Programmer's Guide

For technical personnel with basic programming knowledge and familiar with Cognitive printers.

Cognitive

105-008-01 rev A

To contact Cognitive:

691 Corporate Circle • Golden, CO 80401

TEL: +1 (800) 525-2785 • FAX: +1 (303) 273-1414

www.cogsol.com

Contents

CHAPTER 1

Programming Overview

Bar code printers are programmable devices. Most Cognitive printers use the same command language, which has become an industry standard.

NOTE: Pinnacle printers are an exception. The information in this file is not applicable to Pinnacle. If you are programming a Pinnacle printer, contact our Sales Department and order a copy of the Pinnacle Programmer's Guide, CSI P/N 10-00-0133.

In typical label printing applications, you will use simple ASCII commands to control the printer. You will write these commands in files called label formats. When sent to the printer, each label format tells the printer how to produce one or more labels.

One label format can print many similar labels. Label formats may be sent to the printer individually or in batches, in multiple file uploads. You may combine several different ASCII label formats in a single file, with each format capable of producing a different label.

This help file describes the ASCII and graphics commands used to create label formats, stored objects, and menus, as well as the VARIABLE commands used to configure the printer.

NOTE: If you are using Microsoft Windows and preparing and printing label formats directly from Notepad or another Windows-based program, be aware that most Windows printer drivers will not work with Cognitive printers. The "generic ASCII" printer driver (supplied with Windows) will pass ASCII label formats to the printer without interference. Please install and use this driver when sending ASCII label formats to the printer from the Windows environment. Do not use the Cognitive Intellidriver when sending ASCII formats to the printer. The Intellidriver converts Windows documents to ASCII label formats; thus, your label formats will print as they appear in the text editor rather than directly control the printer as intended.

Label Format Organization

With a few exceptions that are noted in the command descriptions, every label format contains:

- A header line, which defines the overall label characteristics.
- One or more printer commands.
- An END statement, which tells the printer that it has received all required data.

Here is a typical label format:

```
! 0 100 190 3
WIDTH 350
BARCODE UPCA+ 20 75 70 19112610203
END
```

This label format would print a UPCA bar code on a label.

Command Syntax

Cognitive printers will accept most commands in either an explicit (long) or implicit (abbreviated) form. Both command forms, where supported, are shown in the command descriptions. The command descriptions use the following format:

COMMAND NAME

FUNCTION: The purpose of the command is described here.

EXPLICIT FORM: `COMMAND parameters`

PARAMETERS: Any optional or required parameters are described here.

COMMENTS: Any additional comments relating to use of the command are noted here.

SEE ALSO: Lists related commands.

EXAMPLE:

```
SAMPLE PROGRAM CODE IS INCLUDED HERE
SHOWING PROPER USE OF THE COMMAND
```

NOTE: The sample code shown does not always include all the lines in the label format that produced the sample label. Header lines, END statements and the like are often omitted to save space.

Also, the label images shown only illustrate the features or command under discussion. They are not to scale. The labels your printer produces using the sample code will differ considerably from the label images in this help file.

Important Programming Rules

Use blank spaces exactly as shown in the command descriptions. Blank spaces are the delimiters between parameters. Omitting a necessary space may cause incorrect label printing.

Do not send extraneous control characters to the printer.

End every command line with a line feed or a carriage return and line feed. If you create labels using a word processor, confirm that your system uses "hard" carriage returns (inserts ASCII characters 10 and 13 at the end of each line).

Begin every label format with a header line. End every format with an END statement, unless otherwise noted in the command descriptions. (A few commands are "stand alone" and should not be followed by an END statement or any other commands.)

- Not all printers support all commands, and there may be some variation in command use depending on the printer model. Review your printer's User's Guide and the compatibility information in Tables 1, 2, and 3 before you begin writing label formats or software.

CHAPTER 2

Command Compatibility

All commands, bar codes, and fonts do not work with all printers. Commands are added with the introduction of new printers and new firmware releases. Command usage can also vary, depending on the printer's firmware. The tables following provide some general command compatibility guidelines. Y indicates that the command is supported in the current firmware version for the listed printer.

Printer Models

Printer models in the tables are designated as follows:

PW/PT42: Code Courier printers, models PW422003 and PT422003

BD/BT02: Barcode Blaster LS printers, models BD242002, BD422002, BT242002, BT422002, and early Barcode Blaster SR printers

BD/BT05: Barcode Blaster high speed printers, models BD242005, BD422005, BT242004, BT422004

BL4202: Barcode Blaster CL, model BL422003 and BL423002

ADVANTAGE: Barcode Blaster Advantage series, models BD242003, BD422003, BT242003, BT422003, and current Barcode Blaster SR model BT423002

SOLUS: Solus printer series, models SD4TI and ST4TI

Command Compatibility

The following table summarizes commands that are compatible with each printer.

TABLE 1. Printer Command Compatibility

Command Name	**PW/PT 42**	**BD/BT 02**	**BD/BT 04/05**	**BL42**	**Advantage**	**Solus**	**Del Sol**
Adjust	Y	Y	Y	Y	Y	Y	Y
variable parameter for above	-	-	-	Y	Y	Y	Y
ADJUST_DUP	Y	Y	Y	Y	Y	Y	Y
AREA_CLEAR	Y	Y	Y	Y	Y	Y	Y
BARCODE	Y	Y	Y	Y	Y	Y	Y
BARCODE(R) PDF417	Y	-	Y	Y	Y	Y	Y
BARCODE UPS	Y	-	Y	-	Y	Y	Y
BARCODE_FONT	Y	Y	Y	Y	Y	Y	Y
BARCODER	Y	Y	Y	Y	Y	Y	Y
COMMENT	Y	Y	Y	Y	Y	Y	Y
DELIMIT	-	-	-	Y	Y	Y	Y
DEFINE_VAR	-	-	-	Y	Y	Y	Y
DOUBLE	-	-	-	-	Y	Y	Y
DRAW_BOX	Y	Y	Y	Y	Y	Y	Y
END	Y	Y	Y	Y	Y	Y	Y
FILL_BOX	Y	Y	Y	Y	Y	Y	Y
GRAPHIC	-	Y	Y	Y	Y	Y	Y
Graphics mode	Y	Y	Y	Y	Y	Y	Y
Background graphics	Y	Y	Y	Y	Y	Y	Y
HALT	Y	Y	Y	Y	Y	Y	Y
Header line	Y	Y	Y	Y	Y	Y	Y
variable dot time	Y	-	Y	Y	Y	Y	Y
!A automatic header	-	-	-	Y	Y	Y	Y
INDEX	Y	Y	Y	don't use	Y	Y	Y

TABLE 1. Printer Command Compatibility

Command Name	PW/PT 42	BD/BT 02	BD/BT 04/05	BL42	Advantage	Solus	Del Sol
JUSTIFY	Y	Y	Y	Y	Y	Y	Y
LOGO mode	Y	Y	Y	Y	Y	Y	Y
Menu Commands							
MULTIPLE	Y	Y	Y	Y	Y	Y	Y
NOINDEX	Y	Y	Y	Y	Y	Y	Y
PITCH	Y	Y	Y	Y	Y	Y	Y
QUANTITY	-	-	-	Y	Y	Y	Y
QUERY REVISION	-	Y	Y	Y	Y	Y	Y
QUERY STATUS	Y	Y	Y	Y	Y	Y	Y
R90, R180, R270	Y	Y	Y	Y	Y	Y	Y
STRING	Y	Y	Y	Y	Y	Y	Y
TEXT	-	Y	Y	Y	Y	Y	Y
TIME	-	-	-	Y	Y	Y	Y
ULTRA_FONT	Y	Y	Y	Y	Y	Y	Y
Universal clear	Y	Y	Y	Y	Y	Y	Y
Wake-up string	Y	no effect	no effect	no effect	no effect	no effect	no effect
WIDTH	Y	Y	Y	Y	Y	Y	Mandatory[a]
Label pos. w/WIDTH	left	right	right	right	right	right	centered
Data storage commands:							
Delete Stored Object	-	-	-	Y	Y	Y	Y
Format Recall	-	-	-	Y	Y	Y	Y
Format Store	-	-	-	Y	Y	Y	Y
GRAPHIC STORE	-	-	-	Y	Y	Y	Y
GRAPHIC RECALL	-	-	-	Y	Y	Y	Y
Initialize Storage	-	-	-	Y	Y	Y	Y
List Stored Objects	-	-	-	Y	Y	Y	Y
Call Menu	-	-	-	Y	Y	Y	Y
Call Variable	-	-	-	Y	Y	Y	Y
VARIABLE commands:							
VARIABLE ALLOCATE	-	-	-	Y	Y	Y	Y
VARIABLE AUDIO_FREQ	Y	-	-	-	-	-	-

TABLE 1. Printer Command Compatibility

Command Name	PW/PT 42	BD/BT 02	BD/BT 04/05	BL42	Advantage	Solus	Del Sol
VARIABLE AUTOCUT	-	-	-	Y	-	Y	Y
V BUFFER_TIMED_RESET	Y	Y	Y	Y	-	-	-
VARIABLE COMM	Y	Y	Y	Y	Y	Y	Y
VARIABLE DARKNESS	Y	Y	Y	Y	Y	Y	Y
VARIABLE FEED_TYPE	Y	Y	Y	don't use	Y	Y	Y
VARIABLE HIGHSPEED	Y	Y	Y	Y	Y	Y	Y
VARIABLE INDEX SETTING	-	Y	Y	Y	Y	Y	Y
VARIABLE LOWSPEED	Y	Y	Y	Y	Y	Y	Y
VARIABLE LOW_BATTERY	Y	-	-	-	-	-	-
VARIABLE MEDIA _ADJUST	-	-	-	Y	Y	Y	Y
VARIABLE MODE	-	-	Y	-	Y	Y	Y
VARIABLE NO_MEDIA	-	Y	Y	Y	Y	Y	Y
VARIABLE NORMAL	Y	Y	Y	-	Y	Y	Y
VARIABLE ON_TIME	Y	-	-	-	-	-	-
VARIABLES ON/OFF	-	Y	Y	Y	Y	Y	Y
VARIABLE PITCH	-	-	-	Y	Y	Y	Y
VARIABLE POSITION	Y	Y	Y	Y	Y	Y	Y
VARIABLE PRINT_MODE	PT only	BT only	BT only	-	BT only	ST only	DT only
VARIABLE READ	Y	Y	Y	Y	Y	Y	Y
VARIABLE REPORT_LEVEL	-	-	-	-	Y	Y	Y
VARIABLES RESET	-	Y	Y	Y	-	Y	Y
VARIABLE SHIFT_LEFT	-	Y	Y	Y	Y	Y	Y
VARIABLE SLEEP_AFTER	Y	-	-	-	-	-	-
VARIABLE TXTBFR	Y	Y	Y	Y	Y	Y	Y
VARIABLE USER_FEEDBACK	Y	Y	Y	Y	Y	Y	Y
VARIABLE WIDTH	-	Y	Y	Y	Y	Y	Y
VARIABLE WRITE	Y	Y	Y	Y	Y	Y	Y

a. **`On the Del Sol, the WIDTH`** command is mandatory or errors will occur.

Bar Code Support

The following table summarizes bar codes supported by each printer model.

TABLE 2. Printer Bar Code Support

Bar Code	PW/ PT 42	BD/ BT 02	BD/ BT 04/ 05	BL42	Adva ntag e	Solus	Del Sol
ADD2	Y	Y	Y	Y	Y	Y	Y
ADD5	Y	Y	Y	Y	Y	Y	Y
CODE16K	Y	-	Y	-	-	-	-
CODE39	Y	Y	Y	Y	Y	Y	Y
CODE93	-	-	-	Y	Y	Y	Y
CODE128A	Y	Y	Y	Y	Y	Y	Y
CODE128B	Y	Y	Y	Y	Y	Y	Y
CODE128C	Y	Y	Y	Y	Y	Y	Y
CODABAR	Y	Y	Y	Y	Y	Y	Y
EAN8	Y	Y	Y	Y	Y	Y	Y
EAN13	Y	Y	Y	Y	Y	Y	Y
EAN128	Y	Y	Y	Y	Y	Y	Y
MAXICODE	Y	-	Y	Y	Y	Y	Y
MSI	Y	Y	Y	Y	Y	Y	Y
MSI1	Y	Y	Y	Y	Y	Y	Y
PDF417	Y	-	Y	Y	Y	Y	Y
PLESSEY	Y	Y	Y	Y	Y	Y	Y
POSTNET	-	-	-	Y	Y	Y	Y
UPCA	Y	Y	Y	Y	Y	Y	Y
UPCE	Y	Y	Y	Y	Y	Y	Y
UPCE1	Y	Y	Y	Y	Y	Y	Y
UPCA+	Y	Y	Y	Y	Y	Y	Y
I2OF5	Y	Y	Y	Y	Y	Y	Y
D2OF5	-	-	-	Y	Y	Y	Y
S2OF5	-	-	-	Y	Y	Y	Y

Font Support

The following table summarizes the fonts supported by each printer model.

TABLE 3. Printer Font Support

Cognitive Font	PW/ PT 42	BD/ BT 02	BD/ BT 04/ 05	BL42	Advantage	Solus	Del Sol
STRING 3X5	Y	Y	Y	Y	Y	Y	Y
STRING 5X7	Y	Y	Y	Y	Y	Y	Y
STRING 8X8	Y	Y	Y	Y	Y	Y	Y
STRING 9X12	Y	Y	Y	Y	Y	Y	Y
STRING 12X16	Y	Y	Y	Y	Y	Y	Y
STRING 18X23	Y	Y	Y	Y	Y	Y	Y
STRING 24X31	-	-	Y	Y	Y	Y	Y
ULTRA_FONT A	Y	Y	Y	Y	Y	Y	Y
ULTRA_FONT B	Y	Y	Y	Y	Y	Y	Y
ULTRA_FONT C	Y	-	Y	-	Y	Y	Y
TEXT 0	-	Y	Y	Y	Y	Y	Y
TEXT 1	-	Y	Y	Y	Y	Y	Y
TEXT 2	-	Y	Y	Y	Y	Y	Y
TEXT 3	-	Y	Y	Y	Y	Y	Y
TEXT 4	-	Y	Y	Y	Y	Y	Y
TEXT 5	-	-	Y	Y	Y	Y	Y
TEXT 6	-	-	Y	Y	Y	Y	Y
TEXT and ULTRA_FONT							
BARCODE_FONTS	-	-	-	Y	Y	Y	Y
STORED FONTS	-	-	-	Y	Y	Y	Y
DOUBLE byte (Kanji) fonts	-	-	-	-	Y	Y	Y

CHAPTER 3

Standard Printer Commands

This chapter discusses the following commands:

- `ADJUST`
- `ADJUST_DUP`
- `AREA_CLEAR`
- `BARCODE`
- `BARCODE(R) PDF417`
- `BARCODE UPS`
- `BARCODE_FONT`
- `BARCODER`
- `COMMENT`
- `DOUBLE`
- `DRAW_BOX`
- `END`
- `FILL_BOX`
- `GRAPHIC`
- `Graphics Mode`
- `Header Line`
- `INDEX`
- `JUSTIFY`
- `Logo Mode`
- `MULTIPLE`
- `NOINDEX`
- `OFFSET`
- `PITCH`
- `QUANTITY`
- `Query Revision`
- `Query Status`
- `ROTATE R90, R180, R270`
- `STRING`
- `TEXT`
- `TIME`
- `ULTRA_FONT`
- `Universal Clear`
- `Wake-up String`
- `WIDTH`

ADJUST

FUNCTION: Increments or decrements a variable value or numeric data on the preceding command line.

EXPLICIT FORM: `ADJUST variable nnnn`

PARAMETERS: variable = the name of the variable to be adjusted, as specified in its DEFINE_VAR command. The variable value will be adjusted wherever it is called before the ADJUST command in the label format. This is an optional parameter, and is not supported in all printers. If no variable is specified, the printer will adjust the data on the command line immediately preceding the ADJUST command.

NOTE: Do not confuse variable values (as used to represent data) with VARIABLE commands (which control the printer). Also, only printers that support the DEFINE_VAR command will support variable values. Refer to Printer command compatibility for more information.

nnnn = the incrementing or decrementing step size. Positive or negative numbers are accepted.

COMMENTS: Using ADJUST to increment or decrement alpha data can produce unexpected results.

NOTE: ADJUST will not work properly with the !+ header line parameter.

SEE ALSO: The ADJUST_DUP command.

EXAMPLE:

```
! 0 100 200 3
BARCODE CODE39 150 30 30 TEST20
ADJUST -01
STRING 12X16 150 65 ADJUST20
ADJUST 01
END
```

ADJUST_DUP

FUNCTION: Used with the ADJUST command to print non-incremented duplicates of incremented labels.

EXPLICIT FORM: `ADJUST_DUP n`

PARAMETERS: n = the number of duplicate labels printed for each increment specified with the ADJUST command.

NOTE: Only one ADJUST_DUP command is allowed in each label format. Do not use ADJUST_DUP with the HALT command.

SEE ALSO: ADJUST

EXAMPLE:

```
! 0 100 50 2
STRING 8X8 0 0 1000
ADJUST 0001
ADJUST_DUP 2
END
```

AREA_CLEAR

FUNCTION: Clears an area of a label for replotting. AREA_CLEAR may be used in a normal label format, or with the !+ header mode to combine ASCII and graphics.

EXPLICIT FORM: `AREA_CLEAR x y w h`

PARAMETERS:

x = upper left X coordinate of cleared area

y = upper left Y coordinate of cleared area w = width of cleared area

h = height of cleared area

SEE ALSO: Header line.

EXAMPLE:
```
! 0 100 560 1
PITCH 200
JUSTIFY CENTER
ULTRA_FONT A100 (20,3,0) 425 20 COGNITIVE
AREA_CLEAR 288 55 260 27
STRING 18X23 310 60 PRINTERS
END
```

BARCODE

FUNCTION: Prints a bar code, specifying type, position, height, and characters to be coded.

EXPLICIT FORM: `BARCODE typemodifiers x y h characters`

PARAMETERS: type = bar code type. Accepted types are:

UPCA, UPCE, UPCE1, UPCA+, EAN8, EAN13, EAN8+, EAN13+, EAN128, ADD2, ADD5, CODE39, I2OF5, S2OF5, D2OF5, CODE128A, CODE128B, CODE128C, CODABAR, PLESSEY. MSI, MSI1, CODE93, POSTNET, CODE16K, MAXICODE, PDF417

modifiers = optional characters that change the bar code appearance. Modifiers must directly follow the bar code type with no intervening spaces. Multiple modifiers are accepted, and their order does not matter. Available modifiers are:

- (minus) prints the bar code without uncoded subtext (not used with CODE16K, since it never has uncoded subtext).

+ (plus) adds a modulo 43 check digit to CODE39, or when used with S2OF5 or D2OF5, causes the intercharacter spacing to be equal to the width of the wide bar.

(n:w) specifications for the narrow (n) and wide (w) bars. Place these modifiers within parentheses. Allowable range is 1 to 9 for both n and w. For UPC, EAN, ADD2, ADD5, and CODE128 (A, B, and C), n specifies an integral multiplier for the bar code width. For all other codes, this option specifies the width in dots of the narrow and wide bars. The value of w must always be greater than n.

W increases the width ratio of wide to narrow bars (use only with CODE39).

X doubles the width of all bars and spaces in the bar code (use only with CODE39).

x y = Starting position of printed bar code. This is the lower left corner of the bar code block. Extender bars and bar code subtext are below this position.

h = Height of bar code in dots. Allowable range is 1 to 256. This does not include the height of the bar code subtext or extender bars (if any).

characters = ASCII characters to be bar coded.

NOTE: Not all printers support all bar codes. Refer to Table 2 for more information.

You may enter the bar code type as the last few characters of the bar code name (for example, I2OF5 can be entered as F5), but shortened identifiers may need revision if new codes are added to the printer firmware later. Always use at least enough characters to uniquely identify the code.

COMMENTS: Unless modified by a BARCODE_FONT command, all codes except UPCA+, EAN8+, EAN13+, and UPCE use an 8X8 font for bar code subtext. The excepted codes use a 5X7 font, to allow space for extender bars. Bar code subtext begins two dots below the bar code block.

All bar codes are positioned independently. This includes ADD2 and ADD5, which are normally used as add-ons for UPC and EAN codes. The programmer is responsible for the proper placement of all codes, including add-ons.

MAXICODE and PDF417 codes differ considerably from other bar codes. See BARCODE UPS and BARCODE(R) PDF417 for programming information.

SEE ALSO: The BARCODE_FONT and BARCODER commands.

NOTE: Follow the rules of the bar code in use. For example, UPCA and several other codes do not support characters A-Z. Attempting to print unsupported characters will produce bar codes that will not scan. Incorrect bar width ratios can also produce unscannable bar codes.
Character darkness and pitch also affect the reliability of the bar codes. Generally, you should not print bar codes having bar widths of less than 10 mils. Rotated bar codes should not be printed with bar widths of less than 15 mils.

EXAMPLE 1:

```
! 0 100 90 1
PITCH 100
BARCODE I2OF5 1 20 20 0123456789
BARCODE CODE39W- 1 50 20 34A
END
```

EXAMPLE 2:

```
BARCODE UPCA+ 20 75 70 19112610203
```

EXAMPLE 3:

```
BARCODE CODABAR(2:5) 10 30 20 A0123B
```

EXAMPLE 4:

```
BARCODE CODE16K 10 10 15 ab0123456789
```

EXAMPLE 5:

```
! 0 100 120 1
PITCH 100
BARCODE UPCA+ 10 95 70 04644200395
BARCODE_FONT 8X8(00,-73,1,1,1,1)
BARCODE ADD5 120 100 61 34028
STRING 8X8 10 5 ISBN 0-395-34028-4
END
```

BARCODE(R) PDF417

FUNCTION: Prints a two-dimensional bar code on a label, using the PDF417 symbology.

EXPLICIT FORM: `BARCODER PDF417 x y w:h ec% rows:cols bytes T M data`

PARAMETERS:

TABLE 4. Barcode(r) PDF 417 Options

Command	Description
R	Optional; indicates rotated code.
x	X starting position for bar code block.
y	Y starting position for bar code block.
w	Width (x dimension) of the narrowest element (bar or space) in the bar code.
:	Optional. When used, indicates that w and h define aspect ratio rather than absolute size in dots.
h	Height (y dimension) of each bar code row in the symbol.
ec	Error correction level; 0 through 8.
%	Optional; indicates the error correction level is expressed as the percentage of code words in the bar code that are devoted to error correction.
rows	Number of bar code rows, from 3 to 90. Entering 0 will cause the printer to calculate the number of rows.
:	Optional. When used, indicates that rows and cols define the overall symbol aspect ratio rather than absolute number of rows or columns.
cols	Number of bar code columns, from 1 to 30. Entering 0 will cause the printer to calculate the number of columns.
bytes	Number of encoded data bytes, including carriage returns and line feeds. Macro PDF functions will be invoked if this value exceeds 3072 (see comments).
T	Optional; produces a truncated bar code (the right row indicator and stop bar are replaced by a single width bar).
M	Optional; enables Macro PDF functions (see comments).
data	Data to be encoded.

NOTE: The number of bytes specified must exactly equal the number of bytes in the data that follows. The printer may not execute other commands following the BARCODE PDF417 command if the byte value is incorrect.

COMMENTS: PDF417 bar codes never have human-readable subtext.

All dimensions specified in the command are in dots. The starting position of the bar code block is normally its upper left corner, but the JUSTIFY command can right justify or center justify PDF417 codes. Place the proper JUSTIFY command before the BARCODE PDF417 or BARCODER PDF417 command that plots the bar code you wish to justify.

Macro PDF functions provide command enhancements useful for handling large amounts of data. Macro PDF can create a distributed representation of files that are too large for a single PDF417 bar code block. An M before the data will invoke macro PDF, or the printer will invoke it automatically if the supplied data will not fit in a single PDF417 symbol. With macro PDF in effect, you may add the following optional parameters before the data:

TABLE 5. Macro PDF Optional Parameters

Command	Description
I	File ID. Enter the desired file identification as a string after the I character. If the I is not followed by a valid string, the printer will select a random file ID.
N	File name. Enter the desired file name as a string after the N character.
B	Block count. The B character tells the printer to count the number of PDF417 symbols spanned by the data, and attach this number to the code.
P	Time stamp. Follow the P with an eight character string equal to the number of seconds since January 1, 1970, 00:00 GMT.
S	Sender. Place an alphanumeric string after the S character to indicate the sender.
A	Addressee. An alphanumeric string after the A character identifies the addressee.
F	File size, in bytes. The printer will calculate this value automatically.
C	Checksum. The presence of the C character will tell the printer to calculate a 16 bit CRC checksum and append it to the code.

EXAMPLE: Place double quotes (") around all strings used in the macro commands. To include a double quote within the string, precede it by a backslash (\). To include a backslash within a string, precede it with an extra backslash.

EXAMPLE:
```
PITCH 200
BARCODE PDF417 50 10 2 6 1 0 7 309
NAME:JOHN SMITH
ADDRESS:116 WILBUR
BOHEMIA, NY 11716
PHONE:516-555-4907
PHYSICIAN:DR.HARRY KLINE
STONYBROOK MED CTR
INSURANCE:AETNA
POLICY NO:918-1287345
SPOUSE:JENNIFER SMITH
HT:5'9"
WT:165
HAIR COLOR:BROWN
EYE COLOR:BROWN
ALLERGIES:NONE
DISABILITIES:NONE
BLOOD:A
SS#051-98-2374
DOB:5/24/60
END
```

BARCODE UPS

FUNCTION: Prints a two-dimensional bar code on a label, using the MaxiCode symbology. Cognitive printers may implement this bar code two ways, as described below.

IMPLICIT FORM: `BARCODE UPS x y mod data`

ALTERNATE EXPLICIT FORM. `BARCODE UPS x y mod CrLF data`

ALTERNATE IMPLICIT FORM. `B PS x y mod CrLF data`

PARAMETERS:

x x starting position for bar code block.

y y starting position for bar code block.

mod Encoding mode; allowable values are 0 through 6.

`CrLf` Carriage return and line feed.

`data` Data to be encoded. If not preceded by a carriage return and line feed, the data must conform to the format.

```
zip plus4 class ccode bytes LMdata
```

TABLE 6. Data Format

String	Definition
zip	Five digit postal (zip) code, numeric
plus4	Four digit zip code extension, numeric only
class	Three digit service class, numeric only
ccode	Three digit country code, numeric only
bytes	Number of encoded data bytes in the Low Priority Message. The number of bytes includes all carriage returns and line feeds, and must equal 84.
LMdata	Alphanumeric data to be encoded as the Low Priority Message. Uppercase alpha characters only. If the total number of bytes is less than 84, pad the data with the excla-mation point character (!) until there are 84 bytes in the message.

Most printers that support MaxiCode will automatically pad the data as required, but some earlier printers may not. Manually placing the correct number of data bytes in the message will help assure that the code prints and scans correctly when using the command with older printers.

The data does not have to follow the above format if it is preceded by a carriage return and line feed, as shown in the alternate explicit and implicit forms above. The presence of the carriage return/line feed tells the printer to encode all the data as it is encountered, regardless of order or content. But the data must conform to ANSI standards if you are using MaxiCode for common carrier shipment identification. For more information, see ANSI MH10.8.3, or Guide to Bar Coding with UPS, published by United Parcel Service, Inc. This booklet contains detailed UPS MaxiCode requirements, plus programming examples for various printers.

COMMENTS: MaxiCodes cannot be rotated, and never have human-readable subtext.

EXAMPLE: The following example uses the alternate form, with data coded following the ANSI MH10.8.3 standard.

This format will not print if copied directly to the printer. The BARCODE UPS command line is broken for Help window readability. Also, the example uses non-displayable control characters, represented as follows:

TABLE 7. Non-displayable Control Characters

Character	Equivalent
CrLf	Carriage return and line feed
Gs	ASCII 29 (decimal)
Rs	ASCII 30 (decimal)
Eot	ASCII 4 (decimal)

```
! 0 100 570 1
PITCH 200
BARCODE UPS 0 0 2
CrLf[)>Rs01Gs96841706672Gs001Gs1Z12345675GsUPSNGs12345E
Gs089GsGs1/1Gs10.1GsYGsGsGsUTRsEot
END
```

This example codes the following data:

TABLE 8. Example Code Data

Command	Data
[)>Rs	Message Header
01Gs96	Transportation Data Format Header
841706672Gs840	Postal Code, Country Code
Gs001Gs1Z12345675	Class of Service, Tracking Number
GsUPSNGs12345EGs089	SCAC, UPS Shipper Number, Julian Day of Pickup
GsGs1/1	Place holder for Shipment ID Number, Package n/x
Gs10.1GsY	Package Weight, Address Validation
Gs	Place holder for Ship To Street Address
Gs	Place holder for Ship To City
GsUT	Ship To State
Rs	End Of Format
Eot	End Of Message

BARCODE_FONT

FUNCTION: Allows selection of the font type, size, and position of the human-readable characters printed below the bar-coded information.

EXPLICIT FORM: `BARCODE_FONT type`

PARAMETERS: type - font type. Most printers can use any resident font except Ultra Font C. STRING is the default font type and will work with any printer. If you use a STRING font you only need to specify the font size and modifiers. For all other fonts you must spell out the font command name, font size, and modifiers; for example, ULTRA_FONT A20 (5,3,0) or TEXT 3.

NOTE: Not all printers will accept all font types for bar code subtext. Refer to Table 3 for more information.

You may use any font modifiers except for font rotation modifiers (bar code subtext automatically rotates with the bar code). When using STRING fonts, two more optional modifiers are available:

horadj - Offsets the printed text horizontally. Must be two digits; add leading zeros as required. May be positive or negative. If negative, the subtext is moved to the left.

vertadj - Offsets the printed text vertically. Must be two digits; add leading zeros as required. May be positive or negative. Positive values move the sub-text down, negative values move it up.

NOTE: If you use the horadj or vertadj modifiers, you must specify values for all modifiers as described for the STRING command.

See the STRING, TEXT, and ULTRA_FONT commands for other parameter details.

COMMENTS: The default subtext font for all bar codes except UPCA+, EAN8+, EAN13+, and UPCE is an 8X8 string font. The excepted codes use a 5X7 string font, to allow extra space for extender bars. Unless modified using the horadj and vertadj modifiers, the bar code subtext is placed two dot rows below the bar code block.

SEE ALSO: BARCODE and BARCODER commands.

NOTE: This command modifies all bar codes following it in the label format up to the next BARCODE_FONT command.

If you use multiple BARCODE_FONT commands in a label format and specify optional font modifiers for one or more of the commands, you must specify optional font modifiers for all of the BARCODE_FONT commands within the label format.

EXAMPLE 1:

```
BARCODE_FONT 12X16
BARCODE CODE39 5 55 15 ABC
```

EXAMPLE 2:

```
BARCODE_FONT 9x12(00,05,1,1,1,2)
BARCODE UPCAX+ 35 50 50 72773740001
```

EXAMPLE 3:

```
! 0 100 95 1
PITCH 100
BARCODE_FONT 8X8(00,-73,1,1,1,1)
BARCODE ADD5 20 90 61 34028
END
```

BARCODER

FUNCTION: Prints bar codes that are rotated 90 degrees clockwise from horizontal (not used with PDF417 or MAXICODE).

EXPLICIT FORM: `BARCODER typemodifiers x y h characters`

PARAMETERS: See the BARCODE command for parameter details.

NOTE: The starting position for the bar code specified by the x and y parameters is the upper left corner of the bar code block (as viewed when reading the bar code subtext), instead of the lower left corner as for non-rotated codes. The bar code height is measured from the starting position to the bottom of the bar code block.

COMMENTS: Print speed, print darkness, and dot time all affect the appearance of rotated bar codes. For best results, use LOW or NORMAL print speeds when printing rotated codes.

Use bar widths of 15 mils or more when printing rotated codes. Some high resolution printers can print scannable rotated codes with very narrow bars, but this is not recommended. Also, take care to use valid bar width ratios.

SEE ALSO: The BARCODE, BARCODE_FONT, and BARCODE(R) PDF417 commands. For print speed settings, see the VARIABLE NORMAL and VARIABLE LOWSPEED commands.

EXAMPLE: The following label formats print the same bar code using different dot times and bar width ratios. (Please note, though, that these formats use variable dot time. This feature is not supported in all printers.)

```
! 0 75 190 1
PITCH 100
BARCODER CODE39(2:6)- 10 0 30 1A2
END
! 0 150 90 1
PITCH 100
BARCODER CODE39(1:3)- 10 0 30 1A2
END
```

COMMENT

FUNCTION: This command is used for documenting label formats. Comment lines do not affect label printing.

EXAMPLE: `COMMENT characters`

PARAMETERS: characters is the non-printing comment. The printer ignores all characters following the COMMENT command, up to the end of the line as indicated by the presence of a line feed character (ASCII character 10).

COMMENTS: This command is primarily for internal documentation of label formats, but you can also use it to temporarily disable commands. Placing a C or the word COMMENT before the command will disable it.

Comments increase label processing time slightly, since it takes time to send them to the printer. They do not use printer memory when placed in normal label formats, but do use one byte per character when placed in stored label formats.

EXAMPLE: In the label format below, the presence of the C character at the beginning of the third line and the word COMMENT in the fourth line tells the printer to ignore everything that follows up to the next line feed.

```
! 0 100 190 1
STRING 8X8 115 5 THIS COMMENT WILL PRINT
C STRING 8X8 115 15 THIS COMMENT WON'T PRINT
COMMENT this is a comment. It won't print either.
END
```

DOUBLE

FUNCTION: Prints "double byte" text characters from a selected font set. Double byte fonts are used for characters that require greater resolution than can be provided in a single byte (specifically, Kanji characters).

EXPLICIT FORM: `DOUBLE font(eximage,exspace,xmult,ymult) x y mtid characters`

PARAMETERS: font - specifies the text font. There are two double byte fonts available at present:

TABLE 9. Double Byte Fonts

Font	Description
16	16x16 dot Kanji
24	24x24 dot Kanji

mtid - Specifies a font mapping table; must be 1 (specifies mapping table SHIFTJIS). More mapping tables may be added in future firmware releases.

See the STRING command for other parameter details.

EXAMPLE: The following label format will print text using the double byte characters. The Kanji characters are mapped above ASCII 128, so may not display correctly on your terminal.

```
! 0 100 210 1
DOUBLE 24 0 0 1 This is Kanji(24x24) ˆ ˆ¡
DOUBLE 16(0,0,2,2) 0 50 1 This is Kanji(16x16*2) ˆ ˆ¡’@
END
```

DRAW_BOX

FUNCTION: Draws a hollow rectangle on the label.

EXPLICIT FORM: `DRAW_BOX x y w h t`

PARAMETERS:

x = X coordinate of upper left corner of box.

y = Y coordinate of upper left corner of box.

w = Box width, measured in dot columns. Must be greater than zero.

h = Box height, measured in dot rows. Must be greater than zero.

`t` = Optional; specifies line thickness in dots. The default is 1. If t is greater than 1, the x, y, w, and h parameters refer to the box outside dimensions.

COMMENTS: DRAW_BOX can draw horizontal and vertical lines on labels if you set t and h or w equal to 1, but the lines will be two dots wide. Boxes drawn with this command may not print well at 200 DPI or higher resolution unless the t parameter is 2 or more. Some early printers do not support the t parameter, in which case we suggest using multiple FILL_BOX commands to create boxes with thicker lines.

SEE ALSO: FILL_BOX command.

NOTE: Avoid drawing boxes around bar codes. Vertical lines near the bar code edge may be confused by scanners as being part of the code.

EXAMPLE:

```
DRAW_BOX 5 5 100 50
DRAW_BOX 10 10 90 40
DRAW_BOX 20 30 70 1
```

END

FUNCTION: Signals the end of a label format.

EXPLICIT FORM: `END`

PARAMETERS: None

COMMENTS: The END command tells the printer that all data required for the current label format has been sent. When the END command reaches the printer, the label format is processed and printing begins. Printing normally continues until all the labels specified in the label format have printed, but it may be temporarily paused with the HALT command (or by enabling label taken mode in printers so equipped).

SEE ALSO: HALT command.

NOTE: Place an END statement at the end of every ASCII label format unless otherwise noted in specific command instructions. The printer will not print the label format until it receives an END command.

FILL_BOX

FUNCTION: This command inverts every dot in the specified rectangular area. Where the existing field is white, the FILL_BOX command fills it in black, while areas already black are flipped to white.

EXPLICIT FORM: `FILL_BOX  x y w h`

PARAMETERS:

x = X starting position for box (its upper left corner)

y = Y starting position for box (its upper left corner)

w = Width of box, measured in dot columns. Must be greater than zero.

h = Height of box, measured in dot rows. Must be greater than zero.

COMMENTS: The FILL_BOX command can produce many useful effects. Overlaying multiple filled boxes can produce wide or multiple-line borders. Overlaying filled boxes can also produce shadow effects or one-dot-wide lines. The following examples show some possible uses of the command.

NOTE: You can print black labels with white text on them using the FILL_BOX command, but this is not recommended. Printing large dark areas causes excessive heat buildup and is very hard on the printhead. Long horizontal boxes are most strenuous to print. They may adversely affect print quality, and also place considerable drain on a portable printer's battery.

SEE ALSO: DRAW_BOX command.

EXAMPLE 1: `! 0 100 180 1`

```
PITCH 200
FILL_BOX 50 50 750 85
FILL_BOX 50 50 725 60
FILL_BOX 25 25 750 85
FILL_BOX 27 27 750 85
END
```

EXAMPLE 2:

```
! 0 100 90 1
PITCH 100
ULTRA_FONT B25 (6,0,0) 70 5 SHELL 1
FILL_BOX 65 1 130 20
ULTRA_FONT B25 (4,2,0) 71 6 SHELL 1
FILL_BOX 65 1 130 30
ULTRA_FONT B25 (6,0,0) 70 40 SHELL 2
FILL_BOX 65 36 130 20
ULTRA_FONT B25 (4,2,0) 71 40 SHELL 2
END
```

GRAPHIC

FUNCTION: Initializes the printer to receive graphics data, and positions the received graphic on a label. This ASCII command allows you to send standard PCX or BMP format graphics directly from a file.

NOTE: GRAPHIC must be the last command in its label format. Do not follow it with an END command. The printer waits for graphics data and a following printable label file after receiving this command.

EXPLICIT FORM: `GRAPHIC type x y`

PARAMETERS:

`type` - the graphic file type. Allowable types are PCX and BMP.

`x y` - Starting position of the printed graphic; normally the upper-left corner unless changed by a JUSTIFY command.

COMMENTS: After the printer receives the GRAPHIC command it waits for the arrival of graphics data. The data is then mapped in the printer's memory. To print the graphic, send another label format using a !+ header line to the printer.

The printer only prints bi-level (black and white) graphics. Do not try to print color or grayscale graphics. Also, graphics are printed full scale, with each image dot corresponding to one printed dot. The printer will not scale graphics to fit.

EXAMPLE: The following two label formats will print a PCX graphic with its upper left corner at location 416,14 if the appropriate PCX file is sent to the printer following the first label format.

FORMAT 1:

```
! 0 100 600 0
JUSTIFY RIGHT
GRAPHIC PCX 416 14
```

(Send the graphics file here. The printer will print after it receives the label format below.)

FORMAT 2:

```
!+ 0 100 590 1
END
```

Graphics Mode

FUNCTION: Initializes the printer to receive binary data for printing bitmapped graphics. Unlike most commands, this command is not sent as ASCII text. Send all graphics data, including the header line, to the printer in a continuous binary data stream.

EXPLICIT FORM: `mode dottime dotrows numlbls data`

PARAMETERS:

`mode` the graphics command. Use @ for foreground graphics printing, # for background graphics printing. The mode parameter is sent as a one byte binary representation of the ASCII character.

`dottime` Determines the Y dimension of each dot. Dot time is a one byte binary number with a decimal value between 1 and 255.

`dotrows` Number of dot rows required for the image, sent as a two byte binary value with a minimum of 1 and a maximum limited by label size and available printer memory.

`numlbls` Number of duplicate labels to print, sent as a two byte binary number having a decimal value between 0 and 65535. Set this value to 0 when preparing background graphics.

`data` Binary data defining the bitmapped image. The image is coded as a raster scan, with binary 1's representing black dots and binary 0's representing white dots.

NOTE : Dot time values that result in aspect ratios less than 0.8 are not recommended, as poor print quality can result.

COMMENTS: When programming in graphics mode, you control every dot on the printhead separately. All the dots on the finished label can be either dark or light, depending on the setting for each bit in the graphics file.

Send all data to the printer in a continuous binary stream. Enough data must reach the printer to control all the dots for the specified number of dot rows over the print width. That may be quite a lot of data, since each square inch of label when printed at 200 DPI can have 40,000 dots. If your graphics image is much smaller than the printhead width, we recommend using LOGO mode programming if possible. If you cannot use LOGO mode, reducing the width with a WIDTH statement in a normal ASCII label format will help reduce the data requirement.

You cannot use a typical word processor to create binary files. This is because word processors enter every character as an ASCII code. Typing the number zero, for example, produces ASCII 48. This character leaves the computer as binary code 00110000, which is not the same as sending binary 00000000 (true binary zero).

ASCII commands cannot be mixed in the binary graphics file. If you want to print predefined objects (such as bar codes or text) on the same label with binary bitmapped graphics, you have four possible approaches:

1. Program either the graphics or the ASCII components in the background, and put the other components in the foreground.
2. Use the !+ header line mode with AREA_CLEAR to overlay existing data with new data (typically, you send the bitmap to the printer first, then use !+ to put the ASCII components over the bitmap).
3. Use LOGO mode to place graphics where you want them, then use !+ to send ASCII components to the printer.
4. Use the GRAPHIC command to send the graphics to the printer as a BMP or PCX file, then follow it with an ASCII label format containing the predefined objects.

SEE ALSO: Header line, LOGO mode, AREA_CLEAR and GRAPHIC commands.

EXAMPLE: Here is the beginning of a typical foreground graphic in ASCII form, hexadecimal form, and binary. The segment shown covers the graphics header and the beginning of the bitmap. Spaces have been added to the ASCII and hexadecimal forms for clarity. The printer will only accept the binary form.

```
ASCII:  @ 100 100 1 0000000000...
Hex:  40 64 0064 1 000000000...
Binary:
0010100001000000000000000100000000000000000000100000...
```

HALT

FUNCTION: Pauses the printer after it prints one label. If there are more labels left to print in the current batch, pressing the printer's feed switch will signal the printer to print the next label. The HALT command also activates the label cutter in printers so equipped.

EXPLICIT FORM: `HALT`

PARAMETERS: None

This feature is intended for situations that require many similar labels, presented one at a time. For example, tagging a large quantity of boxes on a warehouse floor might call for labels that are identical except for a serial number. You could write the label format using the ADJUST command to change the serial number, and use the HALT command to put printing "on hold." Only one label prints when you upload the label format to the printer. If using a portable printer, you could even disconnect the printer from the host and carry it to the warehouse with the labels ready to print as needed, without any other equipment.

Printers that have an integral cutter and label taken sensor will stream labels without cutting them unless there is a HALT command in the label format, or VARIABLE AUTOCUT is ON.

NOTE : Place the HALT command before the END command but after all commands that map components on the label (such as STRING or BARCODE).

SEE ALSO: VARIABLE AUTOCUT.

Header Line

FUNCTION: Initializes the printer to receive a label format.

EXAMPLE: `mode x dottime maxY  numlbls`

PARAMETERS:

`mode` Sets the encoding mode for the label format. Valid characters are:

TABLE 10. `Header Line` Command

Comman d	Description
!	ASCII mode: The printer treats all incoming data as ASCII commands in this mode.
@	Graphics mode: All incoming data is treated as binary graphics. (See Graphics mode for further details.)
!#	ASCII background mode: Same as ! mode, except does not immediately print labels (see below).
#	Graphics background mode: Same as graphics mode, except does not immediately print labels.
!*	Clear background mode (ASCII or GRAPHICS): Turns off background mode, recombining the foreground and background memory buffers. Does not clear memory.
!+	New start sequence. Plots a label over an existing label without erasing the previous label. (See also AREA_CLEAR command.)
!A	Automatic header line - Same as !, except remaining header line parameters are not required. The printer sets x to zero, dottime to 100, calculates MaxY, and sets numlbls to zero. MaxY is calculated based on the location of the last plotted row in the image buffer. Use this header line with the QUANTITY command or in stored label formats.

`x` X starting position for label. Older printers used this parameter to position labels horizontally so two labels could be printed side-by-side. Always set x equal to zero in any printers that support the MULTIPLE command. High speed Barcode Blaster printers automatically limit their print speed to 3 IPS when x is nonzero.

`dottime` Determines how long the printhead dots stay hot, thereby changing the dot length. Values can range from 0 to 255, but values less than 30 are treated as 30. The dot time will increment in steps of 10 when operating in linear dot time mode (see the VARIABLE MODE command). The specified dot time is rounded down to the nearest step.

NOTE : Dot time values that result in aspect ratios less than 0.8 may cause poor print quality. Dot times under 100 automatically disable high speed printing. Some printers only support dot time 100. Refer to Table 1 for more information.

`maxY` Specifies how many dot rows are memory mapped for each label. Max Y must be large enough to map all label components, but not so large that it causes memory overflow or label skipping.

`numlbls` Sets the quantity of labels to be printed by the label format. The allowable range is 0 to 65535.

COMMENTS: The ASCII background and graphics background modes do not print labels immediately. Background mode splits the printer memory into two segments: the foreground and background memory buffers. Data sent via background mode resides in the background buffer and prints when the next label format containing foreground data reaches the printer. Background data stays in memory until intentionally cleared.

All label formats require a header line. The header line precedes all other commands in the label format, with the exception of the printer wake-up string (if used).

NOTE: A few ASCII commands, such as the QUERY STATUS command, replace the normal header line. These cases are noted in their command descriptions.

Header line parameters are critical for proper printing. For most ASCII label formats you can "rough-in" header line values as follows:

Use **`!`** for ASCII foreground printing.

Use **`0`** for the X starting position.

Use a dot time of 100 for an aspect ratio of 1:1 in linear dot time mode.

Determine an approximate Max Y value by multiplying the label length (or desired label length, for cutter equipped printers) by the print pitch and then reducing the value by 10%. For dot times other than 100, divide the result by the aspect ratio (dot time divided by 100).

SEE ALSO: Graphics mode, LOGO mode, Wake-up string

EXAMPLE:

```
! 0 100 60 13
```

```
COMMENT ASCII FOREGROUND, X START=0,
COMMENT DOTTIME=100, MAX Y=60, 13 LABELS
```

INDEX

FUNCTION: Enables automatic label indexing. When printing is finished the printer feeds the label stock until the next label's first dot row is positioned under the printhead burn line (based on the position of the label's indexing cue; i.e., the black bar or gap location). This is the default operating mode. Label indexing remains on unless disabled with a NOINDEX command.

NOTE: Do not use this command when using continuous-form media. Continuous-form media has no indexing cues, so will feed continuously if automatic label indexing is enabled.

EXAMPLE: `INDEX`

PARAMETERS: None

NOTE: Since INDEX positions the first dot row of the next label at the burn line, with butt-cut label stock the label perforation may be under the edge of the printhead.

The exact index position may vary slightly from printer to printer due to calibration differences. The VARIABLE POSITION command can adjust the index position.

SEE ALSO: NOINDEX and VARIABLE POSITION commands.

EXAMPLE:
```
! 0 0 0 0
C This format will turn indexing on and feed a label
INDEX
END
```

JUSTIFY

FUNCTION: Positions Ultra Font and TEXT printing, PDF417 and MAXICODE bar codes, and GRAPHIC command images either left, right, or center of their horizontal coordinate (X coordinate for nonrotated text and bar codes, Y coordinate for rotated text and bar codes). Once invoked, the justification remains in effect for the rest of the label format or until changed by another JUSTIFY command.

EXPLICIT FORM: `JUSTIFY alignment`

PARAMETERS: alignment - can be either LEFT, RIGHT, or CENTER, and may be abbreviated as L, R, or C. LEFT aligns the component's left edge with its specified horizontal coordinate. RIGHT aligns the component's right edge with the specified horizontal coordinate. CENTER centers the text on the specified horizontal coordinate. Left justification is the default condition.

SEE ALSO: ULTRA_FONT, TEXT, BARCODE(R) PDF417, BARCODE UPS, GRAPHIC.

EXAMPLE 1:

```
JUSTIFY LEFT
ULTRA_FONT A40 (5,0,0) 400 10 LEFT JUSTIFY
JUSTIFY RIGHT
ULTRA_FONT A40 (5,0,0) 400 50 RIGHT JUSTIFY
JUSTIFY CENTER
ULTRA_FONT A40 (5,0,0) 400 90 CENTER JUSTIFY
END
```

EXAMPLE 2:

```
JUSTIFY LEFT
ULTRA_FONT A40 (5,0,90) 150 120 LEFT
JUSTIFY RIGHT
ULTRA_FONT A40 (5,0,90) 100 120 RIGHT
JUSTIFY CENTER
ULTRA_FONT A40 (5,0,90) 50 120 CENTER
END
```

Logo Mode

FUNCTION: Allows placement of bitmapped graphics in specific label areas, or windows. You specify the window sizes and locations individually. Any number of graphics windows are allowed, providing they do not overlap.

NOTE: Unlike most commands, you cannot enter the logo mode command in ASCII form. The printer will only accept it as pure binary data with no extraneous characters.

EXPLICIT FORM: `mode 0 x y w h data`

PARAMETERS:

`mode 0` Logo mode identifier. Send the 0 as one byte (hexadecimal zero). For foreground graphics, mode is the @ sign, or hexadecimal 40. For background graphics, mode is the # sign, or hexadecimal 23.

`x` Graphics window starting point X coordinate, sent as two bytes. The X coordinate equals the dot-column of the desired starting point divided by 8.

`y` Graphics window starting point Y coordinate (dot-row), sent as two bytes.

`w` Graphics window width, sent as two bytes. The width equals the window width in dot-columns divided by 8.

`h` Graphics window height in dots, sent in two bytes.

`data` Binary graphics data. Always send enough data to specify the condition of every dot in the window area. Darkened dots are programmed with a binary 1; light dots are binary 0. Program the graphics image as a raster-scan.

COMMENTS: In background mode, the maximum programmable label size is cut in half. Consider using foreground LOGO mode with the !+ header line when printing combined ASCII and graphics.

When sending the x, y, w, and h parameters, send the most significant byte first.

Clear memory by sending a dummy label format to the printer before and after using LOGO mode. This keeps residual data from spoiling the finished label.

SEE ALSO: Graphics mode, Header line, GRAPHIC command.

MULTIPLE

FUNCTION: Causes the printer to print duplicate labels side-by-side.

EXPLICIT FORM: `MULTIPLE n`

PARAMETERS:

n = the number of duplicate labels to print side-by-side. The acceptable range is 2 to 9.

The labels printed side-by-side must actually fit in the available space, otherwise they will be truncated on the right side.

NOTE: Avoid using this command with high speed Barcode Blaster printers. Barcode Blaster will automatically reduce its print speed to 3 IPS when it encounters this command, due to the increased processing requirements.

COMMENTS: Using this command when printing many small labels can save considerable amounts of label stock. The command is also useful for printing split labels.

The total number of labels printed will be the same as the quantity specified in the header line.

You may use the MULTIPLE command with the ADJUST command to print labels side-by-side with changing numeric data. However, the ADJUSTed values will only change when the printer feeds the next label, so each set of side-by-side labels will be identical.

NOTE : Place the MULTIPLE command after the header line but before any commands that map components on the label, such as STRING or BARCODE.

EXAMPLE:
```
! 0 130 70 2
PITCH 100
WIDTH 224
MULTIPLE 2
STRING 8X8 7 0 ADIDAS  4446
STRING 8X8 7 10 JOGGING
STRING 8X8 7 20 SHORTS
STRING 8X8 37 30 $14.00
BARCODE I2OF5- 17 60 20 3445478940
END
```

NOINDEX

FUNCTION: Disables index detection. The printer will stop feeding the label after printing the last dot row. With most printers, this command remains in effect until you turn the printer off or issue an INDEX command. Portable printers are an exception: in these printers, NOINDEX shuts off when the printer goes to sleep.

EXPLICIT FORM: `NOINDEX`

PARAMETERS: None

COMMENTS: Use NOINDEX when printing on paper without gaps or index bars, or when you want the printer to print multiple small labels on large label stock. You can allow space between labels by increasing max Y in the header line or by sending a dummy label format having the required number of dot rows.

NOTE: Do not use the INDEX and NOINDEX commands in the same label format.

EXAMPLE:
```
! 0 100 60 1
```

```
NOINDEX
```

OFFSET

FUNCTION: Prints labels with their origin removed from the normal 0, 0 origin. This is an obsolete command, included only for compatibility with older printers. Do not use this command when programming currently-manufactured printers. Contact your Cognitive dealer or the factory if you need more information on this command.

PITCH

FUNCTION: Sets the print density in dots per inch.

EXPLICIT FORM: `PITCH nnn`

PARAMETERS: nnn = print pitch in dots per inch (DPI). Allowable values depend on the printhead density:

TABLE 11. Printer Pitch

PH density	Default pitch	Alternate pitch
300	300	150
203	200	100
150	150	75

NOTE: The default pitch changes to the lowest available print pitch if Blazer emulation is enabled. See the VARIABLE MODE command for more information.

Decreasing the print density (decreasing nnn above) increases the size of each printed dot by a corresponding amount. This increases the size of the entire label. For example, a 100 dot-wide box printed at 200 pitch will be 0.5" wide, but the same box printed at 100 pitch will be 1" wide. Both boxes use the same amount of memory, and occupy the same number of dot rows.

Do not use PITCH more than once in any label format. Multiple PITCH commands may produce unusual effects.

NOTE: Place the PITCH command after the header line but before any commands that locate label components, such as STRING or BARCODE.

Portable printers will always default to their highest pitch with each new label format unless explicitly set to a lower pitch.

SEE ALSO: VARIABLE PITCH command.

EXAMPLE:

```
! 0 100 90 1
PITCH 200
JUSTIFY CENTER
ULTRA_FONT B20 (5,0,0) 400 10 UFONT B20 AT 200 PITCH
END
! 0 100 45 1
PITCH 100
JUSTIFY CENTER
ULTRA_FONT B20 (5,0,0) 200 10 UFONT B20 AT 100 PITCH
END
```

QUANTITY

FUNCTION: Sets the quantity of labels to be printed by the label format.

EXPLICIT FORM: `QUANTITY numlabels`

PARAMETERS: numlabels = the number of labels printed by the label format. The allowable range is 0 to 65535.

COMMENTS: Using this command is functionally identical to specifying the number of labels in the header line, but offers the programmer some added flexibility since numlabels can be a variable. The QUANTITY command also lets the programmer specify the number of labels to print when using the automatic header line (!A) feature.

NOTE: The number of labels printed will be the last quantity specified; therefore, the quantity specified by QUANTITY takes precedence over the quantity specified in the header line.

SEE ALSO: DEFINE_VAR command; Header line

EXAMPLE: The following example will print three identical labels:

```
! 0 100 90 1
PITCH 100
BARCODE I2OF5 1 20 20 0123456789
BARCODE CODE39W- 1 50 20 34A
QUANTITY 3
END
```

Query Revision

FUNCTION: Causes the printer to send firmware revision information to the host computer. The printer sends its firmware part number, revision, revision time, and revision date in response to this command. Such information is useful when developing software that must control several different printers.

EXPLICIT FORM: `!QR`

PARAMETERS: None

COMMENTS: This command does not require a header line, but must be followed by a line feed (or carriage return and line feed) and an END statement. Do not use any other commands with QUERY REVISION.

SEE ALSO: QUERY STATUS and VARIABLE USER_FEEDBACK.

EXAMPLE:

```
!QR
END
```

Query Status

FUNCTION: Causes the printer to send a status message to the host computer. The status message indicates the current printer condition (ready, printing, low battery, out of paper, etc.). Such information is useful when controlling the printer from a remote site; however, the host computer must be programmed to properly interpret the incoming data.

EXPLICIT FORM: `!QS`

PARAMETERS: None

COMMENTS: This command replaces the normal header line. It must be followed by a line feed (or carriage return and line feed) and an END statement. Do not use any other commands with QUERY STATUS. The printer can send the following status messages to the host computer when it is queried:

TABLE 12. Printer Status Messages

Message	Description
B	Background. This is the status if there is a valid image in the background memory buffer and no print tasks are pending.
C	Charging (applicable only to portable printers). The printer battery is charging and no print tasks are pending. The printer sends this message once per second if USER_FEEDBACK is ON.
R	Ready. The printer has no pending print tasks, is not charging, and the background buffer is empty.
c[a]	(lowercase letter C) Cutter error (applicable only to cutter equipped printers). The label cutter is jammed or not working correctly.
D[a]	Download error. An error has occurred while attempting to load new firmware.
E[a]	EEPROM error. This error will occur when printer is first turned on if the EEPROM checksum is invalid.
H[a]	Hot printhead. This error will occur if the printhead thermal cycles.
I[a]	Nonvolatile memory failure.

TABLE 12. Printer Status Messages

Message	Description
L[a]	Low battery. The printer sends this message eight times per second if it detects a low power condition and USER_FEEDBACK is ON. The printer will send this message once regardless of the USER_FEEDBACK state if the power drops below the normal operating voltage level (for example, when the power is switched off).
O[a]	Out of paper. The printer sends this message eight times per second if it it runs out of paper and USER_FEEDBACK is ON.
o[a]	(lowercase letter O) Out of ribbon or ribbon stalled. (Only reported by
P[b]	Printing in process, or printer paused during batch mode processing.
S[b]	Serial port communications error.
U[b]	Printhead Up. The printhead is not fully closed.
W[b]	Waiting. The printer is paused during a batch printing operation.

a. The printer sends the error message but will not respond to the QUERY STATUS command if this error occurs.

b. TT printers ONLY. The printer sends this message eight times per second if it runs out of ribbon and USER_FEEDBACK is ON.

The printer sends a five digit number after each letter. If the leading letter is a C (indicating the battery is charging), the number following the C is proportional to the battery charge state. If the leading letter is an E (indicating an EEPROM error), the number following the E is the EEPROM checksum. In all other cases, the five digit number shows the quantity of unprinted labels in the current batch.

E messages have two parts. The first part begins with an E and is followed by the calculated checksum. The second part begins with an E and is followed by the stored checksum.

NOTE: If the printer detects an EEPROM error, it will automatically load the nonvolatile RAM with default values. The printer may work, but print quality may be degraded. Contact your service technician when an EEPROM error occurs.

SEE ALSO: The VARIABLE USER_FEEDBACK command.

EXAMPLE:

```
!QS
```

```
END
```

NOTE: Use this command exactly as shown in the example. Do not use other commands (except the END command) with the QUERY STATUS command.

ROTATE R90, R180, R270

FUNCTION:

Prints STRING text on the label, rotated clockwise in three orientations:
R90Prints text rotated 90 degrees clockwise from horizontal.
R180Prints text rotated 180 degrees clockwise from horizontal (upside-down).
R270Prints text rotated 270 degrees clockwise from horizontal.

EXPLICIT FORM:
```
R90 type(eximage,exspace,xmultiplier,ymulti-
plyer) x y characters
R180 type(eximage,exspace,xmultiplier,ymultiplyer) x y
characters
R270 type(eximage,exspace,xmultiplier,ymultiplyer) x y
characters
```

PARAMETERS:

See the STRING command for parameter details. Note, however, that the X and Y starting coordinates for the string refer to different positions on the text block depending on rotation:

For R90 and R180, the X and Y starting position of the string is in the lower left corner of the text block.

For R270, the X and Y starting position for the string is in the upper left corner of the text block.

NOTE: The parameters in parentheses are optional. There are no spaces after the commas.

SEE ALSO: STRING and ULTRA_FONT commands.

EXAMPLE:
```
! 0 100 100 1
STRING 9X12 32 10 NORMAL
R90 9X12 97 16 RIGHT
R180 9X12 77 50 FLIP
R270 9X12 10 55 LEFT
END
```

STRING

FUNCTION: Prints text (ASCII characters) on a label using CSI fonts. These are non-proportional, non-compressed bitmapped fonts.

EXPLICIT FORM: `STRING type(eximage,exspace,xmult,ymult) x y characters`

PARAMETERS:

`type` - Specifies the basic font size, in dots. There are seven font sizes: 3X5, 5X7, 8X8, 9X12, 12X16, 18X23, 24X31.

The number preceding the X is the approximate character width. The number following the X is the character height. You can specify the font type just by height, if you wish.

Not all printers support all fonts. Refer to Table 3 for more information.

NOTE: The 3X5 and 24X31 fonts are uppercase only.

`x` horizontal starting position for the upper left corner of the first character in a string.

`y` vertical starting position for the upper left corner of the first character in a string.

`characters` ASCII string to be printed.

The following parameters are contained in parentheses and are optional:

`eximage` Produces bolder forms of all character sets by printing the character the number of times specified, offset by one dot column each time. When using this modifier you must specify values for exspace, xmult, and ymult. The allowable range is 1 to 9. For normal boldness, use a value of 1.

`exspace` Modifies the spacing between characters. Specify values for xmult and ymult when using exspace. The allowable range is 1 to 9. For normal spacing, use a value of 1.

`xmult` Independently expands the width of any font. The allowable range is 0 to 9 (0 multiplies by 10), except for the 18X23 and 24X31 fonts, which have ranges of 1 to 8. When using this modifier you must also specify a value for ymult. For normal font width, use a value of 1.

`ymult` Independently expands the height of any font. Range 0 to 9 (0 multiplies by 10) except for the 18X23 and 24X31 fonts, which have ranges of 1 to 8. When using this option you must specify a value for xmult. For normal font height, use a value of 1.

COMMENTS: Text printed with the STRING command is always horizontal on the label. To print rotated text, use the R90, R180, R270, TEXT, or ULTRA_FONT commands.

Some string font characters use slightly more space on the printer's memory grid than the font dimensions indicate. The space occupied by each character is called a cell. The table below shows the cell size for each font type.

TABLE 13. Font Type and Cell Size

Font Type	Cell Size
3X5	4X5
5X7	6X7
8X8	8X8
9X12	9X12
12X16	13X16
18X23	19X23
24X31	25X31

When programming label formats, you can calculate the exact amount of space required by text blocks using the cell sizes shown.

NOTE: Use of the eximage, exspace, xmultiplier, or ymultiplier string modifiers will change the font cell size.

STRING fonts are stored in user-accessible memory in many printers, and are handled as stored objects. This means they may be deleted or replaced in the field, so the actual resident fonts may differ from those delivered in the printer. Take this into account when using the STRING command.

SEE ALSO: The ROTATE, TEXT, and ULTRA_FONT commands.

EXAMPLE 1:
```
! 0 100 100 1
PITCH 100
STRING 8X8 10 0 LETTERS
STRING 8X8(2,1,1,1) 10 10 LETTERS
STRING 8X8(1,2,1,1) 10 20 LETTERS
STRING 8X8(1,1,2,1) 10 30 LETTERS
STRING 8X8(1,1,1,2) 10 40 LETTERS
END
```

EXAMPLE 2:
```
! 0 100 100 1
PITCH 100
STRING 5X7 10 0 LETTERS - 5X7
STRING 8X8 200 0 LETTERS - 8X8
STRING 9X12 10 10 LETTERS - 9X12
STRING 12X16 200 10 LETTERS - 12X16
STRING 18X23 10 28 LETTERS - 18X23
STRING 24X31 10 53 LETTERS - 24X31
END
```

TEXT

FUNCTION: Prints text on a label using compressed bitmapped fonts.

EXPLICIT FORM:
```
TEXT fontID(spacing,rotation,xmult,ymult) x y
characters
```

PARAMETERS:

`fontID` Specifies the font family and size. The font family loaded in most printers at present is CG Triumvirate, with font sizes as follows:

TABLE 14. FontID and Corresponding Point Size

fontID	0	1	2	3	4	5	6
Size (points)	6	8	10	16	24	36	48

NOTE: Not all printers support all fonts. Refer to Table 3 for more information.

`x y` Starting position of the character string. The point on the text block referenced by x y varies with rotation and justification. For left-justified text rotated 0 and 270 degrees, the reference point is the upper left corner of the text block. For left justified text rotated 90 and 180 degrees, the reference point is the lower left corner of the text block.

`characters` ASCII text to be printed.

The following parameters are contained in parentheses and are optional:

TABLE 15. Optional Parameters

Parameter	**Description**
spacing	Sets the spacing between characters. Valid entries are 0 to 255, and set the number of dots between proportionally spaced characters. If a negative sign is placed before the spacing value, the type will be spaced non-proportionally and the spacing value will set the character cell width.
rotation	Clockwise rotation of the printed character string. Valid entries are 0, 90, 180, and 270, with a default of 0.
xmult	Expands the font width. Valid entries are 0 to 4, with a default of 1. When using this parameter, you must also specify a value for ymult.
ymult	Expands the font height. Valid entries are 0 to 4, with a default of 1.

COMMENTS: The JUSTIFY command can left, center, or right justify text printed by the TEXT command.

TEXT fonts are stored in user-accessible memory in many printers, and are handled as stored objects. This means they may be deleted or replaced in the field, so the actual resident fonts may differ from those delivered in the printer. Take this into account when using the TEXT command.

SEE ALSO: The STRING, R90, R180, R270, JUSTIFY, and ULTRA_FONT commands.

EXAMPLE:

```
JUSTIFY CENTER
TEXT 3 410 0 TEXT FONT
TEXT 3(0,0,1,1) 410 40 TEXT FONT
TEXT 3(10,0,1,1) 410 80 TEXT FONT
TEXT 3(20,0,1,1) 410 120 TEXT FONT
TEXT 3(0,0,2,1) 410 160 TEXT FONT
TEXT 3(0,0,1,2) 410 200 TEXT FONT
```

TIME

FUNCTION: Sets or gets data from the real-time clock in printers so equipped. Data derived from the clock resides in the system variable TIME.

EXPLICIT FORM: `TIME action`

PARAMETERS: **`action`** = clock operation. Allowable operations are **`SET`**, **`GET`**, and **`ADD`**, described below.

`TIME SET` sets the clock to the numeric time value following the command. Specify the new clock time in the format YYYY MM DD hh mm ss, where:

TABLE 16. `TIME SET` Variables

Variable	**Description**
YYYY =	Year. Allowable values are 1970 to 2069.
MM =	Month. Allowable values are 01 to 12.
DD =	Day within the month. Allowable values are 01 to 31, as appropriate for the specified month (for example, a DD value of 31 is valid for month 07 but not month 06).

TABLE 16. **TIME SET** Variables

Variable	Description
hh =	Hour. Allowable values are 00 to 23.
mm =	Minute. Allowable values are 00 to 59.
ss =	Second. Allowable values are 00 to 59.

TIME GET returns the current clock time, placing it in an system variable called **TIME**.

TIME ADD adds the specified interval to the last time read by **TIME GET** (that is, to the value currently stored in the variable **TIME**). The specified time interval must be in the form described under **TIME SET**, and you must specify all six parameters. Added intervals can be negative or positive. Addition starts at the seconds field, and added values arithmetically carry or borrow when a field over- or underflows.

NOTE: Not all printers support the **TIME** command, and not all printers that support the command have a real-time clock (this is a hardware option). Without a real-time clock, the **TIME** variable will remain at zero or in an undefined state. **TIME ADD** will change the value stored in **TIME**, but the results are unpredictable if there is no installed clock.

COMMENTS: You access the data provided by the clock using the system variable **TIME**. This variable has formatting parameters, allowing you to extract time data in a variety of forms. Format the variable output by placing a format string after the variable name and bounding the variable and its format string with the variable delimiter (see **DELIMIT**).

NOTE: **TIME** will not return a value unless it is followed by formatting characters and bounded by the variable delimiter.

Here are the allowable formatting characters you can place after the **TIME** variable:

TABLE 17. Time Formatting Characters

Parameter	Returns
%%	% character
%a	Abbreviated weekday name (Sun, Mon, Tue)
%A	Full weekday name (Sunday, Monday, Tuesday)
%b	Abbreviated month name (Jan, Feb, Mar)
%B	Full month name (January, February, March)
%c	Date and time using a two-digit year (mm/dd/yy hh:mm:ss; for example, 09/25/98 15:35:20)
%C	Date and time using a four-digit year (mm/dd/yyyy hh:mm:ss; for example, 09/25/1998 15:35:20)
%d	Two digit day of month (1 -31)
%H	Two digit hour, 24 hour clock (0 - 23)
%I	Two digit hour, 12 hour clock (0 - 12)
%j	Three digit day of year (001 - 366)
%m	Two digit month (1 - 12)
%M	Two digit minute (00 - 59)
%p	AM or PM
%S	Two digit second (00 - 59)
%U	Two digit week number, where Sunday is the first day of the week (00 - 53)
%w	Weekday where 0 is Sunday (0 - 6)
%W	Two digit week number, where Monday is the first day of the week (00 - 53)
%x	Date, with two-digit year (mm/dd/yy; for example, 05/21/98)
%X	Time (hh:mm:ss; for example, 21:45:30)
%y	Two-digit year without century (00 to 99)
%Y	Year with century

You can use multiple formatting characters and mix printable characters with the formatting characters, to present the time or date in unusual ways.

SEE ALSO: **DELIMIT**

EXAMPLE 1: The following example will set the printer's clock to June 17, 2001, 3:30:00 PM:

```
! 0 0 0 0
```

```
TIME SET 2001 06 17 15 30 00
END
```

EXAMPLE 2: The following label format will read the current time from the clock, placing the value in the variable TIME. It will then print a label showing that time, add 4 days, 6 hours, and 17 minutes to it, and print the result:

```
! 0 100 500 1
DELIMIT ~
TIME GET
TEXT 2 20 50 The current date and time are ~TIME %c~
TIME ADD 0000 00 04 06 17 00
TEXT 2 20 100 In 4 days, 6 hours, 17 minutes it will be
~TIME %c~
END
```

EXAMPLE 3: The following command line will print the current date as stored in the TIME variable, in the form mm/dd/yy:

```
TEXT 2 20 100 The date is: ~TIME %m/%d/%y~
```

ULTRA_FONT

FUNCTION: fPrints text on a label, using Ultra Fonts (stroke fonts). Unlike STRING fonts, Ultra Fonts maintain their shape in any size and boldness. Font modifiers can change character bolding, spacing, and rotation. Font type C may also be italicized and "grayed."

EXPLICIT FORM: `ULTRA_FONT Tnnn IGz(bold,space,rot) x y char`

PARAMETERS:

T Font type, A, B, or C. Type A characters have rounded corners. Type B characters have angled corners. Type C resembles Helvetica print.

NOTE: Always use uppercase characters (A, B, or C, not a, b, or c) when specifying the font type. Not all printers support all fonts. See Table 3 for more information.

nnn Font size, in dots. This immediately follows the font type, and may specify both X and Y dimensions (A25X50, for example), or just the Y dimension, in which case X is approximately one-half Y (as in A50). For font types A and B, X and Y can range from 1 to 65535. Font C size can range from 40 to 700.

x y Starting position of the printed character string. The reference point on the text block varies with character rotation. For 0 and 270 degree rotation, the X and Y starting position of the text block is in the upper left corner of the block. For 90 and 180 degree rotation, the X and Y starting position of the string is in the lower left corner of the text block.

char ASCII characters to be printed.

The following parameters are contained in parentheses and are optional:

TABLE 18. ULTRA Optional Parameters

Parameter	Description
I	Specifies italic type (use only with font type C).
G	Specifies gray scale mode, and is followed immediately by the z modifier. The z modifier specifies the gray scale screen, and may be 0, 1, or 2, with 0 the lightest gray and 2 the darkest. The default screen is 2. These modifiers only work with font type C.
bold	Printed character boldness (width of character strokes) in dots. Allowable range is 1 (minimum stroke width) to 255 (maximum stroke width), with a default value of 2. A boldness of 50 at 200 DPI pitch will give a line thickness of 50/200, or 1/4". For font types A and B, boldness specifies thickness for all strokes. Font type C only bolds horizontally.
space	Number of dots between each printed character. Valid entries are 0 to 255 and N, with a default of 1. N sets non-proportional character spacing; that is, each character has the same width. Font type C does not support non-proportional spacing.
rot	Clockwise rotation of the printed characters. Valid entries are 0, 90, 180, and 270. The default is 0.

SEE ALSO: JUSTIFY, STRING, and ROTATE

EXAMPLE:
```
JUSTIFY CENTER
ULTRA_FONT A70X160 (20,5,90) 620 300 ULTRA
ULTRA_FONT A70X160 (20,5,90) 420 300 FONTS
JUSTIFY LEFT
ULTRA_FONT A50 20 30 A50 FONT
ULTRA_FONT B50 20 90 B50 FONT
ULTRA_FONT A100 20 150 A100 FONT
ULTRA_FONT B100 20 250 B100 FONT
JUSTIFY CENTER
U A30 (4,0,0) 220 360 Ultra fonts can be rotated,
U A30 (4,0,0) 220 400 scaled, and bolded to
U A30 (4,0,0) 220 440 produce a wide range of
U A30 (4,0,0) 220 480 unique typefaces.
```

Universal Clear

FUNCTION: Resets the printer to its initial power on state, effectively the same as cycling printer power.

EXPLICIT FORM: `(ASCII):23 23 23 23 23 67 76 69 65 82 23 23 23 23 23`

PARAMETERS: None

COMMENTS: Unlike other commands, you cannot send the Universal Clear command to the printer as a printable ASCII string of letters and/or numbers. Most computer keyboards do not have a character corresponding to ASCII 23.

Although you cannot type the command directly, the host computer can send the command if it is programmed in a high-level language. For example, you could send Universal Clear to the printer using the following BASIC code:

```
LPRINT CHR$(23); CHR$(23);CHR$(23); CHR$(23); CHR$(23);
"CLEAR"; CHR$(23); CHR$(23); CHR$(23); CHR$(23);
CHR$(23); CHR$(13); CHR$(10)
```

NOTE: Universal Clear clears unprocessed data from memory. Do not use this command while the printer is busy printing or risk data loss.

Wake-up String

FUNCTION: Switches the printer from idle to active mode in preparation for incoming data.

EXPLICIT FORM: CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

PARAMETERS: none

COMMENTS: The 4.25" portable printers use an energy-saving "sleep mode" to save battery power during periods of inactivity. These printers won't accept data when they are asleep.

Code Courier printers go to sleep immediately after finishing the current task. Pressing the FEED button will feed a label, after which the printer will fall asleep again. Therefore, the wake-up string must precede every label format.

The minimum wake-up string length depends on the serial port baud rate and printer handshaking. When using RTS/CTS handshaking, use 40 upper-case C's. When using XON/XOFF handshaking, use a wake-up string at least as long as shown in the following table.

TABLE 19. Baud Rate and Wake-up String Length

Baud Rate	String Length
600	5
1200	10
2400	20
4800	40
9600	80
14400	120
19200	155
38400	310
57600	470

NOTE: Aside from a very small amount of added data transmission time, the presence of a wake-up string does not adversely affect the behavior of printers that do not require one. You may want to send a wake-up string to every printer prior to sending your label formats, to help assure compatibility with all printers.

SEE ALSO: Header line.

WIDTH

FUNCTION: Sets the width of the printed label. Typically, this command is for printing on label stock that is narrower than the printhead.

EXAMPLE: `WIDTH nnn`

PARAMETERS: nnn = approximate print width in hundredths of an inch.

COMMENTS: Since data is handled in whole words, any value specified for the WIDTH results in an actual print width that can be expressed in 16 bits. The printer will round up any width value to the nearest whole word. This means that when using a print pitch of 200, the width value will be rounded up to the nearest number that is evenly divisible by 8. When using a print pitch of 100, the width value will be rounded up to the nearest number evenly divisible by 16.

If you do not use the WIDTH command, the printer will default to its maximum print width unless the default width has been changed by a VARIABLE WIDTH command.

Reducing width lets you program longer labels in a given memory area. It can also simplify graphics programming by reducing the number of dot columns you need to program.

When reducing width, Code Courier printers move the label against the left edge of the label stock as it exits the printer. Barcode Blaster, Blaster Advantage, and Solus printers move the label against the right edge of the label as it exits the printer.

NOTE: The WIDTH command is necessary with the Del Sol to correctly position the printing image.
The WIDTH command line must precede any command that maps label components on the printer's memory grid, such as STRING or BARCODE.

High speed Barcode Blasters automatically disable high speed printing when they encounter this command in a label format.

SEE ALSO: The VARIABLE WIDTH command.

EXAMPLE:

```
! 0 100 100 1
STRING 8X8 0 0 THIS IS
STRING 8X8 0 10 WITHOUT
STRING 8X8 0 20 THE WIDTH
STRING 8X8 0 30 COMMAND
END
! 0 100 100 1
WIDTH 80
STRING 8X8 0 0 THIS IS
STRING 8X8 0 10 WITH
STRING 8X8 0 20 THE WIDTH
STRING 8X8 0 30 COMMAND
END
```

CHAPTER 4

Storing Data in the Printer Memory

These commands let you store label formats and graphics in the printer's memory for later use. Internal data storage can help improve label throughput and may also simplify programming in some applications.

NOTE: Internal data storage capability is an optional feature. Not all printers accept these commands, and due to hardware limitations, some printers that accept the commands may not successfully store data. Refer to Table 1 for more information.

Before Using Data Storage Commands

Before using internal data storage, please note that:

1. Changing the Text or Overflow Buffer sizes will delete objects stored in the image buffer. This will not affect normal stored formats or graphics, but will affect enhanced stored formats.
2. Storing objects will leave unknown images in memory. Normal label printing will clear these images, but if you use background mode or the !+ header line mode you must clear the spurious images from memory first. (See Header line.)

3. Storage commands let you specify what memory area will receive the stored object. The memory area is specified as parameter xx, with the following memory areas available:

TABLE 20. Printer Memory Areas

	Memory Area
0	Volatile RAM
1	Reserved[a]
2	Expanded memory
3	Nonvolatile RAM

a. This area reserved for future firmware releases.

NOTE: Cognitive does not recommend using volatile RAM for stored objects, since objects stored there will be lost when the printer is turned off. However, you may want to use volatile RAM to store variables or objects that will change frequently.

When using volatile RAM for object storage, you must allocate space for that purpose. Refer to the VARIABLE ALLOCATE command for more information.

4. You must assign an alphanumeric identifier to every stored object. The identifier can be from one to eight characters long, using any combination of letters and numbers (no punctuation). If an object in memory already uses the specified identifier, the printer deletes the existing object before storing the new one, even if the two objects would reside in different memory areas.

   Also notice that some of these commands are used in place of the normal label format header line, rather than as ASCII commands within a label format. Commands starting with an exclamation point (!) should be treated as header line replacements.

Data Storage Commands

Data storage commands discussed in this guide include:

- `Delete Stored Object`
- `Format Recall`
- `Format Store`
- `GRAPHIC STORE`
- `GRAPHIC RECALL`
- `Initialize Storage`
- `List Stored Objects`
- `DELIMIT`
- `DEFINE_VAR`
- `Call Menu`
- `Call Variable`

Delete Stored Object

FUNCTION: Deletes a stored object from memory.

NOTE: This command replaces the normal header line. Do not use any other commands, including the END command, with this command.

EXPLICIT FORM: `!D Identifier`

PARAMETERS: Identifier = stored object identifier assigned when the object was first stored.

COMMENTS:

EXAMPLE: `The following command will search all available memory areas for a stored object named FORM_1, and then delete the object from memory.`

```
!D FORM_1
```

Format Recall

FUNCTION: Recalls a stored format from memory, merges any incoming variable data with the stored data, and prints labels as required.

NOTE: This command replaces the normal header line. Do not use any other commands, including the END command, with this command.

EXPLICIT FORM: `!R t Identifier x 100 maxY numlabels`

PARAMETERS: t = data type: F indicates standard ASCII format files. E indicates enhanced format files.
Identifier = object identifier assigned when the object was stored.

NOTE: When recalling stored formats, use the same value for t as was used when the format was stored. Specifying the wrong data type when recalling a stored format may cause the label to print incorrectly.

COMMENTS: If variable data is needed to print the finished label, the printer will expect the required data to follow the Format Recall command. A carriage return or carriage return and line feed must precede the variable data.

SEE ALSO: See Header line for other parameter details.

EXAMPLE: The following example assumes that the label format FORM_1 shown in the Format Store example is already stored in the printer.

```
!R F FORM_1 0 100 170 1
MEN'S DRESS SHIRT
$17.85
SHIRT/$17.85
```

Format Store

FUNCTION: Stores a label format in a specified memory area. The stored format can be recalled and printed with the Format Recall command.

There are two general types of stored formats: standard and enhanced. The printer stores standard formats as ASCII data only. Enhanced stored formats are stored as fully mapped images in the image buffer, as well as ASCII data in the specified memory area. This improves label throughput if the stored data plots a very complex label.

EXPLICIT FORM: `!Sd xx tnnn Identifier`

PARAMETERS: d = variable field delimiter.

This can be any printable character except a carriage return or line feed. Within the label format that follows, the data between the delimiter character and the end of the line will not be printed with the rest of the label. Instead, it is sent to the printer's serial port to prompt the user for input when the label is recalled with the FORMAT RECALL command.

If user input is not desired, the data used to fill the fields delimited by d can be supplied with the FORMAT RECALL command. The printer simply fills the fields with the data as it is received.

Placing two variable field delimiters in a row creates a repeating field; i.e., will cause the printer to use the variable data from the previous variable field in the current field. This feature may be useful if you want to print the same data using two different fonts, or want to print the data as a rotated bar code and nonrotated text.

NOTE: Do try to make the first field in a stored format a repeating field (since there will be no data to repeat). Do not place any text after the variable field delimiter when creating a repeating field.

When using enhanced format storage, the variable field delimiter also separates the fixed data commands from the variable data commands. Put all commands that define variable fields at the end of the data file. Precede all of these commands with a single line that only contains the variable field delimiter.

xx = numeric identifier for the desired memory storage area.

t = data type: F indicates standard ASCII format files. E indicates enhanced format files. If storing an enhanced format, the E must be followed by the nnn parameter.

nnn = Number of dot rows the stored format will plot. Only use this parameter when programming an enhanced stored format; that is, with the E data type.

NOTE: Determine nnn using the number of dot rows actually occupied by the stored object, rather than from the number of dot rows available on the physical label. Memory space reserved by nnn is unavailable for other uses. Specifying too large a value wastes image buffer space.

Storing multiple enhanced formats can quickly use up the image buffer. Enhanced formats will not be stored if the total space occupied by all enhanced stored objects exceeds the total image buffer space.

Identifier = Alphanumeric identifier for the stored data. Must be eight or fewer printable characters with no spaces.

COMMENTS: This command replaces the normal label format header line. Normal header line parameters, like dot time, dot rows, and number of labels, are provided with the Format Recall command. The Format Recall command cannot have any other commands with it, so almost everything necessary to define the label must accompany the Format Store command.

The GRAPHIC RECALL command is allowed within stored formats. You can put stored graphics within stored ASCII formats (normal or enhanced) by using GRAPHIC STORE to store the graphic in the usual manner, then using GRAPHIC RECALL within the stored format.

Use comments sparingly or not at all within stored formats, since comments use one byte of memory space per character and do not affect label printing.

SEE ALSO: Format Recall command.

EXAMPLE 1: The following commands store the format that prints the label in the Format Recall example. Data is still needed for the description, price, and combined item/price. The printer will send prompts for this data to the serial port at print time. This is a standard stored format; notice that commands that define fixed data are freely mixed with commands that define variable data.

```
!S~ 3 F FORM_1
WIDTH 224
JUSTIFY CENTER
ULTRA_FONT A24 (3,2,0) 224 0 ~DESCRIPTION>
STRING 18X23 29 30 SALE PRICE:
JUSTIFY LEFT
ULTRA_FONT A50 (5,2,0) 240 30 ~PRICE>
BARCODE CODE39X 20 150 60 ~ITEM/PRICE>
END
```

EXAMPLE 2: This file stores the same label using enhanced format storage. Notice that all commands specifying variable data are grouped at the end of the file, separated from the fixed data commands by the delimiter character (the \ character in this case). Do not send prompts to the printer when using enhanced format storage.

```
!S\ 3 E220 EFORM_1
WIDTH 224
JUSTIFY CENTER
STRING 18X23 29 30 SALE PRICE:
\
ULTRA_FONT A24 (3,2,0) 224 0 \
JUSTIFY LEFT
ULTRA_FONT A50 (5,2,0) 240 30 \
BARCODE CODE39X 20 150 60 \
END
```

GRAPHIC STORE

FUNCTION: Stores the graphics file following the command in the specified memory area.

NOTE: Use this command with a dummy header line, and do not use an END command or any other commands with it. The graphics data to be stored must immediately follow the GRAPHIC STORE command.

EXPLICIT FORM: `GRAPHIC STORE Type xx Identifier`

PARAMETERS: Type = graphic file type. Allowable types are PCX, BMP, and LOGO (proprietary CSI graphics format).

xx = numeric identifier for the desired memory storage area.

Identifier = Alphanumeric identifier for the stored graphic. The identifier can be up to eight characters long, and can consist of any printable characters. Spaces within the identifier are not allowed.

COMMENTS: The printer waits for graphics data after it receives the GRAPHIC STORE command. Incoming data following the GRAPHIC STORE command is stored in the specified memory area. The printer automatically reverts to normal data input processing when it sees the end of the graphics file.

Use the GRAPHIC RECALL command to recall and print the graphic on labels. The printer only prints black-and-white graphics. Also, graphics are printed full-scale, with each image dot corresponding to one printed dot.

SEE ALSO: GRAPHIC RECALL command.

EXAMPLE: `The following commands will store following graphics data in nonvolatile RAM as IMAGE_1.`

```
! 0 0 0 0
GRAPHIC STORE BMP 3 IMAGE_1
```

GRAPHIC RECALL

FUNCTION: Recalls a stored graphic from memory and prints labels as required.

EXPLICIT FORM: `GRAPHIC RECALL Identifier x y`

PARAMETERS: Identifier = object identifier assigned when the graphic was first stored.

x y = Starting position of the printed graphic; normally its upper-left corner. (The JUSTIFY command can position the graphic right, left, or center of the specified coordinates.)

COMMENTS:

SEE ALSO: GRAPHIC STORE command.

EXAMPLE: `The following label format will print a graphic stored as IMAGE_1 at location 30, 10:`

```
! 0 100 500 1
GRAPHIC RECALL IMAGE_1 30 10
END
```

Initialize Storage

FUNCTION: Clears all stored objects from the specified memory area in preparation for new data.

NOTE: This command replaces the normal header line. Do not use any other commands, including the END command, with this command.

EXPLICIT FORM: `!I xx`

PARAMETERS: xx = numeric identifier for the memory area to be cleared.

EXAMPLE: `The following command will clear nonvolatile RAM of all stored objects:`

```
!I 3
```

NOTE: STRING fonts and TEXT fonts are stored objects in most printers that use flash memory (that is, most printers that actually support stored objects). As shipped from the factory, these fonts are stored in memory area 3 (nonvolatile RAM). Initializing memory area 3 will remove STRING fonts and TEXT fonts from memory.

List Stored Objects

FUNCTION: Scans all memory areas for stored objects (label formats and graphics), then prints a list of all the objects, their size in bytes, their memory location, and the amount of available memory in each memory area.

EXPLICIT FORM: `!L`

NOTE: This command replaces the normal header line. Do not use any other commands, including the END command, with this command.

PARAMETERS: `!LS` Lists the stored objects and prints them to the serial port.

EXAMPLE: `!L`

DELIMIT

FUNCTION: Specifies the delimiter used to isolate variables within command lines.

EXPLICIT FORM: `DELIMIT c`

PARAMETERS: c = Delimiter character.

This is a single ASCII character, and must have an ASCII value greater than decimal 32 (i.e., no control characters or spaces are allowed). You must declare the delimiter if you use variables. There is no default delimiter character.

COMMENTS: Define the delimiter character before using any variables in a label format. The delimiter character must precede and follow every variable and its associated parameters.

NOTE: Choose the delimiter character carefully. Avoid using a character that you will need for other purposes within the label format. Especially avoid using the the % sign as a delimiter. It is used within some system variables, making its use as a delimiter inconvenient in many applications.

SEE ALSO: DEFINE_VAR command.

EXAMPLE: `The following label format declares the @ sign as the variable delimiter, then uses the system variable TIME to print the current time:`

```
! 0 100 590 1
DELIMIT @
TIME GET
TEXT 2 20 The time is @TIME %X@
END
```

DEFINE_VAR

FUNCTION: Defines a variable for use within printer commands in a label format. When the printer encounters a variable in a command line, it replaces the variable name with entered or stored data. It can also (optionally) send a prompt for data to the user via the serial port.

EXPLICIT FORM: `DEFINE_VAR location id length type range alignpad "prompt" "initial"`

PARAMETERS:

`location`

Memory location to store the defined variable. The following memory areas are avialable:

TABLE 21. Printer Memory Areas

	Memory Area
0	Volatile RAM
1	Reserved[a]
2	Expanded memory
3	Nonvolatile RAM

a. This area reserved for future firmware releases.

NOTE: Use the VARIABLE ALLOCATE command to reserve space in memory area 0 (volatile RAM).

`id`
The variable name. This must be eight or fewer ASCII alphanumeric characters, with the first character an uppercase or lowercase letter (A - Z or a - z).

NOTE: The printer has several predefined variables. You may not use these variable names when defining variables.

`length`
The maximum number of characters that will be accepted as variable data. Allowable values are 1 to 32767, but available printer memory may impose a lower practical limit on this parameter. The DEFINE_VAR command also has a maximum line length of 254 characters, which will limit the variable length if it must be initialized.

`type`
Variable type. Available types are:

TABLE 22. Variable Types

Variable	Description
P	Protected: The printer will not prompt the user to change the variable value. The value can only be changed by a Call Variable command.
C	Counter: A protected variable that can be changed by the ADJUST command. Store counter variables in memory areas 2 or 3 only.

TABLE 22. Variable Types

Variable	Description
D	Dynamic: The printer will prompt the user for a value at print time, and will retain the entered value for future use. Store dynamic variables in memory area 0 only.
T	Temporary: The printer will prompt the user for a value at print time and erase the value after printing the label format. Store temporary variables in memory area 0 only.

`range`
Input range specifier. Available specifiers are:

TABLE 23. Input Range Specifiers

Variable	Description
A	Only allows alphabetic characters.
N	Only allows alphabetic and numeric characters (no punctuation).
X	Allows any character.
#	Allows signed or unsigned whole numbers (no decimal point).

You can specify a range for alphabetic or numeric variables by adding minimum and maximum values within double quotes. For example, # "100" "199" will allow numeric values between 100 and 199 inclusive. The range specifier A "BOB" "DUN" will allow BOG, DUD, or COG, but will not allow JAG. The range specifiers must both have the same number of digits or characters. Range checking is character-by-character, based on each character's ASCII value.

`align`
Specifies the position of the variable data within the specified length. Allowable values are:

TABLE 24. Align Specifiers

Variable	Description
L	Places the data at the left end of the space specified by length parameter, filling the remaining space to the right with the character specified by pad.
R	Places the data at the right end of the space specified by length parameter, filling the remaining space to the left with the character specified by pad.

TABLE 24. Align Specifiers

Variable	Description
C	Places the data in the center of the space specified by length parameter, filling the remaining space on both sides with the character specified by pad.
N	Tells the printer to ignore the length parameter so the variable data only occupies the space it actually needs.

`pad`
Specifes the character that will be used to fill any space allocated by the length parameter that is not occupied by actual variable data. This is an optional parameter, and immediately follows the align parameter with no intervening spaces. If pad is omitted, the pad character will be the space character (ASCII 20).

`"prompt"`
ASCII text that the printer sends to the serial port when ready to receive data for the defined variable. Enclose the prompt text with quotation marks ("). If no prompt is desired, enter two quotation marks ("") with no characters between them.

To include quotation marks within the prompt, precede the the internal quotation mark with a backslash (\). Use two consecutive backslashes (\\) to include a backslash within the prompt.

`"initial"`
Starting value for the defined variable. Enclose the starting value with quotation marks (") as described for "prompt" above.

COMMENTS: Replacing a fixed value in a printer command with a variable tells the printer to plot the label element using data entered or recalled at print time, rather than data written in the label format when it was programmed.

When using a variable name within a label format, you must enclose it with the delimiter character defined by the DELIMIT command.

SEE ALSO: sCall Variable and DELIMIT commands.

EXAMPLE: The following command defines a dynamic variable called customer, with a maximum length of 20, alphabetic type, no alignment, no initial value.

```
DEFINE_VAR 0 customer 20 A N "Enter customer:" ""
```

Call Menu

FUNCTION: Recalls a stored menu from memory and initiates its execution.

EXPLICIT FORM: `!R M menuname`

NOTE: This command replaces the normal header line. Do not use any other commands, including the END command, with this command. Also, take care that no extraneous control characters follow the command; the printer may interpret them as menu control characters.

PARAMETERS: menuname = the alphanumeric identifier under which the menu was originally stored. This identifier can be up to eight characters long.

COMMENTS: Called menus remain active until the user manually cancels their action or until the printer processes a MENU EXIT command.

SEE ALSO: MENU START, MENU EXIT, and MENU CONTROL commands.

EXAMPLE: `The following label format will start execution of a menu called MAIN:`

```
!R M MAIN
```

Call Variable

FUNCTION: Recalls a stored variable for user input. The printer will send the associated prompt to the serial port and await user input when it encounters this command.

EXPLICIT FORM: `!R V identity`

NOTE: This command replaces the normal header line. Do not use any other commands, including the END command, with this command.

PARAMETERS: identity = the alphanumeric identifier under which the variable was originally stored.

COMMENTS: Use this command within menus to change the value of protected variables.

NOTE: You must define a prompt for the variable in the DEFINE_VAR command in order for a prompt to appear when executing the Call Variable command.

SEE ALSO: The DEFINE_VAR and DELIMIT commands.

EXAMPLE: `The following command will prompt the user to enter a value for a variable called "password":`

```
!R V password
```

CHAPTER 5

Menu Commands

A menu lets the user control the printer at print time through a set of predefined choices. Menus are typically used when the printer is connected to a simple controlling device, such as a keyboard/display unit.

NOTE: Menu programming is inherently complex, and is only useful if the user must control the printer with a non-programmable device (such as a dumb terminal or keyboard). As a general rule, we do not recommend programming menus in the printer if there is processing power available in the host.

Menus are stored objects. Familiarize yourself with the proper use of stored objects before attempting to program menus.

Menu Operation

During menu execution, the printer sends text descriptions associated with menu items ("prompts") to the printer serial port and waits for user input. The user responds to the menu by making a selection, which the controller then sends back to the printer via the serial port.

Choose a menu item using one of the following methods:

1. By "scrolling" to the item and selecting it using keys as defined by the MENU CONTROL command.
2. By entering the number of a menu item. All menu items have a single-digit number associated with them. The printer assigns this number automatically, based upon the menu item location within the menu. Menu items that do not have a visible prompt are still selectable by item number.

The printer executes printer commands contained in the menu definition based on the user's selection. Each menu item can have one or many standard printer commands associted with it; thus, the printer can perform complex tasks in response to a menu selection. They can even call printer commands without any user-selectable menu items, behaving much like stored label formats. Menus can call other menus, and can also recall and print stored objects or data.

Menu Programming

Menus generally conform to the following structure:

```
! 0 0 0 0
MENU CONTROL cn nx pr sl
MENU START x menuname
MENU ITEM "item1"
MENU ACTION command parameters
MENU ACTION command parameters
MENU EXIT
MENU ITEM "item2"
MENU ACTION command parameters
MENU ITEM "item3"
MENU ACTION command parameters
...
MENU END
END
```

MENU CONTROL is an optional command that defines the four keyboard keys that scroll the menu, select menu items, and exit the menu. If used, it must appear before the MENU START command. MENU START signals the beginning of the menu, and MENU END signals the end of the menu. All commands bounded by MENU START and MENU END define menu prompts and actions.

MENU ACTIONs generally follow MENU ITEMs. If no MENU ITEMs are present, the menu will simply execute the MENU ACTIONs in sequence without input from the operator. The effect is similar to using a stored label format.

MENU EXIT commands will terminate execution of the current menu, at which time the printer will continue executing the remaining commands in the label format that called the menu.

Menus cannot be nested within other menus, but they can call other menus.

Menus are stored objects. They must be stored in memory before they are called by the Call Menu command. They follow all of the usual rules for stored objects.

- `MENU ACTION`
- `MENU CONTROL`
- `MENU END`
- `MENU EXIT`
- `MENU ITEM`
- `MENU START`
- `Call Menu`

NOTE: There is no underscore between MENU and the sub-command name. For example, MENU ACTION is correct; MENU_ACTION is incorrect.

Several other commands are especially useful when programming menus. These include:

- `QUANTITY` on page 42
- `List Stored Objects` on page 70
- `DELIMIT` on page 70
- `DEFINE_VAR` on page 71
- `Call Variable` on page 75
- `VARIABLE ALLOCATE` on page 91

For further information about these menu commands, refer to `Standard Printer Commands` on page 11.

MENU ACTION

FUNCTION: Specifies one or more commands that the printer will execute when the user selects the associated MENU ITEM. Any legitimate printer commands are allowed (including the Call Menu command). The printer commands and associated parameters embedded within the MENU ACTION command must be bounded by quotation marks.

EXPLICIT FORM: `MENU ACTION "command parameters"`

PARAMETERS: "command = Any legitimate printer command

parameters" = The parameters required by command

COMMENTS: Quotation marks (") must precede and follow the command and its parameters. If you want to include quoted text within the command, precede each quotation mark within the command with a backslash (\). To include a backslash within the command, use two backslashes (\\). Indicate the end of the command line with \n. This is the character substitution for line feed. To send a carriage return without line feed, use \r.

You can place any number of MENU ACTION commands after one MENU ITEM command. The printer will process all MENU ACTIONs that follow the selected MENU ITEM until it encounters the next MENU ITEM. Thus, the printer can execute complex operations using multiple MENU ACTIONs, but the most common use of the command is to call another menu or recall a stored label format.

If the printer finds a format header line within a MENU ACTION, it will expect all commands associated with that header line, up to and including the END command, to immediately follow within the same MENU ACTION command. You cannot split a label format over multiple MENU ACTION commands.

Menus can contain MENU ACTION commands without any MENU ITEM commands. The printer will execute commands within such menus without operator intervention.

SEE ALSO: The MENU ITEM and Call Menu commands and Data Storage commands.

EXAMPLE 1: The following calls a menu named MAIN when the user selects its associated menu item. When the printer finishes MAIN, control returns to the next menu command in the calling menu:

```
MENU ACTION "!R M MAIN \n"
```

EXAMPLE 2: The following recalls and print one copy of a stored label format called LBL_ONE when the user selects its associated menu item:

```
MENU ACTION "!R F LBL_ONE 0 100 570 1\n"
```

EXAMPLE 3: The following label format stores a menu called MENU_ONE. Each MENU ACTION command will print a different label:

```
! 0 0 0 0
MENU START 3 MENU_ONE
MENU ITEM "Print label 1 \r"
MENU ACTION "! 100 90 1\nW 224\n U A24 20 20 Label
1\nE\n"
MENU ITEM "Print label 2 \r"
MENU ACTION "! 100 90 1\nW 224\n U A24 20 20 Label
2\nE\n"
MENU ITEM "Print label 3 \r"
MENU ACTION "! 100 90 1\nW 224\n U A24 20 20 Label
3\nE\n"
MENU END
END
```

MENU CONTROL

FUNCTION: Specifies the characters used to exit, "scroll," or select an item from a menu. In most applications, these characters are chosen by pressing the four keys on the external keyboard that correspond to keyboard functions "cancel," next," "previous," and "select."

EXPLICIT FORM: `MENU CONTROL cn nx pr se`

PARAMETERS: cn =The decimal ASCII value of the character used to cancel the current action. The default value for cn is 27 (the ESC key).

nx =The decimal ASCII value of the character used to scroll to the next menu item. The default value for nx is 78 (the N key).

pr =The decimal ASCII value of the character used to scroll to the previous menu item. The default value for pr is 80 (the P key).

se =The decimal ASCII value of the character used to select the current menu item. The default value for se is 13 (the ENTER key).

COMMENTS: Menu items will only appear on the display if you specify a prompt string in the MENU ITEM command.

The printer stores the control characters specified by the MENU CONTROL command in a menu called _MENUC_.

SEE ALSO: The MENU ITEM command.

EXAMPLE: `The following command tells the printer to use the "q", period, comma, and "a" keys for the cancel, next, previous, and select functions:`

```
MENU CONTROL 113 46 44 97
```

MENU END

FUNCTION: Signals the end of the menu definition. The printer will store the menu defined by the commands between MENU START and MENU END under the identifier specified in MENU START.

NOTE: This command does not terminate menu execution. Use MENU EXIT for that.

EXPLICIT FORM: `MENU END`

PARAMETERS: None.

COMMENTS: The printer interprets all commands between MENU START and MENU END as part of the menu.

SEE ALSO: The MENU START, MENU ITEM, MENU EXIT, MENU ACTION commands.

EXAMPLE: `MENU END`

MENU EXIT

FUNCTION: Tells the printer to terminate menu processing. The printer will then process any remaining commands in the label format that called the menu.

EXPLICIT FORM: `MENU EXIT`

PARAMETERS: None.

COMMENTS: The printer will only interpret MENU EXIT if it is between the MENU START and MENU END commands.

SEE ALSO: The MENU START, MENU END, and Call Menu commands.

EXAMPLE: `MENU EXIT`

MENU ITEM

FUNCTION: Marks the beginning of a sequence of commands that will execute when the user selects the item from a programmed printer menu, and (optionally) defines text the printer will display when offering the menu to the user. The printer will send this text to the serial port when processing the stored menu, allowing the user to select the item either by item number or by menu scrolling.

EXPLICIT FORM: `MENU ITEM "prompt"`

PARAMETERS: "prompt" = ASCII text that the printer will send to the serial port when the printer encounters the MENU ITEM command. Any printable ASCII character is allowed. Begin and end the text with quotation marks ("). If you wish to include quoted text within the prompt, precede each quotation mark within the prompt with a backslash (\). To include a backslash within the prompt, use two backslashes (\\).

If no prompt is desired, enter two sets of quotation marks ("") with no text or spaces between them. The menu item will still exist and may be called by number (see below), but no prompt or space will appear for the item when the printer processes the MENU ITEM command.

COMMENTS: MENU ITEM commands are only allowed between MENU START and MENU END commands (that is, only within menus). Up to ten menu items are allowed in each menu. The printer assigns a number to every menu item, with item 1 being the first item in the menu, item 2 the second item, etc. The tenth item is item 0.

One or more MENU ACTION commands must follow every MENU ITEM command. MENU ACTION commands tell the printer what to do when the user chooses the menu item. The printer will process the MENU ACTIONs that follow the selected MENU ITEM up to the next MENU ITEM. It will then wait for further user input.

SEE ALSO: The MENU START, MENU ACTION, MENU END commands.

EXAMPLE: The following commands define a simple menu called PRT_LBL to select and print one of two stored label formats:

```
! 0 0 0 0
MENU START 3 PRT_LBL
MENU ITEM "print return address label"
MENU ACTION "!R F RTNLBL 0 100 570 1\n"
MENU ITEM "print shipping label"
MENU ACTION "!R F SHPLBL 0 100 570 1\n"
MENU END
END
```

MENU START

FUNCTION: Signals the beginning of a menu definition and specifies the menu storage location and identifier.

NOTE: This command does not execute the menu, it only prepares the printer to receive the menu for storage. Use Call Menu to execute stored menus.

EXPLICIT FORM: `MENU START x menuname`

PARAMETERS: x = menu storage location. You can store menus in the following memory areas:

TABLE 25. Printer Memory Areas

	Memory Area
0	Volatile RAM
1	Reserved[a]
2	Expanded memory
3	Nonvolatile RAM

a. This area reserved for future firmware releases.

menuname = menu identifier. The menu identifier can be up to eight alphanumeric characters, and the first character must be an uppercase or lowercase letter (A-Z or a-z).

The menu name __BOOT__ (the word BOOT preceded and followed by two underscore characters) is reserved for an autostart routine. Use this menu name only if you want the printer to execute the menu automatically on power-up. The printer will always try to execute __BOOT__ when it is powered on or reset, provided that the printhead is closed.

NOTE: To circumvent a programmed __BOOT__ menu, reset or apply power to the printer with the printhead open.

The menu name _MENUC_ is reserved for storage of the MENU CONTROL characters. Do not use this name for your own menu.

COMMENTS: The printer interprets all commands between MENU START and MENU END as part of the menu.

SEE ALSO: The Call Menu and MENU END commands.

EXAMPLE: The following command marks the beginning of a menu called MAIN, which will be stored in nonvolatile RAM (memory area 3):

```
MENU START 3 MAIN
```

Call Menu

FUNCTION: Recalls a stored menu from memory and initiates its execution.

EXPLICIT FORM: `!R M menuname`

NOTE: This command replaces the normal header line. Do not use any other commands, including the END command, with this command. Also, take care that no extraneous control characters follow the command; the printer may interpret them as menu control characters.

PARAMETERS: menuname = the alphanumeric identifier under which the menu was originally stored. This identifier can be up to eight characters long.

COMMENTS: Called menus remain active until the user manually cancels their action or until the printer processes a MENU EXIT command.

SEE ALSO: The MENU START, MENU EXIT, and MENU CONTROL commands.

EXAMPLE: The following label format will start execution of a menu called MAIN:

```
!R M MAIN
```

CHAPTER 6

Variable Commands

VARIABLE commands let you change some of the printer's characteristics. These changes stay in effect until the printer is turned off or until they are changed by another VARIABLE command.

NOTE: Do not confuse VARIABLE commands (which control the printer) with variable values (which are used to represent data).

Variable Command Rules

It is important to follow a few fundamental rules when using the VARIABLE commands:

- Enter all VARIABLE commands in upper case letters.
- Place any VARIABLE commands immediately after the header line in the label format.
- Enter all VARIABLE commands exactly as shown in the command descriptions. VARIABLE may be abbreviated as V, but do not use any other abbreviations unless specifically allowed in the command description.

- As a general rule, you should not use VARIABLE commands in stored label formats. A few VARIABLE commands that control label appearance (for example, VARIABLE DARKNESS) are permissible in stored formats, but there is little advantage to using them there. It is better to use conventional label formats for printer setup.
- Use VARIABLE WRITE and VARIABLE COMM only in non-printing label formats. Some other VARIABLE commands will not work well except in non-printing formats. These cases are noted in the command descriptions.

NOTE: All VARIABLE commands, and especially the VARIABLE WRITE command, should be used with care since they can change the data in the printer's nonvolatile RAM.

You will probably need to use a few VARIABLE commands as a matter of routine, to set the printer up for various types of print media or print methods. We have prepared some sample label formats to cover these common requirements.

Variable Commands

Printer setup command list

- `VARIABLE ALLOCATE`
- `VARIABLE AUDIO_FREQ`
- `VARIABLE AUTOCUT`
- `VARIABLE BUFFER_TIMED_RESET`
- `VARIABLE COMM`
- `VARIABLE DARKNESS`
- `VARIABLE FEED_TYPE`
- `VARIABLE HIGHSPEED`
- `VARIABLE INDEX SETTING`
- `VARIABLE LOW_BATTERY_IGNORE`
- `VARIABLE LOWSPEED`
- `VARIABLE MEDIA_ADJUST`
- `VARIABLE MODE`
- `VARIABLE NO_MEDIA`
- `VARIABLE NORMAL`
- `VARIABLE ON_TIME`
- `VARIABLES ON/OFFVARIABLE PITCH`
- `VARIABLE POSITION`
- `VARIABLE PRESENTLABEL`
- `VARIABLE PRINT_MODE`
- `VARIABLE READ`
- `VARIABLE REPORT_LEVEL`
- `VARIABLES RESET`
- `VARIABLE SLEEP_AFTER`
- `VARIABLE SHIFT LEFT`
- `VARIABLE TXTBFR`
- `VARIABLE USER_FEEDBACK`
- `VARIABLE WIDTH`
- `VARIABLE WRITE`

VARIABLE ALLOCATE

FUNCTION: Reserves space in the image buffer for stored objects.

EXPLICIT FORM: `VARIABLE ALLOCATE nnn`

IMPLICIT FORM:

PARAMETERS: nnn = amount of memory reserved for stored objects, in whole kilobytes. The allowable range is 0 to 128.

COMMENTS: Flash RAM normally holds all stored objects (graphics and stored formats). The printer reserves all available standard RAM for the text buffer and image buffer. This may not be the best use of standard RAM in some applications. VARIABLE ALLOCATE lets the programmer to put stored objects in part of the image buffer.

When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

NOTE: Changing the memory allocation in the image buffer or text buffer will delete any objects stored in those areas.

EXAMPLE: `The label format below will reserve 4 kBytes in the image buffer for object storage:`

```
! 0 0 0 0
VARIABLE ALLOCATE 4
VARIABLE WRITE
END
```

VARIABLE AUDIO_FREQ

FUNCTION: Sets the audio frequency (pitch) of the printer's "beep" signal.

EXPLICIT FORM: `VARIABLE AUDIO_FREQ nnnnn`

PARAMETERS: nnnnn = speaker beep frequency, in Hertz. The default is 3000. Setting nnnnn to zero disables the speaker.

COMMENTS: Since the speaker frequency will rarely need changing, we recommend using this command with VARIABLE WRITE in a non-printing label format.

EXAMPLE: The label format below will set the beep frequency to 1000 Hz:

```
! 0 0 0 0
VARIABLE AUDIO_FREQ 1000
VARIABLE WRITE
END
```

VARIABLE AUTOCUT

FUNCTION: Enables or disables automatic label cutting in printers so equipped. With automatic label cutting enabled, the printer will cut the label after printing the last dot row. With automatic label cutting disabled, the printer will not cut the label unless there is a HALT command in the label format.

EXPLICIT FORM: `VARIABLE AUTOCUT status`

PARAMETERS: status = automatic cutter status. ON enables automatic label cutting; OFF disables it.

COMMENTS: When enabled, automatic label cutting has the same effect as placing a HALT command in every label format. The printer will cut and pause after printing each label in a batch. Removing the cut label or pressing the FEED button will signal the printer to print the next label in the batch.

When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

SEE ALSO: The HALT command.

EXAMPLE: The format below will enable automatic label cutting:

```
! 0 0 0 0
VARIABLE AUTOCUT ON
VARIABLE WRITE
END
```

VARIABLE BUFFER_TIMED_RESET

FUNCTION: Enables or disables the memory reset timer.

EXPLICIT FORM: `VARIABLE BUFFER_TIMED_RESET duration`

PARAMETERS:

duration = timer duration in 0.1 second intervals. The minimum value is 2 (0.2 seconds), maximum is 59990 (about one hour and forty minutes). The default value varies by printer type:

Code Courier: The default value is 5 (0.5 seconds). You can also specify the value as ON or OFF. ON will set the timer to 8 seconds, and OFF will set it to 512 seconds.

Barcode Blaster: The default value is 6000 (ten minutes).

COMMENTS: The printer expects incoming data to arrive in a timely manner. It will clear memory if it receives the beginning of a line of data and fails to receive the accompanying end-of-line termination (carriage return or line feed) within the time set by BUFFER_TIMED_RESET. Code Courier will also go to sleep at that time. This keeps the printer from waiting indefinitely for incoming data that was lost due to a communications error. The timer value may need adjustment if the printer must communicate with a system that has unusual data transmission timing.

When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

EXAMPLE:

```
! 0 0 0 0
VARIABLE BUFFER_TIMED_RESET OFF
VARIABLE WRITE
END
```

VARIABLE COMM

NOTE: Do not experiment with this command! It's improper use can cause loss of serial communication with the printer.

FUNCTION: Sets new serial port parameters.

EXPLICIT FORM: `VARIABLE COMM speed,parity,length,stop, R`

PARAMETERS: speed - Baud rate. Acceptable values are printer dependent, but will be among the following: 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, and 57600. Consult your printer User's Guide for details.

parity - O (odd), E (even), or N (none).

length - Data word length. Can be 7 or 8.

stop - Number of stop bits. Can be 1 or 2.

R - Enables robust XON/XOFF handshaking (see comments).

Notice that commas are used as delimiters between parameters, and there are no spaces. If you are uncertain of the printer's current serial port parameters, try 9600,N,8,2.

NOTE: Not all personal computers support all baud rates. Make certain that your host system will support your VARIABLE COMM parameters. Setting the printer for a baud rate that your host will not support will cause a break in communication that may be difficult to remedy.

Cognitive printers will not support all possible parity, word length, and stop bit combinations. The acceptable combinations are: N,8,1 N,8,2 O,7,1 O,8,1 E,7,1 E,8,1

COMMENTS: When enabled, robust XON/XOFF handshaking causes the printer to send an XON character once per second while it is ready to receive data. The printer will send an XOFF character for every character that overflows the printer's input buffer. Enabling robust XON/XOFF handshaking also enables double-buffered input, which allows simultaneous label processing and data reception. However, the double-buffered input reduces the image buffer size by 4 kB, thus slightly reducing the maximum label length. Also, you cannot print PDF417 bar codes with double buffering enabled.

This feature is normally disabled. A lowercase "r" character will print on the printer confidence label if robust XON/XOFF is enabled.

For printers having both serial and parallel ports, connect the host computer to the printer's parallel port when using the VARIABLE COMM command. You can use the printer's serial port if the printer or host does not have a parallel port, but the host will lose communication with the printer when the serial port parameters change. Set the new parameters in the host after you change the printer parameters.

Do not send any other data to the printer when changing serial port parameters. Do not use this command in stored label formats.

EXAMPLE:
```
! 0 0 0 0
VARIABLE COMM 9600,N,8,2
VARIABLE WRITE
END
```

VARIABLE DARKNESS

FUNCTION: This command changes the printhead heat, thereby adjusting the darkness at which labels are printed. Always use the lightest acceptable VARIABLE DARKNESS setting to extend the life of the printhead.

EXPLICIT FORM: `VARIABLE DARKNESS n`

PARAMETERS: n = darkness value. The allowable range varies by printer:

Barcode Blaster: -205 to +50

Solus, Blaster Advantage, and Code Courier: -200 to +813

Lower numbers decrease print darkness, higher numbers increase it.

Blaster Advantage TT printers store separate darkness values for DT and TT mode, selecting the appropriate darkness value depending on the print mode as set by the VARIABLE PRINT_MODE command.

COMMENTS: Darkness is set at the factory for optimum printing with typical Cognitive print media. It should not need adjustment under normal circumstances. You may need to adjust darkness if you change print speed or use unusual media.

When using thermal transfer printing, you may observe the print darkness decreasing as you increase the VARIABLE DARKNESS setting. This is caused by excessive heat melting the ribbon dye back onto the ribbon instead of depositing it on the paper, and indicates that the darkness setting is far too high. You may need to experiment with VARIABLE DARKNESS for optimum results with some thermal transfer media.

EXAMPLE: `VARIABLE DARKNESS -25`

NOTE: Always print labels at the lightest acceptable setting, to extend the life of the printhead.

VARIABLE FEED_TYPE

FUNCTION: Selects black bar or gap indexing.

EXPLICIT FORM: `VARIABLE FEED_TYPE mode`

PARAMETERS: mode = index mode, where GAP selects gap indexing and BAR selects black bar indexing.

When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

The FEED_TYPE setting has no effect unless automatic label indexing is enabled. Use the INDEX command to enable automatic label indexing if desired, but do not enable automatic label indexing when using continuous-form media.

The factory setup for some thermal transfer printers restricts the allowable indexing method when a ribbon is installed. Consult the applicable printer User's Guide for more information.

EXAMPLE: `The following label format enables gap indexing:`

```
! 0 0 0 0
VARIABLE FEED_TYPE GAP
VARIABLE WRITE
END
```

VARIABLE HIGHSPEED

FUNCTION: Changes the print speed to its highest available setting. The maximum print speed varies among printer models; consult your printer's User's Guide for more information.

EXPLICIT FORM: `VARIABLE HIGHSPEED`

PARAMETERS: none

COMMENTS: Code Courier automatically adjusts its speed in response to ambient temperature and battery condition. Refer to the Code Courier User's Guide for details.

comBarcode Blaster automatically disables high speed printing whenever it encounters the following commands or command parameters in a label format:

Header line X parameter

Header line dot time values other than 100

MULTIPLE command

OFFSET command

WIDTH command

On some printers, the starting print location may move vertically slightly if the print speed changes. Usually the change in location is negligible, but you can correct the starting location with the VARIABLE POSITION command if desired.

SEE ALSO: VARIABLE LOWSPEED and VARIABLE NORMAL commands.

EXAMPLE: `VARIABLE HIGHSPEED`

NOTE: Print speed can affect bar code scanning reliability, especially when printing rotated bar codes. For best results, use print speeds of 2 IPS or less when printing rotated bar codes.

VARIABLE INDEX SETTING

FUNCTION: Adjusts the index detector for optimum gap detection through a wide range of ribbon and label densities. The command is primarily for use with thermal transfer printers in gap indexing mode. You should not need to use this command when using black bar indexing.

EXPLICIT FORM: `VARIABLE INDEX SETTING mode`

NOTE: Observe that this command consists of three words, separated by spaces.

PARAMETERS: mode = the desired indexing mode. Allowable values are 0, 1, 2, 3, and CALIBRATE, as follows:

0Automatic mode: Sets the printer index sensitivity to mode 1 or 2 values as the print mode changes in response to the VARIABLE PRINT_MODE command.

1. Sets the printer to use direct thermal indexing values. This setting is calibrated at the factory for best gap indexing performance with no ribbon installed.
2. Sets the printer to use thermal transfer indexing values; factory calibrated for best performance with wax ribbon installed.
3. Same as mode 2, except factory calibrated for best indexing performance with a resin-based ribbon installed. The sensitivity of this mode is adjustable with the CALIBRATE mode (see below).

   CALIBRATE. Runs an index calibration, then replaces the index data currently stored for mode 3 with the new data.

NOTE : Confirm that the correct print media is loaded before starting the calibration.

COMMENTS: When using the VARIABLE INDEX SETTING modes 0, 1, or 2, send the command to the printer with VARIABLE WRITE in a non-printing label format. The use of VARIABLE WRITE is optional when using the VARIABLE INDEX SETTING CALIBRATE command. VARIABLE INDEX SETTING CALIBRATE always writes its data to nonvolatile RAM, with or without the use of VARIABLE WRITE.

Do not use this command in stored label formats.

The CALIBRATE mode runs an automatic calibration routine that measures the opacity of the currently loaded ribbon, labels, and backing, then uses these values to set the index detector sensitivity. This takes about 15 seconds in most printers. The printer's READY light flashes during calibration. The ready light will glow green if the calibration is successful, and red if the calibration is unsuccessful. Following a successful calibration, the printer automatically writes the new index calibration to nonvolatile RAM.

EXAMPLE 1: The following label format will set the automatic indexing mode:

```
! 0 0 0 0
VARIABLE INDEX SETTING 0
VARIABLE WRITE
END
```

EXAMPLE 2: The following label format will calibrate the index detector:

```
! 0 0 0 0
VARIABLE INDEX SETTING CALIBRATE
END
```

VARIABLE LOW_BATTERY_IGNORE

FUNCTION: Disables or enables low battery detection in portable printers.

EXPLICIT FORM: `VARIABLE LOW_BATTERY_IGNORE status`

PARAMETERS: status = Condition of low battery detection. ON causes the printer to ignore the battery condition; OFF enables normal low battery detection. The default condition is OFF for portable printers. Printers that are factory configured for fork lift use will be delivered with low battery detection ON.

COMMENTS: This function is intended for use in vehicle-mount printers, or in other situations where the printer is getting power from an external source.

When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

EXAMPLE: The following label format will cause the printer to ignore the battery condition:

```
! 0 0 0 0
VARIABLE LOW_BATTERY ON
VARIABLE WRITE
END
```

VARIABLE LOWSPEED

FUNCTION: Changes the printer speed to its lowest allowable value.

EXPLICIT FORM: `VARIABLE LOWSPEED`

PARAMETERS: None

COMMENTS: Code Courier automatically adjusts its speed in response to ambient temperature and battery condition. Refer to the printer's User's Guide for details.

SEE ALSO: VARIABLE HIGHSPEED and VARIABLE NORMAL commands.

EXAMPLE: The following command will set the printer speed to its lowest value:

```
VARIABLE LOWSPEED
```

NOTE : Do not use this command when programming the thermal transfer Code Courier (model PT422003).

VARIABLE MEDIA_ADJUST

FUNCTION: Adjusts print contrast on object leading edges for optimum print quality. Adjustment of this parameter is not normally required, but may improve rotated bar code reliability in some circumstances.
Printers that support this command employ an advanced "dot history" algorithm, which tracks the activity of each printhead dot from one dot row to the next. If a given dot was off prior to being commanded on, the printer will apply a little extra energy to the dot to force it to come up to full temperature faster. This can significantly improve the reliability of rotated bar codes. This setting is independent of the VARIABLE DARKNESS setting, and only affects the first dot row of each object (or for rotated bar codes, the leading edge of each bar in the code).

NOTE: This feature is only active when printing at 2 IPS with 200 DPI print pitch.

EXPLICIT FORM: `VARIABLE MEDIA_ADJUST n`

PARAMETERS: n - the media adjust value. The allowable range is -3000 to +3000. Fast media (media that prints well with low VARIABLE DARKNESS values) will generally print better with high MEDIA_ADJUST values. Slow media will generally need lower settings.

Optimum values for MEDIA_ADJUST may be found by experimentation (see comments below).

Thermal transfer printers store separate values for MEDIA_ADJUST in DT and TT mode, selecting the appropriate value depending on the current print mode.

COMMENTS: Optimum MEDIA_ADJUST values depend on the currently loaded media (for TT printers, both paper and ribbon). We suggest the following procedure for experimentally finding the optimum MEDIA_ADJUST value for your media.

1. Prepare the following label format:

```
! 0 100 190 1
PITCH 200
VARIABLE DARKNESS -70
VARIABLE MEDIA_ADJUST 3000
BARCODER CODABAR(2:4)- 20 10 200 0123456
BARCODER CODABAR(2:4)- 220 10 200 3456789
END
```

The MEDIA_ADJUST 3000 command in the above label format will effectively turn off MEDIA_ADJUST. The darkness value should be whatever you think will print a readable label on your printer.

2. Send the label format to the printer.
3. Examine the printed bar codes carefully (use a magnifying glass or jeweler's loupe). Ideally, the wide bars should show some breakup between dot rows and the narrow bars should be only one dot row wide. (This setting is probably lighter than you would normally want.)
4. Adjust the darkness value experimentally until you achieve the results described in step 3.
5. Reduce the MEDIA_ADJUST value to 2500. Do not change the DARKNESS value.
6. Print the label format again, and examine the bar codes. You may begin to see some darkening of the narrow bars (although no change may appear

until you have changed the MEDIA_ADJUST value significantly; results vary with media sensitivity). Ideally, the narrow bars should be two dots wide. The leading edge of the wide bars should also darken until they are solid all the way across.

7. Vary the MEDIA_ADJUST value in large increments (about 500) and print the test label until the printed bar code approaches the appearance described in step 6. Change MEDIA_ADJUST in smaller increments as you begin to see improvement in the print results.

After achieving satisfactory visual results, you may want to scan the bar codes with a bar code verifier. No further adjustment is required if the bar codes scan satisfactorily. The experimentally derived value will work with all media that has the same temperature response. You may send the new MEDIA_ADJUST value to the printer with VARIABLE WRITE as shown in the example below, or simply note the value for future reference.

NOTE: This method should provide optimum rotated bar code reliability, but may not produce visually pleasing labels. Labels that "look good" are often too dark for good bar code performance. You may want to increase the print darkness for attractive text and graphics.

Do not change MEDIA_ADJUST if you adjust the print darkness. The optimum value for MEDIA_ADJUST is independent of print darkness, and depends solely on the media sensitivity.

SEE ALSO: VARIABLE DARKNESS

EXAMPLE:
```
! 0 0 0 0
VARIABLE MEDIA_ADJUST 1000
VARIABLE WRITE
END
```

VARIABLE MODE

FUNCTION: Selects Blazer Emulation Mode in printers that support variable dot time, or sets the default print pitch in all other printers except Code Courier, which does not support the command.

EXPLICIT FORM: `VARIABLE MODE n scale`

PARAMETERS: n - mode type. Acceptable values are 0, 1, and 2.

In printers that do not support variable dot time, VARIABLE MODE 0 sets the default print pitch to its highest value. VARIABLE MODE 1 and VARIABLE MODE 2 set the default pitch to its lowest value.

In printers that support variable dot time, the three modes control Blazer Emulation. Printers operating in mode 0 emulate Enhanced Barcode Blazers with linear dot time enabled. Printers operating in mode 1 emulate typical non-enhanced Barcode Blazers. Printers operating in mode 2 also emulate non-enhanced Blazers, except this mode supports an additional parameter (scale).

scale - Scaling factor for mode 2 emulation. This is a required parameter when using mode 2 but is ignored when using modes 0 or 1. Valid values are 1 to 255, and set the scaled print length as a percentage of the original print length.

NOTE: Blazer emulation is normally disabled. Only use Blazer emulation if you are replacing a non-enhanced Barcode Blazer or a Blazer that has unusual dot time or speed characteristics.

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

SEE ALSO: VARIABLE PITCH command.

EXAMPLE:
```
! 0 0 0 0
VARIABLE MODE 1
VARIABLE WRITE
END
```

VARIABLE NO_MEDIA

FUNCTION: Specifies how long the printer will run without detecting a label before assuming that it is out of media. This only applies to gap indexing mode.

EXPLICIT FORM: `VARIABLE NO_MEDIA nn`

PARAMETERS: nn = the number of label inches that the printer will try to feed before assuming it is out of media. Allowable values are 0 to 12, with a default of 1. A value of 0 disables this feature.

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

This feature works by measuring label gap length. If the printer detects a gap that is longer than nn, it assumes that there are no more labels and stops feeding. You will need to change this setting if your label media has gaps greater than 1".

The printer always assumes it is empty if it does not detect a label after feeding 12" of media, regardless of the VARIABLE NO_MEDIA setting. (In black bar mode, the printer always stops running after feeding 12" of media without detecting an index mark.)

EXAMPLE:

```
! 0 0 0 0
VARIABLE NO_MEDIA 3
VARIABLE WRITE
END
```

VARIABLE NORMAL

FUNCTION: Changes the printer speed to a speed approximately halfway between the LOWSPEED and HIGHSPEED settings, or in printers with only two allowable speeds, sets the printer to the lowest speed. Your printer's User's Guide lists available print speeds.

EXPLICIT FORM: `VARIABLE  NORMAL`

PARAMETERS: None

COMMENTS: Code Courier automatically adjusts its speed in response to ambient temperature and battery condition. Refer to the Code Courier User's Guide for details.

SEE ALSO: VARIABLE LOWSPEED and VARIABLE HIGHSPEED commands.

EXAMPLE: The following command will set the printer speed to its "normal" value:

```
VARIABLE NORMAL
```

VARIABLE ON_TIME

FUNCTION: Controls the printer idle power saving feature. The idle shutdown time you set with this command remains in effect until issuing a new VARIABLE ON_TIME command.

EXPLICIT FORM: `VARIABLE ON_TIME n`

PARAMETERS: n = the number of timer intervals that must elapse before the printer shuts itself off during idle periods. Each interval is 512 seconds long. Allowable values for n are 0 to 65535. If n = 0 the printer will never shut down, effectively disabling this feature.

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format.

The printer idle timer begins counting at the end of each printer operation, and resets at the beginning of each new operation. The timer remains at zero while the printer is processing label formats, even if it is paused during a batch printing operation.

Refer to the Code Courier User's Guide for more information regarding printer power management.

EXAMPLE:
```
! 0 0 0 0
      VARIABLE ON_TIME 3
      VARIABLE WRITE
      END
```

VARIABLES ON/OFF

FUNCTION: Enables and disables access to certain protected VARIABLE values.

EXPLICIT FORM: `VARIABLES status`

PARAMETERS: status = Protection status. ON allows access to the protected VARIABLE settings. OFF prohibits access to the protected VARIABLE settings. The default is ON.

COMMENTS: Do not use this command in stored label formats.

This command acts as a "write-protect" feature for certain VARIABLE commands, preventing the user from inadvertently disabling or enabling them. The only protected VARIABLE command at present is VARIABLE MODE. Future firmware revisions may extend this protection to other setup parameters.

EXAMPLE:
```
! 0 0 0 0
      VARIABLES OFF
      VARIABLE WRITE
      END
```

VARIABLE PITCH

FUNCTION: Selects the default print pitch.

EXPLICIT FORM: `VARIABLE PITCH n`

PARAMETERS: n - Default print pitch. Allowable values are as used in the PITCH command.

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

SEE ALSO: The PITCH command.

EXAMPLE:
```
! 0 0 0 0
VARIABLE PITCH 100
VARIABLE WRITE
END
```

VARIABLE POSITION

FUNCTION: Moves the first printed dot row on the label up or down with respect to its last position.

EXPLICIT FORM: `VARIABLE POSITION distance`

PARAMETERS: distance = the distance between the original starting location and the new location, in thousandths of an inch. Positive numbers move the first dot row down; negative numbers move the first dot row up.

NOTE: The first dot row must always be at least 3/16" from the top edge of the label when using black bar indexing. Trying to print too near to the top edge by using the VARIABLE POSITION command causes label skipping.

COMMENTS: Use this command with VARIABLE WRITE in a non-printing label format. After sending the file to the printer, press the FEED button once to set the new index position.

Do not use this command in stored label formats.

The position set with this command only applies to the current indexing method (black bar or gap).

EXAMPLE:
```
! 0 0 0 0
VARIABLE POSITION -10
VARIABLE WRITE
```

```
END
```

VARIABLE PRESENTLABEL

FUNCTION: Controls the dispensing of labels for application. `VARIABLE PRESENT-LABEL` ensures a second set of labels is not printed before the operator is ready.

EXPLICIT FORM: `VARIABLE PRESENTLABEL <parameters>`

PARAMETERS:

`ADVANCE nnn:` Distance the label is advanced in hundreths of an inch.

`REVERSE nnn:` Distance the label is retracted in hundreths of an inch. Default distance equals the `ADVANCE` distance.

`OFF`: Disables the `PRESENTLABEL` feature.

`ON:` Enables the `PRESENTLABEL` feature.

COMMENTS: The `REVERSE` value must be equal to or less than the `ADVANCE` value.

EXAMPLE:

```
! 0 0 0 0
VARIABLE PRESENTLABEL ADVANCE 200
VARIABLE PRESENTLABEL REVERSE 180
VARIABLE WRITE
END
```

VARIABLE PRINT_MODE

FUNCTION: Sets the printer up for direct thermal or thermal transfer printing. The command adjusts print darkness and gap indexing parameters and enables or disables the ribbon-out detector as needed for the selected print method.

EXPLICIT FORM: `VARIABLE PRINT_MODE method`

PARAMETERS: method = print method. DT selects direct thermal printing and TT selects thermal transfer printing. AUTO (available only in Blaster Advantage printers) selects automatic print mode switching. In AUTO mode, the printer will check for the presence of a ribbon when it is turned on or when you lower the printhead. If it detects a ribbon it will set itself for TT mode, otherwise it will automatically select DT mode.

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

EXAMPLE:
```
! 0 0 0 0
VARIABLE PRINT_MODE TT
VARIABLE WRITE
END
```

VARIABLE READ

FUNCTION: Reads variable values in permanent storage (the last saved settings) and uses them as the current values. The effect is the same as turning the printer off and then on again.

EXPLICIT FORM: `VARIABLE READ`

COMMENTS: Do not use this command in stored label formats.

SEE ALSO: The VARIABLE WRITE command.

VARIABLE REPORT_LEVEL

FUNCTION: Sets the manner in which the printer reports recoverable errors.

EXPLICIT FORM: `VARIABLE REPORT_LEVEL n`

PARAMETERS: n = error message delivery method. The printer can report errors three ways:

TABLE 26. Error Reporting

#	Reporting Method
0	No error reporting (the default error level in most printers)
1	Error messages sent to serial port
2'	Error messages sent to serial port and printed on label

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

EXAMPLE: The label format below will tell the printer to send an error message to the printer serial port and print an error label when it encounters a recoverable error:

```
! 0 0 0 0
VARIABLE REPORT_LEVEL 2
VARIABLE WRITE
END
```

VARIABLES RESET

FUNCTION: Returns user-accessible VARIABLE parameters to known values. Affected parameters and their default values vary among printer models.
After it is invoked, the command writes the known values to nonvolatile RAM.

EXPLICIT FORM: `VARIABLES RESET`

PARAMETERS: None

COMMENTS: Do not use this command in stored label formats.

This command allows you to reset the printer to a known state. This is sometimes useful if the printer is not behaving as you expect.

NOTE: VARIABLES RESET immediately writes the new values to nonvolatile RAM without using VARIABLE WRITE.

EXAMPLE:
```
! 0 0 0 0
VARIABLES RESET
END
```

VARIABLE SLEEP_AFTER

FUNCTION: Sets the amount of time a portable printer will stay awake after completing a print job.

EXPLICIT FORM: `VARIABLE SLEEP_AFTER duration`

PARAMETERS: duration = the length of time the printer will stay awake, in seconds. The allowable range is 0 to 255, with a default of one second. Setting duration to 0 will keep the printer awake continuously.

When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

NOTE: We do not recommend setting a long SLEEP_AFTER duration, unless the printer is powered from an external source. Keeping the printer awake for long periods will substantially reduce battery charge life.

SEE ALSO: VARIABLE ON_TIME

EXAMPLE: The following label format tells the printer to stay awake for 30 seconds after each print job:
```
! 0 0 0 0
VARIABLE SLEEP_AFTER 30
VARIABLE WRITE
END
```

VARIABLE SHIFT LEFT

FUNCTION: Shifts the printed image of all labels a specified distance to the left from the normal 0,0 origin.

EXPLICIT FORM: `VARIABLE SHIFT LEFT n`

PARAMETERS: n = The distance the image is shifted to the left, in hundredths of an inch.

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

EXAMPLE: The following command will shift the label image 1.28" to the left of its normal position:

```
VARIABLE SHIFT LEFT 128
```

VARIABLE TXTBFR

FUNCTION: This command sets the size of the text buffer and the text overflow buffer.

EXPLICIT FORM: `VARIABLE TXTBFR txt ovf`

PARAMETERS: txt = size of the text buffer, in bytes. The allowable range is 4096 to 65536, with a default of 4096.

ovf = Optional; specifies the size of the overflow buffer, in bytes. The allowable range is 0 to (txt - 1024).

NOTE: Turn the printer OFF and back ON again just before you send this command to the printer. Some printers may ignore the TXTBFR command if printer power is not cycled first.

COMMENTS: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

The text buffer holds incoming ASCII data before it is processed. The overflow buffer is part of the text buffer. It begins filling after the rest of the text buffer is full, at which time the printer will signal the host to stop sending data.

Data sent via the serial port cannot overflow the text buffer, providing hardware or software flow control is enabled. Data sent via the parallel port will overflow the text buffer if the size of the ASCII label format is larger than (txt -ovf). Overflowing the text buffer in Code Courier or Barcode Blaster will cause the printer to lock up. Blaster Advantage and Solus will print an error label if the text buffer overflows.

Increasing the text buffer size will decrease the size of the image buffer (Total memory size = text buffer + image buffer).

NOTE: Changing the text or overflow buffer sizes will delete any objects stored in the image buffer.

SEE ALSO: The VARIABLE ALLOCATE command.

EXAMPLE: The following label format sets the text buffer size to 4 kB:

```
! 0 0 0 0
VARIABLE TXTBFR 4096
VARIABLE WRITE
END
```

VARIABLE USER_FEEDBACK

FUNCTION: Enables or disables the transmission of certain status messages to the host computer. With this feature enabled, the printer regularly sends some status messages via the serial port. The printer will not send messages if the host computer is busy (as indicated by the condition of its serial port CTS line).

Explicit Form: `VARIABLE USER_FEEDBACK status`

Parameters: status = condition of the user feedback function. ON enables this feature; OFF disables it.

Comments: When using this command, send it to the printer with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

See Also: The QUERY STATUS command.

Example: The following label format will enable user feedback:

```
! 0 0 0 0
VARIABLE USER_FEEDBACK ON
VARIABLE WRITE
END
```

VARIABLE WIDTH

Function: Sets the default print width.

Explicit Form: `VARIABLE WIDTH n`

Parameters: n = print width, in hundredths of an inch.

Comments: Use this command with VARIABLE WRITE in a non-printing label format. Do not use this command in stored label formats.

This command is functionally identical to the WIDTH command, except that it can set the new print width in nonvolatile RAM. Used with VARIABLE WRITE, VARIABLE WIDTH can eliminate the need to include WIDTH commands in label formats designed for narrow printers. It is still good programming practice, though, to include a WIDTH command in every new label format. This helps assure compatibility with other printers that may not support VARIABLE WIDTH.

See Also: WIDTH command.

EXAMPLE: The following command sets the print width to 2.24":

```
VARIABLE WIDTH 224
```

VARIABLE WRITE

FUNCTION: Writes the current variable values to nonvolatile storage. Values in effect when VARIABLE WRITE is executed are retained in memory while the printer power is off.

EXPLICIT FORM: `VARIABLE WRITE`

PARAMETERS: None

COMMENTS: You do not need to use this command every time you want to permanently change a VARIABLE value. The printer retains VARIABLE values as long as its power is on, and a single VARIABLE WRITE will permanently store the entire set of VARIABLE values. Excessive use of VARIABLE WRITE increases label processing time and may also increase the risk of non-volatile RAM failure in some early printers.

NOTE: Follow these rules with respect to VARIABLE WRITE:

Use the VARIABLE WRITE command only in dummy (non-printing) label formats.

Do not use this command in stored label formats.

Only use the VARIABLE WRITE command with other VARIABLE commands. Mixing normal commands with VARIABLE WRITE can corrupt the contents of the nonvolatile RAM in some printers.

EXAMPLE: The example below shows how to set a VARIABLE value in one label format with VARIABLE WRITE. But you must remember that this label format is also writing all other current VARIABLE values to nonvolatile RAM at the same time.

```
! 0 0 0 0
VARIABLE HIGHSPEED
VARIABLE WRITE
END
```

CHAPTER 7

Using Variable Commands

Printer configuration must be handled through VARIABLE commands. The sample label formats in this section are provided for your convenience, to help you set the printer up for some common configurations. You can use these label formats by copying them to Windows Notepad or a similar ASCII text editor, editing them as required, and then printing them.

NOTE! If you are printing the files directly from Notepad or another Windows-based program, be aware that most Windows printer drivers will not work with Cognitive printers. The "generic ASCII" printer driver (supplied with Windows) will pass the label format to the printer without interference. Please install and use this driver when sending these sample formats to the printer. Do not use the Cognitive Intellidriver when sending these formats to the printer from Windows. The Intellidriver converts Windows documents to ASCII label formats; thus, your label formats will print as they appear in the text editor rather than directly control the printer as intended.

Most of these sample formats use the VARIABLE WRITE command. Familiarize yourself with this command before using the samples.

Blazer Compatibility

As shipped, Barcode Blaster, Solus, and Blaster Advantage printers are not compatible with older Barcode Blazers, or with enhanced Blazers operating in nonlinear dot time mode. High speed Blasters can closely emulate Barcode Blazers. Solus and Advantage printers are more limited in their Blazer emulation.

As a rule, you should only use Blazer emulation mode if you are replacing an older Barcode Blazer and must avoid modifying any label formats or software. Do not use Blazer emulation mode if you can print successfully without it.

Blazer emulation mode is controlled by the VARIABLE MODE command. VARIABLES ON/OFF controls access to VARIABLE MODE. Please refer to these command descriptions before using the label formats.

The following label format sets up any Barcode Blaster to emulate an enhanced Barcode Blazer operating in linear dot time mode:

```
! 0 0 0 0
VARIABLE MODE 1
VARIABLE WRITE
END
```

The following label format will set up a high speed Blaster to emulate a non-enhanced Blazer or an enhanced Blazer operating in nonlinear dot time mode. You will have to adjust the second parameter in the VARIABLE MODE command to compensate for the speed of the Blazer you are replacing. The value shown (70) will vertically scale all labels to 70% of their normal height.

```
! 0 0 0 0
VARIABLE MODE 2 70
VARIABLE WRITE
END
```

The following label format disables Blazer emulation mode and returns the printer to normal operation:

```
! 0 0 0 0
VARIABLE MODE 0
VARIABLE WRITE
END
```

There are many aspects to printer compatibility. You may want to request a copy of our Technical Bulletin 10-00-0131, which covers Blazer/Blaster compatibility in detail. Contact our Technical Support Organization for more information.

Setting DT or TT Print Method

Thermal transfer Barcode Blasters can print in direct thermal or thermal transfer mode, but you must set the printer for the correct media type. After loading the appropriate media, set the printer for the correct media type using the VARIABLE PRINT_MODE command.

You can use one of the following label formats to set the print method:

For thermal transfer printing:

```
! 0 0 0 0
VARIABLE PRINT_MODE TT
VARIABLE WRITE
END
```

For direct thermal printing:

```
! 0 0 0 0
VARIABLE PRINT_MODE DT
VARIABLE WRITE
END
```

Setting Bar or Gap Index Type

All currently manufactured Cognitive printers support black bar and gap indexing, and are shipped from the factory set for gap indexing. The VARIABLE FEED_TYPE command controls the index method. Use one of the following label formats to change the index type.

For black bar indexing:

```
! 0 0 0 0
VARIABLE FEED_TYPE BAR
VARIABLE WRITE
```

```
        END
```

For gap indexing:

```
        ! 0 0 0 0
        VARIABLE FEED_TYPE GAP
        VARIABLE WRITE
        END
```

Optimizing Index Detection

NOTE: This information is not applicable to Code Courier printers.

When using gap indexing in thermal transfer mode, the index detector must "see" through the ribbon and detect the difference in opacity between the label and the label backing. Variations in print media may necessitate index detector adjustment in some cases. The VARIABLE INDEX SETTING command controls the index detector sensitivity.

Direct Thermal Printing

The following label format will set the index detector for optimum sensitivity when using direct thermal printing (that is, with no ribbon installed):

```
        ! 0 0 0 0
        VARIABLE INDEX SETTING 1
        VARIABLE WRITE
        END
```

Thermal Transfer Printing with Standard Wax Ribbon

The following label format will set the index detector for thermal transfer printing using Cognitive's standard wax ribbon:

```
        ! 0 0 0 0
        VARIABLE INDEX SETTING 2
        VARIABLE WRITE
        END
```

Thermal Transfer Printing with Resin Ribbon

The following label format will set the index detector for thermal transfer printing with typical resin-based ribbon installed:

```
! 0 0 0 0
VARIABLE INDEX SETTING 3
VARIABLE WRITE
END
```

Automatic Detect

The following label format causes the printer to automatically set the index detector to mode 1 or 2 in response to the current print method (DT or TT) as set by the VARIABLE PRINT_MODE command (this is the default setting):

```
! 0 0 0 0
VARIABLE INDEX SETTING 0
VARIABLE WRITE
END
```

Calibrate the Index

The following label format runs an index detector calibration routine, and then stores the new index detector sensitivity setting in nonvolatile RAM under index sensitivity 3. After calibration, the routine will set the printer in index mode 3:

```
! 0 0 0 0
VARIABLE INDEX SETTING CALIBRATE
END
```

NOTE: You do not need to use VARIABLE WRITE when using the CALIBRATE mode. CALIBRATE automatically writes its data to nonvolatile RAM.

Setting Print Width

You can set the print width in any label format using the WIDTH command, but if you plan to consistently use narrow print media (for example, when using an optional wristband tray) you may want to change the default print width. The VARIABLE WIDTH command lets you do this.

Use the following format to change the default print width (replace the 224 following the VARIABLE WIDTH command with the desired print width, measured in hundredths of an inch):

```
! 0 0 0 0
VARIABLE WIDTH 224
VARIABLE WRITE
END
```

CHAPTER 8

Bar Code Information

All rules of bar code symbologies must be followed when creating bar code commands. Some of the rules for the most commonly used bar code symbologies are listed below.

For more information on bar codes as supported in Cognitive printers, refer to the BARCODE command description and Table 2, Printer bar code support.

Uniform Product Code, or UPC

The Uniform Product Code (UPC) family of codes are typically used for product identification, and include UPCA, UPCA+, UPCE, and UPCE1. These are numeric codes only, supporting the characters 0 through 9. All UPC codes start with a number system digit that identifies the type of product being coded, and end with a checksum digit.

UPCA consists of a number system digit, ten numbers for the product identification, and a checksum digit. All Cognitive printers automatically calculate the checksum. The checksum is not printed in the bar code subtext.

UPCA+ is like UPCA except that extender bars are printed as per UPCA specification, and the checksum is printed in the bar code subtext. If a minus sign modifier is specified, the bar code is the same but the checksum is removed from the bar code subtext.

UPCE is a six-digit variation of the UPC symbology. It always uses number system zero. You must enter the six numbers of the bar code, but not the number system digit or the checksum. The printer calculates the checksum automatically. The bar code is printed with extender bars and bar code subtext. The bar code subtext shows the system number, the six digits of the bar code, and the checksum digit. Using the minus sign bar code modifier removes the checksum from the bar code subtext.

UPCE1 is the same as UPCE, except that it always uses number system one.

I2OF5 AND D2OF5

I2OF5 (Interleaved 2 of 5) is an interleaved code, used mainly in the distribution industry. It supports numbers 0-9 only. It can use a checksum digit, but the user must calculate and enter the checksum manually as part of the code. I2OF5 uses start and stop characters, which the printer generates automatically. Because of the interleaved pattern, an even number of digits (including the checksum, if used) must be placed in the bar code string. For example, 0123 is valid, while 123 is not valid.

D2OF5 (Discrete 2 of 5) is a numeric code, often used for envelope identification and airline ticketing. Data is encoded by bar widths only, with the spaces between bars only serving to separate the individual bars. Five binary bits (three 0 and two 1) encode each character. The binary bits are bar coded in sequential sets of five bars. The code uses start and stop characters, which the printer automatically adds to the printed code.

CODE39 and CODE39+

CODE39 is a widely used alphanumeric code that supports numbers and characters 0-9, A-Z, ".", space, "$", "/", "+" and "%". It does not support lowercase characters. Code 39 is self-checking, and does not normally use a check digit. An asterisk is used as a start and stop character, and the printer automatically adds it to the bar code. The asterisk does not normally print as part of the bar code subtext. If you put an asterisk before and after the bar code data, it will appear with the bar code subtext but will not be added to the code.

CODE39+ is the HIBC (Health Industry Bar Code) standard symbology. It is similar to normal Code 39, except it uses an automatically generated check digit. The check digit does not appear in the bar code subtext.

CODE93

CODE93 is similar to Code 39. It can encode 48 different characters, and with the use of control characters, typically encodes the entire 128 ASCII characters. Each encoded character in a Code 93 symbol is represented by three variable width bars and spaces.

The characters represented by Code 39 are represented in Code 93 as single bar code characters, but all other Code 93 characters are represented by a control character plus another character. You must take this into account when estimating bar code length.

EAN, EAN8, and EAN13

EAN (European Article Numbering) codes are an extension of the UPC system. A bar code scanner set to read EAN can read UPC; however, a scanner set for UPC may not read EAN. The EAN codes use a checksum character, which the printer automatically calculates. EAN codes are available in two versions: EAN13, which codes 13 digits, and EAN8, which codes 8 digits.

EAN13 has the same number of bars as UPCA. It encodes a number system character, eleven data digits, and a checksum. You must enter a number system character and eleven data digits. The printer calculates and adds the check digit, but the check digit does not print in the bar code subtext. EAN13+ is the same code printed with extender bars and with the check digit in the bar code subtext.

EAN8 encodes a number system character, six data digits, and a checksum. The printer calculates and adds the check digit, but the check digit does not print in the bar code subtext. EAN8+ prints the same code with extender bars and with the check digit in the bar code subtext.

ADD2, ADD5

ADD2 is a two digit add-on for UPC and EAN. ADD5 is a five digit add-on for UPC and EAN.

NOTE: Although ADD2 and ADD5 are add-ons to UPC and EAN, they must be entered in label formats as separate bar codes. Placement and size of ADD2 and ADD5 is independent of the UPC or EAN code.

CODABAR

CODABAR supports numbers 0-9 and the special characters ":", ".", "$", "+" and "-". It requires you to frame the numeric data with valid start/stop character pairs; for example, A0123B, where A is the start and B is the stop character. The valid start characters are designated A, B, C, and D. The valid stop characters are designated T, N, *, and E. Since you may use any of the four start/stop characters on either end of the symbol, there are 16 possible combinations. These combinations can identify the product type or other information.

PLESSEY AND MSI1

PLESSEY code supports numerals 0-9, plus six additional characters (typically A-F). PLESSEY uses a check digit, but the check digit may be calculated several different ways. To allow the user some flexibility the printer does not calculate or print the check digit. You must calculate and enter the check digit manually, according to the requirements of your bar code system.

MSI is a modified PLESSEY code that uses two check digits. The printer automatically calculates and adds the check digits. The check digits are not printed in the bar code subtext.

MSI1 is another modified PLESSEY code that uses one check digit. Again, the printer will automatically calculate and add the check digit. The check digits are not printed in the bar code subtext.

MAXICODE

MAXICODE is a fixed-size, two-dimensional bar code symbology consisting of a matrix of hexagonal elements arranged around a bull's-eye "finder pattern." MaxiCode uses five code sets (designated A through E) to encode all 256 characters of the extended ASCII character set. Cognitive's implementation of MaxiCode only supports code set A at present. This code set can represent the uppercase characters A - Z, numerals 0 - 9, most common punctuation marks, and a few special symbols.

PDF417

PDF417 (an abbreviation for Portable Data File 417), originally developed by Symbol Technologies, Inc., is a two-dimensional stacked bar code symbology. It is a highly compact medium for encoding any data representable by the 256 characters of the International Character Set.

The codeword is the basic unit of a PDF417 bar code. All data encoded using PDF417 is first converted to a decimal value between 0 and 928 inclusive, since there are 928 discrete symbols that can be represented by the allowable pattern of bars and spaces in each codeword. The printer converts the raw data to a series of numeric values following rules that provide optimum data compression. PDF417 provides several different rule sets, or modes, for optimum data compression.

PDF417 provides error detection and correction within the bar code block. The thoroughness of the automatic error checking is called the security level of the code. There are eight security levels, numbered 0-8, as shown below.

TABLE 27. PDF417 Error Correction

Security level	Error limit
0	0
1	2
2	6
3	14
4	30
5	62
6	126

TABLE 27. PDF417 Error Correction

Security level	Error limit
7	254
8	510

As long as the number of unreadable or missing code words in the bar code block is less than the number indicated for the applicable security level, the code may be read without error.

A high security level provides very reliable data encoding. However, the bar code block gets bigger with increasing security, since more codewords are needed to provide the necessary error checking data. Processing speed also increases significantly with increasing security, since more error checking calculations are performed.

Cognitive printers automatically handle most of the decisions and tasks associated with printing PDF417 bar codes. They select the best mode for the data, encode it, and do all calculations associated with start and stop characters and error-checking.

POSTNET

POSTNET is designed for use with the nine digit ZIP + 4 postal code. Each character to be encoded is represented by five bar code elements, with each element being either a short or tall bar followed by a space. The bar and space widths are constant.

Postnet uses a start and stop bar and a modulo 10 check digit. Cognitive printers automatically calculate and add this digit to the code.

CODE128 A, B, C

CODE128 uses 106 unique characters in three character sets to represent the numerals 0 through 9, the English alphabet in both upper and lowercase, some punctuation, and some special characters. Cognitive Solutions printers automatically calculate and add the checksum character, as well as any required start and stop characters. CODE128A, CODE128B, and CODE128C are the three character sets of the CODE128 symbology.

CODE128A can encode punctuation, the digits 0 through 9, the English alphabet in uppercase only, the standard ASCII control codes, and the special characters shown in Table 2.

CODE128B can encode punctuation, digits 0 through 9, the English alphabet in both upper and lowercase, and the special characters in the table following.

CODE128C is numeric only, and encodes numbers 00 through 99 plus the special characters shown in the table. It encodes numbers more efficiently than CODE128A or CODE128B, since the numbers are encoded as double digits.

Cognitive printers can handle special characters like the bell character that are contained in the CODE128 specification by using the caret (^) character followed by the two digit number of the character to be printed. The caret character works only with numbers between 0 and 38. The first 32 characters (0 to 31) will print the character represented by its ASCII number; for instance, ^07 will print the BELL character and ^13 will print the carriage return character. The two digit numbers from 32 to 38 will print the special characters shown below.

TABLE 28. CODE128 Special Characters

2 Digit Code	CODE128A	CODE128B	CODE128C	CODE 128 auto
32	FNC3	FNC3	invalid	FNC3
33	FNC2	FNC2	invalid	FNC2
34	SHIFT	SHIFT	invalid	invalid
35	CODE C	CODE C	invalid	invalid
36	CODE B	FNC4	CODE B	FNC4
37	FNC4	CODE A	CODE A	FNC4
38	FNC1	FNC1	FNC1	FNC1

If you want a caret to actually appear on the label, place a caret before every caret you want printed. For example, ^^ prints one caret, and ^^^^ prints two carets.

When the version of CODE128 is selected in the printer command file, the printer takes care of inserting the correct start character and stop character for the version of CODE128 selected. Only visible characters will be centered and printed under the bar code.

Specifying CODE128 without the A, B, or C modifier will cause the printer to automatically select and shift among the three symbology versions for optimum data compression, resulting in the smallest possible bar code.

EXAMPLE 1: Print the CODE128A bar code ABCD followed by a BELL character and a carriage return character.

```
! 0 120 115 1
BARCODE CODE128A 152 62 20 ABCD^07^13
STRING 12X16 149 12 CODE 128A
END
```

EXAMPLE 2: Print the CODE128B bar code ABCD1234. To keep the bar code width as small as possible without automatic mode switching, use ^35 to switch to code C right before the 1234 part of the string.

```
! 0 120 115 1
BARCODE CODE128B 132 62 20 ABCD^351234
STRING 12X16 132 12 CODE128B
END
```

EXAMPLE 3: Print the CODE128C bar code 1234567. CODE128C is a double digit bar code, printing the same bar code number in half the width as CODE128A. Because it is a double digit bar code, it will accept an even number of digits only.

```
! 0 120 115 1
BARCODE CODE128C 152 66 20 12345678
STRING 12 X 16 140 22 CODE 128C
END
```

CODE16K

CODE16K is a multi-row symbology based on CODE128. It offers the features of CODE128 with the added density of a two-dimensional bar code. Each CODE16K symbol consists of from two to sixteen rows. Each row consists of a leading quiet zone, a start character, a guard bar, five symbol characters, a stop character, and a trailing quiet zone. Rows are separated from each other by a separator bar, and there are separator bars at the top and bottom of the symbol as well.

As with CODE128, CODE16K has three unique character sets as shown in table 4. But unlike CODE128, Cognitive printers automatically select the best character set for the encoded data when using CODE16K. The user does not need to specify the character set when programming CODE16K bar codes.

You can encode special characters, such as the bell, using the caret character as with CODE128. The special character assignments for some characters differ slightly, as shown in the table below.

TABLE 29. CODE16K Special Characters

2 Digit Code	Function
33	FNC3
34	FNC2
35	N/A
36	N/A
37	FNC4
38	FNC4
39	FNC1

CHAPTER 9

Media Tips and Tricks

Labels are available in a large variety of types and sizes. Some label characteristics must be taken into account when programming the printer.

Media characteristics that can affect printer programming and printer performance are:

- Label/tag size and shape: Relates to the WIDTH, VARIABLE WIDTH and VARIABLE SHIFT LEFT commands.
- Adhesive type: Can affect printer peelback performance.
- Print method: Relates to the VARIABLE PRINT_MODE command.
- Cut type: Relates to the VARIABLE FEED_TYPE , VARIABLE NO_MEDIA , INDEX and NOINDEX commands.
- Media sensitivity: Relates to the VARIABLE DARKNESS and VARIABLE MEDIA_ADJUST commands.

Label/tag Size and Shape

Label and tag stock is available in an almost infinite variety of sizes and shapes. If an existing style does not fit your requirement, custom stock can usually be created. Your application will determine the design. The only absolute limitation on label size is the printer's maximum print width.

When printing on media that is narrower than the printer's maximum print width, you may have to adjust the width and print position with the WIDTH or VARIABLE WIDTH commands or the VARIABLE SHIFT LEFT command.

Adhesives

Labels can come without adhesive (e.g. tag or receipt stock), or with many different types of adhesive, including:

- Standard adhesives, used for normal applications
- Removable adhesives for labels to be used, then removed
- Special adhesives for use under extreme conditions, such as high or low temperatures and humidity
- Spot adhesives (adhesive applied to selected label areas)

When using a new adhesive type, test samples of labels having the new adhesive before applying them in large quantities.

The type of adhesive used can affect the performance of peeler-equipped printers when using peel back mode. Labels can sometimes cling too tightly to their backing to peel cleanly. Consult your Cognitive Solutions dealer or the factory for peel back mode compatibility information.

Print Method (Direct Thermal or Thermal Transfer)

Thermal printers use either direct thermal or thermal transfer printing. Direct thermal printing creates the printed image by changing the color of the label media with applied heat, while thermal transfer printing creates the printed image by transferring ink from a ribbon to the label media with applied heat. Direct thermal media is suitable for many indoor applications where environmental conditions are mild and labels are not expected to be in place for long periods. Use thermal transfer media when preparing labels for use in more demanding environments.

Cognitive Thermal transfer printers can operate in thermal transfer or direct thermal mode, but must be set for the proper mode with the VARIABLE PRINT_MODE command.

Cut Type (Butt Cut, Gap Cut, or Continuous Form)

Butt cut labels are separated from each other by a simple cut, leaving square edges. Gap cut labels are separated by a gap. Labels may be made in many shapes for special purposes such as the "butterfly" shape used for jewelry and eyeglass labels. Continuous form media has no perforation, gap, or bar and is typically used with cutter-equipped printers like the Blaster CL.

The cut type may affect label indexing:

- Butt-cut labels have no gap, and so must have black bars preprinted on the label backing in order for them to index properly. When using butt-cut labels in your Cognitive printer, set the index type to BAR using the VARIABLE FEED_TYPE command.
- Gap cut labels may or may not actually have a detectable gap, depending on the die design. If the gap cut labels have at least 1/8" gap between them, you may use gap indexing. If the gap is less than 1/8", you must use black bar indexing.

Most printers will assume they are out of media if they are operating in gap indexing mode and do not detect the next label after feeding 1" of media. This behavior is controlled by the VARIABLE NO_MEDIA command. You may have to adjust this parameter when using media that has an unusually long gap between labels.

Continuous form media is typically used with tearbar or cutter-equipped printers. Cutter printers will cut the media after reaching the last dot row as defined by the Max Y value in the Header Line, providing there is an INDEX command in the label format or VARIABLE AUTOCUT is ON.

NOTE: Cutter-equipped printers like the Barcode Blaster CL have very stringent media requirements. Using any media other than that specifically approved for use in the printer may cause serious damage.

Media Sensitivity

Different media types exhibit different heat sensitivity. The VARIABLE_DARKNESS command controls the amount of printhead heat.

Different media types can also exhibit different response times. The media is in motion while the printhead dots are heating and cooling, so areas of the label that theoretically should stay white are subjected to temperatures close to their darkening temperature. This can result in some bleeding of dark areas into light areas. This can adversely affect the reliability of rotated bar codes. Increasing print speed and print darkness tend to increase bleeding, so speed and darkness can interact to affect the reliability of rotated bar codes.

Some Cognitive printers support the VARIABLE MEDIA_ADJUST command, which can reduce print bleeding in some cases. Printers that support this command employ an advanced "dot history" algorithm, which tracks the activity of each print-head dot from one dot row to the next. If a given dot was off prior to being com-manded on, the printer will apply a little extra energy to the dot to force it to come up to full temperature faster, thus improving the contrast on the leading edges of objects. This can significantly improve the reliability of rotated bar codes.

CHAPTER 10 Troubleshooting

Most programming problems quickly resolve themselves with careful examination of the offending label format or program code. When you cannot solve a problem by simply reviewing your work, your best approach is to start troubleshooting using the information in your User’s Guide.

NOTE: Some programming problems can masquerade as hardware problems. If your printer seems to be broken but the information in the User's Guide does not point to a solution, review "Common problems and their solutions" below.

If you know or suspect that you have a hardware problem, refer to your User’s Guide. If you aren’t sure, try these preliminary tests to help isolate the problem:

1. Run a printer self-test.

   If the printer will not print a self-test label, follow the procedures in the User’s Guide to resolve that problem first. The printer will not respond to incoming data if it will not print a self-test label.
2. Print a proven label format.

   If you have a known-good label format, try printing it before troubleshooting new label formats. If the proven format will not print, look for a communication problem. If the proven format does print, you will have eliminated most of the possible hardware-related problems.

3. Try printing a label format written on your system.

   Sometimes the host operating system or text editor produces data that is incompatible with the printer. Creating a simple label format using the host system and sending the format to the printer helps isolate this problem. Something like this will do:

   ```
   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
   ! 0 100 20 1
   STRING 8X8 10 10 HELLO
   END
   ```

If the printer will not print this label but does print labels that were prepared on another system, your system may not be compatible with the printer. The most common cause of this is improper end-of-line termination.

NOTE: Hex dump mode allows you to see every character that reaches the printer. Use of hex dump mode can help resolve some programming problems.

4. Check the label format header line.

   When a particular label format will not print or is cut off at the bottom, the trouble is frequently in the header line. Check the header line before continuing. Especially check that the header line begins with the proper mode character and specifies the correct number of dot rows. When programming portable printers, the header line should be preceded by the wake-up string.

5. "Comment out" portions of the nonworking label format until your results improve.

   You can usually isolate a bad printer command by placing a "C" before selected command lines. Begin by commenting out the most complex lines, then try printing the format again. If the problem persists, try commenting out every line between the header line and the END statement, then add lines one at a time to see which lines are at fault.

   After you have isolated the portion of the format that does not work, examine it for errors or test it by itself in a simpler format.

NOTE: When examining label formats, look for instances where the letters "O" or "I" have been incorrectly entered for the numbers 0 and 1. These are very common typographical errors.

Common Issues

Printer does not respond to incoming data

1. Cause: The label format does not terminate with a proper END command.

   Cure: Terminate all label formats with an END command, unless specifically instructed otherwise by a particular command description. Terminate every format line with a carriage return and line feed.
2. Cause: The host system is not sending pure ASCII data.

   Cure: Check your system documentation to confirm that the host is sending ASCII data. Some software uses formatting that is incompatible with ASCII driven devices.
3. Cause: The label format contains an incompatible command. Some commands will only work with certain printers.

   Cure: Rewrite the label format as required, removing incompatible commands.
4. Cause: The label format contains extraneous control characters, such as "@" or "#."

   Cure: Be careful when using reserved characters in label formats. You can print the @ sign and other reserved characters, but if you use such characters at the beginning of a line the printer may "hang."
5. Cause: If you are sending data to a 4.25" portable printer it may be asleep.

   Cure: Precede every label format sent to a 4.25" portable with the proper wake-up string.

Printer prints the label format text rather than the intended label

1. Cause: You are sending the label format to the printer from Microsoft Windows, and the current printer driver is converting the label format to ASCII commands (i.e., you are using the Cognitive Intellidriver or a similar printer driver).

   Cure: Use the "generic ASCII" driver when sending label formats to the printer from Windows.

   Cure: Instead of printing the files from Windows, invoke MS-DOS and copy the files to the printer port using the DOS COPY command.

Printer prints one label, then stops

1. Cause: The label format contains a HALT command.

   Cure: Remove the HALT command, or press the printer's feed button to print the next label.
2. Cause: The printer's Label Taken mode is enabled, in printers so equipped.

   Cure: Remove each label from the printer as it is finished.

Printed labels are blank

1. Cause: Possible hardware problem.

   Cure: Refer to the printer User's Guide.
2. Cause: The print width is too narrow for the label format design.

   Cure: Correct the print width setting with a WIDTH or VARIABLE WIDTH command.
3. Cause: The label image has been shifted out of the printable area with a VARIABLE SHIFT LEFT command.

   Cure: Send another VARIABLE SHIFT LEFT command to correct the label position.

Printer prints labels, but print quality is poor

1. Cause: Possible hardware problem.

Cure: Refer to the printer User's Guide.

2. Cause: Incorrect print media (for example, using direct thermal paper while printing in thermal transfer mode).

   Cure: Load the correct print media for your printer and application. Confirm that the software is setting the printer for the correct print method.
3. Cause: Using Value Grade media in a printer set up for standard media.

   Cure: Either switch to standard media, or set the printer up for Value Grade media.
4. Cause: Incorrect parameter in a VARIABLE DARKNESS command. This can happen when using a different printer than the one for which the format was originally written. Some older printers use a different value range for VARIABLE DARKNESS than currently-manufactured printers.

   Cure: Experiment with the VARIABLE DARKNESS command to find the best setting for your printer and print media.
5. Cause: Printer speed setting is too high. Some print media do not print well at high speed.

   Cure: Use the appropriate VARIABLE command to reduce printer speed, or try different media.

Print is stretched or shrunk vertically

1. Cause: Dot time in header line is incorrect.
2. Cause: Label format was written for a different (typically, an earlier or later model) printer.

   Cure: Try another value for dot time. If designing a turnkey software package, provide for user adjustment of dot time. Allow for printer limitations in this regard (for example, Barcode Blaster LS and Blaster Advantage printers only support dot time 100).
3. Cause: If programming a high speed Barcode Blaster, Blazer Emulation Mode may be enabled.

   Cure: Disable Blazer Emulation Mode unless your application strictly requires it.

Printer does not index properly

1. Cause: The printer received a NOINDEX command. NOINDEX remains in effect until the printer is turned off or receives an INDEX command.

   Cure: Send the printer an INDEX command. Confirm that your label format does not contain an unwanted NOINDEX.
2. Cause: For printers that can use both black bar and gap indexing, the printer may be set for the wrong index type.

   Cure: Change print media or set the printer up for the proper indexing mode. See the sample printer configuration files for more information.
3. Cause: The printer index detector needs calibration.

   Cure: If you are using a Code Courier printer, calibrate the index detector according to the instructions in your User's Guide. If using a Barcode Blaster or Blaster Advantage, refer to the VARIABLE INDEX SETTING command for index calibration information.
4. Cause: Butt-cut label stock is loaded (the printer may be indexing properly, but with butt-cut stock the labels may not index to the label perforation, depending on the black bar position).

   Cure: None. This is a normal condition. Use gap stock if this behavior is unacceptable.

NOTE: The Del Sol’s lower index sensor must be in the center position for Gap indexing.

Printer skips labels during printing

1. Cause: Max Y value in header line is too large.

   Cure: Recalculate Max Y value.
2. Cause: Dot time in header line is too large.

   Cure: Reduce dot time.
3. Cause: Label format was designed for a different printer model or for a printer with different dot time behavior.

Cure: Adjust the label format accordingly.

Printer prints the wrong number of labels

1. Cause: The header line specifies the wrong number of labels.

   Cure: Correct the header line. Remember that when using the MULTIPLE command, the number of labels specified in the header line is the total quantity, counting side-by-side duplicates, not the number of vertical forms fed.

Label processing stops unexpectedly

1. Cause: The printer has stopped processing due to a hardware problem.

   Cure: Consult your printer's User's Guide.
2. Cause: The printer has encountered a bad label format in a series of formats within one file.

   Cure: Isolate the bad label format by copying each format to a different file and testing it individually. After isolating the bad format, troubleshoot as usual. Reassemble the whole file and try it again after the bad format is working.

Printer will not print at high speed

1. Cause: The printer has received a VARIABLE LOWSPEED or VARIABLE NORMAL command.

   Cure: Send the printer a VARIABLE HIGHSPEED command.
2. Cause: If you are using a high speed Barcode Blaster, the printer may have received one of the following commands or command parameters:

   ```
   MULTIPLE
   OFFSET
   WIDTH
   VARIABLE MODE 1 or VARIABLE MODE 2
   ```

   Nonzero header line X parameter

   These will cause the printer to reduce speed while printing.

   Cure: Rewrite your label format to avoid these commands.

Cutter-equipped printer does not cut labels

1. Cause: VARIABLE AUTOCUT is OFF, and the label formats do not contain a HALT command.

   Cure: Send the printer a VARIABLE AUTOCUT ON command, or place a HALT command in each label format.

Random dots on label, or printed labels are truncated at the bottom

1. Cause: The label format exceeds printer memory limits.

   Cure: Rewrite the format to use less memory.

Printed label is greatly elongated and is "grayed"

1. Cause: The label format contains more than one PITCH command.

   Cure: Remove any extraneous PITCH commands.

Numeric values do not increment properly in response to the ADJUST command

1. Cause: The numeric value you wish to adjust has more digits than the parameter specified in the ADJUST command.

   Cure: Pad the ADJUST parameter with leading zeros as required.
2. Cause: You are trying to adjust a numeric value to a number with more digits than the original value.

   Cure: Pad the original value with leading zeros as required.
3. Cause: You are using a MULTIPLE command with ADJUST.

   Cure: This is acceptable, but remember: ADJUST will only increment the adjusted numeric value when the printer feeds a label. Duplicate labels printed side-by-side will always be identical.
4. Cause: You are trying to increment numeric data that does not appear at the end of a command line.

Cure: This may be acceptable, but the ADJUST command can only increment numeric data. If alpha characters follow the numeric data, you must follow the ADJUST parameter with enough zeros to pad the increment value away from the alpha characters.

Bar code or string positioning is incorrect

1. Cause: A required space was left out or an extra space was inserted somewhere in the BARCODE or STRING command, causing the printer to interpret the parameters incorrectly.

   Cure: Use blank spaces as shown in the command descriptions.

Ultra Font or TEXT font positioning is incorrect

1. Cause: The label format has a JUSTIFY command in it that you have not accounted for.

   Cure: Each JUSTIFY command in a format will remain in effect for the rest of the format or up to the next JUSTIFY. Insert new JUSTIFY commands where required.

STRING or TEXT fonts don' t work

1. Cause: The memory area containing the STRING or TEXT fonts has been unintentionally initialized, erasing the fonts. See the INITIALIZE STORAGE command for more information.

   Cure: You will have to reload the fonts into nonvolatile RAM. Contact the factory for assistance.

Printed bar codes won' t scan

1. Cause: The bar code block is located too near the label edge, or too close to another label component.

   Cure: Redesign the label for better component positioning.
2. Cause: You are trying to encode characters that the selected bar code type will not support.

Cure: Use the information here and in the printer's User's Guide to assure that you are following the rules of the selected bar code symbology.

FILL_BOX does not produce reversed (black to white) text as expected

1. Cause: The order of commands in the label format is incorrect.

   Cure: Rearrange the order of commands in the label format as required. FILL_BOX must follow the commands that define the label component you want to reverse.

The printer ignores some commands in the label format

1. Cause: There is an illegitimate command in the label format. Some printers interpret unknown commands as END commands.

   Cure: Check the label format syntax, especially that of the first ignored command.
2. Cause: The label format was written for a different printer model or firmware revision, and the ignored commands are not supported by your printer.

   Cure: Rewrite the label format as required.

Graphics Programming Issues

Graphics programming is inherently more complex than ASCII programming, but is subject to the same problems. Resolve any ASCII programming problems before spending too much time troubleshooting graphics files.

If you can successfully print ASCII format files, try printing a proven graphics label format file. If a proven graphics file will not print, suspect a printer/host communications problem. Continue troubleshooting using the information below if a proven graphics file prints satisfactorily.

Printer does not print graphics

1. Cause: You are not sending pure binary data to the printer.

Cure: Do not try to send graphics data to the printer using a text editor or word processor. Text editors format their output as printable characters rather than pure binary. Program graphics using software that is designed to handle pure binary data.

2. Cause: the graphics file does not start with the proper character.

   Cure: Always start foreground graphics files with the @ character (ASCII 64). Start background graphics files with the # character (ASCII 35). Labels programmed in the background do not print immediately. They remain in memory and print with a foreground label format.
3. Cause: Your graphics file does not contain enough data to fill the specified number of dot rows.

   Cure: Always send exactly enough data to fill the number of rows specified in the format header line. Consider reducing print width or use the LOGO command when printing small graphics.

Printer prints a random pattern instead of the expected bitmap

1. Cause: The printer is set for a different pitch or width than the graphics file was designed for.
2. Cure: Before sending graphics to the printer, send a dummy label format consisting of a header line, WIDTH command, PITCH command, and END statement. This assures that the printer is set properly before sending the graphics data.
3. Cause: The graphics file is too large for the available memory space.

   Cure: Calculate available printer memory and confirm that your graphics file will fit. Remember that the available memory is cut in half when using background graphics mode. If you are near the memory limits, you may have to rewrite the graphics file.

Random dots or lines appear on the finished label

1. Cause: The host system is appending unwanted end-of-line terminators to the graphics data.

   Cure: Print your graphics format to a file and then upload the file to the printer instead of sending the data directly to the printer.

Cure: Check your development platform’s documentation to see how you can control end-of-line termination.

2. Cause: The graphics file is too large for the available memory space.

   Cure: Rewrite the file as required.

Part of the graphics image is garbled or blank

1. Cause: The graphics file is too large for the available memory space.

   Cure: Rewrite the file as required.

Graphics labels print, but are reversed

1. Cause: Bitmap was mapped backwards.

   Recently-manufactured printers map the printhead from right to left. Some older printers map the printhead from left to right. Graphics files designed for older printers may print backwards when sent to a newer printer.

   Cure: When manually designing graphics files, calculate the data based on a mirror image of the original design. If you are designing a turnkey software package, take the possibility of bitmap reversal into account in your design.