Chapter 5: Text Processing

Contents of this Chapter

This chapter describes the following PCL commands:

5.1 Introduction

Whereas Chapter 4 described the processing of character codes within escape sequences, this chapter describes the processing of character codes other than escape sequences.

PCL devices normally print any character code that is not a valid escape sequences or control code. A device receiving a valid escape sequence or control code suspends character formatting, performs the specified action, and begins formatting again.

The following terms are used in this chapter:

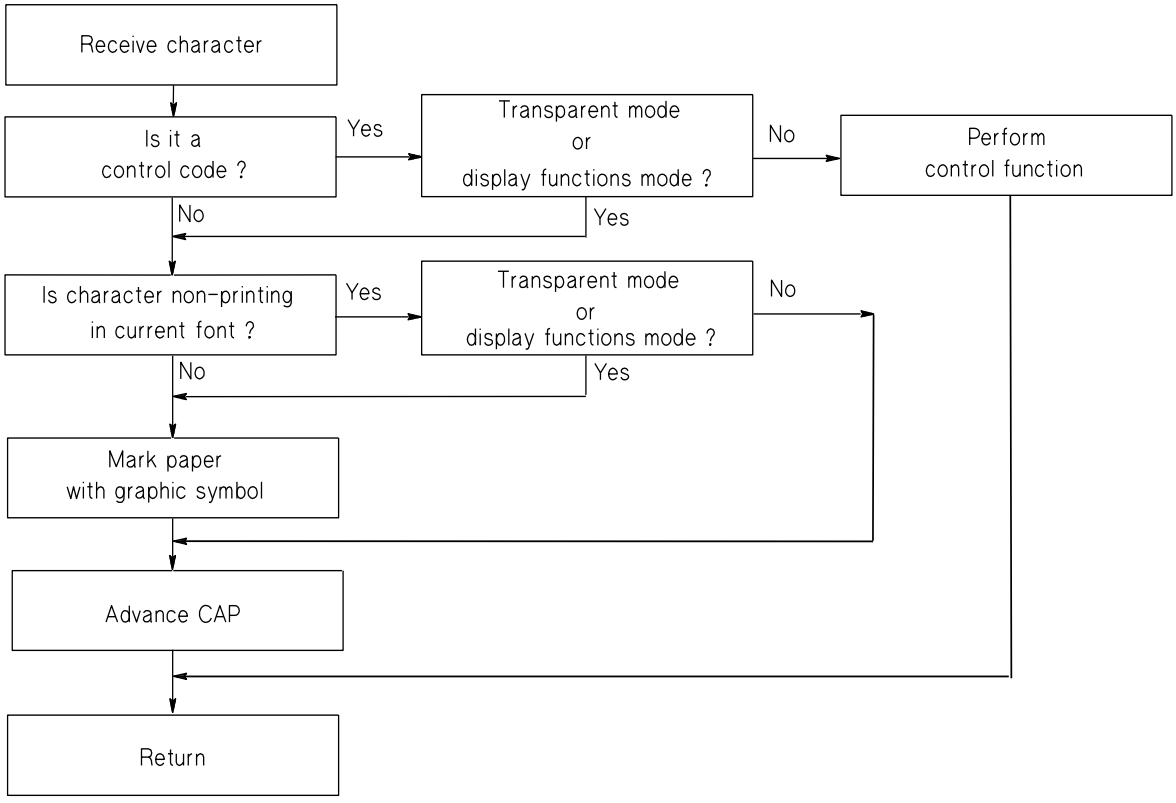
Character Code — n-byte binary code representing a character.

Control Code — A character code that initiates a device function (e.g., formfeed). Graphics symbols associated with control codes may be printed by using the Transparent Data Transfer command.

Symbol Set — A mapping of character codes to glyphs (graphic symbols).

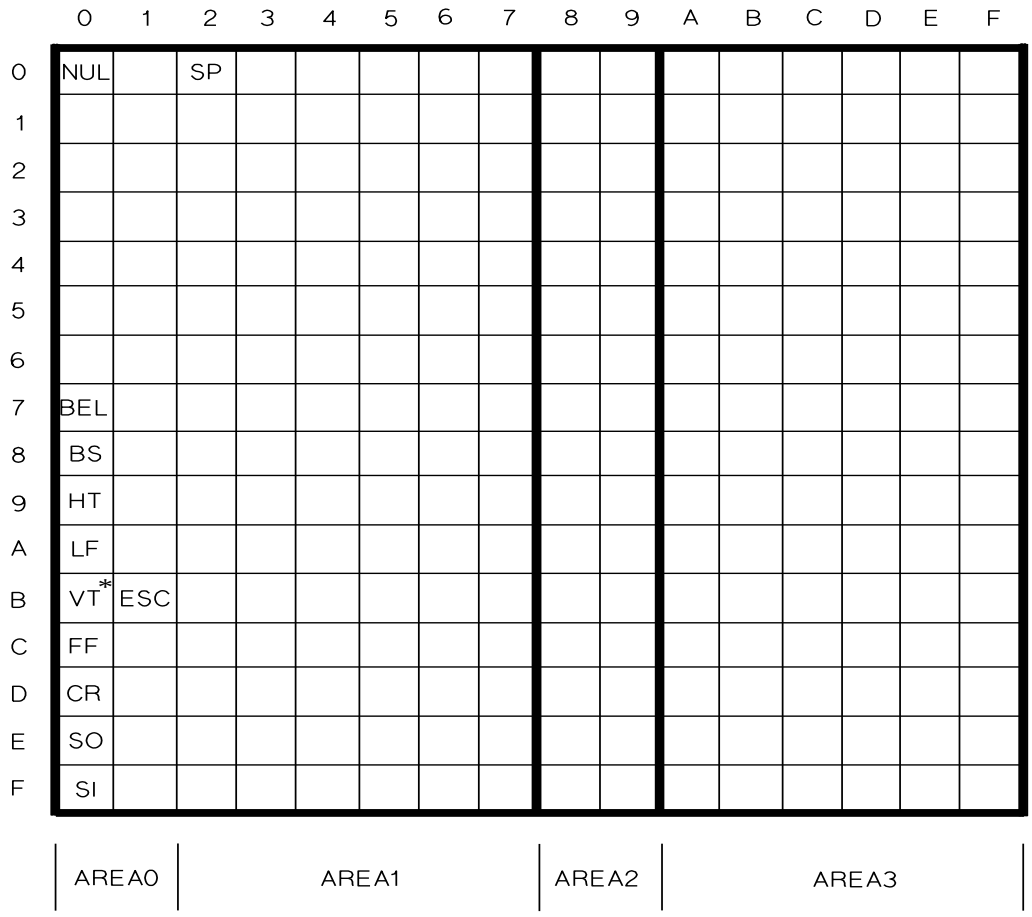
Character Processing

The simplified flowchart below shows how an incoming character code is handled.



Generic Character Code Chart

Figure 5.2 shows a generic character code chart. The areas, special codes, and control codes used in PCL are marked.



* Vertical Tab (VT) is obsolete.

5.2 Control Codes and Special Codes

Control Codes are character codes that execute a unique device function (e.g., formfeed); however, some control codes may cause a non-operation. All the symbol set types defined for PCL devices contain the ten control codes and the SPACE character described below.

Control codes may be associated with graphic symbols that are printable in Transparent Data Transfer or Display Functions Mode. A graphic symbol downloaded to a control code is a *hidden* graphic and is only printable by Transparent Data Transfer (*Esc&p#X*) or Display Functions Mode (*EscY*).

Backspace <*BS*>

Moves CAP one character position backward on the current line.

In horizontal text path mode (*Esc&c#T*), no action occurs if CAP is already at the left margin. If CAP is to the left of the left margin (via a *Move CAP* command), BS functions as if the left margin is column 0, the logical page left boundary.

In vertical text path mode (*Esc&c#T*), no action occurs if CAP is already at the top margin. If CAP is above the top margin (via a Move CAP command), BS functions as if the top margin is row 0, the logical page top boundary.

In proportional spacing, a single BS centers the overstriking character with the character being overstruck. After printing the overstrike character, CAP is at the same position as before the BS. Multiple backspaces each move back the distance of the last printable character or space.

In HP-GL/2 mode, a backspace as the first character of a label is ignored.

Bell <BEL>

Causes a bell or audible alarm to sound if one is present; otherwise causes no action.

Carriage Return <*CR*>

In horizontal text path mode (*Esc&c#T*), CR moves CAP to the left margin on the current line. In vertical text path mode, CR moves CAP to the top margin on the current line. If CAP is left of the left margin, CR moves CAP to the right until it is coincident with the left margin.

In HP-GL/2 mode, the pen location is updated to the carriage return point, usually the pen location when the LB command was executed, adjusted by any line feeds.

Escape <ESC>

Provides supplementary control of printer functions. The escape character itself is a prefix for the string of one or more characters which follow. Once an escape character is received (unless it appears within binary data), normal text processing is suspended until the supplemental function has been activated or the escape sequence has been determined to be invalid.

Formfeed <*FF*>

In horizontal text path mode (*Esc&c#T*), FF moves CAP to the same horizontal position at the top of form on the next page. ***Top of form*** = top margin + (3/4 × line spacing). In vertical text path mode (*Esc&c#T*), FF moves CAP to the same vertical position at [right margin - (3/4 × line spacing)].

Multiple formfeeds in sequence are not interpreted as a single formfeed.

An *implicit* formfeed is performed if printable data has been sent before *EscE*, a LF that moves CAP into the perforation skip area (skip is enabled) or onto the next page, by page size or orientation changes, or by any other command that ejects a "dirty" page.

Horizontal Tab <*HT*>

HT moves CAP to the next tab stop on the current line (in vertical text path mode, the current line extends vertically from top to bottom). Tabs represent a logical position and thus refer to different physical positions for different settings of CMI.

CMI determines current column width. If CMI is changed, the physical location of each tab stop moves. HT has no affect if the CMI is 0.

In horizontal text path mode (*Esc&c#T*), the left margin is the first tab stop; additional tab stops are fixed at every 8 columns to the right margin. In vertical text path mode, the first tab stop is at the top margin; additional tab stops are fixed at increments of (8 * current CMI) to the bottom margin.

The following are some boundary cases for horizontal text path mode:

- If the requested tab stop is to the right of the right margin and CAP is at or to the left of the right margin, HT moves CAP to the right margin.
- If the requested tab stop is to the right of the printable area and CAP is to the right of the right margin, HT moves CAP to the right edge of the printable area.
- If CAP is to the left of the left margin, HT moves CAP to the left margin.

The following are some boundary cases for vertical text path mode:

- If the requested tab stop is outside the bottom margin and CAP is at or above the bottom margin, HT moves CAP to the bottom margin.
- If the requested tab stop is outside the printable area and CAP is below the bottom margin, HT moves CAP to the edge of the printable area.
- If CAP is above the top margin, HT moves CAP to the top margin.

Tabs do not cause lines to be wrapped if end-of-line wrap mode is enabled.

NOTE: The current unit of measure setting (*Esc&u#D*) affects HT on devices implementing the Unit of Measure command (*Esc&u#D*). For example, if the unit of measure is 300 (one PCL Unit = 1/300 inch), the default CMI is rounded to the nearest 1/300 inch. Rounding always occurs, whether CMI has been implicitly set by font selection or explicitly set by the CMI command (*Esc&k#H*).

DEVICE NOTE: Although DJ850C implements the Unit of Measure command, HT is not affected by the unit of measure setting.

Linefeed <*LF*>

In horizontal text path mode (*Esc&c#T*) , LF moves CAP to the same horizontal position one row down. If perforation skip mode is enabled, a LF that would go beyond the text length boundary moves CAP to the same horizontal position at the top of form on the next page. If perforation skip mode is disabled, text is printed to the end of the page and onto the top edge of the next page. Text in the unprintable region may be lost.

DEVICE NOTE: If perforation skip is on and CAP is within the top margin area, a LF moves CAP one line down on LJs and DJs above 1000, and to the top of form on DJs below 1200.

In vertical text path mode (*Esc&c#T*), LF moves CAP to the same vertical position one column to the left. If perforation skip mode is enabled, a LF that would cause CAP to go beyond the left margin moves CAP to the same vertical position at the right margin of the next page; that is, a linefeed causes a perforation skip. If perforation skip mode is disabled, text is printed to the left margin and onto the right edge of the next page. Text in the unprintable region may be lost.

DEVICE NOTE: If perforation skip is on and CAP is within the right margin area, LF moves CAP one column to the left on LJs and DJ6xx, and to the right margin on DJs below 1200.

In HP-GL/2 mode, the pen position is advanced one line. For HP-GL/2 labels, a line is defined to be the height of the character cell.

Null <NUL>

Causes no action in a PCL printer.

Shift Out <SO>

Invokes the currently designated secondary font as the active font. Subsequent characters are printed from the secondary font, which remains active until a Shift In or Reset (*EscE*). The default is **not** Shift Out. (See Chapter 9)

Shift In <SI>

Invokes the currently designated primary font as the active font. Subsequent characters are printed from the primary font, which remains active until a Shift Out. The default is Shift In. (Chapter 9)

Space <*SP*>

Moves CAP forward one character position (defined by the CMI) on the current line.

In horizontal text path mode (*Esc&c#T*), CAP does not move if it is already at the right margin and end-of-line wrap is not enabled. If end-of-line wrap is enabled, CAP moves to the left margin of the next line and then prints the space. Trailing spaces are also printed.

In vertical text path mode, CAP does not move if it is already at the bottom margin and end-of-line wrap is not enabled. If end-of-line wrap is enabled, CAP moves to the top margin of the next line and then prints the space. Trailing spaces are also printed.

Space may be a either printable character or a control code. If a glyph is defined for the space code, space is printable; otherwise, it is a control code. For proportionally-spaced fonts, a space control code updates CAP by the current CMI value; however, a printable space moves CAP the width of the character. For fixed-pitch fonts, a space, whether control code or printable, updates CAP according to the CMI value.

A printable space fixes CAP and makes the page "dirty", subject to conditional page ejects. An unprintable space does not make the page dirty.

DEVICE NOTE: On DJ6xx and 8xx all spaces, printable and unprintable, make the page dirty.

The space character may be redefined in a downloadable font. When space is redefined in this way, the font's default CMI changes to match the new width of space, not the current CMI.

DEVICE NOTE: DeskJets below 600 do not allow redefinition of SPACE.

In HP-GL/2 mode, the pen position is updated one column to the right of the current position. The space width may be modified by the ES command.

Line Termination *Esc & k # g/G*

Controls how CR, LF, and FF are interpreted.

Value(#)	=	0	CR = CR;	LF = LF;	FF = FF
	=	1	CR = CR,LF;	LF = LF;	FF = FF
	=	2	CR = CR;	LF = CR,LF;	FF = CR,FF
	=	3	CR = CR,LF;	LF = CR,LF;	FF = CR,FF
Default	=	0			
Range	=	0 to 3 (out-of-range values are ignored)			

For example, a value field of 1 causes the printer to insert a carriage return (CR) and linefeed (LF) control code for every CR. A linefeed or formfeed is sent as is.

In Display Functions Mode (*EscY*), LF and FF are not executed, but a glyph is printed if it exists for that character code, or a space if it does not. Display Functions handles a CR by printing the glyph or space and then executing a CR-LF.

5.3 Symbol Sets

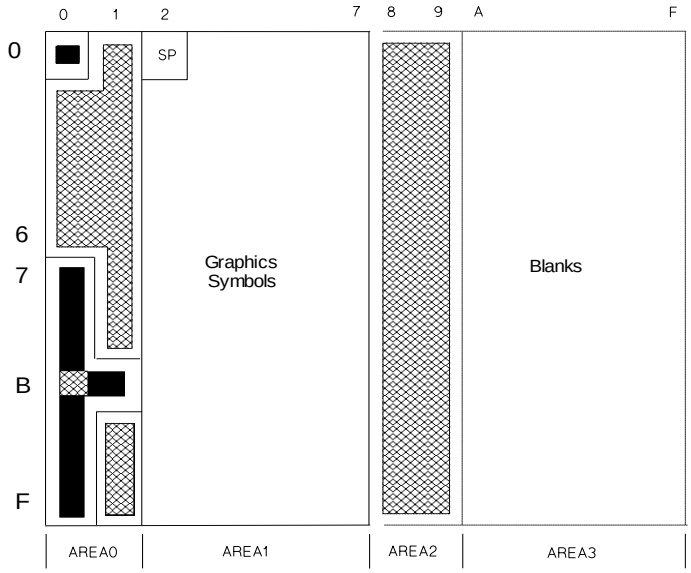
A symbol set is a mapping from character codes to glyphs (graphic symbols). PCL defines four types of symbol sets: HP-7 and HP-8, modeled after the ISO-646 and ISO-8859 definitions for character coding, PC-8 for IBM PC compatibility, and *large* (2-byte) to provide more characters for Asian fonts.

In the diagrams below:

Hatch	Indicates control codes no longer implemented
Solid	Indicates control codes that are still implemented.

HP-7 Compatible (96 Glyph)

The HP-7 type is a unique subset of the HP-8 type, except that characters are defined only in **area0** and **area1** (0 through 127). This type is recommended for new designs.



NOTE: A ***printable*** graphics symbol is one that prints a mark on the paper, or advances CAP.

Area0 contains nonprinting/nonspacing characters that are ignored or processed as control codes. In Transparent Data Transfer a graphic symbol is printed and CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

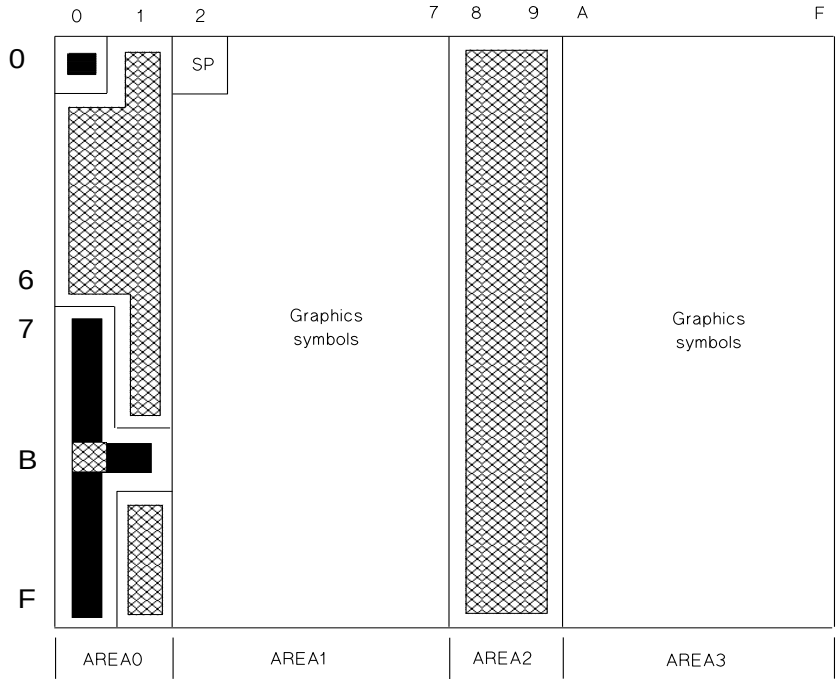
Area1 characters are printable graphic symbols that mark the page and advance CAP. **Area2** characters are nonprinting/nonspacing: no marks are printed and CAP is not advanced. **Area3** characters advance CAP but print no mark. In Display Functions Mode or Transparent Data Transfer, character codes in **area2** and **area3** advance CAP by the CMI distance.

Area0 and **area1 c**haracters may be downloaded with graphic symbols. Attempts to download an **area2** or **area3** character are ignored.

DEVICE NOTE: DeskJets below 600 except the 540 treat a character code in area2 as if it were in area0. A character code in area3 is treated as if like a graphic character in area1.

HP-8 Compatible (192 Glyph)

The HP-8 type is based on ISO-8859 coding for 8-bit symbol sets. As shown below, **area0** and **area2** are treated as nonprinting/nonspacing characters or control codes; and **area1** and **area3** are printing characters.



Area0 contains nonprinting/nonspacing characters that are ignored or processed as control codes. In Transparent Data Transfer a graphic symbol is printed if the glyph exists and a space if it does not; then CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

Area1 and **area3** characters are printable graphic symbols that advance CAP.

Area2 characters are nonprinting/nonspacing. However, in Transparent Data Transfer a graphic symbol is printed and CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

The character at position A0h is treated like any other graphic symbol. Many symbol sets define it as a blank with a fixed width set by the current font; it is then not processed like the space character.

All character codes may be downloaded with graphic symbols. Symbols downloaded to codes located in **area0** and **area2** are hidden graphics and are only printable in Transparent Data Transfer or Display Functions Mode.

PC-8 Compatible (256 Glyph)

The PC-8 type is used for compatibility with IBM PCs. As shown below, **area0**, **1**, **2**, and **3** have no meaning; only control codes 00, 07-0F, and 1B are nonprinting.

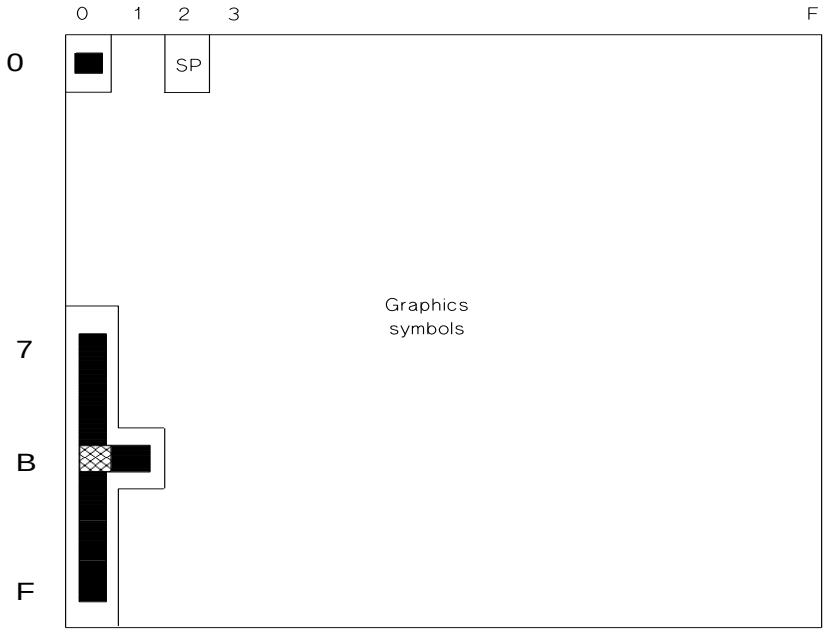


Figure 5-5 PC-8 Compatible Set

Characters defined as control codes are nonprinting/nonspacing and are processed as control codes. In Transparent Data Transfer a graphic symbol is printed and CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

A character not defined as a control code is a printable glyph that advances CAP.

The character at position F/F is treated like any other graphic symbol. Many symbol sets define it to be a blank, but this blank is of a fixed width set by the current font and is not processed like the space character. This is allowed by handling this character as a graphic and not a space.

Symbol sets Offered by HP

Many of the symbol sets offered by HP are based on standard codings provided by organizations like ISO (International Standards Organization) and ANSI (American National Standards Institute). Nonstandard sets have been implemented in unique situations, or when an application has required special characters. See Chapter 9, Font Selection, for a complete list of the symbol sets and PCL identifiers that HP has implemented or reserved for future products.

5.4 Printing Hidden Characters

The following options allow the printing of character codes that normally have hidden graphic symbols. Transparent Data Transfer (*Esc&p#X*) disables control code functions and instead prints graphic symbols for a specified number of bytes. Display Functions Mode disables and prints all escape sequences and control codes (except CR and *EscZ*).

Transparent Data Transfer *Esc & p # x/X*

Prints the graphic symbols associated with hidden character control codes.

Value(#) = Number of binary bytes to be printed ("binary" means not parsed)
Default = NA
Range = 0 to 2^{32}-1

All subsequent character codes for the specified number of bytes are printed with the current font attributes. The parser ignores all control codes, including the *Esc* character; instead, the code's graphic symbol in the current symbol set is printed. For example, in the PC-8 symbol set, *Esc* is printed as a left arrow.

NOTE: Text Parsing Method (*Esc&t#P*) determines how transparent data may be interpreted.

Display Functions Mode ON *Esc Y*

This command prints a character code in the currently active font. Turning this mode ON has the following effects:

- All control code and escape sequence functions except *CR* and *EscZ* are disabled. *CR* marks the paper and executes CR-LF. *EscZ* marks the paper and disables Display Functions Mode.
- All character codes either mark the paper or produce a blank space.

This mode is intended as a programmer's debugging aid, and is not to be used for document preparation.

NOTE: Text Parsing Method (*Esc&t#P*) determines how character codes may be interpreted.

Display Functions Mode OFF *Esc Z*

This command turns off display functions mode. If display functions mode is ON when *EscZ* is received, the characters for the sequence are printed, and the mode is disabled. If display functions mode is OFF when *EscZ* is received, no operation is performed. The default is OFF.

5.5 Text Enhancements

Enable Underline *Esc & d # D*

Enables the automatic text underline enhancement.

Value(#)	=	0	Default single underline
	=	1	Single underline, fixed location below the baseline
	=	2	Double underline, fixed location below the baseline
	=	3	Single underline, font dependent location (floating)
	=	4	Double underline, font dependent location (floating)
Default	=	0	
Range	=	0 to 4 (the default is selected for out-of-range values)	

NOTE: This command must use a capital "D" as a terminator.

When underline is enabled, any positive horizontal motion, including spaces and CAP moves, is underlined (except a CR when CAP is to the left of the left margin). Underline remains enabled until explicitly disabled. The default state is underline disabled.

A single underline is produced if double underline is invoked but unavailable.

The only mode that must be implemented is 0.

In *fixed-position* underlining, the underline is drawn a fixed distance below the baseline. The distance is device-dependent, but font independent.

In *floating-position* underlining, the greatest underline distance specified in the font descriptors of all the fonts printed on the current line determines the underline position.

This command is ignored when the Text Path Direction (*Esc&c#T*) is set to vertical printing (1).

DEVICE NOTE: The floating underline position on DeskJets below 1200 is font-dependent: each font uses a different underline, and the greatest underline distance is ignored (i.e., LaserJets). DeskJets below 600 except the 540 implement a value of 3 or 4 and ignore out-of-range values.

Disable Underline *Esc & d @*

Disables automatic text underlining.

End-of-Line Wrap *Esc & s # c/C*

Defines the action that occurs when a line of text reaches the right margin in horizontal text path mode (*Esc&c#T*), or the bottom margin in vertical text path mode.

Value(#)	=	0	Enables End-of-Line Wrap
	=	1	Disables End-of-Line Wrap
Default	=	1	
Range	=	0,1	

When end-of-line wrap is enabled in horizontal text path mode (*Esc&c#T*), a character or space that would move CAP to the right of the right margin causes a CR-LF to be executed (prior to the printing of the character or space). In vertical text path mode, a character or space moving CAP below the bottom margin causes a CR-LF to be executed (prior to the printing of the character or space).

When end-of-line wrap is disabled in horizontal text path mode, a character or space that would move CAP to the right of the right margin is clipped (i.e., the entire glyph is not printed) and CAP is set to the right margin. In the vertical text path mode, a character or space that would move CAP below the bottom margin is not printed and CAP is set to the bottom margin.

If margins are set so that a given character cannot fit on a line, the character is clipped even if end-of-line wrap is enabled.

The character's escapement (delta X) determines whether or not the glyph fits.

DEVICE NOTE: DJs above 1000 use black width, not delta X, to determine whether to line wrap. DJs below 1000 determine whether to wrap by a character's cell width in fixed and multi-pitched fonts, and by escapement (delta X) in proportionally-spaced fonts.

5.6 Escapement Encapsulated Text

The Escapement Encapsulated Text (EET) command corrects for the following two situations:

- The printer's linear TrueType scaler produces different escapements than the non-linear TrueType scaler used by Windows drivers.
- Unlike printer-based escapements, Windows applications justify text by changing both inter-word and inter-character spacing, causing a different glyph placement.

Horizontal CAP moves (*Esc&a#C*, *Esc&a#H*, *Esc*p#X*) can correct escapement (absolute CAP moves are needed because the driver does not know where the printer had updated CAP); however, then the driver must convert the escapement to an ASCII string and the printer's escape sequence parser must convert the string to a binary number. It is more efficient, using fewer bytes, to send the escapement as binary and not in the escape sequence as ASCII.

Low-end printer memory costs suggest intermixing of characters and escapements, rather than grouping escapements and characters separately. (Grouping was originally proposed because GDI sends the character to the driver before the escapement to accommodate devices that render the characters as soon as they are received.)

The signed nature of the escapement allows for a negative or positive escapement. Text Path Direction (*Esc&c#T*) determines which direction is positive or negative.

The EET command supports "printable" characters by treating all characters like characters in Transparent Data Mode (*Esc&p#X*) — no control code functions will be executed.

Escapement Encapsulated Text *Esc & p # w/W*

Transfers a text string with an encapsulated escapement value for each text character. The escapement contained in font data is overridden.

Value(#) = Number of data bytes
Default = NA
Range = 0 to 2^{32}-1

DEVICE NOTE: DJ850C supports a range of 4 to 32767.

The data field must contain byte-aligned binary data, not ASCII. Invalid configurations are ignored and the specified number of data bytes discarded. Value field signs are ignored. Extra bytes are discarded.

This command handles the character string in a single line. The End-of-Line Wrap (*Esc&s#C*) state has no affect on this command.

Character placement at the right margin follows existing text rules: the character is rendered if either of the following is true; otherwise the entire character is clipped and CAP is unchanged.

- Character origin and escapement are to the left of the right margin.
- Character origin is to the left of the right margin; escapement is to the left of the right edge of the logical page.

If subsequent escapements are negative and characters again follow either of the above two rules, then characters are rendered. If the string starts to the right of the right margin, then character placement at the right edge of the logical page follows existing text rules.

DEVICE NOTE: DJ850C ignores Right Margin (*Esc&a#M*). Character placement at the right edge of the logical page follows text rules.

Unit of Measure (*Esc&u#D*) sets escapement unit size. Escapement units that do not add up to an integral number of dots are converted to a higher internal resolution and the error is collected. When the fractional portion of a dot exceeds 0.5 dots, the next dot position is rounded up.

The current underline state of Enable Underline (*Esc&d#D*) has no effect on this command because encapsulated text is not underlined.

Data field format is as follows:

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Format # = 0 (UBYTE)	Text Character 1 (UBYTE)	1
2	Escapement Value for Text Character 1 (SINT16)		3
4	Text Character 2 (UBYTE)	Escapement Value for Text Character 2 MSB (SINT16)	5
6	Escapement Value for Text Character 2 LSB (SINT16)		7
	...		
3n-5	Text Character *n*-1 (UBYTE)	Escapement Value for Text Character *n*-1 MSB (SINT16)	3n-4
3n-3	Escapement Value for Text Character n-1 LSB (SINT16)	Text Character *n* (UBYTE)	3n-2
3n-1	Escapement Value for Text Character *n* (SINT16)		3n

Byte 0: Format Number

Value = 0 CAP relative escapements

A value of 0 configures for encapsulated 8-bit text characters with escapements relative to the previous CAP. The command is ignored for other values and the specified number of data bytes is discarded.

Byte 1: Text Character 1

Value = 0 to 255

Eight-bit value of the text character 1.

Bytes 2&3: Escapement Value for Text Character 1

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character 1. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

Byte 4: Text Character 2

Value = 0 to 255

Eight-bit value of the text character 2.

Bytes 5&6: Escapement Value for Text Character 2

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character 2. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

Byte 3*n*-5: Text Character *n*-1

Value = 0 to 255

Eight-bit value of the text character *n*-1.

Bytes 3*n*-4 & 3n-3: Escapement Value for Text Character *n*-1

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character n-1. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

Byte 3*n*-2: Text Character *n*

Value = 0 to 255

Eight-bit value of the text character *n*.

Bytes 3*n*-1 & 3*n*: Escapement Value for Text Character *n*

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character *n*. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

5.7 Text Parsing Method

The PCL parser must know whether one-byte or n-byte character codes are being sent. Text Parsing Method (*Esc&t#P*) lets PCL parse multiple-byte characters regardless of the large symbol set (e.g., Unicode, JIS, Shift-JIS) in use.

Text Parsing Method *Esc & t # p/P*

Specifies how the PCL data stream is to be parsed.

Value(#)	=	0,1	Character codes are processed as 1-byte characters.
	=	2	Strict 2-byte processing. Characters and control codes are represented by two bytes. The Unicode standard is an example.
	=	21	Printable characters are 2 bytes; control codes are 1 byte. Control code range is 0 to 0x20. A first byte between 0x21 to 0xFF is considered to be the upper byte of a 2-byte character.
	=	31	Shift JIS parsing. 1-byte control code range is 0-0x20; 1-byte character range is 0x21 to 0x80, 0xA0 to 0xDF, and 0xFD to 0xFF. 2-byte characters have a first byte between 0x81 to 0x9F or 0xE0 to 0xFC.
	=	38	If the 8th bit is set, the byte is the first byte of a 2-byte character code. If the 8th bit is not set, the byte represents a 1-byte character code.
Default	=	31 if the default symbol set is WIN31J; 38 if the default symbol set is GB2312-80; otherwise it is 0.	
Range	=	0,1,2,21,31,38 (other values map to 0)	

The default depends on the default PCL symbol set.

In strict 2-byte parsing (value=2), the parser cannot distinguish between an <ESC> in the data stream and the upper byte of a 2-byte character. Therefore, in strict two-byte processing, the host must send a <NULL> before each escape sequence. To start an escape sequence, 0x001B is sent and then the rest of the escape sequence.

If the value is 21, character codes between 0x21 to 0xFF are processed as the first byte of a two-byte character, and the next byte is processed as the second byte. Character codes outside this range are processed as one-byte values. This method can be used for parsing characters in Asian seven-bit encoding specifications, i.e., JIS X0208 (Japan).

If the value is 31, character codes between 0x81 to 0x9F and 0xE0 to 0xFC are processed as the first byte of a two-byte character, and the next byte is processed as the second byte. All character codes outside this range are processed as one-byte values. This method can be used for parsing characters in the Shift-JIS encoding specification.

If the value field is 38, character codes between 0x80-0xFF are processed as the first byte of a two-byte character, and the next byte is processed as the second byte. All character codes outside this range are processed as one-byte values. This method can be used for parsing Asian eight-bit encoding specifications, such as the Big Five and TCA encoding specifications (Taiwan) and KS C 5601-1992 (Korea), and GB 2312-80 (China).

When printing characters, the font from which the character is printed may be incompatible with the text processing method. For example, the user may have selected strict two-byte parsing, but the current font is an 8-bit font. When this happens, the requested character code is compared against the range of the currently selected font. If the character code is within range, the character is printed; otherwise, the character is considered to be out of range and no action is performed.

The Transparent Data command (*Esc&p#X*) prints character codes that are normally not printed. Since the value field specifies the number of bytes to be interpreted, the Transparent Data command is extended to interpret *n*-byte character codes. The selected text parsing method determines how the transparent print data should be interpreted. If, when reading the last byte of transparent data, another byte of data is required to complete the character, but there is none, the last byte is discarded.

EscE defaults text parsing method, which is part of the modified print environment. Therefore, text parsing method is saved by a macro call or overlay, but not by a macro execution. Text parsing method is defaulted at the beginning of an overlay macro and restored at completion.