CZL

Programmer Manual

rel. 1.00

Sferal wwt srl
Via Martiri d'Italia, 26
10014 Caluso (TO)
Italy

1.CONTENTS

2. INTRODUCTION

In order to operate, thermal printers need to receive a series of commands sent to them in order according to precise procedures.
To address all needs, we have developed the CZL language for its own printers, among others.
In general, a CZL command consists of a prefix, a two-character mnemonic code, and a string of parameters. The prefix, the mnemonic code, and the string of parameters are composed of printable characters without any control character (e.g. escape): this guarantees easy loading of the software onto different types of computers.
For the same reason, the print file to be sent to the printer may be a simple text file (composed with any editor); alternatively, the instructions may be sent in the form of a programming language (e.g. Basic), or the label may be laid out on the monitor screen with WYSIWIG[1] or word processing software that, by means of a driver, converts the label into instructions that can be understood by the printer. In this later case, the user can ignore the programming language described by this manual, even if knowing it is useful to a full understanding of the functions and potential of the machine.
This software is extremely powerful and makes it possible for non-expert users to create labels by using just the mouse and the keyboard, while also making it possible to use our printers with any application (database, word processing, and others). On the other hand, if the user wishes to optimize machine performance, the best approach is to program the printer with the CZL language described in this manual.

[1] What You See Is What You Get.

3. COMMANDS

In the CZL language, all commands can be sent both in upper and lower case characters (or partially in upper case and partially in lower case), which produce the same result and are always preceded by a prefix that identifies their type. The character ^ (*94_{10}* or *$5E_{16}$* on the ASCII table) identifies the ^ *commands* and the character ~ (*126_{10}* or *$7E_{16}$* on the ASCII table) the ~ *commands* (see fig. 1). Some commands support both the prefix ^ and ~[2].

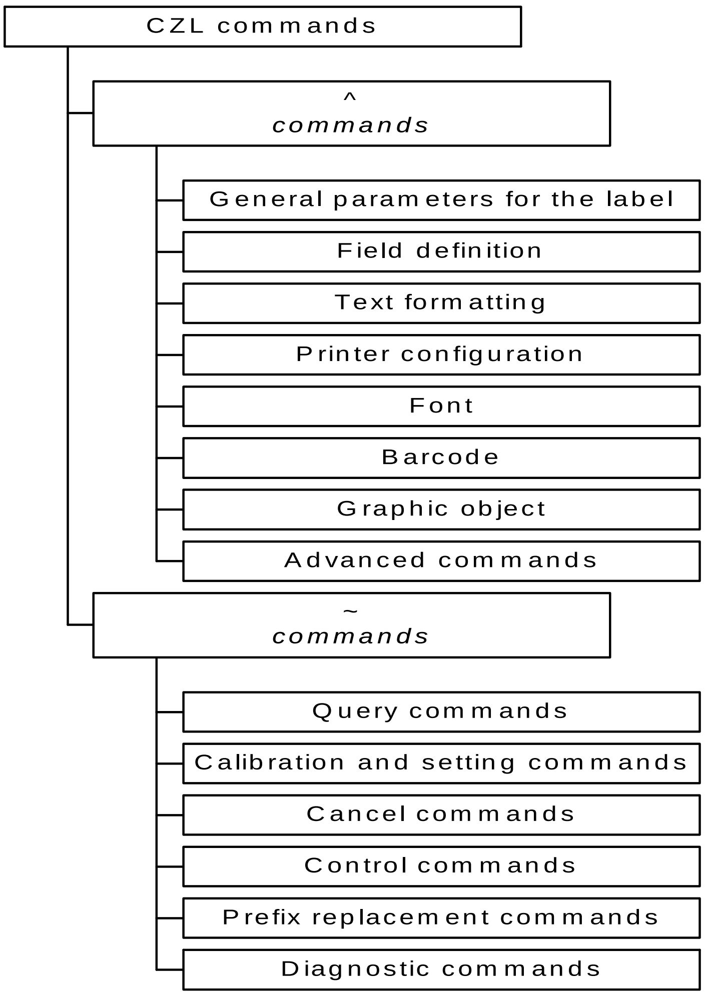

fig. 1 – Classification of commands

[2] Both prefixes can be replaced with the commands *^CC* or *~CC* (Change Caret) and *^CT* or *~CT* (Change Tilde), described on page 46.

The parameters are separated from each other and their respective commands by a separator character, the comma (*44_{10}* or *$2C_{16}$* on the ASCII table).

Please refer to section 5 on page 50, where several examples illustrate the functions of the commands.

3.1 ^ Commands

3.1.1 General parameters for the label

3.1.1.1 Start and end format commands

^XA (Start Format)

When it receives this command, the printer prepares to receive the ^ commands that will be performed in the order in which they have been received. In other words, all ^ commands are preceded by the command *^XA* and followed by the subsequent command *^XZ*.

^XZ (End Format)

This is normally the last in a series of commands sent to the printer. It follows all ^ commands, which can thus be inserted between the command *^XA* and *^XZ*. After this command, the printer only interprets ~ commands or another *^XA* command.

3.1.1.2 Positioning commands

^LHx,y (Label Home)

This command makes it possible to position the origin of the coordinates, or move the entire bitmap away from zero (the upper left corner) (see fig. 2).

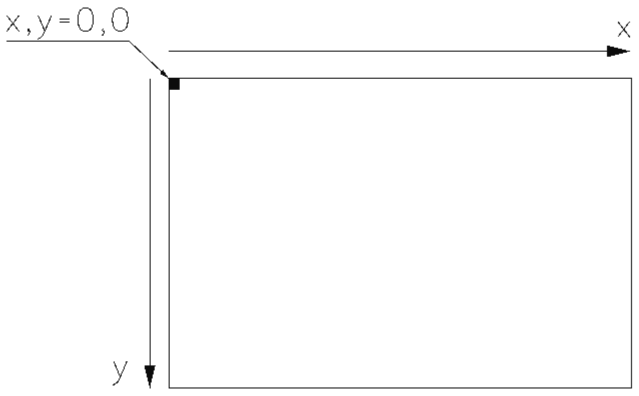

fig. 2 – Origin of the coordinates

x is a whole number from *0*[3] to *9999*[4] and expresses the horizontal offset expressed in dots[5].
y is a whole number from *0*[6] to *9999*[7] and expresses the vertical offset expressed in dots.
After receiving this command (which must be sent before the first command *^FS* as described on page 13) the printer keeps these values until it is reset or until it receives another *^LH* command. The command only affects the commands that follow it, so it is normally one of the first to be sent, in order for it to affect the entire label.
It is useful, for example, in centering text on a pre-printed label.

^LLy (Label Length)

In thermal printers the paper is positioned under the head by means of a reference (the gap between labels, the hole, or the black tick mark on the back). If the paper does not contain any reference marker (meaning that it is a *continuous form*) the printer must be informed of the operating procedure (with the *^MN* command on page 15) and length of the printing page (using the label length command).
The command must be sent before the first *^FS* command (page 13) (it does not matter whether it comes before or after the *^MN* command) and remains in effect until the machine is reset or receives another *^LL* command.

3 Default value.
4 Although it is possible to insert any whole number up to 9999, the limit is determined by the number of dots on the print head (or its width for the set resolution). A greater number displaces the image outside the useful printing area. See the printer User Manual.
5 If the printer resolution (see the User Manual) is 203 dpi, one dot corresponds to 1/8 mm, if the resolution is 300 dpi, to 1/12 mm.
6 Default value.
7 Although it is possible to insert any whole number up to 9999, the limit is determined by the maximum printing length (in turn limited by the available amount of RAM). A larger number displaces the image outside the useful printing area. See the printer User Manual.

y is a whole number from *0*[8] to *9999*[9] and expresses the length of the printing page in *numbers of dots*[10].

^LSx (Label Shift)

In reference to fig. 2 on page 9, this command shifts the entire image of the label to the left by *x* dots.
x is a whole number from *0*[11] to *9999* and expresses the amount of displacement in *numbers of dots*[12].
This command too must be sent before the first *^FS* command (page 13) and remains in effect until the machine is reset or receives the next *^LS* command.
In practice, the amount (in dots) the label is moved from the origin of the coordinates, illustrated in fig. 2 on page 9, is determined by the following formula:
^LHx + ^FOx - ^LSx[13]
If the result of this formula is a negative value, it is interpreted as equal to *0* because it is not possible to print farther to the left than the first useful dot on the print head.

3.1.1.3 Rotation commands

^FWa (Field Orientation)

This permits clockwise rotation by 0°, 90°, 180°, or 270° (according to the value assigned to parameter *a*) of all (and only) those fields which can be assigned a rotation parameter. In other words, the rotation defined by this command is the default setting if another one is not specified.
The parameter *a* may assume the following values:

N[14]: *Normal* (0°) orientation
R: *Rotated* (90°) orientation
I: *Inverted* (180°) orientation
B: *Bottom Up* (270°) orientation

The command affects only those fields which can be assigned a rotation parameter when it is omitted. When the rotation parameter can be assigned and is used, it overrides the *^FW* command.
Like the others, the *^FW* command affects only the commands that follow it until the printer is reset or the next *^FW* command is sent.

[8] Default value.
[9] See note 7 on page 9.
[10] See note 5 on page 9.
[11] Default value.
[12] See note 5 on page 9.
[13] See pages 8 and 11 respectively for a description of the commands *^LH* and *^FO*.
[14] Default value.

3.1.2 Field definition

These commands make it possible to define the type and position of the fields (e.g. alphanumeric strings) that comprise the label.

^FOx,y (Field Origin)

This command determines the position of the upper left corner of the field in relation to the zero coordinate set by the command *^LH* (page 8) and is independent of the rotation. In other words, whatever the rotation, the upper left corner of the rectangle surrounding the field is set to the coordinate *x*, *y* (see fig. 3). When the field is horizontal (whether its rotation is *normal* or *inverted*), it is always in the same position as when it is vertical (whether its rotation is *rotated* or *bottom up*).

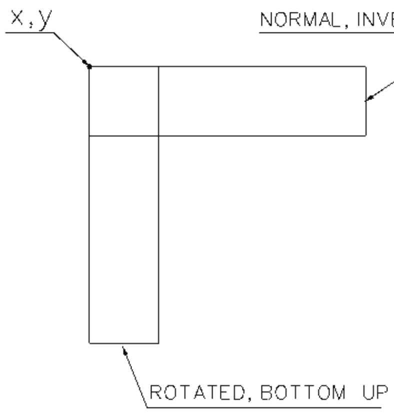

fig. 3 – Rotation of a field positioned with the ^FO command

x is a whole number from *0*[15] to *9999*[16] and is the horizontal position along the *x* axis (see fig. 2 on page 9) in the upper left corner of the field, expressed in dots[17].
y is a whole number from *0*[18] to *9999*[19] and is the vertical position along the *y* axis in the upper left corner of the field, expressed in dots.

^FTx,y (Field Typeset)

This command, like the previous one, sets the position of the origin of the field (in relation to the zero coordinate set by the *^LH* command described on page 8) consistently with the content of the

[15] Default value.
[16] See note 4 on page 9.
[17] See note 5 on page 9.
[18] Default value.
[19] See note 7 on page 9.

field and does not change the rotation. In practice, the field rotates around its origin (generally the lower left corner), as appears in fig. 4.

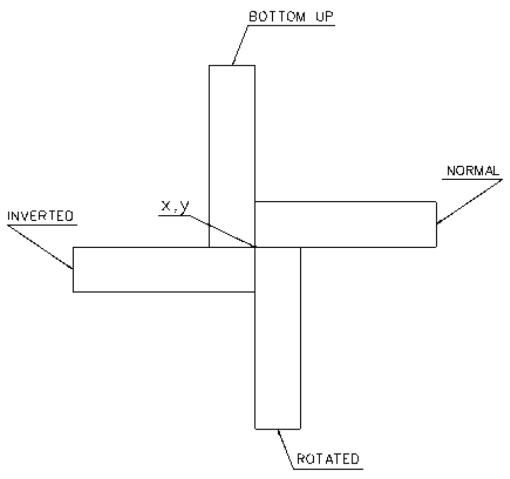

fig. 4 – Rotation of a field positioned with the ^FT command

x is a whole number from *0*[20] to *9999*[21] and is the horizontal position along the *x* axis (see fig. 2 on page 9), expressed in dots[22].

y is a whole number from *0*[23] to *9999*[24] and is the vertical position along the *y* axis, expressed in dots.

The origin of the field, whose coordinates are set by the parameters *x* and *y*, varies according to its type.

In *text* fields, the origin is located at the intersection of the line on which the characters are aligned and the vertical line with which the first character is aligned (in the case of horizontal writing, this is the lower left corner).

If the coordinates are omitted (even just one of these), the coordinates of the last field entered are used as if the label image were virtually shifted with a cursor. This is how the fields can be more easily positioned with respect to one of them: by changing the coordinates of the reference field, all the other fields are moved while preserving their relative position with respect to the first field.

In the *bar-code* fields, the origin is the base of the bar-code, even when there is a *human-readable* field beneath them or when the bar-code provides for a line above and below it.

The origin of the *graphic box* fields is the lower left corner.

The origin of the images is the lower left corner of the rectangle in which the graphic image is inserted.

[20] Default value.
[21] See note 4 on page 9.
[22] See note 5 on page 9.
[23] Default value.
[24] See note 7 on page 9.

^FDaa…a (Field Data)

This command makes it possible to define the string to be entered in the field.

aa…a is an alphanumeric string (up to 3072 characters long) of any printable character, with the exception of the two prefix characters (e.g. ^ and ~). The characters ^ and ~ can be printed if the prefix character is changed with the commands *^CC* (*~CC*) and *^CT* (*~CT*) or with the command *^FH* on page 13.

This last command, *^FH*, is useful for printing all characters after the 127th on the ASCII table (table 9 and table 10 on pages 59 and 60) and, in general, all characters that are not directly available on the keyboard.

Even the characters *<CR>* (13_{10}, $0D_{16}$) and *<LF>* (10_{10}, $0A_{16}$) can be entered in the field.

^FS (Field Separator)

This command is the separator for each field; in other words, it sets the end of a field and the beginning of the next one. Therefore, it is obligatory, and every command for rotation, positioning, etc. in the field takes effect if it is followed by the *^FS* command.

^FHi (Field Hex)

This command makes it possible to print any character through its hexadecimal code. In particular, it is possible to insert the hexadecimal code of the character to be printed directly in the string *aa…a* of the *^FD* command as long as it is preceded by the character *i* defined by the Field Hex command.

Therefore, the *^FH* command must precede every *^FD* command (or any other command where a data field is specified).

i can be any printable character as long as its meaning is unambiguous: in the following *aa…a* string, every appearance of the *i* character signifies “hexadecimal code identifier.” Therefore, it makes sense to use a character that is not printed: The default is the *underline* character _ (95_{10}, $5F_{16}$).

After the *i* identifier, the machine interprets the two subsequent characters as the “hexadecimal code of the character to be printed,” and thus the two following characters can be any two of the following:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, A, a, B, b, C, c, D, d, E, e, F, f.

3.1.3 Text formatting

^FBl,n,s,g,m (Field Block)

This command makes it possible to print different lines of text enclosed in a virtual box that assumes the properties (origin, font, rotation) of the text field. This makes it possible to move, enlarge, and otherwise manipulate the entire box without modifying each string of text. This command incorporates an automatic word wrapping function[25] to make operations easier without worrying about the placement of individual strings.

l is the width of the box expressed in dots[26]. It can be any whole number between *0*[27] and *9999*. However, it must not be smaller than the width of a character or larger than the width of the printing area (see note 4 on page 9). If the width of the block is not sufficient to contain at least one character, nothing is printed; if it is not large enough to contain everything, only the text contained in the block is printed[28].

n is the maximum number of lines that the box can contain. It can be any whole number between *1*[29] and *9999*. In practice, this parameter determines the height of the box (which is thus expressed in *number of lines*). Thus, by changing the font type, the dimensions of the box are automatically adjusted. If the number of lines is not sufficient to contain all the lines of text that are to be printed, the last of these is overwritten (thus becoming illegible) until all characters to be printed are consumed.

s è is a whole number between *–9999* and *+9999*[30] and indicates the space (in dots[31]) to add (if the number is positive) or subtract (if the number is negative) between one line and another. The numbers without sign are read as positive.

g is a letter that can assume the following value on the basis of the type of justification desired: *L*[32] for left justification of the text, *C* to center it in the box, *J* for full justification, and *R* for right justification.

m is a whole number between *0*[33] and *9999* indicating the number of dots that the left margin is indented for lines after the first. Its function is thus to indent text.

To render formatting of text inside the box easier, the following sequences of special characters can be used:

\*&* corresponds to a *<CR>* (Carriage Return) + *<LF>* (Line Feed), to force a carriage return.

\\ corresponds to a word break. If the split word is close to the left margin, the printer adds word-break hyphen. Otherwise, the word break is ignored.

\\\\ serves to print the character \\.

If a word is too long to be contained on a single line (and is not broken by any \\ character) a word-break hyphen is automatically added close to the left margin of the block and the rest of the word is printed on the following line. The language does not provide any rule for syllabification of the words: they are split apart according to their length. Therefore, it is necessary to force their break with the \\ command to respect the syllabification rules of the word itself.

These “special” sequences of characters also fall within the limit of 3072 characters indicated for the *^FD*[34] command. That is, each alphanumeric string *aa…a*, including special sequences, cannot be longer than 3072 characters.

---

[25] Just as in word processing, the text is justified (on the left, center, full justification, or on the right) and returns automatically inside the box.

[26] See note 5 on page 9.

[27] Default value.

[28] Therefore, the parameter *l* is mandatory since the default value *0* would be used instead.

[29] Default value.

[30] *0* is the default value.

[31] See note 5 on page 9.

[32] Default value.

[33] Default value.

The "normal" characters (i.e. different from the *\&* sequence mentioned above) *<CR>* (13_{10}, $0D_{16}$) and *<LF>* (10_{10}, $0A_{16}$) are ignored, just like the spaces (32_{10}, 20_{16}) at the end of the line.
When the *^FT* command is used, the field origin is the lower left corner (see fig. 4 on page 12). Therefore, especially when also using the *^FB* command, it is necessary to check whether, upon enlargement of the font size, there could be a change in the dimensions of the box (from bottom to top) that could spill over the top limit for the label. In this case, only the part of the label contained within the printing area is printed, and the excess amount is left out. In fig. 2 on page 9, the origin of the coordinates corresponds to the upper left corner of the label, and thus everything that goes beyond the origin (with, absurdly, negative coordinates) is lost.
When the *^FO* command is used, upon an increase in font size, the box resizes itself from top to bottom (the origin of the coordinates is the upper left corner, as appears in fig. 3 on page 11) and, therefore, the risk consists in losing the portion extending below the lower edge of the label.
More generally, with the CZL emulation, the useful bitmap is determined by the size of the label: never does it happen that one part of the image is printed on one label and the other part on another.
For the *^FB* command to take effect, it must be preceded by the data string and be followed by the *^FS* command.

3.1.4 Printer configuration

Many of the printer configuration parameters (please refer to the User Manual) can also be defined with software. The parameters defined with software override those defined through the machine configuration menu. Or, if one of the configuration parameters is modified with a software command, the current configuration of the machine is modified until it is reset. When the machine is turned on again, the configuration parameters assume the values of the last saved configuration, and those changed with software are lost if they have not been saved.

^MNa (Media Tracking)

Through this command, it is possible to inform the printer of the type of label in use: continuous form (no reference marker) or labels with a reference marker (be it a gap, black tick mark, or hole).
In the two cases where the parameter *a* assumes the following values:

N for continuous form

Y for the reference label.

With the continuous form, it is necessary to define the page length with the *^LL* command already described above (page 9); where the label is separated by a reference marker (of any type), the length of the printing page is determined by the paper sensor set up to detect the gap (or black tick mark or hole).
The instruction is ignored if the parameter is ignored or different from these two.

34 See page 13.

^MTa (Thermal Media)

In general, thermal printers can print on different types of media using ink ribbon (this type of printing is called *thermal transfer*) or directly onto thermal paper (this is called *direct thermal* printing).
The printing mode is defined with a specific machine configuration parameter or with the following command.
In this case, *a* can assume two different values:

T to set the *thermal transfer* mode
D to set the *direct thermal* mode.

The instruction is ignored if the parameter is omitted or is different from these two.

^MDn (Media Darkness)

Depending on the type of medium that is going to be used, the type of ribbon used, and the printing speed, the print quality is determined by the head temperature. There are 30 possible temperature levels (one of which is set from the panel). By means of this command, it is possible to adjust the head temperature (and thus print density) incrementally according to the value set during configuration. Specifically, the parameter *n* is a whole number between *–30* and *+30*[35] with the convention of assuming all numbers without sign as being positive. After executing the current command, the number *n* is added algebraically to the temperature set during configuration; the result of the operation is the new printer temperature. Even when several *^MD* commands are sent, each one of them refers to the value set in the machine configuration. The effect of the commands is not cumulative: it is the one resulting from the last command sent.
If the result of the command *^MD* produces a temperature value over *30* or less than *0*, the temperature is assumed to be *30* and *0* respectively.

^LTn (Label Top)

In order to define the exact position of the label, this command can be used, where *n* is a whole number between *–64* and *+64*. The result is longitudinal displacement of the entire printing image by *n* dots[36] in reference to the origin of the coordinates as indicated in fig. 2 on page 9. Once again, it is assumed that numbers without sign are positive.

^JZa (Reprint After Error)

In general, after resetting following an operating error (e.g. when the paper or ribbon run out), the printer reprints the whole label that was interrupted due to the error.
This command is used to disable this function.
The parameter *a* can assume one of the following two values:

Y[37] reprinting is enabled
N printing is disabled.

[35] The default value is *0*.
[36] See note 5 on page 9.
[37] Default value.

The instruction is ignored if the parameter is ignored or different from the two allowed, and it only affects subsequent labels.

3.1.5 Font

The language offers a selection of fonts that can be chosen as necessary (see section 4.1 on page 47 for their specifications). It is possible to set a default font: in this case, all alphanumeric fields will be in the font selected if not otherwise specified. Or, it is possible to set the type of font for each field.
The following commands permit you to determine the type of font and some of their features.

^CFt,a,l (Change Default Font)

This command makes it possible to chose the machine default font. After this command, all fields are printed with the selected font (as well as their height and width) until the next *^CF* or *^Ax* command.
t is a letter between *A*[38] and *H* or zero (∅) and identifies the font type (see section 4.1 on page 47).
a is a whole number between *0* and *9999*[39] and indicates the height of the character expressed in dots.
l is a whole number between *0* and *9999*[40] and indicates the width of the character expressed in dots.
If one of the two height (*a*) or width (*l*) parameters is omitted, the other is forced proportionately. Thus, if a width double the normal dimension for that font type is specified without indicating the height, the latter is also assumed to be doubled in order to preserve the proportion of the character.
If both parameters *a* and *l* are omitted, the default values or those of the last *^CF* command received are applied.
The fact is that the language does not allow all possible heights for non-scalable fonts (those from *A* to *H*). Approximations of multiples of the whole numbers closest to standard sizes are used for the dimensions set with parameters *a* and *l*. Referring to table 5 on page 47, if the selected height of font *A* is *16*, the language forces the height to *18*, which corresponds to double the standard height (9 x 2).

^Axr,a,l (Alphanumeric Font)

This command lets you select the font field by field.
x is a letter between *A* and *H* and identifies the font type (see section 4.1 on page 47).
r is one of the following four letters and indicates the clockwise rotation of the field:

N normal rotation of 0°[41]
R rotated by 90°
I inverted by 180°

[38] Default value.
[39] The default value is specified, font by font, in table 5 on page 47.
[40] See note 39.
[41] The default value is *N* or generally the last value received with the *^FW* command described on page 10.

B bottom up rotation of 270°
a is a whole number between *0* and *9999*[42] indicating the height of the character, expressed in dots.
l is a whole number between *0* and *9999*[43] indicating the width of the character, expressed in dots.
Once again, the same observations made regarding the height and width specified by the *^CF* command on page 17 apply, to which the reader is referred.

^A∅r,a,l (Scalable font)

Scalable fonts merit a separate discussion.
They can be of any width or height, and their proportion is guaranteed by the algorithm resident in the programming language. Otherwise, the dimensions of the bitmap fonts (those identified by letters *A – H*) are multiples of the standard. This means that there is a limited number of possible sizes (even if the number is high).
Furthermore, the scalable fonts are proportionately spaced[44], which enhances their appearance.
Like the bitmap fonts, the parameter *r* is one of the following four letters and indicates the clockwise rotation of the field:
N normal rotation of 0°[45]
R rotated by 90°
I inverted rotated by 180°
B bottom up rotation of 270°
a is a whole number between *10* and *1500*[46], indicating the character height in dots.
l is a whole number between *10* and *1500*[47], indicating the width of the character in dots.

3.1.6 Bar-code

3.1.6.1 ^Bx (Bar-code)

In order to insert a bar-code, its type and characteristics (e.g. rotation, height, etc.) must be specified, where some these are typical (see section 4.2 on page 48 for a summary of these characteristics). Since not all of these parameters are mandatory, with others being characteristic of each bar-code, all the *^Bx* commands are explained in detail with the specific parameters for each one. As is true for all the other CZL commands, when a parameter is expressly omitted because the default value or the previously set value are accepted, the *comma* separator must nonetheless be used so that the reader can know which one of the parameters has been “implied.” The separator may be omitted when it is not followed by another parameter.

[42] See note 39.
[43] See note 39.
[44] See note 202 on page 48.
[45] The default value is *N* or generally the last value received with the *^FW* command described on page 10.
[46] The default value is *15* or the value set with the last *^CF* command (page 17).
[47] The default value is *12* or the value set with the last *^CF* command (page 17).

^BYm,r,a (Bar-code default values)

This command makes it possible to set the width of the module, the ratio and height of all bar-codes included in the label, and affects the bar-codes that follow it up to the next *^BY* command.

m is a whole number between *1* and *10*[48] and expresses the width in dots of the narrow bar (the base module of the bar-code)

r is a rational number (with a single decimal-place character) between *2.0* and *3.0*[49] and sets the ratio of the bar-codes. The actual ratio (according to which the bar-code is printed) is limited by the fact that it is not possible to break up a dot into fractions. Therefore, it depends on the parameter *r*, together with the parameter *m* in accordance with table 1. In fixed ratio bar-codes, this parameter has no effect.

a is a whole number between *1* and *9999*[50] and indicates the height of the bar-code, expressed in dots.

Ratio (*r*)	**Module (*m*)**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2.0	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1
2.1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2:1	2.1:1
2.2	2:1	2:1	2:1	2:1	2.2:1	2.16:1	2.1:1	2.12:1	2.1:1	2.2:1
2.3	2:1	2:1	2.3:1	2.25:1	2.2:1	2.16:1	2.28:1	2.25:1	2.2:1	2.3:1
2.4	2:1	2:1	2.3:1	2.25:1	2.4:1	2.3:1	2.28:1	2.37:1	2.3:1	2.4:1
2.5	2:1	2.5:1	2.3:1	2.5:1	2.4:1	2.5:1	2.4:1	2.5:1	2.4:1	2.5:1
2.6	2:1	2.5:1	2.3:1	2.5:1	2.6:1	2.5:1	2.57:1	2.5:1	2.5:1	2.6:1
2.7	2:1	2.5:1	2.6:1	2.5:1	2.6:1	2.6:1	2.57:1	2.65:1	2.6:1	2.7:1
2.8	2:1	2.5:1	2.6:1	2.75:1	2.8:1	2.6:1	2.7:1	2.75:1	2.7:1	2.8:1
2.9	2:1	2.5:1	2.6:1	2.75:1	2.8:1	2.8:1	2.85:1	2.87:1	2.8:1	2.9:1
3.0	3:1	3:1	:31	3:1	3:1	3:1	3:1	3:1	3:1	3:1

table 1 – Relationship between the ratio and width of the module

^B2r,a,h,p,c (Interleaved 2/5)

This is a high-density code, accepts only numerals, and is of variable length with an optional digit check; it accepts ratios from 2 to 3 (see table 8 on page 49).

The command accepts up to 100 characters but the limit is determined by the label width (or by the maximum length of the printing area if the bar-code is rotated by 90°/270°[51]) together with the ratio set by the *^BY* command on page 19. In other words, if the bar-code spills over the boundaries of the printing area due its excessive size (determined by the number of characters to be coded and/or by the ratio), it could be wrongly interpreted by the reader. In any event, the part that is included in the useful printing area is printed.

[48] The default value is *2*.
[49] The default value is *3.0*. The point is used as the decimal separator.
[50] The default value is *10*. The upper limit is determined by the greatest length of the label (see note 7 on page 9).
[51] The maximum length of the printing area is in turn limited by the available RAM. Please refer to the User Manual.

Regardless, the number of characters coded by the 2/5 Interleaved bar-code must be even: if it is not, a zero will be added to the most significant position (on the left).
r is a letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[52]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
a is a whole number between *1* and *9999*[53] and expresses the height of the bar-code expressed in dots[54].
h is the letter *Y*[55] if printing of the *human readable* field is desired; if not, it is the letter *N*.
p is the letter *Y* (*yes*) or *N*[56] (*no*). If the *human readable* field is desired (when *h=Y*), it is possible to position it above (*p=Y*) or below (*p=N*).
c is the letter *Y* (*yes*) or *N*[57] (*no*). *Y* if the check digit is to be added, *N* if not.

^B3r,c,a,h,p (Code 39)

This code accepts numerals and upper-case letters, and its length is variable with optional check digit; it accepts ratios from 2 to 3 (see table 8 on page 49).
The command accepts up to 100 characters, but the limit is determined by the width of the label (or the maximum length of the printing area if the bar-code is rotated by 90°/270°[58]) together with the ratio set by the *^BY* command on page 19. In other words, if the bar-code spills over the boundaries of the printing area due its excessive size (determined by the number of characters to be coded and/or by the ratio), it could be wrongly interpreted by the reader. In any event, the part that is included in the useful printing area is printed.
The two start and stop asterisks provided by Code 39 are automatically added.
r is a letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[59]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
c is the letter *Y* (*yes*) or *N*[60] (*no*). *Y* if the check-digit is to be added, *N* if not.
a is a whole number between *1* and *9999*[61] and expresses the height of the bar-code, expressed in dots[62].
h is the letter *Y*[63] if printing of the *human readable* field is desired, *N* if not.

[52] The default value is that of the last *^FW* command (page 10) received.
[53] The default value is that of the last *^BY* command (page 19) received.
[54] See note 5 on page 9.
[55] Default value.
[56] Default value.
[57] Default value.
[58] The maximum length of the printing area is in turn limited by the available RAM. Please refer to the User Manual.
[59] The default value is that of the last *^FW* command (page 10) received.
[60] Default value.
[61] The default value is that of the last *^BY* command (page 19) received.
[62] See note 5 on page 9.
[63] Default value.

p is the letter *Y* (*yes*) or *N*[64] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).

^*B8r,a,h,p (EAN 8)*

This is a reduced EAN 13 code (see page 22), accepts only numerals, has a fixed length (7 characters) with check digit and a fixed ratio (see table 8 on page 49).
Since its length is fixed, if you attempt to code a string shorter than 7 characters, CZL will add as many zeroes as are missing from the most significant positions (to the left). At the same time, if the string is longer than 7 characters, it is truncated at the seventh character, eliminating all the less significant ones (to the right).
r is a letter determining the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[65]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
a is a whole number between *1* and *9999*[66] and expresses the height of the bar-code as expressed in dots[67].
h is the letter *Y*[68] if printing of the *human readable* field is desired, and *N* if not.
p is the letter *Y* (*yes*) or *N*[69] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).

^*B9r,a,h,p,c (UPC E)*

This code is a reduced version of UPC A (see page 27), accepts only numerals, and has a fixed length (10 characters) with check digit; the ratio is fixed (see table 8 on page 49).
Since its length is fixed, if you attempt to code a string shorter than 10 characters, CZL will add as many zeroes as are missing from the most significant positions (to the left). At the same time, if the string is longer than 10 characters, it is truncated at the seventh character, eliminating all the less significant ones (to the right).
r is letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[70]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
a is a whole number between *1* and *9999*[71] and indicates the height of the bar-code, expressed in dots[72].

---

[64] Default value.
[65] The default value is that of the last ^*FW* command (page 10) received.
[66] The default value is that of the last ^*BY* command (page 19) received.
[67] See note 5 on page 9.
[68] Default value.
[69] Default value.
[70] The default value is that of the last ^*FW* command (page 10) received.
[71] The default value is that of the last ^*BY* command (page 19) received.
[72] See note 5 on page 9.

h is the letter *Y*[73] if printing of the *human readable* field is desired, *N* if not.
p is the letter *Y* (*yes*) or *N*[74] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).
c is the letter *Y* (*yes*) or *N*[75] (*no*). *Y* if the check digit is to be added, *N* if not.

^BCr,a,h,p,c,m (Code 128)

This is a high-density code, accepts numerals and letters (both upper and lower-case), and its length is variable, with optional check digit. Its ratio is fixed (see table 8 on page 49).
Code 128 accepts three character subsets (see table 2), and each of these contains 106 characters that can be printed differently for each subset.

[73] Default value.
[74] Default value.
[75] Default value.

Value	Subset A	Subset B[76]	Subset C	Value	Subset A	Subset B	Subset C
0	Space	Space	00	53	U	U	53
1	!	!	01	54	V	V	54
2	“	“	02	55	W	W	55
3	#	#	03	56	X	X	56
4	$	$	04	57	Y	Y	57
5	%	%	05	58	Z	Z	58
6	&	&	06	59	[	[	59
7	'	'	07	60	\	\	60
8	(	(	08	61	]	]	61
9	)	)	09	62			62
10	*	*	10	63	_	_	63
11	+	+	11	64	NUL	NUL	64
12	,	,	12	65	SOH	SOH	65
13	-	-	13	66	STX	STX	66
14	.	.	14	67	ETX	ETX	67
15	/	/	15	68	EOT	EOT	68
16	0	0	16	69	ENQ	ENQ	69
17	1	1	17	70	ACK	ACK	70
18	2	2	18	71	BEL	BEL	71
19	3	3	19	72	BS	BS	72
20	4	4	20	73	HT	HT	73
21	5	5	21	74	LF	LF	74
22	6	6	22	75	VT	VT	75
23	7	7	23	76	FF	FF	76
24	8	8	24	77	CR	CR	77
25	9	9	25	78	SO	SO	78
26	:	:	26	79	SI	SI	79
27	;	;	27	80	DLE	DLE	80
28	<	<	28	81	DC1	DC1	81
29	=	=	29	82	DC2	DC2	82
30	>	>	30	83	DC3	DC3	83
31	?	?	31	84	DC4	DC4	84
32	@	@	32	85	NAK	NAK	85
33	A	A	33	86	SYN	SYN	86
34	B	B	34	87	ETB	ETB	87
35	C	C	35	88	CAN	CAN	88
36	D	D	36	89	EM	EM	89
37	E	E	37	90	SUB	SUB	90
38	F	F	38	91	ESC	ESC	91
39	G	G	39	92	FS	FS	92
40	H	H	40	93	GS	GS	93
41	I	I	41	94	RS	RS	94
42	J	J	42	95	US	US	95
43	K	K	43	96	FNC3	FNC3	96
44	L	L	44	97	FNC2	FNC2	97
45	M	M	45	98	SHIFT	SHIFT	98
46	N	N	46	99	Subset C	Subset C	99
47	O	O	47	100	Subset B	Subset B	100
48	P	P	48	101	FNC4	FNC4	101
49	Q	Q	49	102	FNC1	FNC1	102
50	R	R	50	103			103
51	S	S	51	104			104
52	T	T	52	105			105

table 2 - Subset of Code 128

[76] Default value.

Each subset can be chosen in one of two ways: using a special character (for “subset selection”) incorporated in the string to be coded, or a code at the beginning of the string (called “default subset selection”).
In the first case, all the characters following the special character belong to the chosen subset up to the next subset selection character (see table 3).

Subset selection character	**Decimal value**	**Subset A**	**Subset B**[77]	**Subset C**
><	62			
>0	30	>	>	
>=	94		~	
>1	95	USQ	DEL	
>2	96	FNC 3	FNC 3	
>3	97	FNC 2	FNC 2	
>4	98	SHIFT	SHIFT	
>5	99	CODE C	CODE C	
>6	100	CODE B	FNC 4	CODE B
>7	101	FNC 4	CODE A	CODE A
>8	102	FNC 1	FNC 1	FNC 1

table 3 – Subset selection characters

In the second case, the entire string is coded in the selected subset unless one of the subset selection characters occurs.

Subset selection character	**Decimal value**	
>9	103	Subset A
>:	104	Subset B[78]
>;	105	Subset C

table 4 – Default subset selection character

In general, the command accepts up to 100 characters, but the limit is determined by the width of the label (or the maximum length of the printing area if the bar-code is rotated by 90°/270°[79]) and the ratio set by the *^BY* command on page 19. In other words, if the bar-code spills over the boundaries of the printing area due its excessive size (determined by the number of characters to be coded and/or by the ratio), it could be wrongly interpreted by the reader. In any event, the part that is included in the useful printing area is printed.
r is a letter that determines the rotation of the bar-code. It can assume the following values:

N: *normal* rotation (0°)[80]
R: *rotated* (90°)

[77] Default value.
[78] Default value.
[79] The maximum length of the printing area is in turn limited by the available RAM. Please refer to the User Manual.
[80] The default value is that of the last *^FW* command (page 10) received.

I: *inverted* (180°)
B: *bottom up* rotation (270°)
a is a whole number between *1* and *9999*[81] and indicates the height of the bar-code, expressed in dots[82].
h is the letter *Y*[83] if printing of the *human readable* field is desired, *N* if not.
p is the letter *Y* (*yes*) or *N*[84] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).
c is the letter *Y* (*yes*) or *N*[85] (*no*). *Y* if the check digit is to be added, *N* if not.
m is the letter *N* or *U* depending on whether the *UCC Case*[86] (*U*) mode is selected or not (*N*).

Code 128 - Subset B

Since subset B is the most common, it is the one chosen by default if no *default subset selection* code is chosen (see table 4 on page 24).
Subset B accepts all alphanumeric characters with the exception of those higher than *94* and the characters ^, > (*30*), ~ (*94*) in table 2 on page 23, for which the subset selection characters in table 3 on page 24 must be used.

Code 128 - Subset A e C

Subsets A and C accept only pairs of numerals.
In the first case (Subset A), the letter corresponding to the pair of numerals as listed in table 2 on page 23 is coded, while in the second (Subset C), the numbers are coded as is.
Therefore, for the codes listed in table 2 on page 23, even alphanumeric characters can be coded with subset A, whereas they cannot be with subset C. In fact, alphanumeric characters are ignored in this last subset. If the letter is in the first position of a pair of characters (e.g. *A1*), it is simply ignored, and if the letter is in the second position (e.g. *1A*) the entire pair is invalidated.
In the case of subset C, there must be an even number of digits, and thus, if there is an odd number of digits, there are different outcomes depending on whether the check digit set by parameter *c* has been enabled. If it is enabled, check digit makes the number of digits balance, and if not, the excess amount is lopped off.
If check digit is enabled and the number of digits is even, it is not printed because, if it were, the even number of digits would be upset.

^BEr,a,h,p (EAN 13)

This bar-code accepts only numerals, is fixed in length (12 digits), the 13th is check digit (which CZL automatically calculates), and the ratio is fixed (see table 8 on page 49).

[81] The default value is that of the last ^*BY* command (page 19) received.
[82] See note 5 on page 9.
[83] Default value.
[84] Default value. If the *UCC Case* mode is used, the default value is *Y*.
[85] Default value.
[86] When the *UCC Case* mode is selected, the Code 128 length becomes the set length (19 characters), accepts only numerals, and subset C is automatically selected with the value FNC1. Characters beyond the first 19 are eliminated and, if there are less than 19 characters, as many zeroes are added as missing in the least significant position (on the right).

Any character beyond the 12th is lopped off (the least significant ones to the right); as many zeroes as necessary to reach 12 digits are added (in the most significant positions to the left).
r is a letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[87]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
a is a whole number between *1* and *9999*[88] and indicates the height of the bar-code, expressed in dots[89].
h is the letter *Y*[90] if printing of the *human readable* field is desired, *N* if not.
p is the letter *Y* (*yes*) or *N*[91] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).

^BKr,c,a,h,p,s,t (ANSI Codabar)

This bar-code accepts alphanumeric characters, numerals, and symbols, its length is variable, check digit is optional, and the ratio falls between 2 and 3 (see table 8 on page 49).
In general, the command accepts up to 100 characters, but the limit is determined by the width of the label (or the maximum length of the printing area if the bar-code is rotated by 90°/270°[92]) and the ratio set with the *^BY* command on page 19. In other words, if the bar-code spills over the boundaries of the printing area due its excessive size (determined by the number of characters to be coded and/or by the ratio), it could be wrongly interpreted by the reader. In any event, the part that is included in the useful printing area is printed.
r is a letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[93]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
c is the letter *Y* (*yes*) or *N*[94] (*no*). *Y* if the check-digit is to be added, *N* if not.
a is a whole number between *1* and *9999*[95] and indicates the height of the bar-code, expressed in dots[96].
h is the letter *Y*[97] if printing of the *human readable* field is desired, *N* if not.
p is the letter *Y* (*yes*) or *N*[98] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).

[87] The default value is that of the last *^FW* command (page 10) received.
[88] The default value is that of the last *^BY* command (page 19) received.
[89] See note 5 on page 9.
[90] Default value.
[91] Default value. If the *UCC Case* mode is used, the default value is *Y*.
[92] The maximum length of the printing area is in turn limited by the available RAM. Please refer to the User Manual.
[93] The default value is that of the last *^FW* command (page 10) received.
[94] Default value.
[95] The default value is that of the last *^BY* command (page 19) received.
[96] See note 5 on page 9.
[97] Default value.
[98] Default value.

s is the *start* character (*A*[99], *B*, *C*, *D*, *, *N*, *E* o *T*).
t is the *start* character (*A*[100], *B*, *C*, *D*, *, *N*, *E* o *T*).

^BMr,c,a,h,p,d (MSI)

This bar-code accepts only numerals, is of variable length (up to 13 digits) when check digit is not enabled, up to 14 digits when the optional check digit is enabled (according to the valued assumed by parameter *c*), and the ratio varies between 2 and 3 (see table 8 on page 49).
r is a letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[101]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
c is *A* if check digit is not enabled, *B*[102] if check digit *1 Mod 10* is enabled, *C* if check digit *2 Mod 10* is enabled, and *D* if check digit *1 Mod 10* and *1 Mod 11* are enabled.
a is a whole number between *1* and *9999*[103] and indicates the height of the bar-code expressed in dots[104].
h is the letter *Y*[105] if printing of the *human readable* field is desired, *N* if not.
p is the letter *Y* (*yes*) or *N*[106] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).
d is letter *N*[107] if printing of the check digit in the *human readable* field is not desired (*h*=*Y*), *Y* if it is desired.

^BSr,a,h,p (UPC/EAN Extensions)

This bar-code is the extension of bar-codes *EAN* and *UPC*. It is treated as a bar-code independently of whether it is used together with one of the first two (otherwise it could result in erroneous readings).
This bar-code accepts only numerals, is of fixed length (2 or 5 digits), does not have check digit, and its ratio is fixed (see table 8 on page 49).
Any character after the second (in the first case) or after the fifth (in the second case) is lopped off (the least significant ones to the right); alternatively, as many zeroes as necessary to reach 2 or 5 digits are added (in the most significant positions to the left).
r is a letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[108]
R: *rotated* (90°)

[99] Default value.
[100] Default value.
[101] The default value is that of the last *^FW* command (page 10) received.
[102] Default value.
[103] The default value is that of the last *^BY* command (page 19) received.
[104] See note 5 on page 9.
[105] Default value.
[106] Default value.
[107] Default value.
[108] The default value is that of the last *^FW* command (page 10) received.

I: *inverted* (180°)
B: *bottom up* rotation (270°)
a is a whole number between *1* and *9999*[109] and indicates the height of the bar-code, expressed in dots[110].
h is the letter *Y*[111] if printing of the *human readable* field is desired, *N* if not.
p is the letter *Y* (*yes*) or *N*[112] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).

^BUr,a,h,p,c (UPC A)

This bar-code accepts only numerals, its length is fixed (11 characters), check digit is optional, and the ratio is fixed (see table 8 on page 49).
Any digit beyond the 11th is lopped off (the least significant ones to the right); as many zeroes as necessary to reach 11 digits are added to the most significant positions to the left.
r is a letter that determines the rotation of the bar-code. It can assume the following values:
N: *normal* rotation (0°)[113]
R: *rotated* (90°)
I: *inverted* (180°)
B: *bottom up* rotation (270°)
a is a whole number between *1* and *9999*[114] and indicates the height of the bar-code expressed in dots[115].
h is the letter *Y*[116] if printing of the *human readable* field is desired, *N* if not.
p is the letter *Y* (*yes*) or *N*[117] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).
c is the letter *Y*[118] (*yes*) or *N* (*no*). *Y* to add check digit, and *N* not to.

^BZr,a,h,p (PostNet)

This bar-code accepts only numerals, is of variable length, its ratio is fixed, and check digit is not enabled (see table 8 on page 49).
In general, the command accepts up to 100 characters, but the limit depends on the width of the label (or the maximum length of the printing area if the bar-code is rotated 90°/270°[119]) and the ratio set with the *^BY* command on page 19. In other words, if the bar-code spills over the boundaries of the printing area due its excessive size (determined by the number of characters to be coded and/or

[109] The default value is that of the last *^BY* command (page 19) received.
[110] See note 5 on page 9.
[111] Default value.
[112] Default value. When the *UCC Case* mode is used, the default value is *Y*.
[113] The default value is that of the last *^FW* command (page 10) received.
[114] The default value is that of the last *^BY* command (page 19) received.
[115] See note 5 on page 9.
[116] Default value.
[117] Default value. If the *UCC Case* mode is used, the default value is *Y*.
[118] Default value.
[119] The maximum length of the printing area is in turn limited by the available RAM. Please refer to the User Manual.

by the ratio), it could be wrongly interpreted by the reader. In any event, the part that is included in the useful printing area is printed.

r is a letter that determines the rotation of the bar-code. It can assume the following values:

N: *normal* rotation (0°)[120]

R: *rotated* (90°)

I: *inverted* (180°)

B: *bottom up* rotation (270°)

a is a whole number between *1* and *9999*[121] and indicates the height of the bar-code, expressed in dots[122].

h is the letter *Y* if printing of the *human readable* field is desired, *N*[123] if not.

p is the letter *Y* (*yes*) or *N*[124] (*no*). If the *human readable* field is desired (when *h*=*Y*) it is possible to position it above (*p*=*Y*) or below (*p*=*N*).

3.1.7 Graphic objects

3.1.7.1 Boxes

^GBl,a,s,c (Graphic Box)

This command makes it possible to draw boxes and/or lines that can mark off or highlight parts of the label.

l is a whole number between *0*[125] and *9999*[126] and expresses the width (in dots[127]) of the box.

a is a whole number between *0*[128] and *9999*[129] and expresses the height (in dots[130]) of the box.

s is a whole number between *1*[131] and *9999*[132] and expresses the thickness (in dots[133]) of the box or line.

c is letter *B*[134] when black lines are desired and *W* when white lines are desired. This latter choice is useful when you want to draw a white box inside a black rectangle without using the *reverse* function.

[120] The default value is that of the last *^FW* command (page 10) received.
[121] The default value is that of the last *^BY* command (page 19) received.
[122] See note 5 on page 9.
[123] Default value.
[124] Default value. If the *UCC Case* mode is used, the default value is *Y*.
[125] Default value. When the width is *0* the box becomes a line of height *a* and thickness *s*.
[126] See note 4 on page 9.
[127] See note 5 on page 9.
[128] Default value. When the width is *0* the box becomes a line of width *l* and thickness *s*.
[129] See note 4 on page 9.
[130] See note 5 on page 9.
[131] Default value.
[132] See note 4 on page 9.
[133] See note 5 on page 9.
[134] Default value.

3.1.7.2 Graphic images

~DGd:aa…a.eee,b,r,xx…x (Download Graphic)

CZL makes it possible to save graphic images in hexadecimal format that are created in any application, whether CAD or a graphic layout program. Once they are saved (in temporary memory or elsewhere), the images can be used on the label as desired.
Refer to section 5.2 on page 53, where a printing example with graphic images is described.
The current command controls the following functions: placing the printer in "graphic mode," giving a name to the image (which is the same that will be used to retrieve the image when the label is created), defining its size, and uploading the image on the printer memory.
The name given does not necessarily have to be the same as the name of the file in which the file is stored on the computer. However, it must adhere to several conventions that are explained below.
The CZL emulation interprets the space, "return" character, or extension as the end of the name. Therefore, in order to preserve the uniqueness of names, it is wise not to use a space inside names because two names (consisting of more than one word) having the same first word will be treated as being the same, and the images to which these names have been applied will no longer be treated as separate and distinct. Therefore, different names must be given, even mnemonic ones, to different images: a file given an existing name will not be saved to memory.

d: is *R:*[135] to save the image to RAM, *B:* when it is saved in the base flash, *C:* or *D:* when it is saved in the flash memory on the expander card[136].

aa…a is the name to be attributed to the graphic image. It can be from 1 to 8 characters long and the default value is *UNKNOWN*.

b is the total (whole) number of bytes of the graphic image.

r is the number of bytes per line.

xx…x is a string of numbers in hexadecimal code comprising the representation of the graphic image.

.eee is the extension of the file name, which must be *.GRF* for graphic images.

This format specifies that each block of 4 bits (nibble) is the image of 4 adjacent dots: the high bit (1) corresponds to the on bit (black), and the low bit (0) corresponds to the off bit (white). The entire graphic image is coded in the same way: in practice, it is a single long string of hexadecimal values.
The total number of bytes *b* is a function of the dimensions of the image and the printer resolution (please see the User Manual) according to the following formula:

$$\frac{(l \cdot ris)}{8} \cdot (a \cdot ris) = b$$

[135] Default value.

[136] Since flash memory is a non-volatile memory, it is available on the expander cards so that it can transport the images it contains. Unit *B:* is the flash memory contained in the base machine, *C:* and *D:* are the memory contained in the expander cards (where provided), and the letter indicates in which of the two available slots the expander card is installed. Please refer to the User Manual for more information.

where *l* is the width of the image (in *mm*), *ris* the resolution (in $dots \cdot mm^{-1}$), *a* the height of the image (in *mm*), 8 is the number of bits per byte, and *b* is the total number of bytes. The result of the division (*l·ris*)/8 must be rounded to the next highest whole number.
The number of bytes per line *r* depends on the width of the image and the printer resolution as appears in the following formula:

$$\frac{(l \cdot ris)}{8} = r$$

From which it is deduced that

$$b = r \cdot (a \cdot ris)$$

The hexadecimal code described above for representation of the graphic images involves sending (and more generally, handling) of files of moderate dimensions.
CZL offers the possibility of reducing file dimensions with a special compression algorithm. More specifically, it often happens that perfectly equal sequences of hexadecimal codes must be sent (this corresponds to having extended black parts in the image or recurrent elementary figures). In this case, the letters after *F* can be used as a multiplier according to the following table:

G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

When it receives *UB*, the machine reads (with the identical result) "15 times *B*", in other words *BBBBBBBBBBBBBBB.*
Likewise for lower case letters:

g	h	i	j	k	l	m	N	o	p	q	r	s	t	u	v	w	x	y	z
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400

Sending *hB* is like sending *B:* 40 times
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB.
The upper and lower case letters can be combined so that, for example, *hUB* (or *UhB*) corresponds to "55 times *B*".
In regard to file compression, CZL specifies the use of several other special characters:
The comma (,) fills the current line (from the right) with pairs (whole bytes) of zeroes (*0*) from its position to the end of the line[137].
The exclamation point (*!*) does the same thing, filling the line with ones (*1*).
The colon (*:*) indicates repetition of the last line.
Since the number of bytes is whole, the hexadecimal characters comprising the representation of the graphic image are always even in number. Therefore, the action of the comma and exclamation point characters described above is calculated on the pairs of characters. In other words, to avoid

[137] The length of the line is determined by the parameter *r*.

misinterpretations, the comma (and/or the exclamation point) are inserted after an even number of characters.
For example, *1FF0,* must be sent rather than *1FF0000000* and not *1FF,*. Otherwise, *1FFF000000* can be transformed into *1FFF,*.
The editor or application being used can introduce special characters (*<CR>* and/or *<LF>*) to break the line: the CZL emulation ignores them.

~DN (Abort Download Graphic)

This command makes it possible to interrupt loading of graphic images onto the printer before it is completed. The printer then returns to normal print mode. However, any other ^ or ~ command interrupts loading.

^XGs:aa…a.eee,m_x,m_y (Recall Graphic)

This command is used to recall the graphic image to be printed (e.g. company logo).
Once the image has been stored in memory with the *~DG* command on page 30, it is possible to recall it whenever necessary (even many times), just as it is possible to use it together with other images and/or fields inside the same label.
s: is the unit where the image is stored in memory. *R:*[138] indicates RAM, *B:* (*C:* or *D:*) flash memory[139]. This parameter is optional.
aa…a is the name attributed to the graphic image. It can be from 1 to 8 characters long and the default value is *UNKNOWN*.
m_x is the multiplier along the *x* axis. It is a whole number between *1*[140] and *10*.
m_y is the multiplier along the *y* axis. It is a whole number between *1*[141] and *10*.
.eee is the extension of the name, which must be *.GRF* for graphic images.

^IMs:aa…a.eee (Image Move)

This command, just like the *^XG* command, permits recovery of a previously saved graphic image. The only difference is that it is not possible to change its size, and it requires less processing time.
s: is the unit where the image is stored. *R:*[142] indicates RAM, *B:* (*C:* or *D:*) flash memory[143]. This parameter is optional.
aa…a is the name attributed to the graphic image. It can be from 1 to 8 characters long, and the default value is *UNKNOWN*.
.eee is the extension of the name, which must be *.GRF* for graphic images.

[138] Default value.
[139] See note 136 on page 30.
[140] Default value.
[141] Default value.
[142] Default value.
[143] See note 136 on page 30.

3.1.7.3 Treating labels as graphic images (saving)

CZL makes it possible to treat labels as if they were graphic images. This means that it is possible to save the label image to memory (both temporary and other according to available computer resources; please refer to the User Manual) and then use it as desired, calling it up with the name assigned to it.

This permits optimal computer performance when similar labels (including complicated ones) varying only in a few fields have to be printed. In this case, rather than sending the entire label every time, the common part can be saved and then recalled later, when the missing fields are added (the variable from label to label)[144], leading to significant savings in processing time (in direct relation to the complexity of the label).

^ISd:aa…a.eee,p (Image Save)

This command is used to save the label image to memory (RAM or flash) as a graphic image. It will then be possible to recall it at any time with the name given to it, and then adding any variable fields.

In light of the fact that this command saves the image of the current label, it is normally the last one, just before the command *^XZ* (page 8)

d: is *R:*[145] for saving the image in RAM, *B:* for saving it in the base flash memory, *C:* or *D:* for saving it in the flash memory on the expander card[146].

aa…a is the name attributed to the graphic image. It can be from 1 to 8 characters long, and the default value is *UNKNOWN*.

.eee is the file extension, which must be *.GRF* for graphic images.

^ILs:aa…a.eee (Image Load)

This command recalls the image of label *aa…a* stored in memory unit *s*.

It is normally the second one, just after the command *^XA* (page 8). This way, the label image is recalled and positioned at *0, 0* (see fig. 2 on page 9). The following commands make it possible to overlay the variable fields on the saved labels to compose the complete image of the desired label.

s: is the unit where the image is saved. *R:*[147] indicates RAM, *B:* (*C:* or *D:*) flash memory[148]. This parameter is optional.

aa…a is the name assigned to the graphic image. It can be from 1 to 8 characters long, and the default value is *UNKNOWN*.

.eee is the file extension, which must be *.GRF* for graphic images.

3.1.7.4 Moving and deleting objects

Images (and, more in general, any object) can be moved from one unit to another in a way similar to the *copy* command in DOS. This is useful for moving graphic images from RAM to flash memory

[144] This operation is often referred to with the term *overlay*.
[145] Default value.
[146] See note 136 on page 30.
[147] Default value.
[148] See note 136 on page 30.

or vice-versa, as well as downloading or uploading images onto the expander cards before inserting them in other machines.
The source and/or destination units must be available on the machine in use, otherwise the move will not be completed. Please refer to the User Manual for more information.

^TOs:$a_s a_s \ldots a_s . e_s e_s e_s$,d:$a_d a_d \ldots a_d . e_d e_d e_d$ (Transfer Object)

s: the source unit from which the object will be moved. *R:* indicates RAM, *B:* (*C:* or *D:*) flash memory[149].
$a_s a_s \ldots a_s$ name of the source graphic file. It can be from 1 to 8 characters long and can include the wildcard characters * and *?*[150].
.$e_s e_s e_s$ is the file name extension for graphic images, which is *.GRF*. It can be replaced by the wildcard characters * and *?*[151].
d: is the destination unit to which the object is moved. *R:* indicates RAM, *B:* (*C:* or *D:*) flash memory[152].
$a_d a_d \ldots a_d$ is the name of the graphic image destination. It can be from 1 to 8 characters long and can include the wildcard characters * and *?*[153].
.$e_d e_d e_d$ is the file name extension for graphic images, which is *.GRF*. It can be replaced by the wildcard characters * and *?*[154].
If the destination unit does not have sufficient space to save all objects, only those that fit are moved. To avoid problems in case any doubt exists, it is good practice to check the list of objects actually moved by using the commands *^HW* (page 39) and/or *^WD* (page 39).

^IDs:aa…a.eee (Item Delete)

This command enables deletion of saved objects (only from RAM), and it is capable of selecting them one by one.
Since this command can include the wildcard characters * and *?*[155], it is possible to cancel groups of objects as well.
s: is the unit where the object is saved (which must be *R:*).
aa…a is the name assigned to the object. It can be 1 to 8 characters long and its default value is *UNKNOWN*.
.eee is the file name extension, which is *.GRF*[156] for graphic images and *.ZPL* for formats (see section 3.1.8.5 on page 37).

[149] See note 136 on page 30.
[150] * means “one or more occurrences of any character,” *?* signifies “one and only one occurrence of any character.”
[151] See note 150 on page 34.
[152] See note 136 on page 30.
[153] See note 150 on page 34.
[154] See note 150 on page 34.
[155] See note 150 on page 34.
[156] Default value.

^EG o ~EG (Erase Downloaded Graphic)

This command deletes all the images contained in the RAM memory.

3.1.8 Advanced commands

All commands that are sufficient for printing in CZL emulation have been described in the preceding sections. Several advanced commands are described here so that the user can take full advantage of all the features of the machine, achieve special effects, and/or optimize its performance.

3.1.8.1 Comments

^FXaa…a (Comment)

This command is used when clarifying comments must be inserted in the printing program. Comments inserted in this way do not have any effect on the printed label.
aa…a is an alphanumeric comment string.
Whatever is written between the *^FX* command and the following character ^ (o ~) has no effect on the printed label: its only function is to introduce a comment string for future reference and greater program clarity.
It is good practice to use these comments for the preparation of basic documentation that will simplify subsequent program maintenance.

3.1.8.2 Graphic effects

^FR (Field Reverse Print)

It is possible to write some fields in *reverse* to highlight them: they appear in white against a black background instead of black on a white background. More generally, what is black becomes white and what is white becomes black.
Refer to section 5.3 on page 55 where two examples of labels in reverse are described.
To produce this effect, a black box of the desired dimensions must be provided (see the *^GB* command on page 29) superimposed on the field in question. As a result, the black of the box is superimposed on the black of the writing according to the *OR*[157] mode and the field is "deleted" from the box.
With the command *^FR* the mode of the current field is locally changed to *XOR*[158] until the next *^FS* (or *^XZ*) command. The resulting effect is of a field in reverse whose colors are inverted. More generally, where the field is superimposed on another, the color becomes white.

[157] *OR* mode means that *black* on *black* produces the result *black* (black + black = black), just as black + white = black.
[158] *XOR* mode means that *black* on *black* produces the result *white* (black ⊕ black = white), just as black ⊕ white = black.

Obviously, the *^FR* command is effective only when the field is superimposed on another: if it is not superimposed an another (whether it is a box or another field), it does not have any effect.

^LRa (Label Reverse Print)

This command is very similar to the preceding command *^FR* but affects all fields (following the *^LR* command) of the current and subsequent labels.
a is the letter *Y* to enable the reverse mode, *N*[159] if not.
The reverse mode remains active until it is disabled with the *^LRN* command or the machine is turned off.

3.1.8.3 Serial numerical fields

^SNaa…a,i,z (Serialization Data)

It is often convenient to have a field that can be automatically increased or decreased by a pre-set value for every label.
Refer to section 5.4 on page 56 where an example of printing with serial fields is illustrated.
This command makes it possible to create numerical fields with up to 12 digits. If the field is alphanumeric, the operation is performed on the first whole number found (starting from the right side of the field itself).

aa…a[160] is the starting string, generally alphanumeric, for the field. The numerical portion of the field that must be increased (decreased) is a whole number with a maximum of 12 digits.

i[161] is a related whole number (with the negative sign if the field must be decreased) with a maximum of 12 digits.

z is the letter *Y* if it is necessary to add zeroes to the most significant positions (to the left), *N*[162] if not.

When zeroes are used in the most significant positions, two distinct cases must be pointed out:
if zeroes are included in the field *aa…a* (e.g. *aa…a = 0001*), they are replaced with spaces when *z = N*, and are printed when *z = Y*; if no zeroes are included and in their place are spaces (e.g. *aa…a = (((1*), they are not printed either when *z = N*, or when *z = Y*.
Even if the zeroes are suppressed, when the field value is zero *0*, it is printed all the same.
In compliance with the instructions issued by the *^JZ* command on page 16, the value of the last label printed (the interrupted one) is retained for progressive numbering, from which the printer restarts after resetting.
The increase or decrease of the fields continues until the last label in the current lot is printed (as determined by the *^PQ* command on page 40).

[159] Default value.
[160] 1 is the default value.
[161] 1 is the default value.
[162] Default value.

3.1.8.4 *Variable fields*

When it is necessary to modify a certain number of fields on a label which is for the most part uniform, printer performance can be optimized by using variable fields. Accordingly, only a portion of the image affected by the modified fields is processed, producing a saving in processing time.
The mechanism by which this effect is produced is similar to what is described in section 3.1.7.3 on page 33. In this case, rather than saving the label image and giving it a name (although the operation is rapid, it nonetheless takes a little time, and even more so if the image is saved in flash memory), it is saved in the working RAM until the machine is turned off or it is explicitly deleted. It forms the background for labels printed afterwards, on which variable fields are updated and placed.
See section 5.5 on page 57 where an example of printing with variable fields is described.

^MCa (Map Clear)

Normally, after a label is printed, its image (bitmap) is deleted from the working RAM. When using variable fields, the image is saved (just as described above) with this command.
a is the letter *Y* for deleting the bitmap, *N*[163] not to delete it.
When the *^MCN* command is received, the machine saves the image of the current label (until it is switched off or it receives the next *^MCY* command) and prints it as the background for all following labels. Accordingly the variable fields can be superimposed on it with the following command *^FV*[164].

^FVaa…a (Variable Field Data)

This command replaces the corresponding command *^FD* (page 13) when the field is variable.
aa…a is an alphanumeric string of at most 255 characters. If it is omitted the command is ignored.
The *^FV* fields are always deleted after they have been printed, while *^FD* fields are not.
If there is more than one variable field on the label, they are updated in the same order in which they were first entered.

3.1.8.5 *Saving the "format"*

Aside from saving the graphic image of the label (see section 3.1.7.3 on page 33), it is possible to save the *formats*. The format is comprised by all the commands that describe the label image.
By saving the format, transmission time is optimized, since similar labels (which differ by only a few fields) are not sent *n* times. However, processing time is not shortened. In fact, when the label is recalled from memory before printing, its image is processed with the addition of any variable fields; this happens whenever the label is printed. Aside from optimizing the aforementioned transmission time, saving the format reduces the memory space used since the space occupied by the format is less than that of the graphic image.
The formats can be saved in both RAM and flash memory within the limits of available space. CZL is not programmed to give any error message if the format is larger than the space available on the destination unit. In this case, the format is not saved in its entirety, and the user becomes aware of

[163] Default value.
[164] The machine operates by default in *OR* mode. See note 157 on page 35.

this only when he or she attempts to recall it or upon recalling the list of objects contained in the memory by means of the *^HW* (page 39) and/or *^WD* commands (page 39).

^EF o ~EF (Erase Format)

This command deletes all the formats saved on the RAM unit.
A formatting selection can be deleted with the *^ID* command on page 34 (the file name extension for formatting objects is *.ZPL*).

^DFd:aa…a.eee (Download Format)

This command makes it possible to save the label descriptions in CZL language as text strings in RAM and/or flash memory while assigning a name to them.
Accordingly, the label can be recalled as desired by using the *^XF* command (page 39), while possibly updating the dynamic fields defined with the *^FN* command (page 38).
This command must be come immediately after the *^XA* command (page 8) and is followed by the commands that must be saved.
d: is the letter which identifies the format destination unit. This is *R:*[165] when the format is saved in RAM, *B:* when it is saved in the base flash memory, *C:* or *D:* when it is saved in the flash memory on the expander card[166].
aa…a is the name to be assigned to the format. It can from 1 to 8 characters long, and the default value is *UNKNOWN*.
.eee is the file name extension, which must be *.ZPL* for formats.
A label containing the *^DF* command is not printed until it is recalled from the memory, and every immediate command contained in it has an indefinite effect.

^FNn (Field Number)

Analogously to the overlay function (see section 3.1.7.3 on page 33) and use of variable fields (see section 3.1.8.4 on page 37), it is possible here as well to update the fields before printing the labels.
In the first case, the fields are superimposed on the image of the label by giving them precise coordinates: since the CZL functions by default in *OR*[167] mode, the effect is superimposition of field on top of the area reserved to it (intentionally left white[168]).
In the second case, the field replaces the variable field defined by the *^FV* command (on page 37).
In this case, the process is similar to but more versatile than the second: since the format is saved with a name, it is necessary to number the variable fields (which are more accurately defined as *dynamic*). Accordingly, it is possible to refer to them directly without respecting their order. It is also possible to assign the same identifying number to more than one field so that they can be updated simultaneously with the same content.
The *^FN* is used instead of the *^FD* command (page 13) upon saving, and together with the *^FD* command when the label is recalled before printing.

[165] Default value.
[166] See note 136 on page 30.
[167] See note 157 on page 35.
[168] Or black if the field is to be in reverse (see section 3.1.8.2 on page 35).

n is the whole number assigned to the dynamic field to identify it. It falls between *1*[169] and *9999*.

^*FAn (Field Allocate)*

The number *n* is a whole number between *1* and *256* that indicates the number of characters and space required to reserve the space to be occupied by the dynamic fields.

^*XFs:aa…a.eee (Recall Format)*

This command recalls from memory unit *s:* the format *aa…a.eee* so that the described image can be completed with the necessary dynamic fields, processed, and then printed.
The format can be recalled whenever necessary, and the dynamic fields are updated periodically with the current value. If dynamic fields (numbered) have been provided for at the time of saving, these fields must be updated (with the ^*FN* command on page 38) when the format is recalled.

s: is the unit where the format is saved. *R:*[170] indicates RAM, *B:* (*C:* or *D:*) flash memory[171]. This parameter is optional.

aa…a is the name assigned to the format. It can be from 1 to 8 characters long and its default value is *UNKNOWN*.

.eee is the file name extension, which must be *.ZPL* for formats.

^*WDs:aa…a.eee (Print Directory on Label)*

This command is necessary for producing a list of the objects contained in the memory units, including an indication of the file name extension and size, and printed on a label.

s: is the unit for which the list is desired. *R:*[172] indicates RAM, *B:* (*C:* or *D:*) the flash memory[173]. This parameter is optional.

aa…a is the name for the list, and it may contain the wildcard characters * and *?*[174]. It can be from 1 to 8 characters long, and its default value is *.

.eee is the file name extension, and it too may contain the wildcard characters * and *?*[175].

^*HWs:aa…a.eee (Host Directory List)*

This command makes it possible to obtain a list of the objects saved in the machine memory (whether in RAM and/or flash memory). Unlike the ^*WD* command, the list is returned to the host as long as the printer is connected through the serial port.

s: is the unit whose list is desired. *R:*[176] indicates RAM, *B:* (*C:* or *D:*) flash memory[177]. This parameter is optional.

[169] The default value is *0*.
[170] Default value.
[171] See note 136 on page 30.
[172] Default value.
[173] See note 136 on page 30.
[174] See note 150 on page 34.
[175] See note 150 on page 34.
[176] Default value.
[177] See note 136 on page 30.

aa…a is the name of the desired list, and it may contain the wildcard characters * and *?*[178]. It can be from 1 to 8 characters long and its default value is *.
.eee is the file name extension, and it too may contain the wildcard characters * and *?*[179].
The strings, fixed in length, returned by the printer have a defined format:

```
<STX><CR><LF>
- DIR R:            xx<CR><LF>
*(aaaaaaaa.eee((nnnnnn(((((<CR><LF>
*(aaaaaaaa.eee((nnnnnn(((((<CR><LF>
<CR><LF>
- xxxxxxx bytes free<CR><LF>
<ETX>
```

where *aaaaaaaa* is the name of the object, *eee* is its extension, and the symbol *(* indicates a space.
This command is processed in the order in which it is sent to the printer. The printer returns the list of objects saved as soon as it can consistently with the operations that it was already performing at the time the command was received.

3.1.8.6 Copies of labels

^PQt,p,d,c (Print Quantity)

This command makes it possible to print the same label *t* times while checking several operations, such as the number of labels, before the machine pauses, and the number of copies for each serial field (see section 3.1.8.3 on page 36).
t is a whole number between *1*[180] and *99,999,999* that indicates the number of copies desired.
p is a whole number between *0*[181] and *99,999,999* that indicates the number of copies printed before the machine is paused.
d is a whole number between *1*[182] and *99,999,999* that indicates at what point in the printing operation it is desired to increase the serial fields. In practice, it is possible to print the same label *d* times without increasing the field.
o is the letter *Y* to prevent the printer from going into pause after *p* labels; it is the letter *N*[183] for enabling the parameter *p,* in other words, to pause the printer after *p* labels.

178 See note 150 on page 34.
179 See note 150 on page 34.
180 Default value.
181 Default value.
182 Default value.
183 Default value.

3.1.8.7 Setting printer speed

^PRs,f (Print Rate)

This command is used to set the print and/or feed rate of the printer independently of the rate set in configuration.
s is a number or letter according to the following scheme:
2 (or *A*[184]) for a speed of 2 inches a second;
3 (or *B*) for a speed of 3 inches a second;
4 (or *C*) for a speed of 4 inches a second;
5 for a speed of 5 inches a second;
6 (or *D*) for a speed of 6 inches a second;
8 (or *E*) for a speed of 8 inches a second;
f is a number or letter according to the following scheme:
2 (or *A*) for a speed of 2 inches a second;
3 (or *B*) for a speed of 3 inches a second;
4 (or *C*) for a speed of 4 inches a second;
5 for a speed of 5 inches a second;
6 (or *D*[185]) for a speed of 6 inches a second;
8 (or *E*) for a speed of 8 inches a second.
The printer operates at the selected speed until it is turned off and/or it receives a new *^PR* command.
Since the speed and temperature corresponding to the type of ribbon and support used affect printing quality, it is advisable to perform a printing test to determine the proper speed.

3.1.8.8 Backfeed

When the peel-off system or cutter is used (see the User Manual for the machine in question), upon completion of printing operations, the paper must stick out far enough for proper alignment of the cutting blade or peel-off system. Before resuming printing of the next label, the paper must then return to position in alignment with the head. This is the *backfeed* movement. The extent of this adjustment can be selected from the panel by means of a special parameter. Please refer to the User Manual for instructions on how to use it.
When the present sensor is enabled (and thus the peel-off system or cutter are used), the backfeed is automatically enabled (by the amount set with the parameter described above). However, to disable it, the following command is used.

^XB (Suppress Backfeed)

This command force disables the backfeed at the end of label printing until the machine is turned off.

[184] Default value.
[185] Default value.

3.1.8.9 Resolution

The maximum resolution of the machine depends on the model being used (please refer to the User Manual). If not required by the application, it is possible to select a resolution that is lower than the allowed maximum in order to optimize processing times.

^JMa (Set resolution)

This command is used to choose between the maximum possible resolution and half of it.
a is letter *A* for maximum resolution (203 dpi/8 dots per mm or 300 dpi/12 dots per mm). It is letter *B* for half resolution (100 dpi/4 dots per mm or 150 dpi/6 dot per mm).
Obviously, doubled dot size corresponds to half resolution, and thus the size of the images appears doubled.

3.2 ~ Commands

Immediate commands are sent to the printer at any time and provide, where possible, for immediate execution. They can be sent in groups or individually.
Commands regarding treatment of graphic images (*~DG* and *~DN*) have already been described in section 3.1.7.2 on page 30.
All immediate commands are preceded by the tilde character (~)[186]. In general, they are "high level" commands through which it is possible to act "physically" on the printer configuration parameters.
Commands requiring a response by the printer work only if it is connected to the host through the serial port since the parallel port is not bi-directional. In other words, to use these commands correctly, the printer must be connected through the serial port: otherwise, the host may wait for a response that will never arrive, and this could lead to malfunctions.

3.2.1 Query commands

~HM (Memory Status)

The printer returns the RAM memory status to the host. The returned string consists of 3 numbers comprised of four digits separated by commas, according to the following format:

$$n_t n_t n_t n_t, n_d n_d n_d n_d, n_l n_l n_l n_l$$

$n_t n_t n_t n_t$ is the total number of Kbytes available on the machine;
$n_d n_d n_d n_d$ is the number of Kbytes available to the user for saving graphic images, etc.;

[186] The tilde control character (~) can be replaced by using the ~CC command on page 46.

$n_l n_l n_l n_l$ is the number of free Kbytes for saving graphic images, etc. taken from $n_d n_d n_d n_d$ of those already used.

Obviously, $n_t n_t n_t n_t > n_d n_d n_d n_d > n_l n_l n_l n_l$.

The first two values are set through configuration of the machine, or the inserted expander cards; the third value changes "dynamically" upon saving the graphic images and/or other objects.

~HS (Host Status)

The printer reports the status through three strings, where each begins with the character *<STX>* and ends with the sequence *<ETX><CR><LF>*, just as the lines are separated by the host for easy reading.

The format of the first one of these is:

<STX>aaa,b,c,dddd,eee,f,g,h,iii,j,k,l<ETX><CR><LF>

aaa: is a base 8 number that describes the communication parameters of the serial port. *aaa* is the octal representation of the binary number $a_8 a_7 a_6\ a_5 a_4 a_3\ a_2 a_1 a_0$.

a_8, together with $a_2\ a_1\ a_0$ indicates the baud rate.

0 001 = 300 baud
0 010 = 600 baud
0 011 = 1,200 baud
0 100 = 2,400 baud
0 101 = 4,800 baud
0 110 = 9,600 baud
0 111 = 19,200 baud
1 001 = 38,400 baud

a_7 indicates the handshake protocol. XON/XOFF when it is low (*0*), hardware when it is high (*1*).

a_6 indicates the parity. Odd when it is low (*0*), even when it is high (*1*).

a_5 indicates whether parity has been enabled. It has not when it is low (*0*), and it has when it is high (*1*).

a_4 indicates the number of stop bits. Two when it is low (*0*), one when it is high (*1*).

a_3 indicates the number of data bits. Seven when it is low (*0*), eight when it is high (*1*).

b: is *1* when the machine is out of paper, *0* in other cases.

c: is *1* when the printer is in pause, *0* in other cases.

dddd: is a whole number of four digits that indicates the length of the current label, expressed in dots.

eee: is the whole number of three digits that indicates the number of labels in the reception buffer.

f: is *1* when the memory buffer is full. *0* in other cases.

g: is *1* when the communications diagnostic mode is enabled. *0* in other cases.

h: is *1* when the current label is being processed. *0* in other cases.

iii: is always *000*.

j: is *1* when the RAM is "corrupted." *0* in other cases.

k: is *1* when the head is too cold. *0* in other cases.

l: is *1* when the head is too hot. *0* in other cases.

The format of the second of these strings is:

<STZ>mmm,n,o,p,q,r,s,t,uuuu,v,www<ETX><CR><LF>

mmm: is a base 8 number that describes the communications parameters of the serial port. *mmm* is the octal representation of the binary number $m_8m_7m_6$ $m_5m_4m_3$ $m_2m_1m_0$.

m_8: is always *0*. Reserved.

m_7 indicates the type of form in use. With a reference marker (gap, black tick mark, or hole) when it is low (*0*), continuous form when it is high (*1*).

m_6 is always *0*. Reserved.

m_5 is always *0*. Reserved.

m_4, m_3, m_2, m_1, are not used and thus always *0*.

m_0 indicates the printing mode. Direct thermal transfer when it is low (*0*), thermal transfer when it is high (*1*).

n: is always *0*. Reserved.
o: is always *0*. Reserved.
p: is high (*1*) when the ribbon is out. *0* in other cases.
r: is always *0*. Reserved.
s: is always *6*. Reserved.
t: is *1* when there is a label under the present sensor. *0* in other cases.
uuuu: is the whole number of four digits that indicates the number of labels in the current batch that remain to be printed.
v: is always *1*. Reserved.
www: is the whole number of 3 digits that indicates the number of graphic images in the RAM memory.
The format of the third string is:

<STZ>xxxx,y<ETX><CR><LF>

xxxx: is always *0000*. Reserved for future uses.
y: is always *0*. Reserved for future uses.

~JR (Power On Reset)

This command has exactly the same effects as shutting off and then turning on the machine. All registers are reset, the RAM is deleted, etc.

3.2.2 Calibration and setting commands

~JC (Set Media Sensor Calibration)

This command forces adjustments in the sensitivity of the paper sensor via software (please refer to the User Manual) and measures the length of the label in use so that it is properly lined up under the head.

~JL (Set Label Length)

This command, unlike the previous command *~JC*, only measures the length of the label being used.

3.2.3 Cancel commands

~JA (Cancel All)

This command terminates printing of the current label and deletes the rest from the command buffer, as if the user had pressed the *CANCEL* button.

~JP (Pause and Cancel Format)

Similarly to the *~JA* command above, this command cancels the current job and places the printer in pause. This command has the same effect as pressing the *CANCEL* button without pausing the printer.

~JX (Cancel Current Partially Input Format)

This command cancels the label that is being sent to the printer without affecting the others (e.g. the one currently being printed).

3.2.4 Control commands

~PH o ^PH (Slew to Home Position)

This command pushes out a blank label just like the *FEED* button. *~PH* works after the current label is printed or when the printer is paused. *^PH* is sent after completion of the entire label printing job.

~PP o ^PP (Programmable Pause)

~PP pauses the printer after the current label is printed. *^PP* pauses it after the entire label printing job is completed. In both cases, the printer remains in pause until the *ON LINE* button is pressed or another *~PS* command is received.

~PS (Print Start)

This command brings the printer back on line after a *~PP* or *^PP* is sent, just as if the *ON LINE* button were pressed on the control panel.

3.2.5 Prefix replacement commands

~CCx o ^CCx (Change Caret)

With some hosts, the CZL command prefixes must be other than a tilde (~) or a caret (^).
The caret character can be changed with any two others from the ASCII table (see table 9 on page 59 and table 10 on page 60) with the current command.
x is any ASCII character, as long as it has not already been used for another prefix. Obviously, the alternative character must be carefully chosen so that it will not be confused with other printable characters and thus lead to problems of interpretation.

~CTx o ^CTx (Change Tilde)

Just like the tilde character, the caret character can be changed. This command is used in the same way as the *~CC* (*^CC*) command above.
x is any ASCII character that has not already been used for another prefix.
Both commands must be bracketed by two limiting commands *^XA* and *^XZ* (see page 8) and apply to all subsequent commands.

3.2.6 Diagnostic commands

~JD (Enable Dump Mode)

In *dump mode,* the printer does not interpret incoming commands but prints them in ASCII with their respective hexadecimal code so that the proper functioning of the printer itself can be evaluated.
The printer can be placed in *dump mode* when it is turned on (please refer to the User Manual) or by software with this command, *~JD.*

~JE (Disable Dump Mode)

This command disables the dump mode, and the printer resumes interpreting the commands it receives.

~WC (Print Configuration Label)

This command produces the print configuration label, just as it can be printed upon turning on the machine (please refer to the User Manual).

4. FONT AND BAR-CODE SPECIFICATIONS

Different types of fonts and bar-codes are available. The printing density depends on the resolution of the head used on the printer, which is better explained in the User Manual. The following specifications summarize the characteristics of the available fonts and bar-codes.

4.1 Fonts

The image of the label to be printed can be built using the commands described in section 3.1. The font used is the default font, which itself can be defined with the *^CF* command (page 17). Following below are the features of each.

Font[187]	**Dimensions of matrix**[188]**. Height x width (in dots)**	**Size of each character**[189]**. Height x width (in mm)**	**Number of characters per mm (in mm^{-1})**
A	9 x 5	1.12 x 0.76	1.31
B	11 x 7	1.37 x 1.12	0.89
C, D	18 x 10	2.26 x 1.50	0.66
E[190]	28 x 15	3.50 x 2.49	0.40
F	26 x 13	3.25 x 2.00	0.50
G	60 x 40	7.49 x 5.00	0.167
H[191]	21 x 13	2.61 x 2.36	0.423
∅[192]	15 x 12[193]	See command *^A∅* on page 18	

table 5 – Specifications of fonts at a resolution of 8 dots/mm (203 dpi)

[187] The fonts *B* and *H* only accept upper case letters and numerals, while all the others accept upper and lower case letters and numerals.
[188] Of the character alone. The height is determined by the upper case characters plus the footer of lower case characters such as *g* or *q*.
[189] Including the space between each one.
[190] OCR B.
[191] OCR A.
[192] In order not to confuse zero with the letter *o*, the convention of assigning ∅ the meaning of zero is adopted here (48_{10}, 30_{16}).
[193] Since the font is scalable, this is the default value.

Font[194]	Dimensions of matrix[195]. Height x width (in dots)	Dimensions of each character[196]. Height x width (in mm)	Number of characters per mm (in mm^{-1})
A	9 x 5	0.75 x 0.50	2.02
B	11 x 7	0.91 x 0.75	1.32
C, D	18 x 10	1.50 x 1.00	1.00
E[197]	42 x 20	1.75 x 1.08	0.92
F	26 x 13	2.16 x 1.34	0.74
G	60 x 40	5.00 x 4.00	0.25
H[198]	34 x 22	2.81 x 2.48	0.40
∅[199]	15 x 12[200]	See command ^A∅ on page 18	

table 6 - Specifications of fonts at a resolution of 12 dots/mm (300 dpi)

Font	Dimensions of matrix[201]. Height x width (in dots)	Space between each character (in dots)	Height of capital letters (in dots)
A	9 x 5	1	7
B	11 x 7	2	11
C, D	18 x 10	2	14
E	28 x 15	5	23
F	26 x 13	3	21
G	60 x 40	8	47
H	21 x 13	6	21
∅	15 x 12	Proportional[202]	3 x height / 4

table 7 – Space between characters and height of upper case characters

4.2 Bar-codes

It is possible to print bar-codes with the CZL language. The following specific types are available: Interleaved 2/5, Code 39, EAN 8, UPC E, Code 128, EAN 13, ANSI Codabar, MSI, UPC/EAN

[194] See note 187 on page 47.
[195] See note 188 on page 47.
[196] See note 189 on page 47.
[197] OCR B.
[198] OCR A.
[199] See note 192 on page 47
[200] See note 193 on page 47.
[201] See note 188 on page 47.
[202] In fonts *A – H* the space between each character is constant, while the spacing in font ∅ is proportional. This means that it is proportioned to the dimensions of the character; in other words, the space after the letter *i* is less than that after the letter *m*.

Extensions, UPC A, PostNet. Each of these is identified with a numeral or a letter that coincides with the parameter to be inserted in the *^Bx* command on page 18 (see table 8).
Each type of bar-code has its own characteristics: some are fixed length, others not, some accept all characters, and others only upper case letters or numerals.
The widths of the bars are defined according to set ratios. Some bar-codes require a fixed ratio to be read properly, while for others the ratio can vary from 2 to 3 without affecting the reliability of readings.

x	**Description**	**Ratio**	**ASCII characters allowed**[203]	**Length**	**Check digit**
2	Interleaved 2/5	2÷3	48÷57	Variable	Optional
3	Code 39	2÷3	32, 36, 37, 43, 45÷57, 65÷90	Variable	Optional
8	EAN 8	Fixed	48÷57	7	Yes
9	UPC E	Fixed	48÷57	10	Yes
C	Code 128	Fixed	32÷93, 95÷126	Variable	Optional
E	EAN 13	Fixed	48÷57	12	Yes
K	ANSI Codabar	2÷3	36, 42, 43, 45÷58, 65÷69, 78, 84	Variable	Optional
M	MSI	2÷3	48÷57	1÷14	Optional
S	UPC/EAN Extensions	Fixed	48÷57	2 or 5	No
U	UPC A	Fixed	48÷57	11	Optional
Z	PostNet	Fixed	48÷57	Variable	No

table 8 – Bar-code specifications

[203] See table 9 and table 10 on pages 59 and 60 respectively.

5. Examples

The following examples clarify the meaning of the commands described in previous sections.
Since all CZL commands are comprised of printable characters, it does not matter whether they are sent through a simple program (e.g. Basic) or with text file.
For a detailed description of each command, please refer to the preceding pages.

5.1 Example with alphanumeric strings, bar-codes, and boxes

The following example was developed in Basic, and its result appears in fig. 5 on page 52.

```
CONST SCRITTA1$ = "1234567890"
CONST SCRITTA2$ = "1234567890"
CONST SCRITTA3$ = "1234567890"
CONST SCRITTA4$ = "1234567890"
CONST SCRITTA5$ = "123456"
CONST CODE2OF5TXT$ = "12345678"
CONST EAN8TXT$ = "12345670"
CONST ANSITXT$ = "123"
CONST SCRITTA6$ = "CODE EAN-8"
CONST SCRITTA7$ = "CODE 2 OF 5"
CONST SCRITTA8$ = "ANSI CODABAR "
CONST SCRITTA9$ = "CZL Language"


'OPEN "lpt1:" FOR OUTPUT AS #1
OPEN "COM2:9600,n,8,1,cs5000" FOR OUTPUT AS #1


PRINT #1, "^XA"
PRINT #1, "^LH0,0"
PRINT #1, "^PRA"
PRINT #1, "^MD7"
PRINT #1, "^FWN"                    'SETTING OF ROTATION
PRINT #1, "^BY2"                    'WIDTH OF NARROW BAR BAR-CODE

PRINT #1, "^FO50,20"                'LARGE BOX
PRINT #1, "^GB720,480,4^FS"

PRINT #1, "^FO100,40"               'SMALL BOX
PRINT #1, "^GB320,210,2^FS"

PRINT #1, "^FO50,260"               'HORIZONTAL LINE
PRINT #1, "^GB720,0,3^FS"

PRINT #1, "^FO450,20"               'VERTICAL LINE
PRINT #1, "^GB0,480,2^FS"
PRINT #1, "^CFA"                     'SET FONT A
PRINT #1, "^FO120,70"                'SET COORDINATE
PRINT #1, "^FD" + SCRITTA1$ + "^FS"  'STRING

PRINT #1, "^CFB"                     'SET FONT B
PRINT #1, "^FO120,90"                'SET COORDINATE
PRINT #1, "^FD" + SCRITTA2$ + "^FS"  'STRING

PRINT #1, "^CFC"                     'SAME AS ABOVE
PRINT #1, "^FO120,110"
PRINT #1, "^FD" + SCRITTA3$ + "^FS"

PRINT #1, "^CFE"                     'SAME AS ABOVE
PRINT #1, "^FO120,130"
PRINT #1, "^FD" + SCRITTA4$ + "^FS"

PRINT #1, "^CFG"                     'SAME AS ABOVE
PRINT #1, "^FO120,170"
PRINT #1, "^FD" + SCRITTA5$ + "^FS"


PRINT #1, "^FT540,170"               'SET STARTING POINT
PRINT #1, "^B8N,100,Y,N,N"           'BAR-CODE
PRINT #1, "^FD" + EAN8TXT$ + "^FS"   'BAR-CODE CODE
PRINT #1, "^CFC"                     'SET FONT
PRINT #1, "^FO550,230"               'SET COORDINATE
PRINT #1, "^FD" + SCRITTA6$ + "^FS"  'STRING UNDER THE BAR-CODE

PRINT #1, "^FO130,280"               'SAME AS ABOVE
PRINT #1, "^B2R,150,Y,Y,Y"
PRINT #1, "^FD" + CODE2OF5TXT$ + "^FS"
PRINT #1, "^CFC"
PRINT #1, "^FWR"
PRINT #1, "^FO93,340"
PRINT #1, "^FD" + SCRITTA7$ + "^FS"

PRINT #1, "^FWN"
PRINT #1, "^FO590,280"                'SAME AS ABOVE
PRINT #1, "^BKR,,80,Y,N,*,T"
PRINT #1, "^FD"; ANSITXT$; "^FS"
PRINT #1, "^AAB,10,10"
PRINT #1, "^FO550,260"
PRINT #1, "^FD" + SCRITTA8$ + "^FS"

PRINT #1, "^FO450,425"                'SET COORDINATE
PRINT #1, "^GB320,0,2^FS"             'HORIZONTAL LINE
PRINT #1, "^CFC"                      'SET FONT
PRINT #1, "^FO460,440"                'SET COORDINATE
PRINT #1, "^FB300,2,10,C"
PRINT #1, "^FD" + SCRITTA9$ + "^FS"  'STRING"

PRINT #1, "^XZ"

CLOSE #1
```

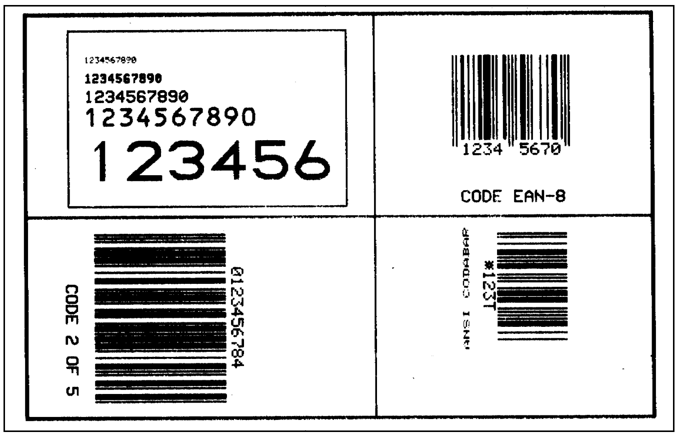

fig. 5 – Image of label with alphanumeric strings, bar-codes, and boxes

Exactly the same result is produced by sending the following text file to the printer:

```
^XA
^LH0,0
^PRA
^MD7
^FWN
^BY2
^FO50,20
^GB720,480,4^FS
^FO100,40
^GB320,210,2^FS
^FO50,260
^GB720,0,3^FS
^FO450,20
^GB0,480,2^FS
^CFA
^FO120,70
^FD1234567890^FS
^CFB
^FO120,90
^FD1234567890^FS
^CFC
^FO120,110
^FD1234567890^FS
^CFE
^FO120,130
^FD1234567890^FS
^CFG
^FO120,170
^FD123456^FS
^FT540,170
^B8N,100,Y,N,N
^FD12345670^FS
^CFC
^FO550,230
^FDCODE EAN-8^FS
^FO130,280
```

```
^B2R,150,Y,Y,Y
^FD12345678^FS
^CFC
^FWR
^FO93,340
^FDCODE 2 OF 5^FS
^FWN
^FO590,280
^BKR,,80,Y,N,*,T
^FD123^FS
^AAB,10,10
^FO550,260
^FDANSI CODABAR ^FS
^FO450,425
^GB320,0,2^FS
^CFC
^FO460,440
^FB300,2,10,C
^FDCZL Language printed by 6xxx
^XZ
```

5.2 Example with graphic images

Ever more applications require printing of logos, which in the simplest of cases is the company logo.
Two operations are performed to do so (see section 3.1.7.2 on page 30): first the logo is loaded onto the memory unit (RAM or flash memory), then it is recalled using its assigned name, and then it is positioned as desired.
The following example (in text format) produces the label in fig. 6.

```
~DGLOGO,01014,013,
0000000000000000000000000
0000000000000000000000000
0000000000180,
000000000003C0,
000000000003C0,
000000000003C0,
000000000007E0,
00000000000FE0,
00000000000FF0,
00000000003FFC,
00000000003FFC,
00000000007FFC,
00000000007FFE,
000000000FFFF,
000000000FFFF,
000000000FFFF,
0000000001FFFF80,
0000000007FFFFE0,
0000000007FFFFE0,
000000000FFFFFF0,
000000000FFFFFF0,
000000001FFFFFF0,
000000001FFFFFF8,
000000003FFFFFFC,
000000003FFFFFFC,
000000003FFFFFFC,
00000000FFFFFFFF,
00000001FFFFFFFF,
00000001FFFFFFFF80,
00000003FFFFFFFFC0,
00000003FFFFFFFFC0,
0000000FFFFFFFFFC0,
0000000FFFFFFFFFF0,
0000001FFFFFFFFFF8,
0000003FFFFFFFFFF8,
0000003FFFFFFFFFF8,
0000003FFFFFFFFFFC,
0000007FFFFFFFFFFE,
0000007FFFFFFFFFFE,
000001FFFFFFFFFFFF80,
000001FFFFFFFFFFFF80,
000003FFFFFFFFFFFFC0,
000003FFFFFFFFFFFFC0,
000007FFFFFFFFFFFFE0,
000007FFFFFFFFFFFFE0,
000007FFFFFFFFFFFFE0,
00000FFFFFFFFFFFFFF0,
00003FFFFFFFFFFFFFFC,
00003FFFFFFFFFFFFFFC,
00003FFFFFFFFFFFFFFC,
00007FFFFFFFFFFFFFFE,
0000FFFFFFFFFFFFFFFE,
0000FFFFFFFFFFFFFFFF,
0001FFFFFFFFFFFFFFFF80,
0001FFFFFFFFFFFFFFFF80,
0007FFFFFFFFFFFFFFFF80,
0007FFFFFFFFFFFFFFFFE0,
000FFFFFFFFFFFFFFFFFF0,
000FFFFFFFFFFFFFFFFFF0,
000FFFFFFFFFFFFFFFFFF0,
001FFFFFFFFFFFFFFFFFF8,
003FFFFFFFFFFFFFFFFFFC,
003FFFFFFFFFFFFFFFFFFC,
00FFFFFFFFFFFFFFFFFFFF,
00FFFFFFFFFFFFFFFFFFFF,
01FFFFFFFFFFFFFFFFFFFF,
01FFFFFFFFFFFFFFFFFFFF80,
03FFFFFFFFFFFFFFFFFFFFC0,
03FFFFFFFFFFFFFFFFFFFFC0,
03FFFFFFFFFFFFFFFFFFFFC0,
07FFFFFFFFFFFFFFFFFFFFE0,
1FFFFFFFFFFFFFFFFFFFFFF8,
1FFFFFFFFFFFFFFFFFFFFFF8,
3FFFFFFFFFFFFFFFFFFFFFFC,
3FFFFFFFFFFFFFFFFFFFFFFC,
7FFFFFFFFFFFFFFFFFFFFFFC,
7FFFFFFFFFFFFFFFFFFFFFFE,
00000000000000000000000000
^XA
^PRB^FS
^PF0^FS
^FO325,179^GB97,0,78,W^FS
^FO325,179^XGLOGO,1,1^FS
^PQ1,0,1,Y
^XZ
```

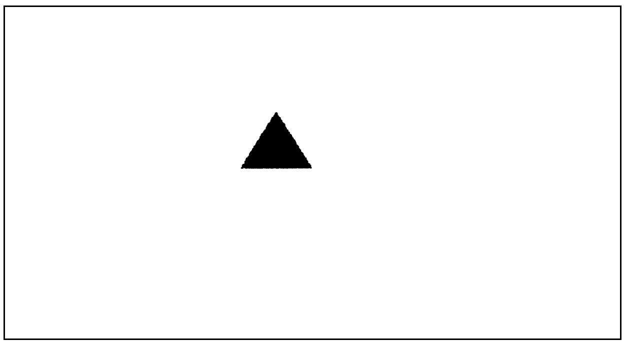

fig. 6 – Image of label with logo

5.3 Example of reverse

The reverse feature (see section 3.1.8.2 on page 35) is useful for highlighting parts of the label. The label in fig. 7 is created with the following file.

```
^XA
^LH33,33
^FXLocally test the reverse mode on one field
^FO0,0^GB250,400,200^FS
^FO30,10^AG^xxx^FR^FS
^FO30,200^B3,,100,,^FDABCDEFGHIJK^FR^FS
^FO30,350^AF^FDTest Local reverse command^FR^FS
^XZ
```

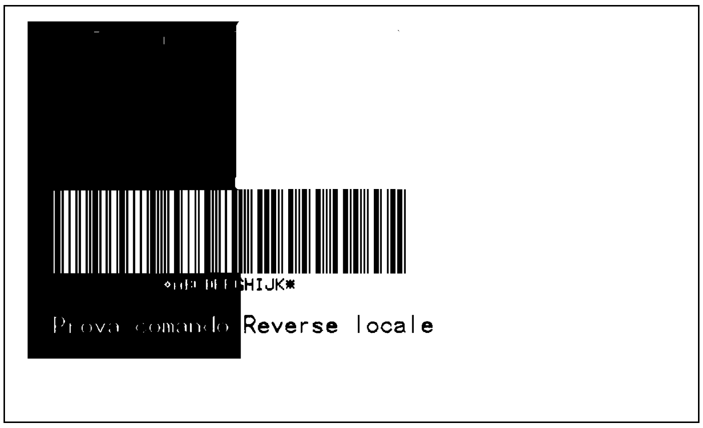

fig. 7 – Image of label with partial reverse effect

The reverse effect can be used on the entire label. The following file produces the label appearing in fig. 8.

```
^XA
^LH33,33
^LRY^FXSet reverse mode
^FO0,0^GB500,400,400^FS
^FO30,10^AG^FDxxx^FS
^FO30,200^B3,,100,,^FDABCDEFGHIJK^FS
^FO30,350^AF^FDTest Global reverse command^FS
^XZ
```

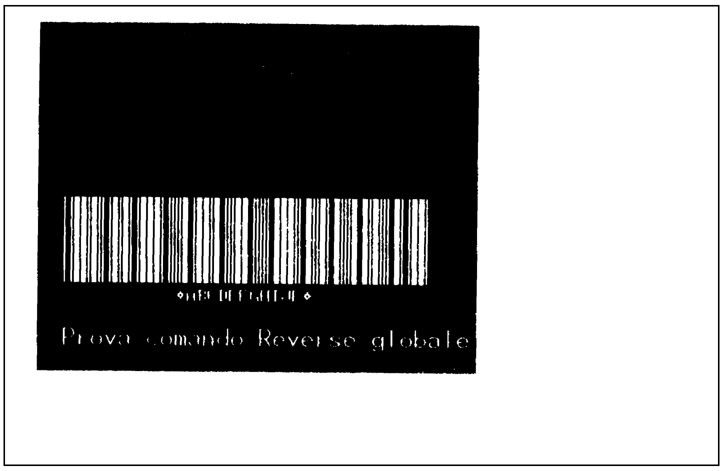

fig. 8 – Image of the label completely in reverse

5.4 Example with serial fields

When a certain number of labels must be produced in a progressive sequence, the CZL emulation supplies a command that can be used to automate the printing process (see section 3.1.8.3 on page 36).
The following file produces the label in fig. 9.

```
^XA
^LH33,33
^FXSerial field test
^FO30,10^AF^SNField n. NNN0001^FS
^FO30,200^B3,,100,,^SNABCDEFGHIJK3003^FS
^FO30,350^AF^FDSerial command test^FS
^PQ2
^XZ
```

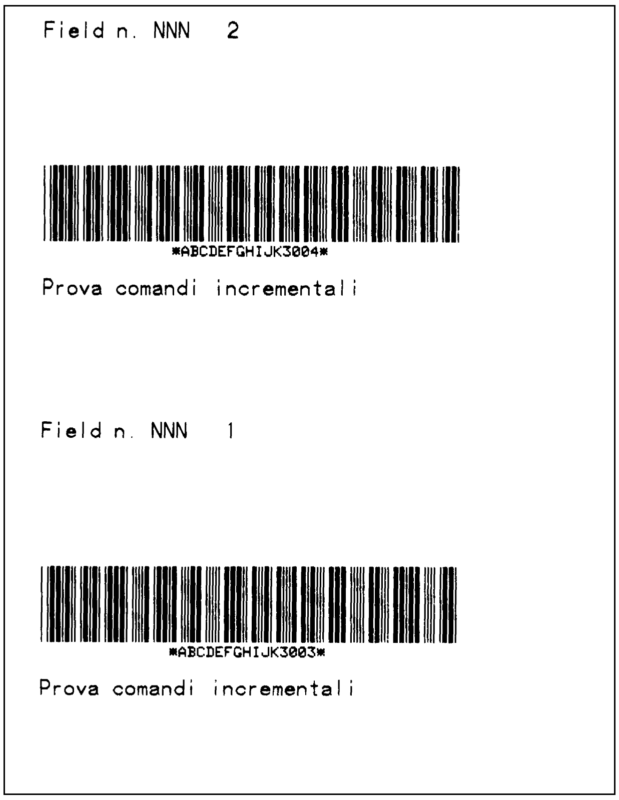

fig. 9 – Image of the label with serial fields

5.5 Example with variable fields

With the CZL emulation, it is possible to save the label to memory (RAM or flash) and recall it whenever it must be printed, updating only the parts that have changed (see 3.1.8.5 on page 37).
The following text file produces the label appearing in fig. 10.

```
^XA
^DFLABEL^FS
^LH33,33
^FXTest variable fields
^FO33,31^AG^FN1^FS
^FO33,133^B3,,50,,^FN3^FS
^XZ

^XA
^XFLABEL^FS
^FXRecall label and replace fields
^FDString 1^FN1^FS
```

```
^FDAAAAA^FN3^FS
^XZ

^XA
^XFLABEL^FS
^FXRecall label and replace fields
^FDString 2^FN1^FS
^FDBBBBB^FN3^FS
^XZ
```

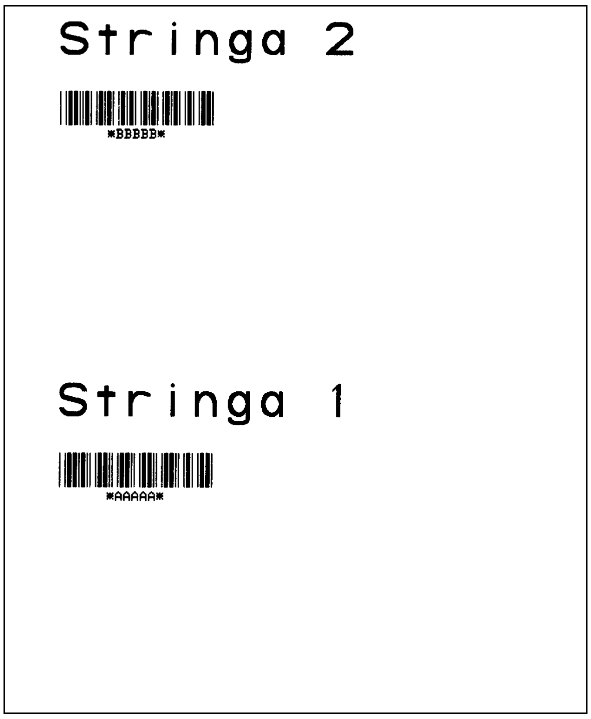

fig. 10 – Image of label with variable fields

6. ASCII TABLES

Char	*Dec*	*Hex*	*Char*	*Dec*	*Hex*	*Char*	*Dec*	*Hex*	*Char*	*Dec*	*Hex*
NULL	0	00		32	20	@	64	40	`	96	60
SOH	1	01	!	33	21	A	65	41	a	97	61
STX	2	02	Ò	34	22	B	66	42	b	98	62
EXT	3	03	#	35	23	C	67	43	c	99	63
EOT	4	04	$	36	24	D	68	44	d	100	64
ENQ	5	05	%	37	25	E	69	45	e	101	65
ACK	6	06	&	38	26	F	70	46	f	102	66
BEL	7	07	Ö	39	27	G	71	47	g	103	67
BS	8	08	(	40	28	H	72	48	h	104	68
HT	9	09	)	41	29	I	73	49	i	105	69
LF	10	0A	*	42	2A	J	74	4A	j	106	6A
VT	11	0B	+	43	2B	K	75	4B	k	107	6B
FF	12	0C	,	44	2C	L	76	4C	l	108	6C
CR	13	0D	-	45	2D	M	77	4D	m	109	6D
SO	14	0E	.	46	2E	N	78	4E	n	110	6E
SI	15	0F	/	47	2F	O	79	4F	o	111	6F
DLE	16	00	0	48	30	P	80	50	p	112	70
DC1	17	11	1	49	31	Q	81	51	q	113	71
DC2	18	12	2	50	32	R	82	52	r	114	72
DC3	19	13	3	51	33	S	83	53	s	115	73
DC4	20	14	4	52	34	T	84	54	t	116	74
NAK	21	15	5	53	35	U	85	55	u	117	75
SYN	22	16	6	54	36	V	86	56	v	118	76
ETB	23	17	7	55	37	W	87	57	w	119	77
CAN	24	18	8	56	38	X	88	58	x	120	78
EM	25	19	9	57	39	Y	89	59	y	121	79
SUB	26	1A	:	58	3A	Z	90	5A	z	122	7A
ESC	27	1B	;	59	3B	[	91	5B	{	123	7B
FS	28	1C	<	60	3C	\	92	5C	|	124	7C
GS	29	1D	=	61	3D	]	93	5D	}	125	7D
RS	30	1E	>	62	3E	^	94	5E	~	126	7E
US	31	1F	?	63	3F	_	95	5F		127	7F

table 9 – ASCII table (0÷127)

Char	*Dec*	*Hex*	*Char*	*Dec*	*Hex*	*Char*	*Dec*	*Hex*	*Char*	*Dec*	*Hex*
Ç	128	80	á	160	A0		192	C0	Ó	224	E0
ü	129	81	í	161	A1		193	C1	β	225	E1
é	130	82	ó	162	A2		194	C2	Ô	226	E2
â	131	83	ú	163	A3		195	C3	Ò	227	E3
ä	132	84	ñ	164	A4		196	C4	õ	228	E4
à	133	85	Ñ	165	A5		197	C5	Õ	229	E5
å	134	86	ª	166	A6	ã	198	C6	µ	230	E6
ç	135	87	º	167	A7	Ã	199	C7	þ	231	E7
ê	136	88	¿	168	A8		200	C8	Þ	232	E8
ë	137	89	®	169	A9		201	C9	Ú	233	E9
è	138	8A		170	AA		202	CA	Û	234	EA
ï	139	8B	½	171	AB		203	CB	Ù	235	EB
î	140	8C	¼	172	AC		204	CC	ý	236	EC
ì	141	8D	¡	173	AD		205	CD	Ý	237	ED
Ä	142	8E		174	AE		206	CE		238	EE
Å	143	8F	¯	175	AF		207	CF		239	EF
É	144	90		176	B0	ð	208	D0		240	F0
æ	145	91		177	B1	Ð	209	D1	±	241	F1
Æ	146	92	²	178	B2	Ê	210	D2		242	F2
ô	147	93	³	179	B3	Ë	211	D3	¾	243	F3
ö	148	94	´	180	B4	È	212	D4		244	F4
ò	149	95	Á	181	B5	ı	213	D5		245	F5
û	150	96	Â	182	B6	Í	214	D6	÷	246	F6
ù	151	97	À	183	B7	Î	215	D7	¸	247	F7
ÿ	152	98	©	184	B8	Ï	216	D8	°	248	F8
Ö	153	99	¹	185	B9		217	D9	¨	249	F9
Ü	154	9A		186	BA		218	DA	·	250	FA
ø	155	9B	»	187	BB		219	DB		251	FB
£	156	9C		188	BC		220	DC		252	FC
Ø	157	9D	¢	189	BD		221	DD		253	FD
×	158	9E	¥	190	BE	Ì	222	DE		254	FE
ƒ	159	9F		191	BF		223	DF		255	FF

table 10 – ASCII table (128÷255)

7. List of figures

8. LIST OF TABLES