SALE PRICE
SALE

SCALABLE CONCATENATION Commands

Scalable concatenation allows you to assign different character styles to strings, printing them with uniform spacing on the same text line. Both scalable and bitmap text can be combined between a CONCAT/ENCONCAT command. See also Text Concatenation Commands

Format:

```
{command} {x} {y}  
<ST> {name} {width} {height} {offset} {data}  
  "      "      "      "      "      "  
<ST> {name} {width} {height} {offset} {data}  
<ENDCONCAT>
```

where:

{command}: Choose from the following:

CONCAT Horizontal concatenation.
VCONCAT Vertical concatenation.

{x}: Horizontal starting position.

{y}: Vertical starting position.

{name}: Font name.

{width}: Font width point size.

{height}: Font height point size.

{offset}: Unit-value to offset text from the starting position. Used to align individual text strings or create superscript/subscript characters.

{data}: Text to be printed.

<ENDCONCAT>: Terminates concatenation.

SCALABLE CONCATENATION Example

Input (STC.LBL):

```
! 0 200 200 210 1
CENTER
; Concatenate 3 scalable font strings and 1
; Resident font string
CONCAT 0 20
4 1 0 2/
ST PLL_LAT.CSF 20 20 15 $
ST PLL_LAT.CSF 40 40 0 22
ST PLL_LAT.CSF 20 20 0 99
ENDCONCAT
FORM
PRINT
```

Output:

2/\$22⁹⁹

ROTATE Commands

ROTATE commands are used to rotate all subsequent scalable text fields at a specified angle. Rotation direction is counter-clockwise about the center point of the text. This rotation remains in effect until another ROTATE command is issued. Default angle is zero degrees.

Format:

{command} {angle}

where:

{command}: Choose from the following:

ROTATE
(or R) Rotates scalable fonts.

{angle}: Degree of rotation (ccw).

ROTATE Example

Input (ROTATE.LBL):

```
! 0 200 200 440 1
CENTER
TEXT 4 1 0 50 Rotate Strings
ROTATE 45
CONCAT 50 300
ST PLL_LAT.CSF 20 20 20 $
ST PLL_LAT.CSF 40 40 0 22
ST PLL_LAT.CSF 20 20 0 99
ENDCONCAT
FORM
PRINT
```

Output:

Rotate Strings

\$22⁹⁹

LINEAR BAR CODES

Bar codes allow easy implementation of automated identification, cataloging and processing of almost any object. They have been successfully used on items ranging in size from boxcars to bumblebees.

This overview of bar code symbologies will help when programming Zebra mobile printers and/or designing labels with Label Vista software.

If you plan to create software using these bar codes, we recommend ordering the uniform symbology specification from AIM or the UCC to determine the uses and limitations pertaining to that type of bar code. The information in this document is in no way complete.

The following discussions contain basic information and some suggested applications for each type of bar code. The quick reference table in Appendix E lists specific data for each bar code in one location. All the information on ideal widths and ratios comes directly from the uniform symbology specification. Please note that all measurements contained in this document are in printer dots. On 200 dot per inch (dpi) printers, one dot is equal to 0.005" or 0.13 millimeters, on 300 dpi printers one dot is equal to 0.003" or 0.07 millimeters.

RESIDENT LINEAR BAR CODE SAMPLES

UPC-A



01234567890

UPC-E



0123450

EAN-13



0012345678905

EAN-8



40153476

UPC/EAN Plus 2
Extender



12

UPC/EAN Plus 5
Extender



12345

CODE 39



CODE 39

CODE 93



C93/Ext.

I 2 OF 5



123456

CODE 128



CODE 128

UCC-128



0123456789

CODABAR



a12345a

PLESSY MS110



12345678

POSTNET



012345678

UPC AND EAN/JAN BAR CODES

UPC and EAN/JAN bar codes are typically used for marking products with a unique code used to look up prices and to track inventories of goods sold. They are also used for store coupons, periodicals, and paperback books. UPC and EAN/JAN bar codes are generally rectangular, contain a fixed amount of data, and in most cases are accompanied by human readable text printed below them. For best results, this text should be an OCR-A (resident font 2), a sans serif font (resident font 7) or an OCR-B font.

The first number in the UPC/EAN bar code is the number system character. The specification lists use of characters 0 through 9 as follows.

- 0 Regular UPC codes (UPC-A and UPC-E)
- 1 Reserved
- 2 Random weight items, like store packaged meat. (UPC-A only)
- 3 National Drug Code and National Health Related Items Code in current 10-digit code length (UPC-A only)*
- 4 In-store marking of non food items without code format restriction and with check digit protection (UPC-A only)
- 5 Coupons (UPC-A only)
- 6 Regular UPC codes (UPC-A only)
- 7 Regular UPC codes (UPC-A only)
- 8 Reserved
- 9 Reserved

* Number system 3 has the following note in the specification. "...the symbol is not affected by the various internal structures possible with the NDC or HRI codes." The users should determine what effect this statement may have on their program. It will not change how bar codes are printed.

The checksum is the last number in the bar code and can be used to make certain that the bar code is decoded properly. This digit is *automatically* calculated by the printer. The UPC bar code specification has the full instructions for calculating this checksum. The methodology is as follows:

For this example, the bar code will be 01234567890.

Step 1: Starting at the left, including the number system character, add up all the numbers in the ODD positions. ($0 + 2 + 4 + 6 + 8 + 0 = 20$)

Step 2: Multiply this sum by 3. ($20 \times 3 = 60$)

Step 3: Starting at the left again, add up all the numbers in the EVEN positions. ($1 + 3 + 5 + 7 + 9 = 25$).

Step 4: Add the results from step 2 and step 3. ($60 + 25 = 85$)

Step 5: The checksum is the smallest number when added to step 4 will equal a multiple of ten. In our example: $85 + 5 = 90$, which is a multiple of 10. Therefore, the check digit should be 5. It is called a modulo checksum because you take the modulo, or remainder, of the sum. For the programmers, it is:

$10 - (85 \bmod 10) = \text{the checksum.}$

UPC-A and EAN13 bar codes can be created with and without a checksum supplied. If the programmer supplies a checksum digit, the printer will create the bar code with that check digit, *even if it is incorrect*. Most laser scanning devices will not be able to decode the bar code if the check digit is incorrect,

UPC-E bar codes, useful for small items like candy and gum, are created through the method of “zero suppression.” For example, if you were to encode 01000000567, the resulting bar code would be a compressed bar code

that only contains the data, the compression scheme, and the checksum without all the extra zeros. For our example, the bar code would decode to 1056707. Please refer to the UPC Symbol Specification Manual from the Uniform Code Council for more information on zero suppression.

UPC-E and EAN8 bar codes have a few restrictions. First, the number system character must be set to 0. Number systems 1 through 9 do not support UPC-E and EAN8 bar codes and may not be decoded by a laser scanning device. In case your application requires it, the number system may be set to something other than 0. Second, if the programmer supplies a checksum digit, the printer will create the bar code with that check digit, *even if it is incorrect*. If the check digit is incorrect, most laser scanning devices will not be able to decode the bar code. Therefore, the programmer may send six digits (no number system, no checksum), seven digits (number system, no checksum), or eight digits (number system and checksum) and create a bar code.

Plus 2 and plus 5 bar code extensions are only used for periodicals and paperback books. Specifically, the bar code specification states that the plus 2 extension should only be used for a periodical issue number. The plus 2 and plus 5 extensions do not contain any checksum according to the bar code specification.

To create an extended bar code, place a space between the data that should go into the UPC/EAN bar code and the data that should go into the extension. You can also use the PLUS2 and PLUS5 bar code types to create the extension separately. Remember to leave ample space (about 9 times the ratio) between the UPC/EAN bar code and the extension.

UPC/EAN SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal Wide/Narrow Ratio	Ideal Narrow Dot Width	Checksum Calculation
UPC-A	UPCA	11 or 12 digits	0-9 only	2:1	2	mod 10
UPC-A plus 2	UPCA2	13 digits	0-9 only	2:1	2	mod 10 (UPC-A)
UPC-A plus 5	UPCA5	16 digits	0-9 only	2:1	2	mod 10 (UPC-A)
UPC-E	UPCE	6, 7 or 11 digits	0-9 only	2:1	2	mod 10
UPC-E plus 2	UPCE2	8 or 13 digits	0-9 only	2:1	2	mod 10 (UPC-E)
UPC-E plus 5	UPCE5	11 or 16 digits	0-9 only	2:1	2	mod 10 (UPC-E)
EAN/JAN-13	EAN13	12 or 13 digits	0-9 only	2:1	2	mod 10
EAN/JAN-13 plus 2	EAN132	14 digits	0-9 only	2:1	2	mod 10 (EAN13)
EAN/JAN-13 plus 5	EAN135	17 digits	0-9 only	2:1	2	mod 10 (EAN13)
EAN/JAN-8	EAN8	6, 7 or 8 digits	0-9 only	2:1	2	mod 10
EAN/JAN-8 plus 2	EAN82	9 digits	0-9 only	2:1	2	mod 10 (EAN8)
EAN/JAN-8 plus 5	EAN85	12 digits	0-9 only	2:1	2	mod 10 (EAN8)

CODE 39 OR CODE 3 OF 9 BAR CODES

The Code 39 bar code is used for many applications including inventories, hospital applications, or any other place where the code length may change between items being scanned (e.g. a bar code stating there were 420 pieces in one box and 20004 pieces in another box would have a different physical length). This bar code can use the characters 0 through 9, A through Z, '-' (dash), "." (period), space, "\$" (dollar sign), "/" (forward slash), "+" (plus) and "%" (percent). There is also a special character called "S/S" used as a start/ stop character. The F39 and F39C types allow the use of carriage return, line feed, and null characters.

The checksum for this bar code is located as the last (or least significant) digit of the decoded bar code. To **assure** data integrity in your application, use a bar code with a checksum. The printer will *automatically* supply

this digit if the user selects a 39C or a F39C bar code.

Check Character Numerical Value Table

Char	Value	Char	Value	Char	Value	Char	Value
0	0	C	12	O	24	-	36
1	1	D	13	P	25	.	37
2	2	E	14	Q	26	SPACE	38
3	3	F	15	R	27	\$	39
4	4	G	16	S	28	/	40
5	5	H	17	T	29	+	41
6	6	I	18	U	30	%	42
7	7	J	19	V	31	\$(full)	43*
8	8	K	20	W	32	% (full)	44*
9	9	L	21	X	33	/(full)	45*
A	10	M	22	Y	34	+(full)	46*
B	11	N	23	Z	35		

* Full represents F39 or F39C for Full ASCII

Refer to the full bar code symbology specification for complete information on checksum calculation. For a short example, take an example bar code with the data "CODE 39".

Step 1: Assign a value to each character per the Character Numerical Value Table above. C=12, O=24, D=13, E=14, space = 38, 3=3, 9=9.

Step 2: Add the values 12+24+13+14+38+3+9=113.

Step 3: Divide this number by 43. The remainder or modulo, 27, is the checksum.

Step 4: Referring to the table, 27 is the character R.

Therefore, the checksum in the bar code should be R. The final code reads "CODE 39R" when it is decoded.

CODE 39 (3 OF 9) SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal	Ideal	Checksum Calculation
				Wide/Narrow Ratio	Narrow Dot Width	
Code 39	39	Variable	Refer text	2.5:1	2	none
	39C	Variable	Refer text	2.5:1	2	mod 43
	F39	Variable	Refer text	2.5:1	2	none
	F39C	Variable	Refer text	2.5:1	2	mod 43

CODE 93 OR CODE 9 OF 3 BAR CODES

The Code 93 bar code is used for applications that require heavy error checking capabilities. To accomplish this, the Code 93 bar code contains two separate error checking checksums that are automatically calculated and placed into the bar code. This bar code is used for inventories, hospital applications, or any other place where the length may change between items being scanned. (See Code 39 above.) This bar code type can use the entire ASCII 128 character set. It is useful for encoding data and phrases like "Code 93".

The two checksums in this bar code are located as the last and second to last characters in the decoded bar code. Code 93 has a complex checksum calculation. Please see the bar code symbology specification for information on how to create and decode this checksum. Please also note that the bar code symbology specification does not state any ideal values for the ratio and the width of the narrow bar.

CODE 93 SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal Wide/Narrow Ratio	Ideal Narrow Dot Width	Checksum Calculation
Code 93	93	Variable	128 ASCII	1.5:1	1	two mod 47

INTERLEAVED 2 OF 5 BAR CODES

The Interleaved 2 of 5 (or ITF) bar code is used for applications that have a fixed data length for all items scanned. A date, telephone number, or a SKU of fixed length would be a good application for this bar code. The symbology specification states that a ITF bar code may be partially decoded without any recognizable difference. Therefore, to prevent this problem, you must keep the length of data to a constant and perform an error checking routine on the decoding program to determine if the data is correct.

Only the digits 0-9 can be encoded, and there should

be an even number of digits in the data. If there is an odd number of digits, the printer will automatically insert a zero (0) at the beginning of the bar code.

There are two bar code varieties with a checksum: Interleaved 2 of 5 “with checksum” and German Post Code. German Post Code has fixed length – either 12 or 14 characters (including checksum).

Here is an example how to calculate modulo 10 checksum:

Step 1: To calculate the checksum, first ensure that you are starting with an odd number of digits in the data. If not, add a zero (0) to the beginning of the data.

Step 2: Multiply every other digit by 3, and add up the numbers. So, if your data was “43827” your calculation should be $(4 \times 3) + 3 + (8 \times 3) + 2 + (7 \times 3) = 62$.

Step 3: Divide this number by 10, resulting in 6 with a remainder of 2. Subtract the remainder from 10. In our example, $10 - 2 = 8$. The checksum is this final number, 8. Append this to the end of your data. Note that if the remainder was a zero, your checksum should be zero.

INTERLEAVED 2 OF 5 SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal Wide/Narrow Ratio	Ideal Narrow Dot Width	Checksum Calculation
Interleaved 2 of 5	I2OF5	Varies	0-9 only	2.5:1	2	See text
Interleaved 2 of 5 with checksum	I2OF5C	Varies	0-9 only	2.5:1	2	See text
German Post Code	I2OF5G	11, 12, 13 or 14	0-9 only	2.5:1	2	mod 10, weights 4,9

CODE 128 AND THE UCC-128 SHIPPING STANDARD

Code 128 is used for applications that need to contain a large amount of data such as shipping applications, marking blood donations, and any other application that can vary in length between bar codes being scanned. The bar code also contains a checksum as the last char-

acter in the code which ensures that data remains intact.

Code 128 can use the entire ASCII 128 character set as well as other subsets available in the universal symbology specification. The three start and stop characters determine which character set to use. The checksum for this bar code is located immediately before the stop character. The bar code symbology specification contains all the information on calculating this checksum. For a short example, we desire to encode "BAR128" in a bar code. We will use "A" as our start and stop character in this example.

Step 1: The symbology specification assigns a numerical value for each character. Find the values of all the characters in the data.

Step 2: Add the value of the start character and all the data characters multiplied by its position in the bar code. For our example, the calculation would be $103 + (34 \times 1) + (32 \times 2) + (50 \times 3) + (17 \times 4) + (18 \times 5) + (24 \times 6) = 672$.

Step 3: Divide this number by 103. The remainder or modulo, 54, is the checksum. The character that assigned to 54 in the specification is "V". Our final code will look like "ABAR128V" where "A" is the start character, "BAR128" is the data, and "V" is the checksum.

The UCC-128 Shipping Standard is part of a document called *Application Standard for Shipping Container Codes* available from the Uniform Code Council. This 90-page guide contains the entire specification on marking any shipment sent anywhere in the United States. Seventeen pages are dedicated to the technical considerations of using, placing, and printing these bar codes. We highly recommend getting this information if your application involves shipping.

CODE 128/ UCC-128 SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal Wide/Narrow Ratio	Ideal Narrow Dot Width	Checksum Calculation
Code 128 /A/B/C/Auto	128	Variable	Refer text	1.5:1	1	mod 103
UCC-128Std.	UCCEAN	Refer text	Refer text	3:1	3	mod 103

CODABAR

Codabar is ideal for applications that contain mostly numeric symbols that may vary in length from bar code to bar code. It can encode the digits from 0 to 9, the characters “-”(dash), “\$”(dollar sign), “:”(colon), “/”(forward slash), “.”(period), and “+”(plus) as well as start/stop characters A through D.

One optional checksum is automatically appended as the least significant digit in the bar code data directly before the stop character. The bar code symbology specification contains all the information on calculating the checksum. As a short example, our data will be “A37859B” where A and B are start/stop characters. The characters 0 through 9 are assigned the numerical values 0 through 9 respectively. “-” is 10, “\$” is 11, “:” is 12, “/” is 13, “.” is 14, “+” is 15, and start/stop characters A B C and D are 16, 17, 18, and 19 respectively.

Step 1: Add the numerical value of all the characters.

$$16 + 3 + 7 + 8 + 5 + 9 + 17 = 65.$$

Step 2: Divide this number by 16 and use the remainder, or modulo. In our example, this is 1

Step 3: Subtract the modulo from 16. This is the smallest number that can be added to the sum in step 1 to make a multiple of 16. (65 + 15 =80. 80 /16= 5)
Therefore, the check sum for our example is 15.

Step 4: The character that corresponds to 15 is “+” and is added in before the stop character. Our final bar code looks like “A37859+B”.

The bar code type NW7 is for reverse compatibility only. We do not recommend using this command for new systems. There is no difference between CODABAR and NW7.

CODABAR SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal Wide/Narrow Ratio	Ideal Narrow Dot Width	Checksum Calculation
Codabar	CODABAR	Variable	0-9,A-D symbol	2.5:1	2	none
	CODABAR	Variable	0-9,A-D, 16 symbol	2.5:1	2	mod 16

MSI PLESSEY BAR CODES

The MSI Plessey bar code is a fixed length code that uses only numerical characters. It is primarily used for grocery applications. Three different types of encoding exist with different levels of data protection. Please see the bar code symbology specification for more information on how to calculate these checksums.

The bar code type “PLESSEY” is used for reverse compatibility only. We do not recommend using this command for new systems. The PLESSEY type will force a 2:1 ratio of the wide to narrow bar width.

MSI PLESSY SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal Wide/Narrow Ratio	Ideal Narrow Dot Width	Checksum Calculation
MSI Plessey	MSI	13 digits max	0-9 only	2:1	2	none
	MSI10	13 digits max	0-9 only	2:1	2	mod 10
	MSI1010	13 digits max	0-9 only	2:1	2	two mod 10
	MSI1110	13 digits max	0-9 only	2:1	2	mod 11 mod 10

POSTNET AND FACING IDENTIFICATION MARKS

The US Postnet bar code is used only to help automate mail delivery. To comply with postal regulations, set the height of the bar code to 30 dots, the wide/narrow bar ratio at 3.5:1, and the width of the narrow bar to 3 dots on a 200 dpi printer. The data sent to the bar code can be 5, 9, or 11 digits long. For example, to send mail to 30 Plan Way, Warwick, RI 02886-1234, the data would be

5 digits- ZIP Code only: 02886

9 digits - ZIP + 4 Code: 028861234

11 digits - ZIP + 4 Code and last two digits in address:
02886123430

The Postnet bar code also contains an automatically calculated checksum as the last character in the decoded bar code. As a short example, our data will be "02881123430"

Step 1: Add the numerical value of all the characters.
 $0+2+8+8+1+1+2+3+4+3+0 = 32$.

Step 2: Divide this number by 10 and use the remainder, or modulo. In our example, this is 2

Step 3: Subtract the remainder (or modulo) from 10 to get the check sum. The check sum for our example would be 8 ($10 - 32 \text{ mod } 10$ for programmers).

A Facing Identification Mark (FIM) is the bar in the upper right corner of an envelope near the stamp. To comply with postal regulations, set the height of the bar code to 125 dots, the ratio to 1.5:1 dots, and the width of the narrow bar to 6 dots. There are three different characters you can send as data: A, B, and C.

FIM A: Courtesy Reply Mail with Postnet Bar code

FIM B: Business Reply Mail, Penalty Mail, or Franked

Mail without Postnet Bar code

FIM C: Business Reply Mail, Penalty Mail, or Franked Mail with Postnet Bar code.

For more information, please see Publication 25 from the USPS Postal Business Center. If you are making a label with an address, try using resident font 7 or font 4 for best results with the optical character recognition software used by the post office.

POSTNET AND FIM SPECIFICATIONS

Bar code Symbology	Bar code Type	Input Length	Characters	Ideal Wide/Narrow Ratio	Ideal Narrow Dot Width	Checksum Calculation
Postnet	POSTNET	5, 9, 11 digits	0-9 only	3.5:1	3	mod 10
Facing						
Ident Mark	FIM	A, B, or C only	A, B, or C	1.5:1	6	N/A

BAR CODE COMMANDS

The following commands are used for the creation and formatting of bar codes on labels. Insure that the bar code symbology chosen agrees with its intended use, and that it conforms to the guidelines in the previous section.

A Quick Reference Guide for the linear bar code symbologies discussed in this manual can be found in Appendix “E”.

Note that the “COUNT” command is also discussed in Section 3 of this manual.

BARCODE Command

The BARCODE command prints bar codes in both vertical and horizontal orientations at specified widths and heights.

Standard Bar Codes

Format:

{command} {type} {width} {ratio} {height} {x} {y} {data}

where:

{command}: Choose from the following:

BARCODE Prints bar code horizontally.
(or B)

VBARCODE Prints bar code vertically.
(or VB)

{type}: Choose from the following table:

<u>Symbology:</u>	<u>use:</u>
UPC-A	UPCA, UPCA2, UPCA5
UPC-E	UPCE, UPCE2, UPCE5
EAN/JAN-13	EAN13, EAN132, EAN135
EAN/JAN-8	EAN8, EAN82, EAN 85
Code 39	39, 39C, F39, F39C
Code 93/Ext. 93	93
Interleaved 2 of 5	I2OF5
Interleaved 2 of 5 with checksum	I2OF5C
German Post Code	I2OF5G
Code 128 (Auto)	128
UCC EAN 128	UCCEAN128
Codabar	CODABAR, CODABAR16
MSI/Plessey	MSI, MSI10, MSI1010, MSI1110
Postnet	POSTNET
FIM	FIM

{width}: Unit-width of the narrow bar.

{ratio}: Ratio of the wide bar to the narrow bar.
Refer to the table in Appendix “E” for appropriate settings.

0 = 1.5 : 1	20 = 2.0:1	26 = 2.6:1
1 = 2.0 : 1	21 = 2.1:1	27 = 2.7:1
2 = 2.5 : 1	22 = 2.2:1	28 = 2.8:1
3 = 3.0 : 1	23 = 2.3:1	29 = 2.9:1
4 = 3.5 : 1	24 = 2.4:1	30 = 3.0:1
	25 = 2.5:1	

Note: The ratios in the Appendix are suggested for best results, however any ratio can be assigned.

{height}: Unit-height of the bar code.

{x}: Horizontal starting position.

{y}: Vertical starting position.

{data}: Bar code data.

BAR CODE Example

Input (BARCODE.LBL):

```
! 0 200 200 210 1
BARCODE 128 1 1 50 150 10 HORIZ.
TEXT 7 0 210 60 HORIZ.
VBARCODE 128 1 1 50 10 200 VERT.
VTEXT 7 0 60 140 VERT.
FORM
PRINT
```

Output:



BARCODE-TEXT Command

The BARCODE-TEXT command is used to label bar codes with the same data used to create the bar code. The command eliminates the need to annotate the bar code using separate text commands. The text will be centered below the bar code.

Use BARCODE-TEXT OFF (or BT OFF) to disable.

Format:

{command} {font number} {font size} {offset}

where:

{command}: BARCODE-TEXT
 (or BT)

{font number}: The font number to use when
 annotating the bar code.

{font size}: The font size to use when annotat-
 ing the bar code.

{offset}: Unit distance to offset text away
 from the bar code.

BARCODE-TEXT Example

Input (BARTEXT.LBL):

```
! 0 200 200 400 1
JOURNAL
CENTER
; Annotate bar codes using font 7 size 0
; and offset 5 dots from the bar code.
BARCODE-TEXT 7 0 5
BARCODE 128 1 1 50 0 20 123456789
VBARCODE 128 1 1 50 40 400 112233445
BARCODE-TEXT OFF
FORM
PRINT
```

Output:



COUNT Command

The COUNT command is used for printing multiple labels where a numeric text field or numeric data encoded in a bar code is to be incremented or decremented for each label. The TEXT/BARCODE command string must contain this numeric data as the last characters of the string. The numeric data portion can be up to 20 characters, and can be preceded by the '-' sign. Counts of 9 → 0 will wrap to 9. Counts of 00 → 99 will wrap to 00. Leading zeros will be retained. Up to 3 COUNT commands can be used in a label file.

The numeric data incremented/decremented is contained in the TEXT or BARCODE command that immediately preceded the COUNT command.

Format:

{command} {numeric value}

where:

{command}: COUNT

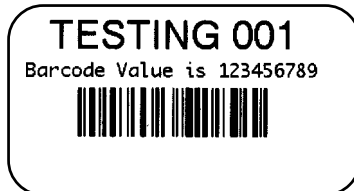
{numeric value}: Any integer value up to 20 characters. The value can be preceded by a '-' sign if decrementing of the TEXT/BARCODE value is desired. Leading zeros will be retained in the output.

COUNT Example

Input (COUNT.LBL):

```
! 0 200 200 210 3
; Print 3 labels
CENTER
TEXT 4 0 0 50 TESTING 001
COUNT 1
TEXT 7 0 0 100 Barcode Value is 123456789
COUNT -10
BARCODE 128 1 1 50 0 130 123456789
COUNT -10
FORM
PRINT
```

Output:



TWO-DIMENSIONAL BAR CODES

Introduction

A two dimensional bar code can be regarded as a “portable database.” For example, if a package has a serial number encoded with a linear bar code, you could scan the serial number, look up the number in a computer system, and gather the information about that package. If the computer system was unavailable for any reason the information you were looking for would also be unavailable. A 2D bar code, however, can contain several different fields of information in them, essentially a database attached to a package.

The first 2D bar code was just an extension of a one-dimensional bar code. The user could stack several Code 39 bar codes on top of another. These types of codes are called “stacked symbology” or “multi-row codes”. Development of scanners capable of 2D scanning allowed use of more compact and useful symbologies like PDF417 from Symbol and MaxiCode from the United Parcel Service.

Zebra mobile printers have capabilities to print PDF417 and MaxiCode symbols. If your application requires the use of PDF417 or MaxiCode symbols, we highly recommend acquiring the universal symbol specification to assist your programming. The specifications include suggestions on how to structure your data in the code to make it easier to use.

Please note that you must load a special application before using PDF417 or MaxiCode on your Zebra mobile printer. The standard application loaded in your printer does not support the ability to print these bar codes. Contact the Help Desk for more information on how to load this application.

PDF417

The PDF417 bar code is a 2D bar code that can contain a very large amount of data in a small space. If you look at a PDF417 bar code carefully, it is actually a stack of smaller bar codes. The number and height of the stacks are controllable by the user. The bar code can contain the entire ASCII 255 set of characters, and has the capability to use different encoding schemes and different levels of security to correct errors. The largest amount of data that can be encoded is 2725 characters

MaxiCode

MaxiCode was originally designed by the United Parcel Service to help them automate package delivery and sort packages coming down a quick conveyor belt. It consists of a bulls eye to help the imaging system “target” the bar code and an array of hexagons to represent the data stored in the bar code. It can contain a maximum of 93 alphanumeric characters or 138 numeric characters, has two different error correction modes, seven different storage modes, the capability to use character sets other than ASCII, and the capability to “link” several MaxiCodes together. Since there are so many different modes, it is suggested that you contact AIM for the symbol specification. If you are developing software to be used with the UPS MaxiCode shipping system, contact UPS for information on how to order the information on the Maxicode shipping system.

TWO-DIMENSIONAL BAR CODE COMMANDS

The following commands are used for the creation and formatting of two-dimensional bar codes. Insure that the bar code symbology chosen agrees with its intended use. We recommend acquiring the specification for the desired code to supplement the discussions in this manual. Note that Codablock “A” and “F” are not fully supported in this release of the manual.

PDF417 (PORTABLE DATA FILE)

Format:

```
{command} {type} {x} {y} [XD n] [YD n] [C n] [S n]  
{data}  
<ENDPDF>
```

where:

{command}: Choose from the following:

BARCODE Prints bar code horizontally.
(or B)

VBARCODE Prints bar code vertically.
(or VB)

{type}: PDF-417

{x}: Horizontal starting position.

{y}: Vertical starting position.

[XD n]: Unit-width of the narrowest element.
Range is 1 to 32.
Default is 2.

[YD n]: Unit-height of the narrowest element.
Range is 1 to 32.
Default is 6.

[C n]: Number of columns to use. Data columns do not include start/stop characters and left/right indicators.
Range is 1 to 30.
Default is 3.

PDF 417 (PORTABLE DATA FILE) continued

(continued)

[S n]: Security level indicates maximum amount of errors to be detected and/or corrected.
Range is 0 to 8.
Default is 1.

{data}: Bar code data.

<ENDPDF>: Terminates PDF-417.

Note: The BARCODE-TEXT command does not work with the PDF-417 bar code type. Any desired human readable text must be entered seperately with the TEXT command as in the example below.

PDF417 Example

Input (PDF.LBL):

```
! 0 200 200 210 1
B PDF-417 10 20 XD 3 YD 12 C 3 S 2
PDF Data
ABCDE12345
ENDPDF
T 4 0 10 120 PDF Data
T 4 0 10 170 ABCDE12345
FORM
PRINT
```

Output:



PDF Data
ABCDE12345

MAXICODE

Revised: Application Version 25 or higher

The Maxicode bar code now handles all the symbols defined by the United Parcel Service as well as the basic fields supported in the standard bar code. Maxicode supports all standard printable characters with automatic conversion of all lowercase letters in the secondary message to uppercase. This revision supports only Mode 2 bar codes.

Format:

```
{command} {type} {x} {y}
{tag} {options}
...
{tag} {options}
<ENDMAXICODE>
```

where:

{command}: BARCODE or B

Prints bar code.

{type}:

MAXICODE

{x}:

Horizontal starting position

{y}:

Vertical starting position

{tag}:

Tags not supplied will be filled with default values. Use only the tags that you require. Tags can be in any order.

<ENDMAXICODE> Final tag in Maxicode bar code.

Tags encoded in the high priority message of all Maxicodes:

<u>Tag</u>	<u>Definition</u>	<u>Default Value</u>
POST	Postal or ZIP Code	-empty-
CC	Country Code (from ISO 3166)	840 (USA)
SC	Service Class	1

Tags to control the type of bar code created:

<u>Tag</u>	<u>Definition</u>	<u>Default Value</u>
UPS5	Use UPS5 tags to create the low priority message. (On: 1, Off: 0)	0
ZIPPER	Turn the zipper and contrast patterns on or off. (On: 1, Off: 0)	0
FILLC	Low priority message fill character (Messages shorter than 84 characters will be padded with this character.)	!

Tags used when UPS5 is turned off:

<u>Tag</u>	<u>Definition</u>	<u>Default Value</u>
MSG	Low priority message field (maximum of 84 characters, overwritten by UPS5 tags)	-empty-

Tags used when UPS5 is turned on:

<u>Tag</u>	<u>Definition</u>	<u>Default Value</u>
LPMS	Low priority message header]>[RS]
HEAD	Transportation data format header	01[GS]98
TN	Tracking Number	[GS]
SCAC	Standard Carrier Alpha Code	UPSN
SHIPPER	UPS Shipper Number	[GS]
PICKDAY	Julian day of pickup	[GS]
SHIPID	Shipment ID Number	[GS]
NX	Package N of X (n/x)	[GS]
WEIGH	Package weight	[GS]
VAL	Address validation (Y or N)	[GS]
STADDR	Ship to street address	[GS]
CITY	Ship to city	[GS]
ST	Ship to state	[GS]
EXTRA	Extra user defined fields	- empty -
EOT	End of transmission character	0x004h
GS	Field separator character [GS]	0x01Dh
RS	Format type separator [RS]	0x01Eh

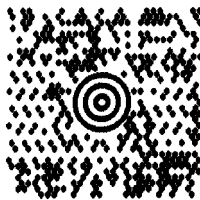
Please see the document “*Guide to Bar Coding with UPS OnLine: for Customers Generating Bar Code Labels, Version 5*” available from the United Parcel Service for more information on creating labels for the UPS shipping system.

MAXICODE Examples

Basic example with minimal required fields to print a bar code:

```
! 0 200 200 600 1  
JOURNAL  
B MAXICODE 20 20  
CC 12345  
MSG This is a MAXICODE low pri-  
ority message.  
SC 12345  
POST 02886  
ENDMAXICODE  
PRINT
```

Output:



Decodes to:

028860000[GS]057[GS]057[GS]This is a
MAXICODE low priority message.

Same example with zipper and contrast patterns:

```
! 0 200 200 600 1  
JOURNAL  
B MAXICODE 20 20  
CC 12345  
MSG This is a MAXICODE low pri-  
ority message.  
SC 12345  
POST 02886  
ZIPPER 1  
ENDMAXICODE  
PRINT
```

Output:

