GENERAL CODE®

SISTEMI DI CODIFICA

BARCODE PRINTERS

ALFA 2050 / 2100

PROGRAMMING MANUAL

Rev. 2.15

GENERAL CODE Thermal Transfer Printers

ALFA 2050 / 2100

Programming Manual

Issue 6, revised March 1995.

Firmware version MA62 V2.13 and above.

GENERAL CODE
Via P.S. Mattarella, 69
30037 GARDIGIANO (VE) - ITALY

Tel: 041-449.888 International: +39-041-449.888
Fax: 041-449.730 International: +39-041-449.730

E-Mail: info@generalcode.it
Internet: http://www.generalcode.it

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

Table of Contents

1 Introduction

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

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

The printer offers the following features :-

- EAN-8, EAN-13, EAN 2 & 5 digit addendums, UPC-A, UPC-E, EAN-128, Code-128 (subsets A, B & C), ITF-14, Interleaved 2 of 5, Code39 & Codabar barcodes.
- High density LEB code.
- USD-5 Dotcode (Optional).
- Bitmap fonts, 6 - 40 point.
- Bitstream® Speedo™ font-scaling technology, with 4 Speedo fonts.
- Line & box graphics with grey shading.
- Bit image graphics.
- Rotations of 0°, 90°, 180° or 270° for all fields individually.
- Variable & incrementing fields.

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

2. Programming Guide

Command Description Conventions

Various typefaces and styles are used in the summary descriptions of commands.

Characters in **bold** type are fixed command descriptors. Of these, characters in angle brackets eg. **<Esc>** are control characters, identified by their common ascii names. The decimal and hexadecimal equivalents are given in the 'Control Codes' section of the programming reference.

Characters in *italics* are data values used by the command. This may be a length, or a text message or barcode data. Where the data is not of a fixed length this is indicated by ellipsis, eg. *ddd...dd*. Optional items are indicated by placing them in square brackets, eg. *[+ii]*.

The name shown to the right of the description is the command name, used for reference purposes only. Spaces are often used to separate the various components of the command in the descriptions, but this is purely for clarity. The spaces are not to be sent as part of the command. Where a literal space is required, this is indicated by the character.

2.1 Label and Continuous Modes

The printer will operate in two modes depending on the stationery type.

In label mode, the stationery is supplied divided into fixed lengths, either as labels on a roll, or continuous perforated paper with index marks. The printer measures the distance between index marks to determine the formlength, and always aligns with respect to the top of the label. Label gaps and index marks may be up to 25 mm in length. If a gap or index mark exceeds this length, it will be interpreted as a PaperOut fault and printing will stop. The label length must lie in the range 5 mm to 600 mm.

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

The power-on default is determined by the DIP switch settings, but can also be changed under software control.

<DC4> **SetLabelMode**
Sets the printer to label mode. If not already in label mode, the label will be measured.

<DC2> **SetContinuous**
Sets the printer to continuous mode.

<Esc> A *llll* **SetFormLength**
llll = form length (mm)
Sets the form length when in continuous mode. This command has no effect in label mode, as the label length is measured automatically. Default length in continuous mode is 15 mm.

<Esc> P *yyyy* **SetPageOffset**

yyyy = page offset (mm)

Sets the offset of the top-of-form from the physical top edge of the label. The default is 0 mm which places the top of the printed area at the top edge of the label. Setting a positive offset means that the labels are synchronised to project further from the print head so that the print starts further down on the label. This may be needed with perforated labels in order to feed the perforations out past the print head.

2.2 Coordinate System & Measurements

Printing is page oriented. Text, barcodes, graphics, etc are placed on the page using an (x,y) coordinate system. Looking out from the print head in the direction of paper motion, x = 0 is the left hand edge of the print head and y = 0 is the top-of-form. Most coordinates are in pixels by default, and other measurements, e.g. formlength, are in mm. The units of measurement can be set globally to be in pixels, mm or points (1/72 inch) using the SetUnits command. In general it is easier to set the units to mm, which not only makes distances easier to visualise, but also makes the programming independent of print head resolution. Some commands, eg. graphics must inherently be described in pixels, and these are not changed by the SetUnits command.

<Esc> Z *u* **SetUnits**

u = M	Millimetres
u = 2	Half-millimetres
u = P	Points (1/72")
u = D	Dots/Pixels
u = O	Original (default) units

Sets the global units of measurement. This affects all following commands that include measurement or coordinate data.

2.3 Field Rotation

Almost all fields can be rotated North, East, West or South. (Some field types cannot be rotated at present, but may become rotatable in future versions, so the rotation should always be set appropriately).

<Esc> V *r* **SetRotation**

r = 1	0° (North)
r = 2	90° (East)
r = 3	180° (South)
r = 4	270° (West)

Sets the rotation of the fields which follow. The default rotation is 1. Field coordinates refer to the top left corner of the field as viewed from within the field.

2.4 Placing Text Fields

The printer is supplied with 9 pre-scaled bitmap fonts and 4 Bitstream® Speedo™ scalable typefaces built in. The bitmap fonts is similar to Helvetica® typeface and are pre-scaled at point sizes of 6, 8, 10, 12, 16, 20, 24, 32 and 40. Because they are pre-scaled, they can be placed on the page more quickly, but the point-sizes are limited and only one typeface can be used. The printer also incorporates the Bitstream® 4-in-1 Processor, V3.1, which provides independent scaling in vertical & horizontal directions (3 - 999 Points), italics and different kerning levels. The scaling process is relatively slow compared to character placement, although a cache can be set up to avoid repeated scaling of the characters. At larger point sizes (> 32 point) cached scalable characters can be placed faster than bitmap characters (see the later section on buffers and caches). Either type of text can be rotated with the RotateField command and bitmap text can be magnified with the SetMagnification command.

<Esc> T *xxxx yyyy ddd...dd* **<Eot>** **PlaceText**

xxxx = x coordinate
yyyy = y coordinate
ddd...dd = text message

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

<Esc> F *n c* **SelectFont**

n = Font number (0-9)
c = Country code.

Fonts 1-9 are bitmapped fonts. Font 0 selects scalable fonts which are set up using the SetScalableFont command. The country code selects the ascii code mapping. The default font and country code are '1 W'.

Font number	Point size
0	Scalable fonts
1	6 point
2	8 point
3	10 point
4	12 point
5	16 point
6	20 point
7	24 point
8	32 point
9	40 point (6 dot/mm only)

Country Code	Name	Code Page
W	Windows ANSI	---
E	English	437
M	Multilingual	850
S	Slavic	852
P	Portugese	860
C	Canadian	863
F	French	863
N	Nordic	865

<Esc> Y *ff vvv hhh i k* **SetScalableFont**

ff = Font number
vvv = Vertical Point Size (3 - 999)
hhh = Horizontal Point Size (3 - 999)
i = Italicize (0 or 1)
k = Kerning level (0 - 3)

Sets the scalable font parameters. Horizontal and vertical point sizes may be set independantly. The text may be italicized (i=1), and the kerning level may be selected. A kerning level of 0 represents no kerning, and a kerning level of 3 represents the tightest available kerning.

Scalable Font Number	Speedo™ Typeface	Similar Typefaces	Print Sample
00	Swiss™ 721	Helvetica®/Arial©	
01	Swiss™ 721 bold	Helvetica/Arial bold	
02	News 701	Ionic No 5™	
03	Impress		

<Esc> Z *b* **SetBaseline**

b = B	Place on baseline
b = T	Place on topline

This command controls how characters are placed with respect to the placement point. The topline refers to a point at the top of the tallest character in the font. The baseline refers to the bottom of the capital letters. The latter is generally a more convenient reference point, and is more consistent between typefaces, but for backward compatibility reasons, the default is Topline.

<Esc> U *www* **SetTabWidth**

www = Tab width (mm)

Sets the width of embedded tabs for the PlaceText command.

<Esc> W *l c* **SetCharGaps**

l = Inter-Line Gap (0 - 9)
c = Inter-Character Gap (0 - 9)

Increases the gaps between characters & between lines by 10% of the point size for each incremental value set. E.g. EscW24 increases the inter-character gap by 20% of the point size and the inter-line gap by 40% of the point size, both with respect to the default gap sizes. This is independant of scalable font kerning and is mainly to be used with bitmap fonts. Default settings are ’0 0’.

<Esc> M *vv hh* **SetMagnification**

vv = Vertical factor (1-15)
hh = Horizontal factor (1-15)

Sets the magnification factors for logos and bitmap fonts (not scalable fonts). Vertical and horizontal are with respect to a non-rotated field. The magnification factors rotate with the magnified object, so e.g. magnification 0201 will produce tall thin characters in all rotations.
Default magnification is ’01 01’.

2.5 Placing Barcodes

The printer supports the most commonly used barcode types directly as well as USD-5 dotcodes. Barcode modules are always scaled in dots (necessary for precise widths), but the module widths can be set up as required. The general format for barcode placement is as follows :-

<Esc> B *xxxx yyyy t hh ddd...dd* **PlaceBarcode**

xxxx,yyyy = Coordinates (pixels)
t = Barcode type
hh = Height (mm)
ddd...dd = barcode data

Barcode type	Description
1	ITF-14
2	EAN-13
3	EAN-8
4	Interleaved 2 of 5 (ITF)
5	Code-39
6	Code-39 (without checksum)
7	Codabar
8	EAN-128
9	Code-128
A	UPC-A
B	UPC-E

<Esc> B *xxxx yyyy* **1** *hh dddddddddddd* **PlaceITF14**

Places an ITF-14 barcode. Exactly 12 data digits must be sent. The leading zero and checksum are added automatically. If the height is specified as 00, then the barcode height is set to the correct value for the current magnification.

<Esc> B *xxxx yyyy* **2** *hh [a] dddddddddddd [dd..]* **PlaceEAN13**

For a standard EAN-13 code, exactly 12 data digits must be sent. The checksum is added automatically. If the height is specified as 00, then the barcode height is set to the correct value for the current magnification. 2 and 5 digit addendums can also be added by sending ’T’ for a 2 digit or ’F’ for a five digit addendum, before the main data digits. The addendum data should then follow immediately after the main data.

<Esc> B *xxxx yyyy* **3** *hh ddddddd* **PlaceEAN8**

Exactly 8 data digits must be sent. The checksum is added automatically. If the height is specified as 00, then the barcode height is set to the correct value for the current magnification.

<Esc> B *xxxx yyyy* **4** *hh ddd...dd* **q** **PlaceInt2of5**

Up to 50 digits may be sent, terminated with ’q’. The barcode height must be supplied.

<Esc> B *xxxx yyyy* **5** *hh ddd...dd* **q** **PlaceCode39C**

Up to 50 data characters may be sent, terminated with ’q’. The character set consists of all the uppercase letters, numbers and also space, stop, asterisk and dash. The checksum is calculated and added automatically.

<Esc> B *xxxx yyyy* **6** *hh ddd...dd* **q** **PlaceCode39**

Up to 50 data characters may be sent, using the character set described above, and terminated with ’q’. No checksum is appended.

<Esc> B *xxxx yyyy* **7** *hh c ddd...dd* **q** **PlaceCodabar**

Up to 50 data characters may be sent, terminated with ’q’. The character set consists of all the numbers and also dash, dollar, colon, oblique, stop, plus , and asterisk. The start and stop codes can be selected using the optional format character.

Format character	Start code	Stop code
@	a	a
A (default)	a	b
B	a	c
C	a	d
D	b	a
E	b	b
F	b	c
G	b	d
H	c	a
I	c	b
J	c	c
K	c	d
L	d	a
M	d	b
N	d	c
O	d	d

<Esc> B *xxxx yyyy* **8** *hh ddd...dd* **<Eot>** **PlaceEAN128**

Up to 50 data characters may be sent. The character set consists of the ascii range 32-127. Parentheses will be retained in the human-readable text, but stripped out of the barcode data, and may thus be used to mark the article identifiers in the human readable text. Any variable length field should be terminated with the GS character (ascii 29). This will not appear in the human readable text, but will be translated into FUNC1 in the barcode. If the height is specified as 00, then the barcode height is set to the correct value for the current magnification.

<Esc> B *xxxx yyyy* **9** *hh nn ddd...dd* **<Eot>** **PlaceCode128**

Up to 50 data characters may be sent. The character set consists of the ascii range 0-127. Codesets A, B and C are used to produce the optimum barcode density. The number of data bytes can normally be set to 00 and the barcode data terminated with <Eot>. If the <Eot> character itself is required in the barcode data, then the number of data bytes in the barcode must be specified, and <Eot> will be encoded, rather than terminating the data.

<Esc> B *xxxx yyyy* **A** *hh ddddddddddd* **PlaceUPCA**

Exactly 11 data digits must be sent. The checksum is added automatically. If the height is specified as 00, then the barcode height is set to the correct value for the current magnification.

<Esc> B *xxxx yyyy* **B** *hh ddddddddddd* **PlaceUPCE**

Exactly 11 data digits must be sent. The data is converted to UPC-E format by zero suppression, and the checksum is added automatically. If the height is specified as 00, then the barcode height is set to the correct value for the current magnification.

<Esc> N *t m w n* **SetBarcodeModules**

t = Barcode type (See PlaceBarcode)
m = Module magnification (1-9)
w = Wide module width (1-9)
n = Narrow module width (1-9)

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

<STX> **EnableBarcodeText**
Enables automatic placement of human readable text under the barcode. This option is on by default. For some barcode types the size and position of the text is pre-defined by the barcode specification, eg. EAN-13. For others an optimum text size is selected and the text is centred under the barcode.

<ETX> **DisableBarcodeText**
Disables placement of human readable text.

<Esc> . *xxxx yyyy tt ss ddd...dd* **PlaceDotCode**

xxxx,yyyy = Coordinates
tt = type (05, 07, 09, 11, 14, 17, 20)
ss = Dot spacing (pixels)
ddd...dd = dotcode data

Places a USD-5 dotcode. The dot spacing is measured from centre to centre of the dots. If the spacing is set to 00, the standard 4 mm spacing is used. The number of digits must match the dot code type, e.g. type 17 requires 17 digits. The checksum is calculated & added automatically.

<Esc> : *xxxx yyyy* **L** *ss ff c ddd...dd* **<Eot>** **PlaceLEBCode**

xxxx,yyyy = Coordinates
ss = module size (1-99 dots)
ff = No of fields (01-11)
c = No of codes (1-3)
ddd...dd = field data

The number of fields and codes must be specified - these give the dimensions of the code. If the data stream is too short, it will be padded with spaces, if too long, the excess data will be discarded. Invalid characters are converted to spaces. By default, data is encoded as Type1 with parity checking. Type2 encoding can be selected by inserting an 'h' character in the data stream, or Type1 by inserting a 'n' character. Thus the encoding type can be switched at will. If the current field is not full when a type switch occurs, it will be padded with spaces and the next field will be of the new type.

The character set includes 5 foreign characters. These can be used directly in the data stream, provided the correct code page is selected. Alternatively, they can be represented by standard ascii characters as follows.
@ = É,   [= Ä,   \ = Ö   ] = Å   ^ = Ü

Note that the module size is always given in dots, regardless of the current units of measurement. This follows the same convention as barcode module sizes and is necessary because of the small dimensions normally used.

2.6 Block Fill, Box & Line Drawing

The BlockFill command can be used to place a block of black, white or grey, or to colour existing fields, e.g. invert text. Basic line drawing can be obtained by specifying a narrow height or width. Box drawing is obtained either with multiple lines, or by overlaying a black box with a slightly smaller white box. Block fills are not affected by rotation, so width is always in the x direction and height in the y direction. (x,y) marks the top left corner.

<Esc> I *xxxx yyyy wwww hhhh c* **BlockFill**

xxxx,yyyy = Coordinates
wwww = Width
hhhh = Height
Colour c

The BlockFill command fills the specified area according to the colour code given. Some of the colour codes can be used to produce reverse image or coloured text, or to draw dotted lines.

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

2.7 Placing Graphics

The printer supports 2 types of graphic images. Logos are small bitmap images which are stored in memory and can be placed on the page at will. Graphics fields transfer the image directly onto the page without storing it in memory. Logos can be magnified and rotated, and can be placed as often as required once downloaded. However, they take up space in the printer memory, reducing the area available for the main image. Graphics fields do not take up any additional space, but must be downloaded each time they are used, which can be extremely slow. Graphics fields cannot be magnified or rotated at present.

In both cases, the data defines a bitmapped image. 1 byte = 8 dots described horizontally. The most significant bit of the first byte appears on the left hand side and consecutive bytes work across to the right until the described width is reached. Any unused bits in the last byte of the line are discarded and the next line begins with the next byte. This continues until the required height and width have been covered. The command finishes when the correct number of bytes have been received. Note that if a graphics command has been sent and insufficient data has been transmitted, any further commands are then ’absorbed’ by the graphics command which is still waiting for data. Thus the commands are not executed, and the printer appears to have crashed. Count the bits & bytes carefully! Note that the width and height of graphics images are always specified in pixels. These values are **not** affected by the SetUnits command.

Graphics command require an 8 bit communications interface to transfer correctly. This can be via the Centronics port, or via the serial port set up for 8 data bits or in printable character mode with 7 data bits.

<Esc> G *xxxx yyyy wwww hhhh ddd...dd* **PlaceGraphics**

xxxx,yyyy = Coordinates
wwww = Width in dots
hhhh = Height in dots
ddd...dd = graphics data

Places the graphic data directly into the image at x,y.

<Esc> K *n tttttttttt wwww hhhh ddd...dd* **DefineLogo**

n = Logo number (0-9)
ttt...tt = Title (Exactly 10 characters)
wwww = Width in dots
hhhh = Height in dots
ddd...dd = logo data

Defines a logo, storing the graphic data in memory. The logo must be placed on the page using the PlaceLogo comand.

<Esc> L *xxxx yyyy n* **PlaceLogo**

xxxx,yyyy = Coordinates
n = Logo number (0-9)

Places a previously defined logo. The logo will be magnified and rotated according to the current magnification and rotation settings.

<Esc> X *n* **DeleteLogo**

n = Logo number n (0 - 9)

Deletes a logo which has previously been downloaded. The command must be sent twice.

2.8 Stored Formats

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

Stored formats take up a small amount of printer memory, but this does not normally reduce the print area significantly. When a format is stored or deleted, the current image will be lost as memory is reallocated.

<Esc> D *n tttttttttt ddd...dd* **<VT>** **DefineFormat**

n = Format number n
ttt...tt = Title (Exactly 10 characters)
ddd...dd = Commands

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

<Esc> J *n* **LoadFormat**

n = Format number

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

<Esc> j *n rrrr* **RepeatLoadFormat**

n = Format number
rrrr = Repeat count

This command executes a stored format repeatedly. This is useful for printing batches of labels with incrementing fields (See variable fields section). Note that repeat printing of a stored format cannot be cancelled with the AbortPrint command as all commands are taken from the stored format until the command terminates. The InlineAbort command should be used instead if required.

<Esc> E *n* **EraseFormat**

n = Format number

Delete format number n from memory. This command must be sent twice.

2.9 Printing Commands

<FF> **PrintFeed**

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

<Esc> R *nnn* **RepeatPrint**

Repeat nnn times

This command produces one or more copies of the current image. If the image has just been printed using PrintFeed or RepeatPrint, then further copies of the same image are produced. If the count is set to 0000, then the printer repeats continuously.

<Esc> r *nnnn* **RepeatPrint4**

Repeat nnnn times

This command is identical to RepeatPrint, except that the maximum count is 9999.

<Esc> a **AbortPrint**

Stops a repeat print run. This command must be issued after a repeat command but before other commands if it is to be effective. If another label definition is sent first, then the AbortPrint command will be held up in the input buffer until the image buffer is free, ie. until all the repeats are finished, and so will have no useful effect.

<Esc> m *nnn* **SetSpeed**

nnn speed (20 - 999)

Sets the print speed. The print speed affects the print quality. See the later section on print quality. The default speed is 55 mm/s. Each printer type has a different maximum speed limit, depending on the print width and resolution. Attempting to set too high a speed sets it to the maximum value.

<Esc> S *nn* **SetSpeed2**

nn speed (40 - 139)

This is an older form of the SetSpeed command, retained for compatibility. The speed is input as 2 digits. Values in excess of 99 mm/s are to be input without the leading 1.

<Esc> h *nnn* **SetHeatLevel**

Level nnn (50 - 200)

The level represents the percentage of the factory preset level. Different media may require different heat levels to obtain the best print quality (see also section 3). The maximum heat setting allowed is 150% in direct thermal mode and 200% in thermal transfer mode. Attempting to set too high a heat value sets it to the maximum value allowed.

2.10 Variable Fields

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

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

Each variable fields can optionally be given an increment value when it is defined. There is a command to update all incrementing variables, which allows incrementing serial numbers to be added to labels automatically.

The normal sequence of events when programming will be :-

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

2.10.1 Defining Fields

<Esc> f d *r ww +ii /uu* **<Eot>** **DefineField**

r = field reference, 0-9 or A-Z.
ww = maximum width of field.
+ii = increment value (optional)
/uu = update frequency (optional)

This command defines a variable or incrementing field which can be used in text or barcode commands at a later stage. The reference letter may be 0-9 or A-Z, allowing up to 36 variables. ww defines the maximum number of characters that will be used in the field. If the field is to be incremented or decremented automatically, then an increment value must be supplied with +ii or -ii (for decrement). If the field is to be updated only every *n* labels, then an update frequency must be supplied with /uu. The <Eot> character ends the definition.

<Esc> v *r ddd...ddd* **<Eot>** **SetFieldValue**

r = field reference
ddd...dd = field value terminated with <Eot>

This command enters a new value for a variable.

<Esc> f l **ListFields**

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

<Esc> f u **UpdateFields**

Updates all incrementing fields, according to their field definitions. An incrementing field will normally be a pure decimal number, but can include any characters. The rules for increment are as follows :-

0 - 9:	0 + 1 = 1, etc. 9 + 1 = 0, carry 1.
A - Z:	A + 1 = B, etc. Z + 1 = A, carry 1.
a - z	a + 1 = b, etc. z + 1 = a, carry 1.
Others:	Skipped over.

Carries from the most significant digit are discarded.

Examples:-
128 + 3 = 131
1A8 + 3 = 1B1
9Z9 + 1 = 1000
1.6 + 25 = 4.1
1*A,8$ + 99 = 1*K,7$

<Esc> f s *DDMMYYhhmmss* **SetRealTimeClock**

DD = day of month (1-31)
MM = month (1-12)
YY = year (80-99 = 1980-1999, 00-79 = 2000-2079)
hhmmss = hours/minutes/seconds (24-hour clock)

Sets the printer’s real-time clock and updates the date/time field. The real-time clock in the printer is software generated from the CPU clock frequency. It is not as accurate as a dedicated clock function over a long time period, but should be accurate enough for time-stamping labels. It is advisable to set the RTC before starting each run of labels.

<Esc> f t **UpdateTimeField**

Field ’t’ is reserved for the date & time field. This field is updated from the real-time clock using this command. The clock is not used directly since it is possible for the clock to change within the formatting time for a label. This could give inconsistent results if the time appeared more than once in a label, e.g. in text and a barcode. Remember to update the field every label if a fresh time stamp is required for each one.

<Esc> f x **ClearFields**

This command clears all the variable fields, freeing the space to start again. This will not normally be necessary as each definition of a variable overwrites the previous one. The ClearBuffers command also clears all variable definitions.

2.10.2 Accessing Fields

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

<Soh> *r format* **<Eot>** **GetVariableFIeld**

r = field reference

This sequence can appear within text or barcode data and provides access to the variable fields, e.g. <Esc> T 0010 0010 Hello <Soh> 1 <Eot>, how are you? <Eot>
inserts variable field 1 into the text. With no formatting information, the variable value appears as is, and takes the minimum width necessary. This will normally be OK for text, but barcodes generally need a particular width. The formatting code can be used to specify a minimum field width, and the padding required. It also allows a decimal point to be inserted in a number. e.g. a price of £3.99 may appear as 3.99 in text and 00399 in a barcode.

If the first format character is '<' the field is left-justified and padded on the right with spaces. If the first character is '>', the field is right justified, padded on the left with spaces. If the first character is '=', the field is padded with leading zeros. If any of these characters are used, they should be followed by a field width, given as a decimal number. Note that this is a minimum width and if the variable exceeds the width specified in the format, it will expand as needed.

If a '.' or ',' character is found in the format information, this specifies a decimal point. It should be followed with a decimal number representing the number of decimal places to use.

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

Escape sequence	Expands to :-
<Soh>1<Eot>	"258"
<Soh>1>5<Eot>	" 258"
<Soh>1<5<Eot>	"258 "
<Soh>1=5<Eot>	"00258"
<Soh>1=10<Eot>	"0000000258"
<Soh>1=3.2<Eot>	"2.58"

<Soh> t *+d..d format* **<Eot>** **GetTimeField.**

+d..d[-] = days offset.

This command is used in the same way as the variable field command, but the information is taken from the date/time field. The format information is also different.
The offset may be used for printing expiry or best-before dates. If followed immediately by a '-' sign, it becomes a negative offset, i.e. back-dated. In the format specifier, the various parts of the date and time are represented by the letters DMYhms for day, month, year, hours, minutes, seconds. Also, 'a' and 'A' can be used to produce an AM/PM indicator. 'a' produces 'a' or 'p', 'A' produces 'A' or 'P'. By default, all fields are represented as 2 digits. This can be modified by preceding the letter with a '#' which has the following effect:- #D omits the leading zero from days 1-9. #M omits the leading zero before months 1-9. #Y gives a 4 digit year. #h gives a 12-hour clock time. Any other characters are passed through unchanged and may be used as separators in the output. For examples :

If the date/time field is 16:23:40 on 18th August 1994 :-
<Soh>t#D-#M-#Y #h:m:s A<Eot> Expands to "18-8-1994 4:23:40 P"
<Soh>t+60D-M-Y<Eot> Expands to "17-10-94"

2.11 Printer Memory Use - Buffers and Caches

2.11.1 Image Buffers

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

By default, there is a single image buffer which occupies the whole of the printer's free memory area, giving the maximum image size. However, once an image is complete, no further formatting can take place until the image is printed and the buffer cleared. To speed up the print formatting, a second buffer can be allocated. This is referred to as DoubleBuffer mode. In this case, a second label can be formatted whilst the first is printing. This allows non-stop printing in a large number of cases. The penalty is that the effective image area is halved as the memory is divided between the two buffers.

When formatting complex labels where the majority of the label is invariant and a few fields change, there are two methods of saving formatting time. The first is to copy previous image and blank out the areas which are changing. This is relatively simple if the variant data is a fixed size, but in some cases it is difficult to predict how much blanking is required. An alternative method is to allocate a BaseImage buffer. The fixed fields are formatted first and the image frozen in the base image buffer. For each subsequent label, the base image is copied and the variant fields added on top. This method is easier and more foolproof than the blanking method, but as with double buffering, requires more memory. Unless the labels are very small, or extra memory is fitted, double buffering and base image buffering will not normally be used together as the printer has only one third of its memory available for the image.

<RS> **SetDoubleBuffer**
Enables double buffering of the image.

<US> **SetSingleBuffer**
Disables double buffering. This command does not disable the base image buffer.

<Esc> i *n* **SetBaseImage**

n = 0 Disabled
n = 1 Enabled

Enables or disables the base image buffer. This command does not affect double/single buffer mode.

<SO> **CopyCompileToBase**
When the base image buffer is enabled this command copies the current compiled image into the base image buffer. Has no effect when the base image buffer is disabled.

<DLE> **CopyBaseToCompile**
When the base image buffer is enabled, this command copies the base image into the compile buffer (current image). When the base image buffer is disabled, it copies the previous printed image into the compile buffer. (This would appear to be a futile operation in single buffer mode - in fact it simply prevents the previous image from being cleared).

<EM> **ClearCompileBuffer**

Clears the current image. The image is normally cleared automatically so this command is only useful for cancelling a partially formed image if a mistake is made.

<CAN> **ClearAllBuffers**
Clears all the image buffers and deletes all logos, stored formats and variable fields. Use with care!

2.11.2 Font Cacheing

Scalable fonts, for all their many features and benefits, do incur a speed penalty in the scaling process. The effect of this can be minimised by cacheing the characters as they are scaled, so that when the same character is used again (in the same point size & rotation), it can be retrieved directly from the cache in a fraction of the time required to regenerate it from scratch. As with multiple image buffers, cacheing consumes memory and reduces the image size, although in general a large cache is not required as most labels only use a limited number of characters.

<Esc> c a *nnnn* **AllocateFontCache**
Cache size nnnn Kbytes
This command creates a cache area for scalable fonts to speed up text placement. The current image is cleared as printer memory is reallocated. No cache is allocated by default. As a rule of thumb, a 20K cache works in most cases, but some experimentation may be required. The Status command can be used to obtain information on cache useage.

<Esc> c d **DisableFontCache**
Disables storing of characters into the cache. The cache will still be searched when characters are placed, but no new characters will be added to the cache. This command may be used to prevent the cache being filled with infrequently used characters, or very large characters.

<Esc> c e **EnableFontCache**
Enables storing of characters into the cache. Caching is enabled automatically when a cache is allocated.

<Esc> c f **FlushFontCache**
Clears all stored characters from the cache.

<Esc> c r **SetCacheReuse**
When the cache becomes full, and further cache space is required, the least-recently used characters will be ejected from the cache to make room for new data. This option should only be used if there is sufficient cache allocated to cover all the characters in one label. If the printer is fitted with additional memory, then a large cache can be allocated at startup, and should require no further attention. If the cache is too small to accomodate all the characters on a repeated label, then performance can become worse than with no cache. When the cache runs out, the earliest characters are ejected. When the next label is printed, using the same character set, the earliest ones are no longer within the cache. The cache reuse process restores these characters, but in the process ejects others which will also be needed, so that the number of cache hits is drastically reduced.

<Esc c n **SetNoCacheReuse**
When the cache becomes full, no further characters are added until the cache is flushed. This is the default mode of operation.

2.11.3 Input Buffer Size

The default size of the input buffer is 2K bytes. If required, a larger (or smaller) buffer size can be set up, e.g. when printing a batch of labels with variable data. Note that the image buffer uses the general memory pool, so specifying a large buffer will reduce the image area available.

<Esc> b S *nn* **ResizeInputBuffer**

Buffer size nn Kbytes

Resizes the input buffer to the size requested, provided that the new size is large enough to contain the data already queued in the buffer. This should ideally take place once at power-on, as changing the buffer size will clear the image buffers along with any stored logos and formats.

2.11.4 Saving Image Space

In a few iolated cases, large labels are required which have a small print area at each end. The image buffer may not be large enough to hold the entire label, but if the unused space in the middle is of a known fixed size, the printer can be set up to split the image area around it. Other (rare) cases involve producing a duplicate copy of the image on a single perforated label. A duplicate image can be set up if required.

<Esc> O *yyyy llll* **SetForwardFeed**

yyyy = Start coordinate
llll = Distance to feed (mm)

This command is used to insert white space in the image without using up image buffer capacity. NOTE: All Y coordinates, including the start coordinate for this command, refer to the distance through the image buffer *before* insertion of white space. This command must be sent for every label requiring it - the effect is lost when the image is cleared.

<Esc> o *yyyy* **SetDuplicateOffset**

yyyy = Offset of duplicate image

This command is used to create a duplicate image without using up image buffer capacity. The offset specified is the distance from the start of the first image to the start of the second image. The second image must lie within the formlength set. To disable duplicate images, set the offset to zero. This command need not be sent for every label - it remains in effect until the offset is set to 0000.

2.12 Cutter Controls

The following commands are only effective in printers fitted with a cutter.

<FS> **EnableCutting**
Enables cutting operations. Enabled by default.

<GS> **DisableCutting**
Disables cutting operations.

<Esc> C *yyyy* **SetCutPosition**
yyyy = Y-Coordinate (mm)
This command sets the distance from the start of the label to the cut position. In label mode cutting then takes place automatically on every label when cutting is enabled. In continuous mode, the cut position must be redefined for each label. Note that repeat printing or copying the print buffer will also copy the cut position.

2.13 Status Reporting

<Esc> e *c* **StatusReport**

c = c	Font cache status
c = e	General status report
c = f	List stored formats
c = l	List stored logos
c = r	List resources loaded
c = t	Show current date/time
c = v	Printer type & software version

Provides various status reports via the serial line, which may be of value to programmers when optimising for throughput or memory usage. Most of the reports are in a verbose format and are not intended to be machine readable. The following have a fixed format :-

Type & version :-
mm..m pp..p **V***v.vvs..*
Printer model mm..m
Program suite pp..p
Version v.vv
Special version s..
e.g.
```
1048 MA62 V2.02C
```
Printer type 1048, program suite MA62, Version 2.01, special version for cutter.

List logos :-

```
nn stored logos
r: wwww hhhh l tttttttttt                     (one line for each logo stored)
```

number of logos nn	
reference r	
width wwww (dots)	
height hhhh (dots)	
location R=ROM, M=memory	
title ttttttttt	

List formats :-

```
nn stored formats
r: ssss l tttttttttt                          (one line for each format stored)
```

number of formats nn	
reference r	
size ssss (bytes)	
location R=ROM, M=memory	
title ttttttttt	

<ENQ> **ShortStatus**

Provides a single character status report via the serial line. In general, numeric codes indicate normal activity, and alphabetics indicate a fault condition. The status values are as follows :-

0	Printer OK
P	Paper out
R	Ribbon out
C	Cutter jammed
M	Media error (head open/rewinder full)

<Esc> s **EnableReporting**

Enables a message reporting the number of characters left in the input buffer after each label is printed. The format is "OK nnnn<CR><LF>", where nnnn is the (hexadecimal) number of bytes remaining unprocessed in the input buffer. This facility is required for backward compatibility with programs which look for this message. Enabled by default at power on.

<Esc> t **DisableReporting**

Disables input buffer reporting.

<Esc> z s **ShortReporting**

Replaces the input buffer report with a single '`*`' character as each label is printed.

2.14 Inline Commands

Most printer commands pass through the input buffer, and any command which is forced to wait for printing to finish, e.g. formatting a label in single buffer mode, will hold up all input data. Also there is no way to cancel logo or graphic data input part way through. Inline commands are intercepted before being placed in the input buffer and can be acted upon almost immediately, irrespective of the state of the input buffer. To maintain compatibility, inline commands must be specifically enabled before they will be trapped.

<Esc> z *n* **SetInline**

n = 0	disabled
n = 1	enabled

<Sub> **EnterInline**
Characters received after <Sub> will be placed in the inline buffer, not the main input buffer. To send a literal <Sub> character, e.g. as part of graphics data, send the character twice.

<Eot> **ExitInline**
Exits back to normal command mode.

e **InlineShortStatus**

a **InlineAbortPrint**

h *nnn* **InlineSetHeat**

m *nnn* **InlineSetSpeed**

These commands are functionally identical to their main input counterparts, except that the effect is immediate.

s **InlineBufferSpace**
Shows input buffer free space as a 4 digit hex value.

x **InlineReset**
Terminates any inline command currently executing, flushes the input buffer and clears the compile buffer, then sends a ShortStatus report. This command can be used to regain control during a graphics download.

2.15 Miscellaneous Commands

<Esc> k *f* **SensorControl**

Flags f
1 = Enable rewinder
2 = Enable label-present sensor

Controls the external rewind motor and the label-present sensor. The flags value is derived by summing the values for the individual controls. E.g. <Esc>k3 enables both the rewinder and the label-present sensor. Note that the label-present sensor must also be physically enabled by the DIP switches, if it is used. Both sensor and rewinder are enabled by default.

<Esc> x **EnableXoff**

Enables XON/XOFF serial data flow control. Enabled by default at power on.

<Esc> y **DisableXoff**

2.16 Error Reporting

The printer checks the validity of all commands received and reports errors where appropriate. The normal causes of errors are :-

- Part of a field is positioned so that it falls outside the print area.
- The command does not make sense, or contains illegal values.
- Logos, formats or variables referred to by an invalid reference.
- Attempt to define a logo or format that already exists.

Not all errors are reported. Some are ignored where nothing reasonable can be done, and some attempt to produce output, even if it is in the wrong place. Text fields will wrap around to the next line if they are too long and back to the top of the page if they are still too long. If the placement coordinates lie outside the print area, the coordinates of the previous field are used.

Where an error message is printed it takes the form :-

```
Barcode error: Page overflow
```

printed at the top right corner of the label. The first part of the message indicates the item which caused the error and the second part indicates the cause of the error. The following causes of error are possible :-

Error message	Possible causes
Invalid data	Command contained an invalid sequence, or referenced a non-existent logo, format or field, or attempted to define a logo or format which already exists.
Illegal command	Attempted to define, execute or delete a format from within another executing format.
Page overflow	An item was placed such that part or all of it fell outside the current print area. Note that available print area depends on the buffer mode in use, stored logos and formats, font cache and input buffer size.
Memory full	There was not enough memory available to store a logo or format.

3 Getting the Best Performance

3.1 Optimising Printer Throughput

The ALFA printer is designed to give maximum throughput, and the following guidelines should be followed when writing programs.

- Use double buffering where possible to allow formatting on the fly.
- Use a font cache when working with scalable fonts.
- Use the base image buffer or copy the previous print buffer when working with complex and mainly invariant labels.
- Do not magnify or rotate logos. If possible, download them in the orientation & size that they will be placed in.
- Use logos rather than graphics if the graphical data will be required more than once, to save download time.
- Trim surrounding white space from logos to minimise the area downloaded, stored & placed.
- Use stored formats and variable fields to reduce the amount of data sent to the printer.
- Avoid unnecessary commands, e.g. repeatedly setting rotation & point size.

3.2 Optimising Print Quality

The ALFA printer has been set up to produce optimum print quality with a broad range of media. The following guidelines will enable you to produce the highest quality output.

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

4 Printable Character Command Mode

This mode is used when the host computer cannot provide 8 bit characters or control codes. This may be the case with some mainframe systems. The method relies on the use of four ’escape’ characters. These are defined in terms of the action taken by the printer :-

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

All incoming characters will be masked to 7 bits, and control characters and DEL will be ignored. The following table shows the conversions for all possible characters.

Printable Character Conversion Table.

000 = #@ NUL	032 =	064 = @	096 = `	128 = %@	160 = !	192 = !@	224 = !`
001 = #A SOH	033 = &!	065 = A	097 = a	129 = %A	161 = !!	193 = !A	225 = !a
002 = #B STX	034 = "	066 = B	098 = b	130 = %B	162 = !"	194 = !B	226 = !b
003 = #C ETX	035 = &#	067 = C	099 = c	131 = %C	163 = !#	195 = !C	227 = !c
004 = #D EOT	036 = $	068 = D	100 = d	132 = %D	164 = !$	196 = !D	228 = !d
005 = #E ENQ	037 = &%	069 = E	101 = e	133 = %E	165 = !%	197 = !E	229 = !e
006 = #F ACK	038 = &&	070 = F	102 = f	134 = %F	166 = !&	198 = !F	230 = !f
007 = #G BEL	039 = '	071 = G	103 = g	135 = %G	167 = !'	199 = !G	231 = !g
008 = #H BS	040 = (	072 = H	104 = h	136 = %H	168 = !(	200 = !H	232 = !h
009 = #I HT	041 =)	073 = I	105 = i	137 = %I	169 = !)	201 = !I	233 = !i
010 = #J LF	042 = *	074 = J	106 = j	138 = %J	170 = !*	202 = !J	234 = !j
011 = #K VT	043 = +	075 = K	107 = k	139 = %K	171 = !+	203 = !K	235 = !k
012 = #L FF	044 = ,	076 = L	108 = l	140 = %L	172 = !,	204 = !L	236 = !l
013 = #M CR	045 = -	077 = M	109 = m	141 = %M	173 = !-	205 = !M	237 = !m
014 = #N SO	046 = .	078 = N	110 = n	142 = %N	174 = !.	206 = !N	238 = !n
015 = #O SI	047 = /	079 = O	111 = o	143 = %O	175 = !/	207 = !O	239 = !o
016 = #P DLE	048 = 0	080 = P	112 = p	144 = %P	176 = !0	208 = !P	240 = !p
017 = #Q DC1	049 = 1	081 = Q	113 = q	145 = %Q	177 = !1	209 = !Q	241 = !q
018 = #R DC2	050 = 2	082 = R	114 = r	146 = %R	178 = !2	210 = !R	242 = !r
019 = #S DC3	051 = 3	083 = S	115 = s	147 = %S	179 = !3	211 = !S	243 = !s
020 = #T DC4	052 = 4	084 = T	116 = t	148 = %T	180 = !4	212 = !T	244 = !t
021 = #U NAK	053 = 5	085 = U	117 = u	149 = %U	181 = !5	213 = !U	245 = !u
022 = #V SYN	054 = 6	086 = V	118 = v	150 = %V	182 = !6	214 = !V	246 = !v
023 = #W ETB	055 = 7	087 = W	119 = w	151 = %W	183 = !7	215 = !W	247 = !w
024 = #X CAN	056 = 8	088 = X	120 = x	152 = %X	184 = !8	216 = !X	248 = !x
025 = #Y EM	057 = 9	089 = Y	121 = y	153 = %Y	185 = !9	217 = !Y	249 = !y
026 = #Z SUB	058 = :	090 = Z	122 = z	154 = %Z	186 = !:	218 = !Z	250 = !z
027 = #[ESC	059 = ;	091 = [	123 = {	155 = %[	187 = !;	219 = ![	251 = !{
028 = #\ FS	060 = <	092 = \	124 = |	156 = %\	188 = !<	220 = !\	252 = !|
029 = #] GS	061 = =	093 =]	125 = }	157 = %]	189 = !=	221 = !]	253 = !}
030 = #^ RS	062 = >	094 = ^	126 = ~	158 = %^	190 = !>	222 = !^	254 = !~
031 = #_ US	063 = ?	095 = _	127 = &d	159 = %_	191 = !?	223 = !_	255 = &D

Note that is used to represent the 'space' character.

E.g. To send "<Esc>T00050005Hello<Eot>" in printable character mode, send :-
"#[T00050005Hello#D"

5 Character Set (Code Page) Tables

W - Windows ANSI

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

E - English (CP437)

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

M - Multilingual (CP850)

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

S - Slavic (CP852)

	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	32	48 0	64 @	80 P	96 `	112 p	128 Ç	144 É	160 á	176	192	208 ð	224 Ó	240 -
1	33 !	49 1	65 A	81 Q	97 a	113 q	129 ü	145 Ĺ	161 í	177	193	209 Đ	225 ß	241
2	34 "	50 2	66 B	82 R	98 b	114 r	130 é	146 ĺ	162 ó	178	194	210 Ď	226 Ô	242
3	35 #	51 3	67 C	83 S	99 c	115 s	131 â	147 ô	163 ú	179	195	211 Ë	227 Ń	243
4	36 $	52 4	68 D	84 T	100 d	116 t	132 ä	148 ö	164 Ą	180	196	212 ď	228 ń	244
5	37 %	53 5	69 E	85 U	101 e	117 u	133 ů	149 Ľ	165 ą	181 Á	197	213 Ň	229 ň	245 §
6	38 &	54 6	70 F	86 V	102 f	118 v	134 ć	150 ľ	166 Ž	182 Â	198 Ă	214 Í	230 Š	246 ÷
7	39 '	55 7	71 G	87 W	103 g	119 w	135 ç	151 Ś	167 ž	183 Ě	199 ă	215 Î	231 š	247 ¸
8	40 (	56 8	72 H	88 X	104 h	120 x	136 ł	152 ś	168 Ę	184 Ş	200	216 ě	232 R̀	248 °
9	41)	57 9	73 I	89 Y	105 i	121 y	137 ë	153 Ö	169 ę	185	201	217	233 Ú	249 ¨
A	42 *	58 :	74 J	90 Z	106 j	122 z	138 Ő	154 Ü	170	186	202	218	234 r̀	250
B	43 +	59 ;	75 K	91 [	107 k	123 {	139 ő	155 Ť	171 ź	187	203	219	235 Ű	251 ű
C	44 ,	60 <	76 L	92 \	108 l	124 |	140 î	156 ť	172 Č	188	204	220	236 ý	252 Ř
D	45 -	61 =	77 M	93]	109 m	125 }	141 Ź	157 Ł	173 ş	189 Ż	205	221 Ţ	237 Ý	253 ř
E	46 .	62 >	78 N	94 ^	110 n	126 ~	142 Ä	158 ×	174 «	190 ż	206	222 Ů	238 ţ	254
F	47 /	63 ?	79 O	95 _	111 o	127 □	143 Ć	159 č	175 »	191	207 ¤	223	239 ´	255

P - Portugese (CP860)

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

C/F - Canadian/French (CP863)

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

N - Nordic (CP865)

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

6 Programming QUICK Reference

6.1 Summary of Control Codes

Some control codes are used as single character commands. Others form part of an escape sequence. The latter use is shown below in brackets.

NUL	^@	0	Ignored
SOH	^A	1	(Introduces variable fields in text & barcodes)
STX	^B	2	EnableBarcodeText
ETX	^C	3	DisableBarcodeText
EOT	^D	4	(Data terminator)
ENQ	^E	5	ShortStatus (serial)
ACK	^F	6	Reserved
BEL	^G	7	Reserved
BS	^H	8	Reserved
HT	^I	9	(Tab in text command)
LF	^J	10	(LineFeed in text command)
VT	^K	11	(Terminates format definition)
FF	^L	12	FeedPrint
CR	^M	13	(CarriageReturn in text command)
SO	^N	14	CopyCompileToBase
SI	^O	15	Reserved
DLE	^P	16	CopyBaseToCompile
DC1	^Q	17	(Xon flow control character)
DC2	^R	18	SetContinuous
DC3	^S	19	(Xoff flow control character)
DC4	^T	20	SetLabelMode
NAK	^U	21	Reserved
SYN	^V	22	StatusReport (Logo directory)
ETB	^W	23	StatusReport (Format directory)
CAN	^X	24	ClearAllBuffers
EM	^Y	25	ClearCompileBuffer
SUB	^Z	26	(Starts inline command)
ESC	**^[**	**27**	**(Starts escape sequence)**
FS	^\	28	EnableCutting
GS	^]	29	DisableCutting
RS	^^	30	SetDoubleBuffer
US	^_	31	SetSingleBuffer

6.2 Summary of Escape Sequences

<Esc> A llll	SetFormlength
<Esc> B *xxxx yyyy* 1 *hh d(12)*	PlaceITF14
<Esc> B *xxxx yyyy* 2 *hh [a] d(12) [d(2/5)]*	PlaceEAN13
<Esc> B *xxxx yyyy* 3 *hh d(7)*	PlaceEAN8
<Esc> B *xxxx yyyy* 4 *hh d(1-50)* q	PlaceInt2of5
<Esc> B *xxxx yyyy* 5 *hh d(1-50)* q	PlaceCode39C
<Esc> B *xxxx yyyy* 6 *hh d(1-50)* q	PlaceCode39
<Esc> B *xxxx yyyy* 7 *hh [c] d(1-50)* q	PlaceCodabar
<Esc> B *xxxx yyyy* 8 *hh d(1-50)* <Eot>	PlaceEAN128
<Esc> B *xxxx yyyy* 9 *hh nn d(1-50)* <Eot>	PlaceCode128
<Esc> B *xxxx yyyy* A *hh d(11)*	PlaceUPCA
<Esc> B *xxxx yyyy* B *hh d(11)*	PlaceUPCE
<Esc> C *yyyy*	SetCutPosition
<Esc> D *n t(10) ddd...dd* <VT>	DefineFormat
<Esc> E *n*	EraseFormat
<Esc> F *f c*	SelectFont
<Esc> G *xxxx yyyy wwww hhhh ddd...dd*	PlaceGraphics
<Esc> I *xxxx yyyy wwww hhhh c*	BlockFill
<Esc> J *n*	LoadFormat
<Esc> K *n t(10) wwww hhhh ddd...dd*	DefineLogo
<Esc> L *xxxx yyyy n*	PlaceLogo
<Esc> M *vv hh*	SetMagnification
<Esc> N *t m w n*	SetBarcodeModule
<Esc> O *yyyy llll*	SetForwardFeed
<Esc> P *llll*	SetPageOffset
<Esc> R *rrr*	RepeatPrint
<Esc> S *ss*	SetSpeed
<Esc> T *xxxx yyyy ddd...dd* <Eot>	PlaceText
<Esc> U *www*	SetTabWidth
<Esc> V *r*	SetRotation
<Esc> W *l c*	SetCharGaps
<Esc> X *n*	DeleteLogo
<Esc> Y *ff vvv hhh i k*	SetScalableFont
<Esc> Z *M2PDO*	SetUnits
<Esc> Z *TB*	SetBaseline
<Esc> a	AbortPrint
<Esc> b S *nn*	ResizeInputBuffer
<Esc> c a *nnnn*	AllocateFontCache
<Esc> c d	DisableFontCache
<Esc> c e	EnableFontCache
<Esc> c f	FlushFontCache
<Esc> d *yyyy*	SetPrintlineToCutter
<Esc> e *ceflrtv*	StatusReport
<Esc> f d *r ww +ii /uu* <Eot>	DefineVariableField
<Esc> f l	ListFields
<Esc> f s DDMMYYhhmmss	SetRealTimeClock
<Esc> f t	UpdateTimeField
<Esc> f u	UpdateFields
<Esc> f x	ClearFields
<Esc> h *nnn*	SetHeat
<Esc> i *n*	SetBaseImage
<Esc> j *n rrrr*	RepeatLoadFormat

<Esc> k *c*	SensorControl
<Esc> o *yyyy*	SetDuplicateOffset
<Esc> r *rrrr*	RepeatPrint4
<Esc> s	EnableReporting
<Esc> t	DisableReporting
<Esc> v *r ddd...dd* <Eot>	SetVariableField
<Esc> x	EnableXoff
<Esc> y	DisableXoff
<Esc> z *n*	SetInline
<Esc> z s	ShortReporting
<Esc> . *xxxx yyyy ss d(17)*	PlaceDotCode
<Esc> : *xxxx yyyy* L *ss ff c d(1-528)*	PlaceLEBCode
<Esc> : *xxxx yyyy* U *tt ss d(5-20)*	PlaceUSD5Code
<Soh> *r [<|>|= n..] [.|, n..]* <Eot>	GetVariableField
<Soh> t *[+d..] [#DMYhmsAa]* <Eot>	GetTimeField

6.3 Summary of DIP switch settings

6.3.1 Data Interface Switches (Left hand bank)

Switch	ON	OFF
1	Cutter fitted	No cutter
2	Command set by printable characters	Command set by control codes
3	Direct printing	Transfer printing
4	Odd parity	Even parity
5	Parity off	Parity on
6	7 data bits	8 data bits
7	-Baud rate-	See below
8	-Baud rate-	See below
9	Label mode	Continuous mode
10	Not used	

Baud Rate	Switch **7**	Switch **8**	Notes
1200	On	On	If no parity, set SW4 Off
2400	On	Off	
4800	Off	On	
9600	Off	Off	
19200	On	On	Set SW4 On, SW5 On
38400	On	Off	Set SW4 On, SW5 On

6.3.2 Media Sensor Switches (Right hand bank)

Switch	ON	OFF
1	Transmissive sensor	Reflective Sensor
2	Present sensor on	Present sensor off

7 Programming Example

This example program is written in BASIC for a 1048 printer.

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

```
soh$=chr$(1)      'introduces variable fields
stx$=chr$(2)      'enable barcode text
etx$=chr$(3)      'disable barcode text
eot$=chr$(4)
lf$ =chr$(10)
vt$ =chr$(11)
ff$ =chr$(12)
so$ =chr$(14)     'copy compile to const
dle$=chr$(16)     'copy print/const to compile
e$  =chr$(27)
gs$ =chr$(29)
rs$ =chr$(30)     'double buffer
us$ =chr$(31)     'single buffer

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

' Set up to use coordinates and distances in millimetres, single buffer.
print#1,e$;"ZM";us$;
```

Print a testcard label with text, barcodes & grey block.

```
print#1,e$;"F6W";e$;"V2";                             ' Large text, rotation 2
print#1,e$;"T01000000ABCDEFGHIJK";lf$;                ' Place 3 lines of text ..
print#1,"abcdefghijklmn";lf$;                         ' starting at (100, 0)
print#1,"1234567890";eot$;
print#1,e$;"B00400026615ABCDq";                       ' Code-39 barcode "ABCD"
                                                      ' at (40,26), height 15 mm
                                                      '
print#1,e$;"I00450000003300050G";                     ' Grey filled box
print#1,e$;"V1";e$;"B0002000200501234567890";         ' EAN-13 barcode
print#1,e$;"F3W";e$;"T00020026";date$;lf$;time$;eot$; ' Date & time
print#1,e$;"F1W";e$;"T00020043abcdefghijklm";lf$;     ' Small text
print#1,"abcdefghijklm";lf$;
print#1,"abcdefghijklm";eot$;

print#1,ff$;                                          ' Formfeed to print label
```

EAN-128 barcode demonstration.
This example is taken from the ANA manual and describes a quantity of 21 and batch number 123456. Note the use of GS to terminate the quantity field (which is of variable length), when concatenated with a batch number.

```
print#1,stx$;e$;"V1";                ’ Set rotation 1, add human-readable text
print#1,e$;"N8300";                  ’ Set barcode magnification 3
’ Place barcode type 8 (EAN-128) at (20, 10), and use the default height.
’ The barcode data which follows uses the EAN-128 data structure.
print#1,e$;"B00200010800(30)21";gs$;"(10)123456";eot$;
print#1,ff$;                         ’ Print label
```

Logo demonstration.
Normally a logo will be a graphic image, but for simplicity, this demo defines a simple chequerboard logo and places it on the page.

```
’ Download logo number 0.
print#1,e$;"X0";e$;"X0";                   ’ Delete any existing logo 0 first
print#1,e$;"K0LogoName  00640064";         ’ Define 64 x 64 pixel logo as Logo 0

’ This loop outputs the logo data for a chequerboard.
for i = 1 to 16
    for j = 1 to 16
      print#1,chr$(204);      ’pattern CCh
    next j
    for j = 1 to 16
      print#1,chr$(51);       ’pattern 33h
    next j
next i

’ The logo is now stored in memory and can be applied anywhere on the label
print#1,e$;"M0505";                        ’ Magnification 5x5
print#1,e$;"L000500050";                   ’ Place logo 0 at (5, 5)
print#1,ff$;                               ’ Print label

print#1,e$;"X0";e$;"X0";e$;"M0101";        ’ Delete logo when no longer required
```

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

```
' Set up the invariant part of the label
print#1,e$;"i1";                         ' Enable base image buffer.
print#1,e$;"F0W";e$;"Y0204004000";       ' Speedo font 02 @ 40 point
print#1,e$;"T00000020ABCDEFGHIJKLMNOPQ";eot$;
print#1,so$;                             ' Copy to base image buffer

' Set up a loop to print varying information.
print#1,e$;"F5W";                        ' Bitmapped font 5
for i = 3 to 1 step -1
    print#1,dle$;                        ' Copy base image to compile buffer
    print#1,e$;"T00000000";i;"Green bottles";eot$; ' Add new text
    print#1,ff$;                         ' Print label
next i
print#1,e$;"i0";                         ' Disable base image buffer
```

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

```
' Set up a string containing DDMMYYhhmmss and set printer date & time
ldate$=date$
ltime$=time$
setdate$=mid$(ldate$,4,2)+left$(ldate$,2)+right$(ldate$,2)
setdate$=setdate$+left$(ltime$,2)+mid$(ltime$,4,2)+right$(ltime$,2)
print#1,e$;"fs";setdate$;

' Define field 0 as an incrementing field, max width 10, & set initial value.
print#1,e$;"fd010+01";eot$;e$;"v0123450";eot$;

print#1,rs$;                             ' Set double buffer mode for speed
print#1,e$;"N4252";                      ' Set Int2of5 module sizes
' Define format 0 (delete any existing version first)
print#1,e$;"E0";e$;"E0";e$;"D0FormatName";
    print#1,e$;"ft";                     ' Update date-time field
    print#1,e$;"F5W";
    print#1,e$;"T00100006Serial Number: ";soh$;"0";eot$;eot$;
    print#1,e$;"F4W";
    print#1,e$;"T00100012Manufactured: ";soh$;"t#D-#M-#Y #h:m:s a";eot$;"m";eot$;
    print#1,e$;"T00100018Display until: ";soh$;"t+30D-M-Y";eot$;eot$;
    print#1,e$;"T00100024Best before: ";soh$;"t+60D-M-Y";eot$;eot$;
    print#1,e$;"B00100030420";soh$;"0=10";eot$;"q";
    print#1,ff$;                         ' Print the label
    print#1,e$;"fu";                     ' Update variable fields
print#1,vt$;                             ' End of format definition

print#1,e$;"j00004";                     ' Replay format 4 times
```

----- *END OF MANUAL* -----