BPL Programming Manual

Blazepoint DUO & TRIO Printers

Firmware:
Blazepoint DUO & TRIO Revision 1.38

Document:
Revision 4
26th November 2004

Blazepoint Ltd.

BPL Programming Manual

Applies to all DUO & TRIO printers having the firmware revision shown on the front page of this manual, or higher. Printer firmware may be upgraded at any time using the blazeConfig utility. The latest version of blazeConfig and printer firmware may be downloaded from the Blazepoint website address shown below.

Trademark Acknowledgements

Printer range is designed and manufactured in the United Kingdom by:

Blazepoint Ltd.
2 Tower Estate,
Chalgrove,
Oxford,
OX44 7XZ
United Kingdom.

Tel: +44 (0) 1865 892 030
Fax: +44 (0) 1865 892 031
Web: www.blazepoint.co.uk
Email: sales@blazepoint.co.uk

Blazepoint products are subject to continuous development and improvement, and consequently may incorporate minor changes from the information contained in this manual.

Contents

Introduction

The Blazepoint thermal printer family offers high-speed, low-noise, on-demand printing with a wide variety of media. The thermal transfer process permits printing onto a range of materials, in a variety of colours. Alternatively, direct thermal printing onto thermal paper offers the highest print speeds without the need for a transfer ribbon.

The printer will measure the label length automatically and synchronise to the label edge, or to an index mark on pre-printed paper. Variable length labels can be produced on continuous stationery.

The printer offers the following features:

- EAN-8, EAN-13, UPC-A, UPC-E, EAN/UPC 2 & 5 digit addendum, EAN-128, Code-128 (subsets A, B & C), ITF-14, Interleaved 2 of 5, Code39, Code93 and Codabar barcodes.
- PDF417, Data Matrix and MaxiCode 2-D codes.
- Bitstream® font-scaling technology, with 6 TrueType™ fonts supplied as standard.
- Line & box graphics with grey shading.
- Bit image graphics.
- Rotations of 0° - 359° for all fields individually.
- Variable & incrementing fields.
- Built-in file system to hold downloaded fonts, graphics, capture-files, filters and more.
- blazeCapture – store print files directly and replay later by pressing the Feed button.
- blazeFilter – reformat incoming data intended for other applications into BPL commands.
- Cash drawer driver, with sense input which can also be used to control printing remotely.

This manual contains information about programming the printer directly (i.e. without using a label design program) for direct connection to stock control systems, etc.

The BPL programming language is based on the GPL2 language originally developed for the Gazelle series of printers. With a few minor exceptions, BPL is backward compatible with the version of GPL2 used in earlier Compact, Qube and Gazelle printers, but now contains many more features as well. To take advantage of all the latest features it is recommended that you download and install the latest firmware from the Blazepoint website.

Programming Guide

Command Description Conventions

Various typefaces and styles are used in the summary descriptions of commands. The fictitious command below provides an example of the features used.

ExampleCommand Esc **E** xxxx **X** n… **M** o… d… Eot

Ascii control characters such as Escape and End-of-text are indicated using a small font. This is to distinguish the single character Esc from the 3 character sequence E-s-c. The Space character is included in this representation and is shown as Spc.

Fixed data is indicated in bold black type. These characters must be supplied exactly as shown.

Variable data which the programmer must supply, such as coordinate data, is shown in a slightly smaller font in blue. Variable data can take several forms as shown below.

xxxx	A repeated letter indicates a fixed number of characters. For numeric values, the data is a decimal number with leading zeros. In this example exactly 4 digits are required.
n…	n followed by 3 dots indicates a numeric value with a variable number of decimal digits
o…	o followed by 3 dots indicates optional parameters which are only supplied if required.
d…	d or t followed by 3 dots indicates general data or text of variable length. Normally there will be a fixed character terminator.

Variable data which defines a distance is assumed to be in units of pixels. Some values have different units as the default and in this case the default units will be indicated. Note that the default units can be overridden by setting a global unit of measurement using the SetUnits command.

A coding example is shown after the command description.

Example:
Select Swiss 721 bold, 24pt high, 20pt wide, not italic, no kerning.

Text	EscY0102402000
Hex	1B 59 30 31 32 34 30 32 30 30 30
Dec	27 89 48 49 50 52 48 50 48 48 48

This may be followed by further description and application details.

Units of Measurement

All coordinate values are in pixels (dots) by default, and most other measurements, e.g. form length, are in mm. This default system has been retained for compatibility with older printers, but it can be confusing and makes the programming dependent on the print resolution. In general it is better to select a global unit of measurement that will apply to everything. Some commands, e.g. graphic images must inherently be described in pixels, and these are not changed by the SetUnits command (although the graphic placement coordinates are).

SetUnits — Esc Z u

Sets the global units of measurement used by all subsequent commands.

Value for u	0	Original (default) units, as defined for each command.
	M	Millimetres
	2	0.5 mm
	m	0.1 mm
	h	0.01 mm
	P	Points (1/72 inch)
	i	0.01 inch
	t	0.001 inch (mil)
	D	Dots (square pixels at printhead resolution)
	A	Absolute dots. The same as Dots, unless the pixels are not square.

Example:
Select units of 0.1 mm.

Text	EscZm
Hex	1B 5A 6D
Dec	27 90 109

This affects all subsequent commands that include measurement or coordinate data. The default units for each command are shown in brackets, e.g. (pixels). Units of 0.1mm or 0.01 inch are recommended for most applications, as it is easier to use and makes the label design less dependent on printhead resolution.

Stationery Type and Label Size

The printer has two basic modes of operation depending on the stationery type.

In label mode, the stationery is considered to be divided into fixed lengths and printing must be referenced to top-of-form. The stationery could be a roll of labels, perforated paper, or pre-printed tickets. The printer measures the distance between index marks or label gaps to determine the form length, and always aligns with respect to the top of the label. If a label gap or index marks exceeds the measured length by 5 mm, it will be interpreted as a Paper Out fault and printing will stop.

In continuous mode, the stationery has no gaps or index marks and the form length is specified by programming. Labels of any length can be printed in this mode. If any gap or index mark is detected, it will be interpreted as a Paper Out fault and printing will stop.

The power-on default is set using the configuration parameters, but can also be changed during normal operation using the SetLabelMode and SetContinuousMode commands.

In label mode, the printer measures the formlength automatically, but in continuous mode, the printer must be told what length of stock to feed when it receives a formfeed command. In both cases the printer uses as many whole forms as are required to print the entire image. Setting a very short formlength in continuous mode, eg. 1 mm, means that the paper will stop just after the last field has been printed.

SetLabelMode `Ctrl-T`

Select label or ticket stock.

Text	`Ctrl-T`
Hex	14
Dec	20

SetContinuousMode `Ctrl-R`

Select continuous stock.

Text	`Ctrl-R`
Hex	12
Dec	18

SetFormLength `Esc` **A** yyyy

Sets the form length to use in continuous stock mode.

yyyy Formlength to use (mm)

Example:
Set form length to 40mm

Text	`Esc`A0040
Hex	1B 41 30 30 34 30
Dec	27 65 48 48 52 48

This command has no effect in label mode, as the label length is measured automatically.

SetVariableFormLength Esc l yyyy

Form length is the printed area plus a bottom margin in continuous stock mode.

yyyy	Bottom margin to use (mm)

Example:
Set bottom margin to 20mm

Text	Esc10020
Hex	1B 6C 30 30 32 30
Dec	27 108 48 48 50 48

When the last line of the image has been printed, the printer adds this set distance as a bottom margin. This command has no effect in label mode, as the label length is measured automatically.

SetTOFoffset Esc P yyyy

Sets the Top-of-Form offset to use in label mode.

yyyy	Y offset (mm) A leading '-' sign before the 4 digits is permitted.

Example:
Set TOF offset to -20mm

Text	EscP-0020
Hex	1B 50 2D 30 30 32 30
Dec	27 80 45 48 48 50 48

In label mode, this sets the Y=0 offset below the physical top-of-form marker used by the label sensor. This is also the position that the printer will stop at (unless paper-advance / backfeed is in operation). This is not normally used with labels, but some ticket stock may have the black mark positioned part way down the ticket, so an offset is required to make Y=0 coincide with the top of the ticket rather than the black mark. Setting a positive offset means that printing will start further down the label/ticket. In most applications, this facility is not required.

A negative TOF offset positions the ticket so that the Y=0 start-of-print position is above the physical top-of-form marker. This may be needed with tickets where the black mark is printed a short distance below the top of the ticket.

This command has no effect in continuous mode, as there is no concept of form alignment or top-of-form.

Media Type

Selecting the media mode has 2 effects. Firstly it enables the ribbon tacho sensor in thermal transfer mode, and secondly it sets the print heat scale factor. The factory default value of 100% produces a higher heat in direct thermal mode than it does in thermal transfer mode, so wax ribbons and typical thermal paper can both be used with the default heat setting of 100%.

The default setting at power on is loaded from the printer configuration which can be changed using blazeConfig.

SetMediaType Esc q n

Select direct thermal or thermal transfer mode at print time.

n	0 = Direct thermal 1 = Thermal transfer

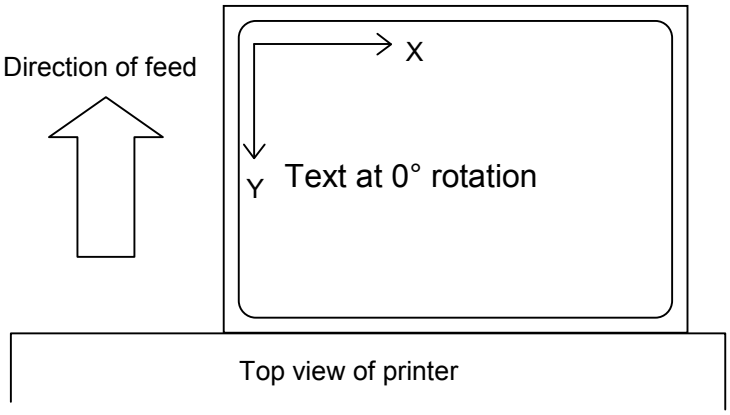
Example:
Select thermal transfer mode

Text	Escq1
Hex	1B 71 31
Dec	27 113 49

Coordinate System and Field Rotation

Printing is always page oriented. Text, barcodes, graphics, etc are placed on the page using an (x,y) coordinate system. The X = 0 datum is the paper edge which slots into the paper sensor, on the right hand edge of the print head when looking at the front of the printer (the left hand edge, from the printer's viewpoint). The Y = 0 datum is the top-of-form. Note that on all Blazepoint printers, the label emerges head-first.

The diagram below shows the X and Y coordinate directions on a label emerging from the printer.



SetRotation Esc V r

Sets the rotation angle to use for subsequent field placement.

Value for r	1	0° rotation
	2	90° rotation
	3	180° rotation
	4	270° rotation

Example:
Set rotation to 180°

Text	EscV3
Hex	1B 56 33
Dec	27 86 51

This command has no effect on block fill fields, for backward compatibility. To rotate a block fill field use the SetRotationAngle command instead.

SetRotationAngle Esc V 0 rrr

Sets the rotation angle to use for subsequent field placement.

rrr	Rotation angle in degrees, 000 - 359

Example:
Set rotation to 45°

Text	EscV0045
Hex	1B 56 30 30 34 35
Dec	27 86 48 48 52 53

Image Size and Position

Blazepoint printers are fitted with a large memory capacity as standard, which means that under normal conditions the image is always multiple buffered. This means that while the current label is printing, the next one can be prepared in advance in a separate area of memory, ready to print as soon as the current label has finished. Rather than allowing the level of buffering to be specified, which makes the programming dependent on the memory fitted, the label height is specified and the printer sets the optimum level of buffering accordingly. By default the label height is at least 500 mm.

When narrow stock is used, it should be aligned with the middle of the print head to achieve a mechanical balance of the head and reduce roller wear. To simplify positioning of fields, you can apply an X-offset which will be added to the X-coordinate of every field. Alternatively, if auto-centre mode is enabled, you can specify the label width and the X-offset will be calculated automatically.

SetLabelHeight — Esc * n… H

Sets the image buffering level according to the label height required.

n…	Label height (mm) 0 = default allocation

Example:
Set label height to 1200 mm

Text	Esc*1200H
Hex	1B 2A 31 32 30 30 48
Dec	27 42 49 50 48 48 72

SetNumBuffers — Esc * n… N

Sets a specific image buffering level.

n…	Number of buffers to allocate. 0 = Automatic allocation (default)

Example:
Set 2 image buffers

Text	Esc*2N
Hex	1B 2A 32 4E
Dec	27 42 50 78

This command is normally only used for test purposes, or to force single buffering. The old commands for single and double buffer modes have been removed because single buffer mode was being selected to achieve maximum label height, which is **not recommended**. To allocate buffers for a specific label height use the SetLabelHeight command instead.

SetFieldXoffset

Esc **wX** xxxx

Sets the X offset value to be applied to subsequent field placement.

xxxx	X offset (mm)

Example:
Set X offset to 25mm

Text	EscwX0025
Hex	1B 77 58 30 30 32 35
Dec	27 119 88 48 48 50 53

SetAutoCentre

Esc ***** n… **Y**

Turn Auto-Centre mode on or off.

n…	0 = Off 1 = On

Example:
Enable auto-centre

Text	Esc*1Y
Hex	1B 2A 31 59
Dec	27 42 49 89

SetLabelWidth

Esc ***** n… **W**

When auto-centre is enabled, this calculates the X-offset to apply to ensure that the label is centred within the print width.

n…	Label width (mm) 0 = Full printhead width

Example:
Set label width to 60 mm

Text	Esc*60W
Hex	1B 2A 36 30 57
Dec	27 42 54 48 87

Placing Text

The printer incorporates the Bitstream® FontFusion processor, which provides independent scaling in vertical & horizontal directions (3 - 999 Points) and italic simulation. To provide backward compatibility with the Blazepoint Compact printer, emulation is also included for the original bitmap fonts, as these are no longer supplied. For all new applications, use of scalable fonts is recommended, as there is far greater flexibility.

PlaceText — Esc T xxxx yyyy t… Eot

Places a text field in the current font and rotation at the coordinates specified.

xxxx	X coordinate
yyyy	Y coordinate
t…	Text, terminated with the Eot character

Example:
Place "Hello" at (10, 30)

Text	EscT00100030HelloEot
Hex	1B 54 30 30 31 30 30 30 33 30 48 65 6C 6C 6F 04
Dec	27 72 48 48 49 48 48 48 51 48 72 101 108 108 111 4

The text is placed at (x,y) in the current font and rotation. The text must consist of valid printable characters, but may also include tabs and newlines. Tabs and newlines are referenced to the placement coordinates. Tabs are set up at a regular spacing, 12.5 mm by default, although the spacing can be set with the SetTabWidth command. A newline will return to the start of the next line down, vertically aligned with the placement coordinates.

SetScalableFont — Esc Y vvv hhh i k

Selects the scalable font parameters.

vvv	Vertical point size (character height)
hhh	Horizontal point size (character width)
i	0 = normal 1 = italic simulation (applies a slant)
k	0 = none Note: Kerning is not currently supported.

Example:
Select Swiss 721 Bold, 24 pt high, 20 pt wide, not italic, no kerning

Text	EscY0102402000
Hex	1B 59 30 31 30 32 34 30 32 30 30 30
Dec	27 89 48 49 48 50 52 48 50 48 48 48

Horizontal and vertical point sizes may be set independently. Font sizes are always measured in points regardless of the default units in effect. However, the PointSizeScaling command can be used to give better resolution.

Six fonts are pre-installed, as shown in the table below, but additional fonts may be installed using the blazeConfig utility.

Scalable Font Number	TrueType™ Typeface	Font Filename	Similar Typefaces	Print Sample
00	Swiss™ 721	tt0003m_.TTF	Arial®	ABC abc 123 %£$?;
01	Swiss™ 721 bold	tt0005m_.TTF	Arial® bold	**ABC abc 123 %£$?;**
02	News 701	tt0318m_.TTF	Times New Roman®	ABC abc 123 %£$?;
03	Impress	tt0209m_.TTF		ABC abc 123 %£$?;
04	Monospace 821	tt0596m_.TTF	Arial® Monospace	ABC abc 123 %£$?;
05	OCR-B	tt0646m_.TTF		ABC abc 123 %£$?;

PointSizeScaling Esc * n… P

Set the scale factor to apply to point size.

n… Scale factor

Example:
Set font scaling to be in units of 0.1 point, then select font 0 at point size 8.5

Text	Esc*10P EscY0008508500
Hex	1B 2A 31 30 50 20 1B 59 30 30 30 38 35 30 38 35 30 30
Dec	27 42 49 48 80 32 27 89 48 48 48 56 53 48 56 53 48 48

Point size selected in the SetScalableFont command is the value specified divided by this scale factor. If point-size scaling is set to 10, then the size is interpreted as being in units of 0.1 points.

SetInlineText Esc * n… T

Selects whether unformatted text is printed.

Value of n…	0	Print only formatted text (default)
	1	Print unformatted text at current cursor position.

Example:
Enable inline text

Text	Esc*1T
Hex	1B 2A 31 54
Dec	27 42 49 84

By default, plain text sent to the printer is not printed, and only the PlaceText command can be used. If InlineText is enabled, the plain text is placed on the label at the current cursor position. An empty PlaceText command can be used to set a specific cursor position if required.

SetCodePage Esc * n… f

Selects the DOS or Windows code page number.

Value of n…	1252	Windows Latin-1 (Western)
	1250	Windows Latin-2 (Eastern Europe)
	1253	Windows Greek
	437	DOS English
	850	DOS Latin-1
	852	DOS Slavic
	860	DOS Portugese
	863	DOS French / Canadian
	865	DOS Nordic
	0	UTF-8

Example:
Select codepage 1250

Text	Esc*1250f
Hex	1B 2A 31 32 35 30 66
Dec	27 42 49 50 53 48 102

Code pages are used to map the 256 characters which are available with an 8-bit character value to the multiplicity of international characters available. Code page 1252 is the most widely used in Windows. UTF-8 is commonly used in Linux.

SetTabWidth Esc U hhh

Selects the scalable font parameters.

hhh	Horizontal tab width (mm)

Example:
Set tab width to 2 cm

Text	EscU020
Hex	1B 55 30 32 30
Dec	27 85 48 50 48

Sets the width of embedded tabs for the PlaceText command. In general, using tabs is not recommended. It is better to place text fields individually, or use a monospaced font to get text to align in columns.

SetTextBaseline Esc Z b

Set the anchor point for subsequent text placements using the PlaceText command.

Value for b	T	Use top of character cell (default).
	B	Use font baseline (recommended).

See SetLeftRightAlignment for an example.

The top of character cell refers approximately to a point at the top of the tallest character in the font, eg. the top of the 'A' character. Font baseline refers to the line on which characters would be drawn, eg. the base of the 'A' character. Baseline is generally easier to work with and more consistent between typefaces.

SetLeftRightAlignment Esc Z a

Set the anchor point for subsequent text and barcode placements.

Value for a	L	Use left corner of text string or barcode (default)
	C	Use centre of text string or barcode
	R	Use right corner of text string or barcode

Example:
Use centre of baseline as anchor point for subsequent text strings.

Text	EscZBEscZC
Hex	1B 5A 42 1B 5A 43
Dec	27 90 66 27 90 67

Using left-aligned strings is the most efficient. With centred and right-justified text, the printer has to measure the width of every character in the string before placing the string in the image, which increases the time taken. For barcodes, the alignment method does not affect placement speed.

SetBitmapFont Esc F n c

Selects the bitmap font (emulation) to use (obsolete command).

n	Font number (0-9)
c	Codepage

Example:
Select bitmap font 3, Windows codepage.

Text	EscF0W
Hex	1B 46 30 57
Dec	27 70 48 87

Fonts 1-9 are emulations of the earlier bitmapped fonts and are **not recommended** for new applications. Font 0 selects scalable fonts that are set up using the SetScalableFont command. Bitmap emulation text is only available at the preset point sizes shown. The country code selects the ASCII code mapping. The default font and country code are '1 W'. To improve compatibility with older printers, the point size of the font does not exactly match the size quoted, and the space width is also adjusted.

Font Number	Point Size
0	Scalable fonts
1	6 point
2	8 point
3	10 point
4	12 point
5	16 point
6	20 point
7	24 point
8	32 point
9	40 point

Country Code	Name	Code Page
W	Windows Latin-1	1252
E	English	437
M	Multilingual	850
S	Slavic	852
P	Portuguese	860
C	Canadian	863
F	French	863
N	Nordic	865
L	Windows Latin 2	1250
G	Windows Greek	1253

SetMagnification Esc M vv hh

Set the magnification factor to apply to bitmap fonts and logos.

vv	Vertical magnification factor
hh	Horizontal magnification factor

Example:
Set x3 vertical and x2 horizontal magnification.

Text	EscM0302
Hex	1B 4D 30 33 30 32
Dec	27 77 48 51 48 50

Magnification makes logo placement much slower, and produces poor quality images due to the larger logo pixels produced. Using this command is therefore **not recommended**.

Placing 1-D Barcodes

The printer supports all the most commonly used barcode types. This section deals with the plain 1-dimensional codes which consist of a single row of vertical bars. The printer also supports a range of 2-dimensional codes which are documented in the *Placing 2-D Barcodes* section. The general command format for barcode placement is as follows :-

PlaceBarcode Esc **B** xxxx yyyy t hh o… d…

Place a barcode (generic command – see individual barcodes for full details).

xxxx	X placement coordinate
yyyy	Y placement coordinate
t	Barcode type reference (see following sections)
hh	Bar height (mm)
o…	Optional parameters
d…	Barcode data

By default, the X and Y placement coordinates refer to the top left corner of the barcode. However, you can change this behaviour using the SetLeftRightAlignment and SetBarAlignment commands.

The bar height is specified as 2 digits by default, which is often insufficient if small unit of measurement are in use. You can increase this to 4 digits using the SetExtendedBarHeight command, which is recommended for all new applications.

In many of the barcode types, the international standard specifies a preferred height for the bars. Setting the bar height to 0 will select the correct height automatically.

Information on the specific barcode types are described in more detail later.

SetBarcodeModules Esc **N** t m w n

Select the overall magnification factor, and the wide and narrow bar widths.

t	Barcode type reference
m	Magnification (1-9)
w	Wide module width multiplier (1-9, A-F)
n	Narrow module width multiplier (1-9, A-F)

Example:
Set Code39 to Mag=2, Wide=3, Narrow=1..

Text	EscN6231
Hex	1B 4E 36 32 33 31
Dec	27 78 54 50 51 49

Sets the magnification factor and the wide & narrow module widths for barcodes. Wide and narrow module widths can only be set for types 1, 4, 6 & 7. Supplying a value of 0 for any parameter leaves that parameter unchanged. e.g. <Esc> N 4200 sets the magnification of type 4 (Interleaved 2 of 5) to 2, but leaves the module widths unchanged.

Module widths are always specified in pixels regardless of the global units of measure. The actual module width in pixels is the product of (Magnification x WidthMultiplier).

SetBarExtendedHeight Esc N h

Selects whether to use 2 digit or 4 digit height fields for barcodes.

Value for h	x	Use 2 digit height (default)
	X	Use 4 digit height (recommended)

Example:
Use 4 digit heights for all barcodes

Text	EscNX
Hex	1B 4E 58
Dec	27 78 88

Sets the number of digits used for barcode height in all barcode types. Using 4 digits allows smaller units of measurement to be used without restricting the barcode height. This mode is recommended for all new applications.

EnableBarcodeHRI Ctrl-B

Enable human-readable interpretation for all barcode types.

Text	Ctrl-B
Hex	02
Dec	2

DisableBarcodeHRI Ctrl-C

Disable human-readable interpretation for all barcode types.

Text	Ctrl-C
Hex	03
Dec	3

Human-readable interpretation text is enabled for all barcodes by default.

SetBarAlignment Esc NZ v

Set the vertical anchor point for subsequent barcode placements.

Value for v	T	Use top edge of barcode (default)
	C	Use centreline of barcode
	B	Use bottom edge of barcode

Example:
Use bottom right corner as anchor point for subsequent barcodes.

Text	EscNZBEscZR
Hex	1B 4E 5A 42 1B 5A 52
Dec	27 78 90 66 32 90 82

SetBarMargins Esc NZ m

Control whether barcode coordinates refer to the edge of the light margin, or the edge of first bar.

Value for v	M	Use default light margins (default)
	m	No light margins on any barcode type.

Example:
Suppress light margins for all subsequent barcodes.

Text	EscNZm
Hex	1B 4E 5A 6D
Dec	27 78 90 109

Some barcode types such as EAN13 have mandatory light margins. By default, the placement coordinates assume that the edge of the barcode is actually the edge of the light margin rather than the first bar. It is often easier to refer to the visible bars, rather than the invisible light margin, so this option is provided to suppress light margin referencing for all barcode types.

PlaceITF14 Esc B xxxx yyyy 1 hh o… d…

Place an ITF14 barcode with bearer bars.

Content of o…	(Nothing)	Original spec, 12 digits data, leading zero will be added.
	I	International version, 13 digits data, leading digits must be supplied (recommended).
d…		Barcode data, fixed number of digits (see above)

Example:
Place international ITF-14 at (25, 80), default height, data 3501234567890.

Text	EscB00250080100I3501234567890
Hex	1B 42 30 30 32 35 30 30 38 30 31 30 30 49 33 35 30 31 32 33 34 35 36 37 38 39 30
Dec	27 66 48 48 50 53 48 48 56 48 49 48 48 73 51 53 48 49 50 51 52 53 54 55 56 57 48

The ITF-14 code is a fixed length code used for distribution marking of packages, and consists of 14 digits. The last digit is a checksum which will be generated automatically. For all new applications, the international version of this command is recommended, which takes the full 13 digits of data. The original version is retained as the default for backward compatibility reasons; it takes only 12 digits and adds a leading zero. The original 12-digit version is **not recommended** for new applications.

PlaceEAN13 — Esc B xxxx yyyy 2 hh o… d…

Place an EAN13 barcode.

Content of o…	(Nothing)	Standard EAN13. Requires 12 digits of data.
	T	EAN13 with 2-digit addendum. Requires 14 digits.
	F	EAN13 with 5-digit addendum. Requires 17 digits.
	N	Suppress the final '>' quiet zone char in HRI.
d…		Barcode data, fixed number of digits (see above)

Example:
Place standard EAN13 at (100, 38), default height, data 501234567890.

Text	EscB0100003820050123456789 0
Hex	1B 42 30 31 30 30 30 30 33 38 32 30 30 35 30 31 32 33 34 35 36 37 38 39 30
Dec	27 66 48 49 48 48 48 48 51 56 50 48 48 53 48 49 50 51 52 53 54 55 56 57 48

The EAN13 barcode is a fixed length numeric code, widely used throughout Europe and Japan for retail product marking. The standard barcode is the most common, and takes 12 digits of data. The check digit is calculated automatically by the printer.

The 2 and 5 digit addendum options are used mainly by publishers to create ISBN barcodes, and to mark the issue number or price of periodicals. The additional digits must be supplied immediately after the standard 12 digits.

PlaceEAN8 — Esc B xxxx yyyy 3 hh o… ddddddd

Place an EAN8 barcode.

Content of o…	(Nothing)	Standard EAN8
	N	Suppress quiet zone chars < and >.
ddddddd		Barcode data, exactly 7 digits.

Example:
Place EAN8 at (1020, 450), height 20, data 1234567.

Text	EscB102004503201234567
Hex	1B 42 31 30 32 30 30 34 35 30 33 32 30 31 32 33 34 35 36 37
Dec	27 66 49 48 50 48 48 52 53 48 51 50 48 49 50 51 52 53 54 55

The EAN8 barcode is a fixed length numeric code, used for retail product marking where the product is too small to hold a full size EAN13 barcode. The check digit is calculated automatically by the printer.

PlaceInt2of5 — Esc B xxxx yyyy 4 hh o… d… q

Place an interleaved 2-of-5 barcode.

Content of o…	(Nothing)	No check digit.
	C	Append Mod10 check digit.
d…		Barcode digits, terminated with 'q'.

Example:
Place ITF barcode at (85, 70), height 30, data 107634, append checksum.

Text	EscB008500704 30C107634
Hex	1B 42 30 30 38 35 30 30 37 30 34 33 30 43 31 30 37 36 33 34
Dec	27 66 48 48 56 53 48 48 55 48 52 51 48 67 49 48 55 54 51 52

Interleaved 2-of-5 is a compact variable length code for numeric data only. Up to 50 digits may be included. This barcode type must inherently have an even number of digits, so if the number supplied is odd, a leading zero will be added automatically. The 'C' option will calculate and append an AIM standard modulo-10 check digit.

PlaceCode39 — Esc B xxxx yyyy 6 hh o… d… q

Place a Code39 barcode.

Content of o…	(Nothing)	No check digit, start & stop codes not shown in HRI.
	c	Append Mod43 check character, shown in HRI
	h	Append Mod43 check character, not shown in HRI
	s	Show start and stop characters in HRI
d…		Barcode data, terminated with 'q'.

Example:
Place Code39 barcode at (0, 0), height 20, data ABC123, append checksum, show start/stop codes and checksum in HRI

Text	EscB00000000620csABC123q
Hex	1B 42 30 30 30 30 30 30 30 30 36 32 30 63 73 41 42 43 31 32 33 71
Dec	27 66 48 48 48 48 48 48 48 48 54 50 48 99 115 65 66 67 49 50 51 113

Code39 (also known as 3 of 9) is a variable length alphanumeric barcode, widely used throughout industry. It is robust, but not very compact.

Up to 50 data characters may be included, terminated with 'q'. The character set consists of all the uppercase letters, numbers and also dash, stop, space, dollar, oblique, plus & percent :-
0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ - . Space $ / + %

Code39 is used to construct HIBC (Health Industry Bar Code).

PlaceCodabar Esc B xxxx yyyy 7 hh o… d… q

Place a Codabar barcode.

Content of o…	(Nothing)	No check digit, start & stop codes not shown in HRI.
	@, A..O	Select start/stop characters (see table below)
	c	Append Mod16 check character, shown in HRI
	h	Append Mod16 check character, not shown in HRI
	s	Show start and stop characters in HRI
	f	Use fixed inter-character gap of 1 narrow space (recommended).
d…		Barcode data, terminated with 'q'.

Example:

Place Codabar barcode at (40, 78), height 15, Start=B, Stop=B, data 12345, append checksum shown in HRI.

Text	EscB00400078715Ecf12345q
Hex	1B 42 30 30 34 30 30 30 37 38 37 31 35 45 63 66 31 32 33 34 35 71
Dec	27 66 48 48 52 48 48 48 55 56 55 49 53 69 99 102 49 50 51 52 53 113

Codabar is an old and not very robust variable length barcode, although common in some industries. Up to 50 data characters may be sent, terminated with 'q'. The character set consists of all the numbers and also dash, dollar, colon, oblique, stop & plus :-
0 1 2 3 4 5 6 7 8 9 - $: / . +

The start and stop characters can be selected using the optional format character in the range @ to O. Refer to the table below.

Format character	Start code	Stop code
@	A	A
A (default)	A	B
B	A	C
C	A	D
D	B	A
E	B	B
F	B	C
G	B	D

Format character	Start code	Stop code
H	C	A
I	C	B
J	C	C
K	C	D
L	D	A
M	D	B
N	D	C
O	D	D

PlaceEAN128 Esc B xxxx yyyy 8 hh d… Eot

Place an EAN128 barcode.

d…	Barcode data terminated with Eot

Example:
Place EAN128 at (20, 45), height 20, Quantity=21, Batch=123456.

Text	EscB00200045820(30)21Gs(10)123456Eot
Hex	1B 42 30 30 32 30 30 30 34 35 38 32 30 28 33 30 29 32 31 1D 28 31 30 29 31 32 33 34 35 36 04
Dec	27 66 48 48 50 48 48 48 52 53 56 50 48 40 51 48 41 50 49 29 40 49 48 41 49 50 51 52 53 54 4

EAN128 is a data standard which uses Code128 as the underlying barcode standard. The data standard is available from AIM international and describes in detail the large number of different data types that can be encoded. The standard includes many rules for the representation and order of presentation of fields. It is important to refer to the standard to ensure that the barcodes produced are correctly formatted.

The character set consists of the printable ascii range (decimal 32-127, hex 20-7F), and up to 50 data characters may be encoded.

The leading Fnc1 character mandated by EAN128 is added automatically. EAN128 also requires the Fnc1 character to be used to terminate variable length fields. In this case the Gs character (decimal 29, hex 1D) should be sent. This character will not appear in the HRI.

Parentheses (round brackets) are often used to mark the application identifiers in the HRI. Parentheses will be stripped out of the barcode data, but retained in the HRI.

PlaceCode128 — Esc B xxxx yyyy 9 hh o… nn d… Eot

Place a Code128 barcode.

o…	'N' character prevents Codeset C optimisation from taking place. This effectively forces Codeset B only.
nn	Exactly 2 digits specifying the number of data characters following. If set to zero, the data must be terminated with Eot
d…	Barcode data, terminated with Eot if nn=0.

Example:
Place Code128 at (90, 220), height 20, Data "Code128 Barcode"

Text	EscB0090022092000Code128 BarcodeEot
Hex	1B 42 30 30 39 30 30 32 32 30 39 32 30 30 30 43 6F 64 65 31 32 38 20 42 61 72 63 6F 64 65 04
Dec	27 66 48 48 57 48 48 50 50 48 57 50 48 48 48 67 111 100 101 49 50 56 32 66 97 114 99 111 100 101 4

Code128 is a compact variable length barcode, widely used worldwide. The character set consists of the full ascii range (decimal 0-127, hex 00-7F), and up to 50 data characters may be encoded.

Codesets A, B and C are selected automatically to produce the optimum barcode density. The number of data bytes can normally be set to 00 and the barcode data terminated with Eot. If the Eot character itself is required in the barcode data, then the number of data bytes in the barcode must be specified, and Eot will be encoded, rather than terminating the data. If an 'N' character is sent before the byte count, this suppresses CodeSet-C optimisations. For normal alpha numeric data, this will force the use of CodeSet-B, even where there are long strings of digits.

PlaceUPCA — Esc B xxxx yyyy A hh o… d…

Place an UPC-A barcode.

Content of o…	(Nothing)	Standard UPCA. Requires 11 digits of data.
	T	UPCA with 2-digit addendum. Requires 13 digits.
	F	UPCA with 5-digit addendum. Requires 16 digits.
d…		Barcode data, fixed number of digits (see above)

Example:
Place standard UPCA at (100, 38), default height, data 01234567890.

Text	EscB01000038A0001234567890
Hex	1B 42 30 31 30 30 30 30 33 38 41 30 30 30 31 32 33 34 35 36 37 38 39 30
Dec	27 66 48 49 48 48 48 48 51 56 65 48 48 48 49 50 51 52 53 54 55 56 57 48

The UPCA barcode is a fixed length numeric code, used worldwide for retail product marking, although Europe and Japan generally favour EAN13. The standard barcode is the most common, and takes 11 digits of data. The check digit is calculated automatically by the printer.

The 2 and 5 digit addendum options are used mainly by publishers to create ISBN barcodes, and to mark the issue number or price of periodicals. The additional digits must be supplied immediately after the standard 11 digits.

PlaceUPCE

`Esc` **B** `xxxx yyyy` **B** `hh o… d…`

Place a UPC-E barcode.

Content of o…	(Nothing)	Full UPCA format. Requires 11 digits of data.
	Z	Zero-suppressed format. Requires 6 digits.
d…		Barcode data, fixed number of digits (see above)

Example:
Place UPCE at (10, 90), default height, data 123456.

Text	EscB00100090B00Z123456
Hex	1B 42 30 30 31 30 30 30 39 30 42 30 30 5A 31 32 33 34 35 36
Dec	27 66 48 48 49 48 48 48 57 48 66 48 48 90 49 50 51 52 53 54

The UPCE barcode is a fixed length numeric code, used where the product is too small to fit a standard UPC-A code. Certain ranges of UPC are allocated to this purpose and can be represented in fewer digits by suppression of zeros. Refer to the AIM documentation for details. The data can be supplied either in its full UPC-A format, or in the zero-suppressed form as it appears on the barcode.

PlaceCode93

`Esc` **B** `xxxx yyyy` **C** `hh Eot d… Eot`

Place a Code 93 barcode.

d…	Barcode data terminated with Eot

Example:
Place Code93 at (20, 45), height 30, Data "Code93 Barcode".

Text	EscB00200045C30EotCODE93 BARCODEEot
Hex	1B 42 30 30 32 30 30 30 34 35 43 33 30 04 43 4F 44 45 39 33 20 42 41 52 43 4F 44 45 04
Dec	27 66 48 48 50 48 48 48 52 53 67 51 48 4 67 79 68 69 57 51 32 66 65 82 67 79 68 69 4

Code93 is a variable length barcode used in many of the same application areas as Code39. However, it is more compact and supports the full Ascii range through the use of shift characters.

The main character set is the same as for Code39 and consists of all the uppercase letters, numbers and also dash, stop, space, dollar, oblique, plus & percent :-
0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ - . Space $ / + %
The full Ascii character set may be used and the printer will generate the shift codes as required. The two checksums are calculated and added automatically.

Placing 2-D Barcodes

By default, the X and Y placement coordinates for all 2-D barcodes refer to the top left corner of the symbol. However, you can change this behaviour using the SetLeftRightAlignment and SetBarAlignment commands.

PlaceDataMatrix

`Esc : xxxx yyyy E rrr ccc mmmm s d… Eot`

Place a DataMatrix 2-D symbol.

xxxx	X placement coordinate
yyyy	Y placement coordinate
rrr	Number of rows (000 = auto fit to data)
ccc	Number of columns (000 = auto fit to data)
mmmm	Module size in current global units (default = pixels)
s	Special start character. A = AIM prefix (FNC1) 5 = 05 macro 6 = 06 macro R = Reader program 0 = No special start character required
d…	Data, terminated with Eot

Example:

Place DataMatrix at (30, 85), Auto fit rows and columns, module size = 16, no special start char, "DataMatrix".

Text	Esc:00300085E00000000160DataMatrixEot
Hex	1B 3A 30 30 33 30 30 30 38 35 45 30 30 30 30 30 30 30 30 31 36 30 44 61 74 61 4D 61 74 72 69 78 04
Dec	27 58 48 48 51 48 48 48 56 53 69 48 48 48 48 48 48 48 48 49 54 48 68 97 116 97 77 97 116 114 105 120 4

The ECC200 form of DataMatrix which includes Reed-Solomon error correction is fully supported. The older forms which use convolutional error correction are not.

For details on the use of special start characters, refer to the DataMatrix specification. Most applications do not use a special start character.

The data is encoded automatically using the optimum compaction modes as recommended in the DataMatrix specification.

The data can contain special characters introduced by the \ escape char.

Data sequence	Expands to …
Gs	FNC1
\ Gs	Gs
\ Eot	Eot
\\	\
\dddddd	Extended Character Interpretation (ECI)

PlacePDF417 Esc : xxxx yyyy P ww hh aa rr cc e t d… Eot

Place a PDF417 2-D symbol.

xxxx	X placement coordinate
yyyy	Y placement coordinate
ww	X module dimension in pixels (00 = default, 0.25 mm)
hh	Y module dimension in pixels (00 = auto)
aa	Aspect ratio height:width of whole symbol (00 = unspecified)
rr	Number of rows (00 = auto fit to data)
cc	Number of columns (00 = auto fit to data)
e	Error correction level (0-8, 9 = auto)
t	Mode: S = standard, T = truncated
d…	Data, terminated with Eot

Example:

Place PDF417 at (120, 75), X=3 pixels, Y=auto, 2:1 aspect ratio, autofit to data, auto error level, standard format, "PDF417 Test".

Text	Esc:01200075P03000200009SPDF417 TestEot
Hex	1B 3A 30 31 32 30 30 30 37 35 50 30 33 30 30 30 32 30 30 30 30 39 53 50 44 46 34 31 37 20 54 65 73 74 04
Dec	27 58 48 49 50 48 48 48 55 53 80 48 51 48 48 48 50 48 48 48 48 57 83 80 68 70 52 49 55 32 84 101 115 116 4

The X and Y module dimension are specified in pixels, regardless of the global units of measure.

If the Y module dimension is specified as 00, then it will be set automatically to 3 or 4 times the X dimension, depending on the error correction level. This follows the procedure set out in the PDF417 spec. If an aspect ratio is specified, then the number of rows and columns are optimised to best fit (row and column values should be set to 0). If aspect ratio is not specified, rows and/or columns must be specified. If both are specified, the symbol is that exact size, otherwise the unspecified dimension is adjusted to fit the quantity of data supplied.

The error correction level can be specified explicitly, or set to 9 for auto error level. This sets the error level to the minimum recommended level for the quantity of data supplied, according to the PDF417 spec.

The data can contain special characters introduced by the \ escape char.

Data sequence	Expands to …
\ Eot	Eot
\\	\

The data is encoded automatically using the optimum compaction modes as defined in the PDF417 specification. Note that other printers may have slightly different optimisation routines, so symbols which appear different may scan to produce identical data.

Truncated PDF417 is supported. Macro PDF417 is not currently supported.

PlaceMaxiCode Esc : xxxx yyyy U m n t e d… Eot

Place a UPS MaxiCode 2-D symbol.

xxxx	X placement coordinate
yyyy	Y placement coordinate
m	Mode 2-6
n	Reserved for structured append. Set to '1'
t	Reserved for structured append. Set to '1'
e	1 = Include Eot terminator in symbol data 0 = Do not include Eot terminator
d…	Data, terminated with Eot

Example:

Place MaxiCode at (270, 285),
Mode=3
Postal code=B1050
Country code=056
Class of service=999
Message=MaxiCode

Text	Esc:0270285U3110B1050Gs056Gs999Gs BlazepointEot
Hex	1B 3A 30 32 37 30 30 32 38 35 55 33 31 31 30 42 31 30 35 30 1D 30 35 36 1D 39 39 39 1D 4D 61 78 69 43 6F 64 65 04
Dec	27 58 48 50 55 48 48 50 56 53 85 51 49 49 48 66 49 48 53 48 29 48 53 54 29 57 57 57 29 77 97 120 105 67 111 100 101 4

MaxiCode is a 2-D optically read code developed by United Parcel Service (UPS).

Modes 2 and 3 are used by UPS for parcel routing information. Modes 4 and 5 can contain general purpose data content and mode 6 is used for programming the reader system. For further details of the different modes provided by this symbology refer to the UPS guidelines.

In modes 2 and 3 the data within the symbol should normally include a terminating Eot character. Use the e parameter to force inclusion of the terminator. If an Eot character is required at any other point within the data, it can be represented using the sequence \Eot.

In modes 2 and 3, if the Structured Carrier header "[)>Rs01Gs" is included in the data stream, it will be recognised and transferred to the start of the secondary message automatically. It is therefore possible to send messages as defined in the MaxiCode spec without re-ordering the fields.

Primary message fields in modes 2 and 3 must be separated using the Gs character, regardless of whether or not the structured carrier header is present.

The MaxiCode spec gives no guidance as to how to pad out unused data, so there is no guarantee that the symbols produced will be identical to the ones generated by other printers or applications, even though the content should be identical when read.

MaxiCode symbols can only be printed at 0, 90, 180 and 270 degree orientations. Requests for other rotation angles will be ignored.

Structured append is not currently supported, but may be in future revisions of firmware.

Block Fill, Box & Line Drawing

BlockFill — Esc I xxxx yyyy wwww hhhh c

Place a filled block on the label.

xxxx	X placement coordinate
yyyy	Y placement coordinate
wwww	Block width
hhhh	Block height
c	Colour fill code (see table below)

Example:
Place a 100x100 block at (140, 200 inverting whatever is underneath.

Text	EscI0140020001000100N
Hex	1B 49 30 31 34 30 30 32 30 30 30 31 30 30 30 31 30 30 4E
Dec	27 73 48 49 52 48 48 50 48 48 48 49 48 48 48 49 48 48 78

The BlockFill command can be used to place a block of black, white or grey, or to colour existing fields, e.g. invert text. Basic line drawing can be obtained by specifying a narrow height or width. Box drawing is obtained either with multiple lines, or by overlaying a black box with a slightly smaller white box. Block fills are not affected by the SetRotation command, for backward compatibility reasons, but they are affected by the SetRotationAngle command.

The colour code determines the colour of the fill pattern, and its effect on anything underlying it in the image.

Colour code	Description	Effect on underlying text
B	Black fill	Erased
W	White fill	Erased
G	Grey fill	Erased
N	Invert	White on black
A	Grey AND	Grey on white
R	Grey OR	Black on grey
I	Grey invert	White on grey

SetGreyShade — Esc * n… S

Selects the grey density to use.

n…	Percentage density. 100% represents full black.

Example:
Select 25% grey

Text	Esc*25S
Hex	1B 2A 32 35 53
Dec	27 42 50 53 83

Placing Graphics

The printer supports 2 types of graphic images. Logos are small bitmaps that are stored separately in memory and can then be placed on the page image buffer as required. Direct graphics fields transfer the bitmap directly onto the page image buffer without storing it in a separate memory location. Logos can be magnified and rotated, and can be placed as often as required once downloaded. However, they take up space in the printer memory. Direct graphics fields do not take up any additional space, but must be downloaded each time they are used. Direct graphics fields cannot be magnified or rotated.

In both cases, the data defines a bitmapped image. 1 byte = 8 dots described horizontally, working left to right and top to bottom. A '1' bit represents a black pixel. The most significant bit of the first byte appears on the left hand side and consecutive bytes work across to the right until the described width is reached. Any unused bits in the last byte of the line are discarded and the next line begins with the next byte. This continues until the required height and width have been covered. The command finishes when the correct number of bytes have been received. Note that the width and height of graphics images are always specified in pixels regardless of the global units of measure.

To save download time, graphics data can be compressed. Use the SetGraphicsCompression command to select the compression method.

Graphics commands require an 8 bit communications interface to transfer correctly. If the serial port is being used, it must set up for 8 data bits or used in printable character mode.

DirectGraphics — Esc G xxxx yyyy wwww hhhh d…

Place a graphical image directly into the image buffer.

xxxx	X placement coordinate
yyyy	Y placement coordinate
wwww	Block width (always in pixels)
hhhh	Block height (always in pixels)
d…	Graphics data

Example:

Place a 12x6 graphic image at (10, 20). Image is a chequerboard, Each square being 2 pixels across. Top left square is black.

Text	EscG0010002000120012 (binary data…)
Hex	1B 47 30 30 31 30 30 30 32 30 30 30 31 32 30 30 31 32 CC C0 CC C0 33 30 33 30 CC C0 CC C0 33 30 33 30 CC C0 CC C0 33 30 33 30
Dec	27 71 48 48 49 48 48 48 50 48 48 48 49 50 48 48 49 50 204 192 204 192 51 48 51 48 204 192 204 192 51 48 51 48 204 192 204 192 51 48 51 48

DefineLogo

`Esc` **K** `n tttttttttt wwww hhhh d…`

Store a logo ready for placement later.

`n`	Logo number 0 – 9
`tttttttttt`	Logo title (exactly 10 characters)
`wwww`	Logo width (always in pixels)
`hhhh`	Logo height (always in pixels)
`d…`	Graphics data

Example:
Define a 12x6 image as logo 2. Image is a chequerboard, each square being 2 pixels across. Top left square is black.

Text	EscK2ChequerBrd00120012 (binary data…)
Hex	1B 4B 32 43 68 65 71 75 65 72 42 72 64 30 30 31 32 30 30 31 32 CC C0 CC C0 33 30 33 30 CC C0 CC C0 33 30 33 30 CC C0 CC C0 33 30 33 30
Dec	27 75 50 67 104 101 113 117 101 114 66 114 100 48 48 49 50 48 48 49 50 204 192 204 192 51 48 51 48 204 192 204 192 51 48 51 48 204 192 204 192 51 48 51 48

For historical reasons the logo must have a 10 character name, although the name is no longer used. The GPL-2 requirement to delete a logo before defining it has been removed.

PlaceLogo

`Esc` **L** `xxxx yyyy n`

Place a stored logo on the label.

`xxxx`	X placement coordinate
`yyyy`	Y placement coordinate
`n`	Logo number 0 – 9

Example:
Place stored logo 2 onto the label at (170, 30), and also at (170, 80)

Text	EscL017000302EscL017000802
Hex	1B 4C 30 31 37 30 30 30 33 30 32 1B 4C 30 31 37 30 30 30 38 30 32
Dec	27 76 48 49 55 48 48 48 51 48 50 27 76 48 49 55 48 48 48 56 48 50

The logo will be magnified and rotated according to the current magnification and rotation settings. Note that magnification and rotation are both relatively slow operations. For best performance, the logo should be stored in the same size and orientation that it will be used at.

DeleteLogo Esc X n

Delete a stored logo from memory.

n	Logo number 0 – 9

Example:
Delete stored logo 2 from memory

Text	EscX2
Hex	1B 58 32
Dec	27 88 50

The GPL-2 requirement to send this command twice has been removed.

SetGraphicsCompression Esc * n… E

Selects the graphics compression mode to use.

Value of n…	0	No compression. Send raw bitmap.
	1	TIFF-4 run-length encoding

Example:
Select TIFF-4 compression

Text	Esc*1E
Hex	1B 2A 31 45
Dec	27 42 49 69

With TIFF-4 compression, the data is sent in small packets, each packet consisting of a count followed by the graphics data. The first byte is the count value, *n*. If *n* lies in the range 0 to 127, then it is followed by *n*+1 bytes of data that are to be used as received. If *n* lies in the range 129 to 255 then it is followed by a single byte which is to repeated 257-*n* times.

Example:

Uncompressed data (hex)	AA 55 00 00 00 00 00 00 00 00 FF FF FF FF F8
Compressed data (hex)	01 AA 55 F9 00 FD FF 00 F8

Most graphic data contains a lot of repeated 00 or FF bytes, and this simple type of compression is usually very effective.

DefineFlashLogo Esc K # n ffff

Copy a logo in RAM to permanent storage in Flash.

n	Logo number to copy 0-9
ffff	Flash logo number 0 – 3999

Example:
Copy logo 2 to flash logo 29.

Text	EscK#20029
Hex	1B 4B 23 32 30 30 32 39
Dec	27 75 35 50 48 48 50 57

Storing logos in Flash memory avoids the need to download the data each time it is used and can increase throughput when used with the serial port. Stored logos are also useful in standalone applications.

PlaceFlashLogo Esc L xxxx yyyy # nnnn

Place a logo stored in Flash onto the label.

xxxx	X placement coordinate
yyyy	Y placement coordinate
nnnn	Flash logo number 0 – 3999

Example:
Place flash logo 29 onto the label at (230, 70)

Text	EscL02300070#0029
Hex	1B 4C 30 32 33 30 30 30 37 30 23 30 30 32 39
Dec	27 76 48 50 51 48 48 48 55 48 35 48 48 50 57

DeleteFlashLogo Esc X # ffff

Delete a logo from the Flash file system.

ffff	Flash logo number 0 – 3999

Example:
Delete flash logo 29.

Text	EscX#0029
Hex	1B 58 23 30 30 32 39
Dec	27 88 35 48 48 50 57

RetrieveFlashLogo Esc H # n ffff

Retrieve a logo from the Flash file system into memory.

n	Logo number in memory, 0-9
ffff	Flash logo number, 0 – 3999

Example:
Copy flash logo 29 to logo 2.

Text	EscH#20029
Hex	1B 48 23 32 30 30 32 39
Dec	27 72 35 50 48 48 50 57

This command is used for backward compatibility with printers having a permanent resource file burned with the firmware, which would copy the flash logos into memory at power-on. Logos retrieved in this way are protected against deletion when the command interpreter is reset by a front panel cancel, and also against deletion by the ClearAllBuffers command.

ProtectLogo Esc H n

Protect a memory logo from deletion.

n	Logo number 0 – 9

Example:
Protect logo 2

Text	EscH2
Hex	1B 48 32
Dec	27 72 50

Applies the same protection to any logo as that provided by the RetrieveFlashLogo command.

Printing Commands

FormFeed — Ctrl-L

Print the current label.

Text	Ctrl-T
Hex	0C
Dec	12

This command prints one copy of the current label. Sending a further FormFeed command will clear the image buffer and feed a blank label. Use the RepeatPrint command to obtain multiple copies.

RepeatPrint — Esc r nnnn

Print multiple copies of the current label.

nnnn — Number of copies to print

Example:
Print 10 copies of the current label.

Text	Escr0010
Hex	1B 72 30 30 31 30
Dec	27 114 48 48 49 48

Unlike the FormFeed command, this command can be sent again without clearing the existing image buffer. A copy count of 0000 is interpreted as an infinite repeat.

RepeatPrint3 — Esc R nnn

Print multiple copies of the current label (obsolete command).

nnn — Number of copies to print

This command is identical in function to RepeatPrint but only takes a 3 digit count. It is retained for backward compatibility but is **not recommended** for new applications.

CancelPrinting — Esc a

Cancel all printing in progress.

Text	Esca
Hex	1B 61
Dec	27 97

If multiple labels have been requested using RepeatPrint, this command allows you to cancel the print job. It can also be used with single label prints, but if it is queued in the input buffer behind several other labels, it will not be actioned until the previous commands have executed, which will delay its effect.

PrintSpecial Esc * K

Print current label, retaining previous buffer.

Text	Esc*K
Hex	1B 2A 4B
Dec	27 42 75

This command is almost identical to the FormFeed command, but allows you to print a special label in the middle of a repeat run, without losing the content of the previous print buffer. This might be used where barcodes are being verified after each print and you need to print a failure notification, then continue with the print run. If this command is followed by a RepeatPrint command, the previous label, not the special label will be repeated. You can send multiple special labels without losing the previous buffer. The only exception to this is if the printer has been explicitly set to single print buffer mode, or if the label length is such that there is only room for a single print buffer.

BackfeedLabel Esc * Z

Print current label, retaining previous buffer.

Text	Esc*Z
Hex	1B 2A 5A
Dec	27 42 90

This command will set up a backfeed such that the next printed label overprints the previous label. The command will only work if there are no error conditions, as these would cause loss of paper alignment. The backfeed does not happen immediately, but only when the next label is printed. This type of backfeeding might be used to obliterate a previously printed label, for example if a ticket is cancelled, or a barcode fails to scan correctly.

Use this command with caution. Unless the paper path is carefully managed, backfeeding peelable labels can cause the labels to peel off and stick to the print mechanism. Use only with short label lengths and well behaved media.

SetHeat — Esc h nnn

Set the print heat.

nnn	Print heat as percentage of factory setting

Example:
Set print heat to 120%

Text	Esch120
Hex	1B 68 31 32 30
Dec	27 104 49 50 48

The heat level represents the percentage of the factory preset level. Different media may require different heat levels to obtain the best print quality (refer to the section on optimising print quality). The maximum heat setting allowed is 150% in direct thermal mode and 200% in thermal transfer mode. Attempting to set too high a heat value sets it to the maximum permitted value. Note that the heat profile is different for direct thermal and thermal transfer applications. Selecting thermal transfer mode automatically reduces the heat by about 30%. The default heat setting of 100% works in most cases for direct thermal papers and wax or wax-resin ribbons.

SetSpeed — Esc m nnn

Set the print speed.

nnn	Print speed in mm/s

Example:
Set print speed to 125 mm/s

Text	Escm125
Hex	1B 6D 31 32 35
Dec	27 109 49 50 53

Maximum settable speed depends on the particular printer type. Attempting to set too high a speed will set it to the maximum permitted value. Print speed also affects the print quality – refer to the section on optimising print quality.

SetSpeed2 — Esc S nn

Set the print speed (obsolete command).

Value of nn		
	40 .. 99	Print speed in mm/s
	00 .. 39	Print speeds 100 to 139 mm/s

Example:
Set print speed to 125 mm/s

Text	EscS25
Hex	1B 53 32 35
Dec	27 83 50 53

This command is identical in function to the SetSpeed command, but takes only 2 digits, giving a very limited range of speeds. It is included only for backward comptibility, but is **not recommended** for new applications.

Stored Formats (Command Macros)

Stored Formats provide a way of storing a set of commands, normally a label definition, so that the commands can be replayed with a short command sequence. This gives a macro facility that can save on communications traffic in critical cases, and also provides a means of batch printing with incrementing variables (refer to the section on variable fields).

DefineFormat — Esc D n tttttttttt d… Ctrl-K

Store a format ready for replay later.

n	Format number 0 – 9
tttttttttt	Format title (exactly 10 characters)
d…	Format content, terminated with Ctrl-K

Example:
Define format 3 which when replayed prints 'Hello' in 12 point Swiss.

Text	EscD3TestFormatEscY0001201200EscT00100010Hello Eot Ctrl-L
Hex	1B 44 33 54 65 73 74 46 6F 72 6D 61 74 1B 59 30 30 30 31 32 30 31 32 30 30 1B 54 30 30 31 30 30 30 31 30 48 65 6C 6C 6F 04 0C
Dec	27 68 51 84 101 115 116 70 111 114 109 97 116 27 89 48 48 48 49 50 48 49 50 48 48 27 84 48 48 49 48 48 48 49 48 72 101 108 108 111 4 12

For historical reasons the format must have a 10 character name, although the name is no longer used. Note that the command will fail if a format already exists with the specified format number. It is safest to send a DeleteFormat command before the DefineFormat command.

The commands are entered directly after the title, terminated with Ctrl-K. Note that Code128 and 2-D barcodes which include Ctrl-K in the data cannot be stored this way. Graphics fields should not be defined in a format either for the same reason. Use logos instead.

The GPL2-requirement to erase an existing format before using it has been removed.

EraseFormat — Esc E n

Erase a stored format.

n	Format number 0 – 9

Example:
Erase format 3.

Text	EscE3EscE3
Hex	1B 45 33 1B 45 33
Dec	27 69 51 27 69 51

The GPL-2 requirement to send this command twice has been removed.

LoadFormat Esc J n

Replay a stored format.

n	Format number 0 – 9

Example:
Replay format 3.

Text	EscJ3
Hex	1B 4A 33
Dec	27 74 51

Execute the specified format as a macro to be replayed once.

RepeatLoadFormat Esc j n rrrr

Replay a stored format multiple times.

n	Format number 0 – 9
rrrr	Number of times to play format

Example:
Replay format 3, 20 times.

Text	Escj30020
Hex	1B 6A 33 30 30 32 30
Dec	27 106 51 48 48 50 48

This command executes a stored format repeatedly. This is particularly useful for printing batches of labels with incrementing fields (Refer to the variable fields section).

Defining Variable Fields

Variable fields are used in conjunction with stored formats. A format may be defined which includes references to variable/incrementing data and whenever that format is printed, the variable values are incorporated in the output.

Variable fields are accessed by replacing the normal data for text or barcodes with an escape sequence that expands to produce the variable data. This allows several variables to be used in one text/barcode field, or the same variable to be used in more than one text/barcode field. It also allows for part of the text to be fixed and part variable.

There are two types of variable field: text variables and numeric variables. Text variables are stored exactly as the information is received, whereas numeric fields can be incremented or decremented and have a number of formatting options.

There is a single command to update all incrementing variables, which allows incrementing serial numbers to be added to labels automatically.

The normal sequence of events when programming will be :-

1) Define variable types & sizes, and assign initial values.
2) Define label format, referencing the variables where needed. If incrementing fields have been used, the format should contain the command to update the variables. The format should contain a formfeed or repeat print command.
3) Repeat-load the format to get labels with auto-incrementing variables.

DefineField — Esc fd r ww o… Eot

Define a variable field.

	r	Field reference 0..9, A..Z
	ww	Maximum field width 01 – 99
Content of o…	+dd	Incrementing field, increment value = dd
	-dd	Decementing field, decrement value = dd
	=	Simple numeric field (no increment or decrement)
	/uu	Update frequency, update every uu requests (default is 1)

Example:
Define variable A, max width 8, increment by 1 every 2nd update. Define variable B, max width 20, plain text variable.

Text	EscfdA08+01/02Eot EscfdB20Eot
Hex	1B 66 64 41 30 38 2B 30 31 2F 30 32 04 1B 66 64 42 32 30 04
Dec	27 102 100 65 48 56 43 48 49 47 48 50 4 27 102 100 66 50 48 4

This command defines a variable text or numeric field that can be used in text or barcode commands at a later stage. The reference letter may be 0-9 or A-Z, allowing up to 36 variables. The width specifier defines the maximum number of characters that can be used in the field. No further information is required for text fields, and Eot terminates the definition.

For a simple numeric field which does not increment or decrement, add '=' after the width value. If the field is to be incremented or decremented automatically, then an increment value must be supplied using '+ii' (or '-ii' for decrement). If the field is to be updated only every n labels, then an update frequency must be supplied with '/uu'. The Eot character ends the definition.

The difference between simple (non-incrementing) numeric fields and plain text fields is that a numeric field will be stripped of leading zeros when stored, whereas a text field is stored exactly as supplied. For either type of field, the formatting instructions can be used to provide leading zeros to pad the field to the width required by the application.

SetFieldValue — Esc **v** r d… Eot

Set the content of a variable field.

r	Field reference 0..9, A..Z
d…	Field content, terminated with Eot

Example:
Set field 3 to the value "ABC123"

Text	Escv3ABC123Eot
Hex	1B 76 33 41 42 43 31 32 33 04
Dec	27 118 51 65 66 67 49 50 51 4

UpdateFields — Esc **fu**

Update all numeric variable fields.

Example:
Update all fields.

Text	Escfu
Hex	1B 66 75
Dec	27 102 117

Updates all numeric fields according to the rules set in DefineField. An incrementing field will normally be a pure decimal number, but can include any characters.
0 increments to 1, 9 increments to 0 with a carry to the next digit.
A increments to B, Z increments to A with a carry to the next digit.
a increments to b, z increments to a with a carry to the next digit.
Non alphanumeric characters are skipped over.
Carries which exceed the allocated width of the variable are discarded, so for a 4 digit variable, 9999 + 1 = 0.

Examples:-
128 + 3 = 131
12A + 2 = 12C
1a8 + 3 = 1b1
99#Z + 1 = 100#A

ListFields — Esc fl

List all variable fields defined.

Example:
List all fields.

Text	Escfl
Hex	1B 66 6C
Dec	27 102 108

A verbose listing of all defined fields is produced on the serial port. This is mainly for debugging purposes.

SetRealTimeClock — Esc fs DD MM YY hh mm ss

Set the printer's real-time clock.

DD	Day of month (1..31)
MM	Month of year (1..12)
YY	Year (00..99 = 2000..2099)
hh	Hour (00..23)
mm	Minute (00..59)
ss	Second (00..59)

Example:
Set RTC to 12:15:37 on 20 May 2008.

Text	Escfs200508121537
Hex	1B 66 73 32 30 30 35 30 38 31 32 31 35 33 37
Dec	27 102 115 50 48 48 53 48 56 49 50 49 53 51 55

Sets the printer's real-time clock and updates the date/time field. Note that most printers do not contain a battery-backed real-time clock so *the time may not be maintained when the printer is switched off*. It is therefore advisable to set the RTC before starting each run of labels.

UpdateTimeField — Esc ft

Update the date-time field from the RTC.

Example:
Update date-time field.

Text	Escft
Hex	1B 66 74
Dec	27 102 116

Field 't' is reserved for the date & time field. Executing this command copies the current value of the real time clock to field ‘t’. The clock itself is not used directly since it is possible for the clock to change within the formatting time for a label. This could give inconsistent results if the time appeared more than once in a label, e.g. in text and a barcode.

Using Variable Fields

Variable fields are accessed by placing a special escape sequence within the data supplied to a normal text or barcode command. Any number of variables can be included within a command.

GetFieldValue Ctrl-A r o… Eot

Get the content of a variable field, with formatting.

	r	Field reference 0..9, A..Z
Content of o…	None	No formatting – present field value as stored.
	< n…	Padded with trailing spaces to at least n characters.
	> n…	Padded with leading spaces to at least n characters.
	= n…	Padded with leading zeros to at least n characters.
	= n… . m…	Padded with leading zeros to at least n characters, with decimal point inserted giving m decimal places.

Example:

Place a text field with fixed text 'Hello ' followed by the content of Field 4, presenting it as 12 characters filled on the right with spaces.

Text	EscT00100010Hello Ctrl-A4<12Eot Eot
Hex	1B 54 30 30 31 30 30 30 31 30 48 65 6C 6C 6F 20 01 34 3C 31 32 04 04
Dec	27 84 48 48 49 48 48 48 49 48 72 101 108 108 111 32 1 52 60 49 50 4 4

This sequence can appear within text or barcode data and provides access to the variable fields, With no formatting information, the variable value appears in its simplest form, and takes the minimum width necessary. This will normally be OK for text, but barcodes often need a particular width. The formatting code can be used to specify a minimum field width, and the padding required. It also allows a decimal point to be inserted in a number. e.g. a price of £3.99 may appear as 3.99 in text and 00399 in a barcode.

Note that the field width specified in the format is a minimum width; if the variable exceeds this width it will not be truncated.

Note that the escape sequence must be terminated with Eot and the text and some barcode placement commands also require an Eot terminator, hence the double termination in the example above.

These are best illustrated by example. Assume field 1 has a value of 258.

Escape sequence:	Expands to:
Ctrl-A 1 Eot	"258"
Ctrl-A 1>5 Eot	" 258"
Ctrl-A 1<5 Eot	"258 "
Ctrl-A 1=5 Eot	"00258"
Ctrl-A 1=3.2 Eot	"2.58"

GetTimeValue Ctrl-A t o… Eot

Get the content of the date-time field, with formatting.

Content of o…	None	Default date & time formatting.
	+ n…	Add n days to the date before formatting.
	- n…	Subtract n days from the date before formatting.
	D	Day of month as 2 digits (01..31)
	#D	Day of month as 1 or 2 digits (1..31)
	J	Julian day as 3 digits (001..366)
	#J	Julian day as 1-3 digits (1..366)
	M	Month as 2 digits (01..12)
	#M	Month as 1 or 2 digits (1..12)
	Y	Year as 2 digits (00..99)
	#Y	Year as 4 digits (2000..2099)
	h	Hour as 2 digits, 24 hour clock (00..23)
	#h	Hour as 1 or 2 digits, 12 hour clock (1..12)
	m	Minute as 2 digits (00..59)
	s	Second as 2 digits (00..59)
	a	AM/PM indicator as 'a' or 'p'
	A	AM/PM indicator as 'A' or 'P'
	Other chars	Passed through unchanged.

Example:
Place a text field with the date and time in YYYY-MM-DD hh:mm:ss format.

Text	EscT00100010Date: Ctrl-At#Y-M-D h:m:sEot Eot
Hex	1B 54 30 30 31 30 30 30 31 30 44 61 74 65 3A 20 01 74 23 59 2D 4D 2D 44 20 68 3A 6D 3A 73 04 04
Dec	27 84 48 48 49 48 48 48 49 48 68 97 116 101 58 32 1 116 35 89 45 77 45 68 32 104 58 109 58 115 4 4

This command is used in the same way as the variable field command, but the information is taken from the date/time field. The format information is also different. The offset may be used for printing expiry or best-before dates. A negative offset implies back-dating.

The following example assume a date/time of 16:23:40 on 18th August 2004.

Escape sequence:	Expands to:
Ctrl-A t#D-#M-#Y #h:m A Eot	"18-8-2004 4:23 P"
Ctrl-A t+30D/M/Y Eot	"17/09/04"

Image Buffers

As with other page printers, e.g. laser printers, fields are placed in an image buffer in printer memory. When the image is complete it is transferred to the paper, the image is cleared and the process can be repeated. BPL has a number of features designed to increase the throughput of the printer.

When formatting complex labels where the majority of the label is invariant and a few fields change, there are two methods of saving formatting time. The first is to copy back the previous image and blank out the areas that are changing. This is relatively simple if the variant data is a fixed size, but in some cases it is difficult to predict how much blanking is required. An alternative method is to allocate a *Background* buffer. The fixed fields are formatted first and the image frozen in the *Background* buffer. For each subsequent label, the *Background* is copied and the variant fields added on top. This method is easier and more foolproof than the blanking method. It does require an additional image buffer, which may reduce the maximum label size. However, since all Blazepoint printers have a generous memory allocation, this is unlikely to be a problem.

BackgroundBuffer `Esc i n`

Enable or disable the background buffer.

`n`	0 = Disabled (default) 1 = Enabled

Example:
Enable the background buffer

Text	`Esci1`
Hex	1B 69 31
Dec	27 105 49

If a particular label height has been requested using SetLabelHeight, the number of image buffers may be reduced. If a particular number of buffers has been requested, the available label height may be reduced. Unless the label size is very large, this is unlikely to cause any problem.

SaveBackground `Ctrl-N`

Save the current image buffer to the background buffer.

Text	`Ctrl-N`
Hex	0E
Dec	14

When the background buffer is disabled, this command has no effect.

CopyBackground `Ctrl-P`

Copy the previous print buffer or the background buffer to the current image buffer.

Text	`Ctrl-P`
Hex	10
Dec	16

When the background buffer is enabled, this command copies its contents to the image buffer, overwriting any existing content. When the background buffer is disabled, this command copies the previous printed image to the image buffer.

ClearImageBuffer `Ctrl-Y`

Erase the content of the current image buffer.

Text	`Ctrl-Y`
Hex	19
Dec	25

Clears the current image. The image is normally cleared automatically so this command is only useful for erasing a partially formed image after creating an image to copy to the background buffer, or if a mistake is made.

ClearAllBuffers `Ctrl-X`

Erase the content of all image buffer, logos, formats & variable fields.

Text	`Ctrl-X`
Hex	18
Dec	24

Clears all the image buffers and deletes all logos, stored formats and variable fields. Protected logos (those loaded to memory from the Flash file system, or specifically protected) are not deleted by this command.

Paper Advance/Backfeed

These commands allow the paper to be advanced after printing a label, with automatic backfeed before the start of the next label. This is most likely to be of use in conjunction with the cutter module or tearbar. Peelable labels should not be advanced so far that the trailing edge of a second label emerges from under the print head, as it may then peel slightly, and stick to the front of the print head when reverse fed.

There are 2 stages involved in using the advance/backfeed commands. The first is to set the paper advance distance, which may be a preset distance to the cutter or tear bar, or a custom distance. The second stage is to specify when the paper advance will occur. Backfeed occurs automatically immediately before printing the next label and does not need to be programmed.

When the label-taken sensor is used, the printer automatically selects the label-taken advance, which feeds the trailing edge of the label clear of the print head and makes peeling easier.

The preset distances for label-taken, cutter, tearbar and custom are preset in the factory but can be adjusted using the blazeConfig program.

SetAdvanceToCutter — Esc p C

Set the advance distance to the preset printhead-to-cutter distance.

Text	EscpC
Hex	1B 70 43
Dec	27 112 67

SetAdvanceToTearBar — Esc p T

Set the advance distance to the preset printhead-to-tearbar distance.

Text	EscpT
Hex	1B 70 54
Dec	27 112 84

SetAdvanceCustom — Esc p U

Set the advance distance to the preset custom distance.

Text	EscpU
Hex	1B 70 55
Dec	27 112 85

SetAdvanceDistance — Esc p L yyyy

Sets a custom advance distance at program run time.

yyyy	Advance distance to use (mm)

Example:
Set advance distance to 20mm

Text	EscpL0020
Hex	1B 70 4C 30 30 32 30
Dec	27 112 76 48 48 50 48

If an advance distance other than the preset distances is required, then this command can be used. However, using presets is preferable as it makes the programming more portable between different printers.

EnableAdvance — Esc p N

Set the printer to advance whenever printing pauses.

Text	EscpN
Hex	1B 70 4E
Dec	27 112 78

Whenever printing pauses, the paper advance will be activated. Printing may pause under these conditions:

- No more label data ready
- Pause button pressed
- Pause-every-label active
- Cutting active

AdvanceImmediate — Esc p I

Enable advance and activate immediately.

Text	EscpI
Hex	1B 70 49
Dec	27 112 73

This command is identical to the EnableAdvance command, except that the advance is activated immediately, regardless of whether the pause conditions above are met. In general this command will not be needed, and its use is **not recommended**.

DisableAdvance — Esc p X

Disable paper advance.

Text	EscpX
Hex	1B 70 58
Dec	27 112 88

Disables further paper advance. If the paper is already advanced, then a final backfeed will occur automatically when the next label is printed.

Cutter Commands

CutEveryNLabels — Esc * n… C

Activate the cutter every Nth printed label.

n… Cut frequency, 0 = disable cutting

Example:
Cut every label

Text	Esc*1C
Hex	1B 2A 31 43
Dec	27 42 49 67

This command activates the cutter after the Nth label has been printed. N is normally set to 1 to activate the cutter every label, but can be set to other values to cut when a fixed batch size is printed. Whenever this command is sent, the batch counter is reset.

Cutting is normally used in conjunction with paper advance, as the cutter is some distance forward of the printhead. The recommended sequence of commands to enable cutting is:

- SetAdvanceToCutter
- EnableAdvance
- CutEveryNLabels(1)
- Print labels…

CutImmediate — Esc C I

Activate the cutter immediately.

Text	EscCI
Hex	1B 70 49
Dec	27 112 73

This command activates the cutter once only, immediately the paper stops moving. If a paper advance is in effect, it will execute before the cut. This can be used to implement cut-on-demand, or to cut at the end of a batch of labels. The recommended sequence of commands to enable cutting at the end of a batch is:

- SetAdvanceToCutter
- EnableAdvance
- Print all labels
- CutImmediate

The following 2 commands are provided purely for backward compatibility and are **not recommended** for new applications.

EnableCutting `Ctrl-\`

Set advance-to-cutter and cut every label.

Text	`Ctrl-\`
Hex	1C
Dec	28

DisableCutting `Ctrl-]`

Disable advance and cutting.

Text	`Ctrl-]`
Hex	1D
Dec	29

Neither of these commands offers the flexibility of their newer counterparts, and they may be removed from later versions.

Pause, Label-Taken and Delay

PauseEveryNLabels Esc * n… b

Pause printing every N[th] printed label.

	n…	Pause frequency, 0 = disable

Example:
Pause every label

Text	Esc*1b
Hex	1B 2A 31 62
Dec	27 42 49 98

Normally the pause function is applied manually using the pause button, or automatically on every printed label using the label-taken sensor. This function allows pause to be applied after a set number of labels have been printed. Press the pause button to release the pause function.

EnableTakenSensor Esc k n

Enable or disable the label-taken sensor at print time.

Value for n	0	Disabled
	2	Enabled

Example:
Enable label-taken sensor

Text	Esck2
Hex	1B 6B 32
Dec	27 107 50

The default state of the label taken sensor is set in the printer configuration using blazeConfig. This command allows the sensor to be enabled or disabled at print time.

PauseControl Esc * n… a

Turn the pause function on or off under program control.

Value for n…	0	Pause off immediately
	1	Pause on immediately
	2	Pause on when printing completes

Example:
Turn pause mode off

Text	Esc*0a
Hex	1B 2A 30 61
Dec	27 42 48 97

UserError Esc * U

Force an error condition, as if a print fault had occurred (eg. paper-out).

Example:
UserError

Text	Esc*U
Hex	1B 2A 55
Dec	27 42 85

This function can be used in conjunction with blazeFilter. For example when a label is scanned after printing, if the scan fails the printer can be put into an error state to alert the user and force a reprint.

DelayProcessing Esc * n… t

Delay command processing for the specified period.

n… Delay period in msec (resolution 10 msec).

Example:
Delay for 0.5 seconds

Text	Esc*500t
Hex	1B 2A 35 30 30 74
Dec	27 42 53 48 48 116

This function can be used to add a delay to label processing. Typically this will only be used for testing purposes. Resolution of this command is 10 msec, so it cannot be used for very short delays

Cash Drawer

The printer incorporates a dual cash drawer driver with 2 independent drawer kicks. There is also a single sense input which can be read back.

ActivateCashDrawer Esc u K n

Activate cash drawer 1 or 2.

n Cash drawer, 1 or 2

Example:
Activate cash drawer 1

Text	EscuK1
Hex	1B 75 4B 31
Dec	27 117 75 49

Sends a pulse timed pulse to the selected cash drawer. By default the pulse length is 100 msec, followed by a 100 msec period during which further cash drawer access is held pending.

SetCashActiveTime Esc u T nnn

Set the active and inactive time constants for the cash drawer driver.

nnn Time in msec (resolution 10 msec).

Example:
Set active and inactive times to 200 msec

Text	EscuT200
Hex	1B 75 54 32 30 30
Dec	27 117 84 50 48 48

If the default times are not suitable for the application (in most cases they are), this command allows the time to be changed. Note that this sets equal time periods for the active pulse and the inactive period following.

SetCashInactiveTime Esc u X nnn

Set the inactive time constant for the cash drawer driver.

nnn Time in msec (resolution 10 msec).

Example:
Set inactive time to 10 msec

Text	EscuX010
Hex	1B 75 58 30 31 30
Dec	27 117 88 48 49 48

This allows the inactive time to be changed if the default of equal active and inactive periods is not appropriate to the application.

CashDrawerStatus Esc u S

Example:
Get cash drawer status report

Text	EscuS
Hex	1B 75 53
Dec	27 117 83

This command returns a status report to show the state of the sense input. The report is a single character. In current versions of firmware the status report is always returned via the serial port, but future versions may return it via the comms channel currently in use. It is recommended to use this command only in serial port applications.

Status report

	Drawer Open	Drawer Closed
Text	o	c
Hex	6F	63
Dec	111	99

Cash drawer status can also be reported automatically whenever it changes using the ConfigAutoStatus command.

Status Reporting

The printer can produce a number of status reports to aid in debugging and automation applications. These reports are in human readable form and the format is not guaranteed to remain the same through firmware releases as the information and layout may be changed to improve clarity and readability. See the next section on machine-readable reports for versions which are more suitable for automation.

All status reports are in plain text format and can be read using a 'dumb' terminal attached to the printer serial port. On a PC, you can use HyperTerminal as supplied with Windows, or TeraTerm, which is a freeware application.

The status report is returned to the currently active comms port. If the port is not read for 45 seconds while status output is queued, the printer will not wait and the remainder of the report is discarded to avoid causing a deadlock. Output is re-enabled automatically when the port is read.

StatusHelp — Esc e ?

Example:
List the available status reports.

Text	Esce?
Hex	1B 65 3F
Dec	27 101 63

This report is simply a memory aid when using 'dumb' terminal testing.

AnalogStatus — Esc e a

Example:
Report the reading from each of the printer's analog-to-digital converter inputs.

Text	Escea
Hex	1B 65 61
Dec	27 101 97

Reports the instantaneous A-D readings, rescaled to show the value being read in useful units.

DigitalMonitor — Esc e b

Example:
Monitor the digital sensor inputs continuously.

Text	Esceb
Hex	1B 65 62
Dec	27 101 98

Reports changes to the digital sensor inputs continuously. Press any key to exit.

FontStatus — Esc e c

Example:
Show the list of installed fonts.

Text	Escec
Hex	1B 65 63
Dec	27 101 99

TachoStatus — Esc e d

Example:
Report the diameter measured by the tacho sensors.

Text	Esced
Hex	1B 65 64
Dec	27 101 100

GeneralStatus — Esc e e

Example:
Report all configuration settings, installed fonts, files and the printer error status.

Text	Escee
Hex	1B 65 65
Dec	27 101 101

This is a general status report giving the same information shown in the printer self-test report

SensorMonitor — Esc e g

Example:
Monitor the label sensor input continuously.

Text	Esceg
Hex	1B 65 67
Dec	27 101 103

Reports the label sensor reading continuously as an A-D value, and as a crude bar graph. Displaying the bar graph requires that the terminal treats the <CR> character as a simple carriage return and does not convert it to a <CR><LF> pair. Press any key to exit.

LogoStatus — Esc e l

Example:
List all logos stored in memory

Text	Escel
Hex	1B 65 6C
Dec	27 101 108

VersionStatus — Esc e v

Example:
Report printer type and firmware version.

Text	Escev
Hex	1B 65 76
Dec	27 101 118

ErrorStatus — Esc e x

Example:
Report error status indicated by Error LED.

Text	Escex
Hex	1B 65 78
Dec	27 101 120

FileStatus — Esc e z

Example:
Report files stored in file system.

Text	Escez
Hex	1B 65 7A
Dec	27 101 122

FileStatusLong — Esc e Z

Example:
Report files stored in file system and checks the integrity of each file.

Text	EsceZ
Hex	1B 65 5A
Dec	27 101 90

ShortStatus — Ctrl-E

Get short status report

Text	Ctrl-E
Hex	05
Dec	5

The short status report is a single character showing whether the printer is ready to print, and the fault condition if not ready. A fault condition is indicated by an uppercase character.

These status characters are also sent whenever the status changes if auto-reporting is configured (refer to ConfigAutoStatus).

Status characters

0	Printer ready
H	Head lock open
P	Paper out
R	Ribbon out
F	Rewinder full
C	Cutter jammed
*	Label printed (Auto report only)
o	Cash drawer open (Auto report only)
c	Cash drawer closed (Auto report only)

Machine-Readable Status

The machine-readable status reports are all presented in the format:
Key=Value<CR><LF>
The key remains constant through firmware updates, unlike the verbose status reports which may be changed to improve clarity. These reports are intended to be read by the blazeConfig program, and the full details of each report are not documented here.

Current versions of firmware write only to the serial port regardless of which port the command arrived on. Later versions may return the status report to the currently active comms port.

MRSensors — Esc e j

Example:
Machine-readable sensor status.

Text	Escej
Hex	1B 65 6A
Dec	27 101 106

MRConfig — Esc e m

Example:
Machine-readable configuration settings.

Text	Escem
Hex	1B 65 6D
Dec	27 101 109

MRFiles — Esc e r

Example:
Machine-readable file list.

Text	Escer
Hex	1B 65 72
Dec	27 101 114

MRCounters — Esc e t

Example:
Machine-readable file NV counters & formlength.

Text	Escet
Hex	1B 65 74
Dec	27 101 116

File System Commands

The printer contains a simple Flash file system which is used to store TrueType® fonts, images, configuration details and other data. The file system does not have a true directory structure, but the filenames can use '/' as a delimiter to make filename organisation easier.

The FileStatus report lists files in alphabetical order, so using directories has the effect of grouping similar files together.

Most file operations are performed through blazeConfig, so these commands are unlikely to be required in direct programming. Using blazeConfig is recommended as it uses CRC checking for the download which is beyond the scope of this manual.

Filenames may contain only printable characters, and may not start with a dot.

WriteTextFile — Esc ~WrItE-FiLe t… Eot T d… Eot

Write a simple text file to the printer Flash file system.

t…	File name, terminated with Eot
d…	File content, terminated with Eot

Example:
Write a file called 'Greet' containing the word 'Hello'.

Text	Esc~WrItE-FiLeGreetEotTHelloEot
Hex	1B 7E 57 72 49 74 45 2D 46 69 4C 65 47 72 65 65 74 04 54 48 65 6C 6C 6F 04
Dec	27 126 87 114 73 116 69 45 70 105 76 101 71 114 101 101 116 4 84 72 101 108 108 111 4

This is the simplest way of writing a text file, but is not suitable for binary files and has no CRC.

WriteBinaryFile — Esc ~WrItE-FiLe t… Eot n… B d…

Write a binary file to the printer Flash file system.

t…	File name, terminated with Eot
n…	File length
d…	File content

Example:
Write a file called 'Greet' containing the word 'Hello'.

Text	Esc~WrItE-FiLeGreetEot5Bhello
Hex	1B 7E 57 72 49 74 45 2D 46 69 4C 65 47 72 65 65 74 04 35 42 48 65 6C 6C 6F
Dec	27 126 87 114 73 116 69 45 70 105 76 101 71 114 101 101 116 4 53 66 72 101 108 108 111

This method is suitable for binary files but has no CRC.

WriteBinaryFileCRC Esc ~WrItE-FiLe t… Eot n… C d… cccccccc

Write a binary file to the printer Flash file system, with CRC to validate content.

t…	File name, terminated with Eot
n…	File length
d…	File content
cccccccc	8 hex digits of CRC for the file content

This method is suitable for binary files and includes a CRC to ensure that the file is only written if the CRC of the data received matches the transmitted CRC value. The CRC algorithm is beyond the scope of this document. This method is used by blazeConfig.

EraseFile Esc ~WrItE-FiLe t… Eot E

Erase a file from the printer Flash file system.

t…	File name, terminated with Eot

Example:
Delete the file called 'Greet'.

Text	Esc~WrItE-FiLeGreetEotE
Hex	1B 7E 57 72 49 74 45 2D 46 69 4C 65 47 72 65 65 74 04 45
Dec	27 126 87 114 73 116 69 45 70 105 76 101 71 114 101 101 116 4 69

This is the recommended method for erasing a file.

EraseFileIndex Esc ~WrItE-FiLe Eot n… I

Erase a file from the printer Flash file system, using its internal reference number.

n…	File index number

Example:
Delete the file with Index 8.

Text	Esc~WrItE-FiLeEot8I
Hex	1B 7E 57 72 49 74 45 2D 46 69 4C 65 04 38 49
Dec	27 126 87 114 73 116 69 45 70 105 76 101 4 56 73

This method can be used on files where the name is difficult to read, eg. contains a number of spaces which cannot be read on a dumb terminal. The file index can be obtained using the FileStatusLong command. Note that the file index may change whenever the file system changes, so a fresh index must be obtained immediately before using this command to avoid delting the wrong file.

ReadFile Esc ~ReAd-FiLe t… Eot

Read a file back from the printer Flash file system.

t…	File name, terminated with Eot

Example:
Read a file called 'Greet'.

Text	Esc~ReAd-FiLeGreetEot
Hex	1B 7E 57 72 49 74 45 2D 46 69 4C 65 47 72 65 65 74 04 54 48 65 6C 6C 6F 04
Dec	27 126 87 114 73 116 69 45 70 105 76 101 71 114 101 101 116 4 84 72 101 108 108 111 4

This command is used by blazeConfig for save/restore operations. The printer responds by sending 'ReAd-FiLe' followed by the file name terminated with Eot. It then sends the number of bytes in the file followed by 'C'. This is followed by the file content, and finally an 8 hexdigit CRC. If the file does not exist, the printer reponds with 'ReAd-FiLe' followed by the filename, followed by 'X'.

WriteFont Esc cl ff t… Eot nnnnnn d… cccccccc

Install a font file to the printer Flash file system, with CRC to validate content.

ff	Font number (used by SetScalableFont command)
t…	File name or description, terminated with Eot
nnnnnn	File length (6 digits)
d…	File content
cccccccc	8 hex digits of CRC for the file content

This command downloads a font file to the Flash file system and installs it, making it ready for use by the SetScalableFont command. The CRC algorithm is beyond the scope of this document.

EraseFont Esc cl ff Eot 000000

Uninstall a font and remove it from the printer Flash file system.

ff	Font number (used by SetScalableFont command)

This command uninstalls a printer font, then removes the font file from the Flash file system.

Input Data Capture

StartCapture — Esc ~CaPtUrE-StArT nn t… Eot

Start capturing input to a file.

nn	Capture file number 01 – 99, or 00 for AutoAllocate
t…	File name, terminated with Eot

Example:
Start capturing to the next available capture file, giving it the name 'TestFIle'

Text	Esc~CaPtUrE-StArT00TestFileEot
Hex	1B 7E 43 61 50 74 55 72 45 2D 53 74 41 72 54 30 30 54 65 73 74 46 69 6C 65 04
Dec	27 126 67 97 80 116 85 114 69 45 83 116 65 114 84 48 48 84 101 115 116 70 105 108 101 4

Capture files are used in Standalone/Demo mode. Typically, the printer will be put into capture mode, and a label will be 'printed' using a standard label design package. Instead of processing the label commands, the printer stores them in a file within the flash file system. The captured label can then be replayed later by pressing the Feed button.

After this command is sent, all data received is stored instead of being processed, until the sequence 'CaPtUrEsToP' is received.

Capture files are numbered. If the number used is 00, the printer will look for the first free capture file number and use that.

DeleteCapture — Esc ~CaPtUrE-DeLeTe nn

Delete a capture file.

nn	Capture file number 01 – 99, or 00 for all capture files.

Example:
Delete capture file 02

Text	Esc~CaPtUrE-DeLeTe02
Hex	1B 7E 43 61 50 74 55 72 45 2D 44 65 4C 65 54 65 30 32
Dec	27 126 67 97 80 116 85 114 69 45 68 101 76 101 84 101 48 50

This command removes a capture file from the flash file system.

Keyboard Data Capture

StartKeyboard — Esc ~KeYbOaRd-StArT nnnn t… Eot

Start capturing input to a file.

nnnn	Keyboard file number 0001 – 9999, or 0000 for AutoAllocate
t…	File name, terminated with Eot

Example:
Start capturing to the next available keyboard file, giving it the name 'KbdTest'

Text	Esc~KeYbOaRd-StArT00KbdTestEot
Hex	1B 7E 4B 65 59 62 4F 61 52 64 2D 53 74 41 72 54 30 30 4B 62 64 54 65 73 74 04
Dec	27 126 75 101 89 98 79 97 82 100 45 83 116 65 114 84 48 48 75 98 100 84 101 115 116 4

Keyboard files are used in conjunction with the Keyboard input data filter. The printer will be put into keyboard capture mode, and a label will be 'printed' using a standard label design package. Instead of processing the label commands, the printer stores them in a file within the flash file system. The captured label can then be loaded later using the attached keyboard, and variable data can be edited.

After this command is sent, all data received is stored instead of being processed, until the sequence 'CaPtUrEsToP' is received.

Keyboard files are numbered. If the number used is 0000, the printer will look for the first free keyboard file number and use that.

DeleteKeyboard — Esc ~KeYbOaRd-DeLeTe nnnn

Delete a keyboard file.

nnnn	Keyboard file number 0001 – 9999, or 0000 for all keyboard files.

Example:
Delete keyboard file 0002

Text	Esc~KeYbOaRd-DeLeTe0002
Hex	1B 7E 4B 65 59 62 4F 61 52 64 2D 44 65 4C 65 54 65 30 30 30 32
Dec	27 126 75 101 89 98 79 97 82 100 45 68 101 76 101 84 101 48 48 48 50

This command removes a keyboard file from the flash file system.

Mainframe Mode Commands

Mainframe mode is used when the host computer cannot provide 8 bit characters or control codes, or where the range of characters that can be output as ASCII is limited. This is often the case with mainframe systems, particularly IBM systems which use the EBCDIC character set. Mainframe mode is also useful in that it allows programming using plain text, which can be entered using any text editor.

The following set of commands themselves consist of purely printable characters, which means that the command to enter Mainframe mode can itself be sent from the mainframe.

Another feature of this set of commands is that they work regardless of whether mainframe mode is in effect.

SetMainframeMode %%XCCS c e

Select the special characters to use in mainframe mode.

c	Control character prefix
e	Escape shortcut

Example:
Set control char to '^'
Set escape char to '['

Text	%%XCCS^[
Hex	25 25 58 43 43 53 5E 5B
Dec	37 37 88 67 67 83 94 91

Selects the character set to be used for all subsequent communication. The Ctrl-prefix character is used to introduce sequences which represent the non-printable characters as described below. The Escape character is a single printable character which can be used to represent Esc which is the most common non-printable in BPL. If the Escape character is set to be the same as the Ctrl-prefix, then this second shortcut is disabled. The % character is not allowed for either of these special characters, as it would prevent any further %%XC commands from working.

Using mainframe mode

ASCII control characters received by the printer (ASCII 0-31) are discarded. In many systems, the print driver assumes that a line printer is connected and tries to divide the output into lines and pages, inserting carriage returns, linefeeds and formfeeds. In printable character mode, these artefacts are ignored.

ASCII control codes must be represented as a 2 character sequence Ctrl-prefix c. The printer performs a bitwise AND of this second character with 1F to generate the control character, e.g. <Eot> is represented as ^D.

A single character whose ASCII value is 127 or greater can be sent directly if the computer system supports these characters. If not, it can be represented as a 4 character sequence: Ctrl-prefix followed by '.' (dot) followed by 2 hex digits representing the 8-bit code. This will normally only be used for inserting non-ASCII characters such as £ or é.

Longer sequences of binary data may be encoded as a 2 character sequence Ctrl-prefix followed by ',' (comma), followed by the 2 hex digits for each byte of data, and terminated by a non-hexadecimal printable character, e.g. Q (which is discarded). This would normally be used for sending logos and other binary data.

If a literal Ctrl-prefix character is required, it is sent as a 2 character sequence Ctrl-prefix followed by '0' (zero).

If a literal Escape-shortcut character is required, it is sent as a 2 character sequence Ctrl-prefix followed by '1' (one).

These methods are best illustrated by example. Consider the following (nonsense) data, which prints both text and graphics, translated to printable character mode.

Control char mode	Ctrl-X EscT00100010Prefix ^ and price £10 Eot EscG0010005000320002 (binary data = 1A 2B 3C 4D 5E 6F 70 81) Ctrl-L
Mainframe mode	%%XCCS^[^X [T00100010 Prefix ^0 and price ^.9C10^D [G0010005000320002^, 1A2B3C4D5E6F7081q ^L

SetLegacyMainframeMode %%XCCS01

The method was used on older GPL2 printers and relies on the use of four special characters. Its use is **not recommended** for new applications, but may be required in older installations.

These are defined in terms of the action taken by the printer :-

#	The following character must be converted to a control character (AND with 1Fh).
!	The following character must have its top bit set (OR with 80h).
%	The following character must be converted to a high control character (AND with 1Fh, then OR with 80h).
&	Special characters: ! # % and & are interpreted literally. d is interpreted as Ascii 127. D is interpreted as Ascii 255. No other characters should be used after &.

All incoming non-printable characters are ignored. The following table shows the conversions for non-printable characters and special characters below 127.

Hex	Code	Hex	Code	Hex	Code	Hex	Code
00	#@	0A (LF)	#J	14	#T	1E	#^
01	#A	0B	#K	15	#U	1F	#_
02	#B	0C (FF)	#L	16	#V	21 (!)	&!
03	#C	0D (CR)	#M	17	#W	23 (#)	&#
04 (Eot)	#D	0E	#N	18	#X	25 (%)	&%
05	#E	0F	#O	19	#Y	26 (&)	&&
06	#F	10	#P	1A	#Z	7F	&d
07	#G	11	#Q	1B	#[(Esc)	FF	&D
08	#H	12	#R	1C	#\		
09	#I	13	#S	1D	#]		

SetControlCharMode %%XCCS00

Disables mainframe mode and reverts to standard control characters.

ExecuteTestPrint %%XCST

This command executes the printer's test printout. The test printout can also be obtained by powering on with the feed button pressed. The test printout provides information on the current printer settings and can be useful for installation testing.

ExecuteAlignmentPrint %%XCAT

This command prints a set of TOF alignment markers and a grey block to check sensor alignment and printhead integrity.

ExecuteTestFeed %%XCFF

This command executes a simple formfeed and is the most basic way of checking communication using printable characters.

ResetCommandInterpreter %%XCRC

This command forces the command interpreter to reset, reloading the input data filter if present.

Printer Configuration

Printer configuration data is stored permanently in configuration files in the Flash file system. The configuration is normally set up using blazeConfig supplied with the printer. However, it is also possible to send the commands directly. After changing the required values, the ConfigUpdate command must be sent to write the updated config file. It is important to note that the ConfigUpdate command should only be issued when re-configuration is required and not as part of a normal label program. The Flash device has a limited number of write cycles (typically 1,000,000), which could easily be exceeded if updated each time a label was printed.

In all cases where a distance is specified, the units are 0.125 mm, regardless of the global units of measure in effect.

ConfigSensor Esc # n… A

Defines how the gap/mark sensor selection operates. The following values for n are valid :-
n = 1 Sensor selected automatically on calibration.
n = 2 Always use reflective (black mark) sensor.
n = 6 Always use transmissive (label gap) sensor.

ConfigBaudRate Esc # n… B

Defines the baud rate to be used for serial communications. The value must lie in the range 110..912600, although values higher than 115200 are not recommended. For backward compatibility, values less than 110 are assumed to mean a multiple of 1200 baud, so a value of 8 represents 9600 baud. This older method is not recommended.

ConfigComms Esc # n… C

Defines the parity, data bits, stop bits and handshaking used for serial communications. This value is made up by summing component parts as follows :

Setting	Component Value
1 stop bit	0
2 stop bits	1
7 data bits	0
8 data bits	2
No parity	0
Odd parity	8
Even parity	12
Hardware handshake	0
Xon-Xoff handshake	16

Examples:
No parity, 8 data, 1 stop, hardware handshake = 0 + 2 + 0 + 0 = 2
Even parity, 7 data bits, 2 stop bits, Xoff = 12 + 0 + 1 + 16 = 29

ConfigRewindMax Esc # n… E

For printers with an internal rewinder fitted with a tacho sensor, these settings specify the limit values. Rewind max is the maximum possible spool diameter and is used to differentiate between very slow tacho output due to a large spool and no tacho output due to the spool being stopped. A value of 0 disables the sensor.

ConfigRewindHigh Esc # n… F

The rewind spool diameter above which a warning light will flash

ConfigRewindFull Esc # n… G

The rewind spool diameter above which the printer will stop printing with a rewind-full fault.

ConfigDefaultHeat Esc # n… H

Defines the print heat setting at power on. Range is 50 – 200%

ConfigRibbonMax Esc # n… I

For printers with a thermal transfer mechanism fitted with a tacho sensor, these settings specify the limit values. Ribbon max is the maximum possible spool diameter and is used to differentiate between very slow tacho output due to a large spool and no tacho output due to the spool being stopped. A value of 0 disables the sensor.

ConfigRibbonLow Esc # n… J

The ribbon spool diameter below which a warning light will flash.

ConfigRibbonOut Esc # n… K

The ribbon spool diameter below which the printer will stop printing with a ribbon-out fault.

ConfigMeasureSpeed Esc # n… M

The speed used for sensor calibration, manual paper-feed and back-feed.

ConfigAdvanceSelect Esc # n… N

Selects whether to set a paper advance by default at power-on. The distances are configured separately.

Setting	Component Value
No advance	0
Custom advance	1
Advance to tear bar	2
Advance to cutter	3

ConfigMediaType Esc # n... Q

Select the media type used at power-on. 0 = direct thermal, 1 = thermal transfer.

ConfigHeadComp Esc # n... R

Set the compensation factor to apply to all print heat settings. This may be used to get an exact match of heat settings when several printers are used together. Print head resistance is much more closely controlled in the manufacture of the printhead than it once was, so this setting is unlikely to be needed. Range is 80% to 120%.

ConfigDefaultSpeed Esc # n... S

The default print speed set at power-on. Range is 10 .. 250.

ConfigTOFoffset Esc # n... T

Set the top-of-form offset to use at power-on. This defines the distance between the black mark/gap and the actual top-of-form where printing starts. This is normally set to 0. A negative number is permitted with this command. Refer to SetTOFoffset for further information.

ConfigStockType Esc # n... V

Select whether to use label or continuous mode at power-on. 0 = label mode, 1 = continuous.

ConfigXOffset Esc # n... X

Set the Xoffset to be used by default at power-on. Normally set to 0.

ConfigAutoCentre Esc # n... Y

Select whether to use AutoCentre mode at power-on. This adjusts the X-offset automatically when the SetLabelWidth command is sent.

ConfigMainframeMode Esc # n... Z

Select whether to use mainframe mode, and which ctrl-prefix and escape-shortcut characters to use. The value given may be 0 to select standard control char mode, 1 to select legacy mainframe mode, or another value composed of (ascii value of the ctrl-prefix) + (256 x ascii value of escape shortcut).

Example:
To set the ctrl-prefix to '^' (ascii 94) and the escape shortcut to '[' (ascii 91), use n = 23390.

ConfigLabelTakenAdvance Esc # n... a

Set the advance distance to be used with the label-taken sensor.

ConfigCustomAdvance Esc # n... b

Set the advance distance to be used with the custom-advance command.

ConfigSensorOffset Esc # n… d

Set the distance between label sensor and printhead.

ConfigFaultAction Esc # n… f

Define the actions to be taken when the Feed button is pressed to clear a fault, and at power on. There are 4 levels of action which can be taken :-

Level 0 Realign to top of next label only if stopped part way through.
Level 1 Always realign to top of next label.
Level 2 Measure label length and realign to top of next label
Level 3 Calibrate sensor, measure and realign to top of next label

Action levels can be assigned to PowerOn, HeadOpen and PaperOut. The value to use is obtained by summing values for the options required as follows :

Setting	Component Value
Paper out – level 0	0
Paper out – level 1	1
Paper out – level 2	2
Paper out – level 3	3
Head open – level 0	0
Head open – level 1	8
Power on – level 0	0
Power on – level 1	16
Power on – level 2	32
Power on – level 3	48

The recommended setting is Level 1 for all 3 conditions, 1 + 8 + 16 = 25

ConfigHeadType Esc # n… h

Select the printhead driver to use. This must match the printhead in use. If this setting is changed the printer must be rebooted or power cycled before it will take effect.

Setting	Value
104 mm, 200 DPI	101
104 mm, 300 DPI	102

ConfigCutOffset Esc # n… k

Set the distance from the printhead to the cutter blades.

ConfigCodePage Esc # n… n

Select the text codepage number to use by default at power-on.

ConfigMotorCurrent Esc # n… o

Set the RMS motor current in mA.

ConfigFormRestore Esc # n… p

Used to restore form length and gap length. If n is less than 1,000,000 it represents the formlength. If n is greater than or equal to 1,000,000 it represents the gap length + 1,000,000.

ConfigLabelTaken Esc # n… q

Select whether or not to use the label-taken sensor at power-on. 0 = disable, 1 = enable.

ConfigSensorRestore Esc # n… s

Used to restore label sensor settings.

Range of n…	Sensor	Threshold
1000..1255	Black mark	n – 1000
2000..2255	Black mark 2	n – 2000
3000..3255	Gap	n – 3000
4000..4255	Spare	n – 4000

ConfigTearOffset Esc # n… t

Set the distance from the printhead to the tear bar.

ConfigCommsModifier Esc # n… u

Set the mode of operation for the ConfigBaud and ConfigComms commands. This allows the serial port settings to be changed on the fly without affecting the stored configuration, and the configuration to be changed without changing the serial port settings until the next reboot.

Value of n	Effect on serial port	Effect on config file
0	Changed	Updated
1	Changed	No change
2	No change	Updated

ConfigMotorAccel Esc # n… x

Set the motor acceleration constant in units of mm/s/s. The motor deceleration is set to twice the acceleration value.

ConfigAutoStatus Esc # n… z

Select which events are reported automatically via the serial port.

Setting	Value
Report status change	1
Report labels printed	2
Report cash drawer status	4

Sum the values associated with the reports required. eg. to report status change and cash drawer status, use a value of (1 + 4) = 5.

ConfigPauseFrequency Esc $ n… B

Set the value of pause-every-label to set at power-on.

ConfigCashDrawer Esc $ n… C

Configure the cash drawer sense input.

By default the input idles at 0V and sees a high input as the active state. Selecting active-low polarity reverses this. Add 1 to the cash drawer function code to select active-low functionality.

Select one of the functions below, which will be executed whenever an inactive-to-active transition is seen.

Function	Value
Standard cash drawer status	0
Set Pause off	2
Set Pause on	4
Toggle Pause	6
Demo file print	8

ConfigUpdate Esc # w

After changing the required values, the ConfigUpdate command must be sent to write the updated config file. It is important to note that the ConfigUpdate command should only be issued when re-configuration is required and not as part of a normal label program. The Flash device has a limited number of write cycles (typically 1,000,000), which could easily be exceeded if updated each time a label was printed.

Reboot Esc # ! !

This command reboots the printer firmware as if it had been switched off and back on.

CalibrateSensor Esc g

All Blazepoint printers include an automatic label sensor calibration feature that simplifies the sensor setup. This command initiates the calibration procedure. Pressing and holding the pause button for about 3 seconds will also initiate calibration. Before starting calibration, ensure that the label gap or black mark lies behind the sensor and within 100 mm so that when the printer feeds forward the gap/mark is drawn through the sensor. Positioning the gap/mark just behind the white roller is ideal.

If AutoSensor selection is enabled, the printer will examine both sensors and pick the one which gives the best signal.

Programming Example

This example program is written in BASIC for a 200 DPI printer.

Setup stage. Define some constants to make the control characters easier. Use COM1 port to send data to the printer. Set the printer buffer mode and measurement units.

```
eot$=chr$(4)
lf$ =chr$(10)
ff$ =chr$(12)
e$  =chr$(27)
gs$ =chr$(29)

open "com1:9600,n,8,1" as #1
```

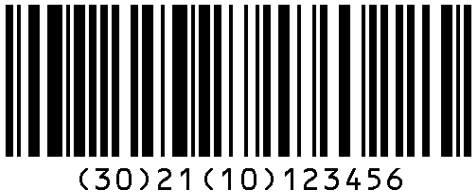
Testcard label with text, barcodes & grey block.

`print#1,e$;"ZM";e$;"ZB";`	Coordinates and distances in mm fonts aligned on baseline.
`print#1,e$;"Y0002002000";e$;"V2";`	20 point text, font 0, rotation 2
`print#1,e$;"T00900001ABCDEFGHIJK";lf$;`	Place 3 lines of text
`print#1,"abcdefghijklmn";lf$;`	starting at (90, 1)
`print#1,"1234567890";eot$;`	
`print#1,e$;"N6231";chr(2);`	Code-39 mag 2, 3:1, HRI on
`print#1,e$;"B00400027615ABCDq";`	Place barcode, height 15mm
`print#1,e$;"V1";e$;"N2200";`	Rotation 1, EAN-13 mag 2
`print#1,e$;"B0002000120050123456789O";`	Place barcode, default height
`print#1,e$;"I00450001002500506";`	Grey-filled box
`print#1,e$;"Y0200800800";`	8 point text, font 2
`print#1,e$;"T00020029";date$;lf$;time$;eot$;`	Show current date & time
`print#1,e$;"Y0000600600";`	6 point text, font 0
`print#1,e$;"T00020043abcdefghijklm";lf$;`	Place 3 lines of text
`print#1,"abcdefghijklm";lf$;`	
`print#1,"abcdefghijklm";eot$;`	
`print#1,ff$`	Print the label

EAN-128 barcode example

This example is taken from the ANA manual and describes a quantity of 21 and batch number 123456. Note the use of GS to terminate the quantity field (which is of variable length), when concatenated with a batch number.

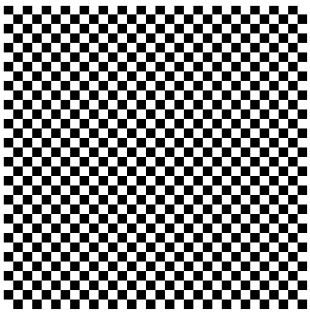
```
print#1,e$;"N8400";                          Barcode mag 4
print#1,e$;"B00200010800";                   EAN-128, default height,
print#1,"(30)21";gs$;"(10)123456";eot$;      barcode data.
print#1,ff$;                                 Print label
```



Logo example

Normally a logo will be a graphic image, but for simplicity, this demo defines a simple chequerboard logo and places it on the page. Note that although this demo uses magnification, it is better to use unmagnified logos in real applications.

```
print#1,e$;"K0LogoDemo  00640064";        Define a 64x64 pixel logo
for i = 1 to 16                           Each row is 8 bytes long.
  for j = 1 to 16                         2 rows of $CC followed by
    print#1,chr$(204);                    2 rows of $33, etc.
  next j
  for j = 1 to 16
    print#1,chr$(51);
  next j
next i
print#1,e$;"M0505";                       Set magnification 5
print#1,e$;"L000500050";                  Place magnified logo at (5,5)
print#1,ff$;                              Print label
```



Background buffer example

Demonstrates how invariant fields can be formatted once and re-used on subsequent labels.

`print#1,e$;"i1";`	Enable background buffer
`print#1,e$;"Y0202402400";`	Set font 2, point size 24
`print#1,e$;"T00050040Printing Supplies";eot$;`	Large text
`print#1,e$;"M0303";e$;"L0050020";`	Place logo 0 again, mag 3x3
`print#1,e$;"N9200";`	Set Code128 to mag 2
`print#1,e$;"B0038001591000Supplies";eot$;`	Place Code128 barcode
`print#1,chr$(14);`	Save background
`print#1,e$;"Y0001601600";`	Set font 0, point size 16
`print#1,e$;"T00400010Printers";eot$;`	Place text
`print#1,ff$;`	Print label
`print#1,chr$(16);`	Copy background
`print#1,e$;"T00400010Labels";eot$;`	Place text
`print#1,ff$;`	Print label
`print#1,chr$(16);`	Copy background
`print#1,e$;"T00400010Ribbons";eot$;`	Place text
`print#1,ff$;`	Print label
`print#1,e$;"i0";e$;"M0101";e$;"X0";`	Background buffer off, mag 1x1, Delete logo 0.

Printing Supplies

Printing Supplies

Printing Supplies

Variable fields example

Sets up a single variable and uses it to print a fixed width barcode and some text.
Also shows the use of the date formatting commands and date offset.

```
ldate$=date$                                                    Copy system date & time
ltime$=time$
set1$=mid$(ldate$,4,2)+left$(ldate$,2)+right$(ldate$,2)         Format as DDMMYYhhmmss
set2$=left$(ltime$,2)+mid$(ltime$,4,2)+right$(ltime$,2)
print#1,e$;"fd010+01";eot$;                                     Define field 0 as max width 10,
                                                                increment by 1
print#1,e$;"v0123450";eot$;                                     Set initial value to 123450
print#1,e$;"N4252";                                             Set Int-2-of-5 module size
print#1,e$;"D0FormatName";                                      Start definition of format 0
print#1,e$;"ft";                                                Update date-time field
print#1,e$;"Y0001601600";                                       Font 0, point size 16
print#1,e$;"T00100010Serial number: ";                          Start of first line
print#1,chr$(1);"0";eot$;eot$;                                  Insert variable 0, and end text.
print#1,e$;"Y0001201200";                                       Font 0, point size 12
print#1,e$;"T00100015Manufactured: ";                           Start of 2nd line
print#1,chr$(1);"t#D-#M-#Y #h:m:s";eot$;eot$;                   Insert date & time, end text
print#1,e$;"T00100020Display until: ";                          Start of 3rd line
print#1,chr$(1);"t+30D-M-Y";eot$;eot$;                          Insert offset date, end text
print#1,e$;"T00100025Best before: ";                            Start of 4th line
print#1,chr$(1);"t+60D-M-Y";eot$;eot$;                          Insert offset date, end text
print#1,e$;"B00100030420";                                      Start of barcode
print#1,chr$(1);"0=10";eot$;"q";                                Variable 0 padded with
                                                                leading 0 to 10 characters.
print#1,ff$;e$;"fu";                                            Print label & update fields
print#1,chr$(11);                                               End of format 0
print#1,e$;"j00003";                                            Execute format 0 three times
```

Serial number: 123450
Manufactured: 24-3-2004 11:50:30
Display until: 23-04-04
Best before: 23-05-04

0000123450

Serial number: 123451
Manufactured: 24-3-2004 11:50:31
Display until: 23-04-04
Best before: 23-05-04

0000123451

Serial number: 123452
Manufactured: 24-3-2004 11:50:32
Display until: 23-04-04
Best before: 23-05-04

0000123452

Application Notes

Optimising Print Throughput

Blazepoint printers are designed to give maximum throughput, and the following general guidelines should be followed when writing programs.

- Use repeat print commands when producing a batch of identical labels.
- Use the background buffer or copy the previous print buffer when working with complex and mainly invariant labels.
- Do not magnify or rotate logos. If possible, download them with the orientation & size that will be used.
- Use logos rather than graphics if the graphical data will be required more than once, to save download time.
- Trim surrounding white space from logos to minimise the area downloaded, stored & placed.
- Use stored formats and variable fields to reduce the amount of data sent to the printer.
- Avoid unnecessary commands, e.g. repeatedly setting rotation & point size.

When designing a program to print non-identical labels, it is important to decide on the basic strategy for printing. If every label is completely different, then all the field layout commands can be sent each time. If the layout is always the same but the data is different, then using stored formats and variable fields can reduce the amount of data traffic, since only the variable information needs to be sent. If 80% of the label design is invariant between labels, then it makes more sense to re-use the invariant part rather than format it for every label, using the background buffer.

Optimising Print Quality

Blazepoint printers are designed to produce optimum print quality with a broad range of media. The following guidelines will enable you to produce the highest quality output.

- Do not set the print speed faster than is really needed. As with all thermal printers, print quality reduces as the speed increases. If the printer is pausing between labels, reducing the speed may allow it to print continuously, improving print quality and reducing noise and wear, without reducing overall throughput.
- Select media appropriate to the conditions of use. Print quality is to a very large extent dependent on the papers and ribbons in use. Some papers are optimised for high speeds, whereas others will blur. Wax ribbons can be used at higher speeds than resin ribbons, although the durability of the print is not as good.
- Print barcodes so that the bars are perpendicular to the print head (picket fence orientation). This gives the best readability as there is no blurring between bars. However, barcodes produced in this orientation can become unreadable if there is any print head damage. Printing barcodes with the bars parallel to the head (step ladder orientation) provides maximum tolerance of print head damage, but print speed must be controlled to prevent the bars from spreading.
- Keep the print head and paper/ribbon path clean. Accumulated dust and dirt on the head reduces the efficiency of the heat transmission to the paper, producing uneven results.

ANA Barcode Standard Sizes

The Article Number Association (ANA) specify a nominal barcode size for several barcode types, along with the range of sizes which are acceptable as a percentage of the nominal size. For example, EAN-13 barcodes may be printed in the range 80% - 200% of the nominal size. This table shows which sizes of barcode can be printed, and the magnification and bar height values to be used.

	8 dots/mm			12 dots/mm		
Barcode type	% of nominal	Mag	Height (mm)	% of nominal	Mag	Height (mm)
EAN-8, EAN13, UPC-A, UPC-E	75%	2	18	75%	3	18
	114%	3	28	100%	4	25
				125%	5	31
	152%	4	37	150%	6	37
				175%	7	43
	189%	5	46			
				200%	8	49
ITF-14 Wide=5, Narrow=2				66%	4	21
	74%	3	24			
				82%	5	26
	98%	4	31	98%	6	31
				115%	7	37
	123%	5	39			
EAN-128	25%	2	20	25%	3	20
				33%	4	20
	37%	3	20			
				42%	5	20
	50%	4	20	50%	6	20
				58%	7	20
	62%	5	20			
				67%	8	21
	75%	6	24	75%	9	24
				82%	A	27
	87%	7	27			
				92%	B	29
	100%	8	32	100%	C	32
				108%	D	34
	112%	9	36			
				117%	E	37

Selecting a Variable Length Barcode

A wide variety of barcode types is available for general data use. This guide explains the strengths and weaknesses of some of the commonest types and the typical application areas.

The barcode sizes are given in dots assuming a magnification of 1. 'w' indicates the wide bar width and 'n' the narrow bar width.

Code-128
Encodes the full ASCII character set. It is widely used and is the most compact barcode for mixed case alphanumerics and control chars, and in many instances for numerics-only. It is also very reliable. Calculating the length of the barcode is not straightforward due to the optimisations built in to the spec which allow for compression of long strings of numbers as 2-per-character, or using control chars which usually require an extra shift character. As a general rule :-

Total dots in barcode = 11 * chars + 35 (magnification = 1)

EAN-128
This is really a data format imposed on a Code-128 barcode, specifically used for distribution data. There is a wide variety of information that can be specified including article numbers, batch codes, best-before dates, shipping code, etc. For full details consult the ANA manual.

Code-39
Encodes uppercase characters, numbers, space and a few punctuation chars. Code39 is widely used throughout the Auto-ID sector and is very reliable, although not very efficient in terms of space. Typically used with wide bar size of 3 and narrow bar of 1.

Total dots in barcode = (3w + 7n) * (chars + 2)

Code-93
A more recent code, which uses the same character set as Code-39, but can also encode the full ASCII character set by use of additional shift characters. This is the most compact barcode for uppercase and numbers only, but is not widely used.

Total dots in barcode = 9 * chars + 37

Interleaved 2-of-5
A well established code, it encodes numbers only. If the number of digits is odd, a leading zero is added to pad to an even number. This is a compact barcode for numbers, although it can be surpassed by Code-128. It should really be used with wide bar set to 5 and narrow set to 2, in which case mag = 1 is roughly equivalent to mag = 2 in other barcodes. It can also be used at wide=3, narrow=1, mag=1. It has a well documented weakness, which is that an oblique scan can result produce a truncated read without showing an error. For this reason, many ITF applications use 'bearer bars' – solid horizontal lines – at top and bottom of the barcode, to force termination of an oblique scan.

Total dots in barcode = (2w + 3n) * chars + 7n + w

Codabar
This is one of the oldest barcodes and does not have high read reliability. However, it is still used in several sectors. It encodes numbers and a few punctuation chars.

Total dots in barcode = (3w + 5n) * (chars + 2)

Connecting to a Personal Computer (PC) Serial Port

Most applications for the printer involve connection to a personal computer or similar small system. This is generally quite straightforward with parallel, USB and network connections, but there are a few issues with the serial port which sometimes cause problems.

The physical connection is normally very simple. The cable can be supplied by Blazepoint Ltd, or you can use a standard 'straight-through' cable. Some PCs have more than one serial (COM) port. Check which port you are using.

Check that the serial port settings are the same on the printer as on the PC – baud rate, parity, data, start and stop bits. The printer is preset at the factory to use 9600,N,8,1 with hardware handshaking.

The program must use some form of flow control, and serial ports are notorious for being awkward in this respect. Firstly, if hardware handshaking is being used, the lines must be physically connected. Most 9-way cables do have all lines connected, but some cables may not. Secondly, the program must be set up to use flow control, and this varies with the programming environment being used. The most common symptom of a flow control failure is label corruption. The first few labels print correctly but subsequent labels have fields missing, misplaced, or in the wrong font or rotation. The printer detects flow control failure and indicates this using an occasional double blink on the Error LED. The only good solution to this problem is to ensure that flow control is implemented correctly, for which you will need to check the application programming manuals. It is possible to include in the application a delay loop between labels, but this is not a good solution for two reasons. Firstly, the delay will depend on the PC hardware and will probably stop working if the PC is upgraded. Secondly, it does not work if there is any sort of hold up at the printer (paused, head open, paper out, etc).

To view the printer status reports, you must have a serial connection to the printer. In Windows you can use HyperTerminal (or TeraTerm, which is faster and more reliable) with a connection direct to COMn. This allows you to enter commands to the printer directly and see the status reports. The printer will support concurrent serial, parallel, USB and network connections, so if your application is using the parallel, USB or network, you can still see request and check status on the serial port. If you application is using the serial port, then you will need to ensure that your application closes the port before trying to connect with HyperTerminal, and you will need to disconnect the HyperTerminal connection before reconnecting your application. Note that in Windows NT, 2000 and XP, if the serial port is assigned to a printer driver you will not be able to access it using any other program, except through the Windows print spooler.

Connecting to Mainframe Systems

Connecting to a mainframe system is more difficult as there are often more components in the chain between application program and printer. This manual cannot deal with particular systems, since every installation is different. The following section outlines the general principles which will apply to most systems.

The physical connection to the printer can be achieved in many ways. Some systems will have spare serial ports on the main computer or on an adapter. This is often the case with Unix systems. Another method is to connect the printer to the printer port of a terminal. Some have serial and some have parallel outputs. In other systems, a print server provides centralised output in serial or parallel data form.

Before connecting the printer, check whether the connection is serial (RS232) or parallel (Centronics) as the two systems are fundamentally different, but often use the same D-type connector. Connecting serial ports to parallel ports can result in physical damage to the parallel port.

For serial connections check the RS232 pinout and make sure that the host transmit lines are routed to printer receive lines. RS232 is notorious for being a 'non-standard' standard. The most reliable check is to use a breakout box which shows which of the RS232 lines are active. Check that the serial port settings are the same on the printer as on the host – baud rate, parity, data, start, stop bits.

Check that the connection has flow control (handshaking) operating, and that it operates correctly back through to the host. Centronics is generally trouble-free in this respect, but RS232 has several different handshake methods, and the printer and host must use the same method. The printer serial port supports RTS/CTS, DTR/DSR and XON/XOFF flow control. RTS/CTS and DTR/DSR are enabled by default.

Many mainframe systems cannot send control characters to the printer, or convert them to different values. Others assume that all printers are line printers and try to divide the output into lines and pages by inserting linefeeds and formfeeds periodically. This can seriously disrupt the commands being sent to the printer. The simplest way to avoid these problems is to use printable character mode, detailed earlier in the manual. In this mode, all printer commands consist of plain printable text characters and any control characters which the system inserts are discarded.

One way of checking exactly what data the printer is receiving is to enable hex-dump mode. This can be enabled by powering on the printer with the Pause button pressed (see the printer operating manual). Another way of checking the data on a serial connection is to connect a dumb terminal in place of the printer. Ideally it should be configured to display all characters rather than interpret control characters.

Once a working connection is established, the most common fault is label corruption, where the first few labels print OK, but subsequent labels have missing fields or unexplained font changes or rotations. This is normally due to a failure of flow control at some point which causes commands to be lost, or to the insertion of extra characters by the system, or to the system changing or blocking special characters, either of which will corrupt the commands received. These problems are dealt with above. The printer can detect flow control errors on the serial port and will give an occasional double blink of the error LED to indicate the problem.

BPL Programming Recommendations

When new programs are being designed it is important to select commands which will reduce the impact of changing printer models in future.

- Select a coordinate and measurement unit which is independent of the print resolution. Most printers currently have a resolution of 8 dots per mm, and 12 dots per mm is also becoming common. In time as printing technology improves the 8 dot print heads may become obsolete. Units of 0.1mm are recommended for general use as the units are easy to use and provide good resolution. Many of the BPL commands still have fixed 4 digit coordinates, so this unit allows for measurements up to 999.9 mm.
- Select Baseline referencing for fonts. For compatibility reasons, the default is the top of an imaginary character cell, but this makes placement highly dependent on the typeface and font size selected. Baseline referencing is strongly recommended.
- Use scalable fonts rather than bitmap fonts. The bitmap fonts are no longer supplied and are implemented as an emulation using the scalable fonts. Using the proper scalable font commands gives far more control over the fonts being used.
- Do not use inter-character gaps. These were a crude method of adjusting font widths when only bitmap fonts were available. The better way is to adjust the point size of scalable fonts. Note that horizontal and vertical point sizes can be independently adjusted.
- Use the SetBarExtendedHeight command to allow 4 digits to be used to represent barcode height, rather than the default of 2. This permits the use of almost any unit of measurement, rather than the default of mm for barcode height and dots for the barcode coordinates.
- Graphics of any sort and barcode module sizes are inherently limited to the printer pixel size, and so cannot be rescaled.
- When using printable character mode (mainframe mode) use the newer form which allows you to select the characters used for control prefix and (optionally) an escape character, rather than the legacy mode which requires the use of 4 special characters, and the need to escape some commonly used characters.
- Printable character mode is very useful when writing trial applications, as everything (except the most basic graphics) can be entered using a standard text editor.

Code Page Table

The printer has several code page tables (refer to SetCodePage command). The default code page is Windows 1252 and the character mapping is shown below. Other code page tables may be printed using the AsciiTable label file in blazeConfig.

The number shown in the top left corner is the decimal character value. The hexadecimal value is given by the number at the top of the column followed by the number of the row containing the character. eg. the 'A' character is decimal 65 and hex 41.

	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	32	48 0	64 @	80 P	96 `	112 p	128 €	144 □	160	176 °	192 À	208 Đ	224 à	240 ð
1	33 !	49 1	65 A	81 Q	97 a	113 q	129 □	145 ‘	161 ¡	177 ±	193 Á	209 Ñ	225 á	241 ñ
2	34 "	50 2	66 B	82 R	98 b	114 r	130 ‚	146 ’	162 ¢	178 ²	194 Â	210 Ò	226 â	242 ò
3	35 #	51 3	67 C	83 S	99 c	115 s	131 ƒ	147 “	163 £	179 ³	195 Ã	211 Ó	227 ã	243 ó
4	36 $	52 4	68 D	84 T	100 d	116 t	132 „	148 ”	164 €	180 ´	196 Ä	212 Ô	228 ä	244 ô
5	37 %	53 5	69 E	85 U	101 e	117 u	133 …	149 •	165 ¥	181 µ	197 Å	213 Õ	229 å	245 õ
6	38 &	54 6	70 F	86 V	102 f	118 v	134 †	150 –	166 ¦	182 ¶	198 Æ	214 Ö	230 æ	246 ö
7	39 '	55 7	71 G	87 W	103 g	119 w	135 ‡	151 —	167 §	183 ·	199 Ç	215 ×	231 ç	247 ÷
8	40 (	56 8	72 H	88 X	104 h	120 x	136 ˆ	152 ˜	168 ¨	184 ¸	200 È	216 Ø	232 è	248 ø
9	41)	57 9	73 I	89 Y	105 i	121 y	137 ‰	153 ™	169 ©	185 ¹	201 É	217 Ù	233 é	249 ù
A	42 *	58 :	74 J	90 Z	106 j	122 z	138 Š	154 š	170 ª	186 º	202 Ê	218 Ú	234 ê	250 ú
B	43 +	59 ;	75 K	91 [	107 k	123 {	139 ‹	155 ›	171 «	187 »	203 Ë	219 Û	235 ë	251 û
C	44 ,	60 <	76 L	92 \	108 l	124 |	140 Œ	156 œ	172 ¬	188 ¼	204 Ì	220 Ü	236 ì	252 ü
D	45 -	61 =	77 M	93]	109 m	125 }	141 □	157 □	173 -	189 ½	205 Í	221 Ý	237 í	253 ý
E	46 .	62 >	78 N	94 ^	110 n	126 ~	142 Ž	158 ž	174 ®	190 ¾	206 Î	222 Þ	238 î	254 þ
F	47 /	63 ?	79 O	95 _	111 o	127 □	143 □	159 Ÿ	175 ¯	191 ¿	207 Ï	223 ß	239 ï	255 ÿ

Alphabetical List of Commands

This section contains a listing of the commands in alphabetical order of the command sequence rather than the command name. This is to assist in decoding an existing printer command file.

Command name	Command sequence summary	Page
GetFieldValue	Ctrl-A r o… Eot	44
GetTimeValue	Ctrl-A t o… Eot	45
EnableBarcodeHRI	Ctrl-B	19
DisableBarcodeHRI	Ctrl-C	19
ShortStatus	Ctrl-E	58
FormFeed	Ctrl-L	36
SaveBackground	Ctrl-N	46
CopyBackground	Ctrl-P	46
SetContinuousMode	Ctrl-R	7
SetLabelMode	Ctrl-T	7
ClearAllBuffers	Ctrl-X	47
ClearImageBuffer	Ctrl-Y	47
EnableCutting	Ctrl-\	51
DisableCutting	Ctrl-]	51
SetFormLength	Esc **A** yyyy	7
CancelPrinting	Esc **a**	36
PlaceITF14	Esc **B** xxxx yyyy **1** hh o… d…	20
PlaceEAN13	Esc **B** xxxx yyyy **2** hh o… d…	21
PlaceEAN8	Esc **B** xxxx yyyy **3** hh o… ddddddd	21
PlaceInt2of5	Esc **B** xxxx yyyy **4** hh o… d… **q**	22
PlaceCode39	Esc **B** xxxx yyyy **6** hh o… d… **q**	22
PlaceCodabar	Esc **B** xxxx yyyy **7** hh o… d… **q**	23
PlaceEAN128	Esc **B** xxxx yyyy **8** hh d… Eot	24
PlaceCode128	Esc **B** xxxx yyyy **9** hh o… nn d… Eot	25
PlaceUPCA	Esc **B** xxxx yyyy **A** hh o… d…	25
PlaceUPCE	Esc **B** xxxx yyyy **B** hh o… d…	26
PlaceCode93	Esc **B** xxxx yyyy **C** hh Eot d… Eot	26
PlaceBarcode	Esc **B** xxxx yyyy t hh o… d…	18
CutImmediate	Esc **C I**	50
EraseFont	Esc **cI** ff Eot 000000	62
WriteFont	Esc **cI** ff t… Eot nnnnnn d… cccccccc	62
DefineFormat	Esc **D** n ttttttttt d… Ctrl-K	39
StatusHelp	Esc **e ?**	56
AnalogStatus	Esc **e a**	56
DigitalMonitor	Esc **e b**	56
FontStatus	Esc **e c**	56
TachoStatus	Esc **e d**	57
GeneralStatus	Esc **e e**	57
SensorMonitor	Esc **e g**	57
MRSensors	Esc **e j**	59
LogoStatus	Esc **e l**	57
MRConfig	Esc **e m**	59

EraseFormat	Esc **E** n	39
MRFiles	Esc **e r**	59
MRCounters	Esc **e t**	59
VersionStatus	Esc **e v**	57
ErrorStatus	Esc **e x**	58
FileStatus	Esc **e z**	58
FileStatusLong	Esc **e Z**	58
SetBitmapFont	Esc **F** n c	17
DefineField	Esc **fd** r ww o… Eot	41
ListFields	Esc **fl**	43
SetRealTimeClock	Esc **fs** DD MM YY hh mm ss	43
UpdateTimeField	Esc **ft**	43
UpdateFields	Esc **fu**	42
CalibrateSensor	Esc **g**	73
DirectGraphics	Esc **G** xxxx yyyy wwww hhhh d…	31
RetrieveFlashLogo	Esc **H #** n ffff	35
ProtectLogo	Esc **H** n	35
SetHeat	Esc **h** nnn	38
BackgroundBuffer	Esc **i** n	46
BlockFill	Esc **I** xxxx yyyy wwww hhhh c	30
LoadFormat	Esc **J** n	40
RepeatLoadFormat	Esc **j** n rrrr	40
DefineFlashLogo	Esc **K #** n ffff	34
EnableTakenSensor	Esc **k** n	52
DefineLogo	Esc **K** n ttttttttt wwww hhhh d…	32
PlaceFlashLogo	Esc **L** xxxx yyyy **#** nnnn	34
PlaceLogo	Esc **L** xxxx yyyy n	32
SetVariableFormLength	Esc **l** yyyy	8
SetSpeed	Esc **m** nnn	38
SetMagnification	Esc **M** vv hh	17
SetBarExtendedHeight	Esc **N** h	19
SetBarcodeModules	Esc **N** t m w n	18
SetBarMargins	Esc **NZ** m	20
SetBarAlignment	Esc **NZ** v	19
SetAdvanceToCutter	Esc **p C**	48
AdvanceImmediate	Esc **p I**	49
SetAdvanceDistance	Esc **p L** yyyy	49
EnableAdvance	Esc **p N**	49
SetAdvanceToTearBar	Esc **p T**	48
SetAdvanceCustom	Esc **p U**	48
DisableAdvance	Esc **p X**	49
SetTOFoffset	Esc **P** yyyy	8
SetMediaType	Esc **q** n	9
RepeatPrint3	Esc **R** nnn	36
RepeatPrint	Esc **r** nnnn	36
SetSpeed2	Esc **S** nn	38
PlaceText	Esc **T** xxxx yyyy t… Eot	13

SetTabWidth	Esc **U** hhh	15
ActivateCashDrawer	Esc **u K** n	54
CashDrawerStatus	Esc **u S**	55
SetCashActiveTime	Esc **u T** nnn	54
SetCashInactiveTime	Esc **u X** nnn	54
SetRotationAngle	Esc **V 0** rrr	10
SetRotation	Esc **V** r	10
SetFieldValue	Esc **v** r d… Eot	42
SetFieldXoffset	Esc **wX** xxxx	12
DeleteFlashLogo	Esc **X #** ffff	34
DeleteLogo	Esc **X** n	33
SetScalableFont	Esc **Y** vvv hhh i k	13
SetLeftRightAlignment	Esc **Z** a	16
SetTextBaseline	Esc **Z** b	16
SetUnits	Esc **Z** u	6
PlaceDataMatrix	Esc **:** xxxx yyyy **E** rrr ccc mmmm s d… Eot	27
PlacePDF417	Esc **:** xxxx yyyy **P** ww hh aa rr cc e t d… Eot	28
PlaceMaxiCode	Esc **:** xxxx yyyy **U** m n t e d… Eot	29
PauseControl	Esc ***** n… **a**	52
PauseEveryNLabels	Esc ***** n… **b**	52
CutEveryNLabels	Esc ***** n… **C**	50
SetGraphicsCompression	Esc ***** n… **E**	33
SetCodePage	Esc ***** n… **f**	15
SetLabelHeight	Esc ***** n… **H**	11
PrintSpecial	Esc *** K**	37
SetNumBuffers	Esc ***** n… **N**	11
PointSizeScaling	Esc ***** n… **P**	14
SetGreyShade	Esc ***** n… **S**	30
DelayProcessing	Esc ***** n… **t**	53
SetInlineText	Esc ***** n… **T**	14
UserError	Esc *** U**	53
SetLabelWidth	Esc ***** n… **W**	12
SetAutoCentre	Esc ***** n… **Y**	12
BackfeedLabel	Esc *** Z**	37
ConfigLabelTakenAdvance	Esc **#** n… **a**	70
ConfigSensor	Esc **#** n… **A**	68
ConfigCustomAdvance	Esc **#** n… **b**	70
ConfigBaudRate	Esc **#** n… **B**	68
ConfigComms	Esc **#** n… **C**	68
ConfigSensorOffset	Esc **#** n… **d**	71
ConfigRewindMax	Esc **#** n… **E**	69
ConfigFaultAction	Esc **#** n… **f**	71
ConfigRewindHigh	Esc **#** n… **F**	69
ConfigRewindFull	Esc **#** n… **G**	69
ConfigHeadType	Esc **#** n… **h**	71
ConfigDefaultHeat	Esc **#** n… **H**	69

Document Change History

Document revision	Firmware release	Change history
1	1.27	Produced from GPL2 programming manual. Completely revised and updated.
2	1.28	Capture and Keyboard sections added. New BPL commands added: PauseControl, CancelPrinting.
3	1.37	UserError command added. DelayProcessing timer granularity reduced from 100 to 10 msec. Status reports sent to the current active comms port.
4	1.38	PrintSpecial and BackfeedLabel commands added.